# On Architectures for Federated DB Systems

Fèlix Saltor
Benet Campderrich
Manuel García-Solaco

Report LSI–94–9–R

# ON ARCHITECTURES FOR FEDERATED DB SYSTEMS

**F. Saltor, B. Campderrich†, & M. García-Solaco**

Dept Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona

†Dept. d'Enginyeria Informàtica
Universitat Rovira i Virgili, Tarragona

**Abstract**

*Different architectures for federated database systems have been proposed. We present three federation rules, analogous to the rules for fragmentation in distributed databases, and show that the reference architecture by Sheth & Larson satisfies them. We then analyze three shortcomings of the reference architecture: databases components of several federations, external schemata in user models different from the canonical model, and multiple semantics at the federated schema level, and we provide better solutions to each of them. With these solutions a new 8-level schema architecture framework is developed.*

## 1. Introduction

The cooperation between several databases (DBs) may arise in a number of cases. Different organizations, each with its own DB, may want to cooperate (subsidiaries of a common parent company, states of a federal country, government agencies, countries forming a common market, etc.). An organization might have several DBs, for example one per division, grown up independently, and new management needs force them to interoperate. Two companies may merge, or a takeover can take place, and keeping their respective DBs and have them cooperate may be preferable to their substitution by a new, common DB.

In each of these cases, we find different DBs, designed independently and operating autonomously, that are bound to cooperate. We assume that the technical form of cooperation is *interoperability*, supporting *integrated access* to the collection of DBs. This means that a user is able to ask a single query (one access), and receives a single, consolidated answer; the individual DBs have cooperated to produce this answer. Which of the DBs have provided which data may be hidden to the user, and then

he has the feeling of accessing a single DB; alternatively, data may be tagged with an identification of the DB supplying them (called "source tagging" in [WanMad90]), depending on the needs of the user.

Building a system that supports this integrated access is not an easy task. The individual databases will, in general, have different data models and data languages, and a user posing queries may use still another model and language. Heterogeneities will also exist between the semantics of the databases, and between these semantics and those of the users, where *semantics* means the conceptualization of the universe of discourse (as represented in the schema of a database). Other problems not satisfactorily solved till now include the management of transactions and the trade off between autonomy of each individual database and interdependency of the whole interoperable system.

Even if difficult problems exist, due to growing user needs and interest to implement database interoperability, the issues involved have become a major topic of research in many centers worldwide. In particular, the architecture of an interoperable database systems is at the core of ongoing research and development efforts.

This paper is organized as follows: Section 2 recalls the reference architecture and terminology, due to Sheth and Larson, that we will be following. Section 3 presents three correctness rules for database interoperability. In section 4 we present three particular problems, not well solved by the reference architecture, and propose new schema levels to deal with them. Section 5 puts together these three new schema levels to form a complete architecture framework. We conclude in section 6.

## 2. The reference architecture

We will say that the individual DBs, called *component DBs*, form a *federation*, or a *federated DB*. Considering their respective DB systems, a *federated DB system* is obtained thru the interoperability of the *component DB systems*. The federated DB system (FDBS) has no data of its own, it answers queries by accessing the component DB systems; it is a layer of software placed on top of the DBMSs of the component DBs (other software and hardware needed are out of the scope of this paper).

To support integrated access, the FDBS must have an adequate architecture and a convenient data model. Many architectures have been proposed, developed in prototypes, or implemented for production use (see for example [LanRos82], [Dev+82], [Lit85], [Rus+89], [SheLar90], [Tho+90], [KamRusShe91], [OzsVal91], [Hsi92], [HsiNeuSac93], [SchSheCze93], [HurBriPak93], [LitSha94]). In this section, we will follow the reference architecture and the terminology proposed by Sheth and Larson in their survey [SheLar90], because it is very general and it encompasses most of the others. This architecture is shown in Figure 1 for the five schema levels, and in Figure 2 for the schema levels and the software processors between these levels.

First, a *canonical* data model (CDM) common to the whole federation, must be adopted. The database schemata of the component DBs (*local schemata*) are transformed from their native models to the CDM by *transforming processors*, giving *component schemata*. Each component schema is filtered by *filtering processors* into one or more *export schemata*. From export schemata of different component DBs, a *federated schema* is constructed --- this process, performed by a *constructing processor*, is called *schema integration*; several federated schemata can exist in a federation. Finally, from a federated schema a number of *external schemata* are derived by *filtering processors*, for different users (or categories of users) of the federated DB.

When a federation is formed, or when a DB is to enter an existing federated DB, a *negotiation* process takes place. Each DB negotiates which of its own data it makes accessible (it "exports") to the federation (to which categories of users of the federation), and which part of these data, if any, it allows not only to be read, but also to be updated (by which categories of users). It is clear, therefore, that a DB may prevent part of its data from access by other DBs; this is the role of the export schemata and of their corresponding filtering processors. Negotiation may include to which point, if any, the *autonomy* of the DB is substituted by an interdependence with the other DBs, without compromising its support for its own preexisting users.

Once the federated system has been built, a query using an external schema will be mapped to its federated schema (filtering processor), decomposed into subqueries to the component DBs concerned (constructing processor), translated to their local schemata (filtering/transforming processors), and submitted to the corresponding DBMSs. These provide (sub)results, which are translated (transforming/filtering processors), consolidated (constructing processor), adapted, formatted (filtering processor), and presented to the user.

Since each component DB was designed independently of each other, the diversity of design decisions will have led to a number of heterogeneities. Examples of *systems heterogeneities* are: different CPUs, operating systems, data models and languages, DBMSs, communication protocols. *Data heterogeneities* are due to the different conceptualizations by the designers of the component DBs (*semantic* heterogeneities), and to the different representations of these conceptualizations in the respective data models and DBMSs (*syntactical* heterogeneities).

The use of a common CDM, and the transformation from local schemata into component schemata by transforming processors, solve the problem of syntactic heterogeneities. Semantic heterogeneities are dealt with in the schema integration process, by which export schemata are integrated into federated schemata by constructing processors.

The interoperability between the component DBs may be tighter or looser. In the case of a *tightly coupled* FDBS, the federation puts in place a DB administration at the federated level, who performs the negotiation process with the DBAs of the component DBs, is responsible for the schema integration process, and maintains the federated and external schemata. A *loosely coupled* FDBS, on the other side, has no federated DB administration, so that each individual user is responsible for selecting which component DBs to access, for the overcoming of syntactical and semantic heterogeneities, including the schema integration process, and for posing his integrated query.

We have briefly shown how this five level architecture supports integrated access to the federated DB. Three layers in this architecture, namely component schemata, export schemata and federated

schemata, are expressed in the CDM (external schemata may be expressed in different models, as we will discuss in section 4.3). How well this architecture solves data heterogeneities depends on the characteristics of the CDM (as analyzed in [SalCasGar91]).

## 3. Rules for federated architectures

Federated DBs and distributed DBs differ in important respects -including the autonomy and heterogeneities, characteristics of FDBSs-, but they have in common the fact that they rely on a network system made of interconnected DB systems (*distributed systems*). To the top down distribution of the conceptual schema in a distributed DBs, it corresponds the bottom up construction of federated schemata in a FDBS. This correspondence extends to the rules to be followed in each case.

In distributed databases, when a relation of the *global schema* is split or distributed into *fragments*, i.e. relations at the *local schemata*, there are three *correctness rules* that the fragmentation must follow ([CerPel84], [OzsVal91]):

1. *Completeness:* Each data item of the global relation must exist in some fragment.

2. *Reconstruction:* It must be possible to reconstruct the global relation from its fragments.

3. *Disjointness:* Each data item of the global relation must exist in only one fragment, except for the key in vertical fragmentation (fragments can then be replicated).

Analogously, in the case of federated databases, three correctness rules for schema integration should be satisfied (called *federation rules* in [GarSal91]):

1. *Completeness:* Each data item that a component DB exports to the federation must be available to authorized users at the federated level.

2. *Decomposition:* Any query at the federated level must be internally decomposable into (sub)queries against the component DBs involved.

3. *Discrimination:* Each data item in the result of a federated query must be traceable to the component DB it comes from, for users authorized to know the source of the data (see [WanMad90]). Note that this rule specifies the contrary of the *transparency* usually found in distributed DBs.

Any architecture for FDBSs should allow the schema integration process to satisfy these rules. In particular, the five level reference architecture supports rules 1 and 2 by singling out the export schemata, the federated schemata, and the constructing processors: these processors construct *complete* federated schemata, and *decompose* federated queries. It allows the support of rule 3 thru the external schemata and their filtering processors: *Discriminators* of data (by component DB) existing at the federated schema level, will be filtered out for external schemata and users not needing the source of the retrieved data, while they will be kept in external schemata for applications and users authorized to know this "source tagging", as explained in [GarSal91]. Another usage of discriminators is made in section 4.3.

**4. Three particular issues**

The reference architecture by Sheth & Larson is generally adequate: it separates the issues into different schema levels and specific processors, and is general enough to allow different particular adaptations to individual architectures. Moreover, it allows the construction of federated schemata satisfying the rules presented in section 3.

However, we find three particular issues that do not fill well within their architecture: databases components of several federations, external schemata in user models different from the canonical model, and multiple semantics at the federated schema level. The solution to each of these issues requires an additional schema level, as we will show.

### 4.1 Negotiable schemata

One of the characteristics of a FDBS is that each component DB keeps its autonomy as much as possible; in particular, the DBA of a component DB of a federation may decide to enter another FDBS. In this way a DB may become component of several federations at the same time. Most likely, these FDBSs will be loosely coupled, with perhaps just one of them being tightly coupled, as shown in figure 3.

In the reference architecture, the local schema of such a component DB will be transformed to the CDMs of each of the federations (CDM-A, CDM-B, ..., CDM-C in figure 3), giving component schemata, and then, for each FDBS, one or more export schemata will be negotiated. It is likely that some data of the component DB will be considered private, and therefore not exportable to any federation: the translation of this private part of the local schema to each CDM will have been useless. Moreover, the maintenance of the local schema during the life of the component DB will imply maintaining accordingly all its component schemata, even for changes affecting only private data.

These problems do not appear in another architecture, in which the role of the local schema is separated into two different levels (Figure 4), both in the native model: the *native schema*, which is the original schema of the component DB (*conceptual schema* in the ANSI/SPARC terminology, *database schema* in the ISO terminology [vGr82]), and the *negotiable schema*, which contains only non private data. It is just the negotiable schema, not the whole native schema, which is transformed to each CDM, producing the negotiable schemata for each federation.

In the case of loosely coupled federations, the transformed negotiable schemata will be directly available to users without further negotiation, that is, will become export schemata in the terminology of the reference architecture. For a tightly coupled FDBS, however, the transformed negotiable schema will, in general, enter the negotiation process to produce one or more export schemata.

Note that the derivation of the negotiable schema from the native schema may be done by the component DBMS itself, using its mechanism to produce its own external schemata (view mechanism), as used for its pre-existing users; this mechanism is available today in most relational DB systems. Therefore this processor (a filtering processor according to the terminology we are using) will not be part of the FDBS as such.

### 4.2 User models and translated schemata

A user of a federation may be trained in a data model different from the CDM of the FDBS. This may arise because this user was previously a user of one of the component DBs of the federation, and as such is familiar with the native model of this component DB, and this particular model was not chosen as the CDM of the FDBS. Alternatively, the user may be a new member of the organization who comes from an environment in which a different data model was in use.

According to the reference architecture, from a federated schema in the CDM, an external schema in the user model is produced (figure 5). This process consists in fact of two transformations: change in the model, and change in the contents, corresponding to two processor types: transforming processor and filtering processor. The separation of issues into schema levels and specific processors is not complete in this case.

We present a different architecture, in which the external schema level is divided into two (figure 6). The upper level schema of the federated user, in his own model, is called *user schema*; the translation of this user schema to the CDM is called *translated schema*. They are at different levels in the architecture. The translated schema is derived from a federated schema by a filtering processor, while the passage from the translated schema to the user schema (and vice versa) is done by a transforming processor. The issues and the processors are in this way clearly separated: only one change is made at each step.

### 4.3 Multiple semantics and application schemata

A federated schema is constructed from export schemata by a constructing processor according to a schema integration process, as we have seen. The export schemata being integrated may represent different semantics, and the resulting federated schema may adopt just one of these semantics, or still another, different semantics, or it may support multiple semantics. Let us illustrate the issue of multiple semantics with an example adapted from the one in [SheLar90].

Assume that component DB1 has an object class SHOES, with an attribute COLOR taking values on the domain {BROWN, TAN, CREAM, WHITE, BLACK}. Further assume that component DB2 has an object class SHOES, too, with an attribute COLOR taking values on the domain {BROWN, TAN, WHITE, BLACK}. The semantics of SHOES.COLOR in the two component databases are different (a given shoe, considered CREAM by DB1, would not be seen of that color by DB2), and these semantics will be transmitted untouched to the corresponding export schemata.

On the user side, we will suppose that *userA* conceives colors of shoes the same way as DB1, while for *userB* they are BROWN, CREAM, WHITE and BLACK. Further assume that the mappings between these user and DB semantics is as follows:

*UserA* maps his CREAM to TAN in DB2 -no shoe in DB2 will be TAN for him-; his mapping to DB1 is the identity mapping.

*UserB* maps her CREAM to TAN **or** CREAM in DB1, and to TAN **or** WHITE in DB2 -no shoe in DB2 will be WHITE for her-.

In the case of loosely coupling, userA will integrate the export schemata according to his mappings to construct a federated schema with his semantics, and similarly for userB: one federated schema for each user semantics. In the case of a tightly coupled FDBS, this same possibility of each federated schema with a different semantics exists, but another solution is the support of multiple semantics in a single federated schema, as explained in [GarSal91]. In this schema, object class SHOES, a generalization of the SHOES in DB1 and DB2, will have a discriminating attribute by component DB,

besides COLOR and other attributes (*upward inherited* as in [SchNeu88]) from DB1 and DB2. A directory at this federated level will contain the mappings to the semantics of userA, of userB, and any other semantics needed. External schemata for each particular user will be derived from this federated schema, by using the mappings corresponding to his semantics. With this approach, the number of federated schemata to be constructed, and, more importantly, to be maintained, is reduced to a minimum, compared with the solution of one federated schema for each semantics.

This approach of having federated schemata with multiple semantics and source tagged, however, in the case of the reference architecture (figure 7) has one shortcoming: the passage between a federated schema and an external schema changes not only from multiple semantics to one semantics, but also from a general universe of discourse using this semantics, which may be common to many users and applications, to the particular universe of discourse of a single user or a few users (not considering the possible change in data model, already discussed in section 4.2).

The changes are clearly separated by introducing another level, the *application schema* level (figure 8). Each application schema reflects just one semantics, common to one or more applications, and its scope is a general universe of discourse; external schemata adapts this application schemata to the needs of particular users. The passage from a federated schema to an application schema is done by a processor restricting from multiple to a single semantics: it is a special case of a filtering processor.

Note that this change in the architecture still satisfies the three federation rules. In particular, discriminators by component DB (source tags) support rule 3 in addition to being the base for multiple semantics.

## 5. A complete 8-level architecture framework

Putting together the three issues discussed in the previous section, we arrive at a new architecture, depicted in figure 9. This architecture has 8 levels, and is more general than any other federated

architecture we know of. It separates the issues involved into different processors and schema levels, so that each processor performs only one change: in data model, or in semantics, or in universe of discourse

We do not contend that this is the architecture to be implemented in a particular FDBS, but that it is a general architecture, that may be used as a reference to discuss about FDBS issues, problems and solutions, and also as a framework from which specific subset architectures can be implemented. In particular, the passage of a query from the user schema level to the native schemata, and of the (sub)results from these back to the user schema, need not be done step by step, one level at a time, at execution time. Specific FDBSs will combine several processors into one module, and will prepare transformations at compile time, to expedite query processing at execution time. The separation of concerns into different levels and processors in our 8-level architecture is done to clarify the issues, not as a blueprint for implementation.

This 8-level architecture complies with the three federation rules of section 3, because each one of the three changes made to the reference architecture has preserved this compliance. In particular, discriminators support both multiple semantics and rule 3.

**6. Conclusions**

We have presented three *federation rules*, analogous to the fragmentation rules in distributed databases, to be satisfied by any federated database architecture, and we have shown that the five level schema reference architecture by Sheth & Larson complies with them.

We have pointed at three particular issues not well solved in the reference architecture, and we have developed a better solution to each of them, with an additional schema level in each case. For databases component of several federations, *negotiable schemata* have been introduced. For external schemata in user models different from the canonical data model, *translated schemata* have been added. In the case of federated schemata supporting multiple semantics, we have an additional *application schema* level.

Putting together these three new schema levels, we have developed a complete, eight level architecture, in which each processor performs just one function, that satisfies the federation rules. It can be used as a reference architecture, or as a framework from which specific architectures can be implemented.

**Acknowledgments**

This work has been partially supported by the Spanish PRONTIC program, under project TIC93-0436.

**References**

[CerPel84] S. Ceri & G. Pelagatti: *Distributed Databses: Principles and Systems.* McGraw-Hill, 1984.

[Dev+82] Devor, Elmasri, Larson, Rahimi & Richardson: "Five schema architecture extends DBMS to distributed applications". *Electronic Design* March 1982.

[GarSal91] M.García & F.Saltor: "Discriminated Operations in Interoperable Databases". In [KamRusShe91].

[Hsi92] D. Hsiao: "Tutorial on Federated Databases and Systems (Parts I & II)". *VLDB Journal, vol1*, #1 & 2 (July & October, 1992).

[HsiNeuSac93] Hsiao, Neuhold & Sacks-Davis (eds): *Interoperable Database Systems (DS-5).* (Proc. of the IFIP WG2.6 Database Semantics Conference on Interoperable Database Systems (DS-5), Lorne, Victoria, Australia, November 1992). IFIP Transactions A-25. North-Holland, 1993.

[HurBriPak93] A. R. Hurson, M. W. Bright, & S. H. Pakzad: *Multidatabase Systems: An Advanced Solution for Global Information Sharing.* IEEE Computer Society Press, 1993.

[KamRusShe91] Y.Kambayashi, M.Rusinkiewicz & A.Sheth (eds.): *Proc. First International Workshop on Interoperability in Multidatabase Systems* (IMS'91, Kyoto). IEEE-CS Press, 1991.

[LanRos82] T. Landers & R. Rosenberg: "An Overview of Multibase". In H-J. Schneider (ed.) *Distributed Databases,* North Holland, 1982.

[Lit85] W. Litwin: "An overview of the multidatabase system MRDSM". In *Proc. ACM National Conference,* ACM, 1985.

[LitSha94] Witold Litwin & Ming Chien Shan: *Introduction to Interoperable Multidatabase Systems.* Prentice Hall, 1994.

[OzsVal91] M. T. Özsu & P. Valduriez: *Principles of Distributes Database Systems.* Prentice Hall, 1991.

[Rus+89] Rusinkiewicz, Elmasri, Czejdo, Georgakopoulos, Karabatis, Jamoussi, Loa & Li: "Omnibase: Design and implementation of a multidatabase system". In *Proc. Symp. on Parallel and Distributed Processing,* May 1989.

[SalCasGar91] F.Saltor, M.G.Castellanos & M.García-Solaco: "Suitability of Data Models as Canonical Models for Federated Databases". In: A.Sheth (ed.): Special issue on Semantic Heterogeneity, *ACM SIGMOD Record, Vol.20,* No.4, (Dec. 1991), pages 44-48.

[SchNeu88] M.Schrefl & E.Neuhold: "Object class definition by generalization using upward inheritance". *Proceedings of the 4th Int. Conf. on Data Engineering,* Los Angeles. IEEE-CS Press, 1988.

[SchSheCze93] H. Schek, A. Sheth & B. Czejdo (ed.): *Proc. 2nd Int. Workshop on Interoperability in Multidatabase Systems* (RIDE IMS-93). IEEE-CS Press. Vienna. Apr. 1993.

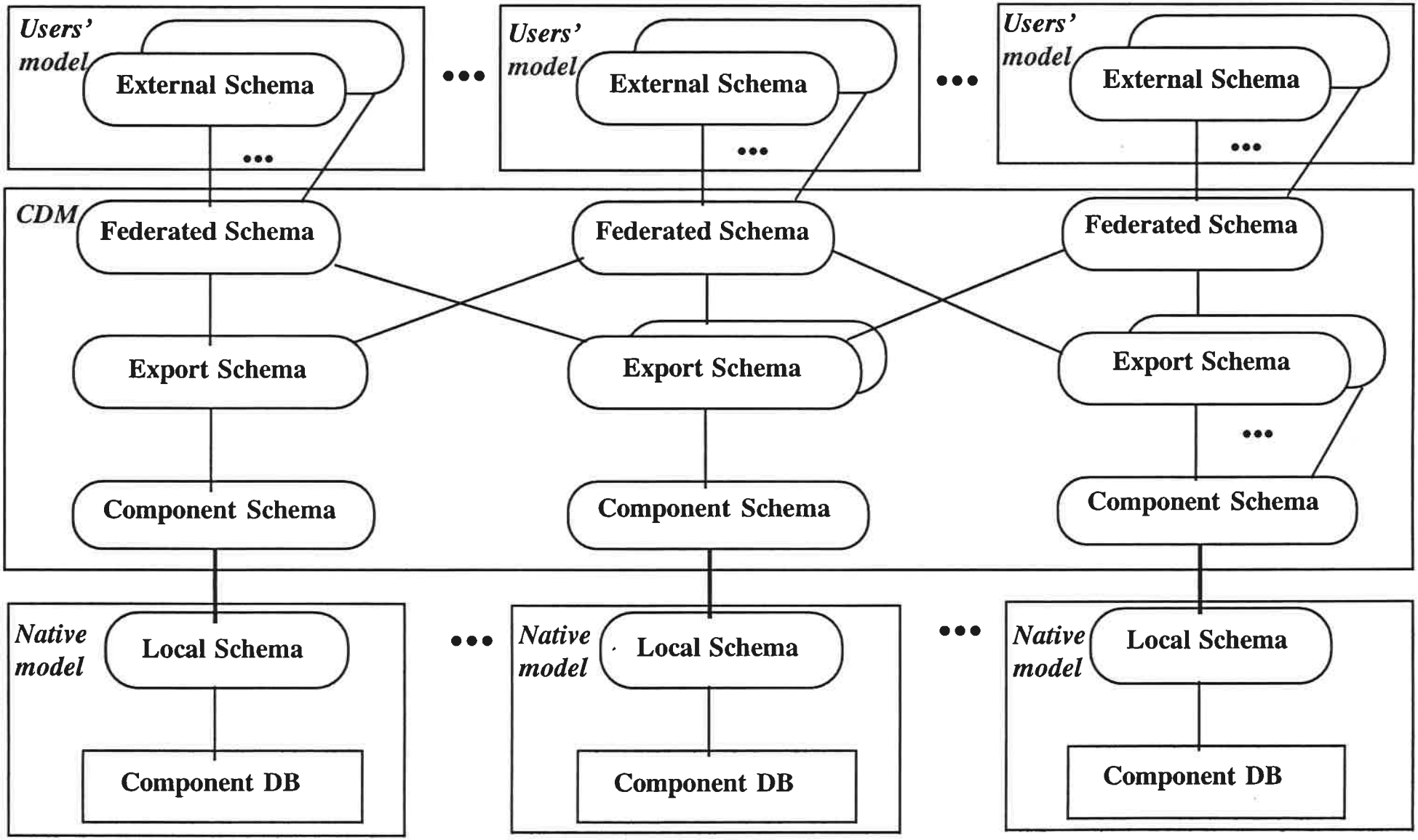
[SheLar90] A.Sheth & J.Larson: "Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases". *ACM Computing Surveys, Vol.22,* No.3, (Sept. 90).

[Tho+90] Thomas, Thompson, Chung, Barkmeyer, Carter, Templeton, Fox & Hartman: "Heterogeneous Distributed Database Systems for Production Use". *ACM Computing Surveys, vol. 22,* No 3 (Sept. 1990).

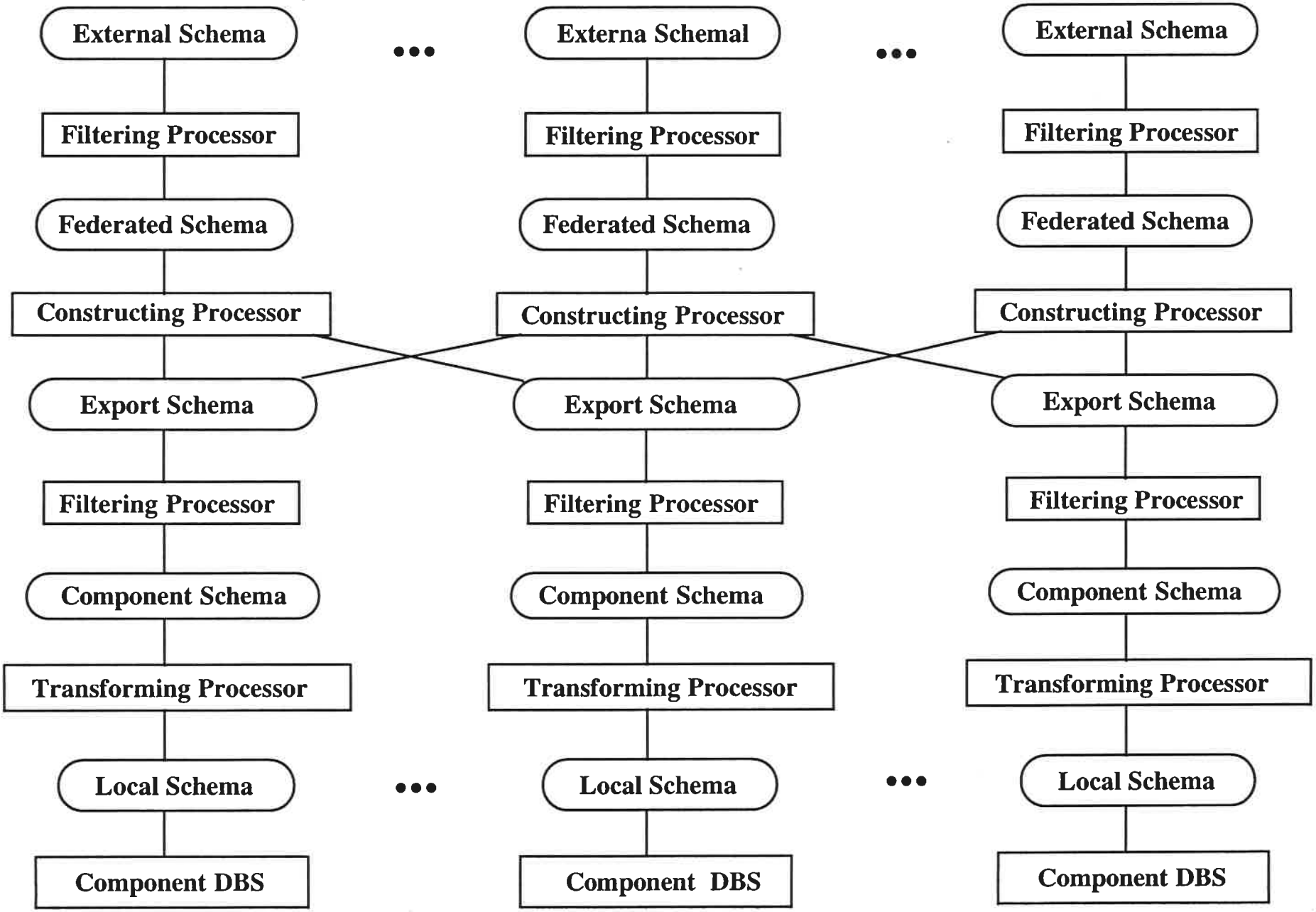
[vGr82] J. van Griethuysen (ed): *Concepts and terminology for the conceptual schema and the information base.* ISO/TC97/SC5/WG3, 1982.

[WanMad90] Y. Wang & S. Madnick: "A Polygen Model for Heterogeneous DBS: the source tagging perspective". *Proc. 16th VLDB* (Brisbane), 1990.

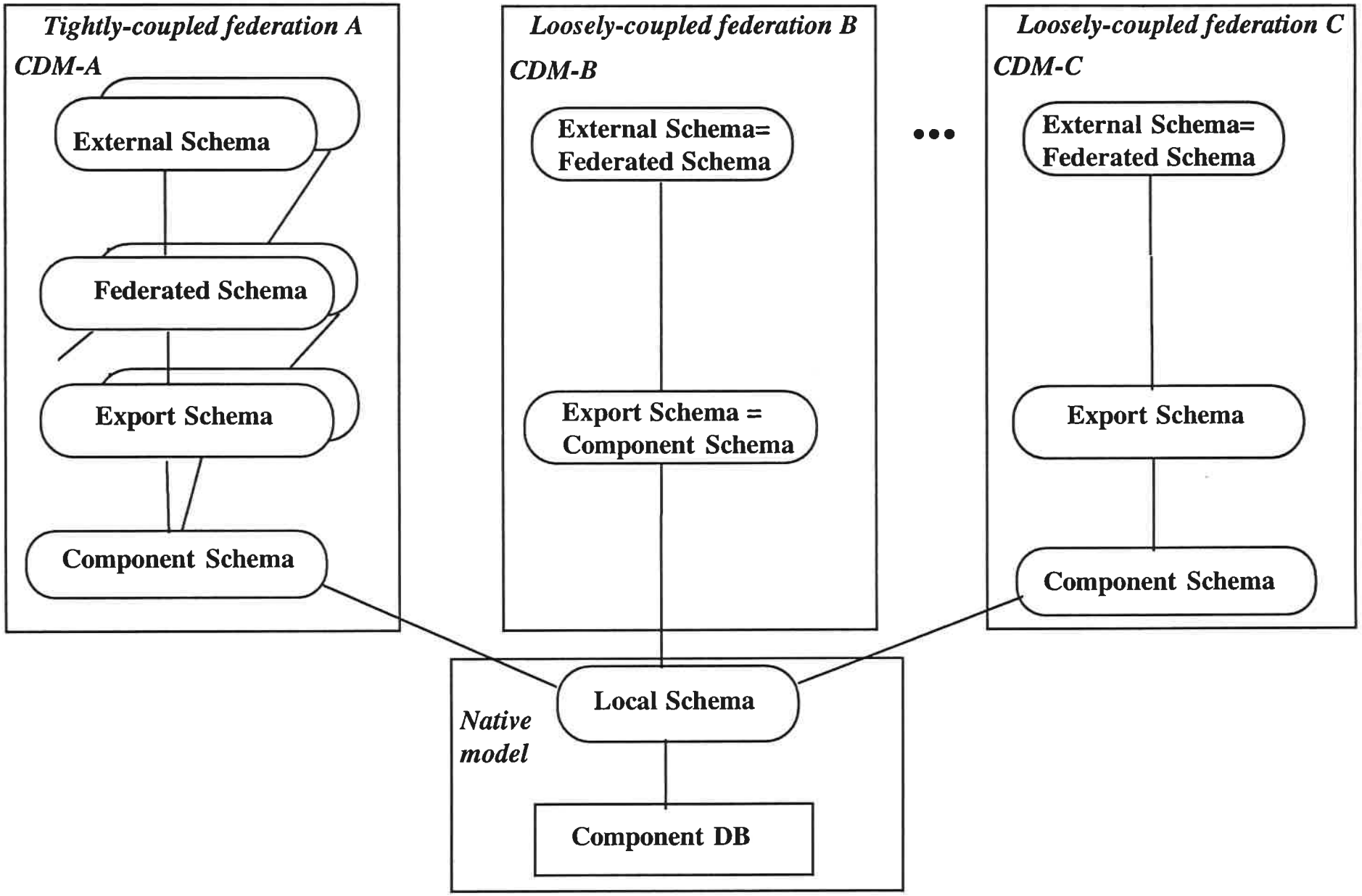
# Five level Schema Architecture of a FDBS (Sheth & Larson)



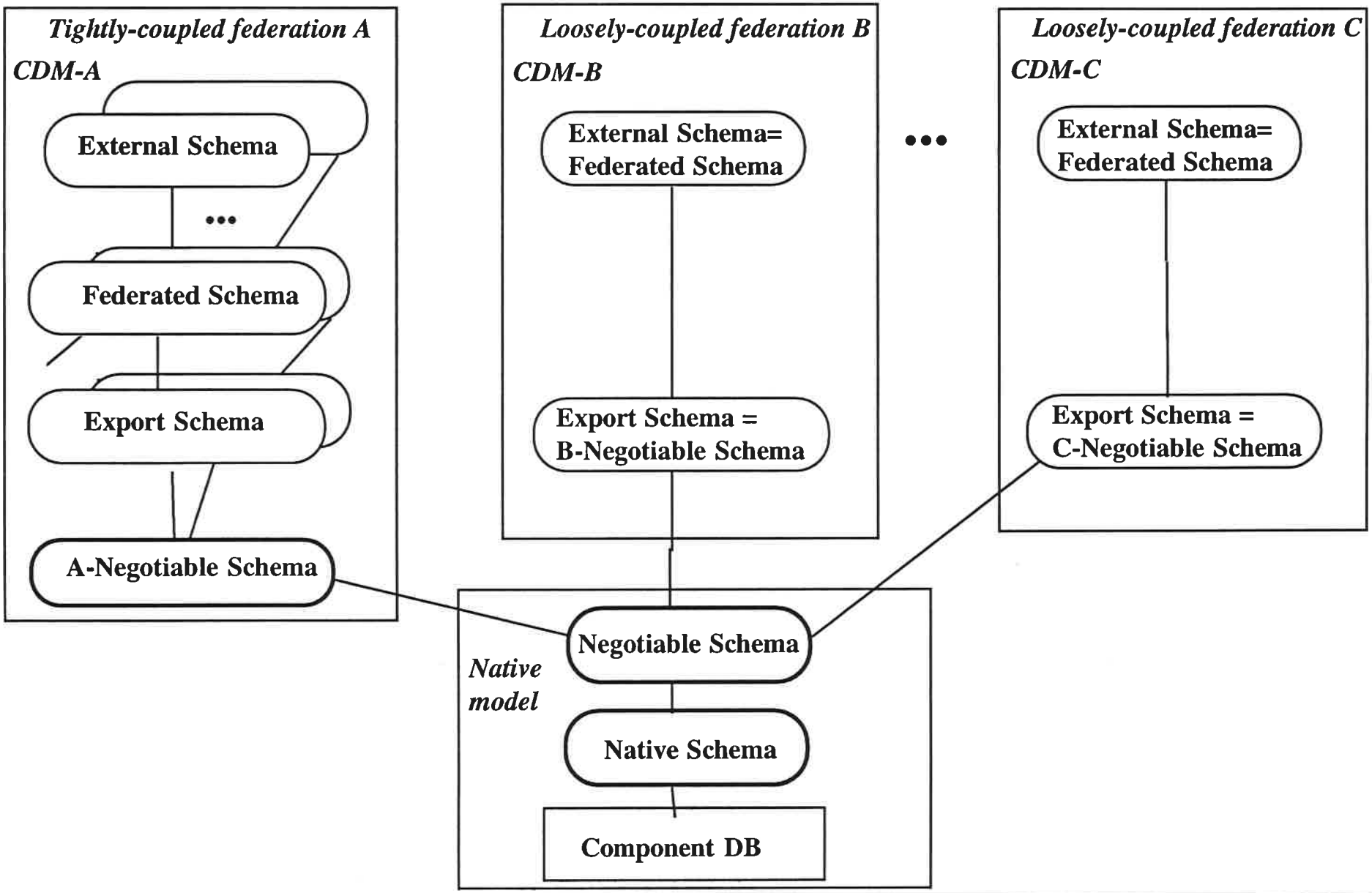
# System Architecture for an FDBS



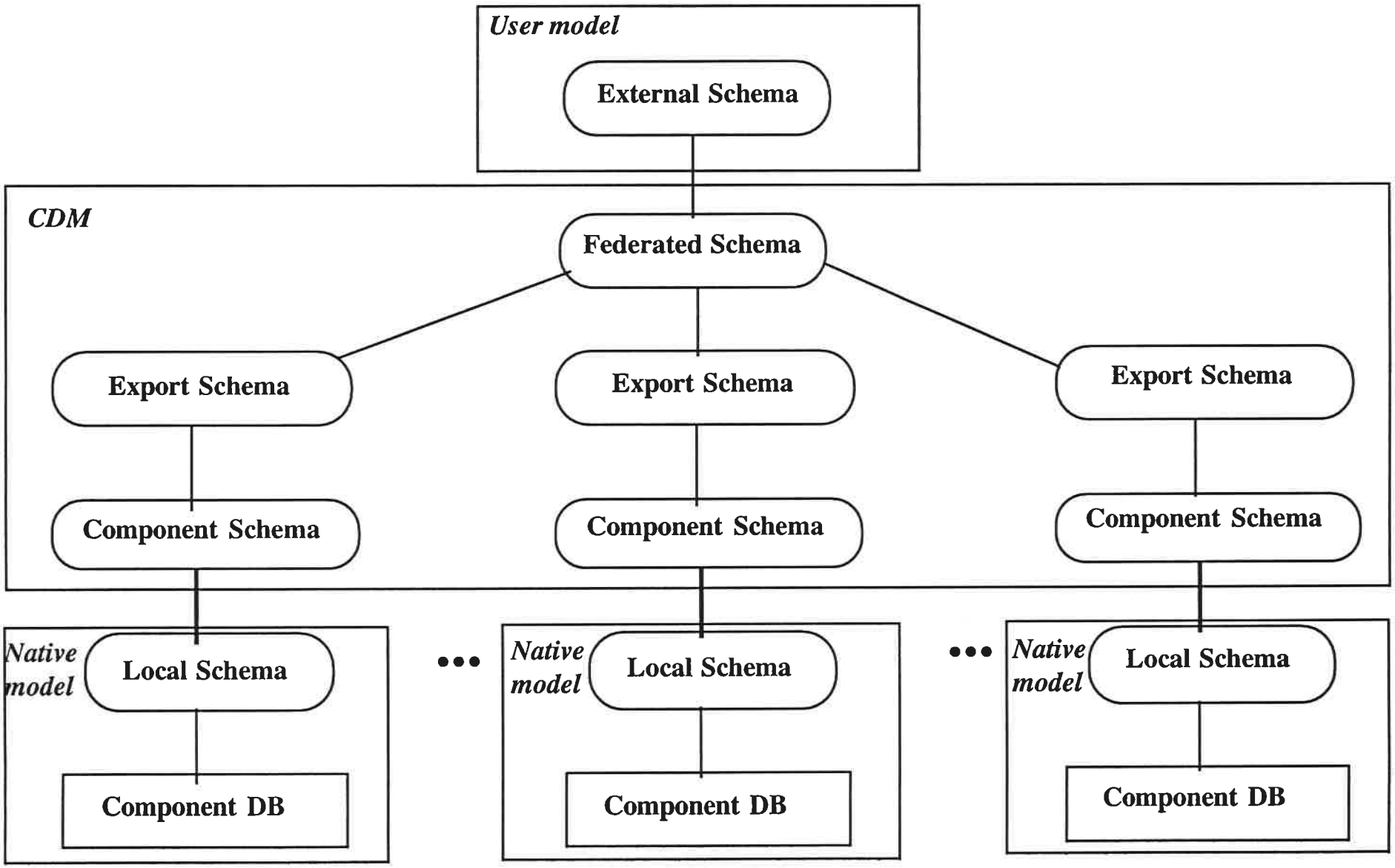
# One database component of several federations



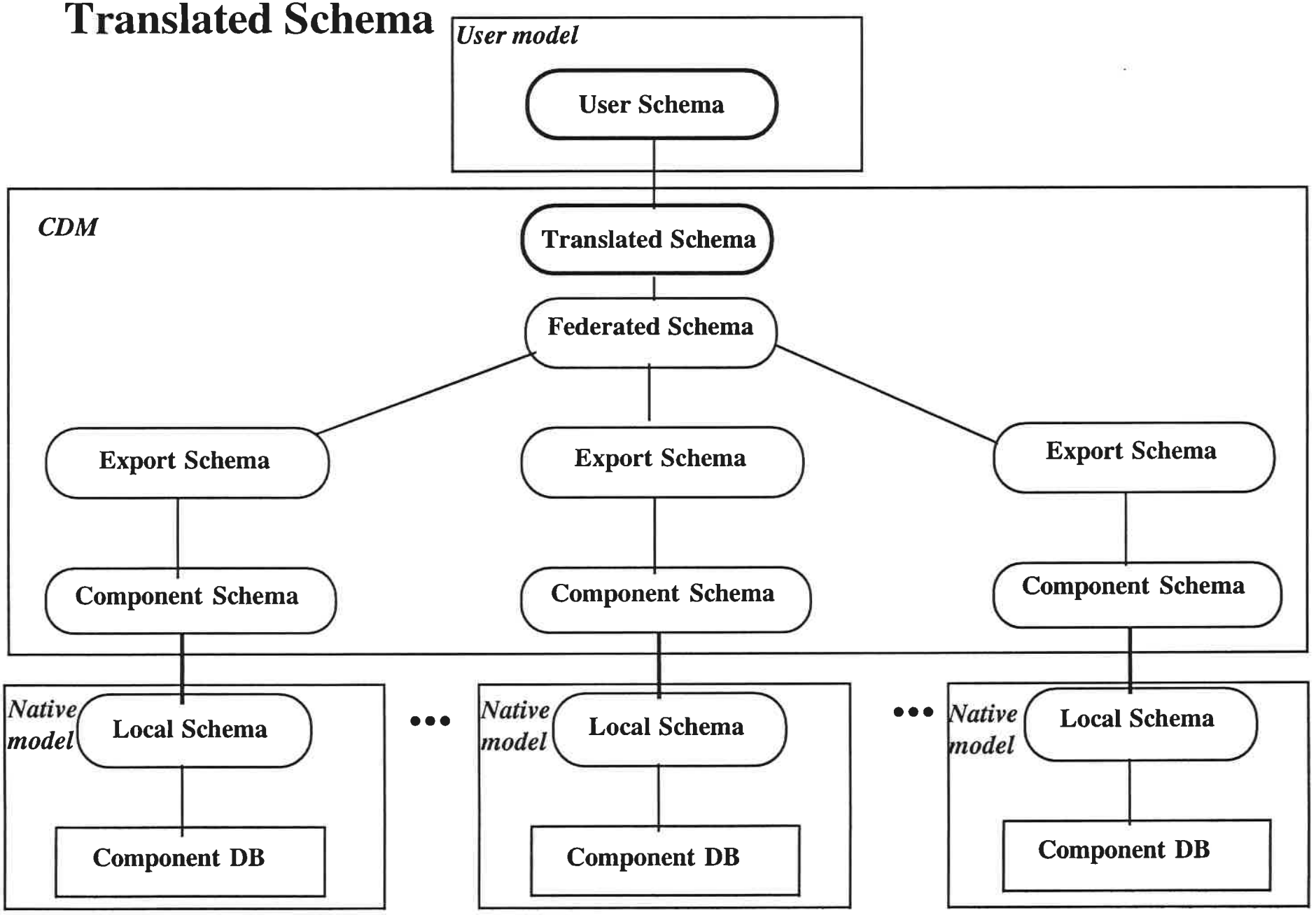
# The negotiable schema



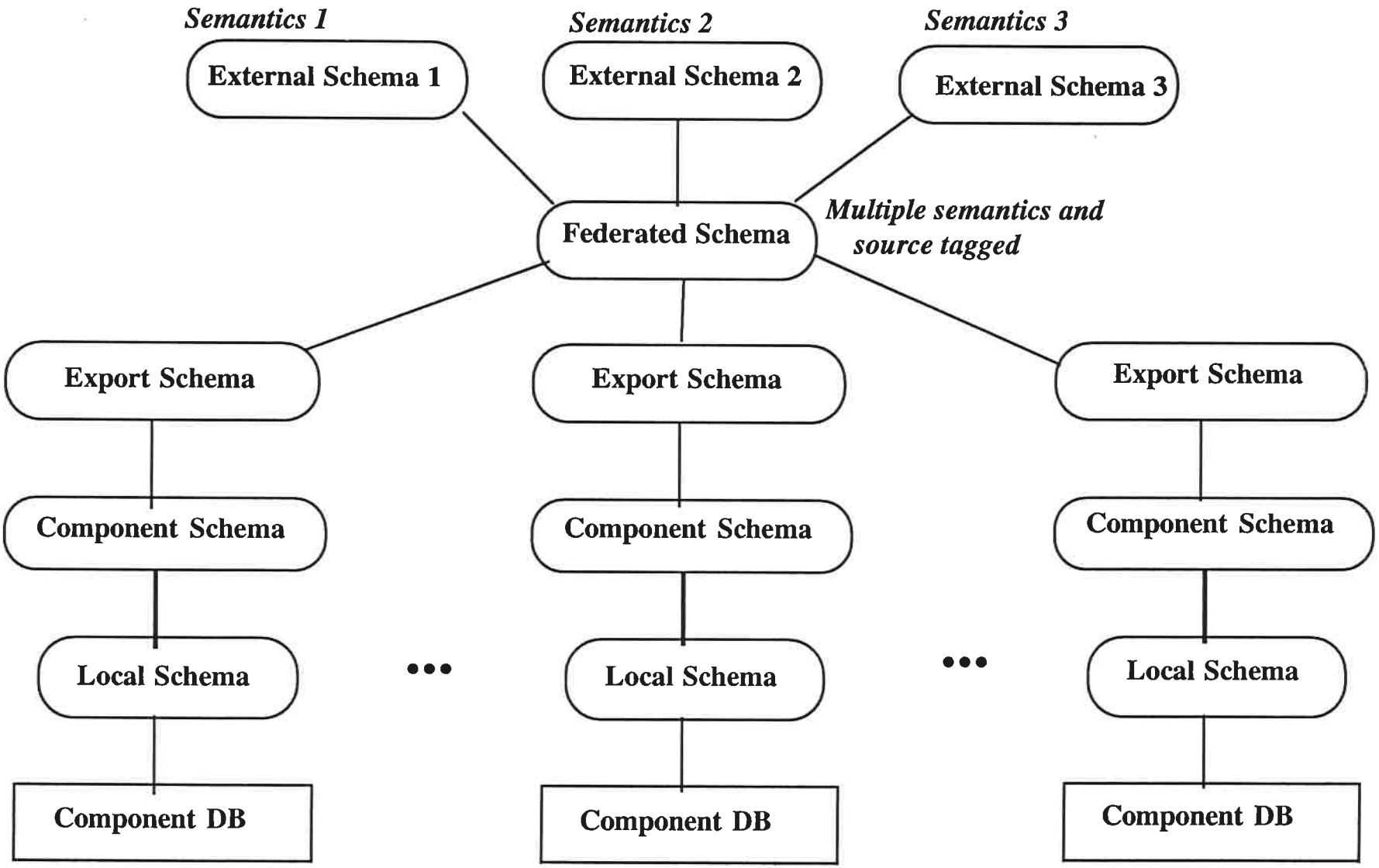
# An external schema not in the CDM



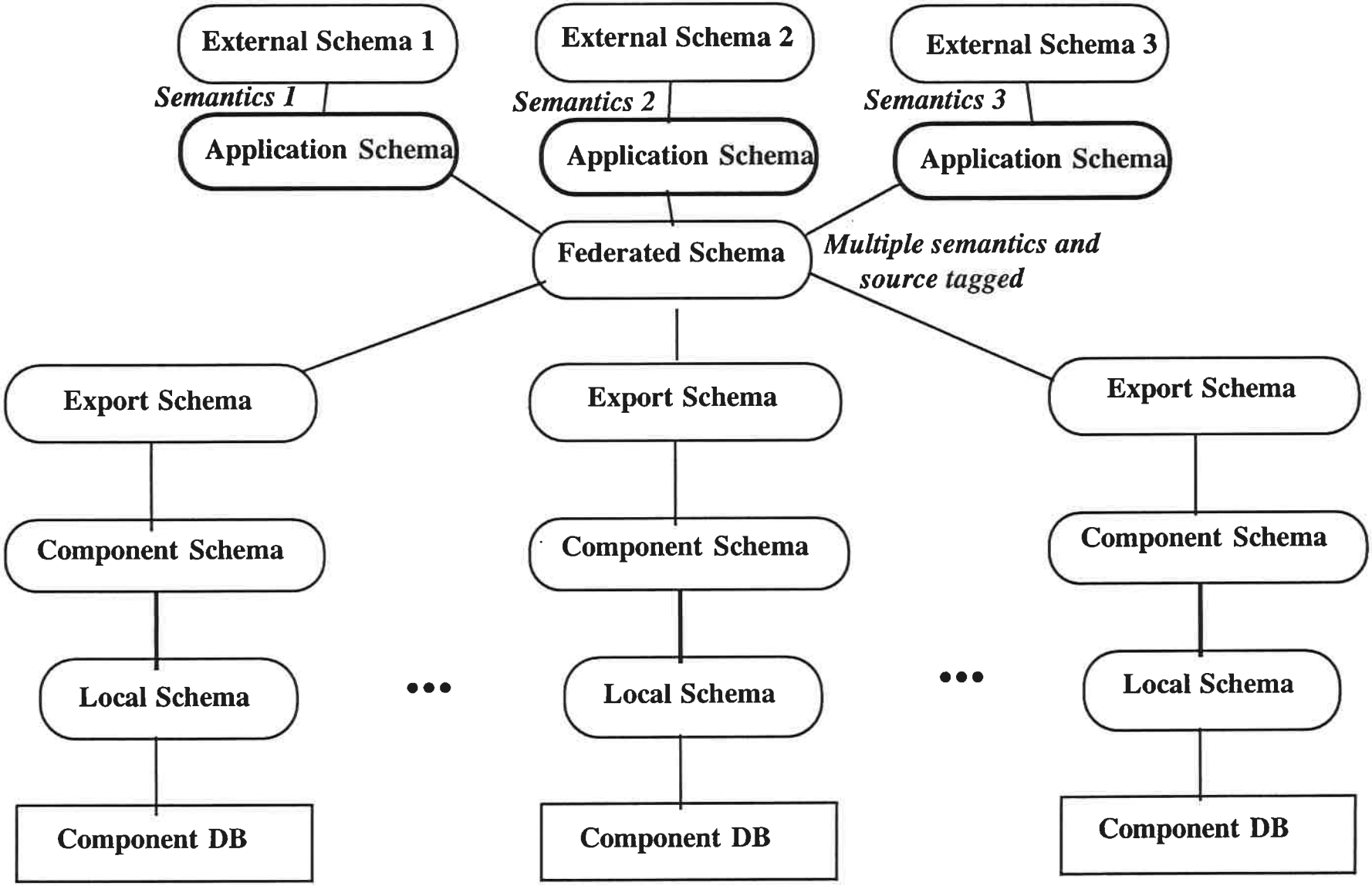
# Translated Schema



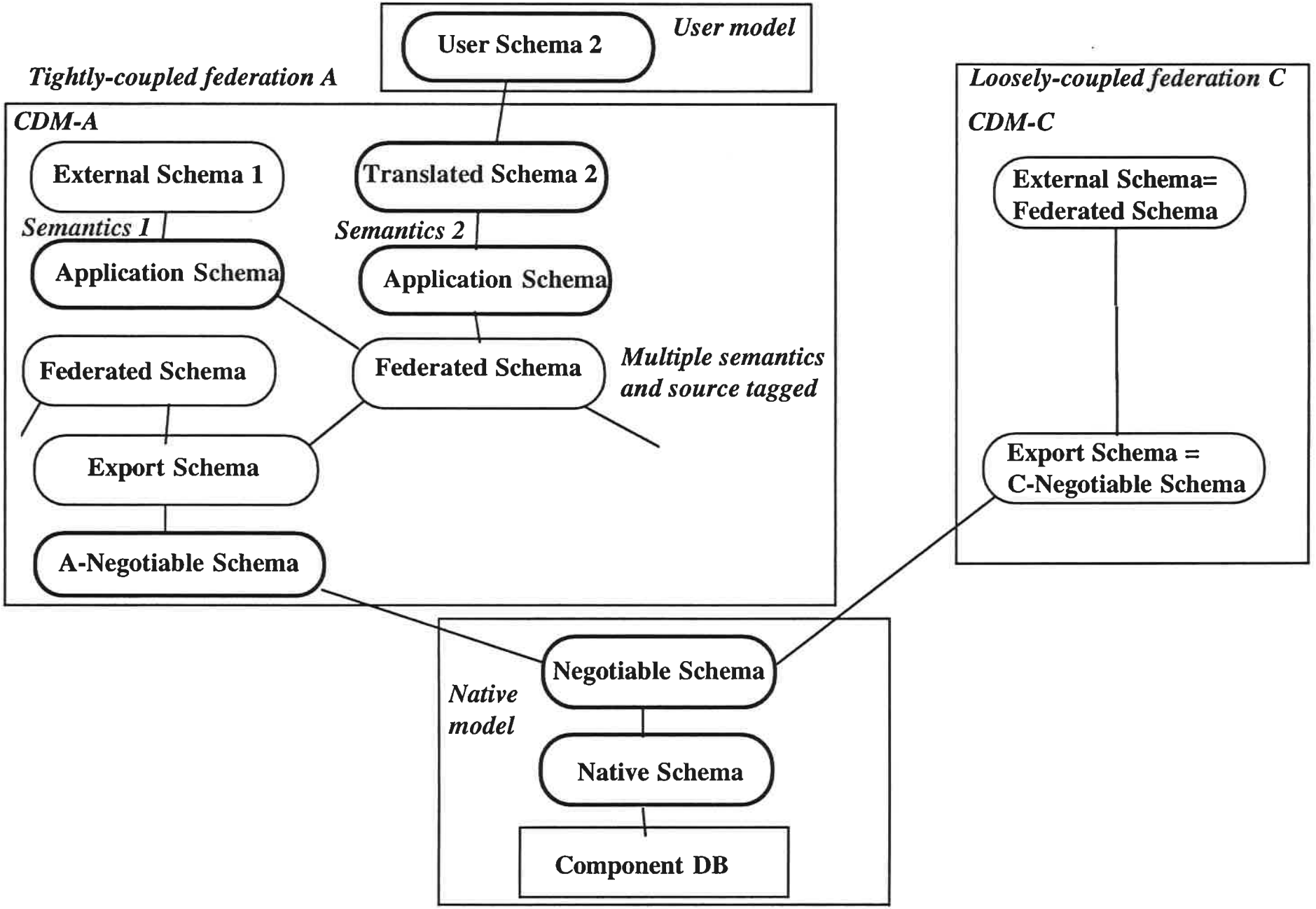
# Multiple semantics versus one semantics



# Multiple semantics versus one semantics: the application schema



# Complete 8-level architecture



**Departament de Llenguatges i Sistemes Informàtics**
Universitat Politècnica de Catalunya

**Research Reports – 1994**

LSI-94-1-R "Logspace and logtime leaf languages", Birgit Jenner, Pierre McKenzie, and Denis Thérien.

LSI-94-2-R "Degrees and reducibilities of easy tally sets", Montserrat Hermo.

LSI-94-3-R "Isothetic polyhedra and monotone boolean formulae", Robert Juan-Arinyo.

LSI-94-4-R "Una modelizaciòn de la incompletitud en los programas" (written in Spanish), Javier Pérez Campo.

LSI-94-5-R "A multiple shooting vectorial algorithm for progressive radiosity", Blanca Garcia and Xavier Pueyo.

LSI-94-6-R "Construction of the Face Octree model", Núria Pla-Garcia.

LSI-94-7-R "On the expected depth of boolean circuits", Josep Díaz, María J. Serna, Paul Spirakis, and Jacobo Torán.

LSI-94-8-R "A transformation scheme for double recursion", José L. Balcázar.

LSI-94-9-R "On architectures for federated DB systems", Fèlix Saltor, Benet Campderrich, and Manuel García-Solaco.

LSI-94-10-R "Relative knowledge and belief: SKL preferred model frames", Matías Alvarado.

---

Copies of reports can be ordered from:

Nuria Sánchez
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo, 5
08028 Barcelona, Spain
secrelsi@lsi.upc.es