International Institute for Applied Systems Analysis
IIASA www.iiasa.ac.at

On Integrity Constraints for a Waste Management Information System

Schreiber, D.

IIASA Working Paper

March 1994

Working Paper

On Integrity Constraints for a Waste Management Information System

Dirk Schreiber

WP-94-16
March 1994

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

On Integrity Constraints for a Waste Management Information System

Dirk Schreiber

WP-94-16
March 1994

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Table of Contents

		Page
1	Introduction	1
2	Case Study: Development of a database system supporting a waste management system for Lower Saxony	3
3	Data collection on quantities of different waste types as a waste management problem area (information requirement analysis/problem description)	5
4	An extended ER-scheme for waste management tasks (Conceptual level)	6
5	Logical design and relational implementation of the extended ER scheme, including the support for referential and semantic integrity constraints	8
5.1	A relational scheme for waste management tasks	9
5.2	Definition of triggers enforcing referential integrity constraints	10
5.3	Definition of triggers enforcing the aggregation hierarchy constraint	13
6	Conclusions	16
	Acknowledgements	17
	References	17

On Integrity Constraints for a Waste Management Information System

*Dirk Schreiber**

There is a *waste problem* in nearly every country. A model of a waste generating system and an efficient waste management information system are the first steps to control this problem. Some countries have already enacted laws which force communities and enterprises to report annually the amounts of wastes produced. For example, the German federal state, Lower Saxony, enacted such a law in 1992. This YSSP-Project deals with a case study on the development of a waste management information system for this state. The quality of the system essentially depends on the consistency of the underlying database system. Therefore, the point of view of a database designer is given. The design of the data structures and the support of integrity constraints in the underlying database system is thereby especially emphasized. The data structures are modelled using an extended entity relationship model. They are implemented with a relational database management system. In contrast to the traditional way of supporting integrity constraints in the application program, we define triggers which are implemented in the database system itself to enforce main consistency rules. In the final chapter some conclusions about further steps are given.

1 Introduction

There is a *waste problem* in nearly every country. Especially the industrialized countries have to control the following aspects:

First of all, the amounts of waste have increased in absolute numbers. This is caused, for instance, by a general tendency towards excess packaging. Secondly, the volumetric weight of waste has decreased, because the composition of waste has changed. Both developments result in a growing demand for disposal capacities. But, in fact, these capacities are decreasing. Furthermore, reusable materials (e.g. glass, paper, etc.) are mixed with other materials on deponies. Therefore, an effective reuse of these materials is often not possible. For example: The estimated value of disposed reusable material is about 1300 million DM in Germany in 1990 [KLEINALTEN...90].

A model of a waste generating system is necessary to deal with these problems. It could be interpreted as a material flux system with two main producing processes, the production of goods (e.g. by industries) and the consumption of goods by the households. A third process contains waste management activities (treatment, disposal of waste). Its interactions with the other processes are shown in figure 1. They model recycling and reuse activities.

*Member of the Young Scientists Summer Program 1993 at IIASA. Home Institute: University of Siegen, Faculty of Economics, Information and Decision Sciences Department, Hölderlinstr.3, D-57068 Siegen, Germany. Email: dirk@fb5.uni-siegen.de

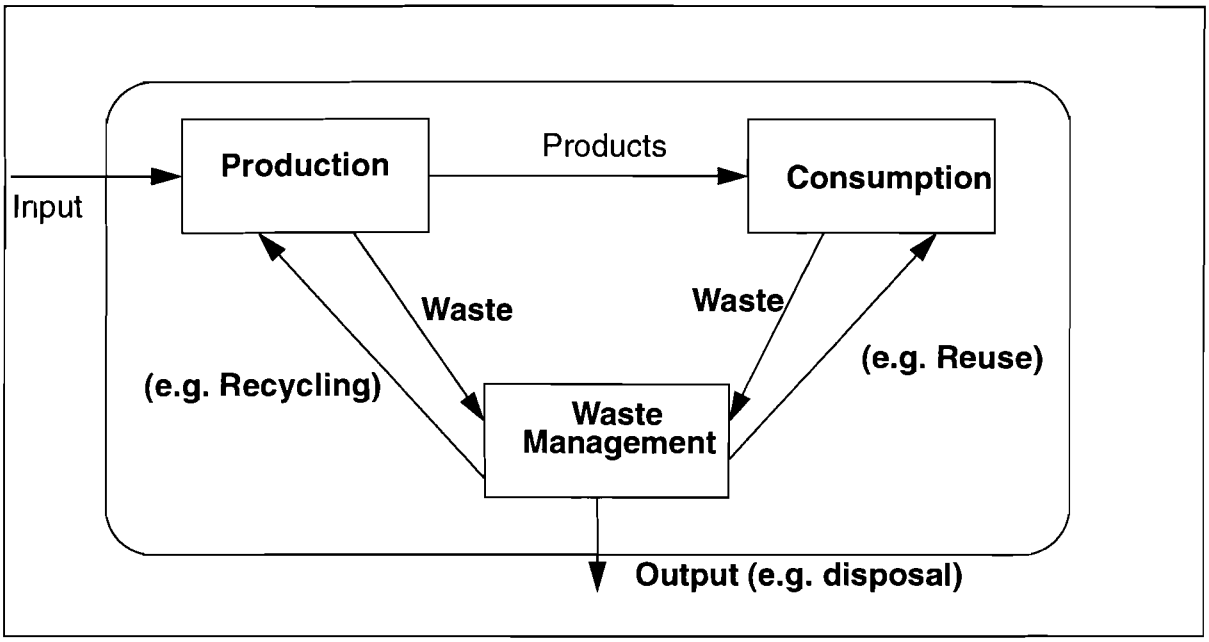


Figure 1: Model of a waste generating system

One main general integrity constraint can be derived from the mass balance approach, where processes are selected through the identification of their input and output flows. According to the second law of thermo dynamics, the sum of inputs of a process must be equal to the output. This means that the output of waste management activities must correspond to the waste generated by production and consumption. More details on this kind of systems can be found in [ANDERBERG93].

In this paper, we will focus on the flows to and from the waste management process (bold text in figure 1). These can be divided into several smaller parts representing flows of different waste types. Especially the waste caused through consumption can be particularized in more detail (see chapter 2.1). The sum of these flows must be equal to the flow modelled firstly.

An efficient waste management information system is a first step towards controlling the modelled system, because it can provide relevant information for several types of waste management tasks. These include among others:

- Decisions in communities and enterprises on recycling strategies or demands for treatment and disposal capacitites, which require reliable information on quantity and composition of waste, current treatment and disposal capacities etc.
- Trends in waste composition and waste quantities which can be given based on data provided by the waste management information system.
- Support of methods for the analysis of municipal solid waste. These methods include direct waste analysis, waste product analysis and market product analysis [BRUNNER86].

At least, a waste management information system can give a detailed description of the current situation with respect to the mentioned waste management tasks. This implies the need for collecting and analyzing data on treatment and disposal of several kinds of waste. Because there is normally an enormous amount of data describing the problem field, a database system is necessary to manage these tasks.

Furthermore, there is a major economical reason for the use of a database system. The methods of collecting the relevant data are mostly time consuming and cost intensive. Data losses or data inconsistencies are therefore expensive events which should be prevented. A database system is an instrument which can store data safely and consistently, if it is used correctly.

Some countries have already enacted laws which force communities and enterprises to report annually the amounts of waste produced by them. For example, the federal state, Lower Saxony, in Germany enacted such a law in 1992. The following case study describes the development of a waste management information system for this state. Emphasis will be on the point of view of a database designer. We will focus on the design of the data structures and the support of integrity constraints in the underlying database system.

2 Case Study: Development of a database system supporting a waste management system for Lower Saxony

The project was initiated in 1990. The main results are:
(1) A questionnaire has been designed to collect the environmental data from the communities of Lower Saxony. It is sent to the communities and could be a framework for their waste management concepts (see chapter 3).

(2) DABI, a prototype application based on the relational-oriented database management system dBaseIV, has been developed to support the process of collecting, aggregating and analyzing the environmental data of the communities. It consists of two application programs. The "Input" program provides a user-friendly, form-based interface for loading the data. The "Output" program offers utilities for evaluation (e.g. calculation of index numbers, preparing balance sheets) [SCHREIBER91], [HEROLD92].

(3) Balance sheets, which contain aggregate information for all the communities in Lower Saxony have been prepared for 1990 [HEROLD91].

In this paper we will focus on some integrity constraints and their enforcement in the database system.

Before describing in detail the problem area itself (including the underlying integrity constraints) and the adopted solution approach, some database basics concerning methods and tools underlying this work are given. We will follow the main database design process consisting of the phases information requirement analysis, conceptual modelling and logical design, and implementation [ELMASRI 89].

Here, the *information requirements* are given by an informal verbal description of the problem area.

On the *conceptual level* an extended Entity Relationship (ER) model proposed by [TEOREY86] is employed. It includes optional relationships and generalization/specialization in addition to the basic concepts of the original ER model [CHEN76]. The extended ER model used in this paper differs slightly from Teorey's proposal in the fact that ER structures are represented by directed, rather than undirected diagrams. This modification allows full aggregation capabilities for modelling ER structures [MARKOWITZ89]. ERDRAW is a graphical X-windows based scheme specification editor to support the design of extended ER diagrams. It is documented by the developers in [SZETO91]. Part 3 shows an extended ER diagram representing the chosen waste management problem.

On the *logical level* the relational model is used [DATE90], [CODD70]. The transformation of an extended ER scheme in relations is based on [MARKOWITZ89]. The *relational implementation* on an SQL-Server, a database management system of SYBASE, is supported by SDT, a database schema design and translation tool developed at Lawrence Berkeley Laboratories [MARKOWITZ93]. This tool uses an extended ER-scheme designed with ERDRAW as input and generates files containing SQL statements for table (relation) definition, index (key) definition and definition of referential integrity constraints.
Principally, SDT can output SQL of the database management systems INFORMIX, INGRES, and SYBASE. We will focus on the SQL statements which create triggers enforcing referential integrity constraints modelled in the underlying extended ER schema in part 2.4. Also some triggers enforcing semantic constraints are shown.
An overview of methods and tools used in the database design phases of the project is given in Figure 2.

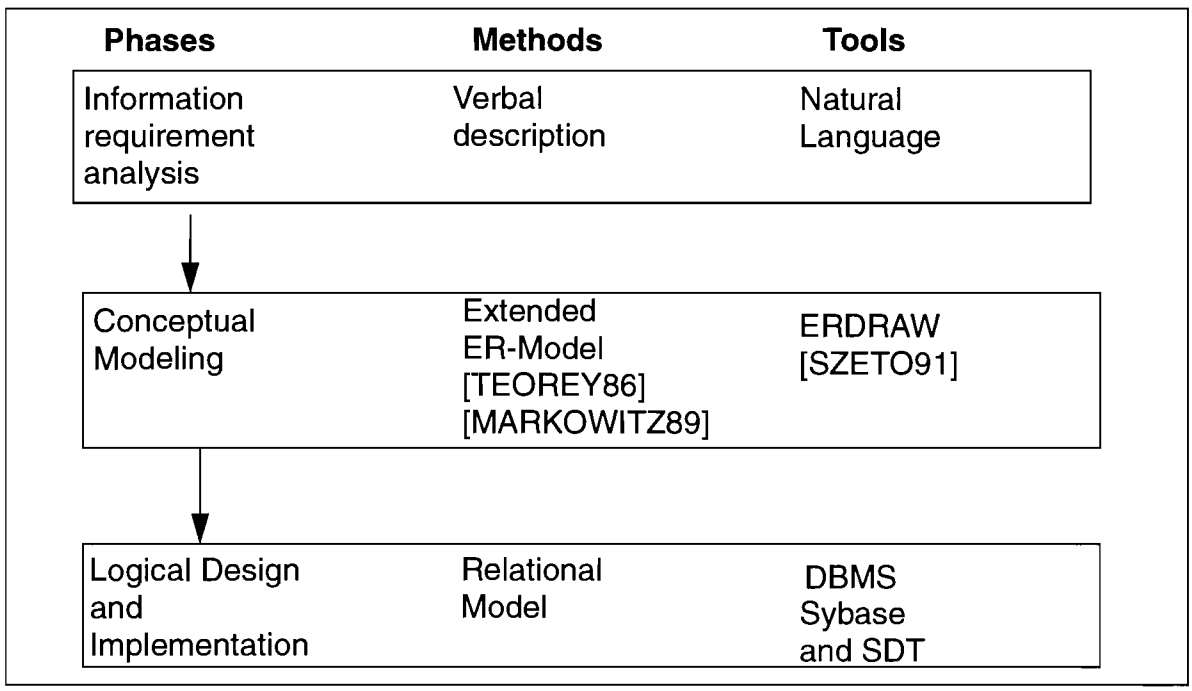


Figure 2: Database design phases, methods and tools of the project

3 Data collection on quantities of different waste types as a waste management problem area (information requirement analysis/ problem description)

The main result of the information requirement analysis has been the design of the questionnaire on waste related data. The Ministry of the Environment in Lower Saxony sends the form each year to the 53 communities, which are obliged to answer.
So the questionnaire collects data for a given year and a given community. It covers the following main aspects:
(1) *Organizational data*
include general, not waste-specific information: For example:
- the address of a community or a district
- information about which community belongs to which district
- address of person responsible for the data collection

(2) *Quantity of waste*
This aspect gathers data concerning the quantity of a particular kind of solid and semisolid waste which has been produced (for a given year and a given community).
(3) *Utilization of reusable materials*
This passage collects data concerning how much of a particular kind of reusable material has been separated in industries and households (for a given year and a given community) and how much of these materials are reused or recycled.
(4) *Collection of pollutants*
This part gathers data on the question of how much of a particular pollutant has been produced (for a given year and a given community).
(5) *Disposal*
Here the question is dealt with on how much waste has been deposited on a depony (for a given year and a given community).

We will analyze aspect (2) in depth to show the problem of designing data structures and enforcing the underlying integrity constraints. Data is collected on the quantities of particular kinds of solid and semisolid waste, which have been produced for a given year and a given community. Figure 3 illustrates structure and content of the analyzed data. The waste classification is based on an order by the government of Lower Saxony, which has already classified many types of solid and semisolid waste (so called Abfallartenkatalog Niedersachsen) [ABFALL91]. The "entirety of waste" is subdivided into "urban waste", "industrial waste", and "rubble" (both builder's and demolition rubble, inc. excavated material). "Urban waste" is particularized into "urban solid waste", "sewage sludge", "medical waste", "separated pollutants", "separated valuable materials" (valuable for recycling and reuse). The class "urban solid wastes" is further broken down into "refuse" (household waste), "commercial waste" (non-industrial waste of enterprises), "street sweepings", "trash" (larger items, which are normally not deposited into garbage cans), "market waste", "garden and park waste", and "other urban solid waste". This shows that some types of waste (e.g. "urban waste", "urban solid waste") are, on the one hand, a superclass of several subordinated types and, on the other hand, a subclass of another kind of waste. Alternative methods for the analysis of municipal solid waste are discussed in [BRUNNER86].

The classification gives a framework for a systematic aggregation of the amounts of waste types. The essential semantic integrity constraint derived from the classification is that the value of a given kind of waste must be equal to the sum of the values of its subordinated waste types. So the wastes types classification includes not only a supertype/subtype, but also a kind of aggregation hierarchy. In the following we refer to this essential semantic integrity constraint as the "aggregation hierarchy" constraint.

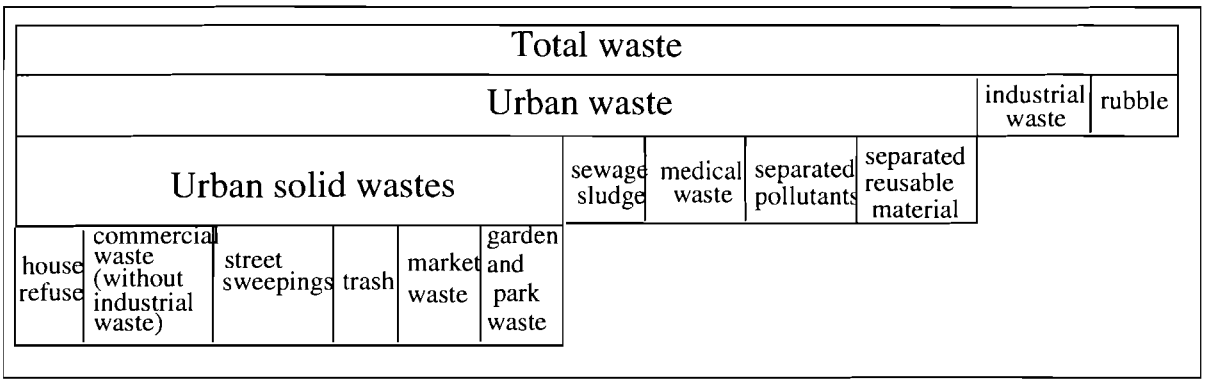


Figure 3: Waste (Solid and Semisolid) Classification of Lower Saxony (Germany)

4 An Extended ER-scheme for waste management tasks (Conceptual level)

An extended ER diagram representing the structure of a database system for waste management tasks (organizational data and amounts of wastes) is given in figure 4. Key attributes (underlined) of the entity types and some other essential attributes of the relationship types are shown, although the diagram originally generated by the tool ERDRAW does not show any attributes. Instead, it offers a detailed report on the attributes in the scheme. Some explanations follow:

The entity sets derived from the verbal description are *communities*, *districts*, *periods* and *waste*. The 1:N relationship type *belong_to* models the actual existence of exactly one district associated to each community. Each district governs at least one, but mostly several communities. The N:M relationship type *gather* represents the various periods (years) for which data for a community exist. Each period is associated with several entities of the entity set *communities*. Some attributes associated with this relationship type are *number of inhabitants* and *number of households*. The N:M relationship type *amounts* is an association of the relationship type *gather* and the entity type *waste*. The N:M cardinality of the relationship type *amounts* models the association of a given combination of community and period with several waste types (instances of the entity set *wastes*). Each waste type is associated with several instances of the relationship type *gather*. The attribute *amount_in_tons* is associated with this relationship. The direction of the connection of the relationship sets *gather* and *amounts* illustrates the use of the abstraction concept of aggregation.

An instance of the relationship type *amounts* is an association of an instance of the entity type *waste* with an instance of the relationship type *gather*, which is already an association of objects.The generalization/specialization hierarchy of the waste types can be modelled

explicitly. Because the subclasses have no specific attributes relevant to the waste management information system and are not participating in a relationship type, it is not neccessary to model the hierarchy explicitly from a data modelling point-of-view [SCHREIBER91]. The 1:N relationship type *hierarchy* is an implicit modelling of a generalization/specialization of waste types. Each waste type in the role of a subclass can belong to a maximum of one waste type, which represents the superclass. On the other hand, a waste type in the role of a superclass can be associated with several other waste types, which are subclasses in this sense.

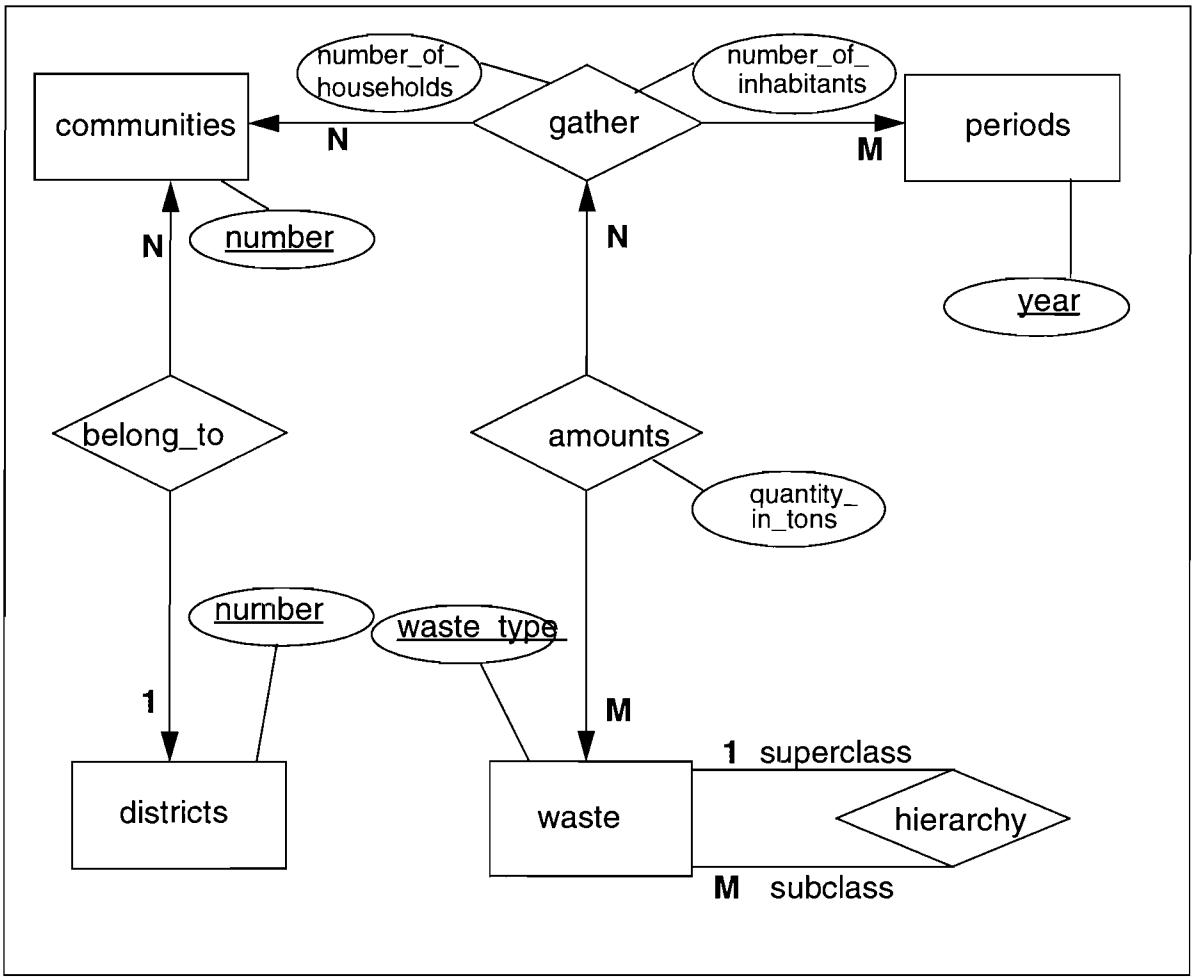


Figure 4: An Extended ER scheme for waste management tasks

While an extended ER scheme can principally support the subtype/supertype constraint of the waste classification, there is no possibility of enforcing the semantic constraint concerning the correct aggregation of waste amounts.

5 Logical design and relational implementation of the extended ER scheme, including the support for referential and semantic integrity constraints

The traditional architecture of a database system and its application programs support simple integrity constraints within the database system, and realize complex (referential and semantical) integrity constraints with application programs. This is insufficient for achieving database consistency in this case. The main problem is that users who are not interacting with the database system via the application can violate the integrity constraints checked by the application. Therefore, it would be useful to control the integrity constraints in the database system itself. Relational database management systems of the second generation (e.g. SYBASE) offer features which can be used for enforcing complex (referential and semantical) integrity constraints by the database system itself [SYBASE91].

Trigger mechanisms are especially useful to realize the integrity constraints. They are fired automatically after a data modification has occurred in the table they belong to. The concept of supporting integrity constraints in the proposed system is compared to the traditional approach in Figure 5.

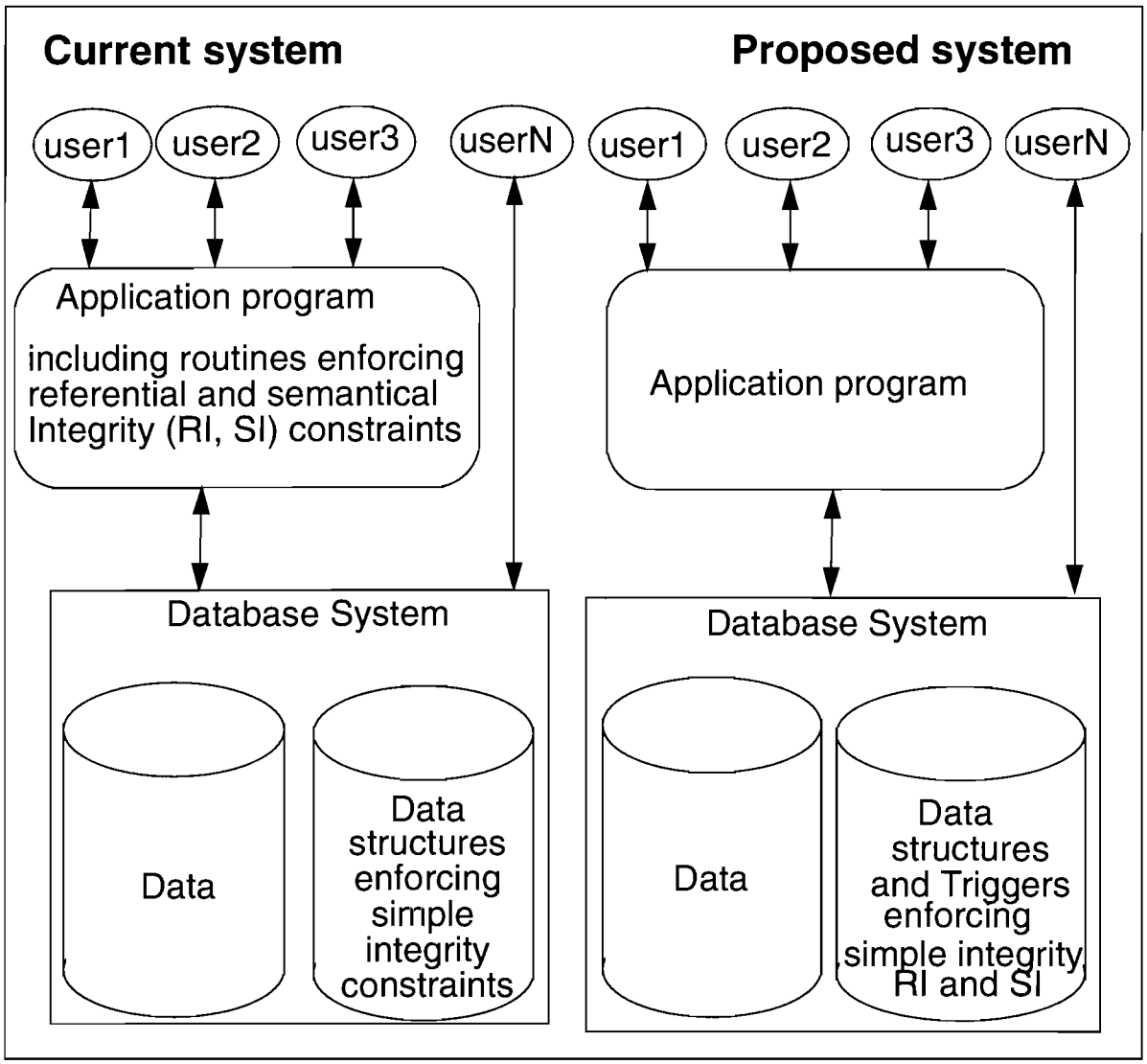
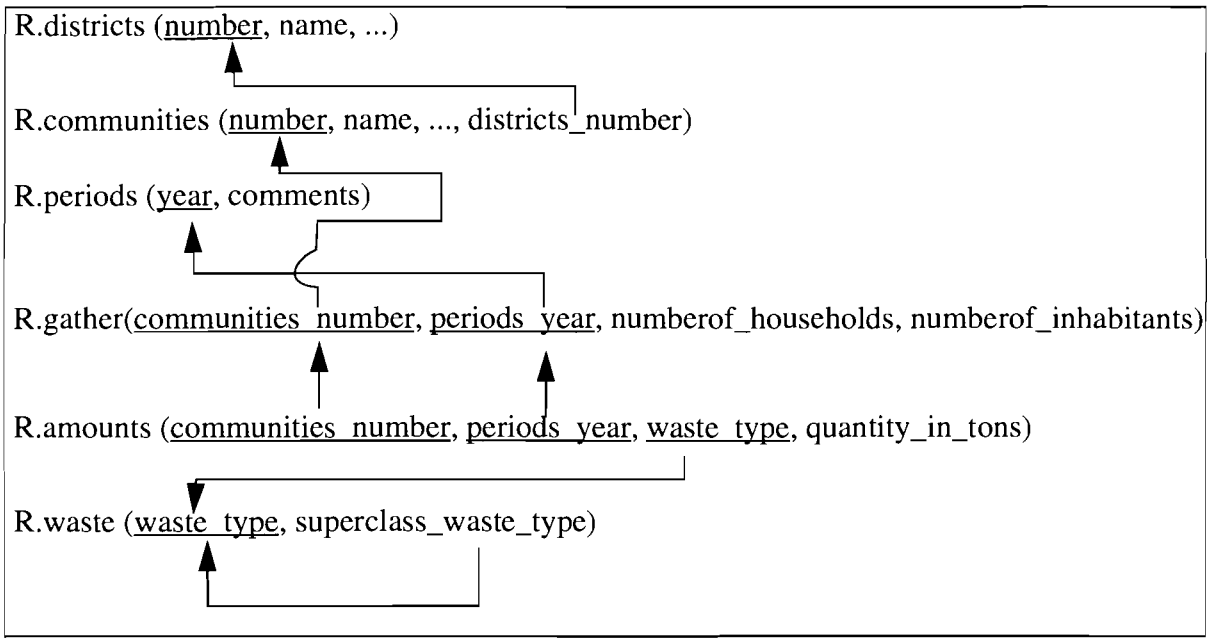


Figure 5: Support of integrity constraints in the traditional and the new approach

5.1 A relational scheme for waste management tasks

The data structures are modelled in a relational scheme. The transformation of the extended ER-diagram in a (merged) relational scheme using SDT creates the following tables:



For each table representing an entity type (*communities*, *districts*, *periods*, *waste*) a delete and an update trigger are defined, enforcing referential integrity. If the table represents an entity type participating on the N side of an 1:N relationship type, an insert-trigger must also be coded (*communities*). For each table representing an N:M relationship type (*gather*, *amounts*) an insert and an update trigger are defined. If the table represents a relationship type, which participates on the 1 side of an 1:M relationship type or is part of an N:M-relationship type (*amounts*), a delete trigger is also created.

The complete conceptual framework on the structure of the triggers generated by SDT are discussed in [MARKOWITZ93b].

SQL statements to create triggers in the database management system, Sybase, and enforcing the referential integrity constraints of the extended ER scheme of figure 4, are shown in figures 6-8. They are part of the output generated by SDT (with slight modifications) when transforming the extended ER scheme. Each "Sybase" trigger begins with the key term CREATE TRIGGER, followed by its name and the table which fires it. At the end of the so-called "trigger head" the programmer defines the operation on the table which will invoke the trigger. Then the definition of the "trigger action" follows. The coding of the action can be done by using procedural extension of SQL ("if -statements", declaration of local variables, etc.). The trigger action normally begins with the declaration of some local variables, then it is checked by a trigger condition to determine whether some referential integrity constraints are violated by the intended operation. This check is normally an attempt to join the tuple to be deleted, inserted or updated with tuples in their referenced or referencing table. The number of successful joins is assigned to the corresponding variable by a SELECT statement. If the

operation does not violate reference integrity, it is committed, otherwise the user receives a message refering to the violation and a list of the tuples involved in it. The transaction containing the operation is then rolled back. The condition of the three trigger types are illustrated for the table *communities*:

5.2 Definition of triggers enforcing referential integrity constraints

Delete trigger (figure 6):

The deletion of tuples of the table *communities* is not permitted if a corresponding tuple in the table *gather* exists. This existence is checked by an attempt to join the table *gather* with the table *deleted* by the attribute *communities_number*. The table *delete* contains all tuples which participate in the actual delete operation. The number of successful joins is calculated by the SQL function count(*) and assigned to the variable @del1collect. The operation is rolled back if the result of count(*) is greater than 1.

```
create trigger deletecommunities
on communities
for delete as
begin
        declare @del1collect int
        select @del1collect = count(*) from deleted, collect
                where deleted.number = collect.community_number
        if @del1collect > 0
        begin
                raiserror 70002 "Cannot delete from communities because
of"
                print "existing reference from collect"
                select * from deleted
                        where exists
                                (select * from collect
                                 where deleted.number =
                                collect.communities_number)
                rollback transaction
        end
end
```

Figure 6: Delete trigger enforcing referential integrity (table *communities*)

```
create trigger insertcommunities
on communities
for insert as
begin
        declare @row int,
                @ins1districts int,
                @null1districts int
        select @row = @@rowcount
        select @null1districts = count(*) from inserted
                where inserted.districts_number = null
        select @ins1districts = count(*) from inserted, districts
                where inserted.districts_number = districts.number
        if @null1districts + @ins1districts != 1 * @row
        begin
                raiserror 70001 "Cannot insert into communities because
of"
                print "missing reference to districts"
                select * from inserted
                        where not exists
                                (select * from districts
                                  where  inserted.districts_number  =
districts.number)
                rollback transaction
        end
end
```

Figure 7: Insert trigger enforcing referential integrity (table *communities*)

Insert trigger (figure 7):

When inserting a tuple into the table *communities*, it must previously be determined whether the inserted value of the foreign key attribute *districts_number* exists in one of the tuples in the table *districts*. Another value not violating referential integrity is NULL. The reference to the table *districts* is checked by an attempt to join this table with the table *inserted* by the attribute *districts_number*. The table *inserted* contains all tuples which participate in the actual insert operation. The number of successful joins is calculated by count(*) and assigned to the variable @ins1districts. The number of tuples in the table *inserted*, which receive a NULL-value in the column districts_number, is calculated by count(*) and assigned to the variable @null1districts. The operation is rolled back, if the number of all tuples inserted by the current operation (assigned to the variable @row) is not equal to the sum of @null1districts and @ins1districts.

```
create trigger updatecommunities
on communities
for update as
begin
   declare @row int,
               @ins1districts int,
               @null1districts int,
               @del1collect int
   select @row = @@rowcount
   if  update (number) or update (districts_number)
   begin
            select @null1districts= count(*) from inserted
                        where inserted.districts_number = null
            select @ins1districts = count(*) from inserted, districts
                        where inserted.districts_number = districts.number
            select @del1collect = count (*) from collect
                        where exists
                                    (select * from deleted
                                     where deleted.number = collect.communities_number)
                         and not exists
                                    (select * from inserted
                                                where            inserted.number            =
collect.communities_number)
            if @null1districts + @ins1districts != 1 * @row + @del1collect
            begin
                        raiserror 70003 "Cannot update communities because of"
                        if @null1districts+ @ins1districts != @row
                        begin
                                    print "missing reference to districts"
                                    select * from inserted
                                                where not exists
                                                            (select * from districts
                                                                                                    where
inserted.districts_number
                                                            = districts.number)
                        end
                        if @del1collect != 0
                        begin
                                    print "existing reference from collect"
                                    select * from deleted
                                                where exists
                                                            (select * from collect
                                                             where deleted.number =
                                                             collect.community_number)
                                                 and not exists
                                                            (select * from inserted
                                                             where deleted.number
                                                             =inserted.number)
                        end
                        rollback transaction
            end
   end
end
```

Figure 8: Update trigger enforcing referential integrity (table *communities*)

Update trigger (figure 8):
Because an update can be interpreted as a delete followed by an insert, the update trigger is a combination of the other trigger types. The update trigger is fired if the column *number* or the attribute *districts_number* are updated.
The reference to the table *districts* is checked by an attempt to join this table with the table *inserted* by the attribute *districts_number* (like with the insert trigger). The reference to the table *gather* is checked by an attempt to join the table *deleted* with the table *gather* (like with the delete trigger) and the table inserted with the table collect. The update violates referential integrity if the old value is a reference to the table *gather* and the new one is not.

5.3 Definition of triggers enforcing the aggregation hierarchy constraint

After having described some basics of the triggers generated by SDT to enforce referential integrity, the triggers of the table *amounts* must be (re)coded to enforce the aggregational hierarchy integrity constraint of the waste classification.

An insert operation on the table *amounts* is committed if no subordinated waste types exist or if the sum of the amounts of the subtypes for the community-year-combination given by the current tuple are equal to the inserted value in the column *quantity_in_tons*. If the current waste type has a superclass, modifications on the quantity in the superclass will be executed.

The insert trigger generated by SDT is enhanced by the following algorithm (figure 9): After having checked the referential integrity constraints, the program asks whether basically subordinated waste types exist. If some subclasses exist, the trigger checks whether the table *amounts* contains any tuples for these subclasses and the community-year-combination given by the current tuple. In case of no tuples, the operation is rolled back. It is therefore impossible to insert a tuple in the table *amounts* if its waste type basically has some subclasses and if no tuples exist in table *amounts* containing quantities of waste for the community-year-combination given by the current tuple. If any amounts of some subclasses exist, the sum of these is compared to the inserted value in column *amount_in_tons* (of their super-waste-type). If the sum is not equal to the inserted value, the operation is rolled back, otherwise it is committed. Then a procedure follows which is executed in the case of no waste subtypes. If a supertype basically exists, there are two possibilities: If there is no tuple in the table *amounts* for the super-waste-type and the current community-year-combination, a new tuple is inserted. It holds the super-waste-type and the current community-year-combination as values for the primary key and the amount_of_tons given in the tuple of the table *inserted*. Otherwise, if there is a tuple a in the table *amounts* about the super-waste-type and the current community-year-combination, the current subtype amount is added to the old supertype amount. If there is basically no upper waste type, then the trigger ends.

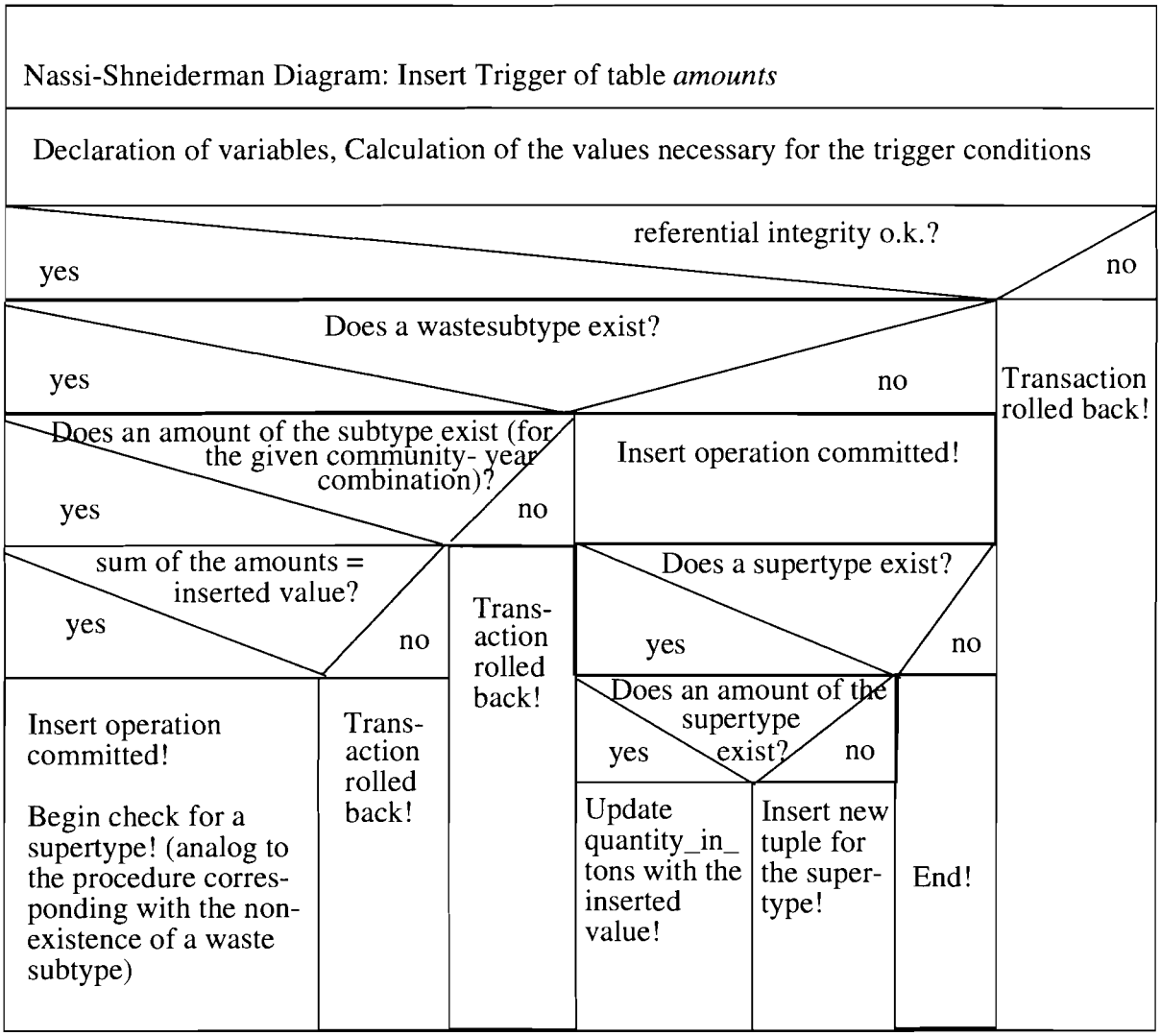


Figure 9: Structogram of the insert trigger

An update on the table *amounts* is similar to an insert. It is committed if no subordinated waste types exist or if the sum of the subtype quantities for the community-year-combination given by the current tuple is equal to the inserted value in column *amount_in_tons*. If the current waste type has a superclass, a modification on the amount of the superclass will be executed.

The update trigger generated by SDT is enhanced by the following algorithm (figure 10): After having checked the referential integrity constraints, the program asks whether basically subordinated waste types exist. If some subclasses exist, their amounts also exist for the community-year-combination given by the current tuple (because of the insert trigger). The sum of these amounts is then compared to the new value in the column *amount_in_tons* (of their super-waste-type). If the sum is not equal to the new value, the operation is rolled back; otherwise it is committed. Then the procedure follows which is also executed in the case of no existing waste subtype. If a supertype basically exists, a tuple must also exist in the table *amounts* for the super-waste-type and the current community-year-combination (because of the insert trigger). This tuple is updated by adding the new amount of the current subtype and subtracting its old value. If there is basically no upper waste type, the trigger ends.

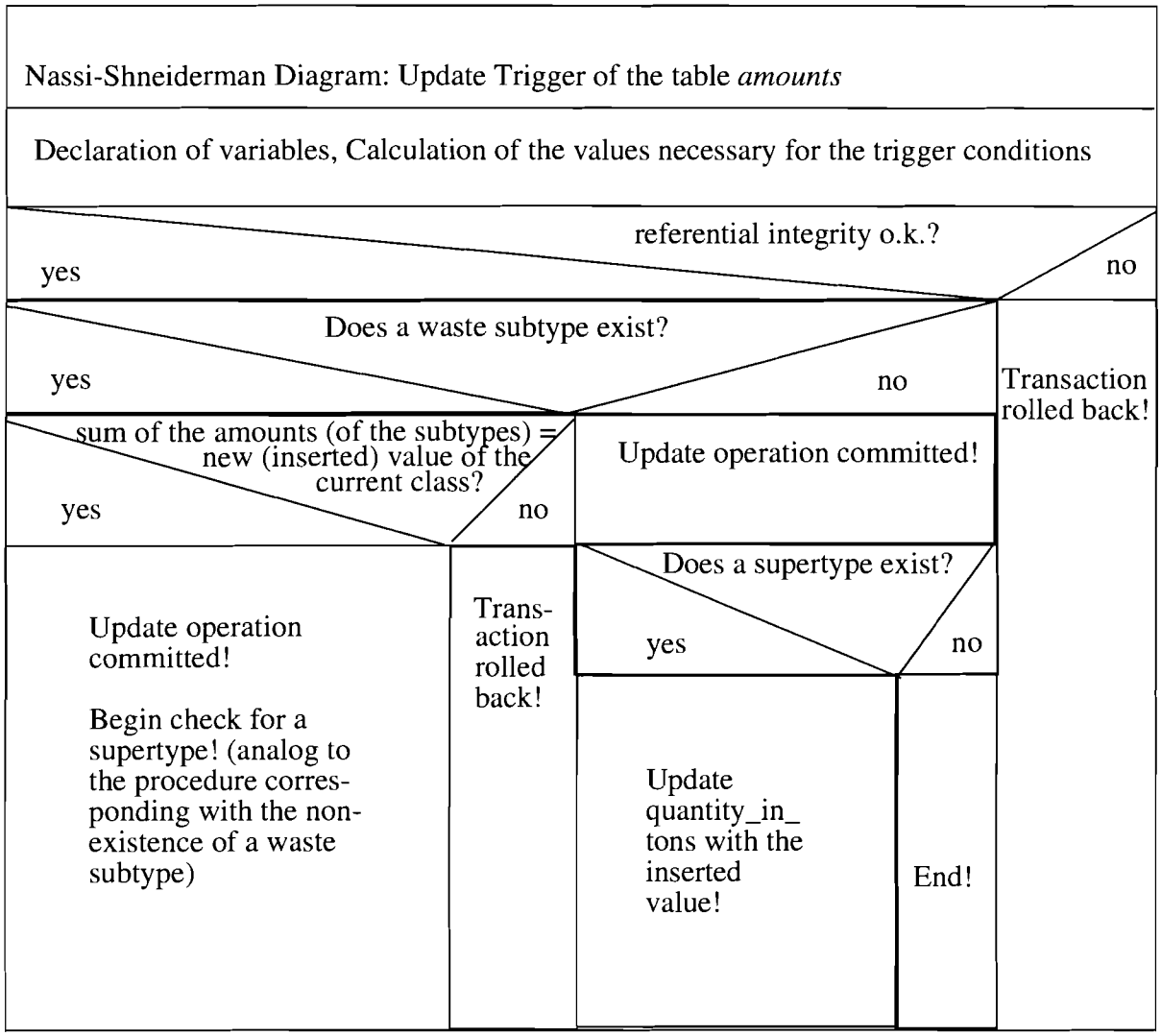


Figure 10: Structogram of the update trigger

A deletion of a tuple in the table *amounts* is committed if no subordinated waste types exist. If the current waste type has a superclass, a modification on the quantity in the superclass will be executed.

The trigger action of the delete trigger is defined by the following algorithmn (figure 11): If quantities of some subordinated waste types exist for the community-year-combination given by the current tuple, the operation is rolled back. In the case of no tuples in the table *amounts* about the quantity of subtypes, the operation is committed. Then the program checks whether a supertype basically exits. If a superclass basically exists, a tuple in the table *amounts* must exist for this supertype and the community-year-combination given by the current tuple (because of the insert-trigger). Then the trigger tests whether there are still any other tuples in the table *amounts* for this community-year-combination given by the current tuple, which are subordinated to the super waste type of the current waste type. If there is no such tuple, the corresponding tuple of the super-waste-type in table *amounts* is deleted. If such a tuple exists,

the amount of the super-waste-type for the given community-year-combination is changed by subtracting the current (deleted) waste type amount. If there is basically no upper waste type, the trigger ends.

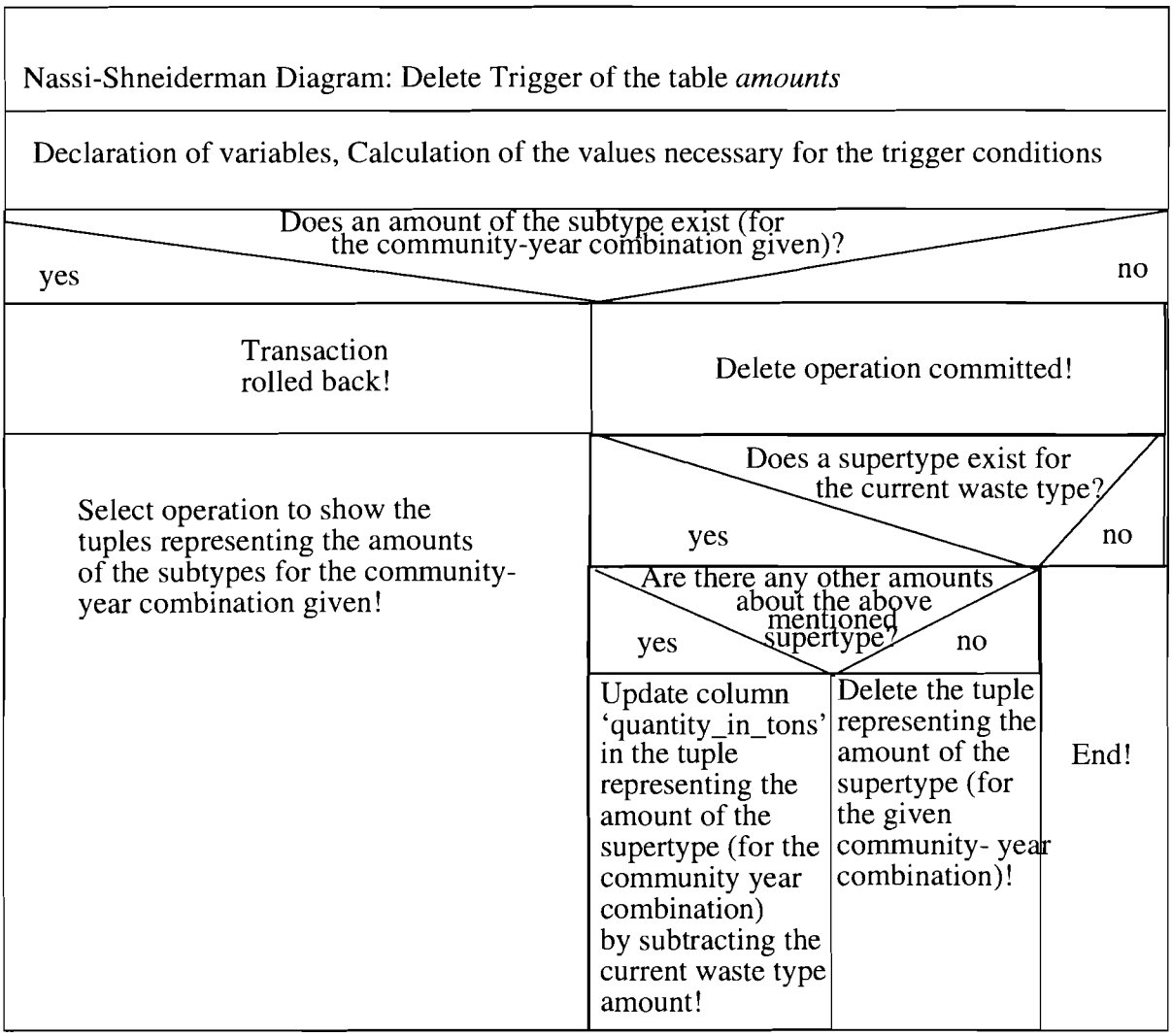


Figure 11: Structogram of the delete trigger

6 Conclusions

The support of integrity constraints in a waste management information system is an important task. Therefore, in addition to the design of the data scheme, an algorithm enforcing a special kind of aggregation constraint has been developed. It is implemented using triggers.There are, however, some problems with this implementation:

The parts of the trigger enforcing referential integrity support operations manipulates several tuples of this table simultaneously. The trigger action enforcing the aggregational constraint only works well when manipulating one tuple for each SQL statement. Therefore applications should not allow operations manipulating several tuples of the table *amounts* simultaneously.

Restriction, Cascading and Nullifying, as reactions to a violation of referential integrity

constraints [DATE 90], can principally be coded as a trigger action in Sybase. The triggers used in this project react restrictively.
They work well because the subclasses of a given waste type are disjunct. If a subclass has multiple superclasses, the cascading modifications are more complex.
Therefore, work is necessary in the future to solve these problems. It will have to include empirical tests concerning the performance of the trigger implementation and also a comparison of the described approach with the traditional method of enforcing integrity constraints by the application program.
The developed algorithms can be used in other problem areas as well. This will be demonstrated by applying them to a project concerning automobile recycling.

Acknowledgements

I would like to thank Prof. Y. Ermoliev (International Institute of Applied Systems Analysis, Austria) and Prof. M. Grauer (University of Siegen, Germany) for their support, without which this paper could not have been completed, and for their many useful and constructive comments.

References:

[ABFALL91] Ausschluß von Abfällen nach § 3 Absatz 3 AbfG - Abfallartenkatalog Niedersachsen, RdErl. d.MU vom 29.06.1991 in Nds. Nummer 36/91: 1278 - 1331.

[ANDERBERG93] Anderberg, S.; Bauer, G.; Ermoliev, Y.; Stigliani, W.: Mathematical Tools for Studies of Industrial Metabolism, IIASA Working Paper WP-93-9, February 1993.

[BRUNNER86] Brunner, P.H.; Ernst, R.W.: Alternative methods for the analysis of municipal solid waste, in: Waste Management and Research, 1986/4

[HEROLD91] Herold, H.: Niedersächsische Abfallbilanz 1990, September 1991.

[CHEN76] Chen, P.: The Entity Relationship Model - Toward a Unified View of Data, in: Transactions on Database Systems 1:1, March 1976.

[CODD70] Codd, E.F.: A relational model for large shared databanks, in: Communication of the ACM, June 1970.

[DATE90] Date, C.J.: An Introduction to Database Systems, Band 1, Addison-Wesley, Reading, 5th edition, Massachusetts, 1990.

[ELMASRI 89] Elmasri, R.; Navathe, S.: Fundamentals of Database Systems, Benjamin/ Cumming, Pub. Comp. 1989.

[HEROLD92] Herold, H.; Schreiber, D.: DABI - ein Datenbanksystem zur Abfallbilanzierung, Abfallwirtschaftsjournal 1992, Nr. 4, pp 287 - 293.

[KLEINALTEN...90] Kleinaltenkamp, M.: Recycling Strategien, Erich Schmidt Verlag, Berlin 1990.

[MARKOWITZ89] Markowitz, V.M.; Shoshani, A.: On the correctness of representing extended entity-relationship structures in the relational model, in: Proceedings of the ACM SIGMOD Conference, June 1989, pp 430-439.

[SZETO91] Szeto, E.; Markowitz, V.: ERDRAW: A Graphical Schema Specificati-

on Tool, Reference Manual Draft 2.2. Technical Report Lawrence Berkeley Laboratory (LBL) PUB -3084.

[MARKOWITZ93] Markowitz, V.M.; Fang, W.: Wang, J.: SDT 5.1, A Schema Definition and Translation Tool for Extended Entity -Relationship Schemas (Technical Report LBL-27843), Lawrence Berkeley Laboratory, 1993.

[SCHREIBER91] Schreiber, D.: Prototyp-Entwicklung eines Datenbanksystems zur überregionalen Abfallbilanzierung, Arbeitsbericht Nr. 5 des Instituts für Wirtschaftsinformatik an der Universität Gesamthochschule Siegen.

[SYBASE91] Sybase Dokumentation: Transact-SQL User's Guide, Emeryville (California), October 1991.

[TEOREY86] Teorey, T.J.; Yang, D.; Fry, J.P.: A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model, in ACM Computer Surveys, Vol. 18 No. 2, June 1986, pp 197-22.