SWE 212 Analysis of Programming Languages

Week 7 Summary of '*A Relational Model of Data for Large Shared Data Banks*' by E.F. Codd

Samyak Jhaveri

This paper not only lays the foundation for the relational database model of large-scale data organization, but also specifies the abstract model of data representation. The author starts by highlighting the problems with existing data storage systems of the time of writing this paper - data models that were dependent on the way data was organized on the machine, and the changeability of data models based on the changes made to the queries and updates done to the data. The author calls this idea data independence - the idea that suggests that the applications programs and terminal activities be independent of changes made to the internal data representation and vice versa. The relational model of data enables a logical approach to derivability, redundancy and consistency of relationships among the data. Further, a relational view/ model also allows for a better evaluation of the scope and logical imitations of the data system.

Data description tables facilitate changing of certain characteristics of the data representation of the data bank, but at the cost of logically impairing some application programs that use the data representation for their tasks. This issue arose from the fact that these application programs were doused in details of how the data was to be stored physically represented on to the machine. To dig deeper into these issues, the author describes three kinds of data dependencies: Ordering Dependence, Indexing Dependence, and Access Path Dependence.

I. **Ordering Dependence** - In some data representation models, the data items may be stored in a particular order, for example by the order in which they are stored physically in the machine. This May seem to be a reasonably fair approach to ordering data, however, such an application could fail terribly if the ordering is changed later on by another ordering.

II. **Indexing Dependence** - Indexes are thought of as tools to improve query response and updates, while slowing down response to insertions and deletions. In a system that uses data banks with indices, it would be necessary to have the functionality for creating and destroying the indexes of the data representation as needed to accommodate for the changing activity patterns in the application program. The question arises whether the application programs and terminal activities would have to change as the indexes of a database management model are updated or changed? Such a dependency on the indexes of a database representation is called Indexing dependency. During the time of writing this paper, the indexes of the systems had to be manually referred t o in the queries themselves - these were called indexing chains. If these chains were changed later on, the application programs will not work correctly.

Iii. **Access Path Dependency** - Since most of the data representation system at the time of this paper used a tree or a network model of data representation, the application programs tended to be logically imparied when these trees or networks were changed in any way in the data bank as they followed a manually specified path along the tree or the network and changing the tree/network would change the path entirely. This is known as Access Path Dependency.

The author proposes the idea of the relational view of data. The term relation is in the same context as used in mathematics. A relation R is defined on n Sets S1 through Sn such that it is a set of n tuples each

having its first element from S1, its second element from S2 and so on. R is of degree which could be the value 1 (unary), 2 (binary), 3 (tertiary), or n (n-ary). The intuitiveness of this model is part of what led to its widespread acceptance and success.
A relation R has the following properties:

1) Each row represents an n-tuple of R
2) The ordering of rows is immaterial
3) All rows are distinct
4) The ordering of columns is significant - it corresponds to the ordering S1, S2, S3, …, Sn of the domains on which R is defined.
5) The significance of each column is partially conveyed by labeling it with the name of the corresponding domain.

In this original model of relations, the ordering of relations matters because you could have two columns from the same domain but they mean slightly different things in each column. Given this definition of relations, all the data in a database can essentially be viewed as a set of relations that vary over time. To simplify, relationships are the unordered outer parts of relations, so that is where the order of columns does not matter.

In many commercial dtaa banks, the user was burdened with remembering and maintaining the harmony of the relations between the data that could have upto 30 degrees. The relational view/model proposes that the user should only be concerned with dealing with the relationships between the data that are domain-unordered counterparts of the domain-ordered relations. Therefore, to be able to uniquely identify domains, they should be qualified with a role name based on their role in the data bank.

Summarizing what the author proposes the relational view should address - the users should interact with a relational model of the data consisting of a collection of time-varying relationships, rather than relations. Each user does not need to know more about any relationship than its name and its domains' names. This information may be presented to the user in menu style byt the system.

Defining some important terms that are commonly used in modern database management systems, the author explains some concepts about the relational model.

1. **Primary key** - A domain or a combination of domains of a given relation has values that uniquely identifies each element of that relation. Note that a primary key is non-redundant i.e. the participating domains are not surplus in uniques identifying each element
2. **Foreign Key** - A domain or a combination of domains of a relation R that is not the primary key of that relation R but its elements are values of the primary key of some other relation is called a foreign key.

Each domain need not be a simple, atomic domain as it is possible to have non-atomic simple values at the heart of a column. This means that relations and their tables can be nested.

Next, the author discusses normalization and the normal form. It is important to be able to eliminate non simple domains of some complicated relations, thus the ability to un-nest a nested table into a simple 2-d table. Consider a table that has nested relations and these relations need to be expressed as a simple 2-D table with atomic domains. To normalize this, all the user has to do is to traverse the nested tree of tables and append the primary key of the parent to the child.

Normalization of relations in normal form is beneficial as it helps better manage storage of the data by reducing redundancy and also aids in communicating bulks of data between systems that use different data representations. The communication form would have the following benefits:

1. It would ot have pointers (so less or no dependence on addressing
2. No dependence on hash-addressing schemes
3. No indices or ordered list.

Now that the author has defined what data can look like when expressed in a relational model, he turns to the issue of how to query that data using a data language. Having a relational model of data makes it possible to have data languages based on predicate calculus which can be used to read and manipulate this data. The paper further specifies some of the high-level properties a data language should have to query the relational model of data. The most crucial property of such a language is its descriptive ability. The language should be able to abstract away how the data from the data representation is obtained, calculated and how the query computes on the data.However, despite this abstraction, the language should be descriptive enough for the user to be able to specify exactly what he/she wants from the data and how it needs to be laid out.

Operations on Relations

1. **Permutation** - A permutation of a relation is simply another relation with the columns permuted.
2. **Projection** - Projection is the operation that takes a relation and extracts a relation which consists of a subset of the domains, or a subset of the columns of that relation. If we take a list of indices that is k long is projected onto a relation n-ary relation R, then the projection of R on l is then the k-ary relation whose jth column is just the jth index from l and we remove all duplication from the resulting rows.
3. **Join** - If we have two binary relations R and S, the join of R and S is defined as a relation which has three tuples a, b, c in which the tuple a is from the relationship R and b and c are from the relationship S, so the element b is in common. When these two relations R and S are joined, the resulting table has all the tuples of both the relations with the common element as the point of joining.
4. Restriction - A restriction is the way in which we can use a relation S to restrict a relation R and arrive at a subset.

Redundancy

In order to talk about redundancy, the notion of derivability first needs to be discussed.
Suppose θ is a collection of operations on relations, it can be said that a relation R is θ-derivable from a set S of relations if there exists a sequence of operations from the collection θ which, for all time, yields R from members of S. A set of relations has strong redundancy if it has a projection which is derivable from other projects of other relations in the set. Strong redundancy exists in the named set of relationships for user convenience. This allows old programs to be able to refer to obsolete relationships by name and run correctly. On the other hand, there exists the concept of weak redundancy that isn't characterized by an equation. A weak redundancy is a phenomenon that happens when a relation of a collection of relations has a projection which is not drivable from other members but is all times a projection of some join of other projectsino of relations in the collection.

Consistency

Lastly, we look at the concept of consistency. Generally, if C is a set of time varying relations and Z is a set of constraint statements , we can take an instantaneous snapshot in time from the relation C and assee if V satisfies the constraints in Z and if it does, we call it consistent. It is challenging to actually check for consistency because snapshots of relations could be really large. There are many ways in which one could check for consistency, one of which is to do them at runtime. Each time you perform an operation, check if it's still consistent with the properties you want to enforce. This would for sure slow these operations as there is more computation happening during each of them. The alternative method is to check for consistency periodically as a batch operation.

To summarize, this paper defined an abstract relational model of data which was data independent - which means that the way that the data was read and processed was independent of the physical layout of the data in the machine’s storage. This paper also defined the operations that could take place on top of this data expressed in a relational model.