Federated database system

A federated database system is a type of meta-database management system (DBMS), which transparently integrates multiple autonomous database systems into a single federated database. The constituent databases are interconnected via a computer network and may be geographically decentralized. Since the constituent database systems remain autonomous, a federated database system is a contrastable alternative to the (sometimes daunting) task of merging together several disparate databases. A federated database, or virtual database, is the fully integrated, logical composite of all constituent databases in a federated database system.

McLeod and Heimbigner[1] were among the first to define a federated database system, as one which "define[s] the architecture and interconnect[s] databases that minimize central authority yet support partial sharing and coordination among database systems". [1]

Through data abstraction, federated database systems can provide a uniform user interface, enabling users and clients to store and retrieve data in multiple noncontiguous databases with a single query -- even if the constituent databases are heterogeneous. To this end, a federated database system must be able to decompose the query into subqueries for submission to the relevant constituent DBMS's, after which the system must composite the result sets of the subqueries. Because various database management systems employ different query languages, federated database systems can apply wrappers to the subqueries to translate them into the appropriate query languages.

Note: this description of federated databases does not accurately reflect the McLeod/Heimbigner[1] definition of a federated database. Rather, this description fits what McLeod/Heimbinger called a composite database. McLeod/Heimbigner's federated database is a collection of autonomous components that make their data available to other members of the federation through the publication of an export schema and access operations; there is no unified, central schema that encompasses the information available from the members of the federation.

Among other surveys, [2] defines a Federated Database as a collection of cooperating component systems which are autonomous and are possibly heterogeneous. The three important components of an FDBS as pointed out in [2] are autonomy, heterogeneity and distribution. Another dimension which has also been considered is the Networking Environment Computer Network, e.g., many DBSs over a LAN or many DBSs over a WAN update related functions of participating DBSs (e.g., no updates, nonatomic transitions, atomic updates). Master data management or MDM is a relatively new term for similar practices.

FDBS Architecture

A DBMS can be classified as either centralized or distributed. A centralized system manages a single database while distributed manages multiple databases. A component DBS in a DBMS may be centralized or distributed. A multiple DBS (MDBS) can be classified into two types depending on the autonomy of the component DBS as federated and non federated. A nonfederated database system is an integration of component DBMS that are not autonomous. A federated database system consists of component DBS that are autonomous yet participate in a federation to allow partial and controlled sharing of their data.

Federated architectures differ based on levels of integration with the component database systems and the extent of services offered by the federation. A FDBS can be categorized as loosely or tightly coupled systems.

- Loosely Coupled require component databases to construct their own federated schema. A user will typically access other component database systems by using a multidatabase language but this removes any levels of location transparency, forcing the user to have direct knowledge of the federated schema. A user imports the data they require from other component databases and integrates it with their own to form a federated schema.
- Tightly coupled system consists of component systems that use independent processes to construct and publicize an integrated federated schema.
Multiple DBS of which FDBS are a specific type can be characterized along three dimensions: Distribution, Heterogeneity and Autonomy. Another characterization could be based on the dimension of networking for e.g. single databases or multiple databases in a LAN or WAN.

**Distribution**

Distribution of data in an FDBS is due to the existence of a multiple DBS before an FDBS is built. Data can be distributed among multiple DB which could be stored in a single computer or multiple computers. These computers could be geographically located in different places but interconnected by a network. The benefits of data distribution help in increased availability and reliability as well as improved access times.

**Heterogeneous Database System/Heterogeneity**

Heterogeneities in databases arise due to several factors. Some of them occur due to differences in structures, semantics of data, the constraints supported or query language. Differences in structure occur when two data models provide different primitives such as object oriented (OO) models that support specialization and inheritance and relational models that do not. Differences due to constraints occur when two models support two different constraints. For example the set type in CODASYL schema may be partially modeled as a referential integrity constraint in a relationship schema. CODASYL supports insertion and retention that are not captured by referential integrity alone. The query language supported by a DBMS can also contribute to heterogeneity between other component DBMSs. For example differences in query languages with same data models or different versions of query languages could contribute heterogeneity.

Semantic heterogeneities arise when there is a disagreement about meaning, interpretation or intended use of data. At the schema and data level, some of the possible classification of Heterogeneities that occur are

- Naming Conflicts e.g. Databases using different names to represent the same concept.
- Domain Conflicts or Data Representation conflicts e.g. Databases using different values to represent same concept.
- Precision Conflicts e.g. Databases using same data values from domains of different cardinalities for same data.
- Metadata Conflicts e.g. same concepts are represented at schema level and instance level.
- Data Conflicts e.g. Missing attributes
- Schema Conflicts e.g. Table versus table conflict which includes naming conflicts, data conflicts etc.

In creating a federated schema, one has to resolve such heterogeneities before integrating the component DB schemas.

**Schema matching, schema mapping**

Dealing with incompatible data types or query syntax is not the only obstacle to a concrete implementation of an FDBS. In systems that are not planned top-down, a generic problem lies in matching semantically equivalent, but differently named parts from different schemas (=data models) (tables, attributes). A pairwise mapping between \( n \) attributes would result in \( \frac{n(n-1)}{2} \) mapping rules (given equivalence mappings) - a number that quickly gets too large for practical purposes. A common way out is to provide a global schema that comprises the relevant parts of all member schemas and provide mappings in the form of database views. Two principal solutions can be realized, depending on the direction of the mapping:

1. Global as View (GaV): the global schema is defined in terms of the underlying schemas
2. Local as View (LaV): the local schemas are defined in terms of the global schema

Both are explained in more detail in the article Data integration. Alternate approaches to the schema matching problem and a classification of the same are explained in more detail in the article Schema Matching.
Autonomy

Fundamental to the difference between an MDBS and an FDBS is the concept of autonomy. It is important to understand the aspects of autonomy for component databases and how they can be addressed when a component DBS participates in an FDBS. There are four kinds of autonomies addressed:

- Design Autonomy which refers to ability to choose its design irrespective of data, query language or conceptualization, functionality of the system implementation.
- Heterogeneities in an FDBS are primarily due to design autonomy.
- Communication autonomy refers to the general operation of the DBMS to communicate with other DBMS or not.
- Execution autonomy allows a component DBMS to control the operations requested by local and external operations.
- Association autonomy gives a power to component DBS to disassociate itself from a federation which means FDBS can operate independently of any single DBS.

The ANSI/X3/SPARC Study Group outlined a three level data description architecture, the components of which are the conceptual schema, internal schema and external schema of databases. The three level architecture is however inadequate to describing the architectures of an FDBS. It was therefore extended to support the three dimensions of the FDBS namely Distribution, Autonomy and Heterogeneity. The five level schema architecture is explained below.

Concurrency control

The Heterogeneity and Autonomy requirements pose special challenges concerning concurrency control in an FDBS, which is crucial for the correct execution of its concurrent transactions (see also Global concurrency control). Achieving global serializability, the major correctness criterion, under these requirements has been characterized as very difficult and unsolved. Commitment ordering, introduced in 1991, has provided a general solution for this issue (See Global serializability; See Commitment ordering also for the architectural aspects of the solution).

Five Level Schema Architecture for FDBSs

The five level schema architecture includes the following:

- Local Schema is the conceptual concept expressed in primary data model of component DBMS.
- Component Schema is derived by translating local schema into a model called the canonical data model or common data model. They are useful when semantics missed in local schema are incorporated in the component. They help in integration of data for tightly coupled FDBS.
- Export Schema represents a subset of a component schema that is available to the FDBS. It may include access control information regarding its use by specific federation user. The export schema help in managing flow of control of data.
- Federated Schema is an integration of multiple export schema. It includes information on data distribution that is generated when integrating export schemas.
- External Schema defines a schema for a user/applications or a class of users/applications.

While accurately representing the state of the art in data integration, the Five Level Schema Architecture above does suffer from a major drawback, namely IT imposed look and feel. Modern data users demand control over how data is presented; their needs are somewhat in conflict with such bottom-up approaches to data integration.
**External links**

- Schema coordination in federated database management: a comparison with schema integration [3]
- Storage of Behaviour of Object Database [4]
- DB2 and Federated Databases [5]
- Tutorial on Federated Database [6]
- GaV and LaV explained [7]
- Issues of where to perform the join aka "pushdown" and other performance characteristics [8]
- Worked example federating Oracle, Informix, DB2, and Excel [9]
- Composite Information Server - a commercial federated database product [10]

**References**


