Check 58. Star Schema Design for Concept Hierarchy in Attribute Oriented Induction.pdf

by Arta Sundjaja

Submission date: 02-May-2019 04:01PM (UTC+0700)

Submission ID: 1102511214

File name: Design_for_Concept_Hierarchy_in_Attribute_Oriented_Induction.pdf (658.73K)

Word count: 5286

Character count: 27910

Star Schema Design for Concept Hierarchy in Attribute Oriented Induction

Spits Warnars
Manchester Metropolitan University, United Kingdom

Abstract—The Concept Hierarchy in attribute oriented induction is a powerful tool for saving the knowledge hierarchy in data, which will be then used to generalize mining rules for data mining. Database design influences the performance applications when reading records in database. When building the database table the allocation of fields will be decided by the lowest and the highest concept within the concept tree. Additionally, the number of hierarchy concept levels in a concept tree will decide the quantity of fields in the table. The number of record tables will be decided by the number of the lowest concepts in concept tree. For numeric values the treatment is different. For efficiency reasons the number of created table records will instead depend on the amount of concepts at the next generalization. All the tables from concept trees will become dimensional tables and the data table will become a fact table. All these tables in fact create a star schema. The Star schema is recognized as data warehouse schema for multidimensional analysis, and will give value add for attribute oriented induction for multidimensional purposes.

Index Terms—Data Mining, concept hierarchy, attribute oriented induction, rule, database.

I. INTRODUCTION

THE attribute oriented induction method integrates a machine learning paradigm – especially learning-from-examples techniques [3] – with database operations, extracts generalized rules from an interesting set of data and discovers high level data regularities [13]. The attribute oriented induction method has been implemented in a data mining system prototype called DBMINER [16,17] which was previously called *DBLearn* [12,14], and has been tested successfully against a large relational database.

The attribute oriented induction approach was developed for learning different kinds of knowledge rules such as characteristic rules, discrimination or classification rules, quantitative rules, data evolution regularities [15], qualitative rules [9], association rules and cluster description rules [13]. Attribute oriented induction has the concept hierarchy as an advantage, where a concept hierarchy as background knowledge can be provided by knowledge engineers or

Manuscript received March 17, 2010.
Spits Warnars is a doctoral student at the Department of Computing and Mathematics, Manchester Metropolitan University, John Dalton Building, Chester Street, Manchester M15GD United Kingdom. (e-mail: s.warnars@mmu.ac.uk).

domain experts [8,10,13]. Concepts are ordered in a concept hierarchy by levels from specific or low level concepts into general or higher level concepts. Generalization is achieved by ascending to the next higher level concepts along the paths of concept hierarchy [11]. The most general concept is the null description as the most specific concepts correspond to the specific values of attributes in the database described as ANY. The concept hierarchy can be balanced or unbalanced, but an unbalanced hierarchy must then be converted to a balanced one.

There are 3 Types of concept generalization on concept hierarchy [6] and in [5] number (2)-(4) as 3 types of rule based concept hierarchy and number (1) as a non rule based concept hierarchy. These are :

1) *Unconditional concept generalization*: the rule associated with the unconditional IS-A type rules. A concept is generalized to a higher level concept because of the *subsumption* relationship indicated in the concept hierarchy.

2) *Conditional/deductive rule generalization*: the rule associated with a generalization path as a deduction rule where the type of rules is conditional and can only be applied to generalize a concept if the corresponding condition can be satisfied. For example, form : A(x) Λ B(x) → C(x) has the meaning that for a tuple x, the concept(attribute value) A can be generalized to concept C if condition B can be satisfied by x. Or concept C can be generalized if it can be satisfied by concept A and B.

3) *Computational rule generalization*: each rule is represented by a condition which is value-based and can be evaluated against an attribute or a tuple or the database by performing some computation. The truth value of the condition would then determine whether a concept can be generalized via the path.

4) *Hybrid rule-based concept generalization*: a hierarchy can have path associated with all the above 3 different types of rules. It has a powerful representation capability and is suitable for many kinds of application.

An implementation needs a database for storing records, while the application needs data mining tools for the user to read records from database and get the rules in for presentation. The database can be implemented using any kind of database management system, such as Microsoft Access, SQL Server, Oracle, MySQL and many other database management systems. The application can be implemented using any kind of language software, such as Java, Active

Server Pages, Visual Basic and others. The choice of the database and application will depend on a number of factors, including on the organizational rules, monetary resources, and the technology.

In considering databases, one important factor is the performance of the database system since it influences the performance of the application as a whole. A good database design will increase the application performance, while a bad database design will reduce the application performance. An additional factor is whether the database is *normalized*. On one hand the normalization of a database provides for the best performance and is useful for transactional processing systems with input, edit and delete operations. On the other hand, an unnormalized database will provide the best approach for online analytical processing where records need only to be read. Moreover, the number of SQL join operations will also influence performance as a whole. Typically, more join operations will need to be made in a normalized database compared to an unnormalized database. The expert knowledge in the area of database design contributes towards obtaining the best design and performance. Additionally, a distributed database can also be used for handling large distributed concept hierarchies.

In this paper we transform the concept hierarchy into tables resulting in a new database design. The concept tree will be used as data input and the rules will be built based on the queries [23,26,28].

Using queries to build rules presents an efficient mechanism for understanding the mined rules ([22,25]). The use of the threshold as a control over the maximum number of tuples (within the target class in the final generalized relation) will no longer be needed. Instead, group by operator in the SQL Select statement will limit the final result generalization.

Setting different thresholds will generate different generalized tuples (needed of global picture of induction). Doing this repeatedly is time-consuming and tedious work [27]. Instead, all the interesting generalized tuples as a multiple rule can be generated for the global picture of induction by using group by operator in the SQL Select statement.

In converting the concept hierarchy into a database table and in order to obtain the best database design, there are a number of questions that need to be answered. These questions include how to perform the conversion (from the concept hierarchy into a table), how many tables will be created and based on which criteria. By answering these questions, we hope that a standard method will emerge for the conversion of concept hierarchies into good database designs. In the first instance the first step will be to explore how to convert the non-rule based concept hierarchy (as an unconditional concept hierarchy). Future work will address the conversion of from rule based concept hierarchy.

In this paper we discuss only the conversion of the non-rule based concept hierarchy into tables in the star schema design. The implementation of the current and proposed attribute oriented induction, including the differentiation between them, can be found in [29].

II. CONVERTING FROM CONCEPT HIERARCHY INTO TABLE

In order to aid the discussion and provide a link between the current work with our previous research, we will use the concept hierarchy based on data examples found in [2,8]. Figure 1 shows an example of a concept hierarchy from university database.
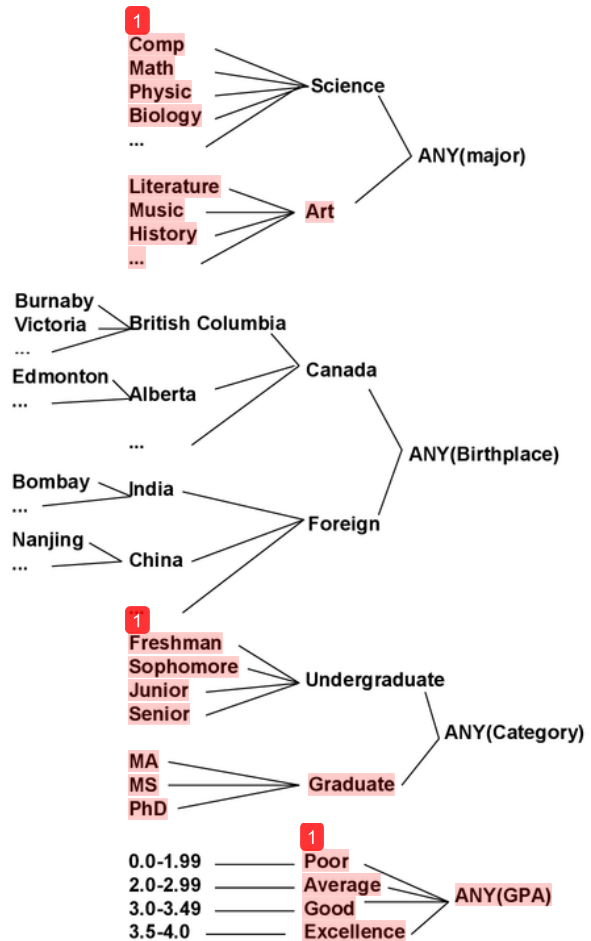

Fig. 1. Example of a Concept hierarchy

Based on concept hierarchy in Figure 1 we will build concept tree that represents a taxonomy of concepts of the values in an attribute domain. The concept tree will be built based on the most general concept which is described as ANY in concept hierarchy. Based on Figure 1 there are four (4) most general concepts with ANY, resulting in concept trees [24]. These are:

1) (science, art)　　　　　　　　　⊂ ANY(major)
2) (graduate, undergraduate)　　　⊂ ANY(category)
3) (Foreign, Canada)　　　　　　　⊂ ANY(birthplace)
4) (poor, average, good, excellence)　⊂ ANY(GPA)

The dotted-line symbol (…) in Figure 1 shows the

possibility of other concepts. For example, there is the possibility to extend the width of concept tree by extending the "Major" beyond "Science" and "Art". Similarly, the "Birthplace" is not limited to just only "Canada" and "Foreign".

For the database implementation all the concept trees will be implemented as tables in the database. The concept tree for "Major" will be implemented as the table entitled *hierarchy_major* (see Table 1 and its explanation in Figure 2). The lowest level concept tree for "Major" in Figure 2 like comp, math, physics, biology, literature, music and history become the first field name Major and has varchar data type with length 15. The next and the last level concept tree for major in figure 2 like science and art become the next field name StudyProg and have varchar data type with length 15.

TABLE I
HIERARCHY_MAJOR TABLE

Field Name	Type
Major	varchar(15)
StudyProg	varchar(15)

Fig. 2. The transformation of the concept tree for Major into *hierarchy_major* table

For example, if there are 10 lowest level concepts in the concept tree "Major" (Figure 2) then 10 records or tuples will be created in the table *hierarchy_major* based on field "Major" as the first field in that table. Each of the records will fill the next field *studyprog* based on generalization the lowest level concept tree major. For example for the record where the major field is "Computing", its next field *studyprog* will be filled with "Science" because the computing as the lowest level concept tree major in Figure 2 has a generalization into Science. Table 2 is the result from the data from the concept tree "Major" in Figure 2.

TABLE II
RECORDS FOR HIERARCHY_MAJOR TABLE

Major	StudyProg
Computing	Science
Math	Science
Biology	Science
Chemistry	Science
Statistics	Science
Physics	Science
Music	Art
History	Art
Literal Arts	Art
Literature	Art

As another example, the concept tree for "Category" will be implemented as table *hierarchy_cat* (see Table 3 and Figure 3). The lowest level concept tree for "Major" in Figure 3 such as "Freshman", "Sophomore", "Junior", "Senior", "MS", "MA" and "PhD" become the first field named "Category" and has *varchar* data type with length 15. The last level concept tree for "Category" in Figure 3, such as "Undergraduate" and "Graduate" becomes the next field name "Study" and has *varchar* data type with length 15.

TABLE III
HIERARCHY_CAT TABLE

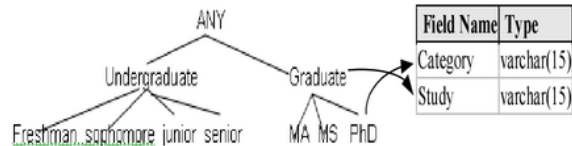| Field Name	Type
Category	varchar(15)
Study	varchar(15)

Fig. 3. The transformation of the concept tree for Category into *hierarchy_cat* table

In Figure 3 there are seven (7) lowest level concepts from concept tree "Category", which will create seven (7) records or tuples in table *hierarchy_cat* based on field Category as the first field in the table. The records filling next field "Study" will be based on the generalization the lowest level concept tree "Category". For example the record where the category field contains "Freshman" will fill the next field ("Study") with "Undergraduate" because the concept "Freshman" as the lowest level concept tree (in "Category" in Figure 3) has a generalization into "Undergraduate". Table 4 shows this based on the data from concept tree "Category" in Figure 3.

TABLE IV
RECORDS FOR HIERARCHY_CAT TABLE

Category	Study
Freshman	undergraduate
Sophomore	undergraduate
Junior	undergraduate
Senior	undergraduate
MS	graduate
MA	graduate
PhD	graduate

The concept tree for "Birthplace" will be implemented as the table *hierarchy_birth* (see Table 5 and Figure 4). The lowest level concept tree for birthplace in Figure 4 (such as Burnaby, Victoria, Edmonton, Bombay and Nanjing) will become the first field name "Birthplace" and has a *varchar* data type with length 15. The next level concept tree for birthplace in Figure 4 (namely British Columbia, Alberta, India and China) will become the next field name "City" and

has *varchar* data type with length 20. Note that this tree is different from the previous concept due to the fact that it has three (3) levels. As such, the next and the last level concept tree for birthplace in Figure 4 (namely Canada and Foreign) will become the next field name "Country" and has a *varchar* data type with length 10. Thus, the number of hierarchy levels will determine the number of the fields that will be created. In the case of the previous concept tree, because there are two (2) hierarchy levels, therefore two fields will be created for each concept tree. Similarly, since the concept tree "Birthplace" has three (3) hierarchy levels, therefore three fields will be created in the table.

TABLE V
HIERARCHY_BIRTH TABLE

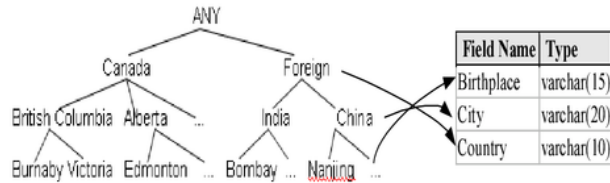| Field Name	Type
Birthplace	varchar(15)
City	varchar(20)
Country	varchar(10)

Fig. 4. The transformation of the concept tree for Birthplace into *hierarchy_birth* table

Following the previous example, there are eleven (11) lowest level concepts in the concept tree "Birthplace" as shown in Figure 4. This will create eleven records or tuples in table *hierarchy_birth* based on field "Birthplace" as the first field in the table. In each record the next field entry will be based on the generalization on concept tree "Birthplace".

TABLE VI
RECORDS FOR HIERARCHY_BIRTH TABLE

Birthplace	City	Country
Bombay	India	Foreign
Burnaby	British Columbia	Canada
Calgary	Alberta	Canada
Edmonton	Alberta	Canada
Nanjing	China	Foreign
Ottawa	Ontario	Canada
Richmond	British Columbia	Canada
Shanghai	China	Foreign
Toronto	Ontario	Canada
Vancouver	British Columbia	Canada
Victoria	British Columbia	Canada

For example the record where the birthplace field contains the value "Bombay" will have the next field ("City") filled with the value "India" because the concept "Bombay" as the lowest level in the concept tree "Birthplace" (see Figure 4) has a generalization into the India concept. Furthermore, the next

field "Country" (which is the last field) with have the value "Foreign" because the concept "India" has a generalization into "Foreign" concept. Table 6 shows the resulting data from concept tree "Birthplace" as shown in Figure 4.

The concept tree "GPA" will be implemented as table *hierarchy_gpa* (see Table 7 and Figure 5). Different to the previous concept trees, here there are a large range data for hierarchy levels. For example, the generalization for concept "Poor" comes from the value range between 0 and 1.99, and thus there will be 199 values possible (covering from 0.00 to 1.99). For efficiency reasons we just record the first value and last value in the range for each of hierarchy levels. As result, we will add one field containing the last value implying that "The number of hierarchy levels will decide the number of fields will be created". We use this notation here because although the concept tree "GPA" (Figure 5) has only two levels (and thereby creating two fields in the database), adding 199 records to each field will result in a table with a total of 400 records. Thus for efficiency reasons we treat concept trees with numeric values differently [11,21,19,20]. Therefore, here we have created only 3 fields with 4 records.

In the case of the GPA, the first field is *GPA_start* and will have a range a values using the float(3,2) data type. Next is field name *GPA_fin* that will also have a range a values using the float(3,2) data type. The last field name "Range" will be the same as the other concept trees where it has "the highest level before the most general concept ANY is the last field". The field range has *varchar* data type with length 15.

TABLE VII
HIERARCHY_GPA TABLE

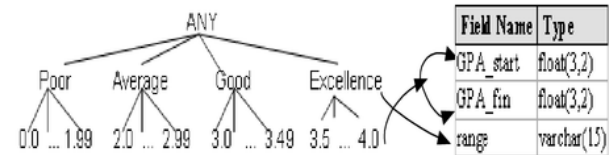| Field Name	Type
GPA_start	float(3,2)
GPA_fin	float(3,2)
range	varchar(15)

Fig. 5. The transformation of the concept tree for GPA into *hierarchy_gpa* table

Because GPA concept tree in Figure 5 handles numeric values, then the number of created records will not depend on the number of concepts at the lowest level in concept tree. Instead for efficiency it will depend on the number of concepts at the next generalization. Now at the next generalization (after the lowest level concept) there are four (4) concepts, namely "Poor", "Average", "Good", and "Excellent". Different to the other cases, the handling of numeric values in the concept tree will be via specialization. For example the first data at the level after the first low level is "Poor" and we assign the *GPA_start* value of 0 and the *GPA_fin* with the

value of 1.99. We do this also for the other fields. The result is shown in Table 8 using the concept tree in Figure 5.

TABLE VIII
RECORDS FOR HIERARCHY_GPA TABLE

GPA_start	GPA_fin	range
0.0	1.99	Poor
2.0	2.99	Average
3.0	3.49	Good
3.5	4.0	Excellent

Based on the previous explanation, below are the summary of the assumptions that we have used to convert a concept hierarchy into a table:

1) The lowest level concept tree maps to the first field and the highest level (before the most general concept ANY) is mapped to the last field.
2) The number of hierarchy levels in a concept tree will decide the number of fields in the table, with the exception of numeric values for efficiency reasons.
3) The number of concepts at the lowest level in a concept tree will become the number of records or tuples in the table, with the exception of numeric values for efficiency reasons.
4) For the efficiency handling numeric values in a concept tree, the number of created table records will not depend on the number of concepts at the lowest level in concept tree, but instead it will depend on the number of concepts at the next generalization.

III. LOGICAL DATA MODEL

Based on data examples found in [2,8], Table 9 shows the corresponding student table.

TABLE IX
STUDENT TABLE

Name	Category	Major	Birthplace	GPA
Anton	M.A.	History	Vancouver	3.5
Andi	Junior	Math	Calgary	3.7
Amin	Junior	Liberal arts	Edmonton	2.6
Anil	M.S.	Physics	Ottawa	3.9
Ayin	Ph.D.	Math	Bombay	3.3
Amir	Sophomore	Chemistry	Richmond	2.7
Acai	Senior	Computing	Victoria	3.5
Abdi	Ph.D.	Biology	Shanghai	3.4
Afun	Sophomore	Music	Burnaby	3.0
Agung	Ph.D.	Computing	Victoria	3.8
Ahing	M.S.	Statistics	Nanjing	3.2
Akuan	Freshman	literature	Toronto	3.9

All the tables that represent a concept hierarchy can be connected to the student table. Figure 6 shows a class diagram with the connectivity between tables. In the context of the Data Warehouse concept, Figure 6 shows a *star schema*, where the student table is viewed as a *fact table* and the other tables from the concept tree are viewed as *a dimensional table*. As a result, the multi dimensional concept in Data Warehouse can be applied here, where the data can be roll up and drill down and the data can be viewed in multiple dimensions with

concept slice, dice and pivot [4, 18]. The aggregate count function and group-by-operator in the SQL Select statement can represent the roll up process [1, 7].
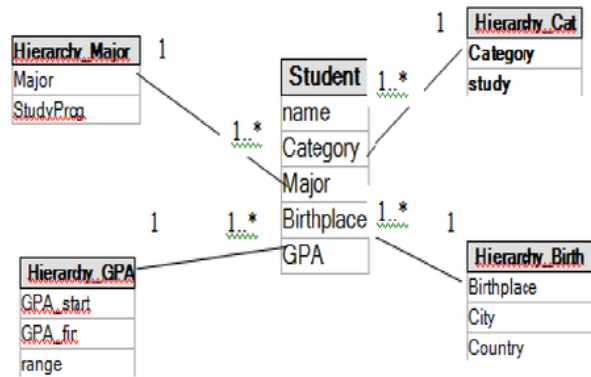

Fig. 6. A Logical data model

To improve the performance the logical data model in Figure 6, it can be changed to become one (1) table (as only one fact table) as shown in Table 10. The data as the representative all the data in Figure 6 is shown in Table 11 and 12. The performance can be improved by using *unnormalized* tables. This is because using one table is faster rather than using 5 tables through the decrease in the number of needed Join operations. However, the limitation of using only one fact table is that it poorly represents the concept hierarchy. For efficiency needs these two options can be used together, where queries will be directed at the single fact table and the concept hierarchy is used as the basis for the logical data model as exemplified by Figure 6.

TABLE X
ONE FACT TABLE

Student
Name
Category
Study
Major
Studyprog
Birthplace
City
Country
GPA
Range

TABLE XI
DATA IN ONE FACT TABLE

Name	Category	Study	Major	StudyProg
Anton	MA	graduate	History	art
Andi	Junior	undergraduate	Math	science
Amin	Junior	undergraduate	Literal Arts	art
Anil	MS	graduate	Physics	science
Ayin	PhD	graduate	Math	science
Amir	Sophomore	undergraduate	Chemistry	science
Acai	Senior	undergraduate	Computing	science

Abdi	PhD	graduate	Biology	science
Afun	Sophomore	undergraduate	Music	art
Agung	PhD	graduate	Computing	science
Ahing	MS	graduate	Statistics	science
Akuan	Freshman	undergraduate	Literature	art

TABLE XII
CONTINUE DATA IN ONE FACT TABLE

Birthplace	City	Country	GPA	Range
Vancouver	British Columbia	Canada	3.5	Excellent
Calgary	Alberta	Canada	3.7	Excellent
Edmonton	Alberta	Canada	2.6	Average
Ottawa	Ontario	Canada	3.9	Excellent
Bombay	India	Foreign	3.3	Good
Richmond	British Columbia	Canada	2.7	Average
Victoria	British Columbia	Canada	3.5	Excellent
Shanghai	China	Foreign	3.4	Good
Burnaby	British Columbia	Canada	3	Good
Victoria	British Columbia	Canada	3.8	Excellent
Nanjing	China	Foreign	3.2	Good
Toronto	Ontario	Canada	3.9	Excellent

Another option for efficiency is by the appropriate coding of the student data which can provide storage space savings, better query performance and easier data management. The following is an example of assigning codes to the field entries:

1) Coding for entries in the field *studyprog*: science→ 01, art → 02.
2) Coding for entries in the field *major*: computing→01, Math→02, Biology→03, Chemistry→04, statistics→05, physics→06, Music→07, History→08, Literal arts→09, Literature →10.
3) Coding for the field *Study*: undergraduate→01, graduate→02.
4) Coding for the field *Category*: Freshman→01, Sophomore→02, Junior→03, Senior→04, MS→05, MA→06, PhD→07.
5) Coding for the field *Birthplace*: Bombay→01, Burnaby→02, Calgary→03, Edmonton→04, Nanjing→05, Ottawa→06, Richmond→07, Shanghai→08, Toronto→09, Vancouver→10, Victoria→11.
6) Coding for the field *City*: India→01, British Columbia→02, Alberta→03, China→04, Ontario→05.
7) Coding for the field *Country*: Canada→01, foreign→02.

Note that the table representing *hierarchy_gpa* cannot be coded because of its numeric data. Table 13 shows the coding result for *hierarchy_birth* table, Table 14 for the *hierarchy_cat* table, Table for the *student* table, and Table 16 shows the coding result for *hierarchy_major* table.

TABLE XIII
CODING FOR HIERARCHY_BIRTH TABLE

BirthID	Birthplace	City	Country
010102	Bombay	India	Foreign
020201	Burnaby	British Columbia	Canada
030301	Calgary	Alberta	Canada
040301	Edmonton	Alberta	Canada
050402	Nanjing	China	Foreign
060501	Ottawa	Ontario	Canada
070201	Richmond	British Columbia	Canada
080402	Shanghai	China	Foreign
090501	Toronto	Ontario	Canada
100201	Vancouver	British Columbia	Canada
110201	Victoria	British Columbia	Canada

TABLE XIV
CODING FOR HIERARCHY_CAT TABLE

CatID	Category	Study
0101	Freshman	undergraduate
0102	Sophomore	undergraduate
0103	Junior	undergraduate
0104	Senior	undergraduate
0205	MS	graduate
0206	MA	graduate
0207	PhD	graduate

TABLE XV
CODING FOR STUDENT TABLE

name	Category	Major	Birthplace
Anton	0206	0208	100201
Andi	0103	0102	030301
Amin	0103	0209	040301
Anil	0205	0106	060501
Ayin	0207	0102	010102
Amir	0102	0104	070201
Acai	0101	0101	110201
Abdi	0207	0103	080402
Afun	0102	0207	020201
Agung	0207	0101	110201
Ahing	0205	0105	050402
Akuan	0101	0210	090501

TABLE XVI
CODING FOR HIERARCHY_MAJOR TABLE

MajorID	Major	StudyProg
0101	Computing	Science
0102	Math	Science
0103	Biology	Science
0104	Chemistry	Science
0105	Statistics	Science
0106	Physics	Science
0207	Music	Art
0208	History	Art
0209	Literal Arts	Art
0210	Literature	Art

IV. CONCLUSION

In order to convert a non-rule based concept hierarchy, there are some guidance which can help in the conversion. First, the lowest level concept is mapped to the first field and the highest level before the most general concept ANY is mapped to the last field. Secondly, the number of hierarchy levels will determine the number of fields except for numeric value (for efficiency reasons). Thirdly, the number of concepts at the lowest level in concept tree will become the number of records or tuples in the table (again with the exception of numeric values for efficiency reasons). Finally, for the efficiency handling numeric values in a concept tree, the number of created table records will not depend on the number of concepts at the lowest level in concept tree, but instead on the number of concepts at the next generalization. From an implementation perspective, there is the option of using one fact table or several combined tables, and there is also the option of using coding to improve efficiency and performance.

REFERENCES

[1] Alves, R. and Belo, O. (2007), 'Multidimensional Data Mining', SDDI'07 – Doctoral Symposium.

[2] Cai, Y. (1989), 'Attribute-oriented induction in relational databases', Master thesis, Simon Fraser University.

[3] Cai, Y., Cercone, N. and Han, J. (2007),'Learning in relational databases: an attribute-oriented approach', Computational Intelligence, 7(3),119-132.

[4] Chen,M., Han, J. and Yu, P.S. (1996), 'Data Mining: An overview from database perspective', IEEE Trans. Knowledge and Data Eng, 8(6),866-883.

[5] Cheung, D.W., Fu, A.W. and Han, J. (1994), 'Knowledge discovery in databases: A rule-based attribute-oriented approach', in Proceedings of Intl Symp on Methodologies for Intelligent Systems, Charlotte, North Carolina,164 -173.

[6] Cheung, D.W., Hwang, H.Y., Fu, A.W. and Han, J. (2000), 'Efficient rule-based attribute-oriented induction for data mining', Journal of Intelligent Information Systems, 15(2), 175-200.

[7] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D. and Venkatrao, M. (1997), 'Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals', Data Mining and Knowledge Discovery, 1, 29-53.

[8] Han, J., Cai, Y. and Cercone, N. (1992), 'Knowledge discovery in databases: An attribute-oriented approach', in Proceedings 18th International Conference Very Large Data Bases, Vancouver, British Columbia, 547-559.

[9] Han,J., Cai, Y., and Cercone, N. (1993), 'Data-driven discovery of quantitative rules in relational databases', IEEE Trans on Knowl and Data Engin, 5(1), 29-40.

[10] Han,J. (1994), 'Towards efficient induction mechanisms in database systems', Theoretical Computer Science, 133(2), 361-385.

[11] Han, J. and Fu, Y.(1994),'Dynamic Generation and Refinement of Concept Hierarchies for Knowledge Discovery in Databases', in Proceedings of AAAI Workshop on Knowledge Discovery in Databases, 157-168.

[12] Han, J., Fu, Y., Huang, Y., Cai, Y., and Cercone, N. (1994), 'DBLearn: a system prototype for knowledge discovery in relational databases', ACM SIGMOD Record, 23(2), 516.

[13] Han,J. and Fu, Y. (1995), 'Exploration of the power of attribute-oriented induction in data mining', in U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, eds. Advances in Knowledge Discovery and Data Mining, 399-421, AAAI/MIT Press.

[14] Han, J., Fu, Y., and Tang, S. (1995), 'Advances of the DBLearn system for knowledge discovery in large databases', in Proceedings of the 14th international Joint Conference on Artificial intelligence, 2049-2050.

[15] Han, J., Cai, Y., Cercone, N. and Huang, Y. (1995), 'Discovery of Data Evolution Regularities in Large Databases', Journal of Computer and Software Engineering, 3(1), 41-69.

[16] Han,J., Fu, Y.,Wang, W., Chiang, J., Gong, W., Koperski, K., Li,D., Lu, Y., Rajan,A., Stefanovic,N., Xia,B. and Zaiane,O.R. (1996), 'DBMiner:A system for mining knowledge in large relational databases', in Proceedings Int'l Conf. on Data Mining and Knowledge Discovery, 250-255.

[17] Han, J., Chiang, J. Y., Chee, S., Chen, J., Chen, Q., Cheng, S., Gong, W., Kamber, M., Koperski, K., Liu, G., Lu, Y., Stefanovic, N., Winstone, L., Xia, B. B., Zaiane, O. R., Zhang, S., and Zhu, H. (1997), 'DBMiner: a system for data mining in relational databases and data warehouses', In Proceedings of the 1997 Conference of the Centre For Advanced Studies on Collaborative Research, 8.

[18] Han,J., Lakshmanan, L.V.S. and Ng, R.T. (1999), 'Constraint-based, multidimensional data mining', IEEE Computer, 32(8), 46-50.

[19] Hsu, C. (2004),'Extending attribute-oriented induction algorithm for major values and numeric values', Expert Systems with Applications, 27, 187-202.

[20] Hu, X. (2003), 'DB-HReduction: A Data Preprocessing Algorithm for Data Mining Applications', Applied Mathematics Letters,16(6),889-895.

[21] Huang, Y. and Lin, S. (1996), 'An Efficient Inductive Learning Method for Object-Oriented Database Using Attribute Entropy', IEEE Transactions on Knowledge and Data Engineering, 8(6),946-951

[22] Imielinski, T. and Virmani, A. (1999), 'MSQL: A Query Language for Database Mining', in Proceedings of Data Mining and Knowledge Discovery, 3, 373-408

[23] Meo, R., Psaila,G. and Ceri,S. (1998),'An Extension to SQL for Mining Association Rules', in Proceedings of Data Mining and Knowledge Discovery,2,195-224.

[24] Muyeba,M.K. and Keane,J.A. (1999), 'Extending attribute-oriented induction as a key-preserving data mining method', in Proceedings 3rd European Conference on Principles of Data Mining and Knowledge Discovery, Lecture Notes in Computer science, 1704, 448-455.

[25] Muyeba, M. (2005),'On Post-Rule Mining of Inductive Rules using a Query Operator', in Proceedings of Artificial Intelligence and Soft Computing.

[26] Muyeba, M. and Marnadapali, R. (2005),'A framework for Post-Rule Mining of Distributed Rules Bases', in Proceeding of Intelligent Systems and Control.

[27] Wu, Y., Chen, Y. and Chang, R. (2009), 'Generalized Knowledge Discovery from Relational Databases', International Journal of Computer Science and Network, 9(6),148-153.

[28] Zaiane, O.R. (2001), 'Building Virtual Web Views', Data and Knowledge Engineering, 39, 143-163.

[29] Warnars, S. (2010),'Attribute oriented induction with star schema',International Journal of Database Management Systems (IJDMS),2(2),20-42.

Spits Warnars received his bachelor's degree in Information System from the University of Budi Luhur, Indonesia in 1995. He finished a master degree in Information Technology from university of Indonesia, in January 2007. Since 1995 he has been a full time lecturer at the Information Technology faculty, at University of Budi Luhur, in Indonesia. He has also been an auxiliary lecturer since January 2008 at the Bina Nusantara University in Indonesia. Spits Warnars has been completing his PhD degree in computer science at Manchester Metropolitan University, UK, since Sept 2008. His area of interest is data mining. He has published 3 papers in international journals, 6 papers at international conferences, 10 papers in Indonesian journals, and 7 papers at national conferences in Indonesia. He has been an ACM student member since 2008. He has been a reviewer for the World Scientific and Engineering Academy and Society since 2006. Since July 2010 he has been on the editorial board of the Journal of Computing and Applications (JCA), International Association for Computer Scientists and Engineers (IACSE) and Journal of Global Research in Computer Science (JGRCS).

Check 58. Star Schema Design for Concept Hierarchy in Attribute Oriented Induction.pdf