The Data Organization

1251 Yosemite Way
Hayward, CA 94545
(510) 303-8868
info@thedataorg.com

The Relational Model Revisited

By Rainer Schoenrank

Data Warehouse Consultant

May 2017

Biography

Rainer Schoenrank is the senior data warehouse consultant for The Data Organization. He has degrees in physics from the University of Victoria and computer science from the University of Victoria and California State University Hayward. He has built data warehouses for clients such as Pacific Bell, Genentech, GE Leasing, SGI, PPFA, Brobeck, BofA, Clorox, Leapfrog and Intuitive Surgical. He can be reached at rschoenrank@computer.org.

Table of Contents

1. Introduction

To create a database, the data being created by the business processes is organized into data structures. Whether a particular database is an appropriate solution to the business problem depends on two things: the database's value and fit. The fit depends on the database making the same assumptions as the assumptions under which the business operates. The value depends how easily and accurately the data can be extracted from the database.

In "A relational model for large shared databanks" Codd invented The Relational Data Model with a relational algebra that provides the specification for extracting data correctly from a set of relations and constraints. The relational algebra has following features

- A set of elements
- Operations on the set of elements
- The operations are closed on the set of elements
- The operations have identity and inverse elements

Unfortunately, the paper mixes the theory of the relational algebra with how a relational data model could be implemented. For example, in its definition of the join operation, it implies the existence of foreign keys which are an artifact of an implementation of the constraints of the relational data model.

This relational algebra is the basis of the current database technology. In the rush to implement the relational algebra as a database management system (DBMS), the implications of the relational algebra were not fully developed. I believe that the implications of the relational algebra and the requirements the relational data model imposes on the database definition have been overlooked.

2. Relational Data Model Definition

Summarizing Codd’s Relational Data Model,

- The elements of the relational data model are:
    - Constraints between relations. A constraint is a logical restriction between two relations.
    - Relations – A relation is a subset of n-tuples taken from the Cartesian product of the ordered set of n unique domains.
    - A domain is a single value function. That is, the abstract data type of a domain cannot be a multivalued data type, such as a relation, list, array, set, or stack.

Summarizing Codd’s relational algebra (the operations on the Relational Data Model)

- The operations of the relational algebra are
    - Permutation – the permutation operator creates a new relation with the domains in a different order. There are n factorial permutations available on a relation with n domains, this includes the identity and inverse permutations.
    - Projection – the projection operator creates a new relation with one or more domains removed from the old relation.
    - Join – the join operator creates a new relation from two relations that are connected by a constraint. The new relation contains the set of domains of the two relations. This definition of join is different from the definition given by Codd. Codd’s definition does the join on the same domain in two relations. In a relational data model, this condition would only occur on an implemented database, not on a logical data model.
    - Composition – the composition operator creates a new relation that projects the first and last domains of the join of two relations. This operation is redundant because it is a very specific join operation followed by a pre-defined projection operation.
    - Restriction – the restriction operator creates a new relation as a subset of the old relation so that the new relation has fewer n-tuples than the old relation.

Using the relational algebra on a set of relations assures us that the results of the operations on the relations will always be logically correct. The relational algebra puts data extraction on a formal mathematical basis allowing a database management system to be provably correct.

3. Implications of the Model

The relational algebra assumes that the data contents of the data model do not change in time. The processing of the data life cycle ensures that this assumption is true for the data in the relational data model.

The relational algebra is a logical construct. Its implementation in a DBMS is an approximation to the relational algebra in the same way computer based integer arithmetic is an approximation to the real thing.

There is no limit to the number of relations and constraints in a relational data model. In a relational data model, each element can exist because the result of a relational operation is an element of the relational data model.

Each unique set of relations, constraints and data is an instance of a relational data model. There is no single relational data model. There is a new relational data model for each different set of relations, constraints and data.

A relation does not require a primary key. By definition, there are no duplicate elements in a set; therefore, every n-tuple in a relation is unique. If two real world objects map onto the same n-tuple, then not all of the domains in the relation have been identified and included in the relation.

The composition operation implies that there is only one constraint between any two relations.

The meaning of a relation is given by its label and domains.

There is a one to one constraint between the relation and each of its domains.

The meaning of a domain is given by its label, values and data type.

4. Database Requirements

Not all databases created in a relational DMBS are instances of a relational data model just as not all numbers are integers. If a database is not an instance of a relational data model, then using the relational operations to extract data from the database will produce inconsistent results.

A database of relations (entities/tables) belongs to one of three separate groups:

1. A collection of relations that is not a set.
2. A set of relations that are not elements of a relational data model (for example, there is more than one relation and no constraints between the relations)
3. A set of relations that are elements of a relational data model (Codd’s description)

The requirements for a database to contain non-duplicated data, allow correct record updating and correct data retrieval are met if the set of relations in the database are a minimum cover set (Garey) for the relational data model.

By definition, a relational data model that consists of a single relation (the first normal form of the database) is a minimum cover set for the database. The minimum cover set contains the domain universe of the relational data model. The domain universe is the set of all the domains of all the relations in the relational data model. The first normal form of a relational data model implies that a domain appears in only one relation within the database.

In principle, a relational data model contains all of the relations that can be created using the relational operations. Kent observed that this meant that the definition of a relation is arbitrary and the requirements for defining the relations from the domain universe had to come from outside the relational data model.

5. Normalization

Codd's normalization procedure requires the partitioning of a minimum cover set into a hierarchy (set of non-overlapping subsets of relations). Essentially, the constraints of the relational data model form a tree (i.e., there are no loops in the graph where the constraints are the edges and the relations are the nodes). Jevons has shown that the partitioning of a set into a hierarchy of non-overlapping subsets is an arbitrary process that has no a priori best solution.

The problem of partitioning a set of sets (the elements of the relational data model) into a database is equivalent to the set cover optimization problem – an NP complete problem. For NP complete problems, solutions can be verified quickly, but the solutions cannot be computed quickly, i.e., there is no available algorithm that will solve the problem in polynomial time.

The design process to create a logical data model is a hierarchical decomposition of the domain universe using the Information Principle (Date) and the Principle of Orthogonal Design (Date).

The Information Principle is the requirement to explicitly describe the properties of the attributes in the data model. The properties of the attributes, such as description, data type, number of occurrences, time dependence, and the relationships between attributes, are the basis for the metadata of the database.

The Principle of Orthogonal Design is basically the reformulation of the requirements that

1. The database is a minimum cover set of the Relational Data Model, that is, there is no duplication of the data in the database.
2. There is no overlap between the relations, that is, the normalization procedure has been applied to the first normal form of the database.

6. Gaps in the Relational Algebra

By definition, there are gaps in the relational algebra. The relational algebra is an axiomatic system and is therefore incomplete (Gödel’s incompleteness theorem). The relational algebra only applies to selecting data from relations that are elements of the relational data model, not to creating relations and populating them with data. Missing from the relational algebra are:

- How to create relations
- How to populate relations with n-tuples

The data aggregation operations of the DBMS are not part of the relational algebra. The processing of the data for reports is:

1. Load the relations with the collected data (not part of the relational algebra)
2. Create the required n-tuples in a relation (using the relational algebra)
3. Aggregate the relation n-tuples to the required reports (not part of the relational algebra)

7. Implementing a Relational Algebra

A DBMS implements the relational algebra of set operations of permutation, projection, join, composition, and restriction. The relational algebra allows the DBMS to use optimizing processes based on the theorems of the relational algebra. Unfortunately, the DBMS does not ensure that the results of the relational operations are relations, that is, the results are not guaranteed to be a set of n-tuples. In the DBMS, there can be duplicate rows as the result of the relational operations on the tables.

The DBMS implementations are:

- The relations are implemented as tables.
- The domains are implemented as columns in the tables.
- The constraints are implemented as foreign key columns and relationships between tables. The insertion of the foreign key columns into the related table violates the minimum cover set property of the database.
- The set property of relations is implemented as keys on the tables. All the columns would be involved in the key to ensure the set property, but this has performance issues. Therefore, the table key is a subset of the table columns which define a unique n-tuple. The use of surrogate keys, timestamp keys and DBMS generated keys violates the set property of relations.
- The DBMS implemented data types are not the user defined abstract data types allowed in the relational data model. The data types used in the domains are abstract logical data types and are not limited by the simple data types of the DBMS implementation.
- The implementation of some abstract data types must be as tables and foreign keys. This complicates the analysis of the implemented database and causes confusion about the meanings of the tables in the database.
- For the DBMS table operations SELECT and JOIN result to be a relation and preserve the set property of the relation, the clause GROUP BY is required on each column in the result.

8. Properties of a Database

For a database, the relations are elements of a relational data model and the set of relations is a minimum cover set of the relational data model. The required properties of the logical data model are:

1. Each entity has at least one relationship to another entity in the database (there are no unconnected entities).
2. There are no loops in the graph of the database entities and relationships, the result of normalization of the first normal form minimum cover set.
3. The entities are distinct and there is no overlap of meaning between the entities (Principle of Orthogonal Design) , that is
   a. Each entity has a natural key
   b. No two entities have the same natural key.
4. The attributes are distinct, that is, an attribute does not appear in more than one entity (the minimum cover set requirement), although many attributes may use the same abstract data type. It is very easy to confuse attributes and logical data types in the data model.
5. The attributes are single value functions, that is, you cannot parse the attribute value to get more data. (Date)
6. All the data is represented explicitly in the database, that is, the entity and attribute labels (database metadata) do not imply information about the data. (Information Principle)

9. Validating a Database

The Relational Data Model is a mathematical construct, like Linear Algebra or Number Theory, not a database design methodology. The Relational Data Model provides a validation technique for a relational data model. The implementation of a logical database (an instance of the relational data model) requires many approximations and engineering choices that invalidate the axioms of the Relational Data Model.

Normalization is a process that validates that a logical database adheres to the requirements of the relational data model. The techniques of normalization are a method for testing the design of an existing logical database to determine whether it is a relational data model and a minimum cover set.

Since the relational data model is a logical construct, the normalization must be done at the logical database level only. When implementing a logical database in a DBMS, engineering compromises are made in the creation of the relations, constraints and domains. These compromises mean that the implemented database is no longer a member of the relational data model. There are no techniques for testing an implemented database since the implementation engineering decisions do not have relational data model requirements.

Using normalization to modify the database design after the logical database is implemented in a DBMS implies:

- a misunderstanding of the relational data model and the database design process
- that the domain requirements used in the relation definition are invalid.

10. Summary

Not all databases built using a relational DMBS are instances of the relational data model. To ensure that the reports extracted from a database are logically correct, the database must meet two conditions:

1. Its logical data model is a relational data model.
2. The conversion from logical data model to physical database was done without compromising the logical data model.

Database design mistakes are caused by misconceptions about the Relational Data Model and misunderstanding the database design process. It is important that the database design is robust because mistakes are very expensive to correct. In general, a database is used by many applications and each change in the database design will cascade into many changes in each application. A robust database design will be able to absorb the changes to the business requirements without changing the existing table definitions.

11. References

Codd, Edgar [1970] A relational model for large shared databanks, Communications of the ACM, vol 13, number 6, June 1970

Date, C.J., [2006] The Relational Data Dictionary, Sebastopol, CA, O’Reilly Media, Inc.

Kent, William [1978] Data and Reality, New York, North Holland Publishing

Jevons, William Stanley [1874] The Principles of Science, New York, Dover Publications

Garey, Michael R and Johnson David S [1979] Computers and Intractability, New York, W. H. Freeman and Co.