Proceedings of the *International Conference on Theory and Application of Mathematics and Informatics ICTAMI 2005* - Alba Iulia, Romania

DATA CONSTRAINT LANGUAGE

Călin Adrian Comes, Nicolae Ghişoiu

Abstract. Data Constraint Language, platform independent specification language, extends the Entity-Relantionship model for RDBMS/ODBMS in idea to support the syntactical and semantical validation for database diagrams

1.Introduction

Procedures and **triggers** store procedural SQL statements in a database for use by all applications. They can include control statements that allow repetition (LOOP statement) and conditional execution (IF statement and CASE statement) of SQL statements. **Procedures** and **triggers** [1] standardize actions performed by more than one application program. By coding the action once and storing it in the database for future use, applications need only call the procedure or fire the trigger to achieve the desired result repeatedly. And since changes occur in only one place, all applications using the action automatically acquire the new functionality if the implementation of the action changes.

Procedures [2] are invoked with a CALL statement, and use parameters to accept values and return values to the calling environment. SELECT statements can also operate on procedure result sets by including the procedure name in the FROM clause. Procedures can return result sets to the caller, call other procedures, or fire triggers. For example, a user-defined function is a type of stored procedure that returns a single value to the calling environment. User-defined functions do not modify parameters passed to them, but rather, they broaden the scope of functions available to queries and other SQL statements.

Triggers [2] are associated with specific database tables. They fire automatically whenever someone inserts, updates or deletes rows of the associated table. Triggers can call procedures and fire other triggers, but they have no parameters and cannot be invoked by a CALL statement.

2.Motivation Scenario

Database design evolved according to the evolution of RDBMSs and data models. When data models with more expressive power were born, RDBMSs were capable of incorporating more semantics, and physical and logical designs started distinguishing one from the other as well. With the appearance of the relational model, DB design focused, especially in the academic field, on the normalization theory. ANSI architecture, with its three levels, also had a considerable influence on the evolution of design methodologies. It helped to differentiate the phases of DB design.

In 1976, the E-R model proposed by Chen [4, 5] introduced a new phase in DB design: **conceptual modeling**.

2.1 Basic E-R Diagram Chen- Data Modeling Schema

The E-R diagram is a semantic data modeling tool that is used to accomplish the goal of abstractly describing or portraying data. Abstractly described data is called a **conceptual model**[4,5]. Our conceptual model will lead us to a "schema." A schema implies a permanent, fixed description of the structure of the data.

Therefore, when we agree that we have captured the correct depiction of reality within our conceptual model, our ER diagram, we can call it a schema. In the Chen-like model, attributes that are **unique identifiers** (candidate keys) are usually underlined. A unique identifier can be an attribute or a combination of attributes. It is not necessary to choose which candidate key will be the primary key at this point, but one could do so. When there is only one candidate key, we will generally speak of it as the primary key, simply because it is obvious that the primary key is a candidate key.

Finally, while on the subject of keys, we will have situations in the ER diagram (in the Chen-like model) where no key is obvious or intended. Entities that have at least one identified key can be called **strong entities**. In Chen's (1976) original article, strong entities were called **regular entities**. Some entities will be discovered which depend on other entities for their being (and

hence their identification). Chen called those entities that rely on other entities for their existence, **weak entities**.

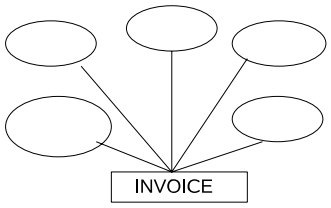

Figure 1: The E-R diagram acording to Chen data modeling schema

2.2 Barker/Oracle-Like Data Modeling Schema

The Chen-like model focuses on modeling data, whereas the Barker/Oracle-like model adapts the data to the relational database concurrently with the design. Therefore, the ER design methodology for the Barker/Oracle-like model will develop differently from the Chen-like model. Further, the Barker/Oracle-like model does not have some of the conventions used in the Chen-like model. For example, the Barker/Oracle-like model does not directly use the concept of composite attributes, multi-valued attributes, or weak entities, but rather handles these concepts immediately in light of the relational model. Because the Barker/Oracle model is so close to the relational model to begin with, the mapping rules are trivial the mapping takes place in the diagram itself.

All attributes in a Barker/Oracle-like model are considered simple or atomic, as in relational databases, this model does not have the concept of composite attributes.

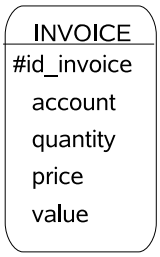

Figure 2: The E-R diagram acording to Barker/Oracle data modeling schema

2.3 Data Constraint Language

We proposed the **data constraint** in idea to model the derivate atribute, in our case the atribute **value** := **quantity** * **price** for each recordset. In database literature the atribute **value** is called **derivated atribute**.

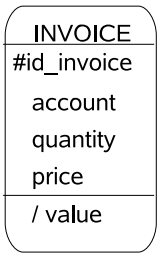

Figure 3: The E-R diagram with Data Constraint

2.4 Triggers and Data Constraint Language

Triggers map our tentative to model the **value** atribute for each recordset, **but** the syntax for them is different in IBM DB2, MySQL, MS SQL, Oracle, ASE Sybase, PostgreSQL. We proposed and defined an common language, Data Constraint Language with the following "code" in OCL style[7] for our sample:

context $Invoice$ **inv** :
pre : $id\_account : Integer = id\_account.Invoice$
pre : $account : Integer = account.Invoice$
pre : $quantity : Integer = quantity.Invoice$
pre : $price : Integer = price.Invoice$
post : $value.Invoice = forAll(id\_account, account, quantity, price, quantity * price)$

3. Conclusion

In this paper we try to define Data Constraint Language, platform independent specification language, in intention to extend the Entity-Relantionship model for RDBMS/ODBMS diagrams in idea to offer a clear picture for database design.

4. Further work

In the future we study the syntax and semantic for Data Constraint Language, platform independent specification language for RDBMS/ODBMS in idea to support the syntactical and semantical validation for database diagrams.

References

[1] IBMDB2 Universal Database, SQL Reference for Cross-Platform Development, Version 1.1, IBM Corporation, North Castle Drive Armonk, NY 10504-1785, U.S.A., Copyright IBM Corp. 1982, 2003

[2] SQL AnywhereStudio Help, Part number: DC38176-01-0902-01, Last modified: Sybase, Inc., iAnywhere Solutions, Inc. October 2004

[3] Mario Piattini, Oscar Diaz, - *Advanced Database Technology and Design*, Artech House, ARTECH HOUSE, INC. 685 Canton Street, Norwood, MA 02062, 2000

[4] Chen, P. P., - *The Entity/Relationship Model: Toward a Unified View, ACM Trans. on Database Systems*, Vol. 1, No. 1, pp. 9-36. Mar. 1976,

[5] Chen, P. P., - *The Entity/Relationship Model: A Basis for the Enterprise View of Data*, AFIPS Conference Proc., Vol. 46, 1977.

[6] Sikha Bagui, Richard Earp, -*Database Design Using Entity-Relationship Diagrams*, Auerbach Publications, Boca Raton, Florida 33431, 2003

[7] OMG, Object Management Group. Object Constraint Language (OCL), OMG Final Adopted Specification (ptc/03-10-04)

Călin Adrian Comes
Department of Computer Science
"Petru Maior" University of Tg-Mureş
Nicolae Iorga, 1, MUREŞ county, 540088
email:*calin.comes@ea.upm.ro*

Nicolae Ghişoiu
Departament of Computer Science
"Babeş-Bolyai" University of Cluj-Napoca
Teodor Mihali, 58-60, CLUJ county, 400591
email:*nghisoiu@email.ro*