The Completeness Problem of Ordered Relational Databases

by

Wei Jiang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Support of order in query processing is a crucial component in relational database systems, not only because the output of a query is often required to be sorted in a specific order, but also because employing order properties can significantly reduce the query execution cost. Therefore, finding an effective approach to answer queries over ordered data is important to the efficiency of query processing in relational databases.

In this dissertation, an ordered relational database model is proposed, which captures both data tuples of relations and tuple ordering in relations. Based on this conceptual model, ordered relational queries are formally defined in a two-sorted ($\boldsymbol{x}$-sort for data attributes and $\boldsymbol{t}$-sort for tuple identifiers) first-order calculus $\boldsymbol{FO}_{\preceq}$, which serves as a yardstick to evaluate expressive power of other ordered query representations.

The primary purpose of this dissertation is to investigate the expressive power of different ordered query representations. Particularly, the completeness problem of ordered relational algebras is studied with respect to the first-order calculus $\boldsymbol{FO}_{\preceq}$: does there exist an ordered algebra such that any $\boldsymbol{FO}_{\preceq}$ ordered relational query can be expressed by a finite sequence of ordered operations? The significance of studying the completeness problem of ordered relational algebras is in that the completeness of ordered relational algebras leads to the possibility of implementing a finite set of ordered operators to express all first-order expressible ordered queries in relational databases.

The dissertation then focuses on the completeness problem of ordered conjunctive queries. This investigation is performed in an incremental manner: first, the ordered conjunctive queries with data-decided (or $\boldsymbol{x}$-decided) order, $\boldsymbol{CQ}^{X}_{\preceq}$, is considered; then, the ordered conjunctive queries with $\boldsymbol{t}$-decided order, $\boldsymbol{CQ}^{T}_{\preceq}$, is studied; finally, the completeness problem for the general ordered conjunctive queries, $\boldsymbol{CQ}_{\preceq}$, is explored. The completeness theorem of ordered algebras is proven for all three classes of ordered conjunctive queries.

Although this ordered relational database model is only conceptual, and ordered operators are not implemented in this dissertation, we do prove that a complete set of ordered operators exists to retrieve all first order expressible ordered queries in the three classes of ordered conjunctive queries. This research sheds light on the possibility of implementing a complete set of ordered operators in relational databases to solve the performance problem of order-relevant queries.

Acknowledgements

First of all, I would like to express my deep gratitude to my supervisor, Dr. David Toman, for his guidance and support during my PhD program. His keen insight and profound knowledge of database theories have been the constant source for my research. This dissertation would have not been possible without his help.

I also want to gratefully acknowledge constructive suggestions from my committee members, Grant Weddell, Tamer Özsu, Leopoldo Bertossi, and Paul Ward. Particularly, I want to express my appreciation to Dr. Weddell, who not only provided insightful suggestions to my work, but also served as my substitute supervisor for several terms. I would like to extend a special thanks to Dr. Özsu for his critical advice on progress schedule of this dissertation.

I would like to thank my friends and colleagues in the Database Group for interesting discussions and warm friendship over the years. Thanks are also due to friends and staff at the School of Computer Science for supporting me during my study.

Finally, and most of all, I am deeply indebted to my parents for their endless love and continued support. My husband Weizhen is always there beside me whenever I need him. I thank him for his love, understanding, and support throughout this journey. This dissertation is dedicated to my dear daughter Audrey and son Andrew, who are the most precious gifts of my life.

Contents

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**

1.1 Motivation 1

1.2 Ordered Relational Databases 7

1.3 Related Work 8

1.3.1 Order Optimization 8

1.3.2 Ordered Representations of Relations 10

1.3.3 Query Rewriting 11

1.3.4 Integrity Constraints Involving Order 13

1.3.5 Partial Order Databases 13

1.3.6 Temporal Databases 14

1.3.7 Preference and Skyline Queries 15

1.3.8 XML Queries 16

1.3.9 Top-k Queries 17

1.4 Organization of Dissertation 18

2 The Ordered Relational Model **19**

2.1 Ordered Relational Databases 19

2.1.1 Ordered Relational Databases 20

2.1.2 Ordered Relational Databases vs. Temporal Relational Databases 25

2.2 Ordered Relational Queries 31
2.2.1 $\boldsymbol{FO}_{\preceq}$ Ordered Relational Queries 31
2.2.2 $\boldsymbol{FO}_{\preceq}$ Ordered Relational Queries vs. 2$\boldsymbol{FOL}$ Temporal Relational Queries 35
2.3 Ordered Relational Algebras 42
2.3.1 Ordered Relational Algebras 42
2.3.2 The Completeness Problem of Ordered Relational Algebras . 45

3 Ordered Conjunctive Queries with Data-Decided Order $\boldsymbol{CQ}^X_{\preceq}$ 48
3.1 Ordered Conjunctive Queries $\boldsymbol{CQ}^X_{\preceq}$ 49
3.2 An Ordered Conjunctive Algebra $\boldsymbol{CA}^X_{\preceq}$ 52
3.2.1 Overview of Ordered Algebra $\boldsymbol{CA}^X_{\preceq}$ 52
3.2.2 Order-preserving Selection 53
3.2.3 Order-preserving Projection 55
3.2.4 Left Nested-loop Products 57
3.2.5 Order Concatenation 60
3.2.6 Order Reduction 62
3.2.7 Order Identity 64
3.2.8 Order Reverse 65
3.3 Transformation Rules 66
3.4 Completeness of $\boldsymbol{CA}^X_{\preceq}$ 77

4 Ordered Conjunctive Queries with *t*-Decided Order $\boldsymbol{CQ}^T_{\preceq}$ 96
4.1 Ordered Conjunctive Queries $\boldsymbol{CQ}^T_{\preceq}$ 96
4.2 An Ordered Conjunctive Algebra $\boldsymbol{CA}^T_{\preceq}$ 99
4.2.1 Overview of Ordered Algebra $\boldsymbol{CA}^T_{\preceq}$ 99
4.2.2 Order Intersection 100
4.2.3 Top Operation 102
4.2.4 Derived Operations 104
4.3 Transformation Rules 105
4.4 Completeness of $\boldsymbol{CA}^T_{\preceq}$ 109

5 Ordered Conjunctive Queries $\boldsymbol{CQ}_{\preceq}$ 128

5.1 Ordered Conjunctive Queries $\boldsymbol{CQ}_{\preceq}$ 128

5.2 An Ordered Conjunctive Algebra $\boldsymbol{CA}_{\preceq}$ 132

5.2.1 Overview of Ordered Algebra $\boldsymbol{CA}_{\preceq}$ 132

5.2.2 Until Operation 133

5.2.3 Derived Operations 136

5.3 Transformation Rules 137

5.4 Completeness of $\boldsymbol{CA}_{\preceq}$ 138

6 Conclusion and Future Work 157

6.1 Summary of Results 157

6.2 Future Work 160

References 161

List of Tables

2.1 The data relation EMP 23

2.2 The order relation $\preceq_{\texttt{EMP}}$ 23

2.3 The ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$ 24

2.4 An example of temporal relations 28

2.5 One ordered relation corresponding to the temporal relation $Class$ 29

2.6 Another ordered relation corresponding to the temporal relation $Class$ 29

2.7 The ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$ 33

2.8 The ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$ 34

5.1 The construction of E_β by induction 146

List of Figures

1.1 An execution plan for the example query 3
1.2 XML trees labelled with intervals 4
1.3 DB2 plan for *roots* 5
1.4 A more efficient algorithm for *roots* 6
2.1 Reduction of a temporal query to an ordered query 36
5.1 An illustration of the conjunct β 146

Chapter 1

Introduction

This dissertation investigates supporting order in query processing in relational database systems. In particular, we propose an ordered relational database model which is independent of data types and specific domains, and we study the expressive power of different query representations over this ordered database structure.

This chapter introduces work that will be addressed in this dissertation. Section 1.1 provides the motivations of the proposed research in order support in relational database systems. In Section 1.2 we explain why the physical order is chosen as the order representation in the proposed investigation, and propose a novel ordered relational database model. In Section 1.3 a brief survey is conducted on the related work in order support in database systems. Section 1.4 outlines the organization of the remainder of the dissertation.

1.1 Motivation

Order is a critical characteristic of data and has been studied for decades in database query processing areas. Inherently, ordering exists among data in all database applications. For example, in scientific databases, each experimental observation might be ordered by observation date. In bank databases, banking transactions might be sorted by transaction time and account number. In stock databases, stock price history might be recorded according to date. The interest in studying data order in databases naturally stems from these database applications.

One of the reasons to consider order in query processing is that the query is required to output the result in a specific order. A financial analyst, for example,

might want a list of stocks in descending order of trade volume for a given day. A scientist might be interested in generating the experimental data in a specific order so that the experimental data could be the input to data analysis software. Efficient order support techniques will directly benefit these queries with ordered output.

Another key reason to consider order in query processing comes from query optimization. Some sort-based operations, such as join and duplicate elimination, can be implemented more efficiently if the query processor takes advantage of the ordering of data. It is well-known that sorting is one of most expensive operations in query processing. If the query processor keeps track of the order in the intermediate result, sort operations can be avoided when the order required for sort-based operations matches the existing order. Even when a sort cannot be avoided, a more efficient plan can be formed by deciding when and how to sort. Therefore, taking advantage of order properties during query processing results in more efficient query execution plans.

Recently, exploiting the order of data has gained new interest and prominence in the database community. The rapid growth of the Internet industry is one of the driving forces behind this trend. Fast and selective results are more desirable for web queries, rather than more accurate and complete results desirable for traditional database queries. One of the applications for order treatment in Internet information retrieval is exemplified by top-k queries, in which the user receives the first k relevant answers to his query in an order that satisfies his or her specific criteria. From the users' perspective, top-k queries enable users to acquire the most important query answers promptly. Alternatively, search engines can retrieve the most relevant answers more efficiently and effectively.

The following example illustrates how crucial the order is during query processing in relational databases and what the existing database systems lack in supporting order.

Example 1.1 We consider a database containing one relation `EMP`, which stores information about a company's employees: Employee Id, Name, Age, and Salary. This relation has a relation schema `EMP` (`EmpId`, `Name`, `Age`, `Salary`, `Department`) with `EmpId` as the primary key. Suppose that an end user desires to retrieve all employees in the department "Sales" in a descending order of salary. Under the circumstances that no index was built on this relation, the user could answer this question by generating an SQL query as follows:

```
SELECT Name
FROM EMP
WHERE Department = 'Sales'
ORDER BY Salary DESC
```

A typical execution plan for this query in relational database processors is shown in Figure 1.1.

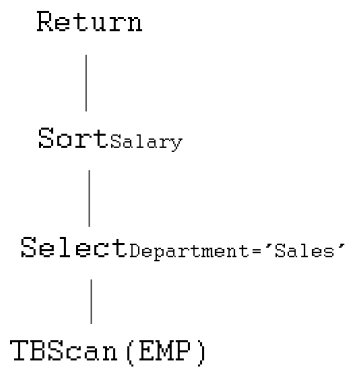

Figure 1.1: An execution plan for the example query

In Figure 1.1, the query processor first scans the relation `EMP` to take tuples into the memory. Then, a selection is applied on all tuples based on a data predicate `Department` = "Sales". Next, the selected tuples are sorted in a descending order of `Salary`. Finally, "Return" returns the selected tuples in the desired order.

This query plan typically takes $O(n \log n)$ to obtain the result, which comes mainly from the sorting operation. However, if the tuples in `EMP` are already in a descending order of `Salary`, the sorting operation can be omitted. The query plan then has an execution cost of $O(n)$. The execution cost changes from $O(n \log n)$ to $O(n)$ if the query processor can detect whether the order of a relation already matches the desired order in the plan.

One solution to using orders in query processing is to record tuples with their order and to provide algebraic operators in an ordered fashion. In Example 1.1, assume that tuples in `EMP` are sorted in a descending order of `Salary`. If the selection operator is order-preserving, which means the ouput of selection is kept in the same order as in the input `EMP`, we know that the selected tuples are in the descending order of . Therefore, the sorting operation can be avoided, and the execution cost changes from $O(n \log n)$ to $O(n)$ in this case.

This example indicates that order support during query processing and optimization is evidently in demand with regard to the efficiency of query processing. However, most conventional relational databases have set or multi-set based data models, which make it difficult to manipulate orders during query processing. Currently, commercial database systems do support orders in their query processing; however, their order support is not in the core of the query engine at all, but is some add-on features on top of traditional query optimization. For example, most existing SQL query optimizers in commercial databases provide a "WINDOW" function to use the data ordering. Unfortunately, this order-relevant feature, although expressive, results in complex formulations for even simple queries [57]. Furthermore, the complex formulation is difficult to optimize in later query processing. However, almost every database query execution nowadays contains sorting and other order-relevant operations. The lack of systematic order support built into the query processor leads to the inefficiency of processing order-relevant queries in relational databases, as illustrated in the following example.

Example 1.2 The Extensible Makeup Language (XML), as a standard format of exchanging data on the Internet, composes documents as ordered and labelled trees. The XML documents can be translated into relations, for example, using interval encoding [33]: an XML document is modelled as an ordered forest of rooted, node-labelled, and ordered trees. In an XML forest, each node is labelled with its left and right endpoints (l, r). The interval of a parent must contain the intervals of all its children, that is, the left endpoints of children must be greater than those of the parent, and the right endpoints of children must be less than those of the parent. Examples of XML trees labelled with intervals are illustrated in Figure 1.2.

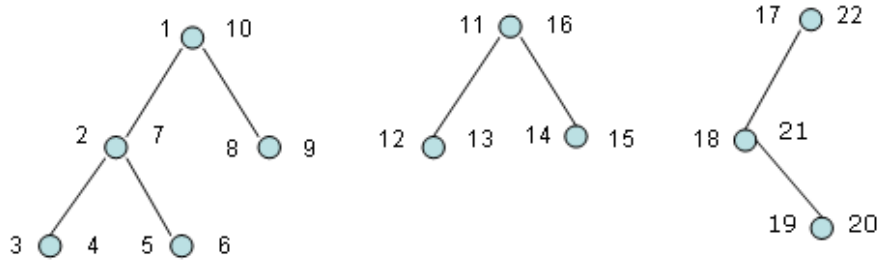

Figure 1.2: XML trees labelled with intervals

In an XML forest, the nodes without parents are called *roots*. Typically, we use roots as starting points to retrieve and update the nodes in the forest. The template for the *roots operator* is defined as follows:

```
CREATE VIEW roots(T) AS
SELECT u.s AS s, u.l AS l, u.r AS r
FROM T u
WHERE NOT EXISTS(
      SELECT *
      FROM T v
      WHERE v.l < u.l AND u.r < v.r)
```

Conventional relational optimizers, such as DB2, generate the plan in Figure 1.3. In this plan, the joins right after table scans are outer-joins because they are performed with inequality predicates; the intermediate results after the outer-joins are joined together with a merge-join. Therefore, the total cost of this plan is $O(n^2 log n^2)$, which is not efficient.

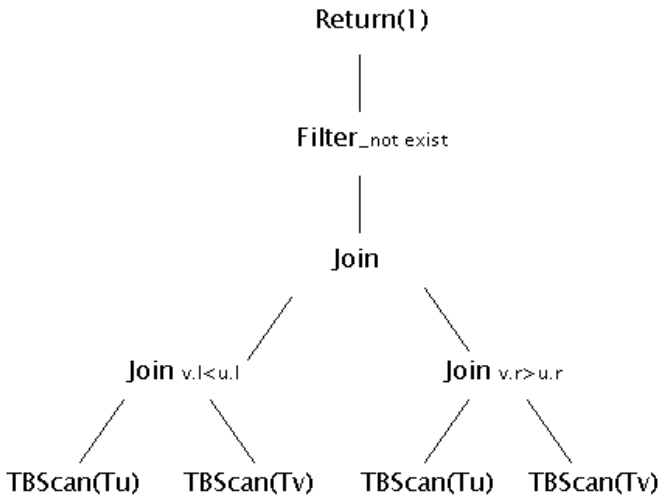

Figure 1.3: DB2 plan for *roots*

More efficient methods exist to solve this problem. One possible approach is to scan the table only once and make use of the order property that the interval of a parent must contain the intervals of all its children. We first scan the table and sort it by left endpoints l, and now we know that the first tuple must be one of the roots, and all child tuples must follow after their parent. Then, we iterate over the table and compare right endpoints of the following tuples with the right endpoint of this root, until we find a tuple with the right endpoint greater than the right

endpoint of this root, which is another root. Repeating the procedure, we will find all roots in a given XML forest. This algorithm is depicted in Figure 1.4:

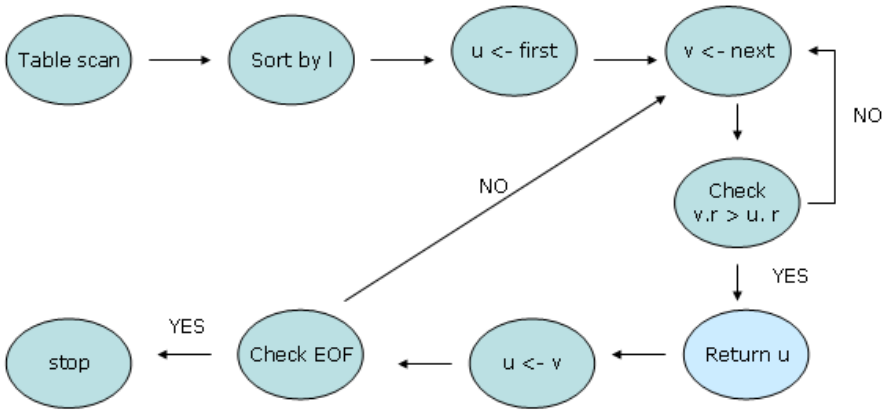

Figure 1.4: A more efficient algorithm for *roots*

With this algorithm, sorting left endpoints takes $O(n \log n)$ and comparing right endpoints takes $O(n)$. Thus, this algorithm takes only $O(n \log n)$ to compute all roots of an XML forest. Furthermore, if the relation is already ordered by left endpoints, the time cost is linear, $O(n)$. Considering the large scale of current database applications, and considering how frequently queries of roots occur in XML databases, the query performance with this algorithm is improved considerably over the performance of the previous query plan.

However, current databases do not support such an algorithm because no operators can express it in query engines. A natural solution to this situation is to add an operator of roots to the current database systems. Thus, the cost of query processing in XML databases will be significantly reduced by an efficient implementation of the operator of roots.

This example demonstrates that those queries which rely heavily on order properties of databases can be processed more efficiently if a suitable set of operators are supplied in XML database systems. Similarly, a set of ordered operators might exist in general relational database systems to efficiently process order-relevant queries. In general, we are interested in finding a complete set of ordered operators, which are independent of database applications and are able to express all potential orders occurring during query processing in relational databases.

Furthermore, this example motivates the study of the completeness problem in

ordered relational databases: whether there exists an ordered relational algebra, equipped with a complete set of ordered operators, to answer all first-order expressible ordered queries. To do this, an ordered database model must be proposed, which is independent of data types and specific domains, and contains a collection of general ordered operators to manipulate the ordered data structure. By considering order properties from the beginning of query processing and optimization, the query processing cost should be reduced significantly for order-relevant queries.

1.2 Ordered Relational Databases

In this dissertation, we provide an ordered relational database model in which relational queries are processed in an ordered fashion. An ordered database is a collection of ordered relations which are composed of data tuples and the ordering among data tuples. An ordered query specifies not only resulting data tuples, but also the ordering of those tuples based on its input. To capture data ordering in this ordered relational database model, we make use of the "physical order" of data, which is the order the iterator retrieve data from a relation. The reasons for choosing the physical order as the order representation in this dissertation are that:

- The physical order is independent of data domains. We can employ the physical order with any data types in any database applications.
- The physical order is more general than the attribute orders. A physical order can always be generated by iterators in a relation; however, a relation may not have any attribute order in it. Furthermore, the physical order can be affiliated with attribute orders by integrity constraints, such as order dependencies [41] [42].

To represent the physical order of a relation, a virtual tuple identifier is attached to each data tuple in the relation. Then, the order of the relation is represented by a binary relation, which indicates orders between all pairs of tuple identifiers and hence forms a linear total order over the relation. This binary relation is called an order relation. An ordered relation contains a regular relation and its order relation. The operators in this ordered database model accept ordered relations as their input and generate an ordered relation as their output.

Based on this ordered database model, ordered relational queries are formally defined in the first-order predicate calculus. The expressive powers of other ordered query representation languages will be studied, such as ordered relational algebras, with respect to the first-order logics. We are particularly interested in the completeness problem of ordered relational algebras: does there exist a complete algebra for all first-order expressible ordered relational queries?

The completeness problem is essential in ordered relational databases because the incompleteness of ordered relational algebras causes serious problems when implementing ordered operators in relational databases. Typical implementations in the conventional databases are based on essentially the equivalence of relational algebra and first-order calculus [5] [40] [27]. Analogously, the completeness of ordered relational algebras decides whether ordered relational algebras possibly could be implemented in which all first-order expressible ordered queries can be answered.

1.3 Related Work

Considerable effort has been made for supporting order in query processing and optimization since the 1970's. In this section, an overview is provided on the research that has been done in the relevant areas, such as query optimization, ordered data representations, and integrity constraints, etc.

1.3.1 Order Optimization

The first study of ordering in query processing was closely interwoven with query optimization. Selinger et al. introduced the "interesting order" in their seminal work of access path selection in System R [69]. *Interesting order* is an order that is generated by an operator, such as the sort operator, or demanded by an operator, such as the ORDER BY or GROUP BY, or is beneficial to exploit and use for operators, such as the merge-join operator. When a query is issued, the optimizer examines all possible execution plans. For each execution plan, both the predicted cost and the relation order are computed. The system chooses the least expensive execution plan among the plans with interesting orders and the plans that produce unordered results. If there is no interesting order in the query, the system chooses the least expensive unordered plan; if there is an interesting order, then it compares

the lease expensive plan with interesting order and the least expensive unordered plan plus the cost of sorting in optimization.

In another relational optimizer, *Starburst* [47], a SQL query is represented using the Query Graph Model (QGM). In QGM, a query is represented as a series of algebraic operations on relations, including SELECT, UNION, INTERSECTION, GROUP BY, etc. Query optimization consists of two phases of processing: query rewrite and plan optimization. In the phase of query rewrite [66], transformation rules are used to rewrite a QGM query into equivalent queries. In the phase of plan optimization [60], the optimizer selects the least costly execution plan of the QGM query among a variety of execution plans. A high-level algebraic operation is realized by a derivation of physical plans, which are associated with their estimated costs and physical properties, such as ordering. Order properties are propagated through physical operations in execution plans and are considered when the system generates the execution plans.

Taking advantage of interesting order allows, in many cases, to avoid storage and sorting of intermediate query results, and raises the interest in the research sub-area of order optimization in query processing. *Order optimization* is the process in which the optimizer detects when interesting orders are generated, whether interesting orders are satisfied, where and how to sort when a sort is unavoidable.

Simmen et al. have presented techniques and overall architecture for order optimization in IBM's DB2 [74], a descendent of the *Starburst* optimizer. In DB2, an order specification is denoted as a list of columns in major to minor order $(o_1, o_2, \cdots, o_n)$. In order optimization, an order specification (an order property or interesting order) is first reduced to its canonical form by using predicates, keys and functional dependencies to remove redundant columns from the order specification. Then, algorithms are used to test whether an interesting order is satisfied, whether two interesting orders can be combined into one, and how to push down an interesting order in a query plan. In particular, this work describes how properties of input relations (order, predicates, keys, and functional dependencies) can be maintained through operators.

More recently, several papers have been published along this line of order optimization [30] [64] [85] [7]. Wang and Cherniack integrate reasoning about grouping and ordering into one framework [85]. Groupings are closely related to orderings in query processing, and can be treated similarly during plan generation. With this approach, keys, order properties, and functional dependencies are associated with each node (operator) in the execution plan, and order properties of intermediate

query results are inferred by using transformation rules. By checking whether a required ordering or grouping is satisfied by the execution plan, unnecessary sorting and grouping are removed from the plan. Essentially, this approach on the order inference part is an extension of Simmen's algorithm.

Inspired by this work, Neumann and Moerkotte propose a combined framework for grouping and order optimization [64] [65]. In this work, a finite state machine (FSM) is used as a tool to track the available orderings and groupings. They focus on including efficient grouping optimization into the same framework of order optimization. They apply Simmen's strategy of order inference in their work.

Both these two approaches adopt Simmen's framework and use functional dependencies to infer the ordering and grouping properties that the query results satisfy.

However, in these approaches with order optimization, interesting orders are represented by attribute orders, and therefore cannot be used to express arbitrary first-order expressible orders occurring in intermediate and final results in query processing. As a result, these methods cannot solve the problems in the motivating examples in Section 1.1.

1.3.2 Ordered Representations of Relations

It is well known that conventional relational algebras use sets or multi-sets as data models. This makes it hard to capture the order properties of data in these data models. Many researchers thus turned to representing data with ordered data structures, such as lists, sequences, or arrays.

Slivinskas et al. [75] distinguish three types of relation: list, multi-set and set. Correspondingly, all transformation rules are classified into three categories: list equivalence, multi-set equivalence, and set equivalence. Two relations are list equivalent if they are identical; multi-set equivalent if they are identical as multi-sets, taking into account duplicates, but not order; and set equivalent if they are identical as sets, ignoring duplicates and order. The operations include not only the conventional relational algebraic operations, such as selection, projection, union-all, cartesian product and difference, but also duplicate elimination, sorting, top-n and aggregation. During query plan generation, the system first decides the type of the root for the query tree, list, multi-set or set. Then, the required equivalence of each operation is derived at each level based on whether the operation is duplicate

and order relevant. The required equivalence of operations constrains the types of transformation rules that can be applied at each step of query enumeration.

Lerner et al. [57] extend SQL with order-related *assuming order* clauses to AQuery, which addresses *order-dependent* queries, *i.e.*, those queries whose results (viewed as multi-sets) change if the order of the input tuples is changed. In their data model, an *arrable* is defined as a finite, unbounded array of elements. Each element in an arrable is associated with an index or position. An arrable may be ordered by a subset of its attributes. A set of order-preserving operators are defined over arrables. In addition to sort elimination and move-around techniques [74] [75], Lerner et al. introduce edge built-in functions, such as *first* and *last*, which allow other aggressive order-related optimization. When an AQuery query is issued by a user, the optimizer matches the query's order with the input existing order, and decides whether, and when, further sorting is necessary. Optimization techniques for order-dependent queries are also addressed in this work. However, they define the operators by algorithms in a general-propose language, in which it is difficult to infer orderings and decide computability. In the proposed approach, the operators are defined using formulas, and therefore, the problems with algorithms can be avoided.

A number of approaches treat order by using sequences to represent relations [61] [70] [71] [67]. Seshadri et al. propose a data model of sequences in [70]. Tuples in a relation are ordered by their positions. They define a set of query operators on sequences. Certain specific operators, such as *positional offset* and *value offset*, are applied on input sequences to manipulate sequences. Seshadri et al. also develop optimization techniques, which are particular to sequences, to perform global and local query optimization.

In general, these approaches with ordered representations of data lack facilities of changing data orderings. Furthermore, none of these approaches has proposed a complete set of algebraic operators to express queries with any first-order expressible order.

1.3.3 Query Rewriting

In many database systems, transformation rules are used to rewrite a query into an equivalent query. Query rewriting is essential in query optimization, because the performance of equivalent queries can vary widely. The most important aspect of

query rewriting is that the resulting query can be executed more efficiently than the original query. This optimization technique has been extensively explored in the literature [45] [66] [22] [21], and widely applied in many systems. For example, query rewriting is used to flatten nested queries and quantifiers, and to avoid expensive nested-loop evaluations [56] [32].

Standard procedure of query rewriting consists of four steps:

1. A given query in SQL or other query language is mapped to an algebraic expression;

2. Conditions of the query are checked to match transformation rules from a set of rules;

3. If a rule is applicable to the query, the transformation rule is applied to obtain a new algebraic expression;

4. The optimal query is chosen among all alternative algebraic expressions, based on their execution cost.

In query rewriting, the resulting alternative algebraic expressions must be equivalent to the initial query. Even though query equivalence (or containment) is undecidable for relational queries in general [20] [51], the equivalence of queries is decidable for fragments of relational queries.

In ordered context, query equivalence is evaluated not only by the output data for general inputs, but also by the output order. In other words, two ordered algebraic expressions are equivalent if they generate the results with the same data set and order properties for any arbitrary input. For example, in [76] [75], transformation rules were studied for operations on ordered data representations and the conditions under which the transformation rules are applied are also illustrated.

In this dissertation, we offer transformation rules for ordered operations in the proposed ordered algebras. These transformation rules keep query equivalence in the way that both data and order equivalence are preserved. Query rewriting with these transformation rules of ordered operations provides an important means to optimize ordered query processing in ordered database systems.

1.3.4 Integrity Constraints Involving Order

Integrity constraints are knowledge about database instances [43], such as the database schema, the domain knowledge, the relationships between data, etc. They are used to define the semantics of the database, and are useful in many database applications, such as database consistency maintenance, view updates, and semantic query optimization [5] [17] [86].

Integrity constraints are used in semantic query optimization to improve the efficiency of query processing [17]. In semantic query optimization, new integrity constraints are inferred from integrity constraints that the relations hold originally. The new constraints convey all the semantic information from the original constraints, and are used later to generate equivalent, but more efficient, queries.

Order, as prevalent semantic information, has been exploited to enhance the query evaluation and optimization [41] [42] [34] [82]. Ginsburg and Hull [41] introduce a new integrity constraint, called *order dependency*, to incorporate semantic information involving orderings of a relation. Generally, order dependency exists between two attributes where one attribute has a specified order, and the other attribute will also have a specified order. Ginsburg and Hull provide a sound and complete set of inference rules for order dependencies, and demonstrate that the implication problem is co-NP complete.

Ginsburg and Hull also introduce the notion of *sort set* in [42]. A sort set is a subset of a relation's attributes that have total orders. They show that a relation has a sort set if and only if functional dependencies exist between any pair of attributes in this sort set. They also define a new integrity constraint *sort-set dependency* (SSD). Given that X is a subset of attributes, such that each attribute in X has a total order, a relation satisfies a sort set dependency $s(X)$ if X is a sort set of this relation. Ginsburg and Hull prove that deciding logical implication for sort-set dependencies requires polynomial time, but if functional dependencies are included, this problem becomes co-NP-complete.

1.3.5 Partial Order Databases

Raymond proposed a partial order data model to manipulate ordered sets of elements in [68]. A partial order is a pair of a finite base set and an order relation. A set of elements in the partial order are structured according to the order relationship. The types and sizes of elements are immaterial because the elements can

be treated as atomic when the order relationship is concerned. Operators can be classified into three categories: those that modify the base set and order relation of a partial order, those that extract suborders, and those that combine partial orders. Raymond also proposed *realizers* as a representation of the partial order model. A linear extension of a partial order is a total order that embeds the partial order. A realizer of a partial order is a set of linear extensions whose intersection is exactly the set of comparability and incomparability defined by the partial order. The realizers and linear extensions of partial orders are designed to hold partial orders. Corresponding operations are defined for individual linear extensions and the realizers.

The goal of this approach is closely relevant to our approach in that it endeavors to provide a systematic treatment for manipulating the ordered data. However, it is distinct from our approach in two essential ways: first, it models the ordering of data with a partial order, rather than a total order as in our approach; second, it represents partial orders with realizers, and denotes individual linear extensions of a partial order with array notations and individual elements of a linear extension with matrix notations. In our approach, the tuple identifiers are virtual, and used as a conceptual tool to express the order of data. The total order of tuples in a relation is captured with a binary order relation, and the order relation is changed along with the data relation when an ordered operation is performed on ordered relations.

1.3.6 Temporal Databases

Temporal relational databases are closely related to ordered relational databases in that both databases are based on ordered relational data models. The tuples in a temporal relation are sorted according to time; the tuples in an ordered relation are ordered in a physical order. Therefore, the temporal logic serves as a primary research foundation for our ordered database theory in this work.

The expressive power of different temporal languages has been one of important research areas in temporal logic. Various temporal languages were proposed on different temporal data models [29] [11] [27]. The choice of time domains is varying in temporal data models. The time representation could be linear or branching, discrete or dense, bounded or unbounded, points or intervals.

It has been proven that for the monadic first-order logic over linear orders, there is an expressively equivalent propositional temporal logic with an expressively

complete set of temporal operators [52] [38] [50]. However, this result cannot be generalized to the two-sorted first-order logic.

The two-sorted first-order language is strictly more expressive than the first-order temporal logic with connectives `since` and `until` for dense and linearly ordered time [53] [4]. Abiteboul *et al.* in [4] showed that this result holds for any finite discrete linear temporal domains. Generally, any first-order temporal calculus with an arbitrary finite set of temporal connectives is strictly less expressive than the two-sorted first-order calculus over linear temporal domains [81] [10] [80]. The separation of first-order temporal calculi from the two-sorted first-order calculus causes the failure of attempts to develop expressively complete temporal relational algebras that are subquery-closed [29] [11].

There have been several temporal algebras proposed in the literature, which are based on different temporal data models. One of representative examples of temporal algebras is introduced in [83]. In this work, Tuzhilin et al. proposed a temporal relational algebra by extending the standard relational algebra with extra temporal operators. However, this algebra is not a complete algebra when being compared to the two-sorted first order calculus. In general, none of temporal query languages satisfies both expressive completeness and subquery closure requirements at the same time.

1.3.7 Preference and Skyline Queries

Another related area to the proposed research is preference and skyline queries. In database systems, a user's preference, such as priority or relevance, is usually captured with preference relations. A preference relation is a binary relation between two query answers, and can be defined using preference constructors [54] [55] or logical formulas [24] [25]. To select the most preferred tuples from input relations according to a given preference relation, a special relational operator is typically added to existing relational query languages. This operator has been called *winnow* [24] [25] or Best-Matches-Only (BMO) operator [54] [55]. Preference can often be viewed as a partial order; therefore, the algebra of preference queries can adopt the partial order algebraic model.

Skyline queries are closely related to preference queries. The skyline operation filters a set of interesting (dominating) points from a large search space by weighting the user's preferences [12] [26]. A point dominates another point if it is as good

or better in all dimensions (such as price, distance) and better in at least one dimension. A point is interesting if it is not dominated by any other point. As the interesting points is decided by the combination of dominance relations, such as maximum, minimum, and difference, on all dimensions, a user's preference can be found among these interesting points. In other words, the skyline must contain any user's preference.

Order plays an important role in evaluating both preference queries and skyline queries. Preference and skyline queries are essentially relational queries that heavily rely on order properties in data relations. They are motivating our research in ordered relational databases, and our study will in turn benefit to the applications in these two areas.

1.3.8 XML Queries

Query optimization in XML database systems is closely related to order treatment in many aspects. The documents in XML themselves can be viewed as ordered trees (forests) with respect to the document order [3]. The query language XQuery, as the proposed standard query language for XML, returns results in a well-defined order. A path expression, which locates nodes by identifying their locations in the hierarchy of an XML document tree, also returns results in document order. Furthermore, the result of a FLOWR expression (with the constructs FOR, LET, ORDER BY, WHERE, and RETURN) reflects both the order imposed by ORDER BY clause and by the expressions in FOR subclauses [1] [2]. Considerable effort has been conducted on treatment of XML data order, for example, by storing and retrieving ordered XML data using relational database technique [79] [84] [33], and by efficiently answering XML queries with ordered results [36] [87] [73].

Typically, an XML database is represented as a forest of ordered, rooted, and labelled trees, with each node corresponding to an element and the edges to element-subelement relationships. The tree structure of XML documents is frequently encoded with numbering schemes [33] [88] [79]. Many numbering schemes have been developed to encode XML data, and one important approach is interval encoding and its variants. In interval encoding, a unique interval is assigned to each node, and the interval of a parent contains the intervals of all its children. Checking structural relationships between nodes, such as ancestor-descendant and parent-child relationships, corresponds to checking containment and direct containment relationships, respectively. Determining containment relationships amounts to checking certain

inequality conditions held between endpoints of intervals, which is basically related to the ordering of endpoints.

Queries in XML query languages, such as XQuery, retrieve data both by XML document structure and the values of document elements. When XML documents are encoded with a numbering scheme, evaluating path expression queries requires checking the structural relationships. Many approaches determine the structural relations by computing *structural joins* [6] [23] [14]. With structural joins, two lists of potential ancestors and potential descendants are joined using their interval containments as the join conditions. The output is a list of pairs of nodes with the required structural relationships.

Order-preserving algebras have been applied to XML databases for many purposes. Many approaches use order-preserving operations to facilitate the XQuery plan generation. For example, order-preserving operations are used to unnest nested queries in the XQuery language [62], or to model XML databases with Order-preserving algebras [9].

1.3.9 Top-k Queries

Another research direction involving orderings is the *top-k query* evaluation. Top-k queries return the first k results based on the value of a given score function. The eligible results are sorted by the score values before returning the first results. Top-k queries are extensively applied in many research areas, such as multi-attribute top-k queries over multimedia repositories [37] [63] [46], or top-k queries over relational databases [16] [19] [48]. Recently, rapid development of Internet boosts the growth of interest in Top-k queries [13].

Many algorithms were proposed to evaluate top-k queries using the scoring functions. In the cases that the top-k queries involve several independent scoring factors, a total score is computed using various aggregate algorithms [37] [63] [46]. However, a naive solution to evaluating top-k queries is prohibitively expensive because it requires producing and sorting all results before returning the k desired results. To date, little work has been done regarding the efficient sorting of tuples.

Ilyas et al. proposed *rank-aware query optimization* to evaluate top-k queries [49] [58] [77]. In this approach, they have extended the relational algebra and defined a rank-relation. A rank relation has its tuples ordered by a score function. Ilyas et al. also extended relational algebra by adding a new rank operator and

modifying the existing relational operators to be "rank-aware." For each tuple in an input relation, they compute the maximal possible score with respect to given scoring functions and predicates. Tuple streams pass through operators in the order of maximal possible scores, and new rank-relations are generated incrementally. When k results are produced or no more results are available, the execution terminates. The method avoids the high cost of sorting the entire relation used in previous approaches by incremental execution of queries.

In summary, approaches that solve top-k queries focus on efficiently fetching the first k results in terms of ranking predicates. They do not address the issues stemming from the total order of relations. Therefore, those approaches cannot solve the problems shown in the motivating examples in the previous section.

1.4 Organization of Dissertation

This chapter provides the motivation to the proposed research in ordered relational databases. The related work in relevant research areas is also briefly reviewed, such as order optimization, ordered representation of data structures, integrity constraints, and potential application areas of ordered relational databases.

The remainder of this dissertation is organized as follows. In Chapter 2 an ordered database model and a two-sorted first order calculus $\boldsymbol{FO}_{\preceq}$ are proposed to formally define the ordered relational queries. A comparison between ordered relational queries and temporal relational queries is conducted. The last section of Chapter 2 formally defines the completeness problem for general ordered relational queries. Chapter 3 investigates the completeness problems for ordered conjunctive queries with $\boldsymbol{x}$-decided order, $\boldsymbol{CQ}^{X}_{\preceq}$. In Chapter 4 the completeness problem is studied for ordered conjunctive queries with $\boldsymbol{t}$-decided order, $\boldsymbol{CQ}^{T}_{\preceq}$. Chapter 5 explores the completeness problem for general ordered conjunctive queries, $\boldsymbol{CQ}_{\preceq}$. The dissertation concludes in Chapter 6 with a summary of results and future work.

Chapter 2

The Ordered Relational Model

The most mature and widely used database model today is the relational data model. While other data models are gaining acceptance and popularity, we believe that the relational data model is still important in database theory. The study of order in the relational data model is fundamental to understanding order properties in more elaborate database models. The order support techniques that are developed for relational databases capture the essence of order support in other database models. Therefore, we begin with the relational data model to investigate ordered data models.

This chapter lays a theoretic foundation for discussing the completeness problem of ordered relational databases. The remainder of this chapter will proceed as follows. In Section 2.1 the ordered relational model is introduced and ordered relational databases are compared with temporal relational databases, which serve as a research foundation for ordered relational databases. In Section 2.2 the ordered relational queries are formally defined in a two-sorted first-order calculus $\boldsymbol{FO}_{\preceq}$; a reduction is shown from 2$\boldsymbol{FOL}$ temporal relational queries (see Definition 2.17 to $\boldsymbol{FO}_{\preceq}$ ordered relational queries. In Section 2.3 a general definition of ordered relational algebras is provided; the completeness problem of ordered relational algebras is proposed.

2.1 Ordered Relational Databases

Conventional relational databases are set or multi-set based, no order exists among tuples in a relation. However, ordering always exists in data if we view data as data

streams flowing through the query processor in relational databases. We can the order in which an iterator retrieve data as the *physical order* of data. The physical order provides a means to capture ordering in data, which is independent of data models. In this dissertation, advantage of the physical order of data is taken to capture and manipulate ordering of data in the ordered relational data model.

2.1.1 Ordered Relational Databases

First, a data model must be specified to represent the physical order of tuples in a relation. The order in this data model must be a total order on all tuples in a relation. In addition, the order should be associated with different properties. For example, this order can be linear or branchy, discrete or dense, bounded or unbounded, among others.

A linear, discrete, and bounded representation is proposed to indicate the order of a relation for the following reasons. First, a relation is a set of tuples; naturally, the order of tuples can be viewed as a discrete order. Second, we consider only finite relation instances; therefore, the order of relation is finite and bounded. Finally, it is simple and natural to treat orders as linear (total) orders in the output of each operator while implementing the query.

To realize the linear order of a relation, the concept of *tuple identifiers* is introduced, which plays an enormous role in our investigation. The tuple identifier, denoted by T, is a virtual attribute, which is only used to represent ordering in relational databases. To capture the data ordering, a value of attribute T is assigned to each tuple in the relation and is called a tuple identifier t. The tuple identifier does not exist physically, and it is only a conceptual representation such that we can use it to indicate order at the algebraic level.

Let $\mathrm{S}(\mathrm{X}_1, \cdots, \mathrm{X}_n)$ be a relation schema in a conventional relational database. In our ordered relational model, a unique tuple identifier is attached to each tuple in the relation, and hence the relation schema is extended to $\mathrm{S}(\mathrm{T}, \mathrm{X}_1, \cdots, \mathrm{X}_n)$, which we call the *extended relation schema*. Formally, the extended relation schema is defined in the ordered context as follows:

Definition 2.1 (Extended Relation Schema)
An extended relation schema *is a pair* $\mathrm{S} = ((\mathrm{T}, \mathrm{X}), (\mathbf{Dom}_\mathrm{T}, \mathbf{Dom}_\mathrm{X}))$*, where* T *is the tuple identifier attribute and* X *is a finite set of attributes* $\{\mathrm{X}_1, \cdots, \mathrm{X}_n\}$*.* $\mathbf{Dom}_\mathrm{T}$ *is the domain of tuple identifiers* T*, and* $\mathbf{Dom}_\mathrm{X} = \bigcup_{i=1,\ldots,n} \mathbf{Dom}(\mathrm{X}_i)$*.*

An order variable *or* ***t***-variable *is a variable over the domain* $\mathbf{Dom}_{\mathrm{T}}$*, denoted by* t*, possibly with subscripts. A data variable or* ***x***-variable *is a variable over the domain* $\mathbf{Dom}_{\mathrm{X}}$*, denoted by* x*, possibly with subscripts.*

To simplify the notation, an extended relation schema is often abbreviated as $\mathrm{S}(\mathrm{T}, \mathrm{X}_1, \cdots, \mathrm{X}_n)$ in ordered relational databases.

Definition 2.2 (Ordered Relation Schema and Database Schema)
An ordered relation schema *is a pair* $\langle \mathrm{S}, \preceq_{\mathrm{S}} \rangle$*, where* S *is an extended relation schema, and* $\preceq_{\mathrm{S}}$ *is a relation schema* $(\mathrm{T}, \mathrm{T}')$*. Here,* T *and* T' *are tuple identifier attributes over* $\mathbf{Dom}_{\mathrm{T}}$*. We call* $\preceq_{\mathrm{S}}$ *the* order schema.
An ordered database schema *is a set of ordered relation schemas, denoted by* $\{\langle \mathrm{S}_1, \preceq_{\mathrm{S}_1} \rangle, \ldots, \langle \mathrm{S}_m, \preceq_{\mathrm{S}_m} \rangle\}$*. Here,* m *is the number of ordered relation schemas.*

In the definition of an ordered relation schema, the order schema $\preceq_{\mathrm{S}}$ defines a binary relation between tuple identifiers T in the extended relation schema S. An order schema is abbreviated as $\preceq_{\mathrm{S}} (\mathrm{T}, \mathrm{T}')$ in the following development. Since all ordered relations have the same order schema, we omit the order schema and refer to the ordered relation schema as the extended relation schema when the meaning can be revealed from the context.

We now define the instance of an ordered relation schema, which is abbreviated as *ordered relation*, when there is no confusion with other aspects of an ordered relation such as its schema.

Definition 2.3 (Ordered Relation)
Let $\mathrm{S}(\mathrm{T}, \mathrm{X}_1, \cdots, \mathrm{X}_n)$ *be an extended relation schema, and* $\langle \mathrm{S}, \preceq_{\mathrm{S}} \rangle$ *be the corresponding ordered relation schema. An* ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ *is an instance of the ordered relation schema* $\langle \mathrm{S}, \preceq_{\mathrm{S}} \rangle$*, where* R *is called the* data relation*, a standard relation conforming to the relation schema* $\mathrm{S}(\mathrm{T}, \mathrm{X}_1, \cdots, \mathrm{X}_n)$*; and* $\preceq_{\mathrm{R}}$ *is called the* order relation*, a binary relation conforming to the order schema* $\preceq_{\mathrm{S}} (\mathrm{T}, \mathrm{T}')$*.*

An ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ *satisfies the following conditions:*

(*a*) *the data relation* R *and order relation* $\preceq_{\mathrm{R}}$ *are inclusively dependant on each other, i.e.,* R *and* $\preceq_{\mathrm{R}}$ *satisfy the following inclusion dependencies:*

(1) $\mathrm{R}[\mathrm{T}] \subseteq \preceq_{\mathrm{R}} [\mathrm{T}] \cup \preceq_{\mathrm{R}} [\mathrm{T}']$*;*

(2) $\preceq_{\mathrm{R}} [\mathrm{T}] \subseteq \mathrm{R}[\mathrm{T}]$*;*

(3) $\preceq_{\mathrm{R}} [\mathrm{T}'] \subseteq \mathrm{R}[\mathrm{T}]$,

where $\mathrm{R}[\mathrm{T}]$ *and* $\mathrm{R}[\mathrm{T}']$ *are the projection of* R *on* T*, and* $\preceq_{\mathrm{R}}[\mathrm{T}]$ *is the projection of* $\preceq_{\mathrm{R}}$ *on* T*.*

(*b*) *the order relation* $\preceq_{\mathrm{R}}$ *is a linear order, i.e.,* $\preceq_{\mathrm{R}}$ *satisfies*

(1) $\forall\ t_1, t_2, t_3 \in \mathrm{R}[\mathrm{T}].\ (t_1, t_2) \in \preceq_{\mathrm{R}} \wedge\ (t_2, t_3) \in \preceq_{\mathrm{R}} \rightarrow\ (t_1, t_3) \in \preceq_{\mathrm{R}}$*;*

(2) $\forall\ t_1, t_2 \in\ \mathrm{R}[\mathrm{T}].\ (t_1, t_2) \in \preceq_{\mathrm{R}} \wedge\ (t_2, t_1) \in \preceq_{\mathrm{R}} \rightarrow\ t_1 = t_2$*;*

(3) $\forall\ t_1, t_2 \in \mathrm{R}[\mathrm{T}].\ (t_1, t_2) \in \preceq_{\mathrm{R}}\ \vee\ (t_2, t_1) \in \preceq_{\mathrm{R}}$*,*

where $\mathrm{R}[\mathrm{T}]$ *is the projection of* R *on* T*.*

Intuitively, $\preceq_{\mathrm{R}}$ is a preceder-successor relation between each pair of tuple identifiers in the data relation R. If $(t_1, t_2) \in \preceq_{\mathrm{R}}$, we say that t_1 is *prior to* t_2 in the ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. Sometimes, we say $t_1 \preceq_{\mathrm{R}} t_2$ when $(t_1, t_2) \in \preceq_{\mathrm{R}}$. We omit the subscript R in $\preceq_{\mathrm{R}}$ if it is not needed for understanding the expression. Sometimes, we abbreviate an ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ as $\mathbf{R}$.

The definition of ordered relations specifies two semantic conditions that an ordered relation should satisfy: (*a*), saying that the data relation R and order relation $\preceq_{\mathrm{R}}$ are inclusively dependent on each other, guarantees that the order relation $\preceq_{\mathrm{R}}$ defines ordering for any pair of tuples in R, and only pairs of tuples in R; (*b*) ensures that the order defined by the order relation $\preceq_{\mathrm{R}}$ must be a linear and total order over the data relation R.

We have defined ordered structures and ordered relations, and are now in a position to give a formal definition to ordered relational databases.

Definition 2.4 (Ordered Relational Databases) *Let* $\mathbf{Dom}_{\mathrm{T}}$ *be the domain of tuple identifiers* T*, and "=" be an equality predicate over* $\mathbf{Dom}_{\mathrm{T}}$*;* $\mathbf{Dom}_{\mathrm{X}}$ *be the domain of data attributes* X*, and "=" be an equality predicate over* $\mathbf{Dom}_{\mathrm{X}}$*. Assume that* $\{\langle \mathrm{S}_1, \preceq_{\mathrm{S}_1} \rangle, \ldots, \langle \mathrm{S}_m, \preceq_{\mathrm{S}_m} \rangle\}$ *is an ordered database schema defined on* $\mathbf{Dom}_{\mathrm{T}}$ *and* $\mathbf{Dom}_{\mathrm{X}}$*.*

An ordered relational database *is a structure* $\langle \mathbf{Dom}_{\mathrm{T}}, =; \mathbf{Dom}_{\mathrm{X}}, =; \mathbf{R}_1, \ldots, \mathbf{R}_m \rangle$*, where* $\mathbf{R}_1, \ldots, \mathbf{R}_m$ *are ordered relations conforming to ordered relation schemas* $\{\langle \mathrm{S}_1, \preceq_{\mathrm{S}_1} \rangle, \ldots, \langle \mathrm{S}_m, \preceq_{\mathrm{S}_m} \rangle\}$*, respectively.*

An example of ordered relational databases is described next.

Example 2.5 Consider a relational database which stores the information of employees in a company. Let `emp` (`EmpId`, `Name`, `Department`, `Salary`) be the only

relation schema in this database, with `EmpId` as the primary key attribute. Thus, the corresponding extended relation schema is as follows:

$$\mathrm{S}(\mathrm{T}, \texttt{EmpId}, \texttt{Name}, \texttt{Department}, \texttt{Salary}).$$

The ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$ is an instance of the ordered relation schema $\langle \mathrm{S}, \preceq_{\mathrm{S}} \rangle$. It contains two components: the data relation `EMP` and the order relation $\preceq_{\texttt{EMP}}$, which are shown in Tables 2.1 and 2.2, respectively.

Table 2.1: The data relation `EMP`

T	EmpId	Name	Department	Salary
12	5134	Eric Cook	Development	80,000
23	5086	Emily Smith	Sales	65,000
8	5002	Kavin Stone	Sales	55,000
15	5258	Jack Anderson	Sales	48,000
...	...	...	...	...

Table 2.2: The order relation $\preceq_{\texttt{EMP}}$

T	T′
...	...
8	8
8	12
8	15
8	23
12	12
12	15
12	23
23	23
23	15
15	15
...	...

In this example instance, the domain of tuple identifiers T is the set of natural numbers $\mathcal{N}$. The data relation `EMP` is a conventional relation under the extended schema $\mathrm{S}(\mathrm{T}, \texttt{EmpId}, \texttt{Name}, \texttt{Department}, \texttt{Salary})$, which is a set of tuples without

ordering. Each tuple in EMP is associated with a unique tuple identifier from $\mathcal{N}$. The order relation $\preceq_{\texttt{EMP}}$ is a binary relation over all pairs of tuple identifiers in EMP, defining a linear order on EMP. The conceptual ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$ is shown in Table 2.3.

Table 2.3: The ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$

T	EmpId	Name	Department	Salary
...	...	...	...	...
8	5002	Kavin Stone	Sales	55,000
...	...	...	...	...
12	5134	Eric Cook	Development	80,000
...	...	...	...	...
23	5086	Emily Smith	Sales	65,000
...	...	...	...	...
15	5258	Jack Anderson	Sales	48,000
...	...	...	...	...

The above example illustrates properties of an ordered relation. Next, we will define equivalence for ordered relations. In ordered relational databases, two ordered relations are equivalent if they have the same tuples regarding to data attributes, and the same ordering among tuples. They may have different tuple identifiers for corresponding data tuples because tuple identifiers are conceptual, only used to indicate orderings among data tuples.

Definition 2.6 (Equivalent Ordered Relations) *Let* $\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle$ *and* $\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle$ *be two ordered relations with the same extended schema.* $\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle$ *and* $\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle$ *are equivalent if there exists a bijection* f *such that for each tuple* t_2 *in* R_2*, there is exactly one tuple* t_1 *in* R_1 *such that*

$$f(t_1) = t_2,$$

and t_1 *and* t_2 *have the same values on data attributes.*

2.1.2 Ordered Relational Databases vs. Temporal Relational Databases

In this dissertation, temporal database theory serves as a research foundation to investigate ordered relational databases and query languages. To make the dissertation self-contained, we conduct a brief review of basic notions and notations of temporal databases.

The precise notion of a temporal database is defined as a temporal structure, which consists of three components: the temporal domain, the data domain, and a set of temporal relations [81]. In the following chapters, we tailor the notions to the one-dimension temporal context, and consider only the timestamp temporal data model.

Definition 2.7 (Temporal Databases) *Let $\langle \mathbf{T}, \leq \rangle$ be a temporal domain with the inequality predicate $\leq$, defining a linear order on $\mathbf{T}$; $\langle \mathbf{D}, = \rangle$ be the data domain with the equality predicate $=$; $\langle s_1, \ldots, s_m \rangle$ be a database schema with relation schemas $s_i(T, X_1, ..., X_{v_i})$ (for $i = 1, ..., m$).*

A temporal database *is a two-sorted structure*

$$\langle \mathbf{T}, \leq; \mathbf{D}, =; r_1, \ldots, r_m \rangle,$$

where r_i is a temporal relation conforming to the relation schema s_i, and the set of tuples $(x_1, \ldots, x_{v_i})$ that satisfy $s_i(t, x_1, \ldots, x_{v_i})$ are finite for every $t \in \mathbf{T}$.

Certain differences exist between temporal and ordered relational databases:

1. In temporal relations, values of the temporal attribute T are time instants. There may be more than one tuple for each time instant t; in ordered relations, values of the T attribute are tuple identifiers. The tuple identifier t is unique to each tuple.

2. In temporal relations, the order of tuples is a partial order, *i.e.*, an order always exists between tuples with different time instants t; however, no order exists among tuples with the same time instant t; in ordered relations, the order of tuples is a linear and total order, *i.e.*, each pair of tuples are comparable, and all tuples must be in a linear order.

3. In temporal databases, the order of time instants is universal, *i.e.*, time instants from different temporal relations can be compared; in ordered databases, only

tuple identifiers from the same ordered relations are in order. There is no way to compare tuple identifiers from different ordered relations.

Given the differences between temporal and ordered relational databases, the following lemma shows that any temporal database can be mapped to an ordered relational database in such a way that the mapping ordered relations retain the order properties of the original temporal relations.

Without lost of generality, we make an assumption for the rest of this section that there is only one temporal relation in the temporal database involved, and we have only one input temporal relation for temporal queries in the following proofs. The reason for this is that multiple temporal relations can be encoded into one relation.

Lemma 2.8 *Let* $\langle \mathbf{T}, \leq; \mathbf{D}, =; r_1, ..., r_n \rangle$ *be an arbitrary temporal database, where* $\mathbf{T}$ *is a discrete, linear and bounded temporal domain,* $\mathbf{D}$ *is the union of domains of data attributes, and* $r_1, ..., r_n$ *are temporal relations.*

There exists a one-to-many correspondence ρ *from the set of temporal relations to the set of ordered relations such that for an arbitrary temporal relation* r*, there is a set of ordered relations* $\{\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle\}$ *and*

$$\rho(r) = \{\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle\}.$$

Proof: Let $\mathbf{Dom}_{\mathrm{T}}$ and $\mathbf{Dom}_{\mathrm{X}}$ be the domains of tuple identifiers and data attributes of an ordered database, respectively. We define $\mathbf{Dom}_{\mathrm{T}}$ as the set of natural numbers $\mathcal{N}$ and combinations of natural numbers, and $\mathbf{Dom}_{\mathrm{X}} = \mathbf{D}$. The equality predicates on the two domains follow the standard meaning.

In temporal relations, tuples can be ordered by time instants, and multiple tuples may have the same time instant t; in ordered relations, each tuple has a unique tuple identifier t. This difference suggests that one temporal relation r is corresponding to multiple ordered relations which have the same data set, and retain the order of time instants.

We are now ready to construct a set of ordered relations $\{\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle\}$ that are corresponding to the temporal relation r and have the same data relation R. To uniquely identify the tuples in the temporal relation r, from now on, we use (t, i) to refer to the particular tuple which is the i-th tuple in the physical order at the time instant t.

First, we need to construct the data relation R which is uniquely mapping to the temporal relation r. We define a function $F : \mathbf{T} \times \mathcal{N} \rightarrow \mathbf{Dom}_{\mathrm{T}}$, where $\mathcal{N}$ is the set of natural numbers, such that the i-th tuple with value $(t, x_1, \ldots, x_n)$ in the temporal relation r, is uniquely mapped to a data tuple $(F(t, i), x_1, \ldots, x_n)$ in the data relation R. In this way, we construct the data relation R that is uniquely mapping to the temporal relation r.

To distinguish tuples at different time instants and, hence, capture the order of time instants in the mapping ordered relations $\{\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle\}$, we insert a special tuple for each time instant t in the data relation R, which we call a *separation tuple*. These separation tuples are constructed in the following way: first, we extend the domain of data attributes with a special value a^0 ($a^0 \notin \mathbf{Dom}_{\mathrm{X}}$), i.e., $\mathbf{Dom}_{\mathrm{X}} \cup \{a^0\}$; then, we add a separation tuple $(F(t, 0), a^0, \ldots, a^0)$ to R for each time instant t in the temporal relation r. The tuples corresponding to the same time instant (including the separation tuple) are called a *cluster*.

Thus, the data relation R is constructed for the ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, which is uniquely mapping to the temporal relation r and includes a separation tuple for each time instant.

Then, we define all possible order relations $\preceq_{\mathrm{R}}$ for the corresponding ordered relations $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. Recall that the tuples in the temporal relation r are in a partial order such that tuples at different time instants are sorted in order of time; tuples at the same time instant are not in order at all. We desire that the corresponding ordered relations $\langle R, \preceq_{\mathrm{R}} \rangle$ keep the same order of time instants as in the temporal relation r.

The order relations $\preceq_{\mathrm{R}}$ are defined in the following way: for any two tuples $F(t_1, i)$ and $F(t_2, j)$ in the data relation R ($i \geq 0$ and $j \geq 0$),

(1) if $t_1 < t_2$, then $(F(t_1, i), F(t_2, j)) \in \preceq_{\mathrm{R}}$;

(2) if $t_1 = t_2$ and $i = 0$, then $(F(t_1, i), F(t_2, j)) \in \preceq_{\mathrm{R}}$;

(3) if $t_1 = t_2$ and $j = 0$, then $(F(t_2, j), F(t_1, i)) \in \preceq_{\mathrm{R}}$;

(4) if $t_1 = t_2$, $i \neq 0$, and $j \neq 0$, $F(t_1, i)$ and $F(t_2, j)$ are arbitrarily ordered such that $\preceq_{\mathrm{R}}$ is a linear order of R.

Thus, the temporal relation r is corresponding to a set of ordered relations $\{\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle\}$ that have the same data relation and order of clusters, but have different orders inside clusters.

We have now constructed a set of ordered relations $\{\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle\}$ corresponding to the temporal relation r; *i.e.*, there is a one-to-many correspondence ρ such that for any temporal relation r, there exists a set of ordered relations $\{\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle\}$ and $\rho(r) = \{\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle\}$. This concludes the proof.

□

Example 2.9 The temporal relation in Table 2.4 shows a fragment of class calendar at the department of computer science. The relation has a schema $Class(Time, Room, Class)$. The attribute $Time$ in this instance is the temporal attribute, and $Room$ and $Class$ are data attributes.

Table 2.4: An example of temporal relations

$Time$	$Room$	$Class$
2010-05-08 9:00	DC1302	Advanced Algorithms
2010-05-08 9:00	DC1304	Database Systems
2010-05-08 10:00	DC1302	Programming Principles
2010-05-08 11:00	DC1302	Data Structures
2010-05-08 11:00	DC1304	Computation Theory
...	...	...

This temporal relation can be mapped to a set of ordered relations using the technique in our proof. Two of the corresponding ordered relations are shown in Tables 2.5 and 2.6.

In both ordered relations, a separation tuple occurs in front of tuples at the same time instant; the tuples at different time instants are sorted according to the order of time instants. The two ordered relations have the same set of data tuples, however they differ at that the tuples at the time instant "2010-05-08 9:00" have different orders.

This example indicates that the corresponding ordered relations of a temporal relation r have the same data relation R, which is uniquely decided by r. The tuples at the same time instant t are mapping to a group of tuples in R, with a separation tuple $F(t, 0)$ in front. The tuples at different time instants t are sorted according to the order of time instants.

Table 2.5: One ordered relation corresponding to the temporal relation *Class*

T	*Room*	*Class*
201005080900000	AAAAAA	AAAAAAAAAAAAAAAA
201005080900001	DC1302	Advanced Algorithms
201005080900002	DC1304	Database Systems
201005081000000	AAAAAA	AAAAAAAAAAAAAAAA
201005081000001	DC1302	Programming Principles
201005081100000	AAAAAA	AAAAAAAAAAAAAAAA
201005081100001	DC1302	Data Structures
201005081100002	DC1304	Computation Theory
...	...	...

Table 2.6: Another ordered relation corresponding to the temporal relation *Class*

T	*Room*	*Class*
201005080900000	AAAAAA	AAAAAAAAAAAAAAAA
201005080900002	DC1304	Database Systems
201005080900001	DC1302	Advanced Algorithms
201005081000000	AAAAAA	AAAAAAAAAAAAAAAA
201005081000001	DC1302	Programming Principles
201005081100000	AAAAAA	AAAAAAAAAAAAAAAA
201005081100001	DC1302	Data Structures
201005081100002	DC1304	Computation Theory
...	...	...

By the construction in the proof of Lemma 2.8, all tuples in the corresponding ordered relation are grouped according to the time instant t. The group of tuples mapping to the same time instant (including the separation tuple) is called a *tuple cluster*, or a *cluster* for abbreviation. The clusters are sorted in the order of time instants and the tuples inside each cluster are sorted in an arbitrary order with the separation tuple in front of the cluster. Separation tuples indicate where clusters begin. Such an ordered relation is called a *clustered ordered relation*. Clustered ordered relations have a two-level structure: clusters of tuples and tuples inside clusters (including separation tuples). We give formal definitions for clusters and the order of clusters as follows:

Definition 2.10 (Clusters) *A cluster K in a clustered ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ is a set of tuples, including a separation tuple.*

Definition 2.11 (The Order of Clusters) *Let $K_1, \ldots, K_n$ be the set of clusters in a clustered ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, we say $K_i \preceq_{\mathrm{R}} K_j$ $(i, j = 1, \ldots, n)$ if for any two tuples $t_1 \in K_i$ and $t_2 \in K_j$, $t_1 \preceq_{\mathrm{R}} t_2$.*

The order of clusters *is the order among clusters $K_1, \ldots, K_n$ in the ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, denoted as $\preceq_{\mathrm{R}}^{K}$.*

Originally, the order of clusters $\preceq_{\mathrm{R}}^{K}$, in a clustered ordered relation, is decided by the order of time instants in the original temporal relation. The order of tuples inside a cluster is arbitrary and differs in different clustered ordered relations that are mapping to the same temporal relation; however, separation tuples are always in front of clusters in the clustered ordered relation.

In **??**, for time instants t_1 and t_2 satisfying $t_1 < t_2$ in the temporal relation r, the cluster mapping to t_1 is always prior to the cluster mapping to t_2 in the mapping ordered relations $\{\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle, \ldots, \langle \mathrm{R}, \preceq'_{\mathrm{R}} \rangle\}$. The order inside a cluster is arbitrary and differs in different mapping ordered relations.

In summary, a temporal relation is corresponding to a set of clustered ordered relations with the same data relation and the same order of clusters, but with different orders inside clusters. We define such a set of clustered ordered relations as *cluster-equivalent* ordered relations.

Definition 2.12 (Cluster-equivalent Ordered Relations) *Let $\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle$ and $\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle$ be two clustered ordered relations, and $\{K_{11}, \ldots, K_{1n}\}$ and $\{K_{21}, \ldots, K_{2m}\}$ be the sets of clusters in them, respectively. Ordered relations $\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle$ and $\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2}$*

$\rangle$ *are* cluster-equivalent *if* $\mathrm{R}_1 = \mathrm{R}_2$*,* $n = m$*,* $K_{11} = K_{21}, \ldots, K_{1n} = K_{2n}$*, and* $\preceq^K_{\mathrm{R}_1} = \preceq^K_{\mathrm{R}_2}$.

In this definition, cluster-equivalent ordered relations have the same order of clusters; however, they have different orders inside the clusters corresponding to the same time instant. In Figure **??**, ordered relations $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ and $\langle \mathrm{R}, \preceq'_{\mathrm{R}} \rangle$ are cluster-equivalent.

2.2 Ordered Relational Queries

In this dissertation, we limit our scope to one of the most important classes of ordered relational queries, the first-order expressible ordered relational queries. Analogously to conventional relational database theory, first-order logic serves as a yardstick to measure the expressive powers of other ordered query languages.

This section introduces the two-sorted first order calculus, $\boldsymbol{FO}_{\preceq}$ and ordered relational queries in $\boldsymbol{FO}_{\preceq}$. Furthermore, ordered relational queries in $\boldsymbol{FO}_{\preceq}$ are compared with temporal relational queries in $2\boldsymbol{FOL}$, and a reduction is conducted from temporal relational queries in $2\boldsymbol{FOL}$ to ordered relational queries in $\boldsymbol{FO}_{\preceq}$.

2.2.1 $\boldsymbol{FO}_{\preceq}$ Ordered Relational Queries

The first-order calculus that is applied to represent ordered relational queries contains two sorts of variables: $\boldsymbol{t}$-variables and $\boldsymbol{x}$-variables. This two-sorted first-order calculus on ordered relational databases is denoted by $\boldsymbol{FO}_{\preceq}$. The formal definitions of the two-sorted first-order calculus $\boldsymbol{FO}_{\preceq}$ and of the ordered relational queries in $\boldsymbol{FO}_{\preceq}$ are given as follows.

Definition 2.13 (Ordered Relational Queries in $\boldsymbol{FO}_{\preceq}$)
$\boldsymbol{FO}_{\preceq}$ *is a two-sorted first-order calculus over an ordered relation schema* $\langle \mathrm{S}(\mathrm{T}, \mathrm{X}_1, \ldots, \mathrm{X}_n), \mathrm{S}'(\mathrm{T}, \mathrm{T}') \rangle$*. A first-order formula* F *in* $\boldsymbol{FO}_{\preceq}$ *is defined by a BNF rule:*

$$
\begin{array}{rcl}
F & ::= & R(t, x_1, \ldots, x_n) \\
& | & t = (t_i, t_j) \\
& | & t_1 \preceq_R t_2 \\
& | & x_i = x_j \\
& | & F \wedge F \\
& | & \neg F \\
& | & \exists t.F \\
& | & \exists x.F,
\end{array}
$$

where t, t_1, t_2 are $\boldsymbol{t}$*-variables and $x_1, \ldots, x_n, x_i, x_j$ are* $\boldsymbol{x}$*-variables. An* $\boldsymbol{FO}_{\preceq}$ *ordered relational query $\langle \varphi, \psi \rangle$ is of the form*

$$\langle\ \{(t, x_1, \ldots, x_n) \mid \varphi(t, x_1, \ldots, x_n)\}, \{(t_1, t_2) \mid \psi(t_1, t_2)\}\ \rangle,$$

where φ is the data query, which is a first-order formula in $\boldsymbol{FO}_{\preceq}$ *with only one free* $\boldsymbol{t}$*-variable; and ψ is the order query, which is a first-order formula in* $\boldsymbol{FO}_{\preceq}$ *with exactly two* $\boldsymbol{t}$*-variables as the only free variables.*

In this definition, $t = (t_i, t_j)$ is a tuple identifier constructor, which is one way to generate a new tuple identifier from inputs. This constructor combines tuple identifiers from two ordered relations into one tuple identifier. More generally, in the case that the output tuple identifiers are combinations of n tuple identifiers from n input relations respectively, we simplify the output tuple identifiers $((...(t_1, t_2), ...), t_n)$ as $(t_1, t_2, ..., t_n)$.

As indicated by our definition of ordered relational queries, the restriction of one free $\boldsymbol{t}$-variable in the data query φ ensures that the data query always generates a data relation; the restriction of two $\boldsymbol{t}$-variables in the order query ψ guarantees that the output of the order query ψ is always an order relation. Thus, for any input ordered relations, an $\boldsymbol{FO}_{\preceq}$ ordered relational query $\langle \varphi, \psi \rangle$ produces an ordered relation as its output. Therefore, the set of $\boldsymbol{FO}_{\preceq}$ ordered relational queries over the universe of ordered relations is closed.

However, the restriction of two $\boldsymbol{t}$-variables as the only free variables in the order query applies only to the whole formulas of ordered queries, not to subformulas of ordered queries. This restriction is imposed because the $\boldsymbol{FO}_{\preceq}$ ordered relational queries are not subquery-closed; that is, a subformula of an $\boldsymbol{FO}_{\preceq}$ ordered relational query may not be an $\boldsymbol{FO}_{\preceq}$ ordered relational query. This situation will lead to difficulties in the search for complete algebras for ordered relational queries, as we shall see later.

Regarding query equivalence, both data and order queries of ordered relational queries have to be taken into account. For equivalent input ordered relations, which means that the input relations have identical values on data attributes and identical ordering among tuples, equivalent ordered queries should output equivalent ordered relations. The formal definition of equivalent ordered queries is given as follows:

Definition 2.14 (Equivalent Ordered Queries) *Two ordered relational queries are equivalent if and only if for equivalent input ordered relations, they output equivalent ordered relations.*

Meanwhile, Definition 2.13 is only a syntactic definition of an ordered relational query. Semantically, the order query ψ must specify a total (linear) order on the data relation φ. Therefore, a *valid ordered relational query* must satisfy both syntactic and semantic restrictions on it as shown in the following definition.

Definition 2.15 (Valid Ordered Relational Query)
An ordered relational query is valid if it always outputs an ordered relation for any input ordered relations.

Example 2.16 Consider an ordered relational database containing two ordered relations $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$ and $\langle \texttt{DEPT}, \preceq_{\texttt{DEPT}} \rangle$. The extended schema of $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$ is EMP (T, EmpId, Name, DeptId, Salary), and the extended schema of $\langle \texttt{DEPT}, \preceq_{\texttt{DEPT}} \rangle$ is DEPT (T, DeptId, Department). The conceptual ordered relations $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$ and $\langle \texttt{DEPT}, \preceq_{\texttt{DEPT}} \rangle$ are shown in Tables 2.7 and 2.8 respectively.

Table 2.7: The ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$

T	EmpId	Name	DeptId	Salary
...	...	...	...	...
8	5002	Kavin Stone	001	55,000
...	...	...	...	...
12	5134	Eric Cook	002	80,000
...	...	...	...	...
23	5086	Emily Smith	001	65,000
...	...	...	...	...
15	5258	Jack Anderson	001	48,000
...	...	...	...	...

Table 2.8: The ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$

T	DeptId	Department
1	001	Sales
2	002	Development
...	...	...

The following are examples of ordered relational queries over this ordered relational, which are formulated in $\boldsymbol{FO}_{\preceq}$.

- The query to output all employees in the department Sales in the input order can be formulated in $FO_{\preceq}$ as follows:

$$\begin{array}{ll} \langle\{(t, x_2)| & \exists x_1, x_3, x_4.\ \texttt{EMP}\ (t, x_1, x_2, x_3, x_4) \\ & \wedge \exists t_1, x_5, x_6.\ \texttt{DEPT}\ (t_1, x_5, x_6) \wedge x_3 = x_5 \wedge x_6 = \text{“}Sales\text{”}\}, \\ \{(t, t')| & \exists t_1, t_2.\ t = (t_1, t_2) \wedge \exists x_1, x_2, x_3, x_4.\ \texttt{EMP}\ (t_1, x_1, x_2, x_3, x_4) \\ & \wedge \exists t_2, x_5, x_6.\ \texttt{DEPT}\ (t_2, x_5, x_6) \wedge x_3 = x_5 \wedge x_6 = \text{“}Sales\text{”} \\ & \wedge\ \exists t'_1, t'_2.\ t' = (t'_1, t'_2) \wedge \exists x'_1, x'_2, x'_3, x'_4.\ \texttt{EMP}\ (t'_1, x'_1, x'_2, x'_3, x'_4) \\ & \wedge \exists t'_2, x'_5, x'_6.\ \texttt{DEPT}\ (t'_2, x'_5, x'_6) \wedge x'_3 = x'_5 \wedge x'_6 = \text{“}Sales\text{”} \\ & \wedge\ t \preceq_{\texttt{EMP}} t'\}\rangle. \end{array}$$

  Please note that the output of the order query is a set of tuple identifiers $\{(t, t')\}$, where t and t' are combinations of tuple identifiers $t = (t_1, t_2)$ and $t' = (t'_1, t'_2)$, constructed by the tuple identifier constructor.

- The query to generate a list of all employees whose salary is \$55,000 per year in the reverse order to the input order can be constructed in $FO_{\preceq}$ using the following formula:

$$\begin{array}{ll} \langle\{(t, x_2)| & \exists x_1, x_3, x_4.\ \texttt{EMP}\ (t, x_1, x_2, x_3, x_4) \wedge x_4 = 55,000\}, \\ \{(t_1, t_2)| & \exists x_{11}, x_{12}, x_{13}, x_{14}.\ \texttt{EMP}\ (t_1, x_{11}, x_{12}, x_{13}, x_{14}) \wedge x_{14} = 55,000 \\ & \wedge\ \exists x_{21}, x_{22}, x_{23}, x_{24}.\ \texttt{EMP}\ (t_2, x_{21}, x_{22}, x_{23}, x_{24}) \wedge x_{24} = 55,000 \\ & \wedge\ \neg(t_1 \preceq_{\texttt{EMP}} t_2)\}\rangle. \end{array}$$

2.2.2 $FO_{\preceq}$ Ordered Relational Queries vs. $2FOL$ Temporal Relational Queries

Various first-order representations of temporal relational queries have been proposed in temporal database communities. To compare with the $\boldsymbol{FO}_{\preceq}$ ordered relational query, we choose one temporal language which represents the temporal variables explicitly. The formal definition of the two-sorted first-order calculus, $2\boldsymbol{FOL}$, is taken from [81].

Definition 2.17 (Temporal Relational Queries in $2\boldsymbol{FOL}$) *The two-sorted first-order calculus $2\boldsymbol{FOL}$ over a temporal database $\langle \mathbf{T}, \leq; \mathbf{D}, =; r_1, \ldots, r_m \rangle$ is defined by the following BNF rule:*

$$
\begin{array}{rcl}
\mathrm{M} & ::= & r(t_i, x_1, \ldots, x_k) \\
& | & t_i < t_j \\
& | & x_i = x_j \\
& | & \mathrm{M} \wedge \mathrm{M} \\
& | & \neg \mathrm{M} \\
& | & \exists t_i . \mathrm{M} \\
& | & \exists x_i . \mathrm{M},
\end{array}
$$

where t_i is called a temporal variable and x_i a data variable. A $2\boldsymbol{FOL}$ temporal query *is an* M *formula with exactly one free temporal variable.*

The database schema in the temporal structure is monadic with respect to the temporal sort $\mathbf{T}$. In other words, the predicate symbols in the database schema always have exactly one temporal argument. Therefore, a *valid* temporal query has to be restricted to have exactly one free temporal variable so that the closure of the query language is preserved, and the result of the query can be stored in the same temporal database. Alternatively, the restriction of one free temporal variable in any temporal query raises the problem that the $2\boldsymbol{FOL}$ query language is not *subquery-closed*. Thus, we cannot define all $2\boldsymbol{FOL}$ queries by combining simpler queries defined in the same query language.

Next, we compare ordered relational queries with temporal relational queries, and show that any $2\boldsymbol{FOL}$ temporal query can be reduced to an $\boldsymbol{FO}_{\preceq}$ ordered query. More precisely, we want to prove that there exists a one-to-one mapping function θ, which maps any temporal query ϕ on r to an ordered relational query $\langle \varphi, \psi \rangle$ such

that for any ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle \in \rho(r)$,

$$\langle \varphi, \psi \rangle \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle \in \rho(\phi(r)).$$

This is illustrated in Figure 2.1.

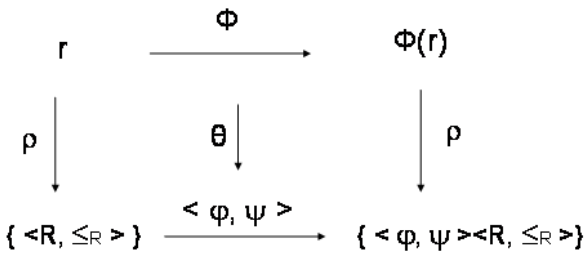

Figure 2.1: Reduction of a temporal query to an ordered query

The intuition of the proof is that we construct the mapping ordered relational query $\langle \varphi, \psi \rangle$ for each temporal query ϕ such that for any input temporal relation r,

- each $\boldsymbol{t}$-variable in ϕ corresponds to a separation tuple;
- the order among separation tuples is exactly the order of time instants in ϕ;
- the individual tuple (t, x) corresponds the data tuple x in r;
- for cluster-equivalent input ordered relations, the outputs should also be cluster-equivalent.

Before proving the following lemma, we must make assumptions on the orders inside clusters for both input and output ordered relations.

Assumption 2.18 (1) In the case that a temporal operation has only one argument, and (t_1, i) and (t_2, j) are two tuples in the result, we assume that the tuple $F(t_1, i)$ is prior to the tuple $F(t_2, j)$ if $t_1 = t_2$ and $i < j$ hold in the argument temporal relation.

(2) In the case that a temporal operation has more than one argument, and (t_1, i) and (t_2, j) are two tuples in the result which are from different inputs, we assume that the order inside clusters in the result is identical to the order inside clusters in the universal temporal relation. All temporal relations can be encoded into one universal temporal relation by lexicographical order of time instant and the physical order of tuples.

Lemma 2.19 *Let r be an arbitrary temporal relation, and $\{\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle\}$ be a set of ordered relations. Assume that these exists a one-to-many correspondence ρ from temporal relations to ordered relations such that $\rho(r) = \{\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle\}$.*

Then, there exists a one-to-one mapping function θ from $2\boldsymbol{FOL}$ temporal queries to $\boldsymbol{FO}_{\preceq}$ ordered queries, such that for an arbitrary $2\boldsymbol{FOL}$ temporal query ϕ over the temporal database $\langle \mathbf{T}, \leq; \mathbf{D}, =; r\rangle$, there exists an ordered relational query $\theta(\phi)$ such that

$$\theta(\phi)\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle \in \rho(\phi(r)),$$

for any ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle \in \rho(r)$.

Proof: Let ϕ be any temporal query in $2\boldsymbol{FOL}$, and r be any temporal relation. Without lost of generality, we assume that the temporal relation r has only one temporal attribute and one date attribute. Let $\langle \varphi, \psi\rangle$ be the ordered relational query such that $\langle \varphi, \psi\rangle = \theta(\phi)$. We need to prove that for any ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle \in \rho(r)$,

$$\langle \varphi, \psi\rangle\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle \in \ \rho(\phi(r)).$$

By Definition 2.17, ϕ is a first-order formula with exactly one free temporal variable, defined by the BNF rule:

$$\phi \ ::= \ r(t, x_1, \ldots, x_n) \mid t_1 < t_2 \mid x_i = x_j \mid \phi \wedge \phi \mid \neg\phi \mid \exists t_i.\phi \mid \exists x_i.\phi.$$

We construct the mapping ordered relational query $\langle \varphi, \psi\rangle = \theta(\phi)$ on $\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle$ by induction on the structure of the temporal query ϕ.

First, we construct the data query φ of the mapping ordered query $\langle \varphi, \psi\rangle$. There are two steps for the construction of $\varphi(t, x)$.

Step 1: We need to construct a function $\bar{\varphi}(t^0, t, x)$ according to $\phi(t, x)$, such that $\bar{\varphi}(t^0, t, x)$ satisfies the following condition: $\phi(t, x)$ implies $\bar{\varphi}(t^0, t, x)$, where t^0 is the seperation tuple of t.

Let ϕ' be any subformula of ϕ, and in the case that ϕ' has the form of

Base case:

(1) $\phi' = r(t, x)$:

The mapping data subformula φ' is constructed as

$$\varphi' \ = \ R(t, x).$$

(2) $\phi' = t_1 = t_2$:

The mapping data subformula is constructed as

$$\begin{aligned}\varphi' \;=\;& R(t_1^0, a^0) \wedge\; t_1^0 \preceq_{\mathrm{R}} t_1 \\ & \wedge \neg\, \exists\, t_1^1.R(t_1^1, a^0) \wedge\; t_1^0 \preceq_{\mathrm{R}} t_1^1 \wedge\; t_1^1 \preceq_{\mathrm{R}} t_1 \\ & \wedge\, R(t_2^0, a^0) \wedge\; t_2^0 \preceq_{\mathrm{R}} t_2 \\ & \wedge \neg\, \exists\, t_2^1.R(t_2^1, a^0) \wedge\; t_2^0 \preceq_{\mathrm{R}} t_2^1 \wedge\; t_2^1 \preceq_{\mathrm{R}} t_2 \\ & \wedge\, t_1^0 = t_2^0.\end{aligned}$$

(3) $\phi' = t_1 < t_2$:

The mapping data subformula is constructed as

$$\begin{aligned}\varphi' \;=\;& R(t_1^0, a^0) \wedge\; t_1^0 \preceq_{\mathrm{R}} t_1 \\ & \wedge \neg\, \exists\, t_1^1.R(t_1^1, a^0) \wedge\; t_1^0 \preceq_{\mathrm{R}} t_1^1 \wedge\; t_1^1 \preceq_{\mathrm{R}} t_1 \\ & \wedge\, R(t_2^0, a^0) \wedge\; t_2^0 \preceq_{\mathrm{R}} t_2 \\ & \wedge \neg\, \exists\, t_2^1.R(t_2^1, a^0) \wedge\; t_2^0 \preceq_{\mathrm{R}} t_2^1 \wedge\; t_2^1 \preceq_{\mathrm{R}} t_2 \\ & \wedge\, t_1^0 \preceq_{\mathrm{R}} t_2^0 \wedge\; \neg t_1^0 = t_2^0.\end{aligned}$$

(4) $\phi' = x_1 = x_2$:

The mapping data subformula φ' is constructed as

$$\varphi' \;=\; x_1 = x_2 \vee\; x_1 = a^0.$$

Induction Step: Suppose that for simpler subformulas ϕ_1 and ϕ_2, the mapping data subformulas are φ_1 and φ_2, respectively.

(1) $\phi' = \phi_1 \wedge \phi_2$:

The mapping data subformula φ' is constructed as

$$\varphi' = \varphi_1 \wedge \varphi_2.$$

(2) $\phi' = \neg\phi_1$:

The mapping data subformula φ' is constructed as

$$\varphi' = \neg\varphi_1.$$

(3) $\phi' = \exists x_i.\ \phi_1$:

The mapping data subformula φ' is constructed as

$$\varphi' = \exists x_i.\ \varphi_1.$$

(4) $\phi' = \exists t_1.\ \phi_1$:

The mapping data subformula φ' is constructed as

$$\begin{aligned}\varphi' \ = \ & \exists t^0.R(t_1^0, a^0) \wedge \exists t_1.t_1^0 \preceq_{\mathrm{R}} t_1 \\ & \wedge \neg\ \exists\ t_1^1.R(t_1^1, a^0) \wedge\ t_1^0 \preceq_{\mathrm{R}} t_1^1 \wedge\ t_1^1 \preceq_{\mathrm{R}} t_1 \\ & \wedge\ \varphi_1.\end{aligned}$$

Step 2: Construct the data query $\varphi(t, x)$ from $\bar{\varphi}(t^0, t, x)$,

$$\varphi(t, x) \ = \ \exists t'.\bar{\varphi}(t', t, x) \vee \exists t'.\bar{\varphi}(t, t', x).$$

We now construct the order query ψ of the mapping ordered query $\langle \varphi, \psi \rangle$.

$$\begin{aligned}\psi(t_1, t_2) \ = \ & (\exists\ t_1', x_1.\bar{\varphi}(t_1, t_1', x_1) \wedge \exists\ t_2', x_2.\bar{\varphi}(t_2, t_2', x_2) \wedge t_1 \preceq_{\mathrm{R}} t_2) \\ & \vee \\ & (\exists\ x.\bar{\varphi}(t_1, t_2, x)) \\ & \vee \\ & (\exists\ t^0.(\exists\ x_1.\bar{\varphi}(t^0, t_1, x_1) \wedge \exists\ x_2.\bar{\varphi}(t^0, t_2, x_2)) \wedge t_1 \preceq_{\mathrm{R}} t_2) \\ & \vee \\ & (\exists\ t_1^0, x_1.\bar{\varphi}(t_1^0, t_1, x_1) \wedge \exists\ t_2^0, x_2.\bar{\varphi}(t_2^0, t_2, x_2)) \wedge t_1^0 \preceq_{\mathrm{R}} t_2^0).\end{aligned}$$

In the construction of the order query ψ, if the temporal query ϕ has only one argument r, then $\preceq_{\mathrm{R}}$ is the order of the corresponding ordered relation of r; if the temporal query ϕ has more than one argument, then $\preceq_{\mathrm{R}}$ is the order of the corresponding ordered relation of the universal temporal relation.

For an arbitrary temporal query ϕ, we have constructed an ordered query $\langle \varphi, \psi \rangle$ such that $\theta(\phi) = \langle \varphi, \psi \rangle$. Next, we need to prove that for the temporal relation r, and each $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle \in\ \rho(r)$, $\langle \varphi, \psi \rangle \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle \in\ \rho(\phi(r))$.

First, we prove that the data relation of $\theta(\phi)\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ is exactly mapping to the data tuples in the result of $\phi(r)$. This is proved by induction on the structure of the temporal query ϕ. For any subformula of ϕ, ϕ', in case of

Base case:

(1) $\phi' = r(t, x)$:

By the construction rule of φ', $\bar{\varphi}(t^0, t, x)$ returns all (t, x)s satisfying $\mathrm{R}(t, x)$, along with their separation tuples. And $\varphi(t, x)$ returns exactly all tuples in R, including separation tuples. We see that (t, x) satisfies $\varphi(t, x)$ if and only if (t, x) satisfies ϕ'.

(2) $\phi' = t_1 = t_2$:

By the construction of φ', φ' produces any pair of tuples, t_1 and t_2, which have the same separation tuples. Therefore, the tuples that satisfy φ' are exactly the same as the mapping tuples of the result of ϕ'.

(3) $\phi' = t_1 < t_2$:

By the construction of $\langle \varphi', \psi' \rangle$, φ' produces any pair of tuples, t_1 and t_2, whose separation tuples t_1^0 and t_2^0 satisfy $t_1^0 \preceq_{\mathrm{R}} t_2^0$ and $t_1^0 \neq t_2^0$. Therefore, the tuples satisfying φ' are exactly the same as the mapping tuples of the result of ϕ'.

(4) $\phi' = x_i = x_j$:

By the construction of $\langle \varphi', \psi' \rangle$, φ' selects both the tuples satisfying $x_i = x_j$ and their separation tuple. Then, the tuples satisfying φ' are exactly the same as the mapping data relation to the result of ϕ'.

Induction Step: We suppose that for simpler 2***FOL*** formulas ϕ_1 and ϕ_2 in the temporal query ϕ, the data tuples of the mapping ordered queries $\langle \varphi_1, \psi_1 \rangle$ and $\langle \varphi_2, \psi_2 \rangle$ are exactly the same as the mapping of outputs of ϕ_1 and ϕ_2. In case that ϕ' has the form of:

(1) $\phi' = \phi_1 \wedge \phi_2$:

By the induction hypothesis, the results of φ_1 and φ_2 are the mapping data relations of ϕ_1 and ϕ_2, respectively. A tuple is in the result of φ_i if and only if the mapping tuple is in the result of ϕ_i. Therefore, a tuple is in the result of $\phi_1 \wedge \phi_2$ if and only if the mapped tuple is in the result of $\varphi_1 \wedge \varphi_2$. That is, the result of $\varphi_1 \wedge \varphi_2$ is exactly the same as the mapping data relation of $\phi_1 \wedge \phi_2$.

(2) $\phi' = \neg\phi_1$:

If t satisfies ϕ' in the temporal relation r, by the construction rule of φ', φ' generates the tuples whom φ_1 does not hold on any tuple in the same cluster. By the induction hypothesis, tuples satisfying φ_1 are exactly the same as the mapping tuples of the result of ϕ_1. Hence, the tuples in R' satisfying φ' are exactly the same as the mapping of the result of $\phi'(r)$.

(3) $\phi' = \exists x_i.\ \phi_1$:

If t satisfies ϕ' in the temporal relation r, by the construction rule of φ', φ' generates the projection of the tuples satisfying φ_1. By the induction hypothesis, tuples satisfying φ_1 are exactly the same as the mapping tuples

of the result of ϕ_1. Hence, the tuples in R' satisfying φ' are exactly the mapping of the result of $\phi'(r)$.

(4) $\phi' = \exists t'.\ \phi_1$:
If t satisfies ϕ' in the temporal relation r, by the construction rule of φ', φ' generates the tuples so that there is at least one cluster such that those tuples satisfy φ_1. By the induction hypothesis, tuples satisfying φ_1 are exactly the same as the mapping tuples of the result of ϕ_1. Hence, the tuples in R' satisfying φ' are exactly the same as the mapping of the result of $\phi'(r)$.

Then, we prove that the order relation of $\theta(\phi)$ is exactly mapping to the order generated by ϕ. By the construction of ψ, t_1 is prior to t_2 in the output if and only if they are in the output data relations and satisfy one of the following conditions:

(1) both t_1 and t_2 are separation tuples, and satisfy $t_1 \preceq_{\mathrm{R}} t_2$;

(2) t_1 is the separation tuple of t_2 in the output;

(3) t_1 and t_2 are in the same cluster in the input, and satisfy $t_1 \preceq_{\mathrm{R}} t_2$;

(4) t_1 and t_2 are in different clusters in the input, and satisfy $t_1^0 \preceq_{\mathrm{R}} t_2^0$.

Particularly, (1) corresponds to the case that the order of time instants is not changed by the temporal query. Therefore, the construction of ψ defines the order among clusters exactly as the order of time instants; (2) reflects that the separation tuple of a cluster is prior to tuples of cluster in the output; (3) shows that the orders inside clusters keep their original orders from the input; (4) states that the order between any tuples, from different clusters, is decided by the order of their clusters.

Therefore, the constructed order query ψ is exactly the mapping order to the order of time instants in the result of ϕ. This finishes the proof.

□

In the proof of Lemma 2.19, the mapping ordered relational query contains the clause $t_1 \preceq_{\mathrm{R}} t_2$ only in two cases: (1) when both t_1 and t_2 are separation tuples; (2) when t_1 is a separation tuple and t_2 is a regular tuple in the same cluster. Intuitively, in case (1), $t_1 \preceq_{\mathrm{R}} t_2$ is used to represent the order between the clusters with t_1 and t_2 as separation tuples, and this order between clusters exactly reflects the order between time instants in the mapped temporal query; in case (2), $t_1 \preceq_{\mathrm{R}} t_2$ is used to restrict the regular tuple t_2 to the cluster with the separation tuple t_1.

Furthermore, all $\boldsymbol{t}$-quantifiers in the temporal query are mapped to quantifiers on variables of separation tuples while $\boldsymbol{x}$-quantifiers in a temporal query are mapped to quantifiers on variables of regular tuples (t, x) in the ordered relational query.

All together, the mapping ordered query is a $\boldsymbol{FO}_{\preceq}$ ordered query whose order of clusters (represented by separation tuples) is identical to the order of time instants in the temporal query, and the orders inside clusters remain in the original order from the temporal query. Such a set of ordered relational queries is a subset of ordered relational queries; therefore, $2\boldsymbol{FOL}$ temporal relational queries can be reduced to a subset of $\boldsymbol{FO}_{\preceq}$ ordered relational queries. This reduction from $2\boldsymbol{FOL}$ temporal relational queries to $\boldsymbol{FO}_{\preceq}$ ordered relational queries leads to the conjecture of incompleteness of ordered relational algebras in the following section.

2.3 Ordered Relational Algebras

The use of algebraic operators provides a distinctly different perspective on relational queries. This section first provides a formal definition of ordered relational algebra and then discusses the expressive power of ordered relational algebras. Particularly, it focuses on the completeness problem of ordered relational algebras.

2.3.1 Ordered Relational Algebras

There are many alternative ways of defining an ordered relational algebra. A proper ordered relational algebra should have certain desirable properties:

(1) To take advantage of the well-studied theories of relational algebra, ordered relational algebra should be a natural extension to the conventional relational algebra. Thus, all relational queries in algebraic expressions in the conventional relational algebra can be expressed by algebraic operators in the ordered relational algebra.

(2) The set of operators should be minimal, *i.e.*, the function of each operator should be orthogonal, such that there is no redundancy in terms of operator functions.

The formal definition of an ordered relational algebra is given as follows.

Definition 2.20 (Ordered Algebraic Operators)
An ordered algebraic operator is a valid ordered relational query in $\boldsymbol{FO}_{\preceq}$.

Definition 2.21 (Ordered Relational Algebra)
An ordered relational algebra is a finite set of ordered algebraic operators.

Many potential sets of ordered operators can be defined over ordered relations. A set of ordered operators should include those operators that are taken from the conventional relational algebra and adapted to the ordered databases. The typical set of those ordered operators are: order-preserving selection, order-preserving projection, nested-loop Cartesian products, set difference, and union.

The following example lists several of the ordered operators derived from conventional relational algebras. This is only one possible way, among many, to define these operators, and there are many alternatives to define them based on different data and order specifications.

Example 2.22 Given ordered relations $\langle R, \preceq_R \rangle$, $\langle R_1, \preceq_{R_1} \rangle$ and $\langle R_2, \preceq_{R_2} \rangle$ with schemas $(\mathrm{T}, X_1, \cdots, X_n)$, $(\mathrm{T}, X_1^1, \cdots, X_n^1)$, and $(\mathrm{T}, X_1^2, \cdots, X_m^2)$, respectively.

(i) Order-preserving selection.

$$\begin{aligned}\sigma_p\langle R, \preceq_R \rangle \;\; = \langle\; & \{(t, x_1, \cdots, x_n) \mid R(t, x_1, \cdots, x_n) \wedge p(t, x_1, \cdots, x_n)\}, \\ & \{(t_1, t_2) \mid \exists x_{11}, ..., x_{1n}.R(t_1, x_{11}, ..., x_{1n}) \wedge p(t_1, x_{11}, ..., x_{1n}) \\ & \wedge \exists x_{21}, ..., x_{2n}.R(t_2, x_{21}, ..., x_{2n}) \wedge \\ & p(t_2, x_{21}, ..., x_{2n}) \wedge t_1 \preceq_R t_2\} \;\rangle,\end{aligned}$$

where p may be in form of $X_k = x$ or $X_k = X_l$.

(ii) Order-preserving projection.

$$\begin{aligned}\pi_{X_1', \cdots, X_k'}\langle R, \preceq_R \rangle \;\; = \langle\; & \{(t, x_1, \cdots, x_k) \mid \exists\, x_{k+1}, \cdots, x_n.R(t, x_1, \cdots, x_n)\}, \\ & \{(t_1, t_2) \mid t_1 \preceq_R t_2\} \;\rangle.\end{aligned}$$

where $\{X_1', \cdots, X_k'\} \subseteq \{X_1, \cdots, X_n\}$. Without loss of generality, we assume that $\{X_1', \cdots, X_k'\}$ are first k attributes in $\{X_1, \cdots, X_n\}$.

(*iii*) Nested-loop cross product.

$$\begin{aligned}\langle R_1, \preceq_{R_1}\rangle \times \langle R_2, \preceq_{R_2}\rangle = \ & \langle\{(t, \bar{x}_1, \bar{x}_2) \mid \ \exists t_1, t_2.t = (t_1, t_2) \wedge \\ & \qquad \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2)\}, \\ & \{(t_1, t_2) \mid \ \exists t_{11}, t_{12}.t_1 = (t_{11}, t_{12}) \\ & \qquad \wedge \exists t_{21}, t_{22}.t_2 = (t_{21}, t_{22}) \\ & \qquad \wedge \exists \bar{x}_{11}.\mathrm{R}_1(t_{11}, \bar{x}_{11}) \\ & \qquad \wedge \exists \bar{x}_{12}.\mathrm{R}_2(t_{12}, \bar{x}_{12}) \\ & \qquad \wedge \exists \bar{x}_{21}.\mathrm{R}_1(t_{21}, \bar{x}_{21}) \\ & \qquad \wedge \exists \bar{x}_{22}.\mathrm{R}_2(t_{22}, \bar{x}_{22}) \\ & \qquad \wedge (t_{11} \preceq_R t_{21} \\ & \qquad \vee (t_{11} = t_{21} \wedge t_{12} \preceq_R t_{22}))\}\rangle.\end{aligned}$$

Note that in this example, the definition of an ordered operation also defines the way that new tuple identifiers in the output are generated. For example, the order-preserving selection and projection keep the original tuple identifiers of the input as the output tuple identifiers; the nested-loop product operation specifies output tuple identifiers as combinations of input tuple identifiers.

To achieve our purpose of capturing any potential order within query plans, we must define more operators which are exclusive to ordered relational algebras. Several of such ordered operations are described in the following example:

Example 2.23 (*i*) The top operator, which retrieves the first tuple from $\langle R, \preceq_R\rangle$.

$$\begin{aligned}Top(\langle R, \preceq_R\rangle) \ = \langle \ & \{(t, x_1, \cdots, x_n) \mid R(t, x_1, \cdots, x_n) \\ & \wedge \neg \exists\ t_1.\ t_1 \preceq_R t \wedge \neg t_1 = t\}, \\ & \{(t', t'') \mid \exists \bar{x}'.\mathrm{R}(t', \bar{x}') \ \wedge \ \exists \bar{x}''.\mathrm{R}(t'', \bar{x}'') \\ & \wedge \neg \exists\ t_1'.\ t_1' \preceq_R t' \wedge t_1' \neq t' \\ & \wedge \neg \exists\ t_1''.\ t_1'' \preceq_R t'' \wedge \ t_1'' \neq t'' \\ & \wedge \ \preceq_R (t', t'')\} \ \rangle.\end{aligned}$$

(*ii*) The reverse operator, which reverses the order of $\langle R, \preceq_R\rangle$.

$$\begin{aligned}Revs\langle R, \preceq_R\rangle \ = \langle \ & \{(t, x_1, \cdots, x_n) \mid R(t, x_1, \cdots, x_n)\}, \\ & \{(t_1, t_2) \mid \exists x_{11}, ..., x_{1n}.R(t_1, x_{11}, ..., x_{1n}) \\ & \quad \wedge \exists x_{21}, ..., x_{2n}.R(t_2, x_{21}, ..., x_{2n}) \wedge \neg\ t_1 \preceq_R t_2\}\rangle.\end{aligned}$$

Note that in this example, "Reverse" is now a first-order expressible algebraic operator. However, it is not definable in conventional relational algebras as it does not change the input relation as a set.

The following example demonstrates how to compose the algebraic expressions for $\boldsymbol{FO}_{\preceq}$ ordered relational queries.

Example 2.24 By using the ordered operators in the above examples, we can express the two ordered relational queries in Example 2.16. The first query can be formulated as

$$\sigma_{\texttt{Department}=\text{“}Sales\text{”}}\langle \texttt{EMP}, \preceq_{\texttt{EMP}}\rangle.$$

The second query can be rewritten as

$$Revs(\sigma_{\texttt{Salary}=55{,}000}\langle \texttt{EMP}, \preceq_{\texttt{EMP}}\rangle).$$

2.3.2 The Completeness Problem of Ordered Relational Algebras

The expressive power of various query languages is one of the topics in database theory that have been most deeply explored. In conventional relational databases, relational calculus is used as the standard to evaluate the expressive power of query languages [31]. A query language which has the same expressive power as relational calculus is called relationally complete.

Analogously, in ordered database theory, we use the two-sorted first-order logic $\boldsymbol{FO}_{\preceq}$ as a metric in evaluating the expressive power of other ordered relational query languages. From the expressibility perspective, we are most interested in the *completeness problem* of ordered relational algebras. Given a domain of tuple identifiers, the completeness problem poses the following question:

> Does there exist an ordered relational algebra equipped with a finite set of ordered operations such that, given an arbitrary ordered relational query in $\boldsymbol{FO}_{\preceq}$, there is an equivalent query expressed by a finite sequence of ordered operations from the algebra?

We say that an ordered relational algebra is $\boldsymbol{FO}_{\preceq}$ *complete* if any $\boldsymbol{FO}_{\preceq}$ expressible ordered relational query can be expressed by a finite sequence of ordered operations in this ordered algebra.

The completeness problem is essential to ordered relational databases because the incompleteness of ordered relational algebras causes serious problems when

implementing ordered query processing. Typical implementations in conventional databases are based on the equivalence of relational algebra and first-order calculus [5] [40] [27].

Before discussing the completeness of the class of ordered relational algebras with respect to the two-sorted first-order logic $FO_{\preceq}$, we review the existing results of the completeness of temporal relational algebras in temporal database theory. It is well known that for monadic first-order logic over linear orders, there is an expressively equivalent propositional temporal logic with an expressively complete set of temporal operators [52, 38, 50]. However, this result cannot be generalized to two-sorted first-order logic.

In [53], it was proved that the two-sorted first-order language 2***FOL*** is strictly more expressive than the first-order temporal logic with the connectives `since` and `until` for dense linearly ordered time. Abiteboul *et al.* in [4] show that this result holds for any finite linear temporal domains. A more general result is shown in [81], [10], and [80] that any first-order temporal calculus with an arbitrary finite set of temporal connectives is strictly less expressive than the two-sorted first-order calculus 2***FOL*** over linear temporal domains.

The separation of first-order temporal calculi from two-sorted first-order calculus causes a failure of attempts to develop expressively complete temporal relational algebras that are subquery-closed [29, 11]. Various temporal algebras have been proposed [83]; however, none of them satisfies both expressive completeness and subquery closure requirements simultaneously.

With regard to the completeness problem of ordered relational algebra, the answer is most likely to be negative. By Lemma 2.19, 2***FOL*** temporal relational queries can be reduced to a subset of $\boldsymbol{FO}_{\preceq}$ ordered relational queries. If we assume that there is a complete ordered relational algebra for $\boldsymbol{FO}_{\preceq}$ ordered relational queries, then there is also a finite set of operators to express all possible 2***FOL*** temporal relational queries. However, this is a contradiction to the incompleteness of temporal relational algebras. The conjecture of the incompleteness of ordered relational algebras is as follows:

> There does not exist an ordered relational algebra equipped with a finite set of ordered operations such that, given an arbitrary ordered relational query in $\boldsymbol{FO}_{\preceq}$, there is an equivalent query expressed by a finite sequence of ordered operations from the algebra.

The possible incompleteness of general ordered relational algebras motivates us to investigate the following question:

> For any sub-class of first-order ordered relational queries, does there exist an ordered algebra such that any ordered queries in this sub-class can be rewritten with a finite sequence of algebraic operations?

We explore this perspective in the following chapters.

In this chapter, we formally defined ordered relational queries using both two-sorted first-order logic $\boldsymbol{FO}_{\preceq}$ and ordered relational algebra. These two paradigms shares common properties as query representations. First, they both are set-based in the sense that they identify and manipulate sets of tuples once at a time, rather than identifying and manipulating individual tuples. Second, they are both abstract so that they are independent from the physical implementation of operations and data storage.

However, the two paradigms are distinguished from each other by their essential differences. The two-sorted first-order logic is conceptually declarative, *i.e.*, the output ordered relation is specified by properties that the data and order relations satisfy respectively; the ordered relational algebra is conceptually procedural, as ordered queries are represented by a sequence of ordered operations that constructs the answer to the query.

In comparison to 2***FOL*** temporal relational queries, $\boldsymbol{FO}_{\preceq}$ ordered relational queries are expected to lack complete relational algebras. Therefore, beginning with the next chapter, we explore the completeness problem of ordered conjunctive queries, which is the most important class of ordered relational queries.

Chapter 3

Ordered Conjunctive Queries with Data-Decided Order $\boldsymbol{CQ}^{X}_{\preceq}$

This and the following chapters focus on exploring conjunctive queries in ordered context. In general, ordered conjunctive queries are those ordered relational queries whose data queries are conjunctive queries. As in conventional relational databases, conjunctive queries are the most common class of relational queries in ordered databases.

We study the ordered conjunctive query $\boldsymbol{CQ}_{\preceq}$ in a gradual way: first, we begin with the ordered conjunctive query with data-decided (or $\boldsymbol{x}$-decided) order, $\boldsymbol{CQ}^{X}_{\preceq}$; then, we examine the ordered conjunctive query with $\boldsymbol{t}$-decided order, $\boldsymbol{CQ}^{T}_{\preceq}$; finally, we investigate the general ordered conjunctive query, $\boldsymbol{CQ}_{\preceq}$. The goal is to study the expressive powers of various query representation languages for ordered conjunctive queries. In particular, we are interested in the completeness problem of ordered conjunctive query: do there exist complete algebras for the three classes of ordered conjunctive queries?

Temporal relational algebras fail in completeness because quantifiers on $\boldsymbol{x}$-variables and $\boldsymbol{t}$-variables are allowed to be interleaved in 2$\boldsymbol{FOL}$ specifications of temporal relational queries; to develop potentially complete algebras for $\boldsymbol{CQ}^{X}_{\preceq}$, we have to impose a restriction on $\boldsymbol{CQ}_{\preceq}$ to avoid this situation.

Definition 3.1 (Bounded x-variables) *An* $\boldsymbol{x}$*-variable* x_i *is* bounded *in a first-order formula* ψ *if each occurrence of* x_i *in* ψ *is only in forms of* $\exists\, x_i.\mathrm{R}(t, \bar{x})$ *or* $\exists\, x_i.\mathrm{R}(t, \bar{x}) \wedge \psi'$ *at the leaf level of* ψ*, where* ψ' *is a boolean combination of predicates on* $\boldsymbol{x}$*-variables.*

Example 3.2 Consider an ordered database $\langle \mathbf{Dom}_{\mathrm{T}}, =; \mathbf{Dom}_{\mathrm{X}}, =; \mathbf{R} \rangle$ with one relation $\mathbf{R}(T, X)$. A $\boldsymbol{FO}_{\preceq}$ formula $\exists t_1, \exists x_1.\mathbf{R}(t_1, x_1) \wedge\ t \preceq_R t_1 \wedge \neg t = t_1$ generates a set of tuples t which could be any tuple in $\mathbf{R}$ except the last tuple. The variable x_1 is bounded because it appears in the form of $\exists x_1.\mathbf{R}(t_1, x_1)$.

Please not that, by Definition 3.1, no quantifier on $\boldsymbol{t}$-variables or quantifier on $\boldsymbol{x}$-variables exists in the scope of the bounded $\boldsymbol{x}$-variable, which ensures that quantifiers on $\boldsymbol{x}$-variables and $\boldsymbol{t}$-variables are not interleaved in the formula ψ; there is exactly one $\boldsymbol{t}$-variable in the scope of the bounded $\boldsymbol{x}$-variable, and thus the $\boldsymbol{x}$-variable can only be used to restrict the behavior of a particular $\boldsymbol{t}$-variable. Hence, putting this restriction on the order specification of ordered conjuctive queries yields the possibility of the existence of a complete algebra for ordered conjunctive queries.

In this chapter, we limit our scope of investigation to ordered conjunctive queries with $\boldsymbol{x}$-decided order, $\boldsymbol{CQ}^X_{\preceq}$. Section 3.1 formally defines $\boldsymbol{CQ}^X_{\preceq}$ in a two-sorted first-order calculus $\boldsymbol{FO}_{\preceq}$. An ordered algebra $\boldsymbol{CA}^X_{\preceq}$ is proposed for $\boldsymbol{CQ}^X_{\preceq}$ in Section 3.2, and transformation rules are provided and proven for ordered operations in $\boldsymbol{CA}^X_{\preceq}$ in Section 3.3. Section 3.4 proves that $\boldsymbol{CA}^X_{\preceq}$ is a complete algebra of $\boldsymbol{CQ}^X_{\preceq}$.

3.1 Ordered Conjunctive Queries $\boldsymbol{CQ}^X_{\preceq}$

The ordered conjunctive query $\boldsymbol{CQ}^X_{\preceq}$ is a subset of the general ordered conjunctive query $\boldsymbol{CQ}_{\preceq}$; quantifiers on $\boldsymbol{t}$-variables are forbidden in order specification (subformula ψ') of a $\boldsymbol{CQ}_{\preceq}$ ordered conjunctive query. A formal definition of the ordered conjunctive query $\boldsymbol{CQ}^X_{\preceq}$ follows.

Definition 3.3 (Ordered Conjunctive Query $\boldsymbol{CQ}^X_{\preceq}$)
A $\boldsymbol{CQ}^X_{\preceq}$ query $\langle \varphi, \psi \rangle$ on ordered relations $\mathbf{R}_1, ..., \mathbf{R}_n$ is constructed as follows:

(*i*) *The data query φ is a conjunctive query and defined by the following BNF rule:*

$$
\begin{array}{rcl}
\varphi & ::= & R_i(t_i, x_1, \ldots, x_n) \\
& | & t = (t_i, t_j) \\
& | & x_i = x_j \\
& | & \varphi \wedge \varphi \\
& | & \exists x_i.\varphi \\
& | & \exists t_i.\varphi
\end{array}
$$

(*ii*) *The order query* ψ *is a first-order formula in the form of*

$$\begin{array}{rcl} \psi & ::= & R_i(t_i, x_1, \ldots, x_n) \\ & | & t = (t_i, t_j) \\ & | & x_i = x_j \\ & | & \psi \wedge \psi \\ & | & \exists x_i.\psi \\ & | & \exists t_i.\psi \\ & | & \psi \wedge \psi' \end{array}$$

where ψ' *is called order specification, defined by the BNF rule*

$$\begin{array}{rcl} \psi' & ::= & t_i \preceq_R t_j \\ & | & x_i = x_j \\ & | & \psi' \wedge \psi' \\ & | & \neg\psi' \end{array}$$

where the order query ψ *has exactly two* $\boldsymbol{t}$*-variables as the only free variables, and the order specification* ψ' *has all* $\boldsymbol{x}$*-variables bounded.*

In the definitions of φ and ψ, the clause $t = (t_i, t_j)$ is a constructor of tuple identifiers. This constructor is used to form the output tuple identifiers when the output tuple identifiers are composed from multiple input tuple identifiers. In the case that the output tuple identifers are constructed from k input ordered relations, the combinations of tuple identifiers $(...(t_{i_1}, t_{i_2}), ..., t_{i_k})$ are abbreviated as $(t_{i_1}, t_{i_2}, ..., t_{i_k})$ in later development.

Intuitively, a $\boldsymbol{CQ}^X_{\preceq}$ query on ordered relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$ has the following form:

$$\begin{array}{ll} \langle\{(t, \bar{x}) \mid & \exists t_1, \ldots, t_n.\ t = (t_1, \ldots, t_n) \wedge\ \varphi(t_1, \ldots, t_n, \bar{x})\}, \\ \{(t_1, t_2) \mid & \exists t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots \bar{x}_{2n}. \\ & t_1 = (t_{11}, \ldots, t_{1n}) \wedge\ t_2 = (t_{21}, \ldots, t_{2n}) \\ & \wedge\ \varphi(t_{11}, \ldots, t_{1n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}) \wedge\ \varphi(t_{21}, \ldots, t_{2n}, \bar{x}_{21}, \ldots, \bar{x}_{2n}) \\ & \wedge\ \psi'(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n})\}\ \rangle, \end{array}$$

where ψ' is constructed by the BNF rule

$$\psi' ::= t_i \preceq_R t_j \mid x_i = x_j \mid \psi' \wedge \psi' \mid \neg\psi',$$

and t_1 and t_2 are the only two free variables in the order specification ψ. Since there is no quantifier on $\boldsymbol{t}$-variables in the order specification ψ', t_1 and t_2 are also

the only two $\boldsymbol{t}$-variables in the order specification ψ'. Hence, the order specification ψ' must have all $\boldsymbol{x}$-variable bounded.

Note that ψ has a two-level structure in the definition of $\boldsymbol{CQ}^X_\preceq$ ordered conjunctive query:

(1) The first level of ψ has the same construction rule as φ. This ensures that ψ defines an order for any pair of tuples t_1 and t_2 that satisfies the data query φ.

(2) The second level of ψ is the subformula ψ', which defines real ordering among the resulting tuples. Since there are no quantifiers on $\boldsymbol{t}$-variables in ψ', all $\boldsymbol{t}$-variables in ψ' appear only in atomic forms $t_{1i} \preceq t_{2i}$ or $\neg t_{1i} \preceq t_{2i}$, where t_{1i} and t_{2i} are originally from the same input ordered relation $\langle R_i, \preceq_{R_i}\rangle$, but are components of different tuples in the output. The construction rule at the second level indicates that the output order is mainly determined by data predicates $x_i = x_j$, which is why we name the ordered specification in Definition 3.3 as $\boldsymbol{CQ}^X_\preceq$. The superscript X suggests that the order is decided by predicates on $\boldsymbol{x}$-variables.

However, an ordered query specification which satisfies the syntax definition in Definition 3.3 may not be a valid ordered conjunctive query, unless it meets the semantic restriction of valid ordered relational query defined in Definition 2.15. In general, for any input ordered relations, the order query ψ of a valid ordered conjunctive query in $\boldsymbol{CQ}_\preceq$ always defines a total linear order on the tuples which satisfy φ.

Definition 3.4 (Valid Ordered Conjunctive Queries in $\boldsymbol{CQ}_\preceq$) *An ordered conjunctive query in $\boldsymbol{CQ}_\preceq$ is valid if it is a valid ordered relational query.*

The ordered conjunctive query $\boldsymbol{CQ}^X_\preceq$ is restricted in its expressive power when being compared with the general ordered conjunctive query $\boldsymbol{CQ}_\preceq$; however, it is still powerful enough to express a class of ordered conjunctive queries whose order depends on the attribute values, as illustrated by the example below:

Example 3.5 Consider the ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}}\rangle$ in Example 2.5, which conforms to an extended relation schema $S(\mathrm{T}, \texttt{EmpId}, \texttt{Name}, \texttt{Department}, \texttt{Salary})$. We show here two queries that can be expressed in $\boldsymbol{CQ}^X_\preceq$.

(1) Find all employees whose salary is \$65,000 per year, and output the names and departments in the original order. This query can be formulated in $\boldsymbol{CQ}^X_{\preceq}$ as follows:

$$\begin{array}{ll}\langle\{(t, x_2, x_3)| & \exists x_1, x_4.\ \texttt{EMP}\ (t, x_1, x_2, x_3, x_4) \wedge x_4 = 65000\},\\ \{(t_1, t_2)| & \exists x_{11}, x_{12}, x_{13}, x_{14}.\ \texttt{EMP}\ (t_1, x_{11}, x_{12}, x_{13}, x_{14}) \wedge x_{14} = 65000\\ & \wedge \exists x_{21}, x_{22}, x_{23}, x_{24}.\ \texttt{EMP}\ (t_2, x_{21}, x_{22}, x_{23}, x_{24}) \wedge x_{24} = 65000\\ & \wedge t_1 \preceq_{\texttt{EMP}} t_2\}\rangle.\end{array}$$

(2) Return a list of all employees in EMP in such an order that the employees in the department Sales are listed in the reverse order, and are moved to the bottom of the list; the rest of the employees are kept in the original order. This query also can be formulated in $\boldsymbol{CQ}^X_{\preceq}$:

$$\begin{array}{ll}\langle\{(t, x_1, x_2, x_3, x_4)| & \texttt{EMP}\ (t, x_1, x_2, x_3, x_4)\},\\ \{(t_1, t_2)| & \exists x_{11}, x_{12}, x_{13}, x_{14}.\ \texttt{EMP}\ (t_1, x_{11}, x_{12}, x_{13}, x_{14})\\ & \wedge \exists x_{21}, x_{22}, x_{23}, x_{24}.\ \texttt{EMP}\ (t_2, x_{21}, x_{22}, x_{23}, x_{24})\\ & \wedge\ ((x_{13} = \text{“}Sales\text{”} \wedge\ x_{23} = \text{“}Sales\text{”} \wedge\ \neg t_1 \preceq_{\texttt{EMP}} t_2)\\ & \vee\ (\neg x_{13} = \text{“}Sales\text{”} \wedge\ x_{23} = \text{“}Sales\text{”})\\ & \vee\ (\neg x_{13} = \text{“}Sales\text{”} \wedge\ \neg x_{23} = \text{“}Sales\text{”} \wedge\ t_1 \preceq_{\texttt{EMP}} t_2))\}\rangle.\end{array}$$

3.2 An Ordered Conjunctive Algebra $CA^X_{\preceq}$

In the previous section, we formally defined ordered conjunctive queries $\boldsymbol{CQ}^X_{\preceq}$. In this section, we propose an ordered algebra $\boldsymbol{CA}^X_{\preceq}$ for ordered conjunctive queries $\boldsymbol{CQ}^X_{\preceq}$, and then study properties and transformation rules of primitive and derived ordered operations in $\boldsymbol{CA}^X_{\preceq}$.

3.2.1 Overview of Ordered Algebra $CA^X_{\preceq}$

The use of algebra operators provides a distinctly different perspective on ordered conjunctive query representation. In general, a proper ordered algebra for ordered relational queries should meet several essential requirements. First, the proposed algebra should be closed; each ordered operator takes one or more ordered relations

as arguments, and produces an ordered relation as a result. Second, it is easy to perform query transformation on ordered queries. Third, it is convenient to trace and manipulate orders through operators. We develop an ordered algebra $\boldsymbol{CA}^X_\preceq$ for the ordered conjunctive query $\boldsymbol{CQ}^X_\preceq$, which satisfies these general requirements for ordered algebras.

In the ordered conjunctive algebra $\boldsymbol{CA}^X_\preceq$, certain ordered operators derive from the conventional relational operators, which are extended to the ordered context by adding order specifications. These operators include order-preserving selection, order-preserving projection, and left nested-loop product. Others are specifically developed for ordered relations, such as order concatenation, order reduction, order identity, and order reverse.

Both data and order specifications of an ordered operator are defined in the two-sorted first-order calculus $\boldsymbol{FO}_\preceq$. Essentially, each ordered operator in $\boldsymbol{CA}^X_\preceq$ is a valid ordered relational query in $\boldsymbol{CQ}_\preceq$. However, some of ordered operators in $\boldsymbol{CA}^X_\preceq$ are not defined in $\boldsymbol{CQ}^X_\preceq$ because they use negations in their data queries. The reason is that we have to include negations in the order specification of $\boldsymbol{CQ}^X_\preceq$ query; otherwise, $\boldsymbol{CQ}^X_\preceq$ will have a very limited expressible power to be able to express interesting orders. When negations are included in the order specifications, the data queries of ordered operations must have negations in order to obtain a complete set of ordered operators for $\boldsymbol{CQ}^X_\preceq$.

In addition, construction rules of tuple identifier are specified in the definition of each ordered operator. In the following development, we abbreviate $\{x_1, \ldots, x_n\}$ as $\bar{x}$ to simplify the notations. Similarly, $\{x_{i1}, \ldots, \boldsymbol{x}_{in}\}$ is abbreviated as $\bar{x}_i$.

3.2.2 Order-preserving Selection

The order-preserving selection operator $\sigma_\mathrm{p} : \mathbf{R}_\preceq \rightarrow \mathbf{R}_\preceq$ in $\boldsymbol{CA}^X_\preceq$ is a counterpart of the well-known selection operator in conventional relational algebras. The subscript p is a selection predicate p on $\boldsymbol{x}$-variables. The schema of the resulting ordered relation is the same as that of the argument ordered relation. The formal definition of order-preserving selection follows:

$$\begin{aligned}\sigma_\mathrm{p}(\mathbf{R}) = \; & \langle \{(t,\bar{x}) | \quad \mathrm{R}(t,\bar{x}) \wedge \mathrm{p}(t,\bar{x})\}, \\ & \{(t_1,t_2) | \quad \exists \bar{x}_1.\mathrm{R}(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \\ & \qquad\quad \exists \bar{x}_2.\mathrm{R}(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \wedge t_1 \preceq_R t_2\}\rangle.\end{aligned}$$

In this definition, the operator takes as an argument an ordered relation, and returns as a result tuples that satisfy the predicate p. The resulting order is the same as the original order in the argument. The tuple identifiers in the resulting ordered relation are identical to those in the argument relation. The definition of the order-preserving operator leads to the following lemma:

Lemma 3.6 *An order-preserving selection is a valid ordered relational query.*

Proof: Let $\sigma_{\mathrm{p}}(\mathbf{R}) = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$, where $\mathbf{R} = \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. Clearly, R′ and $\preceq_{\mathrm{R}'}$ are inclusively dependant on each other by the definition of the order-preserving selection. We need to prove that $\preceq_{\mathrm{R}'}$ is a linear order of R′. This can be accomplished by proving the three conditions in (*b*) of Definition 2.3.

(1) Let $t_1, t_2, t_3 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge\ (t_2, t_3) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\quad & (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
& \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \wedge t_1 \preceq_R t_2) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}(t_1, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \\
& \wedge \exists \bar{x}_3.\mathrm{R}(t_3, \bar{x}_3) \wedge \mathrm{p}(t_3, \bar{x}_3) \wedge t_2 \preceq_R t_3) \\
\Longrightarrow\quad & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
& \wedge \exists \bar{x}_3.\mathrm{R}(t_3, \bar{x}_3) \wedge \mathrm{p}(t_3, \bar{x}_3) \wedge t_1 \preceq_R t_3 \\
\Longrightarrow\quad & (t_1, t_3) \in \preceq_{\mathrm{R}'} .
\end{aligned}
$$

(2) Let $t_1, t_2 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge\ (t_2, t_1) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\quad & (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
& \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \wedge t_1 \preceq_R t_2) \\
& \wedge (\exists \bar{x}'_2.\mathrm{R}(t_2, \bar{x}'_2) \wedge \mathrm{p}(t_2, \bar{x}'_2) \\
& \wedge \exists \bar{x}'_1.\mathrm{R}(t_1, \bar{x}'_1) \wedge \mathrm{p}(t_1, \bar{x}'_1) \wedge t_2 \preceq_R t_1) \\
\Longrightarrow\quad & \bar{x}_1 = \bar{x}'_1 \wedge \bar{x}_2 = \bar{x}'_2 \wedge t_1 \preceq_R t_2 \wedge t_2 \preceq_R t_1 \\
\Longrightarrow\quad & t_1 = t_2
\end{aligned}
$$

(3) Let t_1 and t_2 be two arbitrary tuples in R′. By definition of the order-preserving selection,

$$
\begin{array}{ll}
 & t_1, t_2 \in \mathrm{R}'[\mathrm{T}] \\
\Longrightarrow & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
 & \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \\
\Longrightarrow & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
 & \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \\
 & \wedge (t_1 \preceq_{\mathrm{R}} t_2 \vee t_2 \preceq_{\mathrm{R}} t_1) \\
\Longrightarrow & (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
 & \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}} t_2) \\
 & \vee (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
 & \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \wedge t_2 \preceq_{\mathrm{R}} t_1) \\
\Longrightarrow & t_1 \preceq_{\mathrm{R}'} t_2 \vee t_2 \preceq_{\mathrm{R}'} t_1
\end{array}
$$

Thus, we have proved that the order-preserving selection is a valid ordered relational query. □

By definition, the data query of an order-preserving selection is a conjunctive query. Therefore, it is also a valid ordered conjunctive query in $\boldsymbol{CQ}^X_\preceq$.

3.2.3 Order-preserving Projection

The order-preserving projection operator $\pi_A : \mathbf{R}_\preceq \to \mathbf{R}_\preceq$ corresponds to its counterpart in conventional relational algebras. The subscript A is a list of projection attributes, $\mathrm{X}_1, \ldots, \mathrm{X}_k$ (repeats not permitted). The schema of the resulting relation, $(T, \mathrm{X}_1, \ldots, \mathrm{X}_k)$, is a subset of the schema of the argument relation. The formal definition of the order-preserving projection is shown as follows:

$$
\pi_A(\mathbf{R}) = \quad \langle \{(t, x_1, \ldots, x_k) \mid \exists x_{k+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n)\}, \preceq_{\mathrm{R}} \rangle.
$$

The order-preserving projection operator takes as an argument an ordered relation, and returns an ordered relation projected on given attributes. The output order is the same as the original order in the argument. The tuple identifiers in the resulting ordered relation are identical to those in the argument relation.

The order-preserving projection operator is different from the conventional projection operator in that duplicate tuples (on data attributes) are retained in the resulting relation because each tuple has a unique tuple identifier. We do not include duplicate elimination in the algebra $\boldsymbol{CA}^X_\preceq$ because it is not definable in $\boldsymbol{CQ}^X_\preceq$. Hence, the duplicate elimination operation is not related to the completeness of $\boldsymbol{CA}^X_\preceq$.

Lemma 3.7 *An order-preserving projection is a valid ordered relational query.*

Proof: Let $\pi_A(\mathbf{R}) = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$, where $\mathbf{R} = \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. By definition of the order-preserving projection, R′ and $\preceq_{\mathrm{R}'}$ are inclusively dependant on each other. We now prove that $\preceq_{\mathrm{R}'}$ is a linear order of R′.

(1) Let $t_1, t_2, t_3 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_3) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\ & (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge t_1 \preceq_R t_2) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}(t_1, \bar{x}_2) \wedge \exists \bar{x}_3.\mathrm{R}(t_3, \bar{x}_3) \wedge t_2 \preceq_R t_3) \\
\Longrightarrow\ & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_3.\mathrm{R}(t_3, \bar{x}_3) \wedge t_1 \preceq_R t_3 \\
\Longrightarrow\ & (t_1, t_3) \in \preceq_{\mathrm{R}'} .
\end{aligned}
$$

(2) Let $t_1, t_2 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_1) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\ & (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge t_1 \preceq_R t_2) \\
& \wedge (\exists \bar{x}'_2.\mathrm{R}(t_2, \bar{x}'_2) \wedge \exists \bar{x}'_1.\mathrm{R}(t_1, \bar{x}'_1) \wedge t_2 \preceq_R t_1) \\
\Longrightarrow\ & \bar{x}_1 = \bar{x}'_1 \wedge \bar{x}_2 = \bar{x}'_2 \wedge t_1 \preceq_R t_2 \wedge t_2 \preceq_R t_1 \\
\Longrightarrow\ & t_1 = t_2
\end{aligned}
$$

(3) Let t_1 and t_2 be two arbitrary tuples in R′. By definition of the order-preserving projection,

$$
\begin{aligned}
& t_1, t_2 \in \mathrm{R}'[\mathrm{T}] \\
\Longrightarrow\ & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \\
\Longrightarrow\ & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \\
& \wedge (t_1 \preceq_{\mathrm{R}} t_2 \vee t_2 \preceq_{\mathrm{R}} t_1) \\
\Longrightarrow\ & (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}} t_2) \\
& \vee (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge t_2 \preceq_{\mathrm{R}} t_1) \\
\Longrightarrow\ & t_1 \preceq_{\mathrm{R}'} t_2 \vee t_2 \preceq_{\mathrm{R}'} t_1
\end{aligned}
$$

This finishes the proof that the order-preserving projection is a valid ordered relational query. □

By definition, the data query of an order-preserving projection is a conjunctive query. Therefore, it is also a valid ordered conjunctive query in $\boldsymbol{CQ}^X_{\preceq}$.

3.2.4 Left Nested-loop Products

The left nested-loop product $\ltimes : \mathbf{R}_{\preceq} \times \mathbf{R}_{\preceq} \rightarrow \mathbf{R}_{\preceq}$ is an ordered counterpart to the Cartesian product. The schema of the resulting ordered relation is the extended schema of concatenation of the schemas of two argument relations. The formal definition of the left nested-loop product is shown as follows:

$$\begin{aligned}
\mathbf{R}_1 \ltimes \mathbf{R}_2 = \langle\{(t, \bar{x}_1, \bar{x}_2) \mid\ & \exists t_1, t_2.t = (t_1, t_2) \wedge \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2)\}, \\
\{(t_1, t_2) \mid\ & \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge (t_{11} \preceq_{R_1} t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22}))\}\rangle.
\end{aligned}$$

In this definition, the left nested-loop product takes two ordered relations as arguments, and returns the Cartesian product of two argument data relations in the lexicographic order. The tuple identifiers in the result are constructed by combining tuple identifiers of two arguments. This definition leads to the following lemma on the left nested-loop product:

Lemma 3.8 *A left nested-loop product operation is a valid ordered relational query.*

Proof: Let $\mathbf{R}_1 \ltimes \mathbf{R}_2 = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$, where $\mathbf{R}_1 = \langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle$ and $\mathbf{R}_2 = \langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle$. Note that R' and $\preceq_{\mathrm{R}'}$ are inclusively dependant on each other by definition of the left nested-loop product. We need to prove that $\preceq_{\mathrm{R}'}$ is a linear order of R'.

(1) Let $t_1, t_2, t_3 \in \mathrm{R}'[\mathrm{T}]$,

$$\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge\ (t_2, t_3) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\ & (\exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge (t_{11} \preceq_{R_1} t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22}))) \\
& \wedge (\exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge \exists t_{31}, t_{32}, \bar{x}_{31}, \bar{x}_{32}.t_3 = (t_{31}, t_{32}) \\
& \wedge \mathrm{R}_1(t_{31}, \bar{x}_{31}) \wedge \mathrm{R}_2(t_{32}, \bar{x}_{32}) \\
& \wedge (t_{21} \preceq_{R_1} t_{31} \vee (t_{21} = t_{31} \wedge t_{22} \preceq_{R_2} t_{32})))
\end{aligned}$$

$$
\begin{aligned}
\Longrightarrow\quad & \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge \exists t_{31}, t_{32}, \bar{x}_{31}, \bar{x}_{32}.t_3 = (t_{31}, t_{32}) \\
& \wedge \mathrm{R}_1(t_{31}, \bar{x}_{31}) \wedge \mathrm{R}_2(t_{32}, \bar{x}_{32}) \\
& \wedge (t_{11} \preceq_{R_1} t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22})) \\
& \wedge (t_{21} \preceq_{R_1} t_{31} \vee (t_{21} = t_{31} \wedge t_{22} \preceq_{R_2} t_{32})) \\
\Longrightarrow\quad & \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge \exists t_{31}, t_{32}, \bar{x}_{31}, \bar{x}_{32}.t_3 = (t_{31}, t_{32}) \\
& \wedge \mathrm{R}_1(t_{31}, \bar{x}_{31}) \wedge \mathrm{R}_2(t_{32}, \bar{x}_{32}) \\
& \wedge ((t_{11} \preceq_{R_1} t_{21} \wedge t_{21} \preceq_{R_1} t_{31}) \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22} \wedge t_{21} \preceq_{R_1} t_{31}) \\
& \vee (t_{11} \preceq_{R_1} t_{21} \wedge t_{21} = t_{31} \wedge t_{22} \preceq_{R_2} t_{32}) \\
& \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22} \wedge t_{21} = t_{31} \wedge t_{22} \preceq_{R_2} t_{32})) \\
\Longrightarrow\quad & \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{31}, t_{32}, \bar{x}_{31}, \bar{x}_{32}.t_3 = (t_{31}, t_{32}) \\
& \wedge \mathrm{R}_1(t_{31}, \bar{x}_{31}) \wedge \mathrm{R}_2(t_{32}, \bar{x}_{32}) \\
& \wedge (t_{11} \preceq_{R_1} t_{31} \vee (t_{11} = t_{31} \wedge t_{12} \preceq_{R_2} t_{32})) \\
\Longrightarrow\quad & (t_1, t_3) \in \preceq_{\mathrm{R}'} .
\end{aligned}
$$

(2) Let $t_1, t_2 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_1) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\quad & (\exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge (t_{11} \preceq_{R_1} t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22}))) \\
& \wedge (\exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge (t_{21} \preceq_{R_1} t_{11} \vee (t_{21} = t_{11} \wedge t_{22} \preceq_{R_2} t_{12})))
\end{aligned}
$$

$$
\begin{aligned}
\Longrightarrow\quad & \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge (t_{11} \preceq_{R_1} t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22})) \\
& \wedge (t_{21} \preceq_{R_1} t_{11} \vee (t_{21} = t_{11} \wedge t_{22} \preceq_{R_2} t_{12})) \\
\Longrightarrow\quad & \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge ((t_{11} \preceq_{R_1} t_{21} \wedge t_{21} \preceq_{R_1} t_{11}) \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22} \wedge t_{21} \preceq_{R_1} t_{11}) \\
& \vee (t_{11} \preceq_{R_1} t_{21} \wedge t_{21} = t_{11} \wedge t_{22} \preceq_{R_2} t_{12}) \\
& \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22} \wedge t_{21} = t_{11} \wedge t_{22} \preceq_{R_2} t_{12})) \\
\Longrightarrow\quad & \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge ((t_{11} = t_{21}) \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22} \wedge t_{21} \preceq_{R_1} t_{11}) \\
& \vee (t_{11} \preceq_{R_1} t_{21} \wedge t_{21} = t_{11} \wedge t_{22} \preceq_{R_2} t_{12}) \vee (t_{11} = t_{21} \wedge t_{12} = t_{22})) \\
\Longrightarrow\quad & t_{11} = t_{21} \wedge t_{12} = t_{22} \\
\Longrightarrow\quad & t_1 = t_2.
\end{aligned}
$$

(3) Let t_1 and t_2 be two arbitrary tuples in R',

$$
\begin{aligned}
& t_1, t_2 \in \mathrm{R}'[\mathrm{T}] \\
\Longrightarrow\quad & \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
\Longrightarrow\quad & (\exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22})) \\
& \wedge ((t_{11} \preceq_{R_1} t_{21} \vee t_{21} \preceq_{R_1} t_{11}) \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22} \wedge t_{21} \preceq_{R_1} t_{11}) \\
& \vee (t_{11} \preceq_{R_1} t_{21} \wedge t_{21} = t_{11} \wedge t_{22} \preceq_{R_2} t_{12}) \\
& \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22} \wedge t_{21} = t_{11} \wedge t_{22} \preceq_{R_2} t_{12}))
\end{aligned}
$$

$$
\begin{aligned}
\Longrightarrow \quad & \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}.t_1 = (t_{11}, t_{12}) \\
& \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}.t_2 = (t_{21}, t_{22}) \\
& \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge ((t_{11} \preceq_{R_1} t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22})) \\
& \vee (t_{21} \preceq_{R_1} t_{11} \vee (t_{21} = t_{11} \wedge t_{22} \preceq_{R_2} t_{12}))) \\
\Longrightarrow \quad & t_1 \preceq_{\mathrm{R}'} t_2 \vee t_2 \preceq_{\mathrm{R}'} t_1.
\end{aligned}
$$

Thus, we have proved that the left nested-loop product is a valid ordered relational query. □

By definition, the data query of a left nested-loop product is a conjunctive query. Therefore, it is also a valid ordered conjunctive query in $\boldsymbol{CQ}^X_{\preceq}$.

3.2.5 Order Concatenation

The ordered concatenation operator $\cup : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ is different from the union operator in conventional relational algebras. It is exclusive to ordered algebra and is used to manipulate orders within a single ordered relation. The schemas of both argument relations and the resulting relation are the same. The formal definition of the order concatenation operator is given as follows:

$$
\begin{aligned}
\mathbf{R}_1 \cup \mathbf{R}_2 = \quad & \langle \{(t, \bar{x}) | \quad \mathrm{R}_1(t, \bar{x})) \vee \mathrm{R}_2(t, \bar{x}))\}, \\
& \{(t_1, t_2) | \quad (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2)) \\
& \qquad \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \ t_1 \preceq_{R_1} t_2) \\
& \qquad \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge \ t_1 \preceq_{R_2} t_2)\} \rangle.
\end{aligned}
$$

By this definition, the order concatenation operator appends the second argument ordered relation at the end of the first argument. Hence, tuples from the same argument remain in the same order as in the argument; tuples from the first argument are prior to tuples from the second argument. Furthermore, tuples with duplicate data values are retained in the result because they have different tuple identifiers.

Since the order concatenation operator can only be used to manipulate orders within a single ordered relation, the argument relations of the operation are originally different fragments from the same ordered relation. Thus, it is impossible that

tuples from different fragments of the same relation have the same tuple identifier. This makes it possible that tuple identifiers of the resulting tuples are identical to the original tuple identifiers in arguments.

Lemma 3.9 *An order concatenation operation is a valid ordered relational query.*

Proof: Let $\mathbf{R}_1 \cup \mathbf{R}_2 = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$, where $\mathbf{R}_1 = \langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle$ and $\mathbf{R}_2 = \langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle$. The data relation R' and the order relation $\preceq_{\mathrm{R}'}$ are inclusively dependant on each other by definition of order concatenation. We need to prove that $\preceq_{\mathrm{R}'}$ is a linear order of R'.

(1) Let $t_1, t_2, t_3 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_3) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\quad & ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2)) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \ t_1 \preceq_{R_1} t_2) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge \ t_1 \preceq_{R_2} t_2)) \\
& \wedge ((\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \exists \bar{x}_3.\mathrm{R}_2(t_3, \bar{x}_3)) \\
& \vee (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \ \exists \bar{x}_3.\mathrm{R}_1(t_3, \bar{x}_3) \wedge \ t_2 \preceq_{R_1} t_3) \\
& \vee (\exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge \ \exists \bar{x}_3.\mathrm{R}_2(t_3, \bar{x}_3) \wedge \ t_2 \preceq_{R_2} t_3)) \\
\Longrightarrow\quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge \ \exists \bar{x}_3.\mathrm{R}_2(t_3, \bar{x}_3) \wedge \ t_2 \preceq_{R_2} t_3) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \ \exists \bar{x}_3.\mathrm{R}_2(t_3, \bar{x}_3) \wedge \ t_1 \preceq_{R_1} t_2) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \exists \bar{x}_3.\mathrm{R}_1(t_3, \bar{x}_3) \wedge \ t_1 \preceq_{R_1} t_2 \wedge t_2 \preceq_{R_1} t_3) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge \exists \bar{x}_3.\mathrm{R}_2(t_3, \bar{x}_3) \wedge \ t_1 \preceq_{R_2} t_2 \wedge t_2 \preceq_{R_2} t_3) \\
\Longrightarrow\quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_3.\mathrm{R}_2(t_3, \bar{x}_3)) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_3.\mathrm{R}_1(t_3, \bar{x}_3) \wedge \ t_1 \preceq_{R_1} t_3) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_3.\mathrm{R}_2(t_3, \bar{x}_3) \wedge \ t_1 \preceq_{R_2} t_3) \\
\Longrightarrow\quad & (t_1, t_3) \in \preceq_{\mathrm{R}'} .
\end{aligned}
$$

(2) Let $t_1, t_2 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_1) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\quad & ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2)) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \ t_1 \preceq_{R_1} t_2) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge \ t_1 \preceq_{R_2} t_2)) \\
& \wedge ((\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1)) \\
& \vee (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \ \exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \ t_2 \preceq_{R_1} t_1) \\
& \vee (\exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge \ \exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \ t_2 \preceq_{R_2} t_1))
\end{aligned}
$$

$$
\begin{aligned}
\Longrightarrow\quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge\ \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2)\\
& \wedge t_1 \preceq_{R_1} t_2 \wedge t_2 \preceq_{R_1} t_1)\\
& \vee(\exists \bar{x}_1.\mathrm{R}_2(t_1,\bar{x}_1) \wedge\ \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2)\\
& \wedge t_1 \preceq_{R_2} t_2 \wedge t_2 \preceq_{R_2} t_1)\\
\Longrightarrow\quad & t_1 = t_2.
\end{aligned}
$$

(3) Let t_1 and t_2 be two arbitrary tuples in R'. By definition of order concatenation,

$$
\begin{aligned}
& t_1, t_2 \in \mathrm{R}'[\mathrm{T}]\\
\Longrightarrow\quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \vee\ \mathrm{R}_2(t_1,\bar{x}_1))\\
& \wedge(\exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \vee\ \mathrm{R}_2(t_2,\bar{x}_2))\\
\Longrightarrow\quad & ((\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \vee \exists \bar{x}_1.\mathrm{R}_2(t_1,\bar{x}_1))\\
& \wedge(\exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \vee\ \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2))\\
& \wedge(t_1 \preceq_{\mathrm{R}_1} t_2 \vee t_2 \preceq_{\mathrm{R}_1} t_1)\\
& \wedge(t_1 \preceq_{\mathrm{R}_2} t_2 \vee t_2 \preceq_{\mathrm{R}_2} t_1)\\
\Longrightarrow\quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}_1} t_2)\\
& \vee(\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \wedge t_2 \preceq_{\mathrm{R}_1} t_1)\\
& \vee(\exists \bar{x}_1.\mathrm{R}_2(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}_2} t_2)\\
& \vee(\exists \bar{x}_1.\mathrm{R}_2(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2) \wedge t_2 \preceq_{\mathrm{R}_2} t_1)\\
& \vee(\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2))\\
& \vee(\exists \bar{x}_1.\mathrm{R}_2(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2))\\
\Longrightarrow\quad & t_1 \preceq_{\mathrm{R}'} t_2 \vee t_2 \preceq_{\mathrm{R}'} t_1.
\end{aligned}
$$

Hence, we have proved that the order concatenation is a valid ordered relational query. □

3.2.6 Order Reduction

The order reduction operator $- : \mathbf{R}\ \times \mathbf{R} \rightarrow \mathbf{R}$ is exclusive to the ordered algebra. It is also used to manipulate orders within a single ordered relation by separating the input ordered relation into multiple segments. The schemas of both argument relations and the resulting relation are the same. A formal definition of the order reduction operator is proposed as follows:

$$
\begin{aligned}
\mathbf{R}_1\ -\ \mathbf{R}_2 = \ \langle \{(t,\bar{x}) \mid\ & \mathrm{R}_1(t,\bar{x}) \wedge \neg\ \mathrm{R}_2(t,\bar{x})\},\\
\{(t_1,t_2) \mid\ & (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \neg\ \mathrm{R}_2(t_1,\bar{x}_1))\\
& \wedge(\exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \wedge \neg\ \mathrm{R}_2(t_2,\bar{x}_2))\\
& \wedge\ t_1 \preceq_{R_1} t_2\}\rangle.
\end{aligned}
$$

In this definition, the order reduction operator returns all tuples of the first argument that do not have duplicate tuples in the second argument. The resulting tuples remain in the same order as in the first argument. Here, "duplicate" tuples are those tuples with both identical tuple identifiers and identical data values. The tuple identifiers of the result are the same as those in the first argument.

Since the ordered conjunctive queries do not involve negation in the data query, this order reduction operator is not definable in $\boldsymbol{CQ}^X_{\preceq}$. This operator is necessary in $\boldsymbol{CA}^X_{\preceq}$ because it is used to handle the negations in the order specification of a $\boldsymbol{CQ}^X_{\preceq}$ query. In addition, the order reduction operator is only used to manipulate the order within a single relation, and therefore is not the counterpart of the set difference operator in conventional relational algebras. Furthermore, the definition of order reduction leads to the following lemma:

Lemma 3.10 *An order reduction operation is a valid ordered relational query.*

Proof: Let $\mathbf{R}_1 - \mathbf{R}_2 = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$, where $\mathbf{R}_1 = \langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle$ and $\mathbf{R}_2 = \langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle$. The data relation R' and order relation $\preceq_{\mathrm{R}'}$ are inclusively dependant on each other by definition of the order reduction. We now need to prove that $\preceq_{\mathrm{R}'}$ is a linear order of R'.

(1) Let $t_1, t_2, t_3 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge\ (t_2, t_3) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\quad & ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\ \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\ \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge\ t_1 \preceq_{R_1} t_2) \\
& \wedge ((\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\ \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge (\exists \bar{x}_3.\mathrm{R}_1(t_3, \bar{x}_3) \wedge \neg\ \mathrm{R}_2(t_3, \bar{x}_3)) \\
& \wedge\ t_2 \preceq_{R_1} t_3) \\
\Longrightarrow\quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\ \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_3.\mathrm{R}_1(t_3, \bar{x}_3) \wedge \neg\ \mathrm{R}_2(t_3, \bar{x}_3)) \wedge t_1 \preceq_{R_1} t_3 \\
\Longrightarrow\quad & (t_1, t_3) \in \preceq_{\mathrm{R}'}\ .
\end{aligned}
$$

(2) Let $t_1, t_2 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_1) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow \quad & ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\, \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\, \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge\, t_1 \preceq_{R_1} t_2) \\
& \wedge ((\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\, \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\, \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge\, t_2 \preceq_{R_1} t_1) \\
\Longrightarrow \quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\, \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\, \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge (t_1 \preceq_{R_1} t_2 \wedge t_2 \preceq_{R_1} t_1) \\
\Longrightarrow \quad & t_1 = t_2.
\end{aligned}
$$

(3) Let t_1 and t_2 be two arbitrary tuples in R'. By definition of order reduction,

$$
\begin{aligned}
& t_1, t_2 \in \mathrm{R}'[\mathrm{T}] \\
\Longrightarrow \quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\, \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\, \mathrm{R}_2(t_2, \bar{x}_2)) \\
\Longrightarrow \quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\, \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\, \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge (t_1 \preceq_{\mathrm{R}_1} t_2 \vee t_2 \preceq_{\mathrm{R}_1} t_1) \\
\Longrightarrow \quad & ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\, \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\, \mathrm{R}_2(t_2, \bar{x}_2)) \wedge t_1 \preceq_{\mathrm{R}_1} t_2) \\
& \vee ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\, \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\, \mathrm{R}_2(t_2, \bar{x}_2)) \wedge t_2 \preceq_{\mathrm{R}_1} t_1) \\
\Longrightarrow \quad & t_1 \preceq_{\mathrm{R}'} t_2 \vee t_2 \preceq_{\mathrm{R}'} t_1
\end{aligned}
$$

Hence, we have proved that the order reduction is a valid ordered relational query. □

3.2.7 Order Identity

The order identity operator $\mu : \mathbf{R} \to \mathbf{R}$ is exclusively proposed in ordered context. The argument and resulting ordered relations share the same schema. The formal definition of the order identity operator is given as follows:

$$
\begin{aligned}
\mu(\mathbf{R}) = \quad & \langle \{(t, \bar{x}) \mid \quad \mathrm{R}(t, \bar{x})\}, \\
& \{(t_1, t_2) \mid \quad t_1 \preceq_R t_2\} \rangle.
\end{aligned}
$$

In this definition, the order identity operator takes an ordered relation as the argument and returns the argument relation as the result. The tuple identifier of each data tuple in the resulting ordered relation is identical to the tuple identifier of the same data tuple in the argument relation.

Lemma 3.11 *An order identity operation is a valid ordered relational query.*

Proof:

By definition, the resulting ordered relation of an order identity operation has identical data and identical order relations to the argument. Since the input ordered relation is valid, the output ordered relation is also valid. Hence, the order identity operation is a valid ordered relational query. □

The order identity operator is included in the algebra $\boldsymbol{CA}^X_\preceq$ because we need it to generate a generalized ordered algebraic expression for any $\boldsymbol{CQ}^X_\preceq$ query.

3.2.8 Order Reverse

The order reverse operator $\nu : \mathbf{R} \rightarrow \mathbf{R}$ is specific to the ordered algebra as well. The argument and resulting ordered relations share the same schema. The formal definition of the order reverse operator is given as follows:

$$\begin{aligned} \nu(\mathbf{R}) = \quad & \langle \{(t, \bar{x}) | \quad \mathrm{R}(t, \bar{x})\}, \\ & \{(t_1, t_2) | \quad t_2 \preceq_R t_1\} \rangle. \end{aligned}$$

In this definition, the order reverse operator takes an ordered relation as its argument and returns as its result the ordered relation with the same data relation; however, it is in the reverse order to the argument. The tuple identifier of each data tuple in the resulting ordered relation is identical to the tuple identifier of the same data tuple in the argument relation. The following lemma holds for the order reverse operator:

Lemma 3.12 *An order reverse operation is a valid ordered relational query.*

Proof: Let $\nu(\mathbf{R}) = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$, where $\mathbf{R} = \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. Note that R' and $\preceq_{\mathrm{R}'}$ are inclusively dependant on each other by definition of the order reverse operation. We now need to prove that $\preceq_{\mathrm{R}'}$ is a linear order of R'.

(1) Let $t_1, t_2, t_3 \in \mathrm{R}'[\mathrm{T}]$,

$$\begin{aligned} & (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_3) \in \preceq_{\mathrm{R}'} \\ \Longrightarrow \quad & (t_2 \preceq_R t_1) \wedge (t_3 \preceq_R t_2) \\ \Longrightarrow \quad & t_3 \preceq_R t_1 \\ \Longrightarrow \quad & (t_1, t_3) \in \preceq_{\mathrm{R}'} . \end{aligned}$$

(2) Let $t_1, t_2 \in \mathrm{R}'[\mathrm{T}]$,

$$\begin{aligned} & (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_1) \in \preceq_{\mathrm{R}'} \\ \Longrightarrow \quad & t_2 \preceq_R t_1 \wedge t_1 \preceq_R t_2 \\ \Longrightarrow \quad & t_1 = t_2. \end{aligned}$$

(3) Let t_1 and t_2 be two arbitrary tuples in R'. By definition of the order reverse operation,

$$\begin{aligned} & t_1, t_2 \in \mathrm{R}'[\mathrm{T}] \\ \Longrightarrow \quad & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \\ \Longrightarrow \quad & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \\ & \wedge (t_2 \preceq_{\mathrm{R}} t_1 \vee t_1 \preceq_{\mathrm{R}} t_2) \\ \Longrightarrow \quad & (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge t_2 \preceq_{\mathrm{R}} t_1) \\ & \vee (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}} t_2) \\ \Longrightarrow \quad & t_1 \preceq_{\mathrm{R}'} t_2 \vee t_2 \preceq_{\mathrm{R}'} t_1. \end{aligned}$$

Thus, we have proved that the order reverse operator is a valid ordered relational query. □

3.3 Transformation Rules

In this section, we provide a set of transformation rules for the ordered conjunctive algebra $\boldsymbol{CA}^{X}_{\preceq}$. First, we describe transformation rules that are taken from conventional relational algebras and are tailored to the ordered context. Then, we discuss transformation rules for the order operators that are specifically developed for the ordered conjunctive algebra.

In ordered relational databases, two ordered queries are equivalent if they have both equivalent data queries and equivalent order queries. The following transformation rules guarantee both the equivalence of data queries and the equivalence of order queries for the ordered queries on both sides. In addition, the transformation rules are given in the algebraic equations which represent both the left-to-right

equivalence and the right-to-left equivalence. In these equations, each argument can be either an ordered relation or an expression of ordered operations.

Proposition 3.13 *Let* $\mathbf{Att}(\mathbf{R})$ *be the set of data attributes in the ordered relation* $\mathbf{R}$*, and* $\mathbf{Att}(\mathrm{p})$ *be the set of data attributes the selection predicate* p*. The following transformation rules hold in* $\boldsymbol{CA}^{X}_{\preceq}$*:*

$(R1)$ $\sigma_{\mathrm{p}}(\sigma_{\mathrm{p}'}(\mathbf{R})) = \sigma_{\mathrm{p}\wedge\mathrm{p}'}(\mathbf{R})$

$(R2)$ $\pi_{\mathrm{A}_1}(\pi_{\mathrm{A}_2}(\mathbf{R})) = \pi_{\mathrm{A}_1}(\mathbf{R})$ *if* $\mathrm{A}_1 \subseteq \mathrm{A}_2$

$(R3)$ $\sigma_{\mathrm{p}}(\pi_A(\mathbf{R})) = \pi_A(\sigma_{\mathrm{p}}(\mathbf{R}))$ *if* $\mathbf{Att}(\mathrm{p}) \subseteq A$

$(R4)$ $\sigma_{\mathrm{p}}(\mathbf{R}_1 \ltimes \mathbf{R}_2) = \sigma_{\mathrm{p}}(\mathbf{R}_1) \ltimes \mathbf{R}_2$ *if* $\mathbf{Att}(\mathrm{p}) \subseteq \mathbf{Att}(\mathbf{R}_1)$

$(R5)$ $\sigma_{\mathrm{p}}(\mathbf{R}_1 \ltimes \mathbf{R}_2) = \mathbf{R}_1 \ltimes \sigma_{\mathrm{p}}(\mathbf{R}_2)$ *if* $\mathbf{Att}(\mathrm{p}) \subseteq \mathbf{Att}(\mathbf{R}_2)$

$(R6)$ $\pi_A(\mathbf{R}_1 \ltimes \mathbf{R}_2) = \pi_{A_1}(\mathbf{R}_1) \ltimes \pi_{A_2}(\mathbf{R}_2)$
if $A_1 \subseteq \mathbf{Att}(\mathbf{R}_1) \wedge A_2 \subseteq \mathbf{Att}(\mathbf{R}_2) \wedge A_1 \cup A_2 = A \wedge A_1 \cap A_2 = \emptyset$

$(R7)$ $(\mathbf{R}_1 \ltimes \mathbf{R}_2) \ltimes \mathbf{R}_3 = \mathbf{R}_1 \ltimes (\mathbf{R}_2 \ltimes \mathbf{R}_3)$

$(R8)$ $\sigma_{\mathrm{p}}(\mathbf{R}_1 - \mathbf{R}_2) = \sigma_{\mathrm{p}}(\mathbf{R}_1) - \mathbf{R}_2$

$(R9)$ $\sigma_{\mathrm{p}}(\mathbf{R}_1 - \mathbf{R}_2) = \sigma_{\mathrm{p}}(\mathbf{R}_1) - \sigma_{\mathrm{p}}(\mathbf{R}_2)$

$(R10)$ $\sigma_{\mathrm{p}}(\mathbf{R}_1 \cup \mathbf{R}_2) = \sigma_{\mathrm{p}}(\mathbf{R}_1) \cup \sigma_{\mathrm{p}}(\mathbf{R}_2)$

$(R11)$ $\pi_A(\mathbf{R}_1 \cup \mathbf{R}_2) = \pi_A(\mathbf{R}_1) \cup \pi_A(\mathbf{R}_2)$

$(R12)$ $(\mathbf{R}_1 \cup \mathbf{R}_2) \cup \mathbf{R}_3 = \mathbf{R}_1 \cup (\mathbf{R}_2 \cup \mathbf{R}_3)$

$(R13)$ $\sigma_{\mathrm{p}}(\nu(\mathbf{R})) = \nu(\sigma_{\mathrm{p}}(\mathbf{R}))$

$(R14)$ $\pi_A(\nu(\mathbf{R})) = \nu(\pi_A(\mathbf{R}))$

$(R15)$ $\nu(\mathbf{R}_1 \ltimes \mathbf{R}_2) = \nu(\mathbf{R}_1) \ltimes \nu(\mathbf{R}_2)$

Proof:
Assume that the ordered relation $\mathbf{R} = \langle R, \preceq_R \rangle$ has a schema $\mathrm{R}(\mathrm{T}, X_1, \ldots, X_n)$, and the ordered relation $\mathbf{R}_i = \langle R_i, \preceq_{R_i} \rangle$ has a schema $\mathrm{R}_i(\mathrm{T}, X_{i1}, \ldots, X_{in_i})$, for $i = 1, 2, 3$.

($R1$) Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \sigma_{\mathrm{p}'} \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. By definition of the order-preserving selection,

$$\begin{aligned}
\mathrm{R}' &= \{(t, \bar{x}) \mid \mathrm{R}(t, \bar{x}) \wedge \mathrm{p}'(t, \bar{x})\} \\
\\
\preceq_{\mathrm{R}'} &= \{(t_1, t_2) \mid \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \ \mathrm{p}'(t_1, \bar{x}_1) \\
&\quad \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \ \mathrm{p}'(t_2, \bar{x}_2) \\
&\quad \wedge t_1 \preceq_{\mathrm{R}} t_2\}.
\end{aligned}$$

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}} \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$\begin{aligned}
\mathrm{R}'' &= \{(t, \bar{x}) \mid \mathrm{R}'(t, \bar{x}) \wedge \mathrm{p}(t, \bar{x})\} \\
&= \{(t, \bar{x}) \mid \mathrm{R}(t, \bar{x}) \wedge \mathrm{p}(t, \bar{x}) \wedge \mathrm{p}'(t, \bar{x})\} \\
\\
\preceq_{\mathrm{R}''} &= \{(t_1, t_2) \mid \exists \bar{x}_1.\mathrm{R}'(t_1, \bar{x}_1) \wedge \ \mathrm{p}(t_1, \bar{x}_1) \\
&\quad \wedge \exists \bar{x}_2.\mathrm{R}'(t_2, \bar{x}_2) \wedge \ \mathrm{p}(t_2, \bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}'} t_2\} \\
&= \{(t_1, t_2) \mid \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}'(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
&\quad \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}'(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}} t_2\}.
\end{aligned}$$

Therefore,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p} \wedge \mathrm{p}'} \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle.$$

($\mathrm{R}2$) Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \pi_{\mathrm{A}_2} \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. Without loss of generality, we assume that $\mathrm{A}_2 = \{X_1, \ldots, X_k\}$. By definition of the order-preserving projection,

$$\begin{aligned}
\mathrm{R}' &= \{(t, x_1, \ldots, x_k) \mid \exists x_{k+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n)\} \\
\\
\preceq_{\mathrm{R}'} &= \{(t_1, t_2) \mid \exists x_{11}, \ldots, x_{1n}.\mathrm{R}(t_1, x_{11}, \ldots, x_{1n}) \\
&\quad \wedge \exists x_{21}, \ldots, x_{2n}.\mathrm{R}(t_2, x_{21}, \ldots, x_{2n}) \wedge t_1 \preceq_R t_2\}.
\end{aligned}$$

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \pi_{\mathrm{A}_1} \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Since $\mathrm{A}_1 \subseteq \mathrm{A}_2$ holds, without loss of generality, we assume that $\mathrm{A}_1 = \{X_1, \ldots, X_l\}$, and $l \leq k$. Then,

$$\begin{aligned}
\mathrm{R}'' &= \{(t, x_1, \ldots, x_l) \mid \exists x_{l+1}, \ldots, x_k.\mathrm{R}'(t, x_1, \ldots, x_k)\} \\
&= \{(t, x_1, \ldots, x_l) \mid \exists x_{l+1}, \ldots, x_k.(\exists x_{k+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n))\} \\
&= \{(t, x_1, \ldots, x_l) \mid \exists x_{l+1}, \ldots, x_k, x_{k+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n))\} \\
\\
\preceq_{\mathrm{R}''} &= \{(t_1, t_2) \mid \exists x_{11}, \ldots, x_{1k}.\mathrm{R}'(t_1, x_{11}, \ldots, x_{1k}) \\
&\quad \wedge \exists x_{21}, \ldots, x_{2k}.\mathrm{R}'(t_2, x_{21}, \ldots, x_{2k}) \wedge t_1 \preceq_{R'} t_2\} \\
&= \{(t_1, t_2) \mid \exists x_{11}, \ldots, x_{1n}.\mathrm{R}(t_1, x_{11}, \ldots, x_{1n}) \\
&\quad \wedge \exists x_{21}, \ldots, x_{2n}.\mathrm{R}(t_2, x_{21}, \ldots, x_{2n}) \wedge t_1 \preceq_R t_2\}.
\end{aligned}$$

As a consequence,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \pi_{\mathrm{A}_1} \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle.$$

($R3$) Without loss of generality, we assume that $\mathrm{A} = \{X_1, \ldots, X_k\}$. By definition of the order-preserving projection,

$$\begin{aligned} \pi_A\langle R, \preceq_R \rangle \;=\; & \langle \{(t, x_1, \ldots, x_k) \mid \exists x_{k+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n)\}, \\ & \{(t_1, t_2) \mid \exists x_{11}, \ldots, x_{1n}.\mathrm{R}(t_1, x_{11}, \ldots, x_{1n}) \\ & \wedge \exists x_{21}, \ldots, x_{2n}.\mathrm{R}(t_2, x_{21}, \ldots, x_{2n}) \wedge t_1 \preceq_R t_2\}\rangle. \end{aligned}$$

Then,

$$\begin{aligned} \sigma_{\mathrm{p}}(\pi_A\langle R, \preceq_R \rangle) \;=\; & \langle \{(t, x_1, \ldots, x_k) \mid \exists x_{k+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n) \\ & \wedge \mathrm{p}(t, x_1, \ldots, x_n)\}, \\ & \{(t_1, t_2) \mid \exists x_{11}, \ldots, x_{1n}.\mathrm{R}(t_1, x_{11}, \ldots, x_{1n}) \wedge \mathrm{p}(t_1, x_{11}, \ldots, x_{1n}) \\ & \wedge \exists x_{21}, \ldots, x_{2n}.\mathrm{R}(t_2, x_{21}, \ldots, x_{2n}) \wedge \mathrm{p}(t_1, x_{21}, \ldots, x_{2n}) \\ & \wedge t_1 \preceq_R t_2\}\rangle \\ =\; & \pi_A(\sigma_{\mathrm{p}}(\mathbf{R})). \end{aligned}$$

(R4) Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \ltimes \mathbf{R}_2$. By definition of left nested-loop product,

$$\mathrm{R}' \;=\; \{(t, \bar{x}_1, \bar{x}_2) \mid \exists t_1, t_2.t = (t_1, t_2) \wedge \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2)\}$$

$$\begin{aligned} \preceq_{\mathrm{R}'} = \{(t_1, t_2) \mid\; & \exists t_{11}, \bar{x}_{11}.\mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists t_{12}, \bar{x}_{12}.\mathrm{R}_2(t_{12}, \bar{x}_{12}) \\ & \wedge \exists t_{21}, \bar{x}_{21}.\mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists t_{22}, \bar{x}_{22}.\mathrm{R}_2(t_{22}, \bar{x}_{22}) \\ & \wedge t_1 = (t_{11}, t_{12}) \wedge t_2 = (t_{21}, t_{22}) \\ & \wedge (t_{11} \preceq_R t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_R t_{22}))\}\rangle, \end{aligned}$$

where $\bar{x}_1 = (x_{11}, \ldots, x_{1n})$, and $\bar{x}_{11} = (x_{111}, \ldots, x_{11n})$.

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}}\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$\mathrm{R}'' \;=\; \{(t, \bar{x}_1, \bar{x}_2) \mid \exists t_1, t_2.t = (t_1, t_2) \wedge \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2) \wedge \mathrm{p}(t_1, \bar{x}_1)\}$$

$$\begin{aligned} \preceq_{\mathrm{R}''} = \{(t_1, t_2) \mid\; & \exists t_{11}, \bar{x}_{11}.\mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists t_{12}, \bar{x}_{12}.\mathrm{R}_2(t_{12}, \bar{x}_{12}) \wedge \mathrm{p}(t_{11}, \bar{x}_{11}) \\ & \wedge \exists t_{21}, \bar{x}_{21}.\mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists t_{22}, \bar{x}_{22}.\mathrm{R}_2(t_{22}, \bar{x}_{22}) \wedge \mathrm{p}(t_{21}, \bar{x}_{21}) \\ & \wedge t_1 = (t_{11}, t_{12}) \wedge t_2 = (t_{21}, t_{22}) \\ & \wedge (t_{11} \preceq_R t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_R t_{22}))\}\rangle. \end{aligned}$$

Therefore,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}}(\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle) \ltimes \langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle.$$

($R5$) Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \ltimes \mathbf{R}_2$. By definition of left nested-loop product,

$$\begin{aligned}
\mathrm{R}' \;=\; & \{(t, \bar{x}_1, \bar{x}_2) \mid \; \exists t_1, t_2 . t = (t_1, t_2) \wedge \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2)\} \\
\\
\preceq_{\mathrm{R}'} \;=\; & \{(t_1, t_2) \mid \quad \exists t_{11}, \bar{x}_{11} . \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists t_{12}, \bar{x}_{12} . \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \qquad \wedge \exists t_{21}, \bar{x}_{21} . \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists t_{22}, \bar{x}_{22} . \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \qquad \wedge\; t_1 = (t_{11}, t_{12}) \wedge t_2 = (t_{21}, t_{22}) \\
& \qquad \wedge\; (t_{11} \preceq_R t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_R t_{22}))\}\rangle,
\end{aligned}$$

where $\bar{x}_1 = (x_{11}, \ldots, x_{1n})$, and $\bar{x}_{11} = (x_{111}, \ldots, x_{11n})$.

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}} \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$\mathrm{R}'' \;=\; \{(t, \bar{x}_1, \bar{x}_2) \mid \; \exists t_1, t_2 . t = (t_1, t_2) \wedge \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2)\}$$

$$\begin{aligned}
\preceq_{\mathrm{R}''} \;=\; & \{(t_1, t_2) \mid \quad \exists t_{11}, \bar{x}_{11} . \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists t_{12}, \bar{x}_{12} . \mathrm{R}_2(t_{12}, \bar{x}_{12}) \wedge \mathrm{p}(t_{12}, \bar{x}_{12}) \\
& \qquad \wedge \exists t_{21}, \bar{x}_{21} . \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists t_{22}, \bar{x}_{22} . \mathrm{R}_2(t_{22}, \bar{x}_{22}) \wedge \mathrm{p}(t_{22}, \bar{x}_{22}) \\
& \qquad \wedge\; t_1 = (t_{11}, t_{12}) \wedge t_2 = (t_{21}, t_{22}) \\
& \qquad \wedge\; (t_{11} \preceq_R t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_R t_{22}))\}\rangle.
\end{aligned}$$

Thus,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle \ltimes \sigma_{\mathrm{p}}(\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle).$$

($R6$) Assume that $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \ltimes \mathbf{R}_2$, $A_1 = (\mathrm{X}_{11}, \ldots, \mathrm{X}_{1k})$, $A_2 = (\mathrm{X}_{21}, \ldots, \mathrm{X}_{2l})$, $A_1 \cup A_2 = A$, and $A_1 \cap A_2 = \emptyset$. By definition of the left nested-loop product,

$$\mathrm{R}' \;=\; \{(t, \bar{x}_1, \bar{x}_2) \mid \; \exists t_1, t_2 . t = (t_1, t_2) \wedge \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2)\}$$

$$\begin{aligned}
\preceq_{\mathrm{R}'} \;=\; & \{(t_1, t_2) \mid \quad \exists t_{11}, \bar{x}_{11} . \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists t_{12}, \bar{x}_{12} . \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \qquad \wedge \exists t_{21}, \bar{x}_{21} . \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists t_{22}, \bar{x}_{22} . \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \qquad \wedge\; t_1 = (t_{11}, t_{12}) \wedge t_2 = (t_{21}, t_{22}) \\
& \qquad \wedge\; (t_{11} \preceq_R t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_R t_{22}))\}\rangle,
\end{aligned}$$

where $\bar{x}_1 = (x_{11}, \ldots, x_{1n})$, and $\bar{x}_{11} = (x_{111}, \ldots, x_{11n})$.

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \pi_A \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$
\begin{aligned}
\mathrm{R}'' \;=\;& \{(t, x_{11}, \ldots, x_{1k}, x_{21}, \ldots, x_{2l}) \mid \exists t_1, t_2 . t = (t_1, t_2) \\
& \wedge \exists x_{1,k+1}, \ldots, x_{1n} . \mathrm{R}_1(t_1, \bar{x}_1) \\
& \wedge \exists x_{2,l+1}, \ldots, x_{2m} . \mathrm{R}_2(t_2, \bar{x}_2)\}
\end{aligned}
$$

$$
\begin{aligned}
\preceq_{\mathrm{R}''} \;=\;& \{(t_1, t_2) \mid \exists t_{11}, \bar{x}_{11} . \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists t_{12}, \bar{x}_{12} . \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, \bar{x}_{21} . \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists t_{22}, \bar{x}_{22} . \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge\; t_1 = (t_{11}, t_{12}) \wedge t_2 = (t_{21}, t_{22}) \\
& \wedge\; (t_{11} \preceq_R t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_R t_{22}))\}\rangle.
\end{aligned}
$$

Therefore,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \pi_A(\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle) \ltimes \langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle.$$

($R7$) Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \ltimes \mathbf{R}_2$. By definition of the left nested-loop product,

$$
\begin{aligned}
\mathrm{R}' \;=\;& \{(t, \bar{x}_1, \bar{x}_2) \mid \exists t_1, t_2 . t = (t_1, t_2) \wedge \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2)\}
\end{aligned}
$$

$$
\begin{aligned}
\preceq_{\mathrm{R}'} \;=\;& \{(t_1, t_2) \mid \exists t_{11}, \bar{x}_{11} . \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists t_{12}, \bar{x}_{12} . \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{21}, \bar{x}_{21} . \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists t_{22}, \bar{x}_{22} . \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge\; t_1 = (t_{11}, t_{12}) \wedge t_2 = (t_{21}, t_{22}) \\
& \wedge\; (t_{11} \preceq_R t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_R t_{22}))\}\rangle,
\end{aligned}
$$

where $\bar{x}_1 = (x_{11}, \ldots, x_{1n})$ and $\bar{x}_{11} = (x_{111}, \ldots, x_{11n})$.

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle \ltimes \langle \mathrm{R}_3, \preceq_{\mathrm{R}_3} \rangle$. Then,

$$
\begin{aligned}
\mathrm{R}'' \;=\;& \{(t, \bar{x}_1, \bar{x}_2, \bar{x}_3) \mid \exists t_1, t_2, t_3 . t = ((t_1, t_2), t_3) \\
& \wedge \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2) \wedge \mathrm{R}_3(t_3, \bar{x}_3)\} \\
\preceq_{\mathrm{R}''} \;=\;& \{(t_1, t_2) \mid \exists t_{11}, \bar{x}_{11} . \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists t_{12}, \bar{x}_{12} . \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{13}, \bar{x}_{13} . \mathrm{R}_3(t_{13}, \bar{x}_{13}) \\
& \wedge \exists t_{21}, \bar{x}_{21} . \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists t_{22}, \bar{x}_{22} . \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge \exists t_{23}, \bar{x}_{23} . \mathrm{R}_3(t_{23}, \bar{x}_{23}) \\
& \wedge\; t_1 = ((t_{11}, t_{12}), t_{13}) \wedge t_2 = ((t_{21}, t_{22}), t_{23}) \\
& \wedge\; ((t_{11} \preceq_1 Rt_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_{R_2} t_{22})) \\
& \vee (t_{11} = t_{21} \wedge t_{12} = t_{22} \wedge t_{13} \preceq_{R_3} t_{23}))\}\rangle.
\end{aligned}
$$

$$
\begin{aligned}
= \{(t_1, t_2) \mid\ & \exists t_{11}, \bar{x}_{11}.\mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists t_{12}, \bar{x}_{12}.\mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \wedge \exists t_{13}, \bar{x}_{13}.\mathrm{R}_3(t_{13}, \bar{x}_{13}) \\
& \wedge \exists t_{21}, \bar{x}_{21}.\mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists t_{22}, \bar{x}_{22}.\mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \wedge \exists t_{23}, \bar{x}_{23}.\mathrm{R}_3(t_{23}, \bar{x}_{23}) \\
& \wedge\ t_1 = (t_{11}, (t_{12}, t_{13})) \wedge t_2 = (t_{21}, (t_{22}, t_{23})) \\
& \wedge\ (t_{11} \preceq_1 Rt_{21} \vee (t_{11} = t_{21} \\
& \wedge\ (t_{12} \preceq_{R_2} t_{22} \vee (t_{12} = t_{22} \wedge t_{13} \preceq_{R_3} t_{23}))))\}\rangle.
\end{aligned}
$$

Consequently,

$$
\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle \ltimes (\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle \ltimes \langle \mathrm{R}_3, \preceq_{\mathrm{R}_3} \rangle).
$$

($R8$) Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 - \mathbf{R}_2$. By the definition of set difference,

$$
\begin{aligned}
\mathrm{R}' \ =\ & \{(t, \bar{x}) \mid\ \ \mathrm{R}_1(t, \bar{x}) \wedge \neg\ \exists t'.\mathrm{R}_2(t', \bar{x})\}
\end{aligned}
$$

$$
\begin{aligned}
\preceq_{\mathrm{R}'} \ =\ & \{(t_1, t_2) \mid\ (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\ \exists\ t_1'\mathrm{R}_2(t_1', \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\ \exists\ t_2'\mathrm{R}_2(t_2', \bar{x}_2)) \\
& \wedge\ t_1 \preceq_{R_1} t_2\}
\end{aligned}
$$

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}}\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$
\begin{aligned}
\mathrm{R}'' \ =\ & \{(t, \bar{x}) \mid\ \ \mathrm{R}_1(t, \bar{x}) \wedge \neg\ \exists t'.\mathrm{R}_2(t', \bar{x}) \wedge \mathrm{p}(t, \bar{x})\}
\end{aligned}
$$

$$
\begin{aligned}
\preceq_{\mathrm{R}''} \ =\ & \{(t_1, t_2) \mid\ (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \neg\ \exists\ t_1'\mathrm{R}_2(t_1', \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \neg\ \exists\ t_2'\mathrm{R}_2(t_2', \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2)) \\
& \wedge\ t_1 \preceq_{R_1} t_2\}.
\end{aligned}
$$

Hence,

$$
\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}}(\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle) - \langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle.
$$

($R9$)

$$
\begin{aligned}
\sigma_{\mathrm{p}}(\mathbf{R}_1 - \mathbf{R}_2) \ =\ & \langle \{(t, \bar{x}) \mid \mathrm{R}_1(t, \bar{x}) \wedge \neg \mathrm{R}_2(t, \bar{x}) \wedge \mathrm{p}(t, \bar{x})\}, \\
& \{(t_1, t_2) \mid \exists \bar{x}_1.R_1(t_1, \bar{x}_1) \wedge \neg \mathrm{R}_2(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
& \wedge\ \exists \bar{x}_2.R_1(t_2, \bar{x}_2) \wedge \neg \mathrm{R}_2(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \\
& \wedge\ t_1 \preceq_{\mathrm{R}_1} t_2\}\rangle.
\end{aligned}
$$

Simultaneously,

$$
\begin{aligned}
\sigma_{\mathrm{p}}(\mathbf{R}_1) - \sigma_{\mathrm{p}}(\mathbf{R}_2) &= \langle \{(t,\bar{x}) \mid \mathrm{R}_1(t,\bar{x}) \wedge \mathrm{p}(t,\bar{x}) \wedge \neg(\mathrm{R}_2(t,\bar{x}) \wedge \mathrm{p}(t,\bar{x}))\}, \\
&\quad \{(t_1,t_2) \mid \exists \bar{x}_1.R_1(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \wedge \neg(\mathrm{R}_2(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1)) \\
&\quad \wedge \exists \bar{x}_2.R_1(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \wedge \neg(\mathrm{R}_2(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2)) \\
&\quad \wedge t_1 \preceq_{\mathrm{R}_1} t_2\}\rangle \\
&= \langle \{(t,\bar{x}) \mid \mathrm{R}_1(t,\bar{x}) \wedge \mathrm{p}(t,\bar{x}) \wedge (\neg\mathrm{R}_2(t,\bar{x}) \vee \neg\mathrm{p}(t,\bar{x}))\}, \\
&\quad \{(t_1,t_2) \mid \exists \bar{x}_1.R_1(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \wedge (\neg\mathrm{R}_2(t_1,\bar{x}_1) \vee \neg\mathrm{p}(t_1,\bar{x}_1)) \\
&\quad \wedge \exists \bar{x}_2.R_1(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \wedge (\neg\mathrm{R}_2(t_2,\bar{x}_2) \vee \neg\mathrm{p}(t_2,\bar{x}_2)) \\
&\quad \wedge t_1 \preceq_{\mathrm{R}_1} t_2\}\rangle \\
&= \langle \{(t,\bar{x}) \mid (\mathrm{R}_1(t,\bar{x}) \wedge \mathrm{p}(t,\bar{x}) \wedge \neg\mathrm{R}_2(t,\bar{x})) \\
&\quad \vee (\mathrm{R}_1(t,\bar{x}) \wedge \mathrm{p}(t,\bar{x}) \wedge \neg\mathrm{p}(t,\bar{x}))\}, \\
&\quad \{(t_1,t_2) \mid \exists \bar{x}_1.(R_1(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \wedge \neg\mathrm{R}_2(t_1,\bar{x}_1)) \\
&\quad \vee (R_1(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \wedge \neg\mathrm{p}(t_1,\bar{x}_1)) \\
&\quad \wedge \exists \bar{x}_2.(R_1(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \wedge \neg\mathrm{R}_2(t_2,\bar{x}_2)) \\
&\quad \vee (R_1(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \wedge \neg\mathrm{p}(t_2,\bar{x}_2)) \\
&\quad \wedge t_1 \preceq_{\mathrm{R}_1} t_2\}\rangle \\
&= \langle \{(t,\bar{x}) \mid (\mathrm{R}_1(t,\bar{x}) \wedge \mathrm{p}(t,\bar{x}) \wedge \neg\mathrm{R}_2(t,\bar{x}))\}, \\
&\quad \{(t_1,t_2) \mid \exists \bar{x}_1.(R_1(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \wedge \neg\mathrm{R}_2(t_1,\bar{x}_1)) \\
&\quad \wedge \exists \bar{x}_2.(R_1(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \wedge \neg\mathrm{R}_2(t_2,\bar{x}_2)) \\
&\quad \wedge t_1 \preceq_{\mathrm{R}_1} t_2\}\rangle.
\end{aligned}
$$

Hence,

$$\sigma_{\mathrm{p}}(\mathbf{R}_1 - \mathbf{R}_2) = \sigma_{\mathrm{p}}(\mathbf{R}_1) - \sigma_{\mathrm{p}}(\mathbf{R}_2).$$

($R10$) Assume that $\mathbf{R}_1$ and $\mathbf{R}_2$ share the same schema $(\mathrm{T}, X_1, \ldots, \mathrm{X}_n)$. Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \cup \mathbf{R}_2$. By definition of order concatenation,

$$
\begin{aligned}
\mathrm{R}' &= \{(t,\bar{x}) \mid \quad \mathrm{R}_1(t,\bar{x}) \vee \mathrm{R}_2(t,\bar{x})\} \\
\\
\preceq_{\mathrm{R}'} &= \{(t_1,t_2) \mid \quad (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2)) \\
&\qquad\qquad \vee\ (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge\ \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \\
&\qquad\qquad \wedge\ t_1 \preceq_{R_1} t_2) \\
&\qquad\qquad \vee\ (\exists \bar{x}_1.\mathrm{R}_2(t_1,\bar{x}_1) \wedge\ \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2) \\
&\qquad\qquad \wedge\ t_1 \preceq_{R_2} t_2)\},
\end{aligned}
$$

where $\bar{x}_1 = (x_{11}, \ldots, x_{1n})$.

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}}\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$
\begin{aligned}
\mathrm{R}'' &= \{(t,\bar{x}) \mid \mathrm{R}'(t,\bar{x}) \wedge \mathrm{p}(t,\bar{x})\} \\
&= \{(t,\bar{x}) \mid (\mathrm{R}_1(t,\bar{x}) \vee \mathrm{R}_2(t,\bar{x})) \wedge \ \mathrm{p}(t,\bar{x})\} \\
&= \{(t,\bar{x}) \mid (\mathrm{R}_1(t,\bar{x}) \wedge \ \mathrm{p}(t,\bar{x}) \vee (\mathrm{R}_2(t,\bar{x}) \wedge \ \mathrm{p}(t,\bar{x})\}
\end{aligned}
$$

$$
\begin{aligned}
\preceq_{\mathrm{R}''} &= \{(t_1,t_2) \mid \exists \bar{x}_1.\mathrm{R}'(t_1,\bar{x}_1) \wedge \ \mathrm{p}(t_1,\bar{x}_1) \\
&\quad \wedge \exists \bar{x}_2.\mathrm{R}'(t_2,\bar{x}_2) \wedge \ \mathrm{p}(t_2,\bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}'} t_2\} \\
&= ((\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1)) \\
&\quad \wedge (\exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2))) \\
&\quad \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \\
&\quad \wedge t_1 \preceq_{R_1} t_2) \\
&\quad \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \\
&\quad \wedge t_1 \preceq_{R_2} t_2)\}.
\end{aligned}
$$

Therefore,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}}(\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle) \cup \sigma_{\mathrm{p}}(\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle).$$

($R11$) Without loss of generality, we assume that $\mathbf{R}_1$ and $\mathbf{R}_2$ share the same schema $(\mathrm{T}, X_1, \ldots, \mathrm{X}_n)$, and $A = (\mathrm{T}, X_1, \ldots, \mathrm{X}_k)$, where $k \leq n$. Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \cup \mathbf{R}_2$. By definition of order concatenation,

$$
\begin{aligned}
\mathrm{R}' &= \{(t,\bar{x}) \mid \quad \mathrm{R}_1(t,\bar{x}) \vee \mathrm{R}_2(t,\bar{x})\}
\end{aligned}
$$

$$
\begin{aligned}
\preceq_{\mathrm{R}'} = \{(t_1,t_2) \mid \ & (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2)) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \wedge t_1 \preceq_{R_1} t_2) \\
& \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2) \wedge t_1 \preceq_{R_2} t_2)\},
\end{aligned}
$$

where $\bar{x}_1 = (x_{11}, \ldots, x_{1n})$.

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \pi_A \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$
\begin{aligned}
\mathrm{R}'' &= \{(t,x_1,\ldots,x_k) \mid \exists x_{k+1},\ldots,x_n.\mathrm{R}'(t,\bar{x})\} \\
&= \{(t,x_1,\ldots,x_k) \mid \exists x_{k+1},\ldots,x_n.\mathrm{R}_1(t,\bar{x}) \vee \mathrm{R}_2(t,\bar{x})\} \\
&= \{(t,x_1,\ldots,x_k) \mid \exists x_{k+1},\ldots,x_n.\mathrm{R}_1(t,\bar{x}) \vee \exists x_{k+1},\ldots,x_n.\mathrm{R}_2(t,\bar{x})\} \\
\preceq_{\mathrm{R}''} &= \{(t_1,t_2) \mid \exists \bar{x}_1.\mathrm{R}'(t_1,\bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}'(t_2,\bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}'} t_2\} \\
&= \{(t_1,t_2) \mid (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \\
&\quad \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2)) \\
&\quad \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \wedge t_1 \preceq_{R_1} t_2) \\
&\quad \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1,\bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2,\bar{x}_2) \wedge t_1 \preceq_{R_2} t_2)\}.
\end{aligned}
$$

Thus,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \pi_A(\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle) \cup \pi_A(\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle).$$

($R12$) Assume that $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{R}_3$ share the same schema $(\mathrm{T}, X_1, \ldots, \mathrm{X}_n)$. Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \cup \mathbf{R}_2$. By definition of order concatenation,

$$\begin{array}{rcll}
\mathrm{R}' & = & \{(t, \bar{x}) \mid & \mathrm{R}_1(t, \bar{x}) \vee \mathrm{R}_2(t, \bar{x})\} \\
\\
\preceq_{\mathrm{R}'} & = & \{(t_1, t_2) \mid & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2)) \\
& & & \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_1} t_2) \\
& & & \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_2} t_2)\},
\end{array}$$

where $\bar{x}_1 = (x_{11}, \ldots, x_{1n})$.

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle \cup \mathbf{R}_3$. Then,

$$\begin{array}{rcl}
\mathrm{R}'' & = & \{(t, \bar{x}) \mid \mathrm{R}'(t, \bar{x}) \vee \mathrm{R}_3(t, \bar{x})\} \\
& = & \{(t, \bar{x}) \mid \mathrm{R}_1(t, \bar{x}) \vee \mathrm{R}_2(t, \bar{x}) \vee \mathrm{R}_3(t, \bar{x})\} \\
\\
& = & \{(t, \bar{x}) \mid \mathrm{R}_1(t, \bar{x}) \vee (\mathrm{R}_2(t, \bar{x}) \vee \mathrm{R}_3(t, \bar{x}))\} \\
\preceq_{\mathrm{R}''} & = & (\exists \bar{x}_1.\mathrm{R}'(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2)) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}'(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}'(t_2, \bar{x}_2) \wedge t_1 \preceq_{R'} t_2) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_3(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_3} t_2)\} \\
& = & ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \vee \mathrm{R}_2(t_1, \bar{x}_1)) \wedge \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2)) \\
& & \vee ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2)) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_1} t_2) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_2} t_2)) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_3(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_3} t_2)\} \\
& = & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2)) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2)) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2)) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_1} t_2) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_2} t_2)) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_3(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_3} t_2)\} \\
& = & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge (\exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \vee \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2))) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_1} t_2)) \\
& & \vee ((\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2)) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_2(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_2(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_2} t_2)) \\
& & \vee (\exists \bar{x}_1.\mathrm{R}_3(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}_3(t_2, \bar{x}_2) \wedge t_1 \preceq_{R_3} t_2))\}.
\end{array}$$

Therefore,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle \cup (\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle \cup \langle \mathrm{R}_3, \preceq_{\mathrm{R}_3} \rangle).$$

This completes the proof of the transformation rule ($R12$).

($R13$) Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \nu \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. By definition of the order reverse operation,

$$\begin{array}{rcl} \mathrm{R}' & = & \{(t, \bar{x}) | \mathrm{R}(t, \bar{x})\} \\ \preceq_{\mathrm{R}'} & = & \{(t_1, t_2) | \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge t_2 \preceq_R t_1\}. \end{array}$$

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}} \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$\begin{array}{rcl} \mathrm{R}'' & = & \{(t, \bar{x}) \mid \mathrm{R}'(t, \bar{x}) \wedge \mathrm{p}(t, \bar{x})\} \\ & = & \{(t, \bar{x}) \mid \mathrm{R}(t, \bar{x}) \wedge \mathrm{p}(t, \bar{x})\} \end{array}$$

$$\begin{array}{rcl} \preceq_{\mathrm{R}''} & = & \{(t_1, t_2) \mid \exists \bar{x}_1.\mathrm{R}'(t_1, \bar{x}_1) \wedge \ \mathrm{p}(t_1, \bar{x}_1) \\ & & \wedge \exists \bar{x}_2.\mathrm{R}'(t_2, \bar{x}_2) \wedge \ \mathrm{p}(t_2, \bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}'} t_2\} \\ & = & \{(t_1, t_2) \mid \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\ & & \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \wedge t_2 \preceq_{\mathrm{R}} t_1\}. \end{array}$$

Hence,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \nu(\sigma_{\mathrm{p}} \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle).$$

($R14$) Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \nu \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. By definition of the order reverse operation,

$$\begin{array}{rcl} \mathrm{R}' & = & \{(t, \bar{x}) | \mathrm{R}(t, \bar{x})\} \end{array}$$

$$\begin{array}{rcl} \preceq_{\mathrm{R}'} & = & \{(t_1, t_2) | \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge t_2 \preceq_R t_1\}. \end{array}$$

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \pi_A \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$\begin{array}{rcl} \mathrm{R}'' & = & \{(t, x_1, \ldots, x_k) \mid \exists x_{k+1}, \ldots, x_n.\mathrm{R}'(t, \bar{x})\} \\ & = & \{(t, x_1, \ldots, x_k) \mid \exists x_{k+1}, \ldots, x_n.\mathrm{R}(t, \bar{x})\} \end{array}$$

$$\begin{array}{rcl} \preceq_{\mathrm{R}''} & = & \{(t_1, t_2) \mid \exists \bar{x}_1.\mathrm{R}'(t_1, \bar{x}_1) \\ & & \wedge \exists \bar{x}_2.\mathrm{R}'(t_2, \bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}'} t_2\} \\ & = & \{(t_1, t_2) \mid \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \\ & & \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \wedge t_2 \preceq_{\mathrm{R}} t_1\}. \end{array}$$

Hence,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \nu(pi_A \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle).$$

($R15$) Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \ltimes \mathbf{R}_2$. By definition of the left nested-loop product,

$$\begin{aligned}
\mathrm{R}' \;=\; & \{(t, \bar{x}_1, \bar{x}_2) \mid \; t = (t_1, t_2) \wedge \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2)\} \\
\preceq_{\mathrm{R}'} \;=\; & \{(t_1, t_2) \mid \quad \exists t_{11}, t_{12}, \bar{x}_{11}, \bar{x}_{12}. t_1 = (t_{11}, t_{12}) \wedge \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \qquad \wedge \exists t_{21}, t_{22}, \bar{x}_{21}, \bar{x}_{22}. t_2 = (t_{21}, t_{22}) \wedge \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \qquad \wedge (t_{11} \preceq_R t_{21} \vee (t_{11} = t_{21} \wedge t_{12} \preceq_R t_{22}))\}\rangle.
\end{aligned}$$

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \nu\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$\begin{aligned}
\mathrm{R}'' \;=\; & \{((t_1, t_2), \bar{x}_1, \bar{x}_2) \mid \quad \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_2, \bar{x}_2)\} \\
\preceq_{\mathrm{R}''} \;=\; & \{((t_{11}, t_{12}), (t_{21}, t_{22})) \mid \; \exists \bar{x}_{11}. \mathrm{R}_1(t_{11}, \bar{x}_{11}) \wedge \exists \bar{x}_{12}. \mathrm{R}_2(t_{12}, \bar{x}_{12}) \\
& \qquad \wedge \exists \bar{x}_{21}. \mathrm{R}_1(t_{21}, \bar{x}_{21}) \wedge \exists \bar{x}_{22}. \mathrm{R}_2(t_{22}, \bar{x}_{22}) \\
& \qquad \wedge (t_{21} \preceq_R t_{11} \vee (t_{11} = t_{21} \wedge t_{22} \preceq_R t_{12}))\}\rangle.
\end{aligned}$$

Thus,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \nu(\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle) \ltimes \nu(\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle).$$

□

3.4 Completeness of $\boldsymbol{CA}^X_\preceq$

This section examines the expressive power of the ordered conjunctive algebra $\boldsymbol{CA}^X_\preceq$. As in the conventional relational databases, the expressive power of $\boldsymbol{CA}^X_\preceq$ is evaluated with respect to the first-order calculus. We will prove that $\boldsymbol{CA}^X_\preceq$ is complete: any ordered conjunctive query in $\boldsymbol{CQ}^X_\preceq$ can be expressed by a finite sequence of ordered operations from $\boldsymbol{CA}^X_\preceq$.

First, a number of definitions and notations are provided, which will be used in proofs later. Let $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ be an arbitrary ordered relation and p be an arbitrary data predicate in form of $x_i = a$. The data predicate p separates the ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ into two *partitions*: the set of tuples satisfying p and the set of tuples satisfying $\neg p$. Furthermore, let $\{p_1, \ldots, p_k\}$ be the only k data predicates in the order query ψ of an ordered query $\langle \varphi, \psi \rangle$ on $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. Then, data predicates $\{p_1, \ldots, p_k\}$ separate $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ into 2^k partitions $\{P_j \mid j = 1, ..., 2^k\}$; each partition P_j is a set of tuples satisfying a unique conjunction of data predicates $\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k)$.

Here, $\epsilon(p_i)$ is either p_i or $\neg p_i$ for $i = 1, \ldots, k$. We say that the partitions $\{P_j\}$ are specified by predicates $p_1, \ldots, p_k$, or specified by the formula ψ.

The partitions $\{P_j\}$ of an ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, specified by data predicates $\{p_1, \ldots, p_k\}$, possess the following properties:

(1) The union of all partitions $\{P_j\}$ is exactly the data relation R:

$$\bigcup_{j=1,\ldots,2^k} P_j = \mathrm{R}.$$

(2) The partitions $\{P_j\}$ are pairwise disjoint, *i.e.*, the intersection of any two partitions is an empty set:

$$P_i \cap P_j = \emptyset,$$

for $i, j = 1, ..., 2^k$ and $i \neq j$.

Definition 3.14 (Minimal Partitions) *Let $\{P_j\}$ be the set of partitions in an ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, specified by predicates $\{p_1, \ldots, p_k\}$ in a formula ψ. The partitions $\{P_j\}$ are* minimal partitions *specified by ψ if there does not exist a predicate p_{k+1} in ψ such that $p_{k+1} \neq p_i$ for $i = 1, ..., k$.*

By this definition, minimal partitions determined by ψ cannot be separated into smaller partitions because there is no extra predicate in ψ. Minimal partitions will play an important role in the proof of Completeness Theorem later. Next, we define combinations of partitions for multiple ordered relations.

Definition 3.15 (Combinations of Partitions) *Let $\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle, \ldots, \langle \mathrm{R}_n, \preceq_{\mathrm{R}_n} \rangle$ be n ordered relations, and P_i be an arbitrary partition of $\langle \mathrm{R}_i, \preceq_{\mathrm{R}_i} \rangle$ for $i = 1, ..., n$. A combination of partitions $(P_1, \ldots, P_n)$ is a set of the tuples, each of which is a combination of tuples from $P_1, \ldots, P_n$, respectively.*

Suppose that $(t_i, \bar{x}_i)$ is an arbitrary tuple in the partition P_i of the ordered relation $\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle$, a combination of $(t_i, \bar{x}_i)$ for $i = 1, ..., n$ has the following form:

$$((t_1, ..., t_n), \bar{x}_1, ..., \bar{x}_n).$$

Essentially, a combination of partitions $(P_1, \ldots, P_n)$ from $\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle, \ldots, \langle \mathrm{R}_n, \preceq_{\mathrm{R}_n} \rangle$ is a partition of $\mathrm{R}_1 \times \ldots \times \mathrm{R}_n$. We now define the order of partitions in an ordered relation.

Definition 3.16 (The Order of Partitions) *Let P_1 and P_2 be two partitions of $\langle R, \preceq_R \rangle$. We say $P_1 \preceq_{\mathrm{R}} P_2$ if $t_1 \preceq_{\mathrm{R}} t_2$, for all tuples $t_1 \in P_1$ and $t_2 \in P_2$.*

Then, we prove a lemma for a special case of $\boldsymbol{CQ}^X_\preceq$, the ω queries in $\boldsymbol{CQ}^X_\preceq$. An ω query in $\boldsymbol{CQ}^X_\preceq$ generates an ordered relation whose data relation is identical to the data relation of the input. We first define the general ω queries as a subset of ordered relational queries.

Definition 3.17 (ω Queries) *An ordered relational query $\langle\varphi,\psi\rangle$ is an ω query if it has only one argument ordered relation $\langle R,\preceq_R\rangle$, and the data query φ is also an identity function ω,* i.e., *$\omega(R)=R$. We denote the set of all ω queries on the ordered relation $\langle R,\preceq_R\rangle$ by $\omega(\langle R,\preceq_R\rangle)$.*

Intuitively, $\omega(\langle R,\preceq_R\rangle)$ represents all first-order definable permutations on the ordered relation $\langle R,\preceq_R\rangle$. In the proof of the following lemma and the rest of this dissertation, the form $\bigvee_k \psi^k$ (k is a natural number) is used to indicate a disjunction of a finite number of formulas.

Lemma 3.18 *Any ω query in $\boldsymbol{CQ}^X_\preceq$ is equivalent to an ordered query expressed by a finite sequence of ordered operations from $\boldsymbol{CA}^X_\preceq$.*

Proof:
Let $\langle\varphi,\psi\rangle$, a $\boldsymbol{CQ}^X_\preceq$ query on $\langle \mathrm{R},\preceq_\mathrm{R}\rangle$, be an ω query, and $\langle \mathrm{R}',\preceq_{\mathrm{R}'}\rangle$ be the output ordered relation.

By definition of the ω query, $\langle\varphi,\psi\rangle$ has the form of

$$\langle\varphi,\psi\rangle(\langle R,\preceq_R\rangle)=\langle\{(t,\bar{x})|\mathrm{R}(t,\bar{x})\},\ \{(t_1,t_2)|\psi(t_1,t_2)\}\rangle,$$

and

$$\psi=\exists\bar{x}_1.\mathrm{R}(t_1,\bar{x}_1)\wedge\exists\bar{x}_2.\mathrm{R}(t_2,\bar{x}_2)\wedge\psi'(t_1,\bar{x}_1,t_2,\bar{x}_2).$$

By the definition of $\boldsymbol{CQ}^X_\preceq$, ψ' is a first-order formula defined by the BNF rule

$$\psi' \ = \ t_1\preceq_R t_2 \mid x_i=x_j \mid \neg\psi' \mid \psi'\wedge\psi'.$$

We can transform ψ' into an equivalent query in disjunctive normal form

$$\psi' \ = \ \bigvee_k \psi^k(t_1,\bar{x}_1,t_2,\bar{x}_2)$$

Since there is no existential quantifier on $\boldsymbol{t}$-variables in ψ', t_1 and t_2 are the only two $\boldsymbol{t}$-variables in ψ', occurring only in forms $t_1\preceq_R t_2$ and $t_2\preceq_R t_1$. As a consequence, we can separate ψ' into two parts

$$\psi'=\bigvee_i \alpha^i(\bar{x}_1,\bar{x}_2)\vee\ \bigvee_j \psi^j(t_1,\bar{x}_1,t_2,\bar{x}_2),$$

where, α^is and ψ^js are conjunctions of atoms or negations of atoms. Order predicates $t_1 \preceq_R t_2$ and $t_2 \preceq_R t_1$ appear only in ψ^js and not at all in α^is. It follows that α^is are conjunctions of data predicates in form of $x_i = x_j$, and ψ^js are conjunctions of data predicates $x_i = x_j$, and also of order predicates $t_1 \preceq_R t_2$ and $t_2 \preceq_R t_1$.

First, we look at the first part of ψ', which is $\bigvee_i \alpha^i(\bar{x}_1, \bar{x}_2)$. Since $\bigvee_i \alpha^i$ does not contain $\boldsymbol{t}$-variables, each α^i is a conjunction of data predicates. Let $p_1, \ldots, p_k$ be k data predicates in ψ', and $\epsilon(p_j)$ is either p_j or $\neg p_j$ for $j = 1, \ldots, k$. We will show that $\bigvee_i \alpha^i$ is equivalent to a normal form, $\bigvee_j \alpha'^j$, and

$$\alpha'^j(\bar{x}_1, \bar{x}_2) = (\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k))(\bar{x}_1) \wedge \ (\epsilon'(p_1) \wedge \ldots \wedge \epsilon'(p_k))(\bar{x}_2),$$

where $\epsilon(p_i)$ and $\epsilon'(p_i)$ are either p_i or $\neg p_i$ for $i = 1, \ldots, k$.

This is done by splitting each conjunction $\alpha^i(\bar{x}_1, \bar{x}_2)$ further if possible. Suppose that

$$\alpha^i(\bar{x}_1, \bar{x}_2) = \alpha^i_1(\bar{x}_1) \wedge \ \alpha^i_2(\bar{x}_2).$$

Let $p \in \{p_1, ..., p_k\}$ be a data predicate in ψ'. If neither $p(\bar{x}_1)$ nor $\neg p(\bar{x}_1)$ is in $\alpha^i_1(\bar{x}_1)$, then

$$\begin{aligned} \alpha^i(\bar{x}_1, \bar{x}_2) = \ & ((\alpha^i_1 \wedge p)(\bar{x}_1) \wedge \ \alpha^i_2(\bar{x}_2)) \\ & \vee \ ((\alpha^i_1 \wedge \neg p)(\bar{x}_1) \wedge \ \alpha^i_2(\bar{x}_2)). \end{aligned}$$

If neither $p(\bar{x}_2)$ nor $\neg p(\bar{x}_2)$ is in $\alpha^i_2(\bar{x}_1)$, then

$$\begin{aligned} \alpha^i(\bar{x}_1, \bar{x}_2) = \ & (\alpha^i_1(\bar{x}_1) \wedge \ (\alpha^i_2 \wedge p)(\bar{x}_2)) \\ & \vee \ (\alpha^i_1(\bar{x}_1) \wedge \ (\alpha^i_2 \wedge \neg p)(\bar{x}_2)). \end{aligned}$$

For each $p \in \{p_1, ..., p_k\}$ not in α^i_1 or α^i_2, we repeat the above separation procedure, until we achieve the normal form of $\bigvee_j \alpha'^j$ with each α'^j in form of

$$(\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k))(\bar{x}_1) \wedge \ (\epsilon'(p_1) \wedge \ldots \wedge \epsilon'(p_k))(\bar{x}_2).$$

Let $\{P_1, ..., P_{2^k}\}$ be the partitions specified by these data predicates $\{p_1, ..., p_k\}$ in α'^j. Each partition P_i is specified by a conjunction of data predicates $\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k)$. In addition to $\{p_1, ..., p_k\}$, no other predicates exist in ψ'. Therefore, the partition specified by data predicates in each conjunct α'^j is minimal.

Since $\bar{x}_1$ and $\bar{x}_2$ are the corresponding data attributes of t_1 and t_2 respectively, if $\bar{x}_1$ and $\bar{x}_2$ satisfy any conjunct α'^j, then (t_1, t_2) will be in the resulting order relation $\preceq_{R'}$, *i.e.*, $t_1 \ \preceq_{R'} t_2$. Let P_1 and P_2 be the two partitions whose tuples satisfy data predicates in α'^j respectively. By Definition 3.16, $P_1 \preceq_{R'} P_2$. We declare that α'^j decides the order between P_1 and P_2.

Next, we show that $\bigvee_j \alpha'^j$ defines a linear order of the partitions $\{P_1, ..., P_{2^k}\}$. This is proved by contradiction. Suppose that $\bigvee_j \alpha'^j$ does not define a linear order among the partitions $\{P_1, ..., P_{2^k}\}$, there are two cases.

In the first case, assume that there exist two partitions, P_1 and P_2, such that both $P_1 \preceq_{R'} P_2$ and $P_2 \preceq_{R'} P_1$ hold at the same time. If $t_1 \in P_1 \wedge t_2 \in P_2$, then by assumption $(t_1, t_2) \in \preceq_{R'}$ and $(t_2, t_1) \in \preceq_{R'}$ hold at the same time. This is a contradiction to the fact that the output order $\preceq_{R'}$ is a linear order. Hence, the assumption in Case 1 does not hold.

In the second case, assume that there exist two partitions, P_1 and P_2, such that neither $P_1 \preceq_{R'} P_2$ nor $P_2 \preceq_{R'} P_1$ holds, that is, their order in output is not specified by $\bigvee_j \alpha'^j$.

By this assumption, for any pair of tuples t_1 and t_2 such that $t_1 \in P_1 \wedge t_2 \in P_2$, their order in the output is not defined by $\bigvee_j \alpha'^j$. Because the output order is a linear order on all tuples, the order of t_1 and t_2 must be decided by ψ'. Suppose that ψ' decides $t_1 \preceq_{R'} t_2$. Since P_1 and P_2 are minimal, *i.e.*, they are the smallest partitions that data predicates in ψ' can specify, all pairs of tuples from P_1 and P_2 respectively must have the same order if they have any order at all. Then, for any pair of tuples t_1 and t_2, and $t_1 \in P_1 \wedge t_2 \in P_2$, we have $t_1 \preceq_{R'} t_2$. This follows that $P_1 \preceq_{R'} P_2$, which is a contradiction to the assumption.

We have proven that the first part of ψ', $\bigvee_i \alpha^i$, in the normal form of ψ' defines a linear order of minimal partitions $\{P_1, ..., P_{2^k}\}$, which are specified by the set of data predicates $\{p_1, ..., p_k\}$ in ψ'.

We now consider the second part in the normal form of ψ', which is $\bigvee_j \psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$. Since ψ' defines a linear order on all tuples in $\langle R, \preceq_R \rangle$, for any pair of tuples t_1 and t_2 from each partition P'_i, there must exist a m such that the conjunction $\psi^m(t_1, \bar{x}_1, t_2, \bar{x}_2)$ is true.

Recall that the only $\boldsymbol{t}$-variables in ψ' are t_1 and t_2, and that they occur only in forms $t_1 \preceq_R t_2$ and $t_2 \preceq_R t_1$ in each conjunction $\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$. If ψ^m contains $t_1 \preceq_R t_2$, then the tuples in P'_i keep the original order; if ψ^m contains $t_2 \preceq_R t_1$, then the tuples in P'_i reverse the original order.

Since ψ' is a first-order formula defined by the BNF rule

$$\psi' \ = \ t_1 \preceq_R t_2 \mid x_i = x_j \mid \neg\psi' \mid \psi' \wedge \psi',$$

we know that in $\bigvee_j \psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$, each conjunction $\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$ must have a form

$$\alpha_1(\bar{x}_1) \wedge \alpha_2(\bar{x}_2) \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2),$$

where α_1 and α_2 are conjunctions of data predicates on $\bar{x}_1$ and $\bar{x}_2$ respectively, and $\epsilon(t_1 \preceq_{\mathrm{R}} t_2)$ is either $t_1 \preceq_{\mathrm{R}} t_2$ or $t_2 \preceq_{\mathrm{R}} t_1$.

Next, we will show that $\alpha_1 = \alpha_2$ in each conjunction $\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$. Assume that α_1 and α_2 are different conjunctions of data predicates. We consider the case that $\epsilon(t_1 \preceq_{\mathrm{R}} t_2)$ is $t_1 \preceq_{\mathrm{R}} t_2$ in an arbitrary conjunction $\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$. For any two tuples t_1 and t_2 that satisfy α_1 and α_2, and also $t_2 \preceq_{\mathrm{R}} t_1$, they satisfy $\alpha_1(x_1) \wedge \alpha_2(x_2)$, which means that $t_1 \preceq_{\mathrm{R}'} t_2$. Meanwhile, $t_2 \preceq_{\mathrm{R}} t_1$ implies that $t_2 \preceq_{\mathrm{R}'} t_1$. This is a contradiction. We can prove that this assumption also leads to a contradiction in the case that $\epsilon(t_1 \preceq_{\mathrm{R}} t_2)$ is $t_2 \preceq_{\mathrm{R}} t_1$. Therefore, the assumption does not hold, and $\alpha_1 = \alpha_2$ in each conjunction $\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$.

Thus, each conjunction $\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$ must have a form

$$\alpha(\bar{x}_1) \wedge \alpha(\bar{x}_2) \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2),$$

where α is a conjunction of data predicates on x_1 and x_2 both, and $\epsilon(t_1 \preceq_{\mathrm{R}} t_2)$ is either $t_1 \preceq_{\mathrm{R}} t_2$ or $t_2 \preceq_{\mathrm{R}} t_1$.

Let $p_1, \ldots, p_k$ be the k data predicates in ψ', $\epsilon(p_j)$ be either p_j or $\neg p_j$ for $j = 1, \ldots, n$. We show that $\bigvee_j \psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$ is equivalent to

$$\bigvee_i \psi^i(t_1, \bar{x}_1, t_2, \bar{x}_2),$$

where $\psi^i(t_1, \bar{x}_1, t_2, \bar{x}_2)$ has a normal form

$$(\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k))(\bar{x}_1) \wedge \ (\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k))(\bar{x}_2) \wedge \ \epsilon(t_1 \preceq_{\mathrm{R}} t_2).$$

As shown above that in $\bigvee_j \psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$, any conjunction $\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$ has a form of

$$\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2) = \alpha(\bar{x}_1) \wedge \alpha(\bar{x}_2) \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2).$$

Suppose that $\alpha = \epsilon(p'_1) \wedge \epsilon(p'_2) \wedge \ldots \wedge \epsilon(p'_m)$ and $\{p'_1, \ldots, p'_m\} \subset \{p_1, \ldots, p_k\}$. Assume that $p_i \in \{p_1, \ldots, p_k\}$; however, $p_i \notin \{p'_1, \ldots, p'_m\}$. Then,

$$\begin{aligned}
\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2) &= \alpha(\bar{x}_1) \wedge \alpha(\bar{x}_2) \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2) \\
&= (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m))(\bar{x}_1) \\
&\quad \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m))(\bar{x}_2) \\
&\quad \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)
\end{aligned}$$

$$
\begin{aligned}
= \;& (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge (p_i \vee \neg p_i))(\bar{x}_1) \\
& \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge (p_i \vee \neg p_i))(\bar{x}_2) \\
& \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2) \\
= \;& ((\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i) \\
& \vee (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i))(\bar{x}_1) \\
& \wedge ((\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i) \\
& \vee (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i))(\bar{x}_2) \\
& \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2) \\
= \;& ((\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i)(\bar{x}_1) \\
& \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i)(\bar{x}_2) \\
& \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)) \\
& \vee \\
& ((\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i)(\bar{x}_1) \\
& \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i)(\bar{x}_2) \\
& \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)) \\
& \vee \\
& ((\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i)(\bar{x}_1) \\
& \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i)(\bar{x}_2) \\
& \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)) \\
& \vee \\
& ((\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i)(\bar{x}_1) \\
& \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i)(\bar{x}_2) \\
& \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)).
\end{aligned}
$$

As we proved earlier, for any conjunction $\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2) = \alpha_1(\bar{x}_1) \wedge \alpha_2(\bar{x}_2) \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)$, $\alpha_1 \neq \alpha_2$; otherwise, this conjunction is false. Therefore, in the equation above, the conjunctions $(\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i)(\bar{x}_1) \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i)(\bar{x}_2) \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)$ and $(\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i)(\bar{x}_1) \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i)(\bar{x}_2) \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)$ are always false.

Thus,

$$
\begin{aligned}
\psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2) \;=\; & ((\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i)(\bar{x}_1) \\
& \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge p_i)(\bar{x}_2) \\
& \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)) \\
& \vee \\
& ((\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i)(\bar{x}_1) \\
& \wedge (\epsilon(p'_1) \wedge \ldots \wedge \epsilon(p'_m) \wedge \neg p_i)(\bar{x}_2) \\
& \wedge \epsilon(t_1 \preceq_{\mathrm{R}} t_2)).
\end{aligned}
$$

In this way, we separate each conjunct ψ^j into two conjunctions with an extra data predicate p_i. We repeat this procedure until each conjunct contains all data predicates $p_1, \ldots, p_k$.

Now, we have $\bigvee_j \psi^j(t_1, \bar{x}_1, t_2, \bar{x}_2)$ equivalent to

$$\bigvee_i \psi^i(t_1, \bar{x}_1, t_2, \bar{x}_2),$$

where each conjunction $\psi^i(t_1, \bar{x}_1, t_2, \bar{x}_2)$ has a normal form

$$(\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k))(\bar{x}_1) \wedge \ (\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k))(\bar{x}_2) \wedge \ \epsilon(t_1 \preceq_{\mathrm{R}} t_2).$$

Since any conjunction of data predicates in form of $\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k)$ specifies a minimal partition on the input, the part $\psi^i(t_1, \bar{x}_1, t_2, \bar{x}_2)$ defines a linear order inside a minimal partition as either identical or reverse to the input order.

Let P_i be an ordered minimal partition defined by a conjunction

$$(\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k))(\bar{x}_1) \wedge \ (\epsilon(p_1) \wedge \ldots \wedge \epsilon(p_k))(\bar{x}_2) \wedge \ \epsilon(t_1 \preceq_{\mathrm{R}} t_2).$$

Thus, the ordered minimal partition P_i can be expressed by an ordered algebraic expression E_i in $\boldsymbol{CA}^X_{\preceq}$ as follows:

(1) If P_i is specified by a conjunction of positive data predicates $p_1 \wedge \ldots \wedge p_k$, then

$$E_i = \beta(\sigma_{p_1 \wedge \ldots \wedge p_k}\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle).$$

(2) If P_i is specified by a conjunction of negative data predicates $\neg p_1 \wedge \ldots \wedge \neg p_k$, then

$$E_i = \beta(\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \sigma_{p_1}\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \ldots - \sigma_{p_k}\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle).$$

(3) If P_i is specified by a conjunction of positive and negative data predicates, without lost of generality, we assume that the conjunction has the form of $p_1 \wedge \ldots \wedge p_i \wedge \neg p_{i+1} \wedge \ldots \wedge \neg p_k$, then

$$E_i = \beta(\sigma_{p_1 \wedge \ldots \wedge p_i}\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \sigma_{p_{i+1}}\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \ldots - \sigma_{p_k}\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle),$$

where σ is the order-preserving selection, $-$ the order reduction operation, and β is either the order identity operation μ or the order reverse operation ν in $\boldsymbol{CA}^X_{\preceq}$.

Let $P_1, \ldots, P_{2^k}$ be the sequence of minimal partitions in the output linear order, specified by $\bigvee_j \alpha'^j$. The linear order of $P_1, \ldots, P_{2^k}$ can be expressed by an order concatenation operation on ordered partitions $P_1, \ldots, P_{2^k}$.

All together, the ordered query $\langle \phi, \psi \rangle$ can be expressed by an ordered algebraic expression in $\boldsymbol{CA}^X_\preceq$

$$\bigcup_{i=1,\ldots,2^k} E_i,$$

where $\bigcup$ is an order concatenation operation, and E_i is the algebraic expression of P_i, defined as above.

This ends the proof.

□

Note that in the special case $n = 1$, $\bigvee_i \alpha^i = \bot$, and there is only one minimal partition $P_1 = R$. The ordered query $\langle \phi, \psi \rangle$ can be expressed by an ordered algebraic expression in $\boldsymbol{CA}^X_\preceq$:

$$\beta(\langle \mathrm{R}, \preceq_\mathrm{R} \rangle),$$

where β is an ordered operator μ or ν.

An example of ω queries is shown below. As indicated in the proof above, this ω query in $\boldsymbol{CQ}^X_\preceq$ can be expressed with an ordered algebraic expression in $\boldsymbol{CA}^X_\preceq$.

Example 3.19 The second query in Example 3.5 is an ω query because the data relation in the output is the same as in the input; the output ordered relation is a permutation of the input ordered relation $\langle \mathtt{EMP}, \preceq_\mathtt{EMP} \rangle$. We can formulate this ω query with ordered operations in $\boldsymbol{CA}^X_\preceq$:

$$(\langle \mathtt{EMP}, \preceq_\mathtt{EMP} \rangle - \sigma_{\mathtt{Department}=\text{“}Sales\text{”}} \langle \mathtt{EMP}, \preceq_\mathtt{EMP} \rangle)$$
$$\cup$$
$$\nu(\sigma_{\mathtt{Department}=\text{“}Sales\text{”}} \langle \mathtt{EMP}, \preceq_\mathtt{EMP} \rangle)$$

We have shown that the ω queries in $\boldsymbol{CQ}^X_\preceq$ are expressible by $\boldsymbol{CA}^X_\preceq$. We are now ready for the main result of the investigation in this chapter, which is the completeness theorem for the ordered conjunctive queries $\boldsymbol{CQ}^X_\preceq$. Before proving the completeness of $\boldsymbol{CA}^X_\preceq$, we need to make an assumption. We need this assumption because we need to assume the order in a combination of minimal partitions from all inputs to be a lexicographic order so that we can use ordered operations to express it.

Assumption 3.20 Let $\langle \varphi, \psi \rangle$ be an ordered relational query in $\boldsymbol{FO}_{\preceq}$ on ordered relations $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$, where $n > 1$.

If its data query φ is a Cartesian product of $\mathrm{R}_1, ..., \mathrm{R}_n$, and its order query ψ is constructed by a BNF rule

$$\psi \ = \ R_i(t) \mid t \preceq_{R_i} t' \mid \neg\psi \mid \psi \wedge \psi,$$

then we assume that the output order is a lexicographic order on $\langle R_{i_1}, \preceq_{R_{i_1}} \rangle, \ldots, \langle R_{i_n}, \preceq_{R_{i_n}} \rangle$, where $i_1, ..., i_n$ is an arbitrary permutation of $1, ..., n$.

Theorem 3.21 *Any $\boldsymbol{CQ}^X_{\preceq}$ query is equivalent to an ordered query expressed by a finite sequence of ordered operations from $\boldsymbol{CA}^X_{\preceq}$.*

Proof:
Let $\langle \phi, \psi \rangle$ be an ordered query in $\boldsymbol{CQ}^X_{\preceq}$ on ordered relations $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$.

Case 1: $n = 1$.

In this case, the ordered query $\langle \phi, \psi \rangle$ has only one argument $\langle R_1, \preceq_{R_1} \rangle$, hence it has the form

$$\langle \phi, \psi \rangle(\langle R_1, \preceq_{R_1} \rangle) = \langle \{(t, \bar{x}) \mid \phi(t, \bar{x})\}, \ \{(t_1, t_2) \mid \psi(t_1, t_2)\} \rangle.$$

Here, $\phi(t, \bar{x})$ is a standard conjunctive query, and has the normal form

$$\pi_A(\sigma_{\mathrm{p}}(R)),$$

and

$$\psi = \exists \bar{x}_1 . \phi(t_1, \bar{x}_1) \wedge \exists \bar{x}_2 . \phi(t_2, \bar{x}_2) \wedge \psi'(t_1, \bar{x}_1, t_2, \bar{x}_2).$$

By the definition of $\boldsymbol{CQ}^X_{\preceq}$, ψ' is a first-order formula defined by the BNF rules

$$\psi' \ = \ t_1 \preceq_R t_2 \mid x_i = x_j \mid \neg\psi' \mid \psi' \wedge \psi'.$$

Suppose that $p_1 \wedge \ldots \wedge p_k$ are the only data predicates in ψ'. By Lemma 3.18, there exists an algebraic expression to express the ordered query $\langle \omega, \psi' \rangle$ in $\boldsymbol{CA}^X_{\preceq}$

$$\bigcup_{i=1,\ldots,2^k} E_i,$$

where $\cup$ is an order concatenation operation; E_i is the algebraic expression of partition P_i, and is defined as follows:

(1) If P_i is specified by a conjunction of positive data predicates $p_1 \wedge \ldots \wedge p_k$, then

$$E_i = \beta(\sigma_{p_1 \wedge \ldots \wedge p_k}\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle).$$

(2) If P_i is specified by a conjunction of negative data predicates $\neg p_1 \wedge \ldots \wedge \neg p_k$, then

$$E_i = \beta(\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle - \sigma_{p_1}\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle - \ldots - \sigma_{p_k}\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle).$$

(3) If P_i is specified by a conjunction of positive and negative data predicates, without lost of generality, we assume that the conjunction has the form of $p_1 \wedge \ldots \wedge p_i \wedge \neg p_{i+1} \wedge \ldots \wedge \neg p_k$, then

$$E_i = \beta(\sigma_{p_1 \wedge \ldots \wedge p_i}\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle - \sigma_{p_{i+1}}\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle - \ldots - \sigma_{p_k}\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle),$$

where β is either the order identity operation μ or the order reverse operation ν.

To express the selections and projections on the data relation R in the normal form of ϕ, order-preserving selections and order-preserving projections are applied respectively to the above expression without changing the order. The reason is that order-preserving selections and projections can propagate through order identity, order reverse, and order concatenation operators.

Finally, the ordered $\boldsymbol{CQ}^X_{\preceq}$ query $\langle \phi, \psi \rangle$ is equivalent to the algebraic expression in $\boldsymbol{CA}^X_{\preceq}$:

$$\pi_A(\sigma_{\mathrm{p}}(\bigcup_i E_i)).$$

Case 2: $n > 1$.

Let $\langle \varphi, \psi \rangle$ be a $\boldsymbol{CQ}^X_{\preceq}$ query on ordered relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$, where $n > 1$. The data query φ is a conjunctive query, and hence has the normal form

$$\exists x_1, \ldots, x_m.(\mathrm{R}_1(u_1) \wedge \ \cdots \wedge \mathrm{R}_n(u_n)),$$

where u_i's are tuples containing both variables $t, x_1, \ldots, x_n$ and constants for data attributes. Since the tuple id is unique to each tuple, no duplicate tuples exist in ordered relations. We do not need to consider duplicate elimination in the ordered algebra $\boldsymbol{CA}^X_{\preceq}$.

The order query ψ has the form

$$\begin{aligned}\{(t_1, t_2) \mid \quad & \exists t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots \bar{x}_{2n}. \\ & t_1 = (t_{11}, \ldots, t_{1n}) \wedge t_2 = (t_{21}, \ldots, t_{2n}) \\ & \wedge\ \varphi(t_{11}, \ldots, t_{1n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}) \wedge\ \varphi(t_{21}, \ldots, t_{2n}, \bar{x}_{21}, \ldots, \bar{x}_{2n}) \\ & \wedge\ \psi'(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n})\}.\end{aligned}$$

Intuitively, both t_1 and t_2 must satisfy the order query φ. The order between t_1 and t_2 is decided by the formula ψ'.

Since there is no quantifier on $\boldsymbol{t}$-variables in ψ', no quantifier on $\boldsymbol{x}$-variables occurs in ψ' as well. By the definition of $\boldsymbol{CQ}^X_{\preceq}$, ψ' is a first-order formula defined by the BNF rules

$$\psi' \ = \ t \preceq_{R_i} t' \mid x_i = x_j \mid \neg\psi' \mid \psi' \wedge \psi',$$

and ψ' defines a linear order on the Cartesian product of $\mathrm{R}_1 \times \cdots \times \mathrm{R}_n$.

We can transform ψ' into an equivalent formula in the disjunctive normal form

$$\psi' \ = \ \bigvee_k \psi_0^k(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n})$$

Since there is no existential quantifier on $\boldsymbol{t}$-variables, $t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}$ are the only $\boldsymbol{t}$-variables in ψ', occurring only in forms of $t_{1i} \preceq_{R_i} t_{2i}$ and $t_{2i} \preceq_{R_i} t_{1i}$ for $i = 1, \ldots, n$.

Consequently, we can separate ψ' into two parts

$$\begin{aligned} \psi' = \ & \textstyle\bigvee_i \alpha^i(\bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n}) \\ & \vee \ \textstyle\bigvee_j \psi^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n}), \end{aligned}$$

where α^i's and ψ^j's are conjunctions of atoms or negation of atoms, and predicates on $\boldsymbol{t}$-variables occur only in formulas ψ^j.

First, we consider the first part of ψ', $\bigvee_i \alpha^i(\bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n})$. Since $\bigvee_i \alpha^i$ does not contain $\boldsymbol{t}$-variables, each α^i is a conjunction of data predicates.

Let $p_{i1}, \ldots, p_{ik_i}$ be the k_i data predicates in ψ' over the order relation $\langle \mathrm{R}_i, \preceq_{\mathrm{R}_i} \rangle$, for $i = 1, \ldots, n$. As in Lemma 3.18, we can transform $\bigvee_i \alpha^i$ into $\bigvee_j \alpha'^j$, and each conjunction α'^j is in the form

$$\begin{aligned} & \alpha'^j(\bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n}) \\ & = (\epsilon(p_{11}) \wedge \ldots \wedge \epsilon(p_{1k_1}))(\bar{x}_{11}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{1n}) \\ & \quad \wedge \\ & \quad (\epsilon'(p_{11}) \wedge \ldots \wedge \epsilon'(p_{1k_1}))(\bar{x}_{21}) \wedge \ \ldots \wedge (\epsilon'(p_{n1}) \wedge \ldots \wedge \epsilon'(p_{nk_n}))(\bar{x}_{2n}), \end{aligned}$$

where $\epsilon(p)$ and $\epsilon'(p)$ are either p or $\neg p$.

Let P_{1i} and P_{2i} be two partitions of $\langle R_i, \preceq_{R_i} \rangle$ $(i = 1, \ldots, n)$, whose tuples satisfy data predicates on $\bar{x}_{1i}$ and $\bar{x}_{2i}$ in α'^j, respectively. Since they are specified by a conjunction in form of $\epsilon(p_{i1}) \wedge \ldots \wedge \epsilon(p_{ik_i})$, P_{1i} and P_{2i} are minimal partitions, and therefore cannot be separated into smaller partitions by data predicates.

Let $(P_{11}, \ldots, P_{1n})$ and $(P_{21}, \ldots, P_{2n})$ be the two combinations of minimal partition from $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$, and be specified by α'^j. Let $t_1 = (t_{11}, \ldots, t_{1n})$ and $t_2 = (t_{21}, \ldots, t_{2n})$ be any pair of tuples from $(P_{11}, \ldots, P_{1n})$ and $(P_{21}, \ldots, P_{2n})$ respectively, and $\bar{x}_{11}, \ldots, \bar{x}_{1n}$, $\bar{x}_{21}, \ldots, \bar{x}_{2n}$ be the data attributes of $t_{11}, \ldots, t_{1n}$, $t_{21} \ldots, t_{2n}$ respectively. Since $\bar{x}_{11}, \ldots, \bar{x}_{1n}$, $\bar{x}_{21}, \ldots, \bar{x}_{2n}$ satisfy α'^j, the pair $((t_{11}, \ldots, t_{1n}), (t_{21}, \ldots, t_{2n}))$ will be in the resulting order relation $\preceq_{R'}$, *i.e.*, $t_1 \preceq_{R'} t_2$. Thus, $t_1 \preceq_{R'} t_2$ for any pair of t_1 and t_2 from $(P_{11}, \ldots, P_{1n})$ and $(P_{21}, \ldots, P_{2n})$; hence, $(P_{11}, \ldots, P_{1n}) \preceq_{R'} (P_{21}, \ldots, P_{2n})$.

This follows that any conjunction α'^j defines the order between a pair of combinations of minimal partitions from $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$. Using the same technique in Lemma 3.18, we claim that $\bigvee_j \alpha^j$ defines a linear order among all combinations of minimal partitions $(P_{1i_1}, \ldots, P_{ni_n})$ from $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$. We prove this claim by contradiction as follows.

Suppose that $\bigvee_j \alpha'^j$ does not define a linear order among all combinations of minimal partitions $(P_{i_1 1}, \ldots, P_{i_n n})$ from $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$. There are two cases:

Case 1: Assume that there exist two combinations of partitions, say $(P_{11}, \ldots, P_{1n})$ and $(P_{21}, \ldots, P_{2n})$, such that both $(P_{11}, \ldots, P_{1n}) \preceq_{R'} (P_{21}, \ldots, P_{2n})$ and $(P_{21}, \ldots, P_{2n}) \preceq_{R'} (P_{11}, \ldots, P_{1n})$ hold at the same time. For any $t_1 \in (P_{11}, \ldots, P_{1n})$ and $t_2 \in (P_{21}, \ldots, P_{2n})$, by the assumption, $(t_1, t_2) \in \preceq_{R'}$ and $(t_2, t_1) \in \preceq_{R'}$ hold at the same time. This is a contradiction to the fact that the output order $\preceq_{R'}$ is a linear order. Hence the assumption in Case 1 does not hold.

Case 2: Assume that there exist two combinations of partitions, say $(P_{11}, \ldots, P_{1n})$ and $(P_{21}, \ldots, P_{2n})$, such that neither $(P_{11}, \ldots, P_{1n}) \preceq_{R'} (P_{21}, \ldots, P_{2n})$ nor $(P_{21}, \ldots, P_{2n}) \preceq_{R'} (P_{11}, \ldots, P_{1n})$ holds, that is, their order in output is not specified by $\bigvee_j \alpha'^j$.

For any pair of tuples $t_1 \in (P_{11}, \ldots, P_{1n})$ and $t_2 \in (P_{21}, \ldots, P_{2n})$, by this assumption, their order in the output is not defined by $\bigvee_j \alpha'^j$. Because the output order is a linear order on all combinations of tuples from n ordered relations, the order of t_1 and t_2 must be decided by ψ'. Suppose that ψ' decides $t_1 \preceq_{R'} t_2$. Since $(P_{11}, \ldots, P_{1n})$ and $(P_{21}, \ldots, P_{2n})$ are combinations of minimal partitions, all pairs of tuples from $(P_{11}, \ldots, P_{1n})$ and $(P_{21}, \ldots, P_{2n})$ must have the same order if they have any order at all. Then, for any pair of tuples $t_1 \in (P_{11}, \ldots, P_{1n})$ and $t_2 \in (P_{21}, \ldots, P_{2n})$, we have $t_1 \preceq_{R'} t_2$. It follows $(P_{11}, \ldots, P_{1n}) \preceq_{R'} (P_{21}, \ldots, P_{2n})$, which is a contradiction to the assumption.

We have prove that the first part of ψ' defines a linear order of combinations of partitions $(P_{i_1 1}, \ldots, P_{i_n n})$, which are specified by the set of data predicates in ψ'. Now, we consider the second part of ψ',

$$\bigvee_j \psi^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n}).$$

To impose a linear order on the Cartesian product of data relations $R_1, \ldots, R_n$, the order between each pair of tuples from the same combination of partitions must be decided by the order formula ψ'. This order inside a combination of partitions must be decided by the second part of ψ', $\bigvee_j \psi^j$. For any pair of tuples $(t_{11}, \ldots, t_{1n})$ and $(t_{21}, \ldots, t_{2n})$ from the same combination of minimal partitions $(P_{i_1 1}, \ldots, P_{i_n n})$, we always find a j such that ψ^j decides the order of this pair in the result.

Since there is no existential quantifier on $\boldsymbol{t}$-variables in ψ', $t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}$ are the only $\boldsymbol{t}$-variables in $\bigvee_j \psi^j$, occurring only in forms of $t_{1i} \preceq_{R_i} t_{2i}$ and $t_{2i} \preceq_{R_i} t_{1i}$ for $i = 1, \ldots, n$. By definition, ψ' is a first-order formula defined by the BNF rule

$$\psi' \;=\; t_1 \preceq_R t_2 \mid x_i = x_j \mid \neg\psi' \mid \psi' \wedge \psi',$$

therefore, in $\bigvee_j \psi^j$, each conjunction

$$\psi^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n})$$

must have a form

$$\begin{aligned}&\alpha_{11}(\bar{x}_{11}) \wedge \ldots \wedge \alpha_{1n}(\bar{x}_{1n}) \wedge\ \alpha_{21}(\bar{x}_{21}) \wedge \ldots \wedge \alpha_{2n}(\bar{x}_{2n})\\ &\wedge\psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}),\end{aligned}$$

where α_{1i} and α_{2j} are conjunctions of data predicates on $\bar{x}_{1i}$ and $\bar{x}_{2j}$ respectively, for $i, j = 1, \ldots, n$; the formula $\psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})$ is a logical combination of forms $t_{1k} \preceq_{R_k} t_{2k}$ and $t_{2k} \preceq_{R_k} t_{1k}$ for $k = 1, \ldots, n$.)

Next, we will show that $\alpha_{1i} = \alpha_{2i}$ in each conjunction ψ^j, for $i = 1, \ldots, n$. Assume that α_{1i} and α_{2i} are different conjunctions of data predicates on $\bar{x}_{1i}$ and $\bar{x}_{2i}$ respectively. For any two tuples $(t_{11}, \ldots, t_{1n})$ and $(t_{21}, \ldots, t_{2n})$ that satisfy ψ^j, they satisfy $\alpha_{11}(\bar{x}_{11}) \wedge \ldots \wedge \alpha_{1n}(\bar{x}_{1n})$ and $\alpha_{21}(\bar{x}_{21}) \wedge \ldots \wedge \alpha_{2n}(\bar{x}_{2n})$, which means that $(t_{11}, \ldots, t_{1n}) \preceq_{\mathrm{R}'} (t_{21}, \ldots, t_{2n})$ in the output. They also satisfy $\psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})$, which may lead to $(t_{21}, \ldots, t_{2n}) \preceq_{\mathrm{R}'} (t_{11}, \ldots, t_{1n})$ in the output depending on whether ψ_0^j defines the same order as the conjunctions of data predicates. This is a contradiction. Therefore, the assumption does not hold, and $\alpha_{1i} = \alpha_{2i}$ in each conjunction ψ^j for $i = 1, \ldots, n$.

Similarly to the proof of Lemma 3.18, we can show that each conjunction ψ^j has the normal form

$$\begin{array}{rl} & \psi^j(t_{11},\ldots,t_{1n},t_{21},\ldots,t_{2n},\bar{x}_{11},\ldots,\bar{x}_{1n},\bar{x}_{21},\ldots,\bar{x}_{2n}) \\ = & (\epsilon(p_{11})\wedge\ldots\wedge\epsilon(p_{1k_1}))(\bar{x}_{11})\wedge\ \ldots\wedge(\epsilon(p_{n1})\wedge\ldots\wedge\epsilon(p_{nk_n}))(\bar{x}_{1n}) \\ & \wedge\ (\epsilon(p_{11})\wedge\ldots\wedge\epsilon(p_{1k_1}))(\bar{x}_{21})\wedge\ \ldots\wedge(\epsilon(p_{n1})\wedge\ldots\wedge\epsilon(p_{nk_n}))(\bar{x}_{2n}) \\ & \wedge\ \psi_0^j(t_{11},\ldots,t_{1n},t_{21},\ldots,t_{2n}), \end{array}$$

where $\epsilon(p)$ is either p or $\neg p$, k_i is the number of data predicates over $\langle \mathrm{R}_i, \preceq_{\mathrm{R}_i}\rangle$, and $\psi_0^j(t_{11},\ldots,t_{1n},t_{21},\ldots,t_{2n})$ decides the real order in the combination of minimal partitions and is defined by the BNF rule

$$\psi_0^j = t_{1i} \preceq_{\mathrm{R}} t_{2i} \mid \neg\psi_0^j \mid \psi_0^j \wedge\ \psi_0^j.$$

Let $p_{i1},\ldots,p_{ik_i}$ be the k_i data predicates on $\langle \mathrm{R}_i, \preceq_{\mathrm{R}_i}\rangle$, for $i = 1,\ldots,n$. Without lost of generality, we suppose that $\alpha_{11} = \epsilon(p'_{11})\wedge\ldots\wedge\epsilon(p'_{1m})$ and $\{p'_{11},\ldots,p'_{1m}\} \subset \{p_{11},\ldots,p_{1k_1}\}$. Assume that $p \in \{p_{11},\ldots,p_{1k_1}\}$, however $p \notin \{p'_{11},\ldots,p'_{1m}\}$. Thus,

$$\begin{array}{rl} & \psi^j(t_{11},\ldots,t_{1n},t_{21},\ldots,t_{2n},\bar{x}_{11},\ldots,\bar{x}_{1n},\bar{x}_{21},\ldots,\bar{x}_{2n}) \\ = & (\epsilon(p'_{11})\wedge\ldots\wedge\epsilon(p'_{1m}))(\bar{x}_{11})\wedge\ \ldots\wedge(\epsilon(p_{n1})\wedge\ldots\wedge\epsilon(p_{nk_n}))(\bar{x}_{1n}) \\ & \wedge \\ & (\epsilon(p'_{11})\wedge\ldots\wedge\epsilon(p'_{1m}))(\bar{x}_{21})\wedge\ \ldots\wedge(\epsilon(p_{n1})\wedge\ldots\wedge\epsilon(p_{nk_n}))(\bar{x}_{2n}) \\ & \wedge \\ & \psi_0^j(t_{11},\ldots,t_{1n},t_{21},\ldots,t_{2n}) \\ = & (\epsilon(p'_{11})\wedge\ldots\wedge\epsilon(p'_{1m})\wedge(p\vee\neg p))(\bar{x}_{11})\wedge\ \ldots\wedge(\epsilon(p_{n1})\wedge\ldots\wedge\epsilon(p_{nk_n}))(\bar{x}_{1n}) \\ & \wedge \\ & (\epsilon(p'_{11})\wedge\ldots\wedge\epsilon(p'_{1m})\wedge(p\vee\neg p))(\bar{x}_{21})\wedge\ \ldots\wedge(\epsilon(p_{n1})\wedge\ldots\wedge\epsilon(p_{nk_n}))(\bar{x}_{2n}) \\ & \wedge \\ & \psi_0^j(t_{11},\ldots,t_{1n},t_{21},\ldots,t_{2n}) \\ = & ((\epsilon(p'_{11})\wedge\ldots\wedge\epsilon(p'_{1m})\wedge p)\vee(\epsilon(p'_{11})\wedge\ldots\wedge\epsilon(p'_{1m})\wedge\neg p))(\bar{x}_{11}) \\ & \wedge\ \ldots\wedge(\epsilon(p_{n1})\wedge\ldots\wedge\epsilon(p_{nk_n}))(\bar{x}_{1n}) \\ & \wedge \\ & ((\epsilon(p'_{11})\wedge\ldots\wedge\epsilon(p'_{1m})\wedge p)\vee(\epsilon(p'_{11})\wedge\ldots\wedge\epsilon(p'_{1m})\wedge\neg p))(\bar{x}_{21}) \\ & \wedge\ \ldots\wedge(\epsilon(p_{n1})\wedge\ldots\wedge\epsilon(p_{nk_n}))(\bar{x}_{2n}) \\ & \wedge \\ & \psi_0^j(t_{11},\ldots,t_{1n},t_{21},\ldots,t_{2n}) \end{array}$$

$$
\begin{array}{ll}
= & ((\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge p)(\bar{x}_{11}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{1n}) \\
& \wedge\ (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge p)(\bar{x}_{21}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{2n}) \\
& \wedge\ \psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})) \\
& \vee \\
& (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge p)(\bar{x}_{11}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{1n}) \\
& \wedge\ (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge \neg p)(\bar{x}_{21}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{2n}) \\
& \wedge\ \psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})) \\
& \vee \\
& (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge \neg p)(\bar{x}_{11}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{1n}) \\
& \wedge\ (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge p)(\bar{x}_{21}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{2n}) \\
& \wedge\ \psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})) \\
& \vee \\
& (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge \neg p)(\bar{x}_{11}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{1n}) \\
& \wedge\ (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge \neg p)(\bar{x}_{21}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{2n}) \\
& \wedge\ \psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})) \\
= & ((\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge p)(\bar{x}_{11}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{1n}) \\
& \wedge\ (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge p)(\bar{x}_{21}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{2n}) \\
& \wedge\ \psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})) \\
& \vee \\
& (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge \neg p)(\bar{x}_{11}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{1n}) \\
& \wedge\ (\epsilon(p'_{11}) \wedge \ldots \wedge \epsilon(p'_{1m}) \wedge \neg p)(\bar{x}_{21}) \wedge \ \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{2n}) \\
& \wedge\ \psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}))
\end{array}
$$

In this way, we separate each conjunction ψ^j into two conjunctions by including an extra data predicates p in α_{11}. We repeat this procedure until we have each conjunction containing all data predicates $p_{i1}, \ldots, p_{ik_i}$, for $i = 1, \ldots, n$. This finishes the proof of the normal form of ψ^j.

Intuitively, each conjunction ψ^j defines a linear order in a combination of minimal partitions which is decided by the data conjunction $(\epsilon(p_{11}) \wedge \ldots \wedge \epsilon(p_{1k_1}))(\bar{x}_{11}) \wedge \ldots \wedge (\epsilon(p_{n1}) \wedge \ldots \wedge \epsilon(p_{nk_n}))(\bar{x}_{1n})$; the real ordering in the combination of minimal partitions is decided by $\psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}))$.

We then consider ψ_0^j, which defines the real ordering inside each combination of minimal partitions in the output. The formula ψ_0^j is defined by the BNF rule

$$\psi_0^j = t_1 \preceq_{\mathrm{R}} t_2 \mid \neg\psi_0^j \mid \psi_0^j \wedge \ \psi_0^j.$$

We do not have data predicates and quantifiers of $\boldsymbol{t}$-variables in ψ_0^j. Thus, the formula ψ_0^j is a boolean combination of order predicates $t_{11} \preceq_{\mathrm{R}} t_{21}, \ldots, t_{1n} \preceq_{\mathrm{R}} t_{2n}$.

For each input ordered relation $\langle \mathrm{R}_i, \preceq_{\mathrm{R}_i} \rangle$, we have only $t_{1i} \preceq_{\mathrm{R}} t_{2i}$ or $t_{2i} \preceq_{\mathrm{R}} t_{1i}$ to express its order in the minimal partitions, which is either identical or reverse to the input order. And ψ_0^j is a linear order in a combination of minimal partitions. Therefore, lexicographical orders are the only possible linear orders inside a combination of minimal partitions that is defined by ψ_0^j. We state this more generally. Let $P_{i_j k_j}$ $(j = 1, ..., n)$ be the i_j-th partition in the input ordered relation $\langle R_{k_j}, \preceq_{R_{k_j}} \rangle$. Since the order inside each combination of minimal partitions $(P_{i_1 k_1}, \ldots, P_{i_n k_n})$ is a lexicographical order, which is decided by ψ_0^j. Any lexicographical order that is decided by each ψ_0^j is equivalent to a normal form

$$\begin{aligned} & \psi_0^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}) \\ = \; & \epsilon(t_{1k_1} \preceq_{\mathrm{R}} t_{2k_1}) \vee \; (t_{1k_1} = t_{2k_1} \wedge \; \epsilon(t_{1k_2} \preceq_{\mathrm{R}} t_{2k_2})) \vee \; \ldots \\ & \vee \; (t_{1k_1} = t_{2k_1} \wedge \; \ldots \; \wedge \; t_{1k_{n-1}} = t_{2k_{n-1}} \wedge \; \epsilon(t_{1k_n} \preceq_{\mathrm{R}} t_{2k_n})), \end{aligned}$$

where $\epsilon(t_{1k_n} \preceq_{\mathrm{R}} t_{2k_n})$ is either $t_{1k_n} \preceq_{\mathrm{R}} t_{2k_n}$ or $t_{2k_n} \preceq_{\mathrm{R}} t_{1k_n}$. $(k_1, \ldots, k_n)$ is a permutation of $(1, \ldots, n)$, which decides the major to minor order that $\langle \mathrm{R}_{k_1}, \preceq_{\mathrm{R}_{k_1}} \rangle, \ldots, \langle \mathrm{R}_{k_n}, \preceq_{\mathrm{R}_{k_n}} \rangle$ serve in the output order.

Intuitively, ψ_0^j defines a lexicographical order on $(P_{i_1 k_1}, \ldots, P_{i_n k_n})$, and each minimal partition has an order identical or reverse to the original input order. Therefore, each combination of minimal partitions $(P_{i_1 k_1}, \ldots, P_{i_n k_n})$ can be expressed by an algebraic expression in the ordered conjunctive algebra $\boldsymbol{CA}^X_{\preceq}$ as follows,

$$\beta(E_{i_1 k_1}) \ltimes \; \ldots \ltimes \; \beta(E_{i_n k_n}),$$

where β is either μ or ν, $E_{i_1 k_1}$ is the ordered algebraic expression for $P_{i_1 k_1}$, defined as follows:

(1) If $P_{i_1 k_1}$ is specified by a conjunction of positive data predicates $p_1 \wedge \ldots \wedge p_j$, then

$$E_{i_1 k_1} = \sigma_{p_1 \wedge \ldots \wedge p_j} \mathrm{R}_{k_1}.$$

(2) If $P_{i_1 k_1}$ is specified by a conjunction of negative data predicates $\neg p_1 \wedge \ldots \wedge \neg p_j$, then

$$E_{i_1 k_1} = \mathrm{R}_{k_1} \; - \sigma_{p_{1j}} \mathrm{R}_j \; - \ldots - \sigma p_{k_j j} \mathrm{R}_{k_1}.$$

(3) If $P_{i_1 k_1}$ is specified by a conjunction of positive and negative data predicates, w.l.o.g., we assume that the conjunction has the form of $p_1 \wedge \ldots \wedge p_m \wedge \neg p_{m+1} \wedge \ldots \wedge \neg p_j$, then

$$E_{i_1 k_1} = \sigma_{p_1 \wedge \ldots \wedge p_m} \mathrm{R}_{k_1} - \sigma_{p_{m+1}} \mathrm{R}_{k_1} - \ldots - \sigma_{p_j} \mathrm{R}_{k_1}.$$

Please note that the sequence $(k_1, \ldots, k_n)$ is a permutation of $(1, \ldots, n)$, which indicates the order in which the left-nested-loop product operations are applied to the partitions of $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$.

All together, the ordered query $\langle \phi', \psi' \rangle$ (where the expression of ϕ' is $\mathrm{R}_1 \times \cdots \times \mathrm{R}_n$) can be expressed by

$$\bigcup (\beta(E_{i_1 k_1}) \ltimes \ \ldots \ \ltimes \ \beta(E_{i_n k_n})),$$

where β is either μ or ν, and $\bigcup$ is an order concatenation operator. And $(k_1, \ldots, k_n)$ is a permutation of $(1, \ldots, n)$, different in different combinations of partition $(P_{i_1 k_1}, \ldots, P_{i_n k_n})$.

Back to the data query ϕ, if there are selections and projections in the normal form of ϕ, we add order-preserving selections and order-preserving projections on the above algebraic expression correspondingly as order-preserving selections and projections can pass through left nested-loop product while keeping equivalence.

Therefore, any $\boldsymbol{CQ}^X_\preceq$ query $\langle \phi, \psi \rangle$ on ordered relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$ is equivalent to an ordered query expressed in $\boldsymbol{CA}^X_\preceq$:

$$\pi_A(\sigma_p(\bigcup (\beta(E_{i_1 k_1}) \ltimes \ \ldots \ \ltimes \ \beta(E_{i_n k_n})))).$$

□

Intuitively, for an ω query in $\boldsymbol{CQ}^X_\preceq$, its output order is a linear order of minimal partitions specified by the order query. Minimal partitions can be expressed by order-preserving selections and order reduction operations; a linear order of minimal partitions can be expressed by order concatenations; the order inside a minimal partition can be expressed by the order identity or the order reverse because the only possible orders of a minimal partition is either identical to or reverse of the input order.

For an ordered conjunctive query in $\boldsymbol{CQ}^X_\preceq$ with only one input, the output order can be obtained by attaching order-preserving selections and order-preserving projections to the algebraic expression of the corresponding ω query. For an ordered conjunctive query in $\boldsymbol{CQ}^X_\preceq$ with more than one input, the output order is a linear order of combinations of minimal partitions from each input ordered relation. The combination of minimal partitions can be expressed with left nested-loop products; the rest of the order query can be expressed with ordered operations with similar techniques for the ω query.

The following examples demonstrate how to express $\boldsymbol{CQ}^X_{\preceq}$ queries with order algebraic expressions $\boldsymbol{CA}^X_{\preceq}$.

Example 3.22 Consider the queries in Example 3.5. These $\boldsymbol{CQ}^X_{\preceq}$ queries can be expressed with ordered operators in $\boldsymbol{CA}^X_{\preceq}$.

(1) The first query is an ordered query in $\boldsymbol{CQ}^X_{\preceq}$, changing on both the data relation and order relation in the output. We can formulate this ordered query with ordered operations in $\boldsymbol{CA}^X_{\preceq}$:

$$\pi_{\texttt{Name,Department}}(\sigma_{\texttt{Salary}=65000}\langle \texttt{EMP}, \preceq_{\texttt{EMP}}\rangle).$$

(2) For the second query, we first formulate "the employees in the department Sales in the reverse order" as:

$$\sigma_{\texttt{Department}=\text{“}Sales\text{”}}(\nu(\langle \texttt{EMP}, \preceq_{\texttt{EMP}}\rangle)).$$

Next, "the rest of employees except those in the department Sales" is expressed as:

$$E_2 = \ \langle \texttt{EMP}, \preceq_{\texttt{EMP}}\rangle - \sigma_{\texttt{Department}=\text{“}Sales\text{”}}\langle \texttt{EMP}, \preceq_{\texttt{EMP}}\rangle.$$

Finally, the second query is represented with an ordered algebraic expression in $\boldsymbol{CA}^X_{\preceq}$ as follows:

$$E_2 \cup E_1.$$

This chapter investigated the first class of ordered conjunctive queries $\boldsymbol{CQ}^X_{\preceq}$ and proved that the ordered algebra $\boldsymbol{CA}^X_{\preceq}$ is complete in the sense that it can express the set of ordered conjunctive queries in $\boldsymbol{CQ}^X_{\preceq}$ under certain constraints. The following chapters will explore two other classes of ordered conjunctive queries, $\boldsymbol{CQ}^T_{\preceq}$ and $\boldsymbol{CQ}_{\preceq}$.

Chapter 4

Ordered Conjunctive Queries with t-Decided Order $CQ^T_{\preceq}$

The previous chapter has investigated ordered conjunctive queries with data-decided order, $\boldsymbol{CQ}^X_{\preceq}$. This chapter focuses on the second class of ordered conjunctive queries, ordered conjunctive queries with $\boldsymbol{t}$-decided order, $\boldsymbol{CQ}^T_{\preceq}$. In particular, it explores whether a complete ordered algebra exists for the ordered conjunctive queries $\boldsymbol{CQ}^T_{\preceq}$.

Section 4.1 formally defines ordered conjunctive queries $\boldsymbol{CQ}^T_{\preceq}$ in the first-order logic $\boldsymbol{FO}_{\preceq}$ and describes the representative examples in $\boldsymbol{CQ}^T_{\preceq}$. Section 4.2 proposes an ordered algebra $\boldsymbol{CA}^T_{\preceq}$ for $\boldsymbol{CQ}^T_{\preceq}$, which contains a set of primitive ordered operators and derived operators. Section 4.3 provides transformation rules for ordered operations in $\boldsymbol{CA}^T_{\preceq}$. Section 4.4 presents the completeness theorem for $\boldsymbol{CQ}^T_{\preceq}$, which shows that $\boldsymbol{CA}^T_{\preceq}$ is a complete algebra for $\boldsymbol{CQ}^T_{\preceq}$.

4.1 Ordered Conjunctive Queries $CQ^T_{\preceq}$

Analogously to $\boldsymbol{CQ}^X_{\preceq}$, $\boldsymbol{CQ}^T_{\preceq}$ is another subset of the general ordered conjunctive query $\boldsymbol{CQ}_{\preceq}$. It forbids data predicates in order specification of an ordered conjunctive query; the output order is decided by $\boldsymbol{t}$-variables in the order specification. The formal definition of the ordered conjunctive query $\boldsymbol{CQ}^T_{\preceq}$ follows.

Definition 4.1 (Ordered Conjunctive Query $\boldsymbol{CQ}^T_{\preceq}$)
A $\boldsymbol{CQ}^T_{\preceq}$ *ordered query* $\langle \varphi, \psi \rangle$ *on ordered relations* $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$ *is constructed as follows:*

(*a*) *The data query φ is a conjunctive query and defined by the following BNF rule:*

$$\begin{array}{rcl} \varphi & ::= & R(t_i, x_1, \ldots, x_n) \\ & | & t = (t_1, t_2) \\ & | & t_i \preceq_R t_j \\ & | & x_i = x_j \\ & | & \varphi \wedge \varphi \\ & | & \exists x_i.\varphi \\ & | & \exists t_i.\varphi. \end{array}$$

(*b*) *The order query ψ is a first-order formula in the form of*

$$\begin{array}{rcl} \psi & ::= & R(t_i, x_1, \ldots, x_n) \\ & | & t = (t_1, t_2) \\ & | & x_i = x_j \\ & | & \psi \wedge \psi \\ & | & \exists x_i.\psi \\ & | & \exists t_i.\psi \\ & | & \psi \wedge \psi', \end{array}$$

where the order specification ψ' is defined by the BNF rule:

$$\begin{array}{rcl} \psi' & ::= & R(t_i, x_1, \ldots, x_n) \\ & | & t_i \preceq_R t_j \\ & | & \psi' \wedge \psi' \\ & | & \neg\psi' \\ & | & \exists x_i.\psi' \\ & | & \exists t_i.\psi', \end{array}$$

where ψ has exactly two t-variables as the only free variables, and the order specification ψ' has all $\boldsymbol{x}$-variables bounded.

In this definition, there is a two-level construction in the definition of ψ. The first level of ψ has the same construction rules as φ, ensuring that ψ defines orders for any pair of tuples t_1 and t_2 that satisfy the data query φ. The second level of ψ is the sub-formula ψ', which defines real ordering among the resulting tuples. Since there are no quantifiers on $\boldsymbol{t}$-variables in ψ', all $\boldsymbol{t}$-variables in ψ' appear only in atomic forms $t_{1i} \preceq_R t_{2i}$ or $t_{2i} \preceq_R t_{1i}$, where t_{1i} and t_{2i} are originally from the same input ordered relation $\langle R_i, \preceq_{R_i} \rangle$, but are components of different output tuples. In $\boldsymbol{CQ}^T_{\preceq}$, there is no data predicate $x_i = x_j$ in the BNF rule of ψ', which means that

the resulting order is decided by ordering among tuple identifiers (corresponding to $\boldsymbol{t}$-variables). Thus, the ordered conjunctive query defined in Definition 4.1 is named $\boldsymbol{CQ}^T_\preceq$. The superscript T indicates that the order is decided by predicates on $\boldsymbol{t}$-variables.

Similar to $\boldsymbol{CQ}^X_\preceq$, a $\boldsymbol{CQ}^T_\preceq$ query on ordered relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$ has the following form:

$$
\begin{array}{ll}
\langle\{(t, \bar{x}) \mid & \varphi(t, \bar{x})\}, \\
\{(t_1, t_2) \mid & \exists t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots \bar{x}_{2n}. \\
& t_1 = (t_{11}, \ldots, t_{1n}) \wedge\ t_2 = (t_{21}, \ldots, t_{2n}) \\
& \wedge\ \varphi(t_{11}, \ldots, t_{1n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}) \wedge\ \varphi(t_{21}, \ldots, t_{2n}, \bar{x}_{21}, \ldots, \bar{x}_{2n}) \\
& \wedge\ \psi'(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n})\}\ \rangle,
\end{array}
$$

where ψ' is constructed by the BNF rule

$$\psi' ::= R(t_i, x_1, \ldots, x_n) \mid t_i \preceq_R t_j \mid \psi' \wedge \psi' \mid \neg\psi' \mid \exists x_i.\psi' \mid \exists t_i.\psi',$$

and t_1 and t_2 are the only two free variables in the order specification ψ.

The $\boldsymbol{CQ}^T_\preceq$ query is a class of ordered conjunctive queries whose output ordering is solely decided by the order of tuple identifiers, as illustrated by the example below.

Example 4.2 Consider the ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}}\rangle$ in Example 2.5. The following are two examples of $\boldsymbol{CQ}^T_\preceq$ queries:

(1) Output a list of employees in an order in which the first employee moves to the end of the list, while the rest of employees remain in the original order. This query can be formulated in $\boldsymbol{CQ}^T_\preceq$:

$$
\begin{array}{ll}
\langle\{(t, x_1, & x_2, x_3, x_4) \mid \texttt{EMP}\ (t, x_1, x_2, x_3, x_4)\}, \\
\{(t_1, t_2) \mid & \exists x_{11}, x_{12}, x_{13}, x_{14}.\ \texttt{EMP}\ (t_1, x_{11}, x_{12}, x_{13}, x_{14}) \\
& \wedge \exists x_{21}, x_{22}, x_{23}, x_{24}.\ \texttt{EMP}\ (t_2, x_{21}, x_{22}, x_{23}, x_{24}) \\
& \wedge\ (((\exists t_3, x_{31}, x_{32}, x_{33}, x_{34}.\ \texttt{EMP}\ (t_3, x_{31}, x_{32}, x_{33}, x_{34}) \wedge t_3 \preceq_{\mathrm{R}} t_1) \\
& \wedge\ (\neg\exists t_3, x_{31}, x_{32}, x_{33}, x_{34}.\ \texttt{EMP}\ (t_3, x_{31}, x_{32}, x_{33}, x_{34}) \wedge t_3 \preceq_{\mathrm{R}} t_2)) \\
& \vee\ ((\neg\exists t_3, x_{31}, x_{32}, x_{33}, x_{34}.\ \texttt{EMP}\ (t_3, x_{31}, x_{32}, x_{33}, x_{34}) \wedge t_3 \preceq_{\mathrm{R}} t_1) \\
& \wedge\ (\neg\exists t_3, x_{31}, x_{32}, x_{33}, x_{34}.\ \texttt{EMP}\ (t_3, x_{31}, x_{32}, x_{33}, x_{34}) \wedge t_3 \preceq_{\mathrm{R}} t_2) \\
& \wedge\ t_1 \preceq_{\mathrm{R}} t_2))\}\rangle.
\end{array}
$$

(2) Return a list of employees in the department Sales in an order that is the reverse of the input order. This query can be formulated in $\boldsymbol{CQ}^T_\preceq$ as follows:

$$\langle\{(t, x_1, x_2, x_4)| \quad \exists x_3.\ \texttt{EMP}\ (t, x_1, x_2, x_3, x_4) \wedge x_3 = \text{“}sales\text{”}\},$$
$$\{(t_1, t_2)| \quad \exists x_{11}, x_{12}, x_{13}, x_{14}.\ \texttt{EMP}\ (t_1, x_{11}, x_{12}, x_{13}, x_{14}) \wedge x_{13} = \text{“}Sales\text{”}$$
$$\wedge \exists x_{21}, x_{22}, x_{23}, x_{24}.\ \texttt{EMP}\ (t_2, x_{21}, x_{22}, x_{23}, x_{24}) \wedge x_{23} = \text{“}Sales\text{”}$$
$$\wedge \neg\ t_1 \preceq_{\texttt{EMP}} t_2\}\rangle.$$

4.2 An Ordered Conjunctive Algebra $\boldsymbol{CA}^T_\preceq$

The algebraic paradigm for $\boldsymbol{CA}^T_\preceq$ ordered conjunctive queries is based on a family of unary and binary operators on ordered relations, as for $\boldsymbol{CA}^X_\preceq$ ordered conjunctive queries. This section proposes an ordered conjunctive algebra $\boldsymbol{CA}^T_\preceq$ for ordered conjunctive queries $\boldsymbol{CQ}^T_\preceq$ and defines a set of derived ordered operators which can be composed by ordered operators in $\boldsymbol{CA}^T_\preceq$.

4.2.1 Overview of Ordered Algebra $\boldsymbol{CA}^T_\preceq$

In the ordered conjunctive algebra $\boldsymbol{CA}^T_\preceq$, a set of ordered operators are offered to express $\boldsymbol{CQ}^T_\preceq$ expressible ordered conjunctive queries. Several ordered operators are preserved from $\boldsymbol{CA}^X_\preceq$, and others are exclusively developed for $\boldsymbol{CA}^T_\preceq$. Both data and order specifications of an ordered operator are defined in the two-sorted first-order calculus $\boldsymbol{FO}_\preceq$. Essentially, each ordered operator in this algebra is a valid $\boldsymbol{FO}_\preceq$ ordered relational query. Similar to $\boldsymbol{CA}^X_\preceq$, some ordered operators in $\boldsymbol{CA}^T_\preceq$ are not definable in $\boldsymbol{CQ}^T_\preceq$ because non-conjunctive operators are needed to handle the negations in the order query of a $\boldsymbol{CQ}^T_\preceq$ ordered conjunctive query.

The ordered algebra $\boldsymbol{CA}^T_\preceq$ consists of a set of ordered operators:

(i) Order-preserving selection operator $\sigma : \mathbf{R}_\preceq \times \mathbf{P} \rightarrow \mathbf{R}_\preceq$;

(ii) Order-preserving projection operator $\pi : \mathbf{R}_\preceq \times \mathrm{A} \rightarrow \mathbf{R}_\preceq$;

(iii) Left nested-loop product $\ltimes : \mathbf{R}_\preceq \times \mathbf{R}_\preceq \rightarrow \mathbf{R}_\preceq$;

(iv) Ordered concatenation operator $\cup : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$;

(v) Order reduction operator $- : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$;

(*vi*) Order intersection operator $\cap : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$;

(*vii*) Order identity operator $\mu : \mathbf{R} \rightarrow \mathbf{R}$;

(*viii*) Order reverse operator $\nu : \mathbf{R} \rightarrow \mathbf{R}$;

(*ix*) Top operator $\tau : \mathbf{R} \rightarrow \mathbf{R}$.

The ordered operators in $\boldsymbol{CA}^T_{\preceq}$ are defined in the same way as in $\boldsymbol{CA}^X_{\preceq}$, except the *order intersection* and *top* operators. The following subsections formally define these two operators.

4.2.2 Order Intersection

The order intersection operator $\cap : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$ is exclusive to $\boldsymbol{CA}^T_{\preceq}$. It is not the ordered counterpart of the set intersection operator in relational algebras, and is used only to extract the common segment of two input ordered relations. The schemas of both the argument relations and the resulting relation are the same. The formal definition of the order intersection operator is given as follows:

$$\begin{aligned} \mathbf{R}_1 \cap \mathbf{R}_2 = \langle \{(t, \bar{x}) \mid & \ \mathrm{R}_1(t, \bar{x}) \wedge \mathrm{R}_2(t, \bar{x})\}, \\ \{(t_1, t_2) \mid & \ (\exists \bar{x}_1 . \mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1)) \\ & \wedge (\exists \bar{x}_2 . \mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2)) \\ & \wedge t_1 \preceq_{R_1} t_2\} \rangle. \end{aligned}$$

The order intersection operator takes as its input two ordered relations, which originally come from the same ordered relation, and returns all tuples of the first argument that have duplicate tuples in the second argument. The output order is identical to the original order in the first argument. Here, "duplicate" tuples are those tuples that are identical on both tuple identifiers and data attributes. Therefore, tuple identifiers of the result are the same as those in the first and second arguments.

Analogous to the order concatenation operator and order reduction operator, the order intersection operator is used only to manipulate the order within a single ordered relation in $\boldsymbol{CA}^T_{\preceq}$. Thus, this is not a counterpart of conventional set intersection operation as in relational algebras.

Lemma 4.3 *An order intersection operation is a valid ordered relational query.*

Proof: Let $\mathbf{R}_1 \cap \mathbf{R}_2 = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$, where $\mathbf{R}_1 = \langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle$ and $\mathbf{R}_2 = \langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle$.

Clearly, the data relation R' and the order relation $\preceq_{\mathrm{R}'}$ are inclusively dependant on each other by definition of order intersection. A proof is needed that $\preceq_{\mathrm{R}'}$ is a linear order of R'.

(1) Let $t_1, t_2, t_3 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_3) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow \quad & ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge t_1 \preceq_{R_1} t_2) \\
& \wedge ((\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge (\exists \bar{x}_3.\mathrm{R}_1(t_3, \bar{x}_3) \wedge \mathrm{R}_2(t_3, \bar{x}_3)) \\
& \wedge t_2 \preceq_{R_1} t_3) \\
\Longrightarrow \quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_3.\mathrm{R}_1(t_3, \bar{x}_3) \wedge \ \mathrm{R}_2(t_3, \bar{x}_3)) \wedge t_1 \preceq_{R_1} t_3 \\
\Longrightarrow \quad & (t_1, t_3) \in \preceq_{\mathrm{R}'} .
\end{aligned}
$$

(2) Let $t_1, t_2 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_1) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow \quad & ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge t_1 \preceq_{R_1} t_2) \\
& \wedge ((\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \ \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \ \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge t_2 \preceq_{R_1} t_1) \\
\Longrightarrow \quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge (t_1 \preceq_{R_1} t_2 \wedge t_2 \preceq_{R_1} t_1) \\
\Longrightarrow \quad & t_1 = t_2.
\end{aligned}
$$

(3) Let t_1 and t_2 be two arbitrary tuples in R'. By definition of the order intersection operator,

$$
\begin{aligned}
& t_1, t_2 \in \mathrm{R}'[\mathrm{T}] \\
\Longrightarrow \quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2)) \\
\Longrightarrow \quad & (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2)) \\
& \wedge (t_1 \preceq_{\mathrm{R}_1} t_2 \vee t_2 \preceq_{\mathrm{R}_1} t_1)
\end{aligned}
$$

$$
\begin{aligned}
\Longrightarrow \quad & ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2)) \wedge t_1 \preceq_{\mathrm{R}_1} t_2) \\
& \vee ((\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \ \mathrm{R}_2(t_1, \bar{x}_1)) \\
& \wedge (\exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2)) \wedge t_2 \preceq_{\mathrm{R}_1} t_1) \\
\Longrightarrow \quad & t_1 \preceq_{\mathrm{R}'} t_2 \vee t_2 \preceq_{\mathrm{R}'} t_1
\end{aligned}
$$

It is thus proved that the order intersection is a valid ordered relational query. □

By definition, the data query of an order intersection operator is a conjunctive query. Therefore, it is also a valid ordered conjunctive query in $\boldsymbol{CQ}^T_{\preceq}$.

4.2.3 Top Operation

The top operator $\tau : \mathbf{R} \to \mathbf{R}$ is an operator exclusive to $\boldsymbol{CA}^T_{\preceq}$. It takes an ordered relation as its input and returns the first tuple in the input relation. The input and resulting ordered relations share the same schema. The tuple identifiers in the resulting ordered relation are identical to those in the input relation.

The top operator is not definable in $\boldsymbol{CQ}^T_{\preceq}$ because it has negations in its data definition. The formal definition of the top operator is given as follows:

$$
\begin{aligned}
\tau(\langle R, \preceq_{\mathrm{R}} \rangle) = \ & \langle \{(t, \bar{x}) | \quad \mathrm{R}(t, \bar{x}) \wedge \neg \exists t_1, \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge t_1 \preceq_{\mathrm{R}} t\}, \\
& \{(t_1, t_2) | \quad \exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \qquad \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& \qquad \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& \qquad \wedge t_1 \preceq_R t_2\}\rangle.
\end{aligned}
$$

Lemma 4.4 *The top operation is a valid ordered relational query.*

Proof:

Let $\tau(\mathbf{R}) = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$, where $\mathbf{R} = \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. By definition of the top operation, R' and $\preceq_{\mathrm{R}'}$ are inclusively dependant on each other. There follows proof that $\preceq_{\mathrm{R}'}$ is a linear order of R'.

(1) Let $t_1, t_2, t_3 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_3) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow \quad & (\exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge \preceq_{\mathrm{R}} t_1 \\
& \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& \wedge t_1 \preceq_R t_2) \\
& \wedge (\exists \bar{x}_2, \bar{x}_3.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{R}(t_3, \bar{x}_3) \\
& \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& \wedge \neg \exists t_{31}, \bar{x}_{31}.\mathrm{R}(t_{31}, \bar{x}_{31}) \wedge t_{31} \preceq_{\mathrm{R}} t_3 \\
& \wedge t_2 \preceq_R t_3) \\
\Longrightarrow \quad & (\exists \bar{x}_1, \bar{x}_3.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_3, \bar{x}_3) \\
& \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& \wedge \neg \exists t_{31}, \bar{x}_{31}.\mathrm{R}(t_{31}, \bar{x}_{31}) \wedge t_{31} \preceq_{\mathrm{R}} t_3 \\
& \wedge t_1 \preceq_R t_3) \\
\Longrightarrow \quad & (t_1, t_3) \in \preceq_{\mathrm{R}'} .
\end{aligned}
$$

(2) Let $t_1, t_2 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge (t_2, t_1) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow \quad & (\exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& \wedge t_1 \preceq_{\mathrm{R}} t_2) \\
& \wedge (\exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& \wedge t_2 \preceq_R t_1) \\
\Longrightarrow \quad & \exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& \wedge t_1 \preceq_R t_2 \wedge t_2 \preceq_R t_1 \\
\Longrightarrow \quad & t_1 = t_2.
\end{aligned}
$$

(3) Let t_1 and t_2 be two arbitrary tuples in R'. By definition of the top operation,

$$
\begin{aligned}
& t_1, t_2 \in \mathrm{R}'[\mathrm{T}] \\
\Longrightarrow \quad & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \\
& \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2
\end{aligned}
$$

$$
\begin{aligned}
\Longrightarrow \quad & \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \\
& \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& \wedge (t_1 \preceq_{\mathrm{R}} t_2 \vee t_2 \preceq_{\mathrm{R}} t_1) \\
\Longrightarrow \quad & (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \\
& \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& \wedge t_1 \preceq_{\mathrm{R}} t_2) \\
& \vee (\exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \\
& \wedge \neg \exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& \wedge \neg \exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& \wedge t_2 \preceq_{\mathrm{R}} t_1) \\
\Longrightarrow \quad & t_1 \preceq_{\mathrm{R}'} t_2 \vee t_2 \preceq_{\mathrm{R}'} t_1.
\end{aligned}
$$

This concludes the proof that the τ operation is a valid ordered relational query.

□

4.2.4 Derived Operations

The previous subsection defined a set of primitive ordered operations for $\boldsymbol{CA}^T_{\preceq}$. The following section proves that $\boldsymbol{CA}^T_{\preceq}$ is a complete algebra for ordered conjunctive queries in $\boldsymbol{CQ}^T_{\preceq}$. However, for some queries in $\boldsymbol{CQ}^T_{\preceq}$, the equivalent algebraic expression can be simplified significantly with the introduction of certain new operators derived from the primitive operations in $\boldsymbol{CA}^T_{\preceq}$. We list several derived operators here, which will be used frequently later. However, there are many possibilities for the choices of derived operators.

(1) The first operator $\texttt{first}_k$: $\mathbf{R} \rightarrow \mathbf{R}$ is derived from the top operator. It takes an ordered relation as its input and returns as its result the ordered relation containing the first k tuples in the original order.
$\texttt{first}_1\ (\mathbf{R}) = \ \tau(\mathbf{R})$
$\texttt{first}_k\ (\mathbf{R}) = \ \texttt{first}_{k-1}\ (\mathbf{R}) \cup \tau(\mathbf{R} - \texttt{first}_{k-1}\ (\mathbf{R}))$

(2) The last operator $\texttt{last}_k$: $\mathbf{R} \rightarrow \mathbf{R}$ is derived from the first operator. It takes an ordered relation as its input and returns as a result the ordered relation

containing the last k tuples in the original order.
$\texttt{last}_k\ (\mathbf{R}) = \ \nu\ \texttt{first}_k\ (\nu\mathbf{R})$.

It is noteworthy that, for any input ordered relation and a given number k, $\texttt{first}_k\ (\mathbf{R})$ and $\texttt{last}_k\ (\mathbf{R})$ can be expressed with a fixed sequence of ordered operations from $\boldsymbol{CA}^T_{\preceq}$.

Lemma 4.5 *The derived operations* $\texttt{first}_k\ (\mathbf{R})$ *and* $\texttt{last}_k\ (\mathbf{R})$ *can be expressed with a fixed sequence of ordered operations from* $\boldsymbol{CA}^T_{\preceq}$.

Proof: We first prove that $\texttt{first}_k\ (\mathbf{R})$ can be expressed with a fixed sequence of ordered operations from $\boldsymbol{CA}^T_{\preceq}$. This can be proven by induction on k.

Base case: $k = 1$. By definition of first operator,

$$\texttt{first}_1\ (\mathbf{R}) = \tau(\mathbf{R}).$$

Therefore, $\texttt{first}_1$ can be expressed by one ordered operation from $\boldsymbol{CA}^T_{\preceq}$.

Induction step: Assume that $\texttt{first}_{k-1}$ can be expressed by a fixed number of operations from $\boldsymbol{CA}^T_{\preceq}$. We want to prove that this holds for the case k.

By definition of first operator,

$$\texttt{first}_k\ (\mathbf{R}) = \texttt{first}_{k-1}\ (\mathbf{R}) \cup \tau(\mathbf{R} - \texttt{first}_{k-1}\ (\mathbf{R})).$$

Hence, $\texttt{first}_k$ can be expressed by a fixed number of ordered operations from $\boldsymbol{CA}^T_{\preceq}$.

Then, we prove that $\texttt{last}_k$ can be expressed by a fixed number of ordered operations from $\boldsymbol{CA}^T_{\preceq}$.

By definition of last operator,
$\texttt{last}_k\ (\mathbf{R}) = \ \nu\ \texttt{first}_k\ (\nu\mathbf{R})$. Since $\texttt{first}_k$ can be expressed by a fixed number of ordered operations from $\boldsymbol{CA}^T_{\preceq}$, $\texttt{last}_k$ can also be expressed by a fixed number of ordered operations from $\boldsymbol{CA}^T_{\preceq}$.

This concludes the proof. □

4.3 Transformation Rules

This section provides a set of transformation rules for the ordered conjunctive algebra $\boldsymbol{CA}^T_{\preceq}$. Many of them are the same as those in $\boldsymbol{CA}^X_{\preceq}$, and the others are developed exclusively for $\boldsymbol{CA}^T_{\preceq}$.

In ordered relational databases, two ordered queries are equivalent if they have both equivalent data queries and equivalent order queries. The following transformation rules guarantee both the equivalence of data queries and the equivalence of order queries for the ordered queries on both sides. In addition, the transformation rules are given in the algebraic equations which represent both the left-to-right equivalence and the right-to-left equivalence. In these equations, each argument can be either an ordered relation or an expression of ordered operations.

Proposition 4.6 *In addition to the transformation rules in 3.13, the following transformation rules hold in* $\boldsymbol{CA}^{T}_{\preceq}$*:*

($R16$) $\sigma_{\mathrm{p}}(\mathbf{R}_1 \cap \mathbf{R}_2) = \sigma_{\mathrm{p}}(\mathbf{R}_1) \cap \sigma_{\mathrm{p}}(\mathbf{R}_2)$

($R17$) $\pi_A(\mathbf{R}_1 \cap \mathbf{R}_2) = \pi_A(\mathbf{R}_1) \cap \pi_A(\mathbf{R}_2)$

($R18$) $(\mathbf{R}_1 \cap \mathbf{R}_2) \cap \mathbf{R}_3 = \mathbf{R}_1 \cap (\mathbf{R}_2 \cap \mathbf{R}_3)$

($R19$) $\tau(\pi_A(\mathbf{R})) = \pi_A(\tau(\mathbf{R}))$

Proof:
The proof for the rest of transformation rules can be found in the proof of Proposition 3.13. Transformation rules ($R16$), ($R17$), ($R18$), and ($R19$) will be proved as follows:

($R16$) Suppose that $\mathbf{R}_1$ and $\mathbf{R}_2$ share the same schema $(\mathrm{T}, X_1, \ldots, \mathrm{X}_n)$. Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \cap \mathbf{R}_2$. By definition of order intersection,

$$\mathrm{R}' = \{(t, \bar{x}) \mid \mathrm{R}_1(t, \bar{x}) \wedge \mathrm{R}_2(t, \bar{x})\}$$

$$\begin{aligned} \preceq_{\mathrm{R}'} = \{(t_1, t_2) \mid\ & \exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1) \\ & \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2) \\ & \wedge t_1 \preceq_{R_1} t_2)\}. \end{aligned}$$

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \sigma_{\mathrm{p}} \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$
\begin{array}{lcll}
\mathrm{R}'' & = & \{(t,\bar{x}) \mid & \mathrm{R}'(t,\bar{x}) \wedge \mathrm{p}(t,\bar{x})\} \\
& = & \{(t,\bar{x}) \mid & (\mathrm{R}_1(t,\bar{x}) \wedge \mathrm{R}_2(t,\bar{x})) \wedge \ \mathrm{p}(t,\bar{x})\} \\
& = & \{(t,\bar{x}) \mid & (\mathrm{R}_1(t,\bar{x}) \wedge \ \mathrm{p}(t,\bar{x}) \wedge (\mathrm{R}_2(t,\bar{x}) \wedge \ \mathrm{p}(t,\bar{x})\}
\end{array}
$$

$$
\begin{array}{lcll}
\preceq_{\mathrm{R}''} & = & \{(t_1,t_2) \mid & \exists \bar{x}_1.\mathrm{R}'(t_1,\bar{x}_1) \wedge \ \mathrm{p}(t_1,\bar{x}_1) \\
& & & \wedge \exists \bar{x}_2.\mathrm{R}'(t_2,\bar{x}_2) \wedge \ \mathrm{p}(t_2,\bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}'} t_2\} \\
& = & \{(t_1,t_2) \mid & \exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \wedge \mathrm{R}_2(t_1,\bar{x}_1) \wedge \mathrm{p}(t_1,\bar{x}_1) \\
& & & \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \wedge \mathrm{R}_2(t_2,\bar{x}_2) \wedge \mathrm{p}(t_2,\bar{x}_2) \\
& & & \wedge t_1 \preceq_{R_2} t_2\}.
\end{array}
$$

Therefore,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \ \sigma_{\mathrm{p}}(\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle) \cap \sigma_{\mathrm{p}}(\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle).$$

This ends the proof of the transformation rule ($R16$).

($R17$) Suppose that $\mathbf{R}_1$ and $\mathbf{R}_2$ share the same schema $(\mathrm{T}, X_1, \ldots, \mathrm{X}_n)$. Without lost of generality, we assume that $A = (\mathrm{T}, X_1, \ldots, \mathrm{X}_k)$, where $k \leq n$. Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \cap \mathbf{R}_2$. By definition of order intersection,

$$
\begin{array}{lcll}
\mathrm{R}' & = & \{(t,\bar{x}) \mid & \mathrm{R}_1(t,\bar{x}) \wedge \mathrm{R}_2(t,\bar{x})\}
\end{array}
$$

$$
\begin{array}{lcll}
\preceq_{\mathrm{R}'} & = & \{(t_1,t_2) \mid & \exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \mathrm{R}_2(t_1,\bar{x}_1) \\
& & & \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \wedge \mathrm{R}_2(t_2,\bar{x}_2) \\
& & & \wedge t_1 \preceq_{R_1} t_2)\}.
\end{array}
$$

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \pi_A \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$. Then,

$$
\begin{array}{lcll}
\mathrm{R}'' & = & \{(t,x_1,\ldots,x_k) \mid & \exists x_{k+1},\ldots,x_n.\mathrm{R}'(t,\bar{x})\} \\
& = & \{(t,x_1,\ldots,x_k) \mid & \exists x_{k+1},\ldots,x_n.\mathrm{R}_1(t,\bar{x}) \wedge \mathrm{R}_2(t,\bar{x})\} \\
& = & \{(t,x_1,\ldots,x_k) \mid & \exists x_{k+1},\ldots,x_n.\mathrm{R}_1(t,\bar{x}) \wedge \exists x_{k+1},\ldots,x_n.\mathrm{R}_2(t,\bar{x})\}
\end{array}
$$

$$
\begin{array}{lcll}
\preceq_{\mathrm{R}''} & = & \{(t_1,t_2) \mid & \exists \bar{x}_1.\mathrm{R}'(t_1,\bar{x}_1) \wedge \ \exists \bar{x}_2.\mathrm{R}'(t_2,\bar{x}_2) \wedge t_1 \preceq_{\mathrm{R}'} t_2\} \\
& = & \{(t_1,t_2) \mid & (\exists \bar{x}_1.\mathrm{R}_1(t_1,\bar{x}_1) \wedge \mathrm{R}_2(t_1,\bar{x}_1) \\
& & & \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2,\bar{x}_2) \wedge \mathrm{R}_2(t_2,\bar{x}_2)) \\
& & & \wedge t_1 \preceq_{R_1} t_2)\}.
\end{array}
$$

Consequently,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \ \pi_A(\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle) \cap \pi_A(\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle).$$

This completes the proof of the transformation rule ($R17$).

($R18$) Suppose that $\mathbf{R}_1$, $\mathbf{R}_2$ and $\mathbf{R}_3$ share the same schema $(\mathrm{T}, X_1, \ldots, \mathrm{X}_n)$. Let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle = \mathbf{R}_1 \cap \mathbf{R}_2$. By definition of order intersection,

$$\begin{aligned}
\mathrm{R}' &= \{(t, \bar{x}) \mid \quad \mathrm{R}_1(t, \bar{x}) \wedge \mathrm{R}_2(t, \bar{x})\} \\
\\
\preceq_{\mathrm{R}'} &= \{(t_1, t_2) \mid \exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1) \\
&\qquad\qquad \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2) \\
&\qquad\qquad \wedge t_1 \preceq_{R_1} t_2)\}.
\end{aligned}$$

Let $\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle \cap \mathbf{R}_3$. Then,

$$\begin{aligned}
\mathrm{R}'' &= \{(t, \bar{x}) \mid \mathrm{R}'(t, \bar{x}) \wedge \mathrm{R}_3(t, \bar{x})\} \\
&= \{(t, \bar{x}) \mid \mathrm{R}_1(t, \bar{x}) \wedge \mathrm{R}_2(t, \bar{x}) \wedge \mathrm{R}_3(t, \bar{x})\} \\
&= \{(t, \bar{x}) \mid \mathrm{R}_1(t, \bar{x}) \wedge (\mathrm{R}_2(t, \bar{x}) \wedge \mathrm{R}_3(t, \bar{x}))\}
\end{aligned}$$

$$\begin{aligned}
\preceq_{\mathrm{R}''} &= (\exists \bar{x}_1.\mathrm{R}'(t_1, \bar{x}_1) \wedge \mathrm{R}_3(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}'(t_2, \bar{x}_2) \wedge \mathrm{R}_3(t_2, \bar{x}_2)) \\
&\quad \wedge t_1 \preceq_{R'} t_2) \\
&= (\exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge \mathrm{R}_2(t_1, \bar{x}_1) \wedge \mathrm{R}_3(t_1, \bar{x}_1)) \\
&\quad \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge \mathrm{R}_2(t_2, \bar{x}_2) \wedge \mathrm{R}_3(t_2, \bar{x}_2)) \\
&\quad \wedge t_1 \preceq_{R_1} t_2) \\
&= \exists \bar{x}_1.\mathrm{R}_1(t_1, \bar{x}_1) \wedge (\mathrm{R}_2(t_1, \bar{x}_1) \wedge \mathrm{R}_3(t_1, \bar{x}_1)) \\
&\quad \wedge \exists \bar{x}_2.\mathrm{R}_1(t_2, \bar{x}_2) \wedge (\mathrm{R}_2(t_2, \bar{x}_2) \wedge \mathrm{R}_3(t_2, \bar{x}_2)) \\
&\quad \wedge t_1 \preceq_{R_1} t_2).
\end{aligned}$$

Thus,

$$\langle \mathrm{R}'', \preceq_{\mathrm{R}''} \rangle = \langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle \cap (\langle \mathrm{R}_2, \preceq_{\mathrm{R}_2} \rangle \cap \langle \mathrm{R}_3, \preceq_{\mathrm{R}_3} \rangle).$$

This concludes the proof of the transformation rule ($R18$).

($S19$) Without loss of generality, we assume that $\mathrm{A} = \{X_1, \ldots, X_m\}$. By definition of order-preserving projection,

$$\begin{aligned}
\pi_A\langle R, \preceq_R \rangle = \langle \{(t, x_1, \ldots, x_m) \mid & \exists x_{m+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n)\}, \\
\{(t_1, t_2) \mid & \exists x_{11}, \ldots, x_{1n}.\mathrm{R}(t_1, x_{11}, \ldots, x_{1n}) \\
& \wedge \exists x_{21}, \ldots, x_{2n}.\mathrm{R}(t_2, x_{21}, \ldots, x_{2n}) \wedge t_1 \preceq_R t_2\}.
\end{aligned}$$

Then,

$$\begin{aligned}
\tau(\pi_A\langle R, \preceq_R\rangle) &= \langle\{(t, x_1, \ldots, x_m) \mid & \exists x_{m+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n) \\
& & \wedge\neg\exists t_1, \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge t_1 \preceq_{\mathrm{R}} t\}, \\
& \{(t_1, t_2) \mid & \exists x_{11}, \ldots, x_{1n}.\mathrm{R}(t_1, x_{11}, \ldots, x_{1n}) \\
& & \wedge\ \exists x_{21}, \ldots, x_{2n}.\mathrm{R}(t_2, x_{21}, \ldots, x_{2n}) \\
& & \wedge\neg\exists t_{11}, \bar{x}_{11}.\mathrm{R}(t_{11}, \bar{x}_{11}) \wedge t_{11} \preceq_{\mathrm{R}} t_1 \\
& & \wedge\neg\exists t_{21}, \bar{x}_{21}.\mathrm{R}(t_{21}, \bar{x}_{21}) \wedge t_{21} \preceq_{\mathrm{R}} t_2 \\
& & \wedge\ t_1 \preceq_R t_2\}\rangle \\
&= \pi_A(\tau(\mathbf{R})).
\end{aligned}$$

□

4.4 Completeness of $\boldsymbol{CA}^T_{\preceq}$

This section examines the expressive power of the ordered conjunctive algebra $\boldsymbol{CA}^T_{\preceq}$. The completeness of $\boldsymbol{CA}^T_{\preceq}$ are verified with respect to the first-order calculus and proven such that $\boldsymbol{CA}^T_{\preceq}$ is complete in that any ordered conjunctive query in $\boldsymbol{CQ}^T_{\preceq}$ can be expressed by a finite sequence of ordered operations from $\boldsymbol{CA}^T_{\preceq}$. We first present a number of notations that we use later.

Definition 4.7 (Subsequences of Ordered Relations) *Let $\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle$ be an ordered relation, and ψ be a first-order formula defined by the BNF rule:*

$$\psi \;=\; \mathrm{R}(t, \bar{x}) \mid t \preceq_R t' \mid \neg\psi \mid \psi \wedge \psi \mid \exists x.\psi \mid \exists t.\psi.$$

A subsequence *S of the ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}}\rangle$ is a set of tuples which satisfy a subformula of ψ, ψ', with one $\boldsymbol{t}$-variable as the only free variable. We say the subsequence S is specified by the subformula ψ'.*

Definition 4.8 (Combinations of Subsequences) *Let $\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1}\rangle, \ldots, \langle \mathrm{R}_n, \preceq_{\mathrm{R}_n}\rangle$ be n ordered relations, and ψ be a first-order formula defined by the BNF rule:*

$$\psi \;=\; \mathrm{R}(t, \bar{x}) \mid t \preceq_R t' \mid \neg\psi \mid \psi \wedge \psi \mid \exists x.\psi \mid \exists t.\psi.$$

Let S_{j_i} be an arbitrary subsequence of $\langle \mathrm{R}_i, \preceq_{\mathrm{R}_i}\rangle$ specified by a subformula of ψ, for $i = 1, ..., n$. A combination of subsequences *$(S_{j_1}, \ldots, S_{j_n})$ is a set of the tuples, each of which is a combination of tuples from $S_{j_1}, \ldots, S_{j_n}$, respectively.*

Definition 4.9 (Minimal Subsequences) *Let* $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ *be an ordered relation, and* ψ *be a first-order formula defined by the BNF rule:*

$$\psi \;=\; \mathrm{R}(t, \bar{x}) \mid t \preceq_R t' \mid \neg\psi \mid \psi \wedge \psi \mid \exists x.\psi \mid \exists t.\psi.$$

A subsequence S *of an ordered relation* $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$*, specified by a subformula of* ψ*, is a* minimal subsequence *if there is no subsequence* S' *in* $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ *which is specified by a subformula of* ψ *and satisfies* $S' \subset S$*.*

Then, a lemma is proved for a special case of $\boldsymbol{CQ}^T_\preceq$, the ω queries. An ω query in $\boldsymbol{CQ}^T_\preceq$ is an ordered conjunctive query in $\boldsymbol{CQ}^T_\preceq$ which has an identity data query. The output of an ω query has an identical data relation as the input ordered relation; however, it might have a different order relation.

Lemma 4.10 *Any* ω *query in* $\boldsymbol{CQ}^T_\preceq$ *is equivalent to an ordered query expressed by a finite sequence of ordered operations from* $\boldsymbol{CA}^T_\preceq$*.*

Proof: Let $\langle \varphi, \psi \rangle$, a $\boldsymbol{CQ}^T_\preceq$ query on $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, be an ω query, and $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$ be the output ordered relation. By definition of the ω query, $\langle \varphi, \psi \rangle$ hence has the form

$$\langle \varphi, \psi \rangle(\langle R, \preceq_R \rangle) = \langle \{(t, \bar{x}) | \mathrm{R}(t, \bar{x})\},\ \{(t_1, t_2) | \psi(t_1, t_2)\} \rangle,$$

and

$$\psi = \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \psi'(t_1, t_2).$$

By definition of $\boldsymbol{CQ}^T_\preceq$, ψ' is a first-order formula defined by the BNF rule

$$\psi' \;=\; \mathrm{R}(t, \bar{x}) \mid t \preceq_R t' \mid \neg\psi' \mid \psi' \wedge \psi' \mid \exists x.\psi' \mid \exists t.\psi'.$$

Since there is no data predicates in ψ', $\boldsymbol{x}$-variables appear only in form of $\exists x.\mathrm{R}(t, x)$ in the scope of each $\boldsymbol{t}$-quantifiers $\exists t$. Therefore, to simplify the form of ψ', we omit $\boldsymbol{x}$-variables in ψ' by replacing $\exists x.\mathrm{R}(t, x)$ with $\mathrm{R}(t)$. In this way, we only consider ψ' as a first-order formula defined by the BNF rule

$$\psi' \;=\; \mathrm{R}(t) \mid t \preceq_R t' \mid \neg\psi' \mid \psi' \wedge \psi' \mid \exists t.\psi'.$$

By the BNF rule, ψ' is a boolean combination of existentially quantified formulas with only t_1 and t_2 free, we rearrange the top level boolean connectives, and rewrite ψ' in a disjunction of conjunctions in the form

$$\bigvee_i (\alpha^i_1(t_1) \wedge \alpha^i_2(t_2)) \vee \bigvee_j \beta^j(t_1, t_2),$$

where $\alpha_1^i(t_1)$ and $\alpha_2^i(t_2)$ are existentially quantified formulas with only t_1 free and existentially quantified formulas with only t_2 free, respectively; and $\beta^j(t_1, t_2)$ is an existentially quantified formula with both t_1 and t_2 free. If t_1 and t_2 satisfy one of the disjuncts $\alpha_1^i(t_1) \wedge \alpha_2^i(t_2)$ or $\beta^j(t_1, t_2)$, then $t_1 \preceq_{\mathrm{R}'} t_2$.

We first consider the first part of ψ', which is $\bigvee_i(\alpha_1^i(t_1) \wedge \alpha_2^i(t_2))$. We will prove that $\bigvee_i(\alpha_1^i(t_1) \wedge \alpha_2^i(t_2))$ is equivalent to a disjunction of the conjunctions of the form

$$\bigvee_i(\alpha^i_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \alpha^i_{>t_1}(t_1)) \wedge (\alpha^i_{<t_2}(t_2) \wedge \mathrm{R}(t_2) \wedge \alpha^i_{>t_2}(t_2)),$$

where

- the conjunct $\alpha^i_{<t_j}(t_j)$ $(j = 1, 2)$ is an existentially quantified formula with only t_j free. In $\alpha^i_{<t_j}(t_j)$, all quantifiers are relativized to $< t_j$, which means that the scopes of all quantifiers are restricted to before t_j.
- $\alpha^i_{>t_j}(t_j)$ $(j = 1, 2)$ is an existentially quantified formula with only t_j free. In $\alpha^i_{<>t_j}(t_j)$, all quantifiers are relativized to $> t_j$, which means that the scopes of all quantifiers are restricted to after t_j.

By the Separation Property (ref. Corollary 9.3.3. [39]), the conjunct $\alpha_1^i(t_1)$ is equivalent to a disjunction of the conjunctions of the form

$$\bigvee_i(\alpha^i_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \alpha^i_{>t_1}(t_1)),$$

and $\alpha_2^i(t_2)$ is equivalent to a disjunction of the conjunctions of the form

$$\bigvee_j(\alpha^j_{<t_2}(t_2) \wedge \mathrm{R}(t_2) \wedge \alpha^j_{>t_2}(t_2)).$$

Thus, each conjunction $\alpha_1^i(t_1) \wedge \alpha_2^i(t_2)$ is equivalent to

$$\bigvee_{i,j}(\alpha^i_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \alpha^i_{>t_1}(t_1)) \wedge (\alpha^j_{<t_2}(t_2) \wedge \mathrm{R}(t_2) \wedge \alpha^j_{>t_2}(t_2)).$$

Consequently, the first part of ψ', $\bigvee_i(\alpha_1^i(t_1) \wedge \alpha_2^i(t_2))$, is equivalent to a disjunction of conjunctions in the form

$$\bigvee_i(\alpha^i_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \alpha^i_{>t_1}(t_1)) \wedge (\alpha^i_{<t_2}(t_2) \wedge \mathrm{R}(t_2) \wedge \alpha^i_{>t_2}(t_2)).$$

Then, we need to prove that the conjunct $\alpha^i_{<t_1}(t_1)$ is equivalent to a normal form

$$\exists s_1, ..., s_{i_1}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_1}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_1} \preceq_{\mathrm{R}} t_1$$
$$\wedge$$
$$\neg\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_2} \preceq_{\mathrm{R}} t_1.$$

This can be proven by induction on the depth of quantifiers n in $\alpha^i_{<t_1}(t_1)$. Recall that $\alpha^i_{<t_1}(t_1)$ is defined by the BNF rule

$$\psi' \ = \ \mathrm{R}(t) \mid t \preceq_R t' \mid \neg\psi' \mid \psi' \wedge \psi' \mid \exists t.\psi'.$$

Base case: In the case of $n = 0$, t_1 is the only $\boldsymbol{t}$-variable in $\alpha^i_{<t_1}(t_1)$, so $\alpha^i_{<t_1}(t_1)$ is empty in this case.

In the case of $n = 1$, by construction rule, $\alpha^i_{<t_1}(t_1)$ is either $\exists s_1.\mathrm{R}(s_1) \wedge s_1 \preceq_{\mathrm{R}} t_1$ or $\neg\exists s_1.\mathrm{R}(s_1) \wedge s_1 \preceq_{\mathrm{R}} t_1$. It is already in the normal form.

Inductive step: Assume that the conjunct $\alpha^i_{<t_1}(t_1)$ is equivalent to a formula in the normal form in the case $n-1$. We want to show that this hypothesis holds in the case n.

Let $\varphi(t_1)$ be an arbitrary existentially quantified formula with only t_1 free, and all quantifiers are relativized to $< t_1$. By the induction hypothesis, the formula $\varphi(t_1)$ is equivalent to a normal form

$$\exists s_1, ..., s_{i_1}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_1}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_1} \preceq_{\mathrm{R}} t_1$$
$$\wedge\neg\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_2} \preceq_{\mathrm{R}} t_1,$$

where $i_1, i_2 \leq n-1$. It is sufficient to treat the cases

$$\alpha^i_{<t_1}(t_1) = \exists s.\mathrm{R}(s) \wedge \varphi(t_1) \wedge \varphi'(s, s_1, ..., s_{i_1}, t_1),$$

and

$$\neg\exists s.\mathrm{R}(s) \wedge \varphi(t_1) \wedge \varphi'(s, s_1, ..., s_{i_2}, t_1),$$

where φ' is a conjunction of predicates of the form $t_i \preceq_{\mathrm{R}} t_j$.

First, we consider the case $\alpha^i_{<t_1}(t_1) = \exists s.\varphi(t_1) \wedge \mathrm{R}(s) \wedge \varphi'(s, s_1, ..., s_{i_1}, t_1)$. Since all quantifiers are relativized to $< t_1$ in the formula $\alpha^i_{<t_1}(t_1)$, and we already have $s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_1} \preceq_{\mathrm{R}} t_1$, then $\varphi'(s, s_1, ..., s_{i_1}, t_1)$ has one of the following forms:

1. $s = s_j$ ($j = 1, ..., i_1$),
2. $s \preceq_{\mathrm{R}} s_1$,
3. $s_j \preceq_{\mathrm{R}} s \preceq_{\mathrm{R}} s_{j+1}$ ($j = 1, ..., i_1 - 1$),

4. $s_{i_1} \preceq_{\mathrm{R}} s \preceq_{\mathrm{R}} t_1$.

In the first case, s is one of s_j for $j = 1, ..., i_1$. Thus, the conjunct $\alpha^i_{<t_1}(t_1)$ is equivalent to φ, and hence is equivalent to a normal form by the induction hypothesis. In the last three cases, $\alpha^i_{<t_1}(t_1)$ is equivalent to

$$\exists s'_1, ..., s'_{i_1}, s'_{i_1+1}.\mathrm{R}(s'_1) \wedge \ldots \wedge \mathrm{R}(s'_{i_1}) \wedge \mathrm{R}(s'_{i_1+1}) \wedge s'_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s'_{i_1} \preceq_{\mathrm{R}} s'_{i_1+1} \preceq_{\mathrm{R}} t_1$$
$$\wedge$$
$$\neg\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_2} \preceq_{\mathrm{R}} t_1,$$

where $s'_1, ..., s'_{i_1}, s'_{i_1+1}$ is a rearranged sequence of s and $s_1, ..., s_{i_1}$, and s is placed accordingly for the different cases of φ'.

Then, we consider the case $\alpha^i_{<t_1}(t_1) = \neg\exists s.\varphi(t_1) \wedge \mathrm{R}(s) \wedge \varphi'(s, s_1, ..., s_{i_2}, t_1)$. Since all quantifiers are relativized to $< t_1$ in the formula $\alpha^i_{<t_1}(t_1)$, and we already have $\neg\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_2} \preceq_{\mathrm{R}} t_1$, then $\varphi'(s, s_1, ..., s_{i_2}, t_1)$ has one of the following forms:

1. $s = s_j$ ($j = 1, ..., i_2$),
2. $s \preceq_{\mathrm{R}} s_1$,
3. $s_j \preceq_{\mathrm{R}} s \preceq_{\mathrm{R}} s_{j+1}$ ($j = 1, ..., i_2 - 1$),
4. $s_{i_1} \preceq_{\mathrm{R}} s \preceq_{\mathrm{R}} t_1$.

In the first case, s is one of s_j for $j = 1, ..., i_2$. Thus, $\alpha^i_{<t_1}(t_1)$ is equivalent to φ, and hence is equivalent to a normal form by the induction hypothesis. In the last three cases, $\alpha^i_{<t_1}(t_1)$ is equivalent to

$$\exists s_1, ..., s_{i_1}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_1}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_1} \preceq_{\mathrm{R}} t_1$$
$$\wedge$$
$$\neg\exists s'_1, ..., s'_{i_1}, s'_{i_2+1}.\mathrm{R}(s'_1) \wedge \ldots \wedge \mathrm{R}(s'_{i_2}) \wedge \mathrm{R}(s'_{i_2+1}) \wedge s'_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s'_{i_2} \preceq_{\mathrm{R}} s'_{i_2+1} \preceq_{\mathrm{R}} t_1,$$

where $s'_1, ..., s'_{i_2}, s'_{i_2+1}$ is a rearranged sequence of s and $s_1, ..., s_{i_2}$, and s is placed accordingly for the different cases of φ'.

We have shown that $\alpha^i_{<t_1}(t_1)$ is equivalent to a formula in the normal form

$$\exists s_1, ..., s_{i_1}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_1}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_1} \preceq_{\mathrm{R}} t_1$$
$$\wedge$$
$$\neg\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_2} \preceq_{\mathrm{R}} t_1.$$

Analogously, we can prove that $\alpha^i_{>t_1}(t_1)$ is equivalent to a normal form

$$\begin{array}{l}\exists s_1, ..., s_{i_1}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_1}) \wedge t_1 \preceq_\mathrm{R} s_1 \preceq_\mathrm{R} ... \preceq_\mathrm{R} s_{i_1} \\ \wedge \\ \neg\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge t_1 \preceq_\mathrm{R} s_1 \preceq_\mathrm{R} ... \preceq_\mathrm{R} s_{i_2}.\end{array}$$

By the definitions of ordered operations in $\boldsymbol{CA}^T_\preceq$, the conjunct in the normal form of $\alpha^i_{<t_1}(t_1)$,

$$\exists s_1, ..., s_{i_1}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_1}) \wedge s_1 \preceq_\mathrm{R} ... \preceq_\mathrm{R} s_{i_1} \preceq_\mathrm{R} t_1,$$

can be expressed with an ordered algebraic expression in $\boldsymbol{CA}^T_\preceq$,

$$\langle \mathrm{R}, \preceq_\mathrm{R}\rangle - \mathtt{first}_{i_1}\ (\langle \mathrm{R}, \preceq_\mathrm{R}\rangle).$$

Similarly, the conjunct

$$\neg(\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge s_1 \preceq_\mathrm{R} ... \preceq_\mathrm{R} s_{i_2} \preceq_\mathrm{R} t_1)$$

can be expressed with an ordered algebraic expression in $\boldsymbol{CA}^T_\preceq$,

$$\mathtt{first}_{i_1}\ (\langle \mathrm{R}, \preceq_\mathrm{R}\rangle).$$

In analogy, the conjunct in the normal form of $\alpha^i_{>t_1}(t_1)$, which is

$$\exists s_1, ..., s_{i_1}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_1}) \wedge t_1 \preceq_\mathrm{R} s_1 \preceq_\mathrm{R} ... \preceq_\mathrm{R} s_{i_1},$$

can be expressed with an ordered algebraic expression in $\boldsymbol{CA}^T_\preceq$,

$$\langle \mathrm{R}, \preceq_\mathrm{R}\rangle - \mathtt{last}_{i_1}\ (\langle \mathrm{R}, \preceq_\mathrm{R}\rangle).$$

The conjunct

$$\neg(\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge t_1 \preceq_\mathrm{R} s_1 \preceq_\mathrm{R} ... \preceq_\mathrm{R} s_{i_2})$$

can be expressed with an ordered algebraic expression in $\boldsymbol{CA}^T_\preceq$,

$$\mathtt{last}_{i_1}\ (\langle \mathrm{R}, \preceq_\mathrm{R}\rangle).$$

In general, we can express the conjunction $\alpha^i_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \alpha^i_{>t_1}(t_1)$ with an ordered algebraic expression in $\boldsymbol{CA}^T_\preceq$,

$$E_1 \cap ... \cap E_k,$$

where E_i $(i = 1, ...k)$ could be one of the following forms:

1. $\texttt{first}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
2. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \texttt{first}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
3. $\texttt{last}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
4. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \texttt{last}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$.

Intuitively, $\alpha^i_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \alpha^i_{>t_1}(t_1)$ represents a subsequence S of the ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$.

Analogously, we can show that $\alpha^i_{<t_2}(t_2) \wedge \mathrm{R}(t_2) \wedge \alpha^i_{>t_2}(t_2)$ can be expressed with an ordered algebraic expression in $\boldsymbol{CA}^T_{\preceq}$. Intuitively, $\alpha^i_{<t_2}(t_2) \wedge \mathrm{R}(t_2) \wedge \alpha^i_{>t_2}(t_2)$ represents a subsequence S of the ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$.

We have proved that the conjuncts $\alpha^i_1(t_1)$ and $\alpha^i_2(t_2)$ can be expressed with a sequence of ordered operations in $\boldsymbol{CA}^T_{\preceq}$. Suppose that subsequences S_1 and S_2 are specified by $\alpha^i_1(t_1)$ and $\alpha^i_2(t_2)$, respectively. If any pair of tuples t_1 and t_2 satisfies $\alpha^i_1(t_1) \wedge \alpha^i_2(t_2)$, then $t_1 \in S_1$ and $t_2 \in S_2$ hold, and therefore S_1 is prior to S_2 in the output order $\preceq_{\mathrm{R}'}$.

Let $S_1, \ldots, S_n$ be subsequences specified by $\bigvee_i \alpha^i_1(t_1) \wedge \alpha^i_2(t_2)$. Without lost of generality, we assume that $S_1, \ldots, S_n$ is a set of pairwise disjoint minimal subsequences, and $\bigcup_{i=1,\ldots,n} S_i = \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. (If subsequences are not disjoint or not minimal, we can always separate them into minimal subsequences.) We want to show that $\bigvee_i(\alpha^i_1(t_1) \wedge \alpha^i_2(t_2))$ defines a linear order on subsequences $S_1, \ldots, S_n$.

This is proved by contradiction. Suppose that $\bigvee_i(\alpha^i_1(t_1) \wedge \alpha^i_2(t_2))$ does not define a linear order among the subsequences $S_1, \ldots, S_n$. There are two cases:

Case 1: Assume that there exist two subsequences, say S_1 and S_2, such that both $S_1 \preceq_{R'} S_2$ and $S_2 \preceq_{R'} S_1$ hold. If $t_1 \in S_1 \wedge t_2 \in S_2$, then by the assumption $(t_1, t_2) \in \preceq_{R'}$ and $(t_2, t_1) \in \preceq_{R'}$ hold at the same time. This is a contradiction to the fact that the output order $\preceq_{R'}$ is a linear order. Hence, the assumption in Case 1 does not hold.

Case 2: Assume that there exist two subsequences, say S_1 and S_2, such that neither $S_1 \preceq_{R'} S_2$ nor $S_2 \preceq_{R'} S_1$ holds; their order in output is not specified by $\bigvee_i \alpha^i$. By this assumption, for any pair of tuples $t_1 \in P_1$ land $t_2 \in P_2$, their order in the output is not defined by $\bigvee_i \alpha^i_1(t_1) \wedge \alpha^i_2(t_2)$. Because the output order is a linear order on all tuples, the order of t_1 and t_2 must be decided by ψ'. Suppose that ψ' decides $t_1 \preceq_{R'} t_2$. Since S_1 and S_2 are minimal, all pairs of tuples from S_1 and S_2 respectively must have the same order if they have any order at all. Then, for any

pair of tuples t_1 and t_2, and $t_1 \in S_1 \wedge t_2 \in S_2$, $t_1 \preceq_{R'} t_2$ holds. It follows $S_1 \preceq_{R'} S_2$, which is a contradiction to the assumption.

We now consider the second part of ψ', $\bigvee_j \beta^j(t_1, t_2)$, which is an existentially quantified formula with both t_1 and t_2 free. Each conjunct $\beta^j(t_1, t_2)$ can be rewritten into

$$\beta^j_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \beta^j_{(t_1,t_2)}(t_1, t_2) \wedge \mathrm{R}(t_2) \wedge \beta^j_{>t_2}(t_2),$$

or

$$\beta^j_{<t_2}(t_2) \wedge \mathrm{R}(t_2) \wedge \beta^j_{(t_2,t_1)}(t_1, t_2) \wedge \mathrm{R}(t_1) \wedge \beta^j_{>t_1}(t_1),$$

where $\beta^j_{<t_1}(t_1)$ is an existentially quantified formula with only t_1 free, in which all quantifiers relativized to $< t_1$; $\beta^j_{>t_2}(t_2)$ is an existentially quantified formula with only t_2 free, in which all quantifiers relativized to $> t_2$; and $\beta^j_{(t_1,t_2)}(t_1, t_2)$ is an existentially quantified formula with both t_1 and t_2 free, in which all quantifiers relativized to between t_1 and t_2. The conjuncts $\beta^j_{<t_2}(t_2)$, $\beta^j_{(t_2,t_1)}(t_1, t_2)$ and $\beta^j_{>t_1}(t_1)$ are defined analogously.

As we proved earlier, the conjunct $\beta^j_{<t_1}(t_1)$ is equivalent to a normal form

$$\begin{aligned}&\exists s_1, ..., s_{i_1}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_1}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_1} \preceq_{\mathrm{R}} t_1\\ &\wedge \neg\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_2} \preceq_{\mathrm{R}} t_1.\end{aligned}$$

And the conjunct $\beta^j_{>t_2}(t_2)$ is equivalent to a normal form

$$\begin{aligned}&\exists s_1, ..., s_{i_1}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_1}) \wedge t_1 \preceq_{\mathrm{R}} s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_1}\\ &\wedge \neg\exists s_1, ..., s_{i_2}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_2}) \wedge t_1 \preceq_{\mathrm{R}} s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_2}.\end{aligned}$$

We now want to prove that the conjunct $\beta^j_{(t_1,t_2)}(t_1, t_2)$ is equivalent to a normal form

$$\begin{aligned}&\exists s_1, ..., s_{i_3}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_3}) \wedge t_1 \preceq_{\mathrm{R}} s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_3} \preceq_{\mathrm{R}} t_2\\ &\wedge \neg\exists s_1, ..., s_{i_4}.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_{i_4}) \wedge t_1 \preceq_{\mathrm{R}} s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_{i_4} \preceq_{\mathrm{R}} t_2.\end{aligned}$$

This can be proven by induction on the depth of quantifiers n in $\beta^j_{(t_1,t_2)}(t_1, t_2)$.

Base case: In the case of $n = 0$, t_1 and t_2 are the only $\boldsymbol{t}$-variables in $\beta^j_{(t_1,t_2)}(t_1, t_2)$, hence $\beta^j_{(t_1,t_2)}(t_1, t_2)$ could only be $t_1 \preceq_{\mathrm{R}} t_2$ or $\neg t_1 \preceq_{\mathrm{R}} t_2$ in this case.

In the case of $n = 1$, by the construction rule, $\beta^j_{(t_1,t_2)}(t_1, t_2)$ is either $\exists s.\mathrm{R}(s) \wedge t_1 \preceq_{\mathrm{R}} s \wedge s \preceq_{\mathrm{R}} t_2$ or $\neg\exists s.\mathrm{R}(s) \wedge t_1 \preceq_{\mathrm{R}} s \wedge s \preceq_{\mathrm{R}} t_2$. It is already in the normal form.

Inductive step: Assume that the conjunct $\beta^j_{(t_1,t_2)}(t_1,t_2)$ is equivalent to a formula in the normal form in the case $n-1$. We want to show that this hypothesis holds in the case n.

Let $\varphi(t_1,t_2)$ be an arbitrary existentially quantified formula with only t_1 and t_2 free, in which all quantifiers are relativized to between t_1 and t_2. By the induction hypothesis, $\varphi(t_1,t_2)$ is equivalent to a normal form

$$\begin{aligned}&\exists s_1,...,s_{i_3}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_3})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_3}\preceq_\mathrm{R} t_2\\&\wedge\neg\exists s_1,...,s_{i_4}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_4})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_4}\preceq_\mathrm{R} t_2,\end{aligned}$$

where $i_3,i_4\leq n-1$. It is sufficient to treat the cases

$$\beta^j_{(t_1,t_2)}(t_1,t_2)=\exists s.\mathrm{R}(s)\wedge\varphi(t_1,t_2)\wedge\varphi'(s,s_1,...,s_{i_3},t_1,t_2),$$

and

$$\beta^j_{(t_1,t_2)}(t_1,t_2)=\neg\exists s.\mathrm{R}(s)\wedge\varphi(t_1,t_2)\wedge\varphi'(s,s_1,...,s_{i_3},t_1,t_2),$$

where φ' is a conjunction of predicates in form of $t_i\preceq_\mathrm{R} t_j$.

First, we consider the case of

$$\beta^j_{(t_1,t_2)}(t_1,t_2)=\exists s.\mathrm{R}(s)\wedge\varphi(t_1,t_2)\wedge\varphi'(s,s_1,...,s_{i_3},t_1).$$

Since all quantifiers are relativized to between t_1 and t_2 in $\beta^j_{(t_1,t_2)}(t_1,t_2)$, by the induction hypothesis, $\varphi(t_1,t_2)$ is equivalent to a normal form

$$\begin{aligned}&\exists s_1,...,s_{i_3}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_3})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_3}\preceq_\mathrm{R} t_2\\&\wedge\neg\exists s_1,...,s_{i_4}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_4})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_4}\preceq_\mathrm{R} t_2.\end{aligned}$$

Thus, the conjunct $\varphi'(s,s_1,...,s_{i_3},t_1,t_2)$ has one of the following forms:

1. $s=s_j$ ($j=1,...,i_3$),
2. $t_1\preceq_\mathrm{R} s\preceq_\mathrm{R} s_1$,
3. $s_j\preceq_\mathrm{R} s\preceq_\mathrm{R} s_{j+1}$ ($j=1,...,i_3-1$),
4. $s_{i_3}\preceq_\mathrm{R} s\preceq_\mathrm{R} t_2$.

In the first case, s is one of s_j for $j=1,...,i_3$. Thus, the conjunct $\beta^j_{(t_1,t_2)}(t_1,t_2)$ is equivalent to φ, and therefore is equivalent to a normal form by the induction hypothesis,

$$\begin{aligned}&\exists s_1,...,s_{i_3}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_3})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_3}\preceq_\mathrm{R} t_2\\&\wedge\neg\exists s_1,...,s_{i_4}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_4})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_4}\preceq_\mathrm{R} t_2.\end{aligned}$$

In the last three cases, the conjunct $\beta^j_{(t_1,t_2)}(t_1,t_2)$ is equivalent to

$$\begin{aligned}
&\exists s'_1,...,s'_{i_1},s'_{i_3+1}.\mathrm{R}(s'_1)\wedge\ldots\wedge\mathrm{R}(s'_{i_3})\wedge\mathrm{R}(s'_{i_3+1})\\
&\wedge t_1\preceq_\mathrm{R} s'_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s'_{i_3}\preceq_\mathrm{R} s'_{i_3+1}\preceq_\mathrm{R} t_2\\
&\wedge\\
&\neg\exists s_1,...,s_{i_4}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_4})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_4}\preceq_\mathrm{R} t_2,
\end{aligned}$$

where $s'_1,...,s'_{i_3},s'_{i_3+1}$ is a rearranged sequence of s and $s_1,...,s_{i_3}$, and s is placed accordingly for the different cases of φ'. Therefore, $\beta^j_{(t_1,t_2)}(t_1,t_2)$ is equivalent to a normal form in the last three cases.

Analogously, we can prove that in the case of

$$\beta^j_{(t_1,t_2)}(t_1,t_2)=\neg\exists s.\mathrm{R}(s)\wedge\varphi(t_1,t_2)\wedge\varphi'(s,s_1,...,s_{i_3},t_1),$$

the conjunct $\beta^j_{(t_1,t_2)}(t_1,t_2)$ is equivalent to a normal form. Until now, we have proven that $\beta^j_{(t_1,t_2)}(t_1,t_2)$ is equivalent to a normal form

$$\begin{aligned}
&\exists s_1,...,s_{i_3}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_3})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_3}\preceq_\mathrm{R} t_2\\
&\wedge\neg\exists s_1,...,s_{i_4}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_4})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_4}\preceq_\mathrm{R} t_2.
\end{aligned}$$

We now need to prove that in each conjunct $\beta^j(t_1,t_2)$, the conjunct $\beta^j_{(t_1,t_2)}(t_1,t_2)$ has the depth of quantifiers at most 0. We prove this by contradiction. Assume that we have a conjunct $\beta^j(t_1,t_2)$ in the form

$$\beta^j_{<t_1}(t_1)\wedge\mathrm{R}(t_1)\wedge\beta^j_{(t_1,t_2)}(t_1,t_2)\wedge\mathrm{R}(t_2)\wedge\beta^j_{>t_2}(t_2),$$

where $\beta^j_{(t_1,t_2)}(t_1,t_2)$ is in the normal form

$$\begin{aligned}
&\exists s_1,...,s_{i_3}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_3})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_3}\preceq_\mathrm{R} t_2\\
&\wedge\neg\exists s_1,...,s_{i_4}.\mathrm{R}(s_1)\wedge\ldots\wedge\mathrm{R}(s_{i_4})\wedge t_1\preceq_\mathrm{R} s_1\preceq_\mathrm{R}...\preceq_\mathrm{R} s_{i_4}\preceq_\mathrm{R} t_2.
\end{aligned}$$

This means that for any pair of t_1 and t_2 satisfying $\beta^j_{<t_1}(t_1)$ and $\beta^j_{>t_2}(t_2)$, if there are at least i_3 tuples in the input and at most i_4-1 tuples between t_1 and t_2, then the pair of t_1 and t_2 keeps the original order in the output.

Since the output order is a total and linear order among all input tuples, there must exist other conjuncts in $\bigvee_j\beta^j(t_1,t_2)$, which define the order for the rest of pairs of t_1 and t_2. The rest of pairs of t_1 and t_2 must satisfy $\beta^j_{<t_1}(t_1)$ and $\beta^j_{>t_2}(t_2)$, and have no at least i_3 tuples in the input and at most i_4-1 tuples between t_1 and t_2. There exist two cases for the rest of pairs of t_1 and t_2:

Case 1: All of the rest of pairs t_1 and t_2 also keep the original order, which means that all pairs of t_1 and t_2 satisfying $\beta^j_{<t_1}(t_1)$ and $\beta^j_{>t_2}(t_2)$ keep the original order. In other words, there exists another conjunct $\beta^{j'}(t_1, t_2)$ which is equivalent to

$$\beta^{j'}_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \beta^{j'}_{(t_1,t_2)}(t_1, t_2) \wedge \mathrm{R}(t_2) \wedge \beta^{j'}_{>t_2}(t_2),$$

where

$$\begin{aligned} \beta^{j'}_{<t_1} &= \beta^j_{<t_1}, \\ \beta^{j'}_{>t_2} &= \beta^j_{>t_2}, \\ \beta^{j'}_{(t_1,t_2)} &= \neg\beta^j_{(t_1,t_2)}. \end{aligned}$$

Therefore,

$$\begin{aligned} &\beta^{j'}(t_1, t_2) \vee \beta^j(t_1, t_2) \\ &= (\beta^j_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \beta^j_{(t_1,t_2)}(t_1, t_2) \wedge \mathrm{R}(t_2) \wedge \beta^j_{>t_2}(t_2)) \\ &\quad \vee(\beta^{j'}_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \beta^{j'}_{(t_1,t_2)}(t_1, t_2) \wedge \mathrm{R}(t_2) \wedge \beta^{j'}_{>t_2}(t_2)) \\ &= \beta^j_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \mathrm{R}(t_2) \wedge \beta^j_{>t_2}(t_2). \end{aligned}$$

Thus, the formula $\beta^{j'}(t_1, t_2) \vee \beta^j(t_1, t_2)$ has no quantifiers on both t_1 and t_2, and therefore is a conjunction in $\bigvee_i \alpha^i$, the first part of ψ'.

Case 2: At least a part of the rest pairs of t_1 and t_2 satisfying $\beta^j_{<t_1}(t_1)$ and $\beta^j_{>t_2}(t_2)$ has the reverse order in the output. In other words, there exists at least a conjunct $\beta^{j'}(t_2, t_1)$ which is equivalent to

$$\beta^{j'}_{<t_2}(t_2) \wedge \mathrm{R}(t_2) \wedge \beta^{j'}_{(t_2,t_1)}(t_1, t_2) \wedge \mathrm{R}(t_1) \wedge \beta^{j'}_{>t_1}(t_1),$$

where

$$\begin{aligned} \beta^{j'}_{<t_2} &= \beta^j_{<t_1}, \\ \beta^{j'}_{>t_1} &= \beta^j_{>t_2}, \\ \beta^{j'}_{(t_2,t_1)} &= (\neg\beta^j_{(t_1,t_2)}) \wedge \varphi(t_1, t_2), \end{aligned}$$

and $\varphi(t_1, t_2)$ is an arbitrary formula on t_1 and t_2.

Since $\beta^{j'}_{(t_2,t_1)}$ is a quantified formula with both t_1 and t_2 free, it is equivalent to a normal form

$$\begin{aligned} &\exists s_1, ..., s_m.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_m) \wedge t_2 \preceq_\mathrm{R} s_1 \preceq_\mathrm{R} ... \preceq_\mathrm{R} s_m \preceq_\mathrm{R} t_1 \\ &\wedge \neg\exists s_1, ..., s_n.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_n) \wedge t_2 \preceq_\mathrm{R} s_1 \preceq_\mathrm{R} ... \preceq_\mathrm{R} s_n \preceq_\mathrm{R} t_1. \end{aligned}$$

Since any pair of t_1 and t_2 satisfying $\beta^{j'}(t_2, t_1)$ is in the output order, a pair of t_1 and t_2, satisfying $\beta^{j'}_{<t_2}$, $\beta^{j'}_{>t_1}$ and $\beta^{j'}_{(t_2,t_1)}$, has the reverse order in the output. Then, the pair of t_1 and t_2, which satisfies $\beta^{j'}_{<t_2}$ and $\beta^{j'}_{>t_1}$, and has at least m and at most n tuples between them, has the reverse order in the output. This is impossible because one tuple cannot exchange with more than one tuple simultaneously while keeping the output order as a total linear order for any input. The only possibility is that $\beta^{j'}_{(t_2,t_1)}$ is either $t_1 \preceq_{\mathrm{R}} t_2$ or $\neg t_1 \preceq_{\mathrm{R}} t_2$. As a consequence, the conjunct $\beta^{j'}_{(t_2,t_1)}$ has the quantifier depth 0.

We have proven that in the second part of ψ', which is $\bigvee_j \beta^j(t_1, t_2)$, each conjunct $\beta^j(t_1, t_2)$ is equivalent to a normal form

$$\beta^j_{<t_1}(t_1) \wedge \mathrm{R}(t_1) \wedge \beta^j_{(t_1,t_2)}(t_1, t_2) \wedge \mathrm{R}(t_2) \wedge \beta^j_{>t_2}(t_2),$$

or

$$\beta^j_{<t_2}(t_2) \wedge \mathrm{R}(t_2) \wedge \beta^j_{(t_2,t_1)}(t_1, t_2) \wedge \mathrm{R}(t_1) \wedge \beta^j_{>t_1}(t_1),$$

where $\beta^j_{(t_1,t_2)}$ and $\beta^j_{(t_2,t_1)}$ are either $t_1 \preceq_{\mathrm{R}} t_2$ or $\neg t_1 \preceq_{\mathrm{R}} t_2$.

As we proved earlier, $\beta^j_1(t_1)$ and $\beta^j_2(t_2)$ specify subsequences of input, and can be expressed by a sequence of ordered operations in $\boldsymbol{CA}^T_{\preceq}$. Furthermore, $\beta^j_1(t_1)$ and $\beta^j_2(t_2)$ must specify the same subsequences; otherwise, the conjunction $\beta^j(t_1, t_2)$ defines a partial order for t_1 and t_2 satisfying this conjunction, and it is impossible to define a total linear order for t_1 and t_2 satisfying $\beta^j_1(t_1)$ and $\beta^j_2(t_2)$. Therefore, each conjunct $\beta^j(t_1, t_2)$ defines an order for pairs of tuples from the same subset S, and the only possible orders in S are identical or reverse to the original order.

In conclusion, the linear order decided by ψ' can be expressed by a sequence of ordered operators in $\boldsymbol{CA}^T_{\preceq}$,

$$\bigcup_i \beta(E_{i1} \cap ... \cap E_{ii_k}),$$

where β is either the order identity operation μ or the order reverse operation ν, and expressions $E_{i1}, ...E_{ii_k}$ can be one of the following forms:

1. $\mathtt{first}_k\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
2. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \mathtt{first}_k\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
3. $\mathtt{last}_k\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
4. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \mathtt{last}_k\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$.

□

We have shown that the ω queries in $\boldsymbol{CQ}^T_{\preceq}$ are completely expressible in the ordered conjunctive algebra $\boldsymbol{CA}^T_{\preceq}$. We are now ready for the main result of this chapter, the completeness theorem of ordered conjunctive queries $\boldsymbol{CQ}^T_{\preceq}$.

Theorem 4.11 *Any $\boldsymbol{CQ}^T_{\preceq}$ query is equivalent to an ordered query expressed by a finite sequence of ordered operations from $\boldsymbol{CA}^T_{\preceq}$.*

Proof:
Let $\langle \phi, \psi \rangle$ be an ordered conjunctive query in $\boldsymbol{CQ}^T_{\preceq}$ on ordered relations $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$.

Case 1: $n = 1$.

In this case, the ordered query $\langle \phi, \psi \rangle$ has only one input $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, it hence has the form

$$\langle \phi, \psi \rangle(\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle) = \langle \{(t, \bar{x}) \mid \phi(t, \bar{x})\},\ \{(t_1, t_2) \mid \psi(t_1, t_2)\} \rangle,$$

where $\phi(t, \bar{x})$ is a conjunctive query and has the normal form

$$\pi_A(\sigma_{\mathrm{p}}(\mathrm{R})),$$

and

$$\psi = \exists \bar{x}_1.\phi(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\phi(t_2, \bar{x}_2) \wedge \psi'(t_1, \bar{x}_1, t_2, \bar{x}_2).$$

By definition of $\boldsymbol{CQ}^T_{\preceq}$, ψ' is a first-order formula defined by the BNF rule

$$\psi' \ = \ \mathrm{R}(t, \bar{x}) \mid t \preceq_R t' \mid \neg\psi' \mid \psi' \wedge \psi' \mid \exists x.\psi' \mid \exists t.\psi'.$$

Since no data predicates exist in ψ', $\boldsymbol{x}$-variables appear only in form of $\exists x.\mathrm{R}(t, x)$ in scope of each quantifiers of $\boldsymbol{t}$-variables. To simplify the form of ψ', we omit $\boldsymbol{x}$-variables in ψ' by replacing $\exists x.\mathrm{R}(t, x)$ with $\mathrm{R}(t)$. Therefore, we consider ψ' only as a first-order formula defined by the BNF rule

$$\psi' \ = \ \mathrm{R}(t) \mid t \preceq_R t' \mid \neg\psi' \mid \psi' \wedge \psi' \mid \exists t.\psi'.$$

By Lemma 4.10, there exists an algebraic expression in $\boldsymbol{CA}^T_{\preceq}$ to express the ordered query $\langle \omega, \psi' \rangle$:

$$\bigcup_i i_k \beta(E_{i1} \cap \ldots \cap E_{ii_k})$$

where β is either the order identity operation μ or the order reverse operation ν, and each of $E_{i1}, \ldots E_{ii_k}$ is in one of the following forms:

1. $\texttt{first}_k\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
2. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \texttt{first}_k\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
3. $\texttt{last}_k\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
4. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \texttt{last}_k\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$.

By Proposition 4.6, order-preserving selections and projections can propagate through order identity, reverse, and concatenation operators. To express the selections and projections on the data relation R in the normal form of ϕ, order-preserving selections and order-preserving projections are applied respectively to the above expression without changing the order.

In conclusion, the ordered $\boldsymbol{CQ}^T_{\preceq}$ query $\langle \phi, \psi \rangle$ is equivalent to the algebraic expression in $\boldsymbol{CA}^T_{\preceq}$,

$$\pi_A(\sigma_{\mathrm{p}} \bigcup_k \beta(E_{i1} \cap ... \cap E_{ik}(\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle))),$$

where β is either the order identity operation μ or the order reverse operation ν in $\boldsymbol{CA}^T_{\preceq}$, and algebraic expressions $E_{i1}, ..., E_{ii_k}$ are defined as above.

Case 2: $n > 1$.

Let $\langle \varphi, \psi \rangle$ be a $\boldsymbol{CQ}^T_{\preceq}$ query on ordered relations $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$, where $n > 1$. The data query φ is a conjunctive query, and hence has the normal form

$$\exists x_1, \ldots, x_m.(\mathrm{R}_1(u_1) \wedge \ \cdots \wedge \mathrm{R}_n(u_n)),$$

where u_i's are tuples containing both variables $t, x_1, \ldots, x_n$ and constants for data attributes. The order query ψ has the form

$$\begin{aligned} \{(t_1, t_2) \mid \quad & \exists t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots \bar{x}_{2n}. \\ & t_1 = (t_{11}, \ldots, t_{1n}) \wedge t_2 = (t_{21}, \ldots, t_{2n}) \\ & \wedge\ \varphi(t_{11}, \ldots, t_{1n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}) \wedge \ \varphi(t_{21}, \ldots, t_{2n}, \bar{x}_{21}, \ldots, \bar{x}_{2n}) \\ & \wedge\ \psi'(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n})\}. \end{aligned}$$

By definition of $\boldsymbol{CQ}^T_{\preceq}$, ψ' is a first-order formula defined by the BNF rule

$$\psi' \ = \ \mathrm{R}_i(t) \mid t \preceq_{\mathrm{R}_i} t' \mid \neg\psi' \mid \psi' \wedge \psi' \mid \exists t.\psi',$$

and ψ' defines a linear order on the Cartesian product of $\mathrm{R}_1 \times \cdots \times \mathrm{R}_n$.

By the BNF rule, ψ' is a boolean combination of existentially quantified formulas with only one $\boldsymbol{t}$-variable free, which could be $t_{11}, \ldots, t_{1n}, t_{21}, \ldots,$ or t_{2n}. Thus, we

can rearrange the top level boolean connectives and rewrite ψ' in a disjunction of conjunctions in the form

$$\begin{array}{l} \bigvee_i \alpha^i(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}) \\ \vee \ \bigvee_j \beta^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}), \end{array}$$

where each $\alpha^i(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})$ is of form

$$(\alpha^i_{11}(t_{11}) \wedge \ldots \wedge \alpha^i_{1n}(t_{1n}) \wedge \alpha^i_{21}(t_{21}) \wedge \ldots \wedge \alpha^i_{2n}(t_{2n})),$$

and $\alpha^i_{1k}(t_{1k})$ and $\alpha^i_{2k}(t_{2k})$ are existentially quantified formulas with only t_{1k} free and existentially quantified formulas with only t_{2k} free, respectively. Each formula $\beta^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})$ is in the form

$$(\beta^j_1(t_{11}, t_{21}) \wedge \ldots \wedge \beta^j_n(t_{1n}, t_{2n})),$$

where at least one of $\beta^j_k(t_{1k}, t_{2k})$ is an existentially quantified formula with both t_{1k} and t_{2k} free, for $k = 1, \ldots, n$.

We now consider the first part of ψ', which is $\bigvee_i \alpha^i$. The conjuncts $\alpha^i_{1k}(t_{1k})$ and $\alpha^i_{2k}(t_{2k})$ are existentially quantified formulas with only t_{1k} free and existentially quantified formulas with only t_{2k} free respectively, for $k = 1, \ldots, n$. By the proof of Lemma 4.10, they can be expressed by an ordered algebraic expression in $\boldsymbol{CA}^T_{\preceq}$,

$$\bigcup_i \beta(E_{i1} \cap \ldots \cap E_{ii_k}),$$

where β is either the order identity operation μ or the order reverse operation ν, and expressions $E_{i1}, \ldots E_{ii_k}$ could be one of the following forms

1. $\texttt{first}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
2. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \texttt{first}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
3. $\texttt{last}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
4. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \texttt{last}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$.

Let $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$ be the two combinations of minimal subsequences from $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$, specified by

$$\alpha^i = (\alpha^i_{11}(t_{11}) \wedge \ldots \wedge \alpha^i_{1n}(t_{1n}) \wedge \alpha^i_{21}(t_{21}) \wedge \ldots \wedge \alpha^i_{2n}(t_{2n})).$$

Let $t_1 = (t_{11}, \ldots, t_{1n})$ and $t_2 = (t_{21}, \ldots, t_{2n})$ be a pair of tuples from $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$, respectively. Since $((t_{11}, \ldots, t_{1n}), (t_{21}, \ldots, t_{2n}))$ satisfy α^i, they

will be in the resulting order relation $\preceq_{R'}$, *i.e.*, $t_1 \preceq_{R'} t_2$. It follows $t_1 \preceq_{R'} t_2$ for any pair of tuples t_1 and t_2 from $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$, and hence $(S_{11}, \ldots, S_{1n}) \preceq_{R'} (S_{21}, \ldots, S_{2n})$. In other words, any conjunction α^i in the first part $\bigvee_i \alpha^i$ decides the order between a pair of combinations of minimal subsequences $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$ from $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$.

Using the same technique in the proof of Lemma 4.10, we can prove that $\bigvee_i \alpha^i$ defines a linear order among all combinations of minimal subsequences $(S_{1i_1}, \ldots, S_{ni_n})$ from $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$. Suppose that $\bigvee_i \alpha^i$ does not define a linear order among all combinations of minimal subsequence $(S_{1i_1}, \ldots, S_{ni_n})$ from $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$. There are two cases:

Case 1: Assume that there exist two combinations of subsequence, say $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$, such that both $(S_{11}, \ldots, S_{1n}) \preceq_{R'} (S_{21}, \ldots, S_{2n})$ and $(S_{21}, \ldots, S_{2n}) \preceq_{R'} (S_{11}, \ldots, S_{1n})$ hold. For any $t_1 \in (S_{11}, \ldots, S_{1n})$ and $t_2 \in (S_{21}, \ldots, S_{2n})$, by this assumption, $(t_1, t_2) \in \preceq_{R'}$ and $(t_2, t_1) \in \preceq_{R'}$ hold at the same time. This is a contradiction to the fact that the output order $\preceq_{R'}$ is a linear order. Hence, the assumption in Case 1 does not hold.

Case 2: Assume that there exist two combinations of subsequence, say $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$, such that neither $(S_{11}, \ldots, S_{1n}) \preceq_{R'} (S_{21}, \ldots, S_{2n})$ nor $(S_{21}, \ldots, S_{2n}) \preceq_{R'} (S_{11}, \ldots, S_{1n})$ holds; their order in output is not specified by $\bigvee_i \alpha^i$. By this assumption, for any pair of tuples $t_1 \in (S_{11}, \ldots, S_{1n})$ and $t_2 \in (S_{21}, \ldots, S_{2n})$, their order in the output is not defined by $\bigvee_i \alpha^i$. Because the output order is a linear order on all combinations of tuples from n ordered relations, the order of t_1 and t_2 must be decided by ψ'. Suppose that ψ' decides $t_1 \preceq_{R'} t_2$. Since $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$ are combinations of minimal subsequence, all pairs of tuples from $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$ must have the same order if they have any order. Then, for any pair of tuples $t_1 \in (S_{11}, \ldots, S_{1n})$ and $t_2 \in (S_{21}, \ldots, S_{2n})$, we have $t_1 \preceq_{R'} t_2$. It follows $(S_{11}, \ldots, S_{1n}) \preceq_{R'} (S_{21}, \ldots, S_{2n})$, which is a contradiction to the assumption.

We have proven that $\bigvee_i \alpha^i$ defines a linear order among all combinations of minimal subsequences $(S_{1i_1}, \ldots, S_{ni_n})$, which are specified by all conjuncts α^i. Assume that each subsequence S_{ki_k} $(k = 1, ..., n)$ keeps the original order from the input, and can be expressed by ordered operation

$$E_{ki_k}\langle R_k, \preceq_{R_k}\rangle,$$

where E_{ki_k} is of the form

$$E_{ki_k} = E^1_{ki_k} \cap ... \cap E^j_{ki_k},$$

and $E^1_{ki_k}, ..., E^j_{ki_k}$ could be one of the following forms:

1. $\texttt{first}_i\ (\langle \mathrm{R}, \preceq_\mathrm{R}\rangle)$;
2. $\langle \mathrm{R}, \preceq_\mathrm{R}\rangle - \texttt{first}_i\ (\langle \mathrm{R}, \preceq_\mathrm{R}\rangle)$;
3. $\texttt{last}_i\ (\langle \mathrm{R}, \preceq_\mathrm{R}\rangle)$;
4. $\langle \mathrm{R}, \preceq_\mathrm{R}\rangle - \texttt{last}_i\ (\langle \mathrm{R}, \preceq_\mathrm{R}\rangle)$.

Next, we consider the second part of ψ,

$$\bigvee_j \beta^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}).$$

To compose a linear order on the Cartesian product of data relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$, the order between each pair of tuples from the same combination of subsequences has to be decided by the order formula ψ'. This order inside a combination of subsequences is decided by the second part of ψ,

$$\bigvee_j \beta^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}).$$

Let $(S_{i_11}, \ldots, S_{i_nn})$ be a combination of minimal subsequences specified by $\alpha^j_1(t_{11}), ..., \alpha^j_n(t_{1n})$ on $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$, respectively. For any pair of tuples $(t_{11}, \ldots, t_{1n})$ and $(t_{21}, \ldots, t_{2n})$ from the same combination of minimal subsequences $(S_{i_11}, \ldots, S_{i_nn})$, we always can find a j such that β^j decides the order of this pair in the result because the output order is a total order on all combinations of tuples from input relations.

Thus, the conjunction β^j is equivalent to

$$\begin{aligned}&(\alpha^j_1(t_{11}) \wedge \ldots \wedge \alpha^j_n(t_{1n}) \wedge \alpha^j_1(t_{21}) \wedge \ldots \wedge \alpha^j_n(t_{2n}))\\ &\wedge\ \psi^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}),\end{aligned}$$

where the conjunct$\alpha^j_1(t_{1i})$ defines a minimal subsequence S_{ji} over $\langle \mathrm{R}_i, \preceq_{\mathrm{R}_i}\rangle$; and $\psi^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})$ decides the real order in the combination of minimal subsequences and is defined by the BNF rule

$$\psi^j = \ t \preceq_{\mathrm{R}_i} t' \mid \neg\psi^j \mid \psi^j \wedge\ \psi^j.$$

The conjunct ψ^j has this form because $t_{1k}, \ldots, t_{2k}, t_{2k}, \ldots, t_{1k}$ are only $\boldsymbol{t}$-variables in minimal subsequence S_{ji}, and appear only in forms $t_{1k} \preceq_{R_k} t_{2k}$ and $t_{2k} \preceq_{R_k} t_{1k}$, for $k = 1, \ldots, n$.

Let $S_{i_jk_j}$ $(j = 1, ..., n)$ be the i_j-th subsequence in the input ordered relation $\langle R_{k_j}, \preceq_{R_{k_j}} \rangle$. Since ψ^j defines the lexicographic order inside the combination of minimal subsequence $(S_{i_1k_1}, \ldots, S_{i_nk_n})$, it is equivalent to a normal form

$$\epsilon(t_{1k_1} \preceq_{\mathrm{R}} t_{2k_1}) \vee \ (t_{1k_1} = t_{2k_1} \wedge \ \epsilon(t_{1k_2} \preceq_{\mathrm{R}} t_{2k_2})) \vee \ \ldots$$
$$\vee \ (t_{1k_1} = t_{2k_1} \wedge \ \ldots \ \wedge \ t_{1k_{n-1}} = t_{2k_{n-1}} \wedge \ \epsilon(t_{1k_n} \preceq_{\mathrm{R}} t_{2k_n})),$$

where $\epsilon(t_{1k_n} \preceq_{\mathrm{R}} t_{2k_n})$ is either $t_{1k_n} \preceq_{\mathrm{R}} t_{2k_n}$ or $t_{2k_n} \preceq_{\mathrm{R}} t_{1k_n}$. The combination $(k_1, \ldots, k_n)$ is a permutation of $(1, \ldots, n)$, which decides the major to minor order in which $\langle \mathrm{R}_{k_1}, \preceq_{\mathrm{R}_{k_1}} \rangle, \ldots, \langle \mathrm{R}_{k_n}, \preceq_{\mathrm{R}_{k_n}} \rangle$ serve in the output order.

Intuitively, ψ^j defines a lexicographical order on $(S_{i_1k_1}, \ldots, S_{i_nk_n})$, and each minimal subsequence has an order identical or reverse to the original input order. Therefore, each combination of minimal subsequence $(S_{i_1k_1}, \ldots, S_{i_nk_n})$ can be expressed by an algebraic expression in the ordered conjunctive algebra $\boldsymbol{CA}^T_{\preceq}$ as follows:

$$\beta(E_{i_1,j_1}(\langle \mathrm{R}_{k_1}, \preceq_{\mathrm{R}_{k_1}} \rangle)) \ltimes \ \ldots \ltimes \ \beta(E_{i_n,j_n}(\langle \mathrm{R}_{k_n}, \preceq_{\mathrm{R}_{k_n}} \rangle)),$$

where the sequence $(k_1, \ldots, k_n)$ is a permutation of $(1, \ldots, n)$, which indicates the order in which the left-nested-loop product operations are applied to the subsequences of $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$.

In conclusion, the ordered query $\langle \phi', \psi' \rangle$ (where ϕ' can be expressed by $\mathrm{R}_1 \times \cdots \times \mathrm{R}_n$) can be expressed by a $\boldsymbol{CA}^T_{\preceq}$ expression,

$$\bigcup (\beta(E_{i_1,j_1}(\langle \mathrm{R}_{k_1}, \preceq_{\mathrm{R}_{k_1}} \rangle)) \ltimes \ \ldots \ \ltimes \ \beta(E_{i_n,j_n}(\langle \mathrm{R}_{k_n}, \preceq_{\mathrm{R}_{k_n}} \rangle))).$$

We now consider the data query ϕ. If there are selections and projections in the normal form of ϕ, order-preserving selections and order-preserving projections are applied to the above algebraic expression correspondingly because order-preserving selections and projections can propagate through left nested-loop products.

Therefore, any $\boldsymbol{CQ}^T_{\preceq}$ query $\langle \phi, \psi \rangle$ on ordered relations $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$ is equivalent to an ordered query expressed in $\boldsymbol{CA}^T_{\preceq}$,

$$\pi_A(\sigma_p(\bigcup (\beta(E_{i_1,j_1}(\langle \mathrm{R}_{k_1}, \preceq_{\mathrm{R}_{k_1}} \rangle)) \ltimes \ \ldots \ \ltimes \ \beta(E_{i_n,j_n}(\langle \mathrm{R}_{k_n}, \preceq_{\mathrm{R}_{k_n}} \rangle))))),$$

where β is either μ or ν, and $\bigcup$ is an order concatenation operator. The combination $(k_1, \ldots, k_n)$ is a permutation of $(1, \ldots, n)$, different in different combinations of subsequence $(S_{i_1k_1}, \ldots, S_{i_nk_n})$. The expression E_{i_k,j_k} is of the form

$$E_{i_k,j_k} = E_{i_k} \cap E_{j_k},$$

for $k = 1, ..., n$, and E_{i_k} and E_{j_k} could be one of the following forms:

1. $\texttt{first}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
2. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \texttt{first}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
3. $\texttt{last}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$;
4. $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle - \texttt{last}_i\ (\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)$.

□

The completeness proof for $\boldsymbol{CA}^T_{\preceq}$ is similar to that for $\boldsymbol{CA}^X_{\preceq}$. The only difference is that the output order of an ω query in $\boldsymbol{CQ}^T_{\preceq}$ is a linear order of minimal subsequences; each minimal subsequence can be expressed with top operations, order reverse and order reduction operations. A conjunction of two existentially quantified formulas on $\boldsymbol{t}$-variables can be expressed with an order intersection on their ordered algebraic expressions. The following examples demonstrate how to express $\boldsymbol{CQ}^T_{\preceq}$ queries with order algebraic expressions $\boldsymbol{CA}^T_{\preceq}$.

Example 4.12 Consider the queries in Example 4.2. These $\boldsymbol{CQ}^T_{\preceq}$ queries can be expressed with ordered operators in $\boldsymbol{CA}^T_{\preceq}$.

(1) For the first query, we first formulate "the first employee" as:

$$E_1 = \tau\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle.$$

Next, "the rest of employees except the first two" is expressed as:

$$E_2 = \langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle - E_1.$$

Finally, the first query is represented with an ordered algebraic expression in $\boldsymbol{CA}^T_{\preceq}$ as follows:

$$E_2 \cup E_1.$$

(2) The second query can be formulated with an ordered algebraic expression in $\boldsymbol{CA}^T_{\preceq}$ as follows:

$$\sigma_{\texttt{Department}=\text{“}Sales\text{”}}(\nu(\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle)).$$

This chapter focuses on the second class of ordered conjunctive queries $\boldsymbol{CQ}^T_{\preceq}$, and proves that the ordered algebra $\boldsymbol{CA}^T_{\preceq}$ is complete in the sense that it can express the set of $\boldsymbol{CQ}^T_{\preceq}$ under certain constraints. The following chapter will explore the more general class of ordered conjunctive queries, $\boldsymbol{CQ}_{\preceq}$.

Chapter 5

Ordered Conjunctive Queries $CQ_{\preceq}$

In the previous two chapters two classes of ordered conjunctive queries, $\boldsymbol{CQ}^{X}_{\preceq}$ and $\boldsymbol{CQ}^{T}_{\preceq}$, are explored in terms of their completeness problems. In this chapter, we investigate the general class of ordered conjunctive queries, $\boldsymbol{CQ}_{\preceq}$, whose output orders are decided by both $\boldsymbol{t}$-variables and $\boldsymbol{x}$-variables.

The remainder of this chapter will proceed as follows. In Section 5.1, general ordered conjunctive queries $\boldsymbol{CQ}_{\preceq}$ are formally defined in the first-order logic $\boldsymbol{FO}_{\preceq}$, and then an ordered algebra, $\boldsymbol{CA}_{\preceq}$, is proposed for $\boldsymbol{CQ}_{\preceq}$ in Section 5.2. Transformation rules are introduced and proven for primitive and derived ordered operations in $\boldsymbol{CA}_{\preceq}$ in Section 5.3. Finally, the Completeness Theorem of ordered conjunctive queries $\boldsymbol{CQ}_{\preceq}$ is proven in Section 5.4.

5.1 Ordered Conjunctive Queries $CQ_{\preceq}$

The ordered conjunctive query $\boldsymbol{CQ}_{\preceq}$ is a generalization of the ordered conjunctive queries $\boldsymbol{CQ}^{X}_{\preceq}$ and $\boldsymbol{CQ}^{T}_{\preceq}$. The order specification of a $\boldsymbol{CQ}_{\preceq}$ query combines both data predicates and quantifiers on $\boldsymbol{t}$-variables; however, a $\boldsymbol{CQ}_{\preceq}$ query must satisfy a restriction, which we call the Separation Property.

Definition 5.1 (Separation Property) *Let $\langle \varphi, \psi \rangle$ be an $\boldsymbol{FO}_{\preceq}$ ordered query on an ordered relation $\langle R, \preceq_R \rangle$. The order specification $\psi(t_1, t_2)$ is* separable *in $\boldsymbol{FO}_{\preceq}$,*

if and only if there exists an equivalent formula in $\boldsymbol{FO}_{\preceq}$*, which is a boolean combination of existentially quantified formulas with only* t_1 *free, existentially quantified formulas with only* t_2 *free, and atomic formulas in the form* $t_1 \preceq t_2$ *or* $t_2 \preceq t_1$*.*

Intuitively, an $\boldsymbol{FO}_{\preceq}$ order query $\psi(t_1, t_2)$, satisfying this Separation Property, can be rewritten into a normal form

$$\bigvee \psi_1(t_1) \wedge \psi_2(t_2) \wedge \psi_3(t_1, t_2),$$

where $\psi_1(t_1)$ and $\psi_2(t_2)$ are existentially quantified formulas with only t_1 free and existentially quantified formulas with only t_2 free respectively, and where $\psi_3(t_1, t_2)$ is an atomic formula in the form $t_1 \preceq t_2$ or $t_2 \preceq t_1$. Note that there is no existentially quantified formula with both t_1 and t_2 free in this normal form, so free $\boldsymbol{t}$-variables, t_1 and t_2, always appear in existentially quantified formulas separately.

We need the Separation Property in the definition of general ordered conjunctive queries $\boldsymbol{CQ}_{\preceq}$ because restricting $\boldsymbol{CQ}_{\preceq}$ to those with the Separation Property ensures that the proposed algebra $\boldsymbol{CA}_{\preceq}$ is complete for $\boldsymbol{CQ}_{\preceq}$. We now give a formal definition to general ordered conjunctive queries, $\boldsymbol{CQ}_{\preceq}$, which satisfies the Separation Property.

Definition 5.2 (Ordered Conjunctive Queries $\boldsymbol{CQ}_{\preceq}$)
A $\boldsymbol{CQ}_{\preceq}$ *ordered query* $\langle \varphi, \psi \rangle$ *on ordered relations* $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$ *is constructed as follows:*

$$\begin{array}{rcl} \varphi & ::= & R(t_i, x_1, \ldots, x_n) \\ & | & t = (t_1, t_2) \\ & | & x_i = x_j \\ & | & \varphi \wedge \varphi \\ & | & \exists x_i.\varphi \\ & | & \exists t_i.\varphi. \end{array}$$

And, the order query ψ *is a first-order formula in the form of:*

$$\begin{array}{rcl} \psi & ::= & R(t_i, x_1, \ldots, x_n) \\ & | & t = (t_1, t_2) \\ & | & x_i = x_j \\ & | & \psi \wedge \psi \\ & | & \exists x_i.\psi \\ & | & \exists t_i.\psi \\ & | & \psi \wedge \psi', \end{array}$$

where the order specification ψ' *is defined by the BNF rule:*

$$
\begin{array}{rcl}
\psi' & ::= & R(t_i, x_1, \ldots, x_n) \\
& | & t_i \preceq_R t_j \\
& | & x_i = x_j \\
& | & \psi' \wedge \psi' \\
& | & \neg\psi' \\
& | & \exists x_i.\psi \\
& | & \exists t_i.\psi,
\end{array}
$$

where ψ has exactly two $\boldsymbol{t}$-variables, t_1 and t_2, as the only free variables and ψ satisfies the Separation Property; the order specification ψ' has all $\boldsymbol{x}$-variables bounded.

Again, there is a two-level structure in the definition of the order query ψ. In a valid ordered conjunctive query in $\boldsymbol{CQ}_{\preceq}$, the first level of ψ ensures that ψ defines an order for any pair of tuples that satisfies the data query φ. The second level of ψ is the subformula ψ', which defines real ordering among the resulting tuples.

In this definition, the construction rule of the order query ψ' contains both data predicates on $\boldsymbol{x}$-variables and quantifiers on $\boldsymbol{t}$-variables, which combines the previous two languages $\boldsymbol{CQ}^X_{\preceq}$ and $\boldsymbol{CQ}^T_{\preceq}$. Furthermore, the order query ψ' satisfies the Separation Property (see Definition 5.1), which leads to the possibility that there is a complete ordered algebra for the generalized $\boldsymbol{CQ}_{\preceq}$.

The language $\boldsymbol{CQ}_{\preceq}$ expresses a class of ordered conjunctive queries whose output ordering is decided by both $\boldsymbol{x}$-variables and $\boldsymbol{t}$-variables, as illustrated by the example below.

Example 5.3 Consider the ordered relation $\langle \texttt{EMP}, \preceq_{\texttt{EMP}} \rangle$ in Example 2.5. Here are some examples of $\boldsymbol{CQ}_{\preceq}$ queries:

(1) Return all employees in EMP in an order such that tuples before the first employee, whose salary is \$60000, move to the end of the relation. This query can be formulated in $\boldsymbol{CQ}_{\preceq}$:

$$
\begin{array}{ll}
\langle\{(t, x_1, x_2, x_3, x_4)| & \texttt{EMP}\ (t, x_1, x_2, x_3, x_4)\}, \\
\{(t_1, t_2)| & \exists x_{11}, x_{12}, x_{13}, x_{14}.\ \texttt{EMP}\ (t_1, x_{11}, x_{12}, x_{13}, x_{14}) \\
& \wedge \exists x_{21}, x_{22}, x_{23}, x_{24}.\ \texttt{EMP}\ (t_2, x_{21}, x_{22}, x_{23}, x_{24}) \\
& \wedge\ (((\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}) \\
& \wedge x'_{14} = 60000 \wedge t'_1 \preceq_{\texttt{EMP}} t_1) \\
& \wedge\ (\neg\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}) \\
& \wedge x'_{14} = 60000 \wedge t'_1 \preceq_{\texttt{EMP}} t_2)) \\
& \vee
\end{array}
$$

$$
\begin{aligned}
&((\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14})\\
&\wedge x'_{14} = 60000 \wedge t'_1 \preceq_{\texttt{EMP}} t_1)\\
&\wedge\ (\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14})\\
&\wedge x'_{14} = 60000 \wedge t'_1 \preceq_{\texttt{EMP}} t_2)\\
&\wedge\ t_1 \preceq_{\texttt{EMP}} t_2)\\
&\vee\\
&((\neg\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14})\\
&\wedge x'_{14} = 60000 \wedge t'_1 \preceq_{\texttt{EMP}} t_1)\\
&\wedge\ (\neg\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14})\\
&\wedge x'_{14} = 60000 \wedge t'_1 \preceq_{\texttt{EMP}} t_2)\\
&\wedge\ t_1 \preceq_{\texttt{EMP}} t_2)\}\rangle.
\end{aligned}
$$

(2) Output all employees in an order which reverses the employees between the first and second employees in the department Sales, while keeping the rest in the original order. This query can be formulated in $\boldsymbol{CQ}_{\preceq}$:

$$
\begin{aligned}
\langle\{(t, x_1, x_2, x_3, x_4)|\ &\texttt{EMP}\ (t, x_1, x_2, x_3, x_4)\},\\
\{(t_1, t_2)|\ &\exists x_{11}, x_{12}, x_{13}, x_{14}.\ \texttt{EMP}\ (t_1, x_{11}, x_{12}, x_{13}, x_{14})\\
&\wedge\exists x_{21}, x_{22}, x_{23}, x_{24}.\ \texttt{EMP}\ (t_2, x_{21}, x_{22}, x_{23}, x_{24})\\
&\wedge\ (((\neg\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14})\\
&\wedge x'_{13} = \text{“}Sales\text{”} \wedge t'_1 \preceq_{\texttt{EMP}} t_1)\\
&\wedge\ (\neg\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14})\\
&\wedge x'_{13} = \text{“}Sales\text{”} \wedge t'_1 \preceq_{\texttt{EMP}} t_2)\\
&\wedge t_1 \preceq_{\texttt{EMP}} t_2)\\
&\vee\\
&((\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14})\\
&\wedge x'_{13} = \text{“}Sales\text{”} \wedge t'_1 \preceq_{\texttt{EMP}} t_1\\
&\wedge\ \neg\exists t'_2, x'_{21}, x'_{22}, x'_{23}, x'_{24}, t'_3, x'_{31}, x'_{32}, x'_{33}, x'_{34}.\\
&\texttt{EMP}\ (t'_2, x'_{21}, x'_{22}, x'_{23}, x'_{24}) \wedge x'_{23} = \text{“}Sales\text{”}\\
&\wedge\ \ \texttt{EMP}\ (t'_3, x'_{31}, x'_{32}, x'_{33}, x'_{34}) \wedge x'_{33} = \text{“}Sales\text{”}\\
&\wedge\ t'_3 \preceq_{\texttt{EMP}} t'_2 \wedge t'_2 \preceq_{\texttt{EMP}} t_1)\\
&\wedge\ (\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}.\ \texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14})\\
&\wedge x'_{13} = \text{“}Sales\text{”} \wedge t'_1 \preceq_{\texttt{EMP}} t_2
\end{aligned}
$$

$\wedge$ $(\neg\exists t'_2, x'_{21}, x'_{22}, x'_{23}, x'_{24}, t'_3, x'_{31}, x'_{32}, x'_{33}, x'_{34}.$
$\texttt{EMP}\ (t'_2, x'_{21}, x'_{22}, x'_{23}, x'_{24}) \wedge x'_{23} = \text{“}Sales\text{”}$
$\wedge\ \ \texttt{EMP}\ (t'_3, x'_{31}, x'_{32}, x'_{33}, x'_{34}) \wedge x'_{33} = \text{“}Sales\text{”}$
$\wedge\ t'_3 \preceq_{\texttt{EMP}} t'_2 \wedge t'_2 \preceq_{\texttt{EMP}} t_2))$
$\wedge \neg t_1 \preceq_{\texttt{EMP}} t_2)$
$\vee$
$((\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}, t'_2, x'_{21}, x'_{22}, x'_{23}, x'_{24}.$
$\texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}) \wedge x'_{13} = \text{“}Sales\text{”}$
$\wedge\ \texttt{EMP}\ (t'_2, x'_{21}, x'_{22}, x'_{23}, x'_{24}) \wedge x'_{23} = \text{“}Sales\text{”}$
$\wedge\ t'_1 \preceq_{\texttt{EMP}} t'_2 \wedge t'_2 \preceq_{\texttt{EMP}} t_1)$
$\wedge\ (\exists t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}, t'_2, x'_{21}, x'_{22}, x'_{23}, x'_{24}.$
$\texttt{EMP}\ (t'_1, x'_{11}, x'_{12}, x'_{13}, x'_{14}) \wedge x'_{13} = \text{“}Sales\text{”}$
$\wedge\ \texttt{EMP}\ (t'_2, x'_{21}, x'_{22}, x'_{23}, x'_{24}) \wedge x'_{23} = \text{“}Sales\text{”}$
$\wedge\ t'_1 \preceq_{\texttt{EMP}} t'_2 \wedge t'_2 \preceq_{\texttt{EMP}} t_2)$
$\wedge\ t_1 \preceq_{\texttt{EMP}} t_2))\}\rangle.$

5.2 An Ordered Conjunctive Algebra $CA_{\preceq}$

In the previous section general ordered conjunctive queries $\boldsymbol{CQ}_{\preceq}$ was formally defined. In this section an ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$ is proposed for general ordered conjunctive queries $\boldsymbol{CQ}_{\preceq}$, and then transformation rules are studied for primitive and derived ordered operations in $\boldsymbol{CA}_{\preceq}$.

5.2.1 Overview of Ordered Algebra $CA_{\preceq}$

In the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$, a set of ordered operators are proposed to express all $\boldsymbol{CQ}_{\preceq}$ expressible ordered conjunctive queries. Certain ordered operators are preserved from $\boldsymbol{CA}^X_{\preceq}$ and $\boldsymbol{CA}^T_{\preceq}$ and others are exclusively developed for $\boldsymbol{CA}_{\preceq}$.

Next, formal definitions are provided for ordered operations in $\boldsymbol{CA}_{\preceq}$. Both data and order specifications of an ordered operator are defined in the two-sorted first-order calculus $\boldsymbol{FO}_{\preceq}$. Essentially, each ordered operator in this algebra is a valid $\boldsymbol{CQ}_{\preceq}$ ordered relational query. The ordered algebra $\boldsymbol{CA}_{\preceq}$ consists of a set of ordered operators:

(i) Order-preserving selection operator $\sigma : \mathbf{R}_{\preceq} \times \mathbf{P} \rightarrow \mathbf{R}_{\preceq}$;

(ii) Order-preserving projection operator $\pi : \mathbf{R}_{\preceq} \times \mathrm{A} \rightarrow \mathbf{R}_{\preceq}$;

(iii) Left nested-loop product $\ltimes : \mathbf{R}_{\preceq} \times \mathbf{R}_{\preceq} \rightarrow \mathbf{R}_{\preceq}$;

(iv) Ordered concatenation operator $\cup : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$;

(v) Order reduction operator $- : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$;

(vi) Order intersection operator $\cap : \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$;

(vii) Order identity operator $\mu : \mathbf{R} \rightarrow \mathbf{R}$;

($viii$) Order reverse operator $\nu : \mathbf{R} \rightarrow \mathbf{R}$;

(ix) Top operator $\tau : \mathbf{R} \rightarrow \mathbf{R}$;

(x) Until operator $\mathtt{until}_{\mathrm{p}}$: $\mathbf{R} \rightarrow \mathbf{R}$.

The ordered operators in $\boldsymbol{CA}_{\preceq}$ are defined in analogy to those in $\boldsymbol{CA}^{T}_{\preceq}$, with the exception of the *until* operator. In the following subsections, the until operator and derived operators are formally defined.

5.2.2 Until Operation

The until operator $\mathtt{until}_{\mathrm{p}}$: $\mathbf{R} \rightarrow \mathbf{R}$ is an operator exclusive to the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$. The subscript p is a predicate on $\boldsymbol{x}$-variables. The argument and resulting ordered relations share the same schema. The formal definition of the until operator is given as follows:

$$
\begin{aligned}
\mathtt{until}_{\mathrm{p}}\left(\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle\right) = \; \langle \{(t, \bar{x}) | \quad & \mathrm{R}(t, \bar{x}) \wedge \exists t_1, \bar{x}_1 . \mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\
& \wedge \; t \preceq_{\mathrm{R}} t_1 \wedge \neg t = t_1 \\
& \wedge \neg \exists t_2, \bar{x}_2 . \mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{p}(t_2, \bar{x}_2) \\
& \wedge \; t_2 \preceq_{\mathrm{R}} t_1 \wedge \neg t_2 = t_1 \}, \\
\{(t_1, t_2) | \quad & \exists \bar{x}_1, \bar{x}_2 . \mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge \; \exists t'_1, \bar{x}' . \mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \wedge \neg \exists t'_2, \bar{x}'_2 . \mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
& \wedge \; t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
& \wedge \; t_1 \preceq_{\mathrm{R}} t'_1 \wedge t_2 \preceq_{\mathrm{R}} t'_1 \wedge \; \neg t_1 = t'_1 \wedge \neg t_2 = t'_1 \\
& \wedge \; t_1 \preceq_{R} t_2 \} \rangle .
\end{aligned}
$$

In this definition, the operator takes an ordered relation as its argument and returns, as its result, the tuples before the first tuple satisfying the predicate. The

tuples in the result are kept in the original order of the argument. The tuple identifiers in the resulting ordered relation are identical to those in the argument relation. Please note that the until operation is not definable in $\boldsymbol{CQ}_{\preceq}$ because it has negations in its data query.

Lemma 5.4 *An until operation is a valid ordered relational query.*

Proof: Let $\texttt{until}_{\mathrm{p}}\ (\mathbf{R}) = \langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$, where $\mathbf{R} = \langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$. By definition of the until operation, R' and $\preceq_{\mathrm{R}'}$ are inclusively dependant on each other. We now prove that $\preceq_{\mathrm{R}'}$ is a linear order of R'.

(1) Let $t_1, t_2, t_3 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge\ (t_2, t_3) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\quad & (\exists \bar{x}_1, \bar{x}_2 . \mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge\ \exists t'_1, \bar{x}' . \mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \neg \exists t'_2, \bar{x}'_2 . \mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
& \wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
& \wedge\ t_1 \preceq_{\mathrm{R}} t'_1 \wedge t_2 \preceq_{\mathrm{R}} t'_1 \wedge\ \ \neg t_1 = t'_1 \wedge \neg t_2 = t'_1 \\
& \wedge\ t_1 \preceq_R t_2) \\
& \wedge (\exists \bar{x}_2, \bar{x}_3 . \mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{R}(t_3, \bar{x}_3) \\
& \wedge\ \exists t'_1, \bar{x}' . \mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \neg \exists t'_2, \bar{x}'_2 . \mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
& \wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
& \wedge\ t_2 \preceq_{\mathrm{R}} t'_1 \wedge t_3 \preceq_{\mathrm{R}} t'_1 \wedge\ \ \neg t_2 = t'_1 \wedge \neg t_3 = t'_1 \\
& \wedge\ t_2 \preceq_R t_3) \\
\Longrightarrow\quad & (\exists \bar{x}_1, \bar{x}_3 . \mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_3, \bar{x}_3) \\
& \wedge\ \exists t'_1, \bar{x}' . \mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \neg \exists t'_2, \bar{x}'_2 . \mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
& \wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
& \wedge\ t_1 \preceq_{\mathrm{R}} t'_1 \wedge t_3 \preceq_{\mathrm{R}} t'_1 \wedge\ \ \neg t_1 = t'_1 \wedge \neg t_3 = t'_1 \\
& \wedge\ t_1 \preceq_R t_3) \\
\Longrightarrow\quad & (t_1, t_3) \in \preceq_{\mathrm{R}'} .
\end{aligned}
$$

(2) Let $t_1, t_2 \in \mathrm{R}'[\mathrm{T}]$,

$$
\begin{aligned}
& (t_1, t_2) \in \preceq_{\mathrm{R}'} \wedge\ (t_2, t_1) \in \preceq_{\mathrm{R}'} \\
\Longrightarrow\quad & (\exists \bar{x}_1, \bar{x}_2 . \mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge\ \exists t'_1, \bar{x}' . \mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \neg \exists t'_2, \bar{x}'_2 . \mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2)
\end{aligned}
$$

$$
\begin{aligned}
&\wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
&\wedge\ t_1 \preceq_{\mathrm{R}} t'_1 \wedge t_2 \preceq_{\mathrm{R}} t'_1 \wedge\ \neg t_1 = t'_1 \wedge \neg t_2 = t'_1 \\
&\wedge\ t_1 \preceq_R t_2) \\
&\wedge(\exists \bar{x}_2, \bar{x}_1.\mathrm{R}(t_2, \bar{x}_2) \wedge \mathrm{R}(t_1, \bar{x}_1) \\
&\wedge\ \exists t'_1, \bar{x}'.\mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
&\neg\exists t'_2, \bar{x}'_2.\mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
&\wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
&\wedge\ t_2 \preceq_{\mathrm{R}} t'_1 \wedge t_1 \preceq_{\mathrm{R}} t'_1 \wedge\ \neg t_2 = t'_1 \wedge \neg t_1 = t'_1 \\
&\wedge\ t_2 \preceq_R t_1) \\
\Longrightarrow\quad & \exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge\ \exists t'_1, \bar{x}'.\mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \neg\exists t'_2, \bar{x}'_2.\mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
& \wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
& \wedge\ t_1 \preceq_{\mathrm{R}} t'_1 \wedge t_2 \preceq_{\mathrm{R}} t'_1 \wedge\ \neg t_1 = t'_1 \wedge \neg t_2 = t'_1 \\
& \wedge t_1 \preceq_R t_2 \wedge t_2 \preceq_R t_1 \\
\Longrightarrow\quad & t_1 = t_2.
\end{aligned}
$$

(3) Let t_1 and t_2 be two arbitrary tuples in R'. By definition of the order-preserving projection,

$$
\begin{aligned}
& t_1, t_2 \in \mathrm{R}'[\mathrm{T}] \\
\Longrightarrow\quad & \exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge\ \exists t'_1, \bar{x}'.\mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \neg\exists t'_2, \bar{x}'_2.\mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
& \wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
& \wedge\ t_1 \preceq_{\mathrm{R}} t'_1 \wedge t_2 \preceq_{\mathrm{R}} t'_1 \wedge\ \neg t_1 = t'_1 \wedge \neg t_2 = t'_1 \\
\Longrightarrow\quad & \exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge\ \exists t'_1, \bar{x}'.\mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \neg\exists t'_2, \bar{x}'_2.\mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
& \wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
& \wedge\ t_1 \preceq_{\mathrm{R}} t'_1 \wedge t_2 \preceq_{\mathrm{R}} t'_1 \wedge\ \neg t_1 = t'_1 \wedge \neg t_2 = t'_1 \\
& \wedge(t_1 \preceq_{\mathrm{R}} t_2 \vee t_2 \preceq_{\mathrm{R}} t_1) \\
\Longrightarrow\quad & (\exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge\ \exists t'_1, \bar{x}'.\mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \neg\exists t'_2, \bar{x}'_2.\mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
& \wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
& \wedge\ t_1 \preceq_{\mathrm{R}} t'_1 \wedge t_2 \preceq_{\mathrm{R}} t'_1 \wedge\ \neg t_1 = t'_1 \wedge \neg t_2 = t'_1 \\
& \wedge\ t_1 \preceq_R t_2)
\end{aligned}
$$

$$
\begin{aligned}
& \vee(\exists \bar{x}_1, \bar{x}_2.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{R}(t_2, \bar{x}_2) \\
& \wedge\ \exists t'_1, \bar{x}'.\mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}'_1) \\
& \neg\exists t'_2, \bar{x}'_2.\mathrm{R}(t'_2, \bar{x}'_2) \wedge \mathrm{p}(t'_2, \bar{x}'_2) \\
& \wedge\ t'_2 \preceq_{\mathrm{R}} t'_1 \wedge \neg t'_2 = t'_1 \\
& \wedge\ t_1 \preceq_{\mathrm{R}} t'_1 \wedge t_2 \preceq_{\mathrm{R}} t'_1 \wedge\ \neg t_1 = t'_1 \wedge \neg t_2 = t'_1 \\
& \wedge\ t_2 \preceq_R t_1) \\
\Longrightarrow\quad & t_1 \preceq_{\mathrm{R}'} t_2 \vee t_2 \preceq_{\mathrm{R}'} t_1.
\end{aligned}
$$

This finishes the proof that the until operation is a valid ordered relational query. □

5.2.3 Derived Operations

The previous section has defined all ordered operations in the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$. Before the completeness theorem is proven for the general ordered conjunctive queries in $\boldsymbol{CA}_{\preceq}$ in the following section, a number of derived operations are introduced, which are defined by ordered operations in $\boldsymbol{CA}_{\preceq}$. These derived operations are used to simplify the notations when it is applied in the following development.

(1) The *first* operator $\mathtt{first}_k$: $\mathbf{R} \to \mathbf{R}$. Its definition is analogous to that in $\boldsymbol{CA}^T_{\preceq}$.

(2) The *last operator* $\mathtt{last}_k$: $\mathbf{R} \to \mathbf{R}$. Its definition is also analogous to that in $\boldsymbol{CA}^T_{\preceq}$.

(3) The *since* operator $\mathtt{since}_{\mathrm{p}}$: $\mathbf{R} \to \mathbf{R}$ is derived from the until operator. The subscript p is also a predicate on $\boldsymbol{x}$-variables. A formal definition of the since operator is given as follows:

$$
\mathtt{since}_{\mathrm{p}}\ (\mathbf{R}) = \ \nu(\mathtt{until}_{\mathrm{p}}\ (\nu(\mathbf{R}))),
$$

where p is a predicate on $\boldsymbol{x}$-variables.

The since operator takes an ordered relation as its argument and returns, as its result, the tuples after the first tuple satisfying the predicate. The tuples in the result are kept in the original order of the argument. The tuple identifiers in the resulting ordered relation are identical to those in the argument relation.

5.3 Transformation Rules

In this section, we provide a set of transformation rules for the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$. Many of them are identical to those in $\boldsymbol{CA}^{X}_{\preceq}$ and $\boldsymbol{CA}^{T}_{\preceq}$; they are related to those ordered operators that are also included in $\boldsymbol{CA}^{X}_{\preceq}$ and $\boldsymbol{CA}^{T}_{\preceq}$. The other transformation rules are developed exclusively to $\boldsymbol{CA}_{\preceq}$; they are related to those ordered operators that exclusive to the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$.

In ordered relational databases, two ordered queries are equivalent if they have both equivalent data queries and equivalent order queries. In the following transformation rules, each argument can be either an ordered relation or an expression of ordered operations.

Proposition 5.5 *In addition to the transformation rules in Propositions 3.13 and 4.6, the following transformation rule holds in* $\boldsymbol{CA}_{\preceq}$*:*

$(R20)$ $\mathtt{until}_{\mathrm{p}}\ (\pi_A(\mathbf{R})) = \pi_A(\mathtt{until}_{\mathrm{p}}\ (\mathbf{R}))$ *if* $\mathbf{Att}(\mathrm{p}) \subseteq A$

Proof:

$(R20)$ Without loss of generality, we assume that $\mathrm{A} = \{X_1, \ldots, X_k\}$, and $k \leq n$. By definition of the order-preserving projection,

$$\begin{array}{rcl} \pi_A\langle R, \preceq_R\rangle & = & \langle\{(t, x_1, \ldots, x_k) \mid \exists x_{k+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n)\}, \\ & & \{(t_1, t_2) \mid \exists x_{11}, \ldots, x_{1n}.\mathrm{R}(t_1, x_{11}, \ldots, x_{1n}) \\ & & \wedge\exists x_{21}, \ldots, x_{2n}.\mathrm{R}(t_2, x_{21}, \ldots, x_{2n}) \wedge t_1 \preceq_R t_2\}\rangle. \end{array}$$

Then,

$$\begin{array}{rcl} \mathtt{until}_{\mathrm{p}}\ (\pi_A\langle R, \preceq_R\rangle) & = & \langle\{(t, x_1, \ldots, x_k) \mid \exists x_{k+1}, \ldots, x_n.\mathrm{R}(t, x_1, \ldots, x_n) \\ & & \wedge\exists t_1, \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \mathrm{p}(t_1, \bar{x}_1) \\ & & \wedge\ t \preceq_{\mathrm{R}} t_1 \wedge \neg t = t_1\}, \\ & & \{(t_1, t_2) \mid \exists x_{11}, \ldots, x_{1n}.\mathrm{R}(t_1, x_{11}, \ldots, x_{1n}) \\ & & \wedge\exists t', \bar{x}'.\mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}') \\ & & \wedge\ t_1 \preceq_{\mathrm{R}} t' \wedge \neg t' = t_1 \\ & & \wedge\exists x_{21}, \ldots, x_{2n}.\mathrm{R}(t_2, x_{21}, \ldots, x_{2n}) \\ & & \wedge\exists t', \bar{x}'.\mathrm{R}(t', \bar{x}') \wedge \mathrm{p}(t', \bar{x}') \\ & & \wedge\ t_2 \preceq_{\mathrm{R}} t' \wedge \neg t' = t_2 \\ & & \wedge t_1 \preceq_R t_2\}\rangle \\ & = & \pi_A(\mathtt{until}_{\mathrm{p}}\ (\mathbf{R})). \end{array}$$

□

5.4 Completeness of $\boldsymbol{CA}_{\preceq}$

In this section, the expressive power of the general ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$ is examined by being compared with the first-order query language $\boldsymbol{CQ}_{\preceq}$. Especially, the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$ is proven to be complete in that any ordered conjunctive query in $\boldsymbol{CQ}_{\preceq}$ can be expressed by a finite sequence of ordered operations from $\boldsymbol{CA}_{\preceq}$. To prove the completeness of $\boldsymbol{CA}_{\preceq}$, a number of notations need to be defined, which will be used in proofs later.

Definition 5.6 (Subsets of Ordered Relations) *Let $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ be an ordered relation, and ψ be a first-order formula defined by the BNF rule:*

$$\psi \;=\; \mathrm{R}(t, \bar{x}) \mid x_i = x_j \mid t \preceq_R t' \mid \neg\psi \mid \psi \wedge \psi \mid \exists x.\psi \mid \exists t.\psi.$$

A subset *S of an ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ is a set of tuples which satisfy a subformula of ψ, ψ', with one $\boldsymbol{t}$-variable as the only free variable. We say the subset S is specified by the subformula ψ'.*

Definition 5.7 (Combinations of Subsets) *Let $\langle \mathrm{R}_1, \preceq_{\mathrm{R}_1} \rangle, \ldots, \langle \mathrm{R}_n, \preceq_{\mathrm{R}_n} \rangle$ be n ordered relations, and ψ be a first-order formula defined by the BNF rule:*

$$\psi \;=\; \mathrm{R}(t, \bar{x}) \mid x_i = x_j \mid t \preceq_R t' \mid \neg\psi \mid \psi \wedge \psi \mid \exists x.\psi \mid \exists t.\psi.$$

Let S_{j_i} be an arbitrary subset of $\langle \mathrm{R}_i, \preceq_{\mathrm{R}_i} \rangle$, specified by a subformula of ψ, for $i = 1, ..., n$. A combination of subsets *$(S_{j_1}, \ldots, S_{j_n})$ is a set of the tuples, each of which is a combination of tuples from $S_{j_1}, \ldots, S_{j_n}$, respectively.*

Note that a subset S is a partition when S is specified by a conjunction of data predicates on $\boldsymbol{x}$-variables; a subset S is a subsequence when S is specified by a boolean combination of predicates on $\boldsymbol{t}$-variables. Furthermore, a combination of subsets $(S_{j_1}, \ldots, S_{j_n})$ is a subset of $\mathrm{R}_1 \times \ldots \times \mathrm{R}_n$.

Definition 5.8 (Minimal Subsets) *Let $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ be an ordered relation, and ψ be a first-order formula defined by the BNF rule:*

$$\psi \ = \ \mathrm{R}(t, \bar{x}) \mid x_i = x_j \mid t \preceq_R t' \mid \neg\psi \mid \psi \wedge \psi \mid \exists x.\psi \mid \exists t.\psi.$$

A subset S of an ordered relation $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, specified by a subformula of ψ, is a minimal subset *if there is no subset S' in $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$ which is specified by a subformula of ψ and satisfies $S' \subset S$.*

We first prove a lemma for a special case of $\boldsymbol{CQ}_{\preceq}$, which is the ω queries. The output of an ω query has the same data relation as the input ordered relation. An ω query in $\boldsymbol{CQ}_{\preceq}$ is an ordered conjunctive query in $\boldsymbol{CQ}_{\preceq}$ with an identity data query.

Lemma 5.9 *Any ω query in $\boldsymbol{CQ}_{\preceq}$ is equivalent to an ordered query expressed by a finite sequence of ordered operations from $\boldsymbol{CA}_{\preceq}$.*

Proof: Let a $\boldsymbol{CQ}_{\preceq}$ query $\langle \varphi, \psi \rangle$ be an ω query on $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, and let $\langle \mathrm{R}', \preceq_{\mathrm{R}'} \rangle$ be the output ordered relation.

By definition of the ω query, the query $\langle \varphi, \psi \rangle$ has the form of

$$\langle \varphi, \psi \rangle(\langle R, \preceq_R \rangle) = \langle \{(t, \bar{x}) | \mathrm{R}(t, \bar{x})\}, \ \{(t_1, t_2) | \psi(t_1, t_2)\} \rangle,$$

and

$$\psi = \exists \bar{x}_1.\mathrm{R}(t_1, \bar{x}_1) \wedge \exists \bar{x}_2.\mathrm{R}(t_2, \bar{x}_2) \wedge \psi'(t_1, t_2).$$

By definition of $\boldsymbol{CQ}_{\preceq}$, ψ' is a first-order formula defined by the BNF rule

$$\psi' \ = \ \mathrm{R}(t, \bar{x}) \mid t \preceq_R t' \mid x_i = x_j \mid \neg\psi' \mid \psi' \wedge \psi' \mid \exists x.\psi' \mid \exists t.\psi',$$

and ψ' has the Separation Property.

Let t_1 and t_2 be the two free $\boldsymbol{t}$-variables in ψ'. If t_1 and t_2 satisfy $\psi'(t_1, t_2)$, then $t_1 \preceq_{\mathrm{R}'} t_2$. By definition, ψ' is a boolean combination of existentially quantified formulas with only t_1 free, with existentially quantified formulas with only t_2 free, and with atomic formulas $t_1 \preceq t_2$ or $t_2 \preceq t_1$. Thus, the top level boolean connectives can be rearranged and ψ' can be rewritten as a disjunction of conjunctions in the form

$$\bigvee_i (\alpha_1^i(t_1) \wedge \alpha_2^i(t_2)) \vee \bigvee_j \beta^j(t_1, t_2),$$

where $\alpha_1^i(t_1)$ is an existentially quantified formula with only t_1 free, and $\alpha_2^i(t_2)$ is an existentially quantified formula with only t_2 free, and $\beta^j(t_1, t_2)$ is a conjunction of existentially quantified formulas with only t_1 free, with existentially quantified formulas with only t_2 free, and with atomic formulas $t_1 \preceq t_2$ or $t_2 \preceq t_1$. Therefore, if t_1 and t_2 satisfy one of $\alpha_1^i(t_1) \wedge \alpha_2^i(t_2)$ or $\beta^j(t_1, t_2)$, then $t_1 \preceq_{\mathrm{R}'} t_2$.

Next, we consider the first part of the order formula ψ', which is $\bigvee_i(\alpha_1^i(t_1) \wedge \alpha_2^i(t_2))$. It will be proven that $\bigvee_i(\alpha_1^i(t_1) \wedge \alpha_2^i(t_2))$ is equivalent to a disjunction of conjunction of the form

$$\bigvee_i(\alpha^i_{<t_1}(t_1) \wedge \alpha^i_{t_1}(t_1) \wedge \alpha^i_{>t_1}(t_1)) \wedge (\alpha^i_{<t_2}(t_2) \wedge \alpha^i_{t_2}(t_2) \wedge \alpha^i_{>t_2}(t_2)),$$

where

- $\alpha^i_{<t_j}$ $(j = 1, 2)$ is an existentially quantified formula on t_j, and all quantifiers are relativized to $< t_j$.
- $\alpha^i_{t_j}$ $(j = 1, 2)$ is an atomic formula on t_j.
- $\alpha^i_{>t_j}$ $(j = 1, 2)$ is an existentially quantified formula on t_j, and all quantifiers are relativized to $> t_j$.

Since ψ' has the Separation Property, and all $\boldsymbol{x}$-variables are bounded, all data predicates can be viewed as predicates on $\boldsymbol{t}$-variables. Thus, ψ' can be viewed as a formula without data predicates, *i.e.*, a formula with only $\boldsymbol{t}$-variables and predicates on $\boldsymbol{t}$-variables. Hence, by the Separation Theorem [39], the formula $\alpha_1^i(t_1)$ is equivalent to a disjunction of the conjunctions of the form

$$\bigvee_i(\alpha^i_{<t_1}(t_1) \wedge \alpha^i_{t_1}(t_1) \wedge \alpha^i_{>t_1}(t_1)),$$

and $\alpha_1^i(t_2)$ is equivalent to a disjunction of the conjunctions of the form

$$\bigvee_j(\alpha^j_{<t_2}(t_2) \wedge \alpha^j_{t_2}(t_2) \wedge \alpha^j_{>t_2}(t_2)).$$

Then, each conjunct $\alpha_1^i(t_1) \wedge \alpha_2^i(t_2)$ is equivalent to

$$\bigvee_{i,j}(\alpha^i_{<t_1}(t_1) \wedge \alpha^i_{t_1}(t_1) \wedge \alpha^i_{>t_1}(t_1)) \wedge (\alpha^j_{<t_2}(t_2) \wedge \alpha^j_{t_2}(t_2) \wedge \alpha^j_{>t_2}(t_2)).$$

Consequently, $\bigvee_i(\alpha_1^i(t_1) \wedge \alpha_2^i(t_2))$ is equivalent to a disjunction of conjunctions

$$\bigvee_i(\alpha^i_{<t_1}(t_1) \wedge \alpha^i_{t_1}(t_1) \wedge \alpha^i_{>t_1}(t_1)) \wedge (\alpha^i_{<t_2}(t_2) \wedge \alpha^i_{t_2}(t_2) \wedge \alpha^i_{>t_2}(t_2)).$$

We now need to prove that $\alpha^i_{<t_1}$ is equivalent to a normal form of a conjunction of the form (or a negation of the form)

$$\exists s_1, x_1, ..., s_n, x_n.\alpha_1(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n) \wedge s_1 \preceq_R ... \preceq_R s_n \preceq_R t_1,$$

where α_j $(j = 1, ..., n)$ is a conjunction of atoms in forms of $\mathrm{R}(s, \bar{x})$ or $\mathrm{R}(s, \bar{x}) \wedge x_i = x_j$.

This can be proven by induction on the depth of quantifiers n in $\alpha^i_{<t_1}$. Recall that $\alpha^i_{<t_1}$ is defined by the BNF rule

$$\psi' \ = \ \mathrm{R}(t) \mid t \preceq_R t' \mid x_i = x_j \mid \neg\psi' \mid \psi' \wedge \psi' \mid \exists t.\psi'.$$

Base case: in the case that $n = 0$, t_1 is the only possible $\boldsymbol{t}$-variable in $\alpha^i_{<t_1}$. Hence, $\alpha^i_{<t_1}$ is empty in this case.

In the case that $n = 1$, by the construction rule, $\alpha^i_{<t_1}$ may have the form $\exists s_1, x_1.\alpha^i(s_1, x_1) \wedge s_1 \preceq_{\mathrm{R}} t_1$ or $\neg\exists s_1, x_1.\alpha^i(s_1, x_1) s_1 \preceq_{\mathrm{R}} t_1$. This is already in the normal form.

Inductive step: Assume that $\alpha^i_{<t_1}$ has the normal form in the case n. It needs to show that this hypothesis holds in the case $n + 1$.

Let φ be an arbitrary existentially quantified formula on t_1 and all quantifiers are relativized to $< t_1$. By the induction hypothesis, φ has the normal form of a conjunction of the form (or a negation of the form)

$$\exists s_1, x_1, ..., s_n, x_n.\alpha_1(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n) \wedge s_1 \preceq_R ... \preceq_R s_n \preceq_R t_1.$$

For the case of $n + 1$, it is sufficient to treat the cases

$$\exists s, x.\varphi(t_1) \wedge \alpha(s, x) \wedge \varphi'(s, s_1, x_1, ..., s_n, t_1),$$

and

$$\neg\exists s, x.\varphi(t_1) \wedge \alpha(s, x) \wedge \varphi'(s, s_1, x_1, ..., s_n, t_1),$$

where φ has the quantifier depth n, the part $\alpha(s, x)$ is a conjunction of forms $\mathrm{R}(s, x)$ and $\mathrm{p}(x)$, and φ' is a conjunction of forms $t_i \preceq_{\mathrm{R}} t_j$.

First, the case $\exists s, x.\varphi(t_1) \wedge \alpha(s, x) \wedge \varphi'(s, s_1, x_1, ..., s_n, t_1)$ is considered. Suppose that s is related to one of conjuncts in φ in the above form. Since all quantifiers are relativized to $< t_1$, and $s_1 \preceq_R ... \preceq_R s_n \preceq_{\mathrm{R}} t_1$, so φ' must be in one of the following forms:

1. $s = s_j$ $(\ j = 1, ..., n)$,

2. $s \preceq_R s_1$,
3. $s_j \preceq_R s \preceq_R s_{j+1}$ ($j = 1, ..., n-1$),
4. $s_n \preceq_R s \preceq_R t_1$.

In the first case, s is one of s_j, for $j = 1, ..., n$. Thus, $\exists s, x.\varphi(t_1) \wedge \alpha(s, x) \wedge \varphi'(s, s_1, x_1, ..., s_n, t_1)$ is equivalent to $\varphi \wedge \alpha(s, x)$, and hence is equivalent to a normal form by the induction hypothesis.

In the last three cases, the formula $\exists s, x.\varphi(t_1) \wedge \alpha(s, x) \wedge \varphi'(s, s_1, x_1, ..., s_n, t_1)$ is equivalent to a normal form such that each conjunct is the same as its counterpart in the normal form of φ, except that the conjunct which is related to s changes to the following form

$$\exists s'_1, ..., s'_n, s'_{n+1}.\alpha'_1(s'_1) \wedge \ldots \wedge \alpha'_n(s'_n)$$
$$\wedge \alpha'_{n+1}(s'_{n+1}) \wedge s'_1 \preceq_R ... \preceq_R s'_n \preceq_R s'_{n+1} \preceq_R t_1,$$

where $s'_1, ..., s'_n, s'_{n+1}$ is a rearranged sequence of s and $s_1, ..., s_{i_1}$ with s placed accordingly to the different cases. The subformulas $\alpha'_1, ..., \alpha'_n, \alpha'_{n+1}$ are corresponding to $s'_1, ..., s'_n, s'_{n+1}$ respectively.

Then, the case of $\neg\exists s, x.\varphi(t_1) \wedge \alpha(s, x) \wedge \varphi'(s, s_1, x_1, ..., s_n, t_1)$ is considered. Suppose that s is related to one of the conjuncts of φ in the above form. Since all quantifiers are relativized to $< t_1$, and $s_1 \preceq_R ... \preceq_R s_n \preceq_{\mathrm{R}} t_1$, the formula φ' must be in one of the following forms:

1. $s = s_j$ ($j = 1, ..., n$),
2. $s \preceq_R s_1$,
3. $s_j \preceq_R s \preceq_R s_{j+1}$ ($j = 1, ..., n-1$),
4. $s_n \preceq_R s \preceq_R t_1$.

In the first case, s is one of s_j, for $j = 1, ..., n$. Thus, the formula $\neg\exists s, x.\varphi(t_1) \wedge \alpha(s, x) \wedge \varphi'(s, s_1, x_1, ..., s_n, t_1)$ is equivalent to $\varphi \wedge \alpha(s, x)$, and hence is equivalent to a normal form by the induction hypothesis.

In the last three cases, $\neg\exists s, x.\varphi(t_1) \wedge \alpha(s, x) \wedge \varphi'(s, s_1, x_1, ..., s_n, t_1)$ is equivalent to a normal form such that each conjunct is the same as its counterpart in the normal form of φ, except that the conjunct which is related to s changes to the following form

$$\neg\exists s'_1, ..., s'_n, s'_{n+1}.\alpha'_1(s'_1) \wedge \ldots \wedge \alpha'_n(s'_n) \wedge \alpha'_{n+1}(s'_{n+1})$$
$$\wedge s'_1 \preceq_R ... \preceq_R s'_n \preceq_R s'_{n+1} \preceq_R t_1,$$

where $s'_1, ..., s'_n, s'_{n+1}$ is a rearranged sequence of s and $s_1, ..., s_{i_1}$ with s placed accordingly to the different cases. The subformulas $\alpha'_1, ..., \alpha'_n, \alpha'_{n+1}$ are corresponding to $s'_1, ..., s'_n, s'_{n+1}$, respectively.

This completes the proof that $\alpha^i_{<t_1}$ is equivalent to a formula in the normal form of a conjunction of the form (or a negation of the form)

$$\exists s_1, x_1, ..., s_n, x_n.\alpha_1(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n) \wedge s_1 \preceq_R ... \preceq_R s_n \preceq_R t_1,$$

where α^i_j $(j = 1, ..., i_1)$ is a conjunction of predicates in the form $\mathrm{R}(s, x)$ or $\mathrm{R}(s, x) \wedge p(x)$.

Analogously, it can be proven that $\alpha^i_{>t_1}$ is equivalent to a formula in the normal form of a conjunction of the form (or a negation of the form)

$$\exists s_1, x_1, ..., s_n, x_n.\alpha_1(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n) \wedge t_1 \preceq_{\mathrm{R}} s_1 \preceq_R ... \preceq_R s_n,$$

where α^i_j $(j = 1, ..., i_1)$ is a conjunction of predicates in the form $\mathrm{R}(s, x)$ or $\mathrm{R}(s, x) \wedge p(x)$. Analogously, $\alpha^i_{<t_2}$, and $\alpha^i_{>t_2}$ are also equivalent to formulas in the normal form.

In conclusion, $\bigvee_i(\alpha^i_1(t_1) \wedge \alpha^i_2(t_2))$ is equivalent to a disjunction of conjunctions

$$\bigvee_i(\alpha^i_{<t_1} \wedge \alpha^i_{t_1} \wedge \alpha^i_{>t_1}) \wedge (\alpha^i_{<t_2} \wedge \alpha^i_{t_2} \wedge \alpha^i_{>t_2}),$$

where $\alpha^i_{<t_1}$, $\alpha^i_{>t_1}$, $\alpha^i_{<t_2}$, and $\alpha^i_{>t_2}$ are conjunctive formulas in the normal form.

Next, it needs to be proven that each positive conjunct β in the normal form of $\alpha^i_{<t_1}$, which is

$$\exists s_1, x_1, ..., s_n, x_n.\alpha_1(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n) \wedge s_1 \preceq_R ... \preceq_R s_n \preceq_R t_1,$$

can be expressed with a finite sequence of ordered operations from $\boldsymbol{CA}_{\preceq}$. This can be proven by induction on the quantifier depth n of the conjunct β.

Base case: In the case $n = 0$, t_1 is the only possible $\boldsymbol{t}$-variable, so the conjunct β is always empty.

In the case $n = 1$, the conjunct β has the form $\exists s_1, x_1.\alpha_1(s_1, x_1) \wedge s_1 \preceq_R t_1$. Thus, in the case that $\alpha_1(s_1, x_1) = R(s_1, x_1)$, the conjunct β can be expressed with ordered operations

$$\mathbf{R} - \tau(\mathbf{R});$$

in the case that $\alpha_1(s_1, x_1) = R(s_1, x_1) \wedge \mathrm{p}(x_1)$, the conjunct β can be expressed with ordered operations

$$\mathbf{R} - \mathtt{until}_{\mathrm{p}}\ (\mathbf{R}).$$

Induction Step: Assume that the conjunct with the quantifier depth n can be expressed by an ordered algebraic expression in $\boldsymbol{CA}_{\preceq}$. We want to prove that the conjunct β with the quantifier depth $n+1$ can also be expressed by an ordered algebraic expression in $\boldsymbol{CA}_{\preceq}$.

Without lost of generality, we assume that β is in the normal form

$$\begin{aligned}&\exists s_1, x_1, ..., s_n, x_n, s_{n+1}, x_{n+1}.\alpha_1(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n)\\ &\wedge\alpha_{n+1}(s_{n+1}, x_{n+1}) \wedge s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_n \preceq_R s_{n+1} \preceq_{\mathrm{R}} t_1.\end{aligned}$$

By induction hypothesis, the formula

$$\begin{aligned}&\exists s_2, x_2, ..., s_n, x_n, s_{n+1}, x_{n+1}.\alpha_2^i(s_2, x_2) \wedge \ldots \wedge \alpha_n^i(s_n, x_n)\\ &\wedge\alpha_n^i(s_{n+1}, x_{n+1}) \wedge s_2 \preceq_R ... \preceq_R s_n \preceq_R s_{n+1} \preceq_R t_1,\end{aligned}$$

can be expressed by an ordered algebraic expression $E(\mathbf{R})$ in $\boldsymbol{CA}_{\preceq}$, then

(i) in the case that $\alpha_1^i(s_1, x_1) = R(s_1, x_1)$, the conjunct β can be expressed by

$$E(\mathbf{R} - \tau(\mathbf{R}));$$

(ii) in the case that $\alpha_1^i(s_1, x_1) = R(s_1, x_1) \wedge \mathrm{p}(x_1)$, the conjunct β can be expressed by

$$E(\mathbf{R} - \mathtt{until}_{\mathrm{p}}\ (\mathbf{R})).$$

Heretofore, it has been proven that β in the positive conjunct form can be expressed by a finite sequence of ordered operations from $\boldsymbol{CA}_{\preceq}$. Consequently, we can express $\neg\beta$,

$$\begin{aligned}&\neg\exists s_1, x_1, ..., s_n, x_n, s_{n+1}, x_{n+1}.\alpha_1(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n)\\ &\wedge\alpha_{n+1}(s_{n+1}, x_{n+1}) \wedge s_1 \preceq_R ... \preceq_R s_n \preceq_R s_{n+1} \preceq_R t_1,\end{aligned}$$

by an ordered algebraic expression in $\boldsymbol{CA}_{\preceq}$,

$$\mathbf{R} - E(\mathbf{R}),$$

where $E(\mathbf{R})$ is the ordered algebraic expression of β.

Analogously, it can be proven that each positive conjunct β in the normal form of $\alpha^i_{>t_1}$,

$$\exists s_1, ..., s_n.\mathrm{R}(s_1) \wedge \ldots \wedge \mathrm{R}(s_n) \wedge t_1 \preceq s_1 \preceq_{\mathrm{R}} ... \preceq_{\mathrm{R}} s_n,$$

can be expressed with ordered operations in $\boldsymbol{CA}_{\preceq}$. This is also proven by induction on the quantifier depth n of the conjunct β.

Base case: In the case $n = 0$, t_1 is the only possible $\boldsymbol{t}$-variable, so the conjunct β is always empty.

In the case $n = 1$, the conjunct β could be a positive conjunct $\exists s_1, x_1.\alpha_1(s_1, x_1) \wedge t_1 \preceq_R s_1$. Thus,

(i) in the case that $\alpha^i_1(s_1, x_1) = \mathrm{R}(s_1, x_1)$, the conjunct β can be expressed with ordered operations

$$E_\beta = \mathbf{R} - \mathtt{last}_1\ (\mathbf{R});$$

(ii) in the case that $\alpha^i_1(s_1, x_1) = \mathrm{R}(s_1, x_1) \wedge \mathrm{p}(x_1)$, the conjunct β can be expressed with ordered operations

$$E_\beta = \mathbf{R} - \mathtt{since}_{\mathrm{p}}\ (\mathbf{R}).$$

Induction Step: The conjunct with the quantifier depth n can be expressed by an ordered algebraic expression in $\boldsymbol{CA}_{\preceq}$. It needs to prove that the conjunct β with the quantifier depth $n+1$ also can be expressed by an ordered algebraic expression in $\boldsymbol{CA}_{\preceq}$.

Without lost of generality, we assume that β is in the normal form

$$\begin{aligned}&\exists s_1, x_1, ..., s_n, x_n, s_{n+1}, x_{n+1}.\alpha_1(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n)\\ &\wedge \alpha_{n+1}(s_{n+1}, x_{n+1}) \wedge t_1 \preceq_R s_1 \preceq_R ... \preceq_R s_n \preceq_R s_{n+1}.\end{aligned}$$

By induction hypothesis, the subformula

$$\begin{aligned}&\exists s_1, x_1, ..., s_n, x_n.\alpha_2(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n)\\ &\wedge t_1 \preceq_R s_1 \preceq_R ... \preceq_R s_n,\end{aligned}$$

can be expressed by an ordered algebraic expression $E(\mathbf{R})$ in $\boldsymbol{CA}_{\preceq}$, then

(i) in the case that $\alpha^i_1(s_1, x_1) = \mathrm{R}(s_1, x_1)$, the conjunct β can be expressed by

$$E_\beta = E(\mathbf{R} - \mathtt{last}_1\ (\mathbf{R}));$$

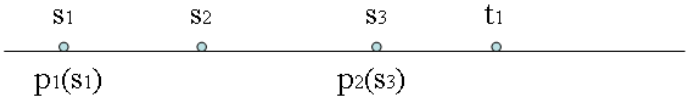

Figure 5.1: An illustration of the conjunct β

β	E_β
$\exists s_3, x_3.\mathrm{R}(s_3, x_3) \wedge \mathrm{p}_2(x_3) \wedge s_3 \preceq_\mathrm{R} t_1$	$\mathbf{R}-\ \mathtt{until}_{\mathrm{p}_2}\ (\mathbf{R})$
$\exists s_2, x_2, s_3, x_3.\mathrm{R}(s_2, x_2) \wedge \mathrm{R}(s_3, x_3) \wedge \mathrm{p}_2(x_3)$ $\wedge s_2 \preceq_\mathrm{R} s_3 \wedge s_3 \preceq_\mathrm{R} t_1$	$(\mathbf{R} - \tau(\mathbf{R}))$ $-\ \mathtt{until}_{\mathrm{p}_2}\ (\mathbf{R} - \tau(\mathbf{R}))$
$\exists s_1, x_1, s_2, x_2, s_3, x_3.\mathrm{R}(t_1, x_1) \wedge \mathrm{p}_1(x_1)$ $\wedge \mathrm{R}(t_2, x_2) \wedge \mathrm{R}(t_3, x_3) \wedge \mathrm{p}_2(x_3)$ $\wedge s_1 \preceq_\mathrm{R} s_2 \wedge s_2 \preceq_\mathrm{R} s_3 \wedge s_3 \preceq_\mathrm{R} t_1$	$(\mathbf{R}' - \tau(\mathbf{R}'))$ $-\ \mathtt{until}_{\mathrm{p}_2}\ (\mathbf{R}' - \tau(\mathbf{R}'))$, where $\mathbf{R}' = \mathbf{R}-\ \mathtt{until}_{\mathrm{p}_1}\ (\mathbf{R})$

Table 5.1: The construction of E_β by induction

(ii) in the case that $\alpha_1^i(s_1, x_1) = \mathrm{R}(s_1, x_1) \wedge \mathrm{p}(x_1)$, the conjunct β can be expressed by

$$E_\beta = E(\mathbf{R}-\ \mathtt{since}_{\mathrm{p}}\ (\mathbf{R})).$$

In Figure 5.1, an example of conjunct β is illustrated. The conjunct β is in the normal form

$$\begin{aligned}\beta = \quad & \exists s_1, x_1, s_2, x_2, s_3, x_3.\mathrm{R}(t_1, x_1) \wedge \mathrm{p}_1(x_1) \\ & \wedge \mathrm{R}(t_2, x_2) \wedge \mathrm{R}(t_3, x_3) \wedge \mathrm{p}_2(x_3) \\ & \wedge s_1 \preceq_\mathrm{R} s_2 \wedge s_2 \preceq_\mathrm{R} s_3 \wedge s_3 \preceq_\mathrm{R} t_1.\end{aligned}$$

The algebraic expression of β in $\boldsymbol{CA}_\preceq$, E_β, is constructed by induction, as shown in Table 5.1.

Consequently, we can express the conjunct $\neg\beta$,

$$\begin{aligned}&\neg\exists s_1, x_1, ..., s_n, x_n, s_{n+1}, x_{n+1}.\alpha_1(s_1, x_1) \wedge \ldots \wedge \alpha_n(s_n, x_n) \\ &\wedge \alpha_{n+1}(s_{n+1}, x_{n+1}) \wedge t_1 \preceq_\mathrm{R} s_1 \preceq_\mathrm{R} ... \preceq_\mathrm{R} s_n \preceq_\mathrm{R} s_{n+1},\end{aligned}$$

by the ordered algebraic expression in $\boldsymbol{CA}_\preceq$,

$$\mathbf{R} - E_\beta(\mathbf{R}),$$

where $E_\beta(\mathbf{R})$ is the ordered algebraic expression of β in $\boldsymbol{CA}_{\preceq}$.

Last, we consider $\alpha^i_{t_1}$ in ψ'. Since there is no quantifier in $\alpha^i_{t_1}$, and it is a conjunction of predicates $R(t,x)$ and $p(x)$, it can be expressed by the order identity operation or by ordered selection operations in $\boldsymbol{CA}_{\preceq}$.

In general, the conjunction $\alpha^i_{<t_1} \wedge \alpha^i_{t_1} \wedge \alpha^i_{>t_1}$ can be expressed with ordered operations

$$E_1 \cap E_2 \cap \ldots \cap E_k,$$

where each E_i $(i = 1, ..., k)$ is an ordered algebraic expression, which is composed of ordered operations in $\boldsymbol{CA}_{\preceq}$: top operation τ, first operation $\texttt{first}_i$, last operation $\texttt{last}_i$, until operation $\texttt{until}_\text{p}$, since operation $\texttt{since}_\text{p}$, ordered selection σ, and set difference operation $-$. Intuitively, the conjunction $\alpha^i_{<t_1} \wedge \alpha^i_{t_1} \wedge \alpha^i_{>t_1}$ represents a subset S_1 of $\langle \text{R}, \preceq_\text{R} \rangle$, which is decided by data predicates, by order of tuple identifers, or by both data predicates and order of tuple identifers in this conjunct.

Analogously, it can be proven that $\alpha^i_{<t_2} \wedge \alpha^i_{t_2} \wedge \alpha^i_{>t_2}$ can be expressed with ordered operations in $\boldsymbol{CA}_{\preceq}$. Intuitively, the conjunction $\alpha^i_{<t_2} \wedge \alpha^i_{t_2} \wedge \alpha^i_{>t_2}$ represents a subset S_2 of $\langle \text{R}, \preceq_\text{R} \rangle$, which is decided by data predicates, by order of tuple identifiers, or by both data predicates and order of tuple identifers in this conjunct.

It has been proven that we can express $\alpha^i_1(t_1)$ and $\alpha^i_2(t_2)$ with a sequence of ordered operations in $\boldsymbol{CA}_{\preceq}$. Suppose that S_1 and S_2 are specified by $\alpha^i_1(t_1)$ and $\alpha^i_2(t_2)$ respectively. If any pair of tuples, t_1 and t_2, satisfies $\alpha^i_1(t_1) \wedge \alpha^i_2(t_2)$, then $t_1 \in S_1$ and $t_2 \in S_2$ hold. It follows that S_1 is prior to S_2 in the output order $\preceq_{\text{R}'}$.

Let $S_1, \ldots, S_n$ be subsets specified by $\bigvee_i \alpha^i_1(t_1) \wedge \alpha^i_2(t_2)$. Without lost of generality, we assume that $S_1, \ldots, S_n$ is a set of disjoint minimal subsets, and $\bigcup_{i=1,\ldots,n} S_i = \langle \text{R}, \preceq_\text{R} \rangle$. (If subsets are not disjoint or not minimal, we can always separate them into minimal subsets.) It needs to be shown that $\bigvee_i (\alpha^i_1(t_1) \wedge \alpha^i_2(t_2))$ defines a linear order on subsets $S_1, \ldots, S_n$.

This can be proven by contradiction. Suppose that $\bigvee_i (\alpha^i_1(t_1) \wedge \alpha^i_2(t_2))$ does not define a linear order among the subsets $S_1, \ldots, S_n$. There are two cases:

In the first case, assume that there exist two subsets, say S_1 and S_2, such that both $S_1 \preceq_{R'} S_2$ and $S_2 \preceq_{R'} S_1$ hold at the same time. If $t_1 \in S_1 \wedge t_2 \in S_2$, then by the assumption $(t_1, t_2) \in \preceq_{R'}$ and $(t_2, t_1) \in \preceq_{R'}$ hold at the same time. This is a contradiction to the fact that the output order $\preceq_{R'}$ is a linear order. Hence, the assumption in Case 1 does not hold.

In the second case, assume that there exist two subsets, say S_1 and S_2, such that neither $S_1 \preceq_{R'} S_2$ nor $S_2 \preceq_{R'} S_1$ holds; their order in output is not specified by $\bigvee_i \alpha^i$.

By this assumption, for any pair of tuples $t_1 \in S_1$ and $t_2 \in S_2$, their order in the output is not defined by $\bigvee_i \alpha_1^i(t_1) \wedge \alpha_2^i(t_2)$. Because the output order is a linear and total order on all tuples, the order of t_1 and t_2 must be decided by ψ'. Suppose that ψ' decides $t_1 \preceq_{R'} t_2$. Since S_1 and S_2 are minimal, all pairs of tuples from S_1 and S_2 respectively must have the same order if they have any order at all. Thus, for any pair of tuples t_1 and t_2, if $t_1 \in S_1 \wedge t_2 \in S_2$, then $t_1 \preceq_{R'} t_2$. It follows $S_1 \preceq_{R'} S_2$, which is a contradiction to the assumption.

Since ψ' defines a linear order on all tuples in the input $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, the order inside each subset S_i has to be decided by the second part of ψ', which is $\bigvee_j \beta^j(t_1, t_2)$. By definition of $\boldsymbol{CQ}_{\preceq}$, ψ' satisfies the Separation Property, and therefore is equivalent to a normal form

$$\bigvee_j (\beta_1^j(t_1) \wedge \beta_2^j(t_2) \wedge \beta_3^j(t_1, t_2)),$$

where $\beta_1^j(t_1)$ and $\beta_2^j(t_2)$ are existentially quantified formulas with only t_1 free and with only t_2 free respectively, and $\beta_3^j(t_1, t_2)$ is either $t_1 \preceq_{\mathrm{R}} t_2$ or $t_2 \preceq_{\mathrm{R}} t_1$.

As we proved earlier, $\beta_1^j(t_1)$ and $\beta_2^j(t_2)$ specify subsets of the input, and can be expressed by a sequence of ordered operations in $\boldsymbol{CA}_{\preceq}$. Furthermore, $\beta_1^j(t_1)$ and $\beta_2^j(t_2)$ must specify the same subsets; otherwise, $\beta_1^j(t_1) \wedge \beta_2^j(t_2) \wedge \beta_3^j(t_1, t_2)$ defines a partial order for t_1 and t_2 satisfying this conjunct, and it is impossible to define a total linear order for t_1 and t_2 which satisfy $\beta_1^j(t_1)$ and $\beta_2^j(t_2)$. Therefore, each conjunct $\beta_1^j(t_1) \wedge \beta_2^j(t_2) \wedge \beta_3^j(t_1, t_2)$ defines an order for pairs of tuples from the same subset S_i, and the only possible orders in S_i are identical or reverse to the original order.

In summary, the linear order decided by ψ' can be expressed by a sequence of ordered operators,

$$\bigcup_i \beta(E_{i1}(\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle) \cap ... \cap E_{ik_i}(\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)),$$

where β is either the order identity operation μ or the order reverse operation ν, and $E_{i1}, ..., E_{ik_i}$ are ordered algebraic expressions constructed by induction as above, each of which is composed of ordered operations in $\boldsymbol{CA}_{\preceq}$: the top operation τ, first operation $\texttt{first}_i$, last operation $\texttt{last}_i$, until operation $\texttt{until}_{\mathrm{p}}$, since operation $\texttt{since}_{\mathrm{p}}$, ordered selection σ, and set difference operation $-$.

□

We have shown that the ω queries in $\boldsymbol{CQ}_{\preceq}$ are completely expressible by $\boldsymbol{CA}_{\preceq}$. We are now ready for the main result of the investigation in this chapter, which is the completeness theorem for the general ordered conjunctive queries $\boldsymbol{CQ}_{\preceq}$.

Theorem 5.10 *Any $\boldsymbol{CQ}_{\preceq}$ query is equivalent to an ordered query expressed by a finite sequence of ordered operations from $\boldsymbol{CA}_{\preceq}$.*

Proof: Let $\langle \phi, \psi \rangle$ be an ordered conjunctive query in $\boldsymbol{CQ}_{\preceq}$ on ordered relations $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$.

Case 1: $n = 1$.

In this case, the ordered conjunctive query $\langle \phi, \psi \rangle$ has only one argument $\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle$, and hence has the form

$$\langle \phi, \psi \rangle(\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle) = \langle \{(t, \bar{x}) \mid \phi(t, \bar{x})\},\ \{(t_1, t_2) \mid \psi(t_1, t_2)\} \rangle,$$

where $\phi(t, \bar{x})$ is a conjunctive query and has the normal form

$$\pi_A(\sigma_{\mathrm{p}}(\mathrm{R})),$$

and

$$\psi = \exists \bar{x}_1 . \phi(t_1, \bar{x}_1) \wedge \exists \bar{x}_2 . \phi(t_2, \bar{x}_2) \wedge \psi'(t_1, \bar{x}_1, t_2, \bar{x}_2).$$

By definition of $\boldsymbol{CQ}_{\preceq}$, ψ' is a first-order formula defined by the BNF rule

$$\psi' \ = \ \mathrm{R}(t, \bar{x}) \mid x_i = x_j \mid t \preceq_R t' \mid \neg \psi' \mid \psi' \wedge \psi' \mid \exists x . \psi' \mid \exists t . \psi',$$

and ψ' has the Separation Property (see Definition 5.1).

By Lemma 5.9, there exists an algebraic expression in $\boldsymbol{CA}_{\preceq}$ to express the ordered query $\langle \omega, \psi' \rangle$,

$$\bigcup_i \beta(E_{i1}(\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle) \cap \ldots \cap E_{ik_i}(\langle \mathrm{R}, \preceq_{\mathrm{R}} \rangle)),$$

where β is either the order identity operation μ or the order reverse operation ν, and $E_{i1}, \ldots, E_{ik_i}$ are constructed by induction on the depth of quantifiers as in the proof of Lemma 5.9. Each of expressions $E_{i1}, \ldots, E_{ik_i}$ is composed of ordered

operations in $\boldsymbol{CA}_{\preceq}$: top operation τ, first operation $\texttt{first}_i$, last operation $\texttt{last}_i$, until operation $\texttt{until}_\text{p}$, since operation $\texttt{since}_\text{p}$, order-preserving selection σ, and set difference operation $-$.

By Proposition 5.5, order-preserving selections and projections can propagate through order identity, order reverse, and order concatenation operators. To express the selections and projections on the data relation R in the normal form of ϕ, order-preserving selections and order-preserving projections are applied respectively to the above expression without changing the order.

Finally, the ordered $\boldsymbol{CQ}_{\preceq}$ query $\langle \phi, \psi \rangle$ is equivalent to the algebraic expression in $\boldsymbol{CA}_{\preceq}$,

$$\pi_A(\sigma_\text{p} \bigcup_i \beta(E_{i1}(\langle \text{R}, \preceq_\text{R} \rangle) \cap ... \cap E_{ik_i}(\langle \text{R}, \preceq_\text{R} \rangle))),$$

where β is either the order identity operation μ or the order reverse operation ν, and expressions $E_{i1}, ..., E_{ik_i}$ are constructed by induction on the depth of quantifiers as in the proof of Lemma 5.9. Each of expressions $E_{i1}, ..., E_{ik_i}$ is composed of ordered operations in $\boldsymbol{CA}_{\preceq}$: top operation τ, first operation $\texttt{first}_i$, last operation $\texttt{last}_i$, until operation $\texttt{until}_\text{p}$, since operation $\texttt{since}_\text{p}$, order-preserving selection σ, and set difference operation $-$.

Case 2: $n > 1$.

Let $\langle \varphi, \psi \rangle$ be a $\boldsymbol{CQ}_{\preceq}$ query on ordered relations $\langle R_1, \preceq_{R_1} \rangle, \ldots, \langle R_n, \preceq_{R_n} \rangle$, where $n > 1$. The data query φ is a conjunctive query and hence has the normal form

$$\exists x_1, \ldots, x_m.(\text{R}_1(u_1) \wedge \ \cdots \wedge \text{R}_n(u_n)),$$

where the symbols u_i are tuples containing both variables $t, x_1, \ldots, x_n$ and constants for data attributes.

The order query ψ has the form

$$\begin{aligned} \{(t_1, t_2) \mid \quad & \exists t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots \bar{x}_{2n}. \\ & t_1 = (t_{11}, \ldots, t_{1n}) \wedge \cdots \wedge t_n = (t_{21}, \ldots, t_{2n}) \\ & \wedge \varphi(t_{11}, \ldots, t_{1n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}) \wedge \ \varphi(t_{21}, \ldots, t_{2n}, \bar{x}_{21}, \ldots, \bar{x}_{2n}) \\ & \wedge \psi'(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}, \bar{x}_{11}, \ldots, \bar{x}_{1n}, \bar{x}_{21}, \ldots, \bar{x}_{2n})\}. \end{aligned}$$

By definition of $\boldsymbol{CQ}_{\preceq}$, ψ' is a first-order formula defined by the BNF rule

$$\psi' \ = \ \text{R}(t, \bar{x}) \mid x_i = x_j \mid t \preceq_R t' \mid \neg\psi' \mid \psi' \wedge \psi' \mid \exists x.\psi' \mid \exists t.\psi',$$

and ψ' has the Separation Property. Semantically, ψ' defines a linear order on the Cartesian product of $\text{R}_1 \times \cdots \times \text{R}_n$.

By the definition, ψ' is a boolean combination of existentially quantified formulas with only one $\boldsymbol{t}$-variable free, which could be $t_{11}, \ldots, t_{1n}, t_{21}, \ldots,$ or t_{2n}. Therefore, we can rearrange the top level boolean connectives and rewrite ψ' in a disjunction of conjunctions in the form

$$\bigvee_i \alpha^i(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}) \vee \bigvee_j \beta^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}),$$

where each $\alpha^i(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})$ is of the form

$$(\alpha^i_{11}(t_{11}) \wedge \ldots \wedge \alpha^i_{1n}(t_{1n}) \wedge \alpha^i_{21}(t_{21}) \wedge \ldots \wedge \alpha^i_{2n}(t_{2n})),$$

and each $\beta^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})$ is of the form

$$(\beta^j_1(t_{11}, t_{21}) \wedge \ldots \wedge \beta^j_n(t_{1n}, t_{2n})),$$

where $\alpha^i_{1k}(t_{1k})$ and $\alpha^i_{2k}(t_{2k})$ are existentially quantified formulas with only t_{1k} free and existentially quantified formulas with only t_{2k} free, respectively; the conjunct $\beta^j_k(t_{1k}, t_{2k})$ is a conjunction of existentially quantified formulas with only t_{1k} free, existentially quantified formulas with only t_{2k} free, and atomic formulas $t_{1k} \preceq_{\mathrm{R}_k} t_{2k}$ or $t_{2k} \preceq_{\mathrm{R}_k} t_{1k}$, for $k = 1, \ldots, n$.

First, we consider the first part $\bigvee_i \alpha^i$ of the order formula ψ'. For each $\alpha^i(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})$, the conjuncts $\alpha^i_{1k}(t_{1k})$ and $\alpha^i_{2k}(t_{2k})$ are existentially quantified formulas with only t_{1k} free and existentially quantified formulas with only t_{2k} free respectively, for $k = 1, \ldots, n$. By Lemma 5.9, they can be expressed by an ordered algebraic expression in $\boldsymbol{CA}_{\preceq}$,

$$\bigcup_i \beta(E_{i1}\langle \mathrm{R}_k, \preceq_{\mathrm{R}_k}\rangle \cap \ldots \cap E_{ik_i}\langle \mathrm{R}_k, \preceq_{\mathrm{R}_k}\rangle),$$

where β is either the order identity operation μ or the order reverse operation ν, and expressions $E_{i1}, \ldots, E_{ik_i}$ are constructed by induction on the depth of quantifiers as in the proof of Lemma 5.9. Each of expressions $E_{i1}, \ldots, E_{ik_i}$ is composed of ordered operations in $\boldsymbol{CA}_{\preceq}$: top operation τ, first operation $\mathtt{first}_i$, last operation $\mathtt{last}_i$, until operation $\mathtt{until}_{\mathrm{p}}$, since operation $\mathtt{since}_{\mathrm{p}}$, order-preserving selection σ, and set difference operation $-$.

Let $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$ be the two combinations of minimal subsets of ordered relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$, specified by

$$\alpha^i = (\alpha^i_{11}(t_{11}) \wedge \ldots \wedge \alpha^i_{1n}(t_{1n}) \wedge \alpha^i_{21}(t_{21}) \wedge \ldots \wedge \alpha^i_{2n}(t_{2n})).$$

Let $t_1 = (t_{11}, \ldots, t_{1n})$ and $t_2 = (t_{21}, \ldots, t_{2n})$ be a pair of tuples from $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$, respectively. Since the pair $((t_{11}, \ldots, t_{1n}), (t_{21}, \ldots, t_{2n}))$ satisfies

α^i, they will be in the resulting order relation $\preceq_{R'}$, *i.e.*, $t_1 \preceq_{R'} t_2$. Thus, $t_1 \preceq_{R'} t_2$ holds for any pair of tuples t_1 and t_2 from $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$, which leads to $(S_{11}, \ldots, S_{1n}) \preceq_{R'} (S_{21}, \ldots, S_{2n})$. Therefore, any conjunction α^i in the first part $\bigvee_i \alpha^i$ decides the order between a pair of combinations of minimal subsets $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$ of input ordered relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$.

Using the same technique in Lemma 5.9, we can prove that $\bigvee_i \alpha^i$ defines a linear order among all combinations of minimal subsets $(S_{1i_1}, \ldots, S_{ni_n})$ of ordered relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$. Suppose that $\bigvee_i \alpha^i$ does not define a linear order among all combinations of minimal subsets $(S_{1i_1}, \ldots, S_{ni_n})$ from input relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$. There are two cases.

In the first case, we assume that there exist two combinations of subsets, say $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$, such that both $(S_{11}, \ldots, S_{1n}) \preceq_{R'} (S_{21}, \ldots, S_{2n})$ and $(S_{21}, \ldots, S_{2n}) \preceq_{R'} (S_{11}, \ldots, S_{1n})$ hold at the same time. For any $t_1 \in (S_{11}, \ldots, S_{1n})$ and $t_2 \in (S_{21}, \ldots, S_{2n})$, by the assumption, $(t_1, t_2) \in \preceq_{R'}$ and $(t_2, t_1) \in \preceq_{R'}$ hold at the same time. This is a contradiction to the fact that the output order $\preceq_{R'}$ is a linear order. Hence, the assumption in Case 1 does not hold.

In the second case, we assume that there exist two combinations of subsets, say $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$, such that neither $(S_{11}, \ldots, S_{1n}) \preceq_{R'} (S_{21}, \ldots, S_{2n})$ nor $(S_{21}, \ldots, S_{2n}) \preceq_{R'} (S_{11}, \ldots, S_{1n})$ holds; their order in output is not specified by $\bigvee_i \alpha^i$.

By this assumption, for any pair of tuples $t_1 \in (S_{11}, \ldots, S_{1n})$ and $t_2 \in (S_{21}, \ldots, S_{2n})$, their order in the output is not defined by $\bigvee_i \alpha^i$. Because the output order is a linear order on all combinations of tuples from n ordered relations, the order of t_1 and t_2 must be decided by ψ'. Suppose that ψ' decides $t_1 \preceq_{R'} t_2$. Since $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$ are combinations of minimal subsets, all pairs of tuples from $(S_{11}, \ldots, S_{1n})$ and $(S_{21}, \ldots, S_{2n})$ must have the same order if they have any order at all. Then, for any pair of tuples $t_1 \in (S_{11}, \ldots, S_{1n})$ and $t_2 \in (S_{21}, \ldots, S_{2n})$, it satisfies the condition $t_1 \preceq_{R'} t_2$. It follows $(S_{11}, \ldots, S_{1n}) \preceq_{R'} (S_{21}, \ldots, S_{2n})$, which is a contradiction to the assumption.

We have proven that $\bigvee_i \alpha^i$ defines a linear order among all combinations of minimal subsets $(S_{1i_1}, \ldots, S_{ni_n})$ from $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$. Assume that each subset S_{ki_k} $(k = 1, ..., n)$ keeps the original order from the input and can be expressed by ordered operations

$$E_{ki_k}\langle R_k, \preceq_{R_k}\rangle,$$

where E_{ki_k} is of the form

$$E_{ki_k} = E^1_{ki_k} \cap ... \cap E^j_{ki_k},$$

and $E^1_{ki_k}, ..., E^j_{ki_k}$ are constructed by induction on the depth of quantifiers as in the proof of Lemma 5.9.

Next, we consider the second part of ψ',

$$\bigvee_j \beta^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}).$$

To compose a linear order on the Cartesian product of data relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$, the order between each pair of tuples from the same combination of subsets has to be decided by the order formula ψ'. This order inside a combination of subsequence is decided by the second part of ψ,

$$\bigvee_j \beta^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}).$$

Let $(S_{i_11}, \ldots, S_{i_nn})$ be a combination of minimal subsets specified by $\alpha^j_1(t_{11}), ..., \alpha^j_n(t_{1n})$ on $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$, respectively. For any pair of tuples $(t_{11}, \ldots, t_{1n})$ and $(t_{21}, \ldots, t_{2n})$ from the same combination of minimal subsets $(S_{i_11}, \ldots, S_{i_nn})$, there exists a j such that β^j decides the order of this pair in the result because the output order is a total order on all combinations of tuples from each input. Thus, the conjunction β^j is equivalent to

$$\begin{aligned}&(\alpha^j_1(t_{11}) \wedge \ldots \wedge \alpha^j_n(t_{1n}) \wedge \alpha^j_1(t_{21}) \wedge \ldots \wedge \alpha^j_n(t_{2n}))\\ &\wedge\ \psi^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n}),\end{aligned}$$

where $\alpha^j_1(t_{1i})$ defines a minimal subset S_{ji} over $\langle \mathrm{R}_i, \preceq_{\mathrm{R}_i}\rangle$, and the formula $\psi^j(t_{11}, \ldots, t_{1n}, t_{21}, \ldots, t_{2n})$ decides the real order in the combination of minimal subsets, and is defined by the BNF rule

$$\psi^j = \ t \preceq_{\mathrm{R}_i} t' \mid \neg\psi^j \mid \psi^j \wedge\ \psi^j.$$

The conjunct ψ^j has this form because variables $t_{1k}, \ldots, t_{2k}, t_{2k}, \ldots, t_{1k}$ are only $\boldsymbol{t}$-variables in the minimal subset S_{ji}, and appear only in forms $t_{1k} \preceq_{R_k} t_{2k}$ and $t_{2k} \preceq_{R_k} t_{1k}$, for $k = 1, \ldots, n$.

Since the lexicographical order is the only possible linear order inside a combination of minimum subsets, ψ^j specifies a lexicographical order inside the combination

of minimal subsets. Let $S_{i_jk_j}$ $(j = 1, ..., n)$ be the i_j-th subset in the input ordered relation $\langle R_{k_j}, \preceq_{R_{k_j}}\rangle$. The conjunct ψ^j defines the lexicographic order inside the combination of minimal subsets $(S_{i_1k_1}, \ldots, S_{i_nk_n})$, and is equivalent to a normal form

$$\epsilon(t_{1k_1} \preceq_{\mathrm{R}} t_{2k_1}) \vee \ (t_{1k_1} = t_{2k_1} \wedge \ \epsilon(t_{1k_2} \preceq_{\mathrm{R}} t_{2k_2})) \vee \ \ldots$$
$$\vee \ (t_{1k_1} = t_{2k_1} \wedge \ \ldots \ \wedge \ t_{1k_{n-1}} = t_{2k_{n-1}} \wedge \ \epsilon(t_{1k_n} \preceq_{\mathrm{R}} t_{2k_n})),$$

where $\epsilon(t_{1k_n} \preceq_{\mathrm{R}} t_{2k_n})$ is either $t_{1k_n} \preceq_{\mathrm{R}} t_{2k_n}$ or $t_{2k_n} \preceq_{\mathrm{R}} t_{1k_n}$. The sequence $(k_1, \ldots, k_n)$ is a permutation of $(1, \ldots, n)$, which decides the major to minor order that ordered relations $\langle \mathrm{R}_{k_1}, \preceq_{\mathrm{R}_{k_1}}\rangle, \ldots, \langle \mathrm{R}_{k_n}, \preceq_{\mathrm{R}_{k_n}}\rangle$ serve in the output order.

Intuitively, ψ^j defines a lexicographical order on $(S_{i_1k_1}, \ldots, S_{i_nk_n})$, and each minimal subset has an order identical or reverse to the original input order. Therefore, each combination of minimal subsets $(S_{i_1k_1}, \ldots, S_{i_nk_n})$ can be expressed by an algebraic expression in the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$ as follows:

$$\beta(E_{i_1,j_1}(\langle \mathrm{R}_{k_1}, \preceq_{\mathrm{R}_{k_1}}\rangle)) \ltimes \ \ldots \ltimes \ \beta(E_{i_n,j_n}(\langle \mathrm{R}_{k_n}, \preceq_{\mathrm{R}_{k_n}}\rangle)).$$

In addition, the sequence $(k_1, \ldots, k_n)$ is a permutation of $(1, \ldots, n)$, which indicates the order in which the left nested-loop product operations are applied to the subsets of $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$. In general, the ordered query $\langle \phi', \psi'\rangle$ (where the algebraic expression of ϕ' is $\mathrm{R}_1 \times \cdots \times \mathrm{R}_n$) can be expressed by

$$\bigcup (\beta(E_{i_1,j_1}(\langle \mathrm{R}_{k_1}, \preceq_{\mathrm{R}_{k_1}}\rangle)) \ltimes \ \ldots \ \ltimes \ \beta(E_{i_n,j_n}(\langle \mathrm{R}_{k_n}, \preceq_{\mathrm{R}_{k_n}}\rangle))).$$

Consider the data query ϕ of the $\boldsymbol{CQ}_{\preceq}$ query $\langle \phi, \psi\rangle$. If there are selections and projections in the normal form of ϕ, order-preserving selections and order-preserving projections are applied to the above algebraic expression correspondingly because order-preserving selections and order-preserving projections can propagate through left nested-loop products while keeping equivalence.

In conclusion, any $\boldsymbol{CQ}_{\preceq}$ ordered conjunctive query $\langle \phi, \psi\rangle$ on ordered relations $\langle R_1, \preceq_{R_1}\rangle, \ldots, \langle R_n, \preceq_{R_n}\rangle$ is equivalent to an ordered query expressed in $\boldsymbol{CA}_{\preceq}$:

$$\pi_A(\sigma_p(\bigcup (\beta(E_{i_1,j_1}(\langle \mathrm{R}_{k_1}, \preceq_{\mathrm{R}_{k_1}}\rangle)) \ltimes \ \ldots \ \ltimes \ \beta(E_{i_n,j_n}(\langle \mathrm{R}_{k_n}, \preceq_{\mathrm{R}_{k_n}}\rangle))))),$$

where β is either μ or ν, and $\bigcup$ is an order concatenation operator, $(k_1, \ldots, k_n)$ is a permutation of $(1, \ldots, n)$, and E_{i_k,j_k} is an ordered algebraic expression composed of ordered operations in $\boldsymbol{CA}_{\preceq}$, constructed by induction as above.

□

We have proven the completeness of the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$ under some restrictions. We now show some examples of $\boldsymbol{CQ}_{\preceq}$ queries, which can be expressed in the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$.

Example 5.11 Consider the $\boldsymbol{CQ}_{\preceq}$ queries in Example 5.3. We can express these $\boldsymbol{CQ}_{\preceq}$ queries with ordered operators in $\boldsymbol{CA}_{\preceq}$.

(1) The first query can be answered by the ordered algebraic expression in $\boldsymbol{CA}_{\preceq}$ as follows:

$$(\langle \mathtt{EMP}, \preceq_{\mathtt{EMP}} \rangle - \ \mathtt{until}_{\mathtt{Salary}=60000}\ \langle \mathtt{EMP}, \preceq_{\mathtt{EMP}} \rangle)$$
$$\cup\ (\mathtt{until}_{\mathtt{Salary}=60000}\ \langle \mathtt{EMP}, \preceq_{\mathtt{EMP}} \rangle).$$

(2) For the second query, we first express "all employees until the first one in the department Sales" as

$$E_1 = \mathtt{until}_{\mathtt{Department}=\text{“}Sales\text{”}}\ \langle \mathtt{EMP}, \preceq_{\mathtt{EMP}} \rangle.$$

Next, we express "employees between the first and second employees in the department Sales, including the second one " as

$$E_2 = \ \mathtt{until}_{\mathtt{Department}=\text{“}Sales\text{”}}\ (\langle \mathtt{EMP}, \preceq_{\mathtt{EMP}} \rangle - E_1)$$
$$-\ \mathtt{last}\ (\mathtt{until}_{\mathtt{Department}=\text{“}Sales\text{”}}\ (\langle \mathtt{EMP}, \preceq_{\mathtt{EMP}} \rangle - E_1)).$$

Then, we express "employees after the second employee in the department Sales" as

$$E_3 = \langle \mathtt{EMP}, \preceq_{\mathtt{EMP}} \rangle - E_1) - E_2.$$

Finally, we express the second query with the ordered algebraic expression in $\boldsymbol{CA}_{\preceq}$ as follows:

$$E_1 \cup \nu(E_2) \cup E_3.$$

This chapter has defined general ordered conjunctive queries in $\boldsymbol{CQ}_{\preceq}$, which are restricted to ordered conjunctive queries in $\boldsymbol{FO}_{\preceq}$ with the Separation Property. To answer general ordered conjunctive queries in $\boldsymbol{CQ}_{\preceq}$, an ordered algebra $\boldsymbol{CA}_{\preceq}$ is proposed with a set of primitive ordered operators and derived operators. The completeness theorem of the ordered conjunctive algebra $\boldsymbol{CA}_{\preceq}$ is then shown with regard to general ordered conjunctive queries in $\boldsymbol{CQ}_{\preceq}$. The chapter closes by presenting examples of ordered conjunctive queries in $\boldsymbol{CQ}_{\preceq}$, which are expressible or inexpressible in $\boldsymbol{CA}_{\preceq}$.

We have investigated the three classes of ordered conjunctive queries, $\boldsymbol{CQ}^{X}_{\preceq}$, $\boldsymbol{CQ}^{T}_{\preceq}$, and $\boldsymbol{CQ}_{\preceq}$, and examined the expressive power of corresponding algebras

$\boldsymbol{CA}^{X}_{\preceq}$, $\boldsymbol{CA}^{T}_{\preceq}$, and $\boldsymbol{CA}_{\preceq}$. In particular, the completeness problems are proven for the three classes of ordered conjunctive queries. The next chapter will conclude this dissertation with research contributions and future work of this investigation.

Chapter 6

Conclusion and Future Work

This chapter summarizes the main results of this dissertation and discuss several research directions, in which this work can be extended.

6.1 Summary of Results

This dissertation provides a systematic approach to support order in query processing in relational databases. Three important issues are studied for order support: a novel ordered database model is proposed; various ordered query representation languages are formally defined; and the expressive power of ordered query languages are investigated.

An ordered relational database model
An ordered relational database model has been proposed to capture and manipulate data orderings in relational databases. The novelties of this ordered relational database model are as follows:

- Each tuple in a relation is associated with a virtual tuple identifier. The ordered relation is composed of a data relation and such an order relation. The order relation is a virtual binary relation over tuple identifiers, which is separated from the data information of the relation. This binary order relation contains any pairs of tuple identifiers in the data relation, which defines a total linear order over tuples. An ordered relational database structure is formally defined based on this model.

- In this ordered relational database model, relational queries are processed in an ordered fashion. Both the data and order of the input are stored when a relational query is issued to the query processor. The query processor traces the order of an intermediate result at each phase of query processing and produces the resulting relation with a specific order. Given ordered inputs, each operator generates an ordered output.
- This ordered relational database structure is compared with the temporal relational structure; both differences and connections between the two structures are studied. The lemma has been proven that any temporal database can be mapped to an ordered relational database such that the mapped ordered database retains all order properties in the original temporal database.

Ordered relational queries

Ordered relational queries are defined in various ordered query representation languages, based on the ordered relational database model. The completeness problem of ordered relational algebras is also studied.

- A formal definition of ordered relational queries is provided in a two-sorted first-order calculus $\boldsymbol{FO}_{\preceq}$.
- Ordered relational queries in $\boldsymbol{FO}_{\preceq}$ are compared with temporal relational queries in 2***FOL***and the lemma has been proven that any 2***FOL*** temporal relational query can be reduced to an $\boldsymbol{FO}_{\preceq}$ ordered relational query.
- The complete problem of ordered relational algebras is defined formally. Analogously to the completeness problem of temporal relational algebras, the completeness problem of the general ordered relational algebras is most likely to have a negative answer.

Ordered conjunctive queries with data-decided order $\boldsymbol{CQ}^{X}_{\preceq}$

An ordered algebra was proposed for ordered conjunctive queries with data-decided order and was proven to be complete with respect to the two-sorted first order logic.

- A formal definition of ordered conjunctive queries $\boldsymbol{CQ}^{X}_{\preceq}$ is provided in a two-sorted first-order calculus $\boldsymbol{FO}_{\preceq}$.
- An ordered algebra $\boldsymbol{CA}^{X}_{\preceq}$, equipped with a set of ordered operations, is proposed for the ordered conjunctive query $\boldsymbol{CQ}^{X}_{\preceq}$; transformation rules of ordered operators are provided and proven.

- The completeness theorem is proven for ordered conjunctive queries $\boldsymbol{CQ}^{X}_{\preceq}$, which states that a complete algebra exists to express any $\boldsymbol{CQ}^{X}_{\preceq}$ ordered conjunctive queries in a finite sequence of ordered operators.

Ordered conjunctive queries with *t*-decided order $\boldsymbol{CQ}^{T}_{\preceq}$

An ordered algebra was proposed for ordered conjunctive queries with ***t***-decided order and was proven to be complete with respect to the two-sorted first order logic.

- A formal definition of ordered conjunctive queries $\boldsymbol{CQ}^{T}_{\preceq}$ is provided in a two-sorted first-order calculus $\boldsymbol{FO}_{\preceq}$.
- An ordered algebra $\boldsymbol{CA}^{T}_{\preceq}$, equipped with a set of ordered operations, is proposed for the ordered conjunctive query $\boldsymbol{CQ}^{T}_{\preceq}$; transformation rules of ordered operators are provided and proven.
- The completeness theorem is proven for ordered conjunctive queries $\boldsymbol{CQ}^{T}_{\preceq}$, which states that a complete algebra exists to express any $\boldsymbol{CQ}^{T}_{\preceq}$ ordered conjunctive queries in a finite sequence of ordered operators.

Ordered conjunctive queries $\boldsymbol{CQ}_{\preceq}$

An ordered algebra was proposed for more general ordered conjunctive queries and was proven to be complete with respect to the two-sorted first-order logic.

- A formal definition of ordered conjunctive queries $\boldsymbol{CQ}_{\preceq}$ is provided in a two-sorted first-order calculus $\boldsymbol{FO}_{\preceq}$, which restricts $\boldsymbol{CQ}_{\preceq}$ ordered conjunctive queries to those satisfying the Separation Property.
- An ordered algebra $\boldsymbol{CA}_{\preceq}$, equipped with a set of ordered operations, is proposed for the ordered conjunctive query $\boldsymbol{CQ}_{\preceq}$; transformation rules of ordered operators are provided and proven.
- The completeness theorem is proven for the ordered conjunctive queries $\boldsymbol{CQ}_{\preceq}$, which states that a complete algebra exists to express any $\boldsymbol{CQ}_{\preceq}$ ordered conjunctive queries in a finite sequence of ordered operators.

6.2 Future Work

The work in this dissertation can be extended in many directions. Some of them are briefly mentioned here.

Ordered conjunctive queries without separation property
In the definition of the general ordered conjunctive query $\boldsymbol{CQ}_{\preceq}$in Chapter 5, $\boldsymbol{CQ}_{\preceq}$ order conjunctive queries are restricted to those satisfying the Separation Property. Then, a complete algebra was proposed and the completeness theorem is proven for this restricted ordered conjunctive queries. This work can be extended to the general class of ordered conjunctive queries, which is the ordered conjunctive queries with or without the Separation Property.

In particular, two questions need to be investigated for the general class of ordered conjunctive queries. First, an interesting question is whether the completeness theorem holds for the general class of ordered conjunctive queries: does there exist a complete algebra, equipped with a finite set of ordered operators, to express the general class of ordered conjunctive queries? Second, if such a complete ordered algebra exists, then what operators should be in this algebra? This ordered algebra should answer the query in Example **??**.

Ordered relational algebras
The techniques is this dissertation cannot be extended to $\boldsymbol{FO}_{\preceq}$ ordered relational queries in a straightforward manner because this technique cannot manage the negation in the data specification of an ordered relational query. Although complete algebras might not exist for the general $\boldsymbol{FO}_{\preceq}$ relational queries, many approaches remain to further investigate $\boldsymbol{FO}_{\preceq}$ ordered relational queries.

In Chapter 2, we have a proposition that complete algebras do not exist for general ordered relational queries. An interesting question is whether we can find any subset of ordered relational queries which has an expressively equivalent algebra.

Implementation of an ordered relational database
The work in this dissertation focuses on the conceptual ordered relational database model, and the query representation languages defined over it. However, this work sheds light on the possibility of implementing an ordered query engine to solve the order-relevant performance problem. Implementing such an ordered query engine would be an interesting direction to extend this work.

Ideally, a group of general ordered operators are implemented efficiently in this ordered query engine, which can express any first-order expressible ordered query. For given input relations, each ordered operator generate a relation with a specific order. Thus, sorting-based operations can be avoided at intermediate optimization steps if the existing orders match the desirable order properties. Furthermore, more optimization techniques can be developed to enhance the query processing in this ordered query engine. This work provides a direction in which the general query performance might be significantly improved by supporting relational query processing in an ordered fashion.

References

[1] XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999. http://www.w3.org/TR/xpath.

[2] XQuery 1.0: An XML Query Language, W3C Recommendation 23 January 2007. http://www.w3.org/TR/xquery/.

[3] XQuery 1.0 and XPath 2.0 Data Model (XDM), W3C Recommendation 23 January 2007. http://www.w3.org/TR/xpath-datamodel/.

[4] Serge Abiteboul, Laurent Herr, and Jan Van Den Bussche. Temporal versus First-Order Logic to Query Temporal Databases. In *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 49–57, 1996.

[5] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[6] Shurug Al-Khalifa, H. V. Jagadish, Nick Koudas, Jignesh M. Patel, Divesh Srivastava, and Yuqing Wu. Structural Joins: A Primitive for Efficient XML Query Pattern Matching. In *Proceedings of the 18th International Conference on Data Engineering*, pages 141–152, 2002.

[7] Mihnea Andrei, Xun Cheng, Sudipto Chowdhuri, Curtis Johnson, and Edwin Seputis. Ordering, Distinctness, Aggregation, Partitioning and DQP Optimization in Sybase ASE 15. In *Proceedings of the 2009 ACM International Conference on Management of Data*, pages 917–924, 2009.

[8] Catriel Beeri and Tova Milo. Schemas for Integration and Translation of Structured and Semi-structured Data. In *Proceedings of the 15th International Conference on Data Engineering*, pages 296–313, 1999.

[9] Catriel Beeri and Yariv Tzaban. SAL: An Algebra for Semistructured Data and XML. In *International Workshop on the Web and Databases*, pages 37–42, 1999.

[10] Nicole Bidoit, Sandra De Amo, and Luc Segoufin. Order Independent Temporal Properties. *Journal of Logic and Computation*, 14(2):277–298, 2004.

[11] Michael H. Böhlen, Christian S. Jensen, and Richard Thomas Snodgrass. Temporal Statement Modifiers. *ACM Transactions On Database Systems*, 25(4):407–456, 2000.

[12] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering*, pages 421–430, 2001.

[13] Nicolas Bruno, Luis Gravano, and Amelie Marian. Evaluating Top-k Queries over Web-Accessible Databases. In *Proceedings of the 18th International Conference on Data Engineering*, 2002.

[14] Nicolas Bruno, Nick Koudas, and Divesh Srivastava. Holistic Twig joins: Optimal XML Pattern Matching. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 310–321, 2002.

[15] Peter Buneman, Wenfei Fan, Jérôme Siméon, and Scott Weinstein. Constraints for Semistructured Data and XML. *ACM SIGMOD Record*, 30(1):47–55, 2001.

[16] Michael J. Carey and Donald Kossmann. On Saying "Enough Already!" in SQL. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 219–230, 1997.

[17] Upen S. Chakravarthy, John Grant, and Jack Minker. Logic-Based Approach to Semantic Query Optimization. *ACM Transactions on Database Systems (TODS)*, 15(2):162–207, 1990.

[18] Surajit Chaudhuri. An Overview of Query Optimization in Relational Systems. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 34–43, 1998.

[19] Surajit Chaudhuri and Luis Gravano. Evaluating Top-k Selection Queries. In *Proceedings of 25th International Conference on Very Large Data Bases*, pages 397–410, 1999.

[20] Surajit Chaudhuri and Moshe Y. Vardi. Optimization of Real Conjuctive Queries . In *Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 59–70, 1993.

[21] Mitch Cherniack and Stan Zdonik. Changing the Rules: Transformations for Rule-based Optimizers. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 61–72, 1998.

[22] Mitch Cherniack and Stanley B. Zdonik. Rule Languages and Internal Algebras for Rule-based Optimizers. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 401–412, 1996.

[23] Shu-Yao Chien, Zografoula Vagena, Donghui Zhang, Vassilis J. Tsotras, and Carlo Zaniolo. Efficient Structural Joins on Indexed XML Documents. In *Proceedings of 28th International Conference on Very Large Data Bases*, pages 263–274, 2002.

[24] Jan Chomicki. Preference Formulas in Relational Queries. *ACM Transactions on Database Systems*, 28(4):427–466, 2003.

[25] Jan Chomicki. Semantic Optimization of Preference Queries. *CDB*, pages 133–148, 2004.

[26] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. Skyline with Presorting. In *Proceedings of the 19th International Conference on Data Engineering*, pages 717–816, 2003.

[27] Jan Chomicki and David Toman. Temporal Databases. In *Handbook of Temporal Reasoning in Artificial Intelligence*, pages 429–467. 2005.

[28] Jens Claussen, Alfons Kemper, and Donald Kossmann. Order-Preserving Hash Joins: Sorting (Almost) For Free. Technical Report Technique Report MIP-9810, Fakultät für Mathematik und Informatik, Universität Passau, 1998.

[29] James Clifford, Albert Croker, and Alexander Tuzhilin. On Completeness of Historical Relational Query Languages. *ACM Transactions on Database Systems*, 19(1):64–116, 1994.

[30] Neil Coburn and Grant Weddell. A Logic for Rule-Based Query Optimization in Graph-Based Data Models. In *Proceedings of the 3rd International Conference on Deductive and Object-Oriented Databases (DOOD)*, volume 760 of *Lecture Notes in Computer Science*, pages 120–145. Springer-Verlag, 1993.

[31] Edgar F. Codd. Relational Completeness of Database Sublanguages. In R. Rustin, editor, *Database Systems*, pages 65–98. Prentice-Hall, 1972.

[32] Umeshwar Dayal. Of Nests and Trees: A Unified Approach to Processing Queries that Contain Nested Subqueries, Aggregates, and Quantifiers. In *Proceedings of 13th International Conference on Very Large Data Bases*, pages 197–208, 1987.

[33] David DeHaan, David Toman, Mariano P. Consens, and M. Tamer Özsu. A Comprehensive XQuery to SQL Translation using Dynamic Interval Encod-

ing. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 623–634, 2003.

[34] Jirun Dong and Richard Hull. Applying Approximate Order Dependency to Reduce Indexing Space. In *Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data*, pages 119–127, 1982.

[35] A. Ehrenfeucht. An Application of Games to the Completeness Problem for Formalized Theories. *Fundamenta Mathematicae*, 49:129–141, 1961.

[36] Maged El-Sayed, Katica Dimitrova, and Elke A. Rundensteiner. Efficiently Supporting Order in XML Query Processing. In *Proceedings of the 5th ACM International Workshop on Web Information and Data Management*, pages 147–154, 2003.

[37] Ronald Fagin. Combining Fuzzy Information from Multiple Systems. In *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 216–226, 1996.

[38] D. M. Gabbay. Expressive Functional Completeness in Tense Logic. In *Aspects of Philosophical Logic*, pages 91–117. 1981.

[39] Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal logic (vol. 1): Mathematical Foundations and Computational Aspects*. Oxford University Press, Inc., New York, NY, USA, 1994.

[40] Hector Garcia-Molina, Jeff Ullman, and Jennifer Widom. *Database System Implementation*. Prentice Hall, 2000.

[41] Seymour Ginsburg and Richard Hull. Order Dependency in the Relational Model. *Theoretical Computer Science*, 26:149–195, 1983.

[42] Seymour Ginsburg and Richard Hull. Sort Sets in the Relational Model. In *Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 332–339, 1983.

[43] Parke Godfrey, John Grant, Jarek Gryz, and Jack Minker. Integrity Constraints: Semantics and Applications. In *Logics for Databases and Information Systems*, pages 265–306, 1998.

[44] Goetz Graefe. Query Evaluation Techniques for Large Databases. *ACM Computing Surveys*, 25(2):73–170, 1993.

[45] Goetz Graefe and David J. DeWitt. The EXODUS Optimizer Generator. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pages 160–172, 1987.

[46] Ulrich Güntzer, Wolf-Tilo Balke, and Werner Kiessling. Optimizing Multi-Feature Queries for Image Databases. In *Proceedings of 26th International Conference on Very Large Data Bases*, pages 419–428, 2000.

[47] Laura Haas, Johann Freytag, Guy Lohman, and Hamid Pirahesh. Extensible Query Processing in Starburst. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 377–388, 1989.

[48] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. A Survey of Top-k Query Processing Techniques in Relational Database Systems. *ACM Computing Surveys*, 40(4), 2008.

[49] Ihab F. Ilyas, Rahul Shah, Walid G. Aref, Jeffrey Scott Vitter, and Ahmed K. Elmagarmid. Rank-Aware Query Optimization. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 203–214, 2004.

[50] Neil Immerman and Dexter Kozen. Definability with Bounded Number of Bounded Variables. *Information and Computation*, 83:121–139, 1989.

[51] Yannis E. Ioannidis and Raghu Ramakrishnan. Containment of Conjunctive Queries: Beyond Relations as Sets . *ACM Transactions on Database Systems*, 20(3):288–324, 1995.

[52] Hans Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

[53] Hans Kamp. Formal Properties of 'Now'. *Theoria*, 37:227–273, 1971.

[54] Werner Kie. Foundations of Preferences in Database Systems. In *Proceedings of 28th International Conference on Very Large Data Bases*, pages 311 – 322, 2002.

[55] Werner Kieand Gerhard Köstler. Preference SQL - Design, Implementation, Experiences. In *Proceedings of 28th International Conference on Very Large Data Bases*, pages 990–1001, 2002.

[56] Won Kim. On Optimizing an SQL-like Nested Query. *ACM Transactions on Database Systems*, 7(3):443–469, 1982.

[57] Alberto Lerner and Dennis Shasha. AQuery: Query Language for Ordered Data, Optimization Techniques, and Experiments. In *Proceedings of 29th International Conference on Very Large Data Bases*, pages 345–356, 2003.

[58] Chengkai Li, Kevin Chen-Chuan Chang, Ihab F. Ilyas, and Sumin Song. RankSQL: Query Algebra and Optimization for Relational Top-k Queries. In

Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pages 131–142, 2005.

[59] Leonid Libkin, Rona Machlin, and Limsoon Wong. A Query Language for Multidimensional Arrays: Design, Implementation, and Optimization Techniques. In *ACM SIGMOD Record*, pages 228–239, 1996.

[60] Guy M. Lohman. Grammar-like Functional Rules for Representing Query Optimization Alternatives. In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, pages 18–27, 1988.

[61] David Maier and Bennet Vance. A Call to Order. In *Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 1–16, 1993.

[62] Norman May, Sven Helmer, and Guido Moerkotte. Nested Queries and Quantifiers in an Ordered Context. In *Proceedings of 20th International Conference on Data Engineering*, pages 239–250, 2004.

[63] Surya Nepal and M.V. Ramakrishna. Query Processing Issues in Image (Multimedia) Databases . In *Proceedings of the 15th International Conference on Data Engineering*, pages 22–29, 1999.

[64] Thomas Neumann and Guido Moerkotte. A Combined Framework for Grouping and Order Optimization. In *Proceedings of 30th International Conference on Very Large Data Bases*, pages 960–971, 2004.

[65] Thomas Neumann and Guido Moerkotte. An Efficient Framework for Order Optimization. In *Proceedings of 20th International Conference on Data Engineering*, pages 461–472, 2004.

[66] Hamid Pirahesh, Joseph M. Hellerstein, and Waqar Hasan. Extensible Rule Based Query Rewrite Optimization in Starburst. In *Proceedings of the 1992 ACM SIGMOD international conference on Management of data*, pages 39–48, 1992.

[67] Raghu Ramakrishnan, Donko Donjerkovic, Arvind Ranganathan, Kevin S. Beyer, and Muralidhar Krishnaprasad. SRQL: Sorted Relational Query Language. In *Proceedings of International Conference on Statistical and Scientific Database Management*, pages 84–95, 1998.

[68] Darrell Raymond. *Partial Order Databases*. PhD thesis, Department of Computer Science, University of Waterloo, 1996.

[69] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access Path Selection in a Relational Database Management System.

In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, pages 23–34, 1979.

[70] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. Sequence Query Processing. In *ACM SIGMOD Record*, pages 430–441, 1994.

[71] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. SEQ: A Model for Sequence Databases. In *Proceedings of the 11th International Conference on Data Engineering*, pages 232–239, 1995.

[72] Jayavel Shanmugasundaram, Kristin Tufte, Chun Zhang, Gang He, David J. DeWitt, and Jeffrey F. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In *Proceedings of 25th International Conference on Very Large Data Bases*, pages 302–314, 1999.

[73] William M. Shui, Franky Lam, Damien K. Fisher, and Raymond K. Wong. Querying and Maintaining Ordered XML Data Using Relational Databases. In *Proceedings of the 16th Australasian Database Conference*, pages 85–94, 2005.

[74] David E. Simmen, Eugene J. Shekita, and Timothy Malkemus. Fundamental Techniques for Order Optimization. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 57–67, 1996.

[75] Giedrius Slivinskas, Christian S. Jensen, and Richard T. Snodgrass. Bringing Order to Query Optimization. *ACM SIGMOD Record*, 31(2):5–14, 2002.

[76] Giedrius Slivinskas, Christian S. Jensen, and Richard Thomas Snodgrass. A Foundation for Conventional and Temporal Query Optimization Addressing Duplicates and Ordering. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):21–49, 2001.

[77] Mohamed A. Soliman and Ihab F. Ilyas. Ranking with Uncertain Scores. In *Proceedings of the 25th International Conference on Data Engineering*, pages 317–328, 2009.

[78] Igor Tatarinov, Zachary G. Ives, Alon Y. Halevy, and Daniel S. Weld. Updating XML. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 413–424, 2001.

[79] Igor Tatarinov, Stratis Viglas, Kevin S. Beyer, Jayavel Shanmugasundaram, Eugene J. Shekita, and Chun Zhang. Storing and Querying Ordered XML Using a Relational Database System. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 204–215, 2002.

[80] David Toman. On Incompleteness of Multi-dimensional First-order Temporal Logics. In *IEEE International Symposium on Temporal Representation and Reasoning and International Conference on Temporal Logic*, pages 99–106, 2003.

[81] David Toman and Damian Niwinski. First-Order Queries over Temporal Databases Inexpressible in Temporal Logic. In *Proceedings of the 5th International Conference on Extending Database Technology*, volume 1057, pages 307–324, 1996.

[82] David Toman and Grant Weddell. On Attributes, Roles, and Dependencies in Description Logics and the Ackermann Case of the Decision Problem. In *Working Notes of the 2001 International Description Logics Workshop*, pages 76–85, 2001.

[83] Alexander Tuzhilin and James Clifford. A Temporal Relational Algebra as Basis for Temporal Relational Completeness. In *Proceedings of 16th International Conference on Very Large Data Bases*, pages 13–23, 1990.

[84] Zografoula Vagena, Nick Koudas, Divesh Srivastava, and Vassilis J. Tsotras. Answering Order-Based Queries Over XML Data. In *WWW '05: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, pages 1162–1163, 2005.

[85] Xiaoyu Wang and Mitch Cherniack. Avoiding Sorting and Grouping in Processing Queries. In *Proceedings of 29th International Conference on Very Large Data Bases*, 2003.

[86] Grant E. Weddell. Reasoning about Functional Dependencies Generalized for Semantic Data Models. *ACM Transactions on Database Systems*, 17(1):32–64, 1992.

[87] Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura, and Shunsuke Uemura. XRel: A Path-based Approach to Storage and Retrieval of XML Documents Using Relational Databases. In *ACM Transactions on Internet Technology*, volume 1, pages 110–141, 2001.

[88] Chun Zhang, Jeffrey F. Naughton, David J. DeWitt, Qiong Luo, and Guy M. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 425–436, 2001.