Distributed Database Design –
Top-Down Design

Ileana ŞTEFAN and Maricel POPA

Abstract: The design of distributed databases is an optimization problem requiring solutions to several interrelated problems: data fragmentation, allocation, and local optimization. Each problem can be solved with several different approaches thereby making the distributed database design a very difficult task.

1. Introduction

The creation of a distributed database system, which reflects the desired structure, permits data access form all the units and stores data in the vicinity of the sites that will use it most, has to improve the portioning of data and to offer easy and fast access for all the users, no matter where they are situated.

The distributed systems must aid resolving the data isles issues. Sometimes, databases are considered as electronic isles, which point to distinct and inaccessible places, similar to far-off islands. This can be a result of geographical division, incompatible architectures, incompatible communication protocols, etc.

Those that work with RDBMS software might ask themselves why they should worry about designing the databases, noticing that the majority of RDBMS suites come with sample databases that can be copied or modified according to necessities. These tables can be taken form the sample databases and used in a user-defined database. Some programs provide tools for the users to help them in creating and defining the tables. Nevertheless, these instruments don’t actually contribute to the design of the database; they merely help for the physical creation of the tables that will be part of a database.

The reason for which databases must be well designed is that the design has an important role in maintaining the integrity and the accuracy of the data stored in a database. If the database is not carefully designed, it would be hard to find all types of information, and some of them might be erroneous. Broken data represent the worst outcome of a badly designed database.

Design methods. The strategies for designing a distributed database, as in the case of centralized databases, are:

- Top-down design, which aims for an optimal distribution of data. Homogenous systems are usually a result of using this design;
- Bottom-up design, which is used for local databases that need to be integrated into an unitary system;
- Mixed design, which uses both methods. This design suits practical cases best.

2. Top-down design

This design is a process of creating data models that contain high-level entities and relationships, to which successive refinements are applied, in order to identify the corresponding low-level entities, relationships and attributes. The top-down approach is illustrated by using the concepts of the entity-relationship model.

The top-down approach consists of taking the following steps:
• Analyzing requirements;
• View integration and conceptual design;
• Data distribution design;
• Local physical schema design.

Analyzing requirements is the stage in which the database users are defined, as well as the work environment and the information processing requirements. Starting with these requirements the view design and the conceptual schema will be created next. The visualization activity aims to define the interfaces for the end users. The conceptual design consists of defining the entity types and the relationships between them (entity analysis), followed by acknowledging the activities that the organization undertakes (functional analysis). Next is the data distribution stage, the output of which is the local conceptual schemas obtained by distributing the entities on the different stations of the distributed systems. The distribution of relationships is preceded by the creation fragmentation model of these relationships. As a result, the data distribution design consists of two steps: fragmentation of the relationships of the global conceptual schema and allocation of the fragments attained.

The fragmentation process consists of splitting a global relationship into a number of sub relationships, called fragments, which can be stored optimally in the distributed database. The fragmentation must maintain the semantic coherence of the database, must make a split without information loss and duplicates, and to permit the rebuild of the initial relationship from its fragments. There are three types of fragmentation: horizontal, vertical, and mixed. The fragments are allocated to one or more sites. The fragments can be replicated, in order to obtain high availability and performance.

Fragment definition and allocation should depend on how the database will be used. This implies analysis of the applications. Practical experience saw that 20% of the most used queries represent 80% of the total data accesses, and this rule can be used as an indicator when the analysis is performed.

The design process should depend on quantity information, as well as on quality information. Quality information is used in the fragmentation process, as quantity information is used in the allocation process. Quality information refers to transactions run by the application (including relationships, attributes and tuples), access time (read or write) and predicates of read operations.

Quantity information refers to the frequency at which an application is used, the site where the application is run from and the criteria for transaction and application performance.

The degree at which a database should be fragmented has an effect on query execution performance. Fragmentation degree varies form one extreme, which is no fragmentation, to the other, where fragmentation should be applied at tuple or attribute levels. In practice, a reasonable degree of fragmentation should be used.

After the database is fragmented, the allocation of fragments on the network stations follows. The designer of the database must decide if the fragments would be replicated, and their degree of replication. The fragments can be multiplied, or kept into one copy. Data replication increases the fault-tolerance of the system and the speed of data retrieval.

Top-down design can be illustrated in a schema as in the figure below:

3. Bottom-up design

This approach starts from the fundamental level of attributes, which are grouped in relationships that represent entity types and associations between them, as a result of analyzing associations between attributes. The normalization process constitutes a bottom-up approach. Normalization implies identifying attributes and placing them in normalized tables, based on functional dependences between attributes.
Bottom-up modeling is indicated to be used for systems where databases already exist, but have to be integrated in a single database. The process consists of integrating local conceptual schemas into the global conceptual schema. This is equivalent in means to correlating the common data descriptions and resolving the conflicts that may appear. If the existing databases have used different Database Management Systems, it is possible that they also use different DBMS after integration. As a result, a heterogeneous system adds the necessity of translating between different representations of data, besides the complexity of data integration.

Bottom-up approach consists of taking the following steps:
- Choosing a common data model for describing the global schema;
- Translation of each local scheme into a common data model;
- Integration of the local schemas into a global schema.
References

6. IBM Corporation – Distributed Relational Database Architecture Reference
9. Oracle 8i - Distributed Database System, 1999
11. Ileana Popescu – Modelarea bazelor de date, Editura Tehnică, București, 2001
12. Ileana Popescu – Prelucrarea avansată a informațiiei Oracle8, Editura tehnică, București, 1999

Ileana ȘTEFAN
“Petru Maior” University Of Târgu Mureș

Maricel POPA
OMEGA-Tehnoton Iași