

RELATIONAL DATABASE DESIGN THEORY: AN OVERVIEW

Faith Renee Sloan,
Parke-Davis Pharmaceutical Research Division, Ann Arbor, Michigan

INTRODUCTION

With the addition of Structured Query Language (SQL), a database language that has become the American National Standards Institute (ANSI) standard for relational database language, in Version 6 SAS[®] software, it is beneficial that SAS system designers and developers become proficient in relational database design theory. This knowledge will enable us to better design relational databases and tables in order to facilitate the efficient querying of the same by our users. In this paper some design issues regarding relational databases will be discussed.

RELATIONAL THEORY AND DESIGN

Relational theory was developed in the late 1960s and published in 1970 by Edgar F. Codd, then an IBM researcher. This theory assists in the design of relational databases and in the efficient processing of user requests for information from the database.

The following four basic premises can be derived from Codd's 12 established rules of relational theory:

1. **TABLES** are used to store information in a database.
2. The tables can be related to one another in different manners to represent various data relationships and views by using **RELATIONAL ALGEBRA**.
3. **NORMALIZATION** provides a standard methodology, which yields a consistent approach to relational database design.
4. **DATA CONSTRAINTS** limit the type of data and functionality of data model attributes.

A relational database consists of a collection of tables, each of which is assigned a unique name. Each table has rows and columns. A row in a table represents a relationship among a set of values. Since a table is a collection of such relationships, there is a close correspondence between the concept of table and the mathematical concept of relation, from which the relational data model takes its name.

Structured Query Language (SQL) implements the basic relational algebra operators through the **select**, **from**, and **where** clauses. The result of a SQL query is a relation. Following is a simple query using the **EXGABNOR** sample table found in Figure 1. "Display the trial, patient number, and EKG abnormality of all patients with an EKG abnormality code that is between 100 and 200 inclusive."

```
proc sql;
  select trial, ptne, ekgabn;
  from ekgabnor;
  where ekgabn between 100 and 200;
```

The result of this SQL query is the following relation:

TRIAL	PTNO	EKGABN
1	2	105
1	2	102

In order to describe the structure of a database, we need to define the concept of a data model. A data model is a collection of conceptual tools for describing data, data relationships, and consistency constraints. Any system's file structure is a model of data that pertains to the part of the real world relevant to the application.

Designers and developers can reap many benefits from good data modeling. It makes code more straightforward, greatly reduces development and maintenance time, can improve system performance, and protects data integrity. It also gives the system greater durability in the face of inevitable changes, allowing for a longer useful lifetime.

A poorly designed data structure, however, causes trouble early in the development process and it keeps creating problems for as long as the installed system is

maintained. At best, it makes code hard to understand, difficult to maintain, and inefficient to run. At worst, we may get well into the development process before realizing that the system is incapable of producing some required output elements.

ENTITY-RELATIONSHIP MODEL

One of the primary modeling techniques is that of the entity-relationship approach. The others include data flow diagrams and Hierarchic process diagrams as well. In this paper the entity-relationship model will be discussed since it has gained acceptance as an appropriate data model for database design and is widely used in practice. Using this approach, designs can be built around the identification, analysis, description, and relationships of real-world functions, processes, and data entities of the business. The E-R model is a model comprised of the following three types of constructs:

1. entities and attributes (data items)
2. entities relationships
3. attribute constraints or domain

An entity is a person, place, or thing (real or abstract) about which we store data. Examples of entity types are PATIENT, VITAL SIGNS, MONITOR CONTACT, MEDICATION, THERAPEUTIC AREA, ADVERSE EVENT, DRUG INVENTORY LOCATION, DELIVERY SCHEDULE, etc. An individual entity will become a row in a relational table. An entity set is a set of entities of the same type. This will become a table of a relational database. For example, the set of all patients with EKG abnormalities are defined as the entity set **EXGABNOR** in Figure 1. Similarly, the set of all abnormality codes and their descriptions can be defined as the entity set **ABCODES**, for example.

Attributes are the smallest units of data in the data model. These will become the columns in the database tables. An entity is represented by a set of attributes. Attributes of the **EXGABNOR** entity are trial, patient-number, visit, and EKG abnormality code. Possible attributes of the **ABCODES** entity could be EKG abnormality code and EKG abnormality description.

A relationship is an association among several entities. Each entity or table may be related to any or all of the other tables in any number of ways. For example, we may define a relationship set **VITEKGAB** to denote the association between the table in Figure 2, **VITALS**, patients vital signs and the table in Figure 1, **EXGABNOR**, patients EKG abnormalities. The association is depicted in Figure 3. The SAS SQL code to generate this association or join is as follows:

```
proc sql;
  select *
  from vitals v, ekgabnor e
  where v.protocol=e.protocol
  and v.trial=e.trial
  and v.ptno=e.ptne
  and v.visit=e.visit;
```

For each data element or attribute there is a set of allowed values, called the domain of that attribute. NOT NULL (NN), NUMERIC (N), CHARACTER (C), PRIMARY KEY (PK), etc.

The effectiveness of the E-R approach in defining structures is constrained by the following three factors:

1. entity identification
2. entity definition
3. entity relationships identification and definition

First, the analyst or designer must recognize the various entities and determine why they are of interest to the organization and then name them. The specific identification of an entity has meaning only within the context of the organization. It is whatever the business defines it to be, and that definition must make sense within the context of the organization.

Second, it must be determined as to which attributes of each of the identified entities are needed by the organization and why. Make a tentative list of all the attributes that might conceivably be relevant to each of the identified entities. Review the list carefully.

When the right attributes are matched up with the right entities, figure out how each attribute might best be represented as one or more columns in the table associated with its entity. For example, we need to carefully consider the primary key for each entity in that we need to identify one that will uniquely identify each row of the entity (for example, patient number concatenated with protocol and trial).

And finally, the analyst or designer must identify and define the relationships which exist between the identified and defined entities and their relative importance to the organization, noting the primary and secondary keys that may be necessary to represent these relations. These relationships can be difficult to define since the relationships are dynamic depending upon the situational requirements.

The above is an iterative process. Deciding what entities and attributes are relevant to our application is a very important and creative aspect of system analysis and design. It helps a lot to know the business or industry in which you are working, and it helps even more to have a broad understanding about how your potential users think.

DATABASE NORMALIZATION

The relational model provides a number of powerful analytic tools that offer invaluable assistance in designing and optimizing the data structures for our applications. The tools are called the normal forms, and the process of applying them is called normalization.

Normalization is a formal, systematic procedure for testing your tentative data structure against a series of well-defined rules that derive logically from the mathematical theory underlying the relational data model. It insures that the data model is in the most stable (i.e., least likely to change) form. If we use the rules of normalization to create data structures, we are likely to save a lot of work in the development process.

Four important properties are associated with a table that represents a relation in a database:

1. No two rows can be identical.
2. The ordering of these rows is insignificant.
3. The ordering of the columns is insignificant.
4. Each data item should be atomic (i.e., nondecomposable, such as an integer or character string).

When these four properties exist, the relation is called normalized. The goal of normalization is to minimize the potential for data maintenance anomalies and redundancies.

The normalization process consists of sequential steps for refining the data model. A design, which has been refined through the first step is said to be in first normal form. If the second step is completed, the design is in the second normal form. The first, second, and third normal forms yield substantial practical benefits.

Complying with the first normal form requires elimination of repeating columns or groups of columns from each table. All columns in a table must be related to the primary key of that table in a one-to-one manner. In Figure 4 the primary key of the DRUG table is PROTOCOL. The table is in violation of the first normal form rule since each protocol may have more than one investigator. PROTOCOL and DRUG is repeated for PROTOCOL 999.

As illustrated in Figure 4, violating the first normal form rule may cause data redundancy since entire rows must be repeated to allow for multiple values of the incorrectly placed column; inability to completely identify what the incorrectly placed column represents; and maintenance difficulties (i.e., if adding a new investigator means adding a new row, all the protocol's attributes (e.g., drug name) must be completely reentered. It becomes apparent that investigator name is not an attribute of a drug and thus INVESTNAME should not be a column in the DRUG table.

Figure 5 shows the DRUG table restructured to correct for the first normal form violation by deleting the INVESTNAME column. Of course, an investigator entity would have to be created with its appropriate attributes.

The second degree of normalization involves tables that have composite primary keys (PK) (i.e., primary keys that are constructed by concatenation of two or more secondary keys). Violating the second normal form rule may cause: data redundancy due to one or more columns containing frequently repeating data values; maintenance

difficulties; and erroneously combining an entity table and a relation table into a single table.

As illustrated in Figure 6, the DRUGNAME is an attribute of the PROTOCOL rather than of the TRIAL within the PROTOCOL. Thus it is dependent only upon PROTOCOL rather than upon the concatenation of both PROTOCOL and TRIAL. But the STATUS column denotes the status of the TRIAL within the PROTOCOL for the study. It is dependent upon the entire primary key and it is legitimate in this table. See Figure 7 for a possible solution to this problem.

The third form of normalization involves eliminating the dependence of any non-key field upon any other fields except the primary key. No two non-primary key columns in a table may be related in a one-to-one manner (transitive dependence). Violating the third normal form will cause the same symptoms as when violating the first and second normal forms.

As illustrated in Figure 8, the STUDYTEAM table, the problem here is that we have two non-key fields dependent upon each other rather than upon the primary key field, EMPLOYEE-CODE. TEAM and LEADER-CODE are transitively dependent upon one another. Making a change to either of these two fields without simultaneously making appropriate change to the other may introduce anomalies into our database.

One possible solution is to re-structure the table into multiple tables as in Figure 9.

We could go on normalizing to the nth form. But further normalizing is impractical and unnecessary. For more detailed normalization research, I would recommend that the reader refer to 'A Relational Model of Data for Large Shared Data Banks' by Edgar F. Codd published in Communications of the ACM, Vol 13 No. 6, June 1970, 'Relational Databases: The Real Story' by Steven J. Vaughan-Nichols in BYTE, Vol 15 No. 14, December 1990, and 'Interfacing Normalized Relational Database Structures With SAS Software' by James R. Johnson and Roger D. Cornejo of Glaxo published in SAS Users Group International 15th Annual Conference proceedings (pp.421-427).

CONCLUSION

In a relational database, even the most complicated data relationships can be reduced to two-dimensional table formats via the data normalization process. This process makes changing and displaying information easier than it is under other data models.

But, normalization is not a panacea! It will not validate the selection of entities in the data model; detect redundant entities, or validate entity relationships. What practical normalization will do is minimize data maintenance anomalies; minimize data redundancy; ensure that the internal table structures are in the most stable form; and sometimes detect multiple entities which were erroneously combined into one entity in the data model (third normal form violation).

In general, the goal of a relational database design is to generate a set of relation schemes that allow us to store information without unnecessary redundancy, yet allow us to retrieve information easily. With the use of SQL, end-users would now be able to access data in an ad hoc way.

Acknowledgements

I wish to thank Gail Scherer and Ira J. Mohebalian for their technical review of this paper.

SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA. ORACLE is a registered trademark of Oracle Corporation.

To contact the author: Faith R. Sloan
313)996-7103
Parke-Davis Pharmaceutical
Research Division
Clinical Programming Dept
2800 Plymouth Road
Ann Arbor, Michigan 48106

EKGABNOR TABLE

PROTOCOL	TRIAL	PTNO	VISIT	EKGABN
N,NN,PK	N,NN,PK	N,NN,PK	N,NN,PK	N
999	1	1	3	230
888	1	2	3	105
999	1	2	3	102
999	2	1	4	609

Figure 1 Sample EKG Abnormality Table

VITALS TABLE

PROTOCOL	TRIAL	PTNO	VISIT	DBP	SBP	HRT	WGT
N,NN,PK	N,NN,PK	N,NN,PK	N,NN,PK	N	N	N	N
999	1	1	0	82	142	72	200
999	1	1	3	88	132	80	195
999	1	2	3	96	138	68	155

Figure 2 VITALS Entity Set or Table

VITEKGAB

PROTOCOL	TRIAL	PTNO	VISIT	DBP	SBP	HRT	WGT	EKGABN
N,NN,PK	N,NN,PK	N,NN,PK	N,NN,PK	N	N	N	N	N
999	1	1	3	88	132	80	195	230
999	1	2	3	96	138	68	155	102

Figure 3 Relationship between VITALS and EKGABNOR tables

DRUG TABLE

PROTOCOL	DRUGNAME	INVESTNAME
N,NN,PK	C	C
888	Primon	Doe
999	Bacrinol	Stein
999	Bacrinol	Johnson
999	Bacrinol	Miller

Figure 4 Violation of First Normal Form Rule

DRUG TABLE

PROTOCOL	DRUGNAME
N,NN,PK	C
888	Primon
999	Bacrinol

Figure 5 Correction of Violation of First Normal Form Rule

TRIAL TABLE

PROTOCOL	TRIAL	DRUGNAME	STATUS
N,NN,PK	N,NN,PK	C	C
888	1	Primon	CONT
888	2	Primon	WITHDR
999	1	Bacrinol	CONT
999	2	Bacrinol	CONT

Figure 6 Violation of Second Normal Form Rule

TRIAL TABLE

PROTOCOL	TRIAL	STATUS
N,NN,PK	N,NN,PK	C
888	1	CONT
888	2	WITHDR
999	1	CONT
999	2	CONT

Figure 7 Correction of Violation of Second Normal Form Rule

STUDY TABLE

EMPLOYEE-CODE	TEAM	LEADER-CODE
C,NN,PK	C,NN	C
426111817	A888	359111111
357888222	B888	426111817
111224444	Z921	431445555
999999999	Z999	888998888

Figure 8 Violation of Third Normal Form Rule

STUDY TABLE

STUDY-CODE	STUDY-NAME	LEADER-CODE
C,NN,PK	C	C
111	Tacrine1	426111817
222	Pirmenol1	222334444
333	Tacrine2	888998888
444	Quinapril1	345678901

STUDY-TEAM TABLE

EMPLOYEE-CODE	STUDY-CODE
C,NN,PK	C,NN,FK
426111817	111
357888222	111
111224444	333
999999999	444

Figure 9 Correction of Violation of Third Normal Form Rule