# Integration of Inheritance Trees as Part of View Generation for Database Federations*

Ingo Schmitt & Gunter Saake

Institut für Technische Informationssysteme
Otto-von-Guericke-Universität Magdeburg
Postfach 4120, D–39016 Magdeburg
E-mail: {schmitt|saake}@iti.cs.uni-magdeburg.de

**Abstract**

Schema integration is the basis for successfully building a database federation. Current proposals directly integrate the different schemata using a semantical powerful data model, for example an extended ER model or an object-oriented model. We propose instead to use for integration a semantical poor model and build semantically rich representations as external views only. This approach enables a flexible integration and derivation especially for inheritance trees. We present an algorithm enabling the derivation of different inheritance trees as external views onto an integrated schema. The resulting inheritance tree can be influenced by giving priorities to input classes or attribute combinations and satisfies several formalized quality criteria for external views.

**Keywords:** database federation, schema integration, view derivation, inheritance hierarchy, extensional and intensional conflicts

## 1 Introduction

The integration of heterogeneous schemata and the derivation of views is an essential task in the area of multidatabase systems. Parts of local schemata have to be integrated and transformed to specific views of global applications. The data itself is integrated only virtually.

The main problem which arises during the integration of different schemata is how to correctly deal with the heterogeneity. Data model heterogeneity can be resolved by transforming each local schema into the canonical data model. After that schema heterogeneity has to be overcome. Many methodologies were proposed (e.g. [BLN86, LNE89, SPD92, Dup94, RPRG94, PBE95]) which describe various conflict resolution techniques.

In order to support logical data independence, different external schemata must be derivable from the integrated schema. From now on, we refer to such views as *external integrated schemata*. The generation of a view has to fit to the understanding of a specific user group. Users want to influence the generation of their external integrated schema.

There are different requirements attributed to an external integrated schema (adopted from [BLN86, page 337]:

- *Completeness and Correctness*: The integrated schema must contain all concepts present in any component schema correctly. The integrated schema must be a representation of the union of the application domains associated with the schemata.

*This research is partially supported by the German Country Sachsen-Anhalt under FKZ: 1987A/0025 (Federating Heterogeneous Database Systems and Local Data Management Components for Global Integrity Maintenance).

- *Minimality*: If the same concept is represented in more than one component schema, it must be represented only once in the integrated schema.
- *Understandability*: The integrated schema should be easy to understand for the designer and the end user. This implies that among the several possible representations of results of integration by a data model, the one that is (qualitatively) the most understandable should be chosen.

These requirements and especially the support of different views are not sufficiently met by the proposed methodologies. Especially the integration of different inheritance hierarchies results in inheritance hierarchies which do not sufficiently fulfill the demand for minimality and understandability (cf. [TS93, GCS95]). Most of the proposed integration methodologies resolve semantical overlappings by introducing generalized classes (upward inheritance), e.g. [RPRG94, GCS95]. That is, the original inheritance hierarchies are subhierarchies of the resulting integrated hierarchy. Henceforth, the resulting hierarchy can be unnecessarily complex. In our approach the original inheritance hierarchies can be completely restructured in order to support logical data independence. Therefore, we offer more flexibility to integrate hierarchies.

Our approach differs from most existing proposals in using a *semantically poor* canonical data model. The data model used for integration is based on a few orthogonal data structuring concepts, namely classes, attributes and object identification. To be used for the integration process, it supports fundamental integrity constraints capturing part of the semantics lost during the transformation towards a semantically poor data model.

We will show that this approach allows the use of basic information on intensional (set of attributes) and extensional (set of possible objects) overlappings of original classes for an automatic integration and application specific view (or external schema) generation. Especially, the integration of *inheritance hierarchies* in object-oriented approaches is supported by a view derivation algorithm controlled by marking classes. Classes which are marked as preferred will strongly influence the structure of the derived inheritance trees, that is, they will be located close to the roots of the derived inheritance trees. In this way users are able to influence the derivation of their external integrated schemata.

## 2 Generic Integration Model

The right choice of the canonical data model is essential to the design process of a federated database system (FDBS, cf. [SL90]). A discussion about the right canonical data model can be found in [SCG91]. The choice of the canonical data model influences the quality of the integration step, which is the most difficult step of the design process. Most of the recent proposals prefer an object-oriented data model (e.g. in [TS93, RPRG94, GCS95, PBE95, Bra93]). It is often argued that only a semantically rich data model can be the right canonical data model because no or only little semantics is lost during the transformation from an object-oriented local schema to the canonical data model (see also [SCG91]). Instead, we propose the use of a canonical data model which is semantically poor. On the basis of the semantically poor data model *GIM*, abbreviation of *Generic Integration Model*, we propose an algorithm which generates external object-oriented schemata automatically. In this way we do not have the problem of semantical loss. A more detailed discussion about the advantages of a semantically poor canonical data model can be found in [Sch95].

Now we introduce the *Generic Integration Model* used for integration only. Because of space restrictions we are not able to describe GIM completely. We only describe some fundamental elements of our model[1].

[1]A complete formalization of GIM will be given in a forthcoming paper.

In the following we use some mathematical conventions. A function is considered as a set of ordered pairs of elements which assigns to each element of the domain (abbreviated by *dom*) exactly one element of the range. A special function is a set function. Its range contains sets.

We assume that the set $\mathcal{D}at$ and the set function $\mathcal{D}omain$ are given. $\mathcal{D}at$ contains the names of elementary data types and the set function $\mathcal{D}omain$ is a function over $\mathcal{D}at$. $\mathcal{D}omain$ assigns to each data type name a set possible values.

- Class $K$:
  $K \stackrel{\text{def}}{=} (Kname, Att)$, $Kname$ is the name of the class and $Att$ is a partial function from its defined attribute names into $\mathcal{D}at$ defining the intension of a class. Each class has an attribute $Id$ which represents the object identifier. An object identifier is an integer value. $Att$ can also contain reference attributes. Integrity constraints concerning reference attributes are not considered in this paper.

- Schema $S$:
  $S \stackrel{\text{def}}{=} (Sname, \mathcal{K})$, $Sname$ is the name of the schema and $\mathcal{K}$ is the set of classes $\{K_1, K_2, \ldots, K_n\}$.

- Extension $\mathcal{E}xt_K$ of class $K$:
  $\mathcal{E}xt_K$ is a set of functions. Each function represents an object and defines the attribute values for the given attributes. The object identifiers have to be unique.
  $\forall f \in \mathcal{E}xt_K : f$ is a function over $dom(K.Att)$ and
  $\forall x \in dom(f) : f(x) \in (\mathcal{D}omain(K.Att(x)) \cup \{NULL\})$ and
  $\forall f_1, f_2 \in \mathcal{E}xt_K : f_1(Id) = f_2(Id) \Rightarrow f_1 = f_2$

- Extension $Ext_{Id,K}$ contains the identifiers of the extension of class $K$
  $Ext_{Id,K} \stackrel{\text{def}}{=} \{f(Id) \mid f \in \mathcal{E}xt_K\}$.

- Database state $\mathcal{S}tate_S$ is a set function over the classes $\mathcal{K}$ of schema $S$. To each class of the schema its extension is assigned.
  $\forall K \in S.\mathcal{K} : \mathcal{S}tate_S(K) = \mathcal{E}xt_K$ with
  $\forall K_1, K_2 \in S.\mathcal{K} : Ext_{Id,K_1} \cap Ext_{Id,K_2} = \emptyset \vee Ext_{Id,K_1} = Ext_{Id,K_2}$.
  The extensions of two classes have to be either equal or disjoint.

Due to the semantical poverty of the concepts listed above GIM should not function as interface to applications. In consequence, external schemata expressed in another data model than GIM have to be derived from the integrated schema. One way to do this is shown in Section 5.

# 3 Homogenization in GIM

In this section, we informally describe the process of transformation of object-oriented schemata to GIM and the resolving of extensional and intensional conflicts.

The original database schemata to be federated have to be transformed into GIM. This transformation is a complex task. For simplicity, we combine here the process of transformation to GIM and the process of integration into a GIM schema into one process. In general, the input database schemata are expressed in heterogeneous data models. Therefore, at first naming, attribute and constraint conflicts have to be detected and eliminated[2].

[2] A taxonomy of conflicts is given in [SK93, Kim95].

For our discussion, we use a very simplified example focusing on the problem of solving extensional and intensional overlappings. We assume that other conflicts are resolved (e.g. naming conflicts, meta conflicts [KLK91, CL94] and attribute conflicts [LNE89]). Both example schemata are object-oriented, and attributes with identical name have identical semantics. The example schemata are taken from a library application and from a university administration, respectively (see Figure 1). The filled subclass-symbol denotes a non-exclusive specialization whereas an unfilled subclass-symbol denotes an exclusive specialization. In the second schema, a *bad guy* is a person who disregarded a library rule, e.g. he damaged a book.

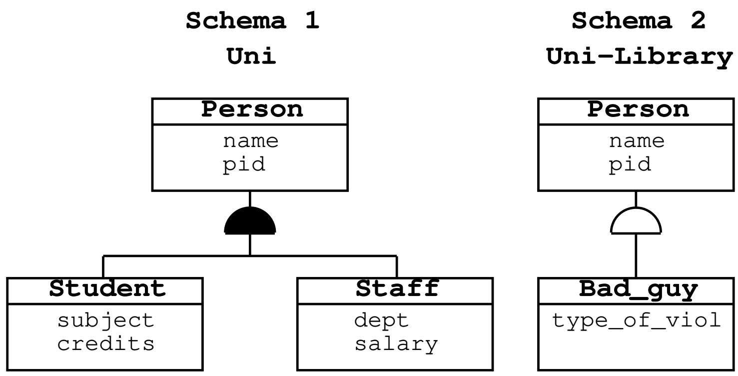


Figure 1: Two input schemata

Using this example, we will now discuss the integration process informally. A more detailed discussion of the following steps is given in [CHJ+96]. As a first step, we analyze the extensional overlappings of the five classes. This can be done for example using data inspection and key constraint analysis [ZHKF95] and must be assisted by an expert of the application domain. The result of the *analysis of class extensions* is shown in Figure 2. Each person of schema 1 is either a student or a staff member. There are persons which are both student and staff member. Each person of schema 2 is a person of schema 1. Only students can be registered as bad guys.

Based on the extension analysis, we then perform an *extension decomposition.* In this step, we generate a partition of all class instances based on all possible non-empty intersections of the input class populations. After performing this partition, we obtain eight disjoint partition classes as shown in Figure 2. For each of these partition classes we have a unique mapping to a combination of input classes and vice versa.

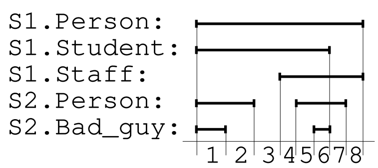


Figure 2: Extensional overlappings and decomposition

Besides the extensional part, the *intensional* dimension has to be analyzed, too. As a result of the *intension analysis* we detect the intensional overlappings depicted in Figure 3.

The *intensional decomposition* results in the following attribute partition:

```
1: name, pid
2: subject, credits
3: dept, salary
4: type_of_viol
```

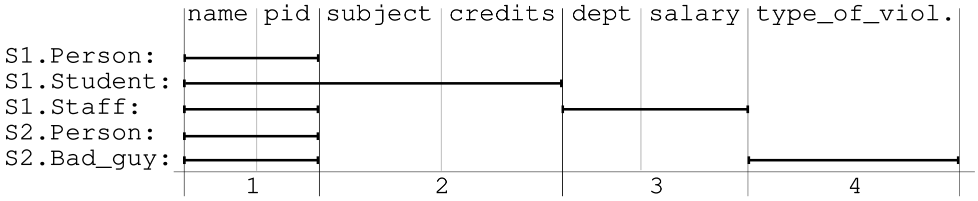


Figure 3: Intensional overlappings and decomposition

Now we are able to generate the *normalized schema*. Because the normalized schema is a GIM schema and not suited for global applications we do not use the term *integrated schema*.

An FDBS has to manage some schemata and mappings. Therefore, we introduce the super-schema $\Sigma = (\mathcal{C}s, \mathcal{M}ap_{\mathcal{C}s}, \mathcal{N}s, \mathcal{E}s, \mathcal{M}ap_{\mathcal{E}s}, \mathcal{S}ub_{\mathcal{E}s})$ with

- $\mathcal{C}s = \{S_{CS_1}, S_{CS_2}, \ldots\}$ is the set of component schemata in GIM,
- $\mathcal{M}ap_{\mathcal{C}s} = \{map_{CS_1,\mathcal{N}s}, map_{CS_2,\mathcal{N}s}, \ldots\}$
  for each component schema there exists a set function which assigns to each component class the respective classes of the normalized schema,
- $\mathcal{N}s$ is the normalized GIM schema,
- $\mathcal{E}s = \{S_{ES_1}, S_{ES_2}, \ldots\}$ is the set of external schemata,
- $\mathcal{M}ap_{\mathcal{E}s} = \{map_{ES_1,\mathcal{N}s}, map_{ES_2,\mathcal{N}s}, \ldots\}$
  for each external schema there exists a set function which assigns to each external class the respective classes of the normalized schema,
- $\mathcal{S}ub_{\mathcal{E}s} = \{sub_{ES_1}, sub_{ES_2}, \ldots\}$ with $sub_{ES_i} \subset S_{ES_i}.\mathcal{K} \times S_{ES_i}.\mathcal{K}$
  for each object-oriented external schema exists a binary relation on its external classes which defines the superclass-subclass relation

The normalized schema is the result of the transformation and integration steps. It can be derived from the extensional and intensional decompositions:

$$\mathcal{N}s = (\text{`}\mathtt{NS}', \mathcal{K}_{\mathcal{N}s}) \quad \text{with} \quad \mathcal{K}_{\mathcal{N}s} = \{K^{1,1}_{\mathcal{N}s}, \ldots, K^{i,j}_{\mathcal{N}s}, \ldots\}$$

The indices $i, j$ denote a pair of an extensional partition and an intensional partition. For each combination $(i, j)$ of extensional partitions and intensional partitions which is occupied by an input class there exists a class $K^{i,j}_{\mathcal{N}s}$ of the normalized schema. The attributes of a class $K^{i,j}_{\mathcal{N}s}$ are defined by the intensional partition $i$. Its extension can be derived from the local classes which cover the extensional partition $j$.

In our example the normalized schema $\mathcal{N}s$ contains the following classes:

$$\mathcal{K}_{\mathcal{N}s} = \{K^{1,1}_{\mathcal{N}s}, K^{1,2}_{\mathcal{N}s}, K^{1,4}_{\mathcal{N}s}, \ldots, K^{8,3}_{\mathcal{N}s}\}$$

The class $K^{1,3}_{\mathcal{N}s}$ cannot be a class of the normalized schema, because no input class covers the extensional partition 1 as well as the intensional partition 3.

The normalized schema can be represented by a matrix as shown in Figure 4. The horizontal dimension enumerates the eight partition classes resulting from the extensional decomposition. The vertical dimension is labeled by the four attribute partitions. The row 0 contains the indices of the extensional partitions and the column 0 contains the indices of the intensional partitions. We denote the value of a matrix element on column $i$ and row $j$ by $Mat^{i,j}$. The order of the

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | X | X | X | X | X | X | X | X |
| 2 | X | X | X | X | X | X | | |
| 3 | | | | X | X | X | X | X |
| 4 | X | | | | | X | | |

Figure 4: Decomposed Schema

rows and the order of the columns in the matrix is irrelevant. Only the index row and the index column must be the row 0 and column 0, respectively.

This notation can be directly used to derive view classes following the GIM notation. Each rectangle completely filled with crosses is a candidate for a view class. For example, the first row is a candidate for a class containing all instances and having the attributes `name` and `pid`.

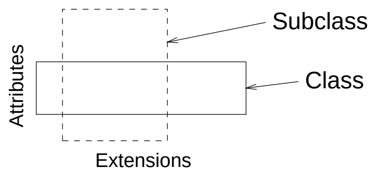


Figure 5: Class and Subclass Candidates in GIM

Candidates for subclasses are rectangles, which are horizontally included in another class rectangle but vertically completely cover it. This is graphically depicted in Figure 5. With other words, a subclass contains a subset of the superclass instance population and a superset of the superclass attributes.

However, the order of both attribute and class labels in the diagrammatic notation is arbitrarily chosen. As a result, a different order of the labels may show different rectangles as candidates corresponding to different class hierarchies for external classes.

## 4 Criteria of Quality

The normalized GIM schema is the basis for the generation of external schemata. The normalized schema itself can be an external schema because it is complete and correct. However, the demand for an understandable external integrated schema is not fulfilled. For global applications it is too restrictive that GIM classes must have exclusive or identical extensions. Therefore, classes of the normalized schema have to be composed to greater ones. If an external schema is an object-oriented one then the external classes should be in subclass-relations to each other. Each external class can be defined by its mapping to classes of the normalized schema.

The problem is: **How to compose normalized GIM classes to greater ones?**

Before we can give a solution to this question, we sketch some criteria of quality attributed to an external schema $S_{ES}$. Our criteria are adopted from [BLN86] and adapted to our scenario.

- Correctness:

  To all attributes of an object of the external class $K_{ES}$ their values must be derivable. In graphically form this means: the area of the composed normalized classes must be a rectangle in at least one order of the extensional and intensional indices.
  $\forall i_1, j_1, i_2, j_2 : K_{\mathcal{N}s}^{i_1,j_1} \in map_{ES,\mathcal{N}s}(K_{ES}) \wedge K_{\mathcal{N}s}^{i_2,j_2} \in map_{ES,\mathcal{N}s}(K_{ES}) \Rightarrow$
  $K_{\mathcal{N}s}^{i_1,j_2} \in map_{ES,\mathcal{N}s}(K_{ES})$

  The $sub_{ES}$-relations must be correct. Subclasses have a subset of the instance population and a superset of the attributes.

- $\forall (K_{ES}^1, K_{ES}^2) \in sub_{ES} : K_{\mathcal{N}s}^{i,j} \in map_{ES,\mathcal{N}s}(K_{ES}^2) \Rightarrow \exists k : K_{\mathcal{N}s}^{k,j} \in map_{ES,\mathcal{N}s}(K_{ES}^1)$
- $\forall (K_{ES}^1, K_{ES}^2) \in sub_{ES} : K_{\mathcal{N}s}^{i,j} \in map_{ES,\mathcal{N}s}(K_{ES}^1) \Rightarrow \exists k : K_{\mathcal{N}s}^{i,k} \in map_{ES,\mathcal{N}s}(K_{ES}^2)$

- Completeness:

  Each normalized GIM class has to be a component of at least one external class.
  $\forall K_{\mathcal{N}s}^{i,j} \in \mathcal{K}_{\mathcal{N}s} : \exists K_{ES} \in \mathcal{K}_{ES} : K_{\mathcal{N}s}^{i,j} \in map_{ES,\mathcal{N}s}(K_{ES})$

  The $sub_{ES}$-relation must be complete. In order to formalize this requirement, we construct the transitive closure $sub_{ES}^*$ from $sub_{ES}$. All external classes which can be in a subclass-relation must be in the $sub_{ES}^*$-relation.
  $\forall K_{ES}^1, K_{ES}^2 \in \mathcal{K}_{ES} : (\forall K_{\mathcal{N}s}^{i,j} \in map_{ES,\mathcal{N}s}(K_{ES}^1) : \exists K_{\mathcal{N}s}^{i,k} \in map_{ES,\mathcal{N}s}(K_{ES}^2)) \wedge (\forall K_{\mathcal{N}s}^{i,j} \in map_{ES,\mathcal{N}s}(K_{ES}^2) : \exists K_{\mathcal{N}s}^{k,j} \in map_{ES,\mathcal{N}s}(K_{ES}^1)) \wedge K_{ES}^1 \neq K_{ES}^2 \Rightarrow (K_{ES}^1, K_{ES}^2) \in sub_{ES}^*$

- Minimality in view of completeness and correctness:

  Among complete and correct external schemata we prefer the schema with the minimal number of classes.

  Among complete and correct external schemata we prefer the schema with the minimal number of subclass-relations.

- Understandability:

  Each object is instance of external classes forming exactly one path of the inheritance hierarchy:
  $\forall K_{\mathcal{N}s}^{i,j} \in \mathcal{K}_{\mathcal{N}s} : \forall K_{ES}^1, K_{ES}^2 \in S_{ES}.\mathcal{K} : \forall (K_{ES}^{1'}, K_{ES}^1) \in sub_{ES} : \forall (K_{ES}^{2'}, K_{ES}^2) \in sub_{ES} : K_{\mathcal{N}s}^{i,j} \in map_{ES,\mathcal{N}s}(K_{ES}^1) \wedge K_{\mathcal{N}s}^{i,j} \notin map_{ES,\mathcal{N}s}(K_{ES}^{1'}) \wedge K_{\mathcal{N}s}^{i,j} \in map_{ES,\mathcal{N}s}(K_{ES}^2) \wedge K_{\mathcal{N}s}^{i,j} \notin map_{ES,\mathcal{N}s}(K_{ES}^{2'}) \Rightarrow K_{ES}^1 = K_{ES}^2$
  This is also required by the ODMG-93 standard.

  In our opinion, exclusive specialization improve the understandability of an external schema:
  $\forall K_{ES}, K_{ES}^1, K_{ES}^2 \in S_{ES}.\mathcal{K} : \forall K_{\mathcal{N}s}^{i,j} \in S_{\mathcal{N}s}.\mathcal{K} : (K_{ES}^1, K_{ES}), (K_{ES}^2, K_{ES}) \in sub_{ES} \wedge K_{\mathcal{N}s}^{i,j} \in map_{ES,\mathcal{N}s}(K_{ES}^1) \wedge K_{\mathcal{N}s}^{i,k} \in map_{ES,\mathcal{N}s}(K_{ES}^2) \Rightarrow K_{ES}^1 = K_{ES}^2$

- Similarity to a given local schema:

  A measure of similarity between a local class and an external schema is $Z_{K_{LS},ES}$ which specifies the number of external classes which are necessary to represent the extension of the local class. The similarity of an external schema to a local schema can be quantified by the following formula:

  $$Z_{LS,ES} \stackrel{\text{def}}{=} \sum_{K_{LS} \in S_{LS}.\mathcal{K}} \frac{Z_{K_{LS},ES}}{\mid S_{LS}.\mathcal{K} \mid}$$

  $Z_{LS,ES}$ should be minimal.

The proposed requirements have different priorities. Each external integrated schema has to be correct and complete. However, the requirements for minimality and understandability are sometimes contradictory. Compromises must be made. Similarity of an external schema is sometimes necessary and can be quantified by the given formula.

With the criteria described above we are able to assess the quality of a given external schema.

# 5 Algorithm for Deriving External Schemata

A naive approach to generate the "best" external integrated schema is to generate all external classes by finding all possible compositions of normalized classes. After that all possible external schemata can be generated by computing the power set on the set of all external classes. By means of the quality criteria the "best" external schema can then be found. The disadvantage of this approach is its complexity. The complexity is exponentially dependent from the number of extensional partitions of the normalized schema.

A more practicable solution is given in this section. The steps of the algorithm are explained using our example. In our algorithm we transform the matrix in such a way that we are able to derive directly external classes from rectangulars.

At the beginning of the algorithm some variables must be initialized. We need an empty external schema $S_{ES}.\mathcal{K} := \emptyset$, an empty map function $map_{ES,\mathcal{N}s} := \emptyset$, and an empty subclass relation $sub_{ES,\mathcal{N}s} := \emptyset$. New generated external classes will be inserted into $S_{ES}.\mathcal{K}$. We distinguish the new generated classes by giving them a consecutively numbered index which starts from zero. The index counter will be assigned to the external classes following their insertion order. Furthermore we need the variable $Sup := 0$ in order to generate subclass relations. $ze$ is the number of rows and $sp$ is the number of columns.

1. Intensional and extensional sorting:

   We consider a row as a binary number and introduce a function on the present column order which computes to each row (without row 0) its row value:

   $$\forall i \in [1 \ldots ze) \, : row(i) = \sum_{j \in [1 \ldots sp)} Mat^{j,i} * 2^{sp-j-1}$$

   We consider a column as binary number and introduce a further function on the present row order which computes to each column (without column 0) its column value:

   $$\forall i \in [1 \ldots sp) \, : col(i) = \sum_{j \in [1 \ldots ze)} Mat^{i,j} * 2^{ze-j-1}$$

   The matrix has to be transformed by exchanging rows and exchanging columns so that the following both conditions are true:

   $$\begin{aligned} \forall i \in [1 \ldots ze-2] &\quad : \quad row(i) \geq row(i+1) \\ \forall i \in [1 \ldots sp-2] &\quad : \quad col(i) \geq col(i+1) \end{aligned}$$

   Notice that an exchange of rows can change the computed column values and vice versa. Both conditions must be true basing on the current matrix order.

   There is an algorithm which fulfills both conditions. First of all the rows have to be sorted in correspondence to their row values. Then the columns have to be sorted in correspondence to their column values. Since the column sorting changes the row values a row disorder is possible. Therefore, we continue both steps until the conditions are true. It can easily be proven that these both steps have to be repeated at most as often as different columns exist.

   If we apply the row sort operation on the matrix of our example (left matrix depicted in Figure 6) then we obtain the middle matrix. The right matrix is the middle matrix sorted by column values. It fulfills both conditions mentioned above.

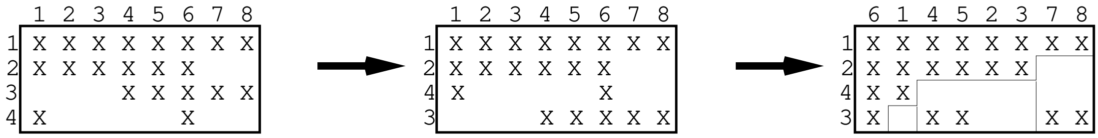

Figure 6: Row and column sorting

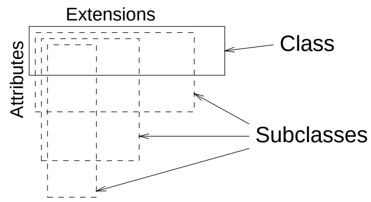


Figure 7: Subclass candidates in triangle shape

2. Deriving external classes:

   After the sorting steps have been carried out most crosses in the matrix are clustered in the left upper corner of the diagram building the shape of a triangle. This triangle can be interpreted as rectangles building a branch of a class hierarchy as depicted in Figure 7.

   As we can see in Figure 6 the diagonal of the triangle has some 'step corners' (at positions (8,1),(6,2),(2,3),(1,4) in our example). These 'corners' have to be detected. The sequence $Cor_{\mathcal{M}at}$ contains these 'corners' in the order of their vertical position within the matrix that is the uppermost 'corner' is the first 'corner':

   $Cor_{\mathcal{M}at} =< Pos_0^{i_0,j_0}, Pos_1^{i_1,j_1}, \ldots Pos_{n-1}^{i_{n-1},j_{n-1}} >$ with:
   $\forall k \in [0 \ldots n) : \forall ii \in [1 \ldots i_k] : Mat^{ii,j_k} = 1 \wedge (i_k + 1 \geq sp \vee Mat^{i_k+1,j_k} = 0) \wedge$
   $\forall jj \in [1 \ldots j_k] : Mat^{i_k,jj} = 1 \wedge (j_k + 1 \geq ze \vee Mat^{i_k,j_k+1} = 0) \wedge (k < n-1 \Rightarrow j_k > j_{k+1})$

   The sequence $Cor_{\mathcal{M}at}$ can be generated as follows:

   A cursor starts from the left upper position (1,1) of the matrix and moves to the right until the element at the right of the cursor is not set (0) or the right boundary is reached. If the matrix element directly below is not set (0) or the lower boundary is reached then the cursor position is a 'corner position' and must be inserted into the sequence. Then we start the same procedure at the next row starting from column 1 until the lower boundary is reached.

   Now we set the variable $anz :=\mid S_{ES}.\mathcal{K} \mid$.

   From the sequence $Cor_{\mathcal{M}at}$ we are able to generate external classes. From each 'corner' $k$ we generate an external class which contains all normalized classes within the rectangle formed by column 1, column $i_k$, row 1 and row $j_k$. The order of the 'corners' within $Cor_{\mathcal{M}at}$ builds the specialization order of the generated external classes:

   **for** k:=0 **to** n-1
   **begin**
   $S_{ES}.\mathcal{K} := S_{ES}.\mathcal{K} \cup \{K_{ES}^{anz+k} \mid Pos_k^{i_k,j_k} = Cor_{\mathcal{M}at}(k)\}$ with
   $map_{ES,\mathcal{N}s} = map_{ES,\mathcal{N}s} \cup$
   $(K_{ES}^{anz+k}; \{K_{\mathcal{N}s}^{m,n} \mid o \in [1 \ldots i_k] : p \in [1 \ldots j_k], m = Mat^{o,0}, n = Mat^{0,p}\})$ and
   $$sub_{ES,\mathcal{N}s} = sub_{ES,\mathcal{N}s} \cup \begin{cases} \{(K_{ES}^{anz+k}, K_{ES}^{anz+k-1})\} : k > 0 \\ \{(K_{ES}^{anz}, Sup)\} : Sup \neq 0 \wedge k = 0 \\ \emptyset : else \end{cases}$$
   **end**

The resulting external classes constitute a branch of an inheritance tree. The first class is the root of the inheritance tree. Extensions and intensions of the generated external classes can be derived from the information shown in Figures 3 and 2. Using these information, a designer has to find class names corresponding to the semantics of these derived classes. As result we obtain the following four classes from the right matrix of Figure 6:

```
Extension        Intension  Class
--------------------------------------------
1,2,3,4,5,6,7,8  1          Person
1,2,3,4,5,6      1,2        Student
1,6              1,2,4      Bad_guy
6                1,2,3,4    Assisting_bad_guy
```

The intension *Att* of a generated external class can be derived from the map-functions and is not further considered here.

3. Submatrices:

   After these steps have been carried out we may have some crosses left. In our example, left crosses are at positions (3,4), (4,4), (7,4), and (8,4). From such remaining crosses we construct the remaining branches of the inheritance tree by recursively applying our algorithm. The area below the triangle is partitioned by vertical lines starting from the 'corner' positions and ending at the lower boundary. Each resulting partitioned area can be considered as an isolated submatrix and if it contains at least one cross, it has to be transformed into the triangle form by a further intensional and extensional sorting on the isolated submatrix. These triangles correspond to other branches of the constructed inheritance tree. They are located at specific positions in the diagram defined by the variable *Sup*.

   Each partition constitutes a submatrix.

   We generate from each 'corner' position $\forall m \in [0 \dots n-1) \, : Pos_m^{i_m, j_m}$ a submatrix $\mathcal{M}at_{m+1}$ with number of columns $sp_{m+1} = i_m - i_{m+1} + 1$ and number of rows $ze_{m+1} = ze - j_m$:

   - $\forall k \in [1 \dots sp_{m+1}) \, : Mat_{m+1}^{k+i_{m+1},0} = Mat^{k+i_{m+1},0}$
   - $\forall k \in [1 \dots ze_{m+1}) \, : Mat_{m+1}^{0,k} = Mat^{0,k+i_m}$
   - $\forall k \in [1 \dots sp_{m+1}) \, : \forall l \in [1 \dots ze_{m+1}) \, : Mat_{m+1}^{k,l} = Mat^{k+i_{m+1},l+j_m}$

   In addition to these generated submatrices we generate a further submatrix $\mathcal{M}at_n$ with number of columns $sp_n = sp - i_0$ and number of rows $ze_n = ze$:

   - $\forall k \in [1 \dots ze_n) \, : Mat_n^{0,k} = Mat^{0,k}$
   - $\forall k \in [1 \dots sp_n) \, : \forall l \in [0 \dots ze_n) \, : Mat_n^{k,l} = Mat^{k+i_0,l}$

   In our example two non-empty submatrices are generated. The first submatrix contains two crosses at position (7,4) and (8,4) and the second one contains two crosses at position (3,4) and (4,4). This can be seen in Figure 6.

4. Recursion:

   On each submatrix $\mathcal{M}at_m$ generated by the previous step which is not empty, that is

   $$\exists i \in [1 \dots sp_m) \, : \exists j \in [1 \dots ze_m) \, : Mat_m^{i,j} = 1$$

the steps starting from step 1 must be applied. The variable $Sup$ contains the superclass of the tree branch generated from the submatrix $\mathcal{M}at_m$ and is defined by:

$$Sup = \begin{cases} K_{ES}^{anz+m-1} : m \neq n \\ 0 : else \end{cases}$$

If we apply this recursion to our example we obtain two further external classes:

| Extension | Intension | Class |
|---|---|---|
| 4,5 | 1,2,3 | Assistant |
| 7,8 | 1,3 | Staff-Student |

5. Discriminants:

   In the last step, we have to analyze whether all columns are distinguished by their mapping to external classes. In order to avoid information loss we have to add an artificial *discriminant* attribute to the respective external class if some columns are not distinguishable.

   In our example columns 7 and 8 are not distinguishable. Therefore we have to enhance the intension of the external class `Person` by a discriminant attribute which contains the column number of a given object. As result we obtain the external schema as depicted in Figure 8.

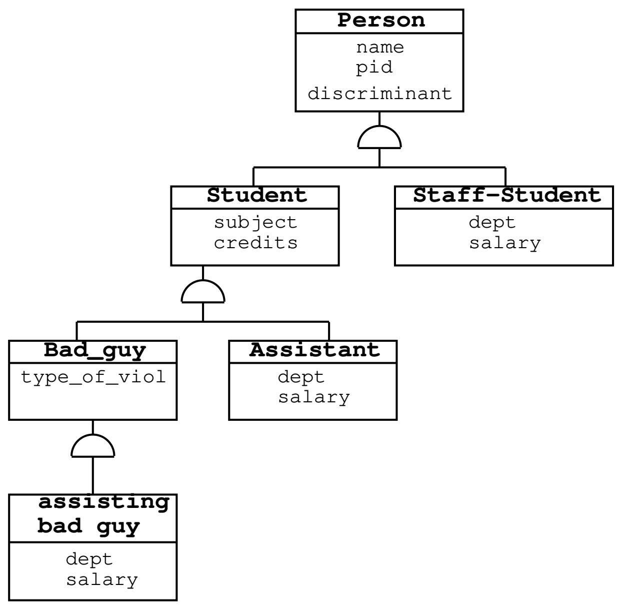


Figure 8: External schema

How are the criteria of quality of Section 4 fulfilled in our example?

It can be easily seen that an external schema generated by our algorithm is complete, correct and understandable. However, how can similarity to a given local schema be generated? The priorities of the input extension order influence the results of the algorithm. If the extensional partitions of a local class have the highest priority (most left in the matrix) then there is exactly one external class in the resulting inheritance tree, which corresponds to the local class.

As an example, we order the extensions in such a way, that all extensions (4,5,6,7,8) which correspond to `Staff` having the highest priority. The result of the sort steps is shown in Figure 9.

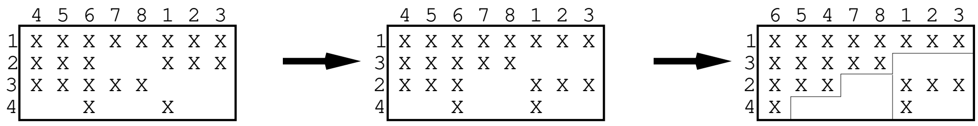

Figure 9: Staff-oriented composition

The derived inheritance tree is depicted in Figure 10. It is different from the tree derived before (cf. Figure 8). In contrast to the first tree, all staff members are clustered in one branch of the inheritance tree.

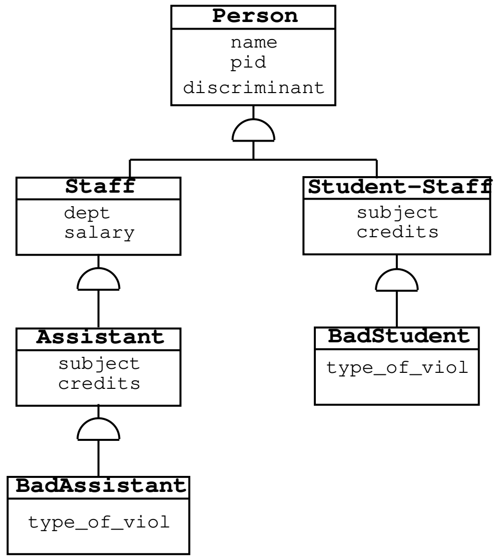


Figure 10: Staff-oriented external schema

Therefore, before our algorithm should be applied the designer has to order the extensional dimension by *priority*. The order of the extensional dimension influences the result of the derivation process: classes corresponding to extensions with higher priority will be located closer to the root of the inheritance tree of the resulting external schema.

Our algorithm generates a minimal, complete, correct, and understandable external schema. We demonstrated this using an example.

# 6 Conclusion and Outlook

We presented a framework for schema integration including a practicable algorithm to derive class hierarchies meeting the demand for minimality, correctness, completeness and understandability. Whereas the derivation of an optimal inheritance tree is not tractable, our algorithm has the complexity of $O(t^4 * log(t))$ where $t$ is the maximum number of rows and columns of the GIM representation (which corresponds to the number of attributes and disjoint class populations of the original input schemata). The result of the algorithm can be influenced by prioritizing extensional GIM classes.

Future work will concentrate on integrating more semantical assertions (e.g. integrity constraints) into the framework. Moreover, the aspect of multiple inheritance in derived inheritance trees has to be investigated. Other data models as target for external views have to be analyzed, too. The presented methods can be applied to scenarios different from the integration problem, e.g., for schema evolution, deriving views in object-oriented database systems, and classification problems.

A design tool supporting the presented methods and algorithms is currently under development. The presented work is part of a larger project on federating heterogeneous databases [CHJ+96]. In this context the integration of the presented methods into a larger design framework will be investigated. Especially, the role of constraints in federated database design is one focal point of the project.

## References


[BLN86] C. Batini, M. Lenzerini, and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, December 1986.

[Bra93] S.E. Bratsberg. *Evolution and Integration of Classes in Object-Oriented Databases.* Dissertation, The Norwegian Institute of Technology, University of Trondheim, June 1993.

[CHJ+96] S. Conrad, M. Höding, S. Janssen, G. Saake, I. Schmitt, and C. Türker. Integrity Constraints in Federated Database Design. Preprint Nr. 2, Fakultät für Informatik, Universität Magdeburg, April 1996.

[CL94] J. Chomicki and W. Litwin. Declarative Definition of Object-Oriented Multidatabase Mappings. In T. Özsu, U. Dayal, and P. Valduriez, editors, *Distributed Object Management*, pages 375–392. Morgan Kaufmann Publishers, San Mateo, CA, 1994.

[Dup94] Y. Dupont. Resolving Fragmentation Conflicts in Schema Integration. In P. Loucopoulos, editor, *Entity-Relationship Approach — ER'94, Proc. of the 13th Int. Conf., Manchester, UK*, pages 513–532. LNCS 881, Springer-Verlag, December 1994.

[GCS95] M. Garcia-Solaco, M. Castellanos, and F. Saltor. A Semantic-Discriminated Approach to Integration in Federated Databases. In S. Laufmann, S. Spaccapietra, and T. Yokoi, editors, *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS'95), Vienna, Austria*, pages 19–31, May 1995.

[Kim95] W. Kim, editor. *Modern Database Systems.* ACM Press, New York, 1995.

[KLK91] R. Krishmamurthy, W. Litwin, and W. Kent. Language Features for Interoperability of Databases with Schematic Discrepancies. In Y. Kambayashi, M. Rusinkiewicz, and A. Sheth, editors, *Proc. of the 1st Int. Workshop on Interoperability in Multidatabase Systems (IMS'91), Kyoto, Japan*, pages 144–151. IEEE Computer Society Press, April 1991.

[LNE89] J.A. Larson, S.B. Navathe, and R. Elmasri. A Theory of Attribute Equivalence in Databases with Application to Schema Integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.

[PBE95] E. Pitoura, O. Bukhres, and A. K. Elmagarmid. Object Orientation in Multidatabase Systems. *ACM Computing Surveys*, 27(2):141–195, June 1995.

[RPRG94] M. P. Reddy, B. E. Prasad, P. G. Reddy, and A. Gupta. A Methodology for Integration of Heterogeneous Databases. *IEEE Transactions on Knowledge and Data Engineering*, 6(6):920–933, December 1994.

[SCG91] F. Saltor, M. Castellanos, and M. Garcia-Solaco. Suitability of Data Models as Canonical Models for Federated Databases. In J. Clifford and R. King, editors, *Proc. of the 1991 ACM SIGMOD Int. Conf. on Management of Data, Denver, Colorado, SIGMOD RECORD 20(2)*, pages 44–48. ACM Press, June 1991.

[Sch95] I. Schmitt. Flexible Integration and Derivation of Heterogeneous Schemata in Federated Database Systems. Preprint Nr. 10, Fakultät für Informatik, Universität Magdeburg, November 1995.

[SK93] A. Sheth and V. Kashyap. So Far (Schematically) yet So Near (Semantically). In D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, editors, *Interoperable Database Systems (DS-5), Proc. of the IFIP WG 2.6 Database Semantics Conf., Lorne, Victoria, Australia, November, 1992*, pages 283–312. North-Holland, 1993.

[SL90] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[SPD92] S. Spaccapietra, C. Parent, and Y. Dupont. Model Independent Assertions for Integration of Heterogeneous Schemas. *The VLDB Journal*, 1(1):81–126, 1992.

[TS93] C. Thieme and A. Siebes. Schema Integration in Object-Oriented Databases. In C. Rolland, F. Bodart, and C. Cauvet, editors, *Proc. of the 5th Conf. on Advanced Information System Engineering (CAiSE'93), Paris, France*, pages 55–70. LNCS 685, Springer-Verlag, June 1993.

[ZHKF95] G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Using Object Matching and Materialization to Integrate Heterogeneous Databases. In S. Laufmann, S. Spaccapietra, and T. Yokoi, editors, *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS'95), Vienna, Austria*, pages 4–18, May 1995.