Hi! Welcome to the fascinating world of DBMS. Here in this lecture we are going to discuss about the Relational Model.

We have discussed the E-R Model, a technique for building a logical model of an enterprise. Logical models are high level abstract views of an enterprise data. In building these models little thought is given to representing or retrieving data in the computer. At the next lower level, a number of data models are available that provide a mapping for a logical data model like the E-R model. These models specify a conceptual view of the data as well as a high level view of the implementation of the database on the computer. They do not concern themselves with the detailed bits and bytes view of the data.

What is relational model?

The relational model was proposed by E. F. Codd in 1970. It deals with database management from an abstract point of view. The model provides specifications of an abstract database management system. The precision is derived from solid theoretical foundation that includes predicate calculus and theory of relations. To use the database management systems based on the relational model however, users do not need to master the theoretical foundations. Codd defined the model as consisting of the following three components:

Data Structure - a collection of data structure types for building the database.

Data Manipulation - a collection of operators that may be used to retrieve, derive or modify data stored in the data structures.

Data Integrity - a collection of rules that implicitly or explicitly define a consistent database state or changes of states.

Data Structure

Often the information that an organization wishes to store in a computer and process is complex and unstructured. For example, we may know that a department in a university has 200 students, most are full-time with an average age of 22 years, and most are females. Since natural language is not a good language for machine processing, the information must be structured for efficient processing. In the relational model the information is structures in a very simple way.

The beauty of the relational model is its simplicity of structure. Its fundamental property is that all information about the entities and their attributes as well as the relationships is presented to the user as tables (called *relations*) and nothing but tables. The rows of the tables may be considered records and the columns as fields. Each row therefore consists of an entity occurrence or a relationship occurrence. Each column refers to an attribute. The model is called relational and not tabular because tables are a lower level of abstractions than the mathematical concept of relation. Tables give the impression that positions of rows and columns are important. In the relational model, it is assumed that no ordering of rows and columns is defined.

We consider the following database to illustrate the basic concepts of the relational data model.

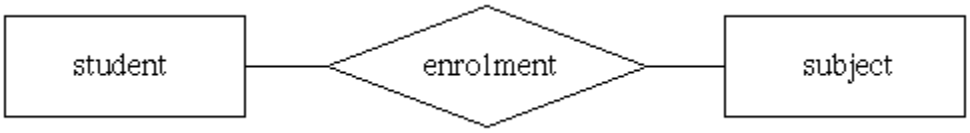

Figure 3.1 A simple student database

We have not included the attributes in the above E-R diagram. The attributes are student id, student name and student address for the entity set student and subject number, subject name and department for the entity set subject.

The above database could be mapped into the following relational schema which consists of three relation *schemes*. Each relation scheme presents the structure of a relation by specifying its name and the names of its attributes enclosed in parenthesis. Often the primary key of a relation is marked by underlining.

student(student_id, student_name, address)
enrolment(student_id, subject_id)
subject(subject_id, subject_name, department)

An example of a database based on the above relational model is:

student_id	student_name	address
8656789	Peta Williams	9, Davis Hall
8700074	John Smith	9, Davis Hall
8900020	Arun Kumar	90, Second Hall
8801234	Peter Chew	88, Long Hall
8654321	Reena Rani	88, Long Hall
8712374	Kathy Garcia	88, Long Hall
8612345	Chris Watanabe	11, Main Street

Table 1. The relation student

student_id	subject_id
8700074	CP302
8900020	CP302
8900020	CP304
8700074	MA111
8801234	CP302
8801234	CH001

Table 2. The relation enrolment

subject_id	subject_name	department
CP302	Database Management	Comp. Science
CP304	Software Engineering	Comp. Science
CH001	Introduction to Chemistry	Chemistry
PH101	Physics	Physics
MA111	Pure Mathematics	Mathematics

Table 3. The relation subject

We list a number of properties of relations:

Each relation contains only one record type.

Each relation has a fixed number of columns that are explicitly named. Each attribute name within a relation is unique.

No two rows in a relation are the same.

Each item or element in the relation is atomic, that is, in each row, every attribute has only one value that cannot be decomposed and therefore no repeating groups are allowed.

Rows have no ordering associated with them.

Columns have no ordering associated with them (although most commercially available systems do).

The above properties are simple and based on practical considerations. The first property ensures that only one type of information is stored in each relation. The second property involves naming each column uniquely. This has several benefits. The names can be chosen to convey what each column is and the names enable one to distinguish between the column and its domain. Furthermore, the names are much easier to remember than the position of the position of each column if the number of columns is large.

The third property of not having duplicate rows appears obvious but is not always accepted by all users and designers of DBMS. The property is essential since no sensible context free meaning can be assigned to a number of rows that are exactly the same.

The next property requires that each element in each relation be atomic that cannot be decomposed into smaller pieces. In the relation model, the only composite or compound type (data that can be decomposed into smaller pieces) is a relation. This simplicity of structure leads to relatively simple query and manipulative languages.

A relation is a set of tuples. As in other types of sets, there is no ordering associated with the tuples in a relation. Just like the second property, the ordering of the rows should not be significant since in a large database no user should be expected to have any understanding of the positions of the rows of a relation. Also, in some situations, the DBMS should be permitted to reorganize the data and change row orderings if that is necessary for efficiency reasons. The columns are identified by their names. The tuple on the other hand are identified by their contents, possibly the attributes that are unique for each tuple.

The relation is a set of tuples and is closely related to the concept of relation in mathematics. Each row in a relation may be viewed as an assertion. For example, the relation *student* asserts that a student by the name of *Reena Rani* has *student_id 8654321* and lives at *88, Long Hall*. Similarly the relation *subject* asserts that one of the subjects offered by the Department of Computer Science is *CP302 Database Management*.

Each row also may be viewed as a point in a n-dimensional space (assuming n attributes). For example, the relation *enrolment* may be considered a 2-dimensional space with one dimension being the *student_id* and the other being *subject_id*. Each tuple may then be looked at as a point in this 2-dimensional space.

In the relational model, a relation is the only compound data structure since relation does not allow repeating groups or pointers.

We now define the relational terminology:

Relation - essentially a table
Tuple - a row in the relation
Attribute - a column in the relation
Degree of a relation - number of attributes in the relation
Cardinality of a relation - number of tuples in the relation
N-ary relations - a relation with degree N
Domain - a set of values that an attribute is permitted to take. Same domain may be used

by a number of different attributes.

Primary key - as discussed in the last chapter, each relation must have an attribute (or a set of attributes) that uniquely identifies each tuple.

Each such attribute (or a set of attributes) is called a *candidate key* of the relation if it satisfies the following properties:

(a) The attribute or the set of attributes uniquely identifies each tuple in the relation (called *uniqueness*), and

(b) if the key is a set of attributes then no subset of these attributes has property (a) (called *minimality*).

There may be several distinct set of attributes that may serve as candidate keys. One of the candidate keys is arbitrarily chosen as the *primary key* of the relation.

The three relations above *student*, *enrolment* and *subject* have degree 3, 2 and 3 respectively and cardinality 4, 6 and 5 respectively. The primary key of the relation *student* is *student_id*, of relation *enrolment* is *(student_id, subject_id)*, and finally the primary key of relation *subject* is *subject_id*. The relation *student* probably has another candidate key. If we can assume the names to be unique than the *student_name* is a candidate key. If the names are not unique but the names and address together are unique, then the two attributes *(student_id, address)* is a candidate key. Note that both *student_id* and *(student_id, address)* cannot be candidate keys, only one can. Similarly, for the relation *subject*, *subject name* would be a candidate key if the subject names are unique.

The definition of a relation presented above is not precise. A more precise definition is now presented. Let $D_1, D_2, \ldots, D_n$ be n atomic domains, not necessarily distinct. R is then a degree n relation on these domains, if it is a subset of the cartesian product of the domains, that is, $D_1 X D_2 X \ldots X D_n$. If we apply this definition to the relation *student*, we note that *student* is a relation on domains *student_id* (that is, all permitted *student_id*'s), *student_name* (all permitted student names) and *address* (all permitted values of

addresses). Each instance of the relation *student* is then a subset of the product of these three domains.

We will often use symbols like *R* or *S* to denote relations and *r* and *s* to denote tuples. When we write $r \in R$, we assert that tuple *r* is in relation *R*.

In the relational model, information about entities and relationships is represented in the same way i.e. by relations. Since the structure of all information is the same, the same operators may be applied to them. We should note that not all relationship types need to be represented by separate relations. For 1:1 and 1:m relationships, key propagation may be used. Key propagation involves placing the primary key of one entity within the relation of another entity. Many-to-many relationships however do require a separate relation to represent the association.

We illustrate the use of key propagation by considering the following E-R model:

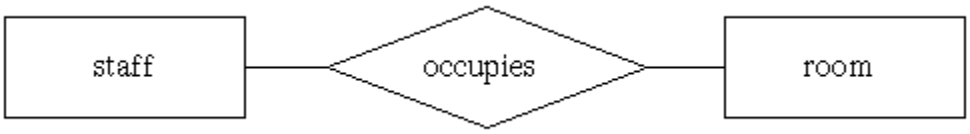

Figure 3.2 a simple database

Let the relationship *occupies* be many-to-one, that is, one staff may occupy only one room but a room may be occupied by more than one staff. Let the attributes of the entity *staff* be staff number (*s_num*), staff name (*name*) and *status*. Attributes of the entity *room* are room number (*r_num*), *capacity*, *building*. *occupies* has no attributes. *s_num* and *r_num* are the primary keys of the two entities *staff* and *room*.

One way to represent the above database is to have the following three relations:

staff(s_num, name, status)
occupies(s_num, r_num)
room(r_num, capacity, building)

There is of course another possibility. Since the relationship *occupies* is many-to-one, that is, each staff occupies only one room, we can represent the relationship by including the primary key of relation *room* in the entity *staff*. This is propagation of the key of the entity *room* to entity *staff*. We then obtain the following database:

staff(s_num, name, status, r_num)
room(r_num, capacity, building)

We should note that propagating the key of the entity *staff* to the entity *room* would not be appropriate since some rooms may be associated with more than one staff member. Also note that key propagation would not work if the relationship *occupies* is many-to-many.

Data Manipulation

The manipulative part of relational model makes set processing (or relational processing) facilities available to the user. Since relational operators are able to manipulate relations, the user does not need to use loops in the application programs. Avoiding loops can result in significant increase in the productivity of application programmers.

The primary purpose of a database in an enterprise is to be able to provide information to the various users in the enterprise. The process of querying a relational database is in essence a way of manipulating the relations that are the database. For example, in the student database presented in Tables 1-3, one may wish to know

names of all students enrolled in CP302, or

names of all subjects taken by John Smith.

We will show how such queries may be answered by using some of several data manipulation languages that are available for relational databases. We first discuss the two formal languages that were presented by Codd. These are called the *relational*

algebra and *relational calculus*. These will be followed by a discussion of a commercial query language SQL.

Questions

1. What do you mean by relational model?
2. Explain what is a data structure?
3. Explain what do you mean by data manipulation?

References

1. Codd, E. F. (1974), "A Relational Model of Data for Large Shared Data Banks", Comm. ACM, Vol. 13, No. 6, pp. 377-387.
2. Codd, E. F. (1974), "Recent Investigations in Relational Data Base Systems", in Information Processing 74, North Holland.
3. Codd, E. F. (1982), "Relational Database: A Practical Foundation for Productivity", in Comm. ACM, Vol. 25, No. 2, pp. 109-117.
4. Codd, E. F. (1981), "The Capabilities of Relational Database Management Systems", IBM Report RJ3132.