Specifying Database Updates Using A Subschema

Sonja Ristić¹, Pavle Mogin², Ivan Luković³

¹ Business College, V. Periça 4, 21000 Novi Sad, Yugoslavia
² Victoria University of Wellington, School of Mathematical and Computing Sciences, P.O. Box 600, Wellington, New Zealand
³ University of Novi Sad, Faculty of Technical Sciences, Trg D. Obradovića 6, 21000 Novi Sad, Yugoslavia

Abstract. The notion of a subschema, as a formal and abstract definition of data, constraints, and database update activities that are needed to make a transaction program, is introduced in the paper. Subschema is a component of a transaction program specification. It is designed using a user request and an existing relational database schema. The principles of a database update using subschema concepts are introduced at the abstraction level of instances to express the fact that a subschema and the corresponding database schema must satisfy certain conditions to allow safe database updates using a program made in accordance with subschema concepts. The conditions of the formal subschema and database schema consistency are introduced at the schema abstraction level, as well. It is shown that formal consistency implies database update principles, which leads to the conclusion that a subschema design process should adhere to formal consistency conditions if it is to constitute a component of a transaction program specification.

1. Introduction

Specifying transaction programs is an important and time-consuming methodological task in the process of information system design. The design specification of a transaction program is a formalized description of that program. It is aimed at supporting the implementation of an end user business task that is defined by means of a user request. It is usually assumed that the transaction programs will be executed against a database. Accordingly, the basic structural components of a transaction program specification are:

- A specification of the human-computer interface;
- A data definition; and
- A formal description of a data processing procedure.

In this paper, the data definition part of the transaction program specification is called a subschema. A subschema is a formal and abstract definition of data, constraints, and database update activities that are needed to make a transaction program.

A subschema is designed with respect to a user request and an existing relational database schema. Accordingly, a subschema describes data of a relatively small part of a database, and consists of a set of relation schemes and a set of interrelation constraints. Each relation scheme of a subschema consists of a set of attributes and a set of local constraints. A role and a set of modifiable attributes, defining possible database update activities, are also assigned to each relation scheme. Each relation scheme of a subschema may be considered as a view on a single base relation scheme. Subschema instances are not materialized.

To allow safe database updates using a program made in accordance with a subschema, the subschema has to be consistent with the corresponding database schema in a formal sense. This consistency is based on the principles of a database update using subschema concepts that are introduced in the paper, as well. From an end user point of view, using a transaction program means applying a sequence of update activities on a subschema instance. The fact that a subschema instance does not physically exist has two consequences. These are:

- A subschema instance has to be imagined as the result of applying appropriate join, select and project operations on a database instance; and
- The only way to execute a transaction program is to transform the sequence of update activities defined over the subschema concepts into a corresponding sequence of update activities defined over the database schema concepts and then apply them on the database instance.
Normally, this update changes only that part of the database instance that is related to subschema concepts. An intuitive expectation is that applying the join, select and project operations on the updated database instance should produce the same subschema instance that would be obtained by direct updating the initial subschema instance. The aim of this paper is to show that it is possible to formulate conditions at schema level of abstraction, by means of which this expectation will be preserved.

We have chosen the concepts of the relational data model to define: the notion of a subschema, the database update principles, and the formal schema and subschema consistency, since we needed its powerful mathematical formalism to express our ideas and solutions in a precise way. But we believe that our results may be applied to other data models, too.

The notion of the formal consistency is defined in the paper. It is shown that the formal consistency implies database update principles. This result leads to the conclusion that a subschema design process should adhere to formal consistency conditions if this subschema is intended to constitute a component of a transaction program specification.

The standard notion of a database SQL view is in some sense related to the notion of the subschema as introduced in the paper. Both of them represent an interface between a user (or program) and the database. This interface provides a user with a specific way of looking at the data in the database. Apart from that, a subschema is as a structure over the simple views, and that structure contains information about database constraints and allowable database operations. Additionally, a subschema is a component of a program specification, and this specification may be implemented in many different ways, under many, even technologically different, programming systems.

The issues of relational database update using views are considered in [6, 7, 8, 9]. The conclusions of considerations in [6] are formulated as recommendations for the vendors of RDBMS. Some of the contemporary RDBMS’s allow database update using a view if the requested update activity may be unambiguously performed on the base relations; otherwise it is rejected. This might lead to the conclusion that it would be sufficient to use a view instead of a relatively complex structure as a subschema in the transaction program specification. We believe that a view carries insufficient information to make a correct database update program. In particular, a view hides from a programmer information about:

- The structure over the needed base relations,
- The constraints needed to make an user-friendly screen form control program, and
- The allowed operations on the base relations.

All this information is contained in a subschema.

In [8], Dayal and Bernstein explore the problem of transforming update operations that are defined over either a complex or a simple view into corresponding update operations over a database schema in detail. In our paper the transforming problem is simpler, since each relation scheme of a subschema corresponds to exactly one relation scheme from the database schema. Some consequences of this restriction and possible solutions will be briefly discussed at the end of the paper.

In [9], Bancilhon and Spyropoulos consider the problem of view updateability in a very general and formal way at the level of instance abstraction, by introducing the concepts of complementary view and g-translatability of update operations. We see our approach to database update principles as a special and practically important case, when a view instance and its complement make a database schema instance.

In [7], Langerak also considers the problem of view updateability, but functional dependencies, in the presence of null values, are the only integrity constraints that have been taken into consideration. Only total projections of database relations as view instances are analyzed and the selection operator is not considered.

This paper extends the results presented in [7, 8, 9] by introducing and emphasizing the roles of allowable operations and modifiable attributes in the definition of the database update principles.

Apart from the Introduction and Conclusion, the paper has four sections. Section two formally introduces the notion of a subschema. Sections three and four describe the principles of a database update using subschema concepts, and Section five is concerned with formal consistency.

### 2. The Subschema

A relational database schema is a pair \((S, I)\), where \(S\) is a set of relation schemes and \(I\) is a set of interrelation constraints. It is supposed in the paper that the database schema is produced using a well-defined methodological approach, and no further attention is paid to this issue.

Each relation scheme from \(S\) is a named triple: \(N(R, C, Kp(R))\), where \(N\) is a unique name, \(R\) is an attribute set, and \(C\) is a specification of constraints. A relation scheme will be often referred simply by its name \(N\). The specification of constraints \(C\) is a triple \((\mathcal{K}, \tau(N), \mathcal{Uniq}(N))\), where \(\mathcal{K}\) is a set of keys, \(\tau(N)\) will be called tuple integ-
Example 1. Suppose a database schema \( S, I \) and subschemas \( P_1(S_1, I_1), P_2(S_2, I_2), P_3(S_3, I_3) \) are given, where:

- \( S = \{ \text{ORDER}(R_1, C_1), \text{SHIPMENT}(R_2, C_2), \text{CUSTOMER}(R_3, C_3) \} \);  
  - \( R_1 = \{ \text{OrdId}, \text{OrdDate}, \text{CustId}, \text{Origin}, \text{Total} \} \);  
  - \( C_1 = (K_1, \tau(N_1), \text{Uniq}(N_1)) = (\{ \text{OrdId} \}, \tau(\text{ORDER}), \emptyset) \);  
  - \( K_0(R_1) = \{ \text{OrdId} \} \);  
  - \( R_2 = \{ \text{ShipId}, \text{Ordid}, \text{ShipDate}, \text{ShipTotal} \} \);  
  - \( C_2 = (K_2, \tau(N_2), \text{Uniq}(N_2)) = (\{ \text{ShipId} \}, \tau(\text{SHIPMENT}), \emptyset) \);  
  - \( K_0(R_2) = \{ \text{ShipId} \} \);  
  - \( R_3 = \{ \text{CustId}, \text{ CustName}, \text{ CustAddr} \} \);  
  - \( C_3 = (K_3, \tau(N_3), \text{Uniq}(N_3)) = (\{ \text{CustId} \}, \tau(\text{CUSTOMER}), \emptyset) \);  
  - \( K_0(R_3) = \{ \text{CustId} \} \);  
- \( I = \{ \text{ORDER}[\text{CustId}] \subseteq \text{CUSTOMER}[\text{CustId}], \text{SHIPMENT}[\text{OrdId}] \subseteq \text{ORDER}[\text{OrdId}], \text{CUSTOMER}[\text{CustId}] \subseteq \text{ORDER}[\text{CustId}] \} \).

- \( S_1 = \{ \text{Dom}_\text{Shipped}_\text{Order}(R_1^2, C_1^2), \text{Shipment}(R_2^2, C_2^2) \} \);  
  - \( R_1^2 = \{ \text{OrdId}, \text{ OrDate}, \text{ CustId}, \text{ Origin}, \text{ Total} \} \);  
  - \( C_1^2 = (K_1^2, \tau(N_1^2), \text{Uniq}(N_1^2)) = (\{ \text{OrdId} \}, \tau(\text{Dom}_\text{Shipped}_\text{Order}), \emptyset) \);  
  - \( K_0(R_1^2) = \{ \text{OrdId} \} \);  
  - \( R_2^2 = \{ \text{ShipId}, \text{ Ordid}, \text{ ShipDate}, \text{ ShipTotal} \} \);  
  - \( C_2^2 = (K_2^2, \tau(N_2^2), \text{Uniq}(N_2^2)) = (\{ \text{ShipId} \}, \tau(\text{Shipment}), \emptyset) \);
• $K_1(R_1) = \{\text{ShipId}\}$; 
• $\text{`Role}(P_1, \text{Dom}_\text{Shipped Order}) = \text{`Role}(P_1, \text{Shipment}) = \{i, r\}$; 
• $\text{`Mod}(P_1, \text{Dom}_\text{Shipped Order}) = \text{`Mod}(P_1, \text{Shipment}) = \emptyset$; 
• $\exists \text{`Role}(P_1, \text{Dom}_\text{Shipped Order}) = \text{ORDER}$; 
• $\exists \text{`Role}(P_1, \text{Shipment}) = \text{SHIPSMENT}$; 

$I_1 = \{\text{Shipment}[\text{OrdId}] \subseteq \text{Dom}_\text{Shipped Order}[\text{OrdId}], \text{Dom}_\text{Shipped Order}[\text{OrdId}] \subseteq \text{Shipment}[\text{OrdId}]\}$; 

$S_2 = \{\text{Order}(R_1', C_1', \ldots), \text{Customer}(R_1', C_1', \ldots)\};$ 
$R_1' = \{\text{OrdId}, \text{OrDate}, \text{Customer, Origin, Total}\};$ 
$C_1' = (\text{`K}_1', \tau(\text{N}_1'), \text{Uniq}(\text{N}_1')) = (\{\text{OrdId}\}, \tau(\text{Order}), \emptyset);$ 
$K_2(R_1') = \{\text{OrdId}\};$ 
• $\text{`Role}(P_2, \text{Order}) = \{i, r\}$; 
• $\text{`Mod}(P_2, \text{Order}) = \emptyset$; 
• $\exists \text{`Role}(P_2, \text{Order}) = \text{ORDER}$; 
• $R_2' = \{\text{CustId, CustName, CustAdrr}\}$; 
• $C_2' = (\text{`K}_2', \tau(\text{N}_2'), \text{Uniq}(\text{N}_2')) = (\{\text{CustId}\}, \tau(\text{Customer}), \emptyset);$ 
• $K_3(R_2') = \{\text{CustId}\};$ 

$I_2 = \{\text{Order}[\text{CustId}] \subseteq \text{Customer}[\text{CustId}]\}$; 

$S_3 = \{\text{Order}(R_1', C_1', \ldots), \text{Shipment}(R_1', C_1', \ldots)\};$ 
$R_1' = \{\text{OrdId, OrDate, CustId, Origin, Total}\};$ 
$C_1' = (\text{`K}_1', \tau(\text{N}_1'), \text{Uniq}(\text{N}_1')) = (\{\text{OrdId}\}, \tau(\text{Order}), \emptyset);$ 
$K_2(R_1') = \{\text{OrdId}\};$ 
• $\text{`Role}(P_3, \text{Order}) = \{r\}$; 
• $\text{`Mod}(P_3, \text{Order}) = \emptyset$; 
• $\exists \text{`Role}(P_3, \text{Order}) = \text{ORDER}$; 
$R_2' = \{\text{ShipId, OrdId, ShipDate, ShipTotal}\};$ 
$C_2' = (\text{`K}_2', \tau(\text{N}_2'), \text{Uniq}(\text{N}_2')) = (\{\text{ShipId}\}, \tau(\text{Shipment}), \emptyset);$ 
$K_3(R_2') = \{\text{ShipId}\};$ 
• $\text{`Role}(P_3, \text{Shipment}) = \{i, r\}$; 
• $\text{`Mod}(P_3, \text{Shipment}) = \emptyset$; 
• $\exists \text{`Role}(P_3, \text{Shipment}) = \text{SHIPSMENT}$; 

$I_3 = \{\text{Shipment}[\text{OrdId}] \subseteq \text{Order}[\text{OrdId}]\}$; 

Subschema $P_1$ is aimed for entry of domestic orders and shipments within one transaction, $P_2$ is aimed for order entry, and $P_3$ is aimed for entry of shipments that are initiated by orders. Subschema $P_1$ is associated with a business unit whose task is to control domestic orders. In this business unit users are not interested in customer data except for customer id number. The attribute domain constraint for Origin in subschema $P_1$ is more restrictive than the corresponding constraint in the database schema, because only the orders of domestic customers are required. Subschema $P_3$ also contains the inclusion dependency Dom_Shipped_Order[OrdId] \subseteq Shipment[OrdId] that forces inserting and selecting only those order tuples from a relation over ORDER that are referenced by some shipment tuples. 

Suppose all the tuple integrity constraints that are embedded into relation schemes whose name has the same meaning are the same, except the domain constraint for attribute Origin. The values of this attribute determine whether a domestic or foreign customer issued a particular order. In the relation scheme Dom_Shipped_Order of the subschema $P_3$, dom(Origin) = $\{d\}$ holds, whereas in the relation scheme Order of $P_2$, $P_1$ and in the relation scheme ORDER of the database schema, dom(Origin) = $\{d, f\}$, where $d$ stands for domestic and $f$ for foreign. 

These subschemas and database schema will be referenced in all other examples of this paper. □

A database schema and a subschema contain the following concepts:

• Database schema attribute set $U = \bigcup_{R_i}^{R_j} R_{i,j}$ and subschema attribute set $U_s = \bigcup_{R_i}^{R_j} R_{i,j}$; 

• Sets of relation scheme attribute sets $S_1 = \{R_i \mid i \in \{1, \ldots, n\}\}$ and $S_2 = \{R_j \mid j \in \{1, \ldots, t\}\}$; and
a unique (hypothetical) subschema instance, called corresponding subschema instance, may be produced by applying appropriate relational join, project and select operations on a database schema instance; and
2. If an update of a hypothetical subschema instance executed by $T_k$ would be successful, then $T$ must be committed by DBMS.

If a subschema is intended for queries only, then it must satisfy only Condition 1. If a subschema is intended for updating, then it must satisfy both conditions. In this paper, the aforementioned conditions are called principles of a database update using subschema concepts. A subschema that satisfies these conditions is said to be consistent with the corresponding database schema.

Let $SAT(S, T)$ denote the set of all instances over a database schema $(S, T)$. If $I = \emptyset$, then $SAT(S, \emptyset)$ stands for $SAT(S, I)$. Let $SAT(R, C)$ denote the set of all instances over a relation scheme $N_j(R, C, K_p(R))$. If $C = \emptyset$, then $SAT(R)$ stands for $SAT(R, \emptyset)$. A set $s = \{ r(R_1), ..., r(R_t) \}$, where $r(R_t) \in SAT(R_t, C_t)$ and $t = |S_I|$ is an instance over the database schema $(S, T)$, i.e., $s \in SAT(S, T)$, if $I$ is satisfied.

Suppose a transaction program is based on subschema $P_j(S_k, I_k)$ concepts, where $S_k = \{ N_1^k, ..., N_n^k \}$, and let $s_k$ be an instance over the subschema $P_j$ that is a corresponding instance for some database instance $s$. The corresponding instance is a set of hypothetical relations $s_k = \{ r(R_1'), ..., r(R_n') \}$ that are produced by applying join, select and project operations to some relations from the database instance $s$. The composition of operators that is used to produce a corresponding instance $s_k$ from an instance $s$ will be denoted by $P_\Sigma \times \Pi \Omega$. Thus, $s_k = P_\Sigma \times \Pi \Omega(s, P_j)$.

From an end user point of view, using a transaction program based on the concepts of a subschema $P_j$ means updating a subschema instance $s_k$, regardless of the fact that this instance does not physically exist and hence is just a view on a database schema instance $s$. Let $H$ denote a sequence of operations that is initiated by a transaction program, and suppose these operations are defined using the concepts of a subschema $P_j$. This sequence of operations could be applied only to an instance $s_k$ over $P_j$. As a result, a set $s_k^{up} = \{ r_1^{up}, ..., r_n^{up} \}$ of updated relations would be produced, where $r_j^{up}$ is an abbreviated notation for $r(R_j')$. For the end user $s_k$ is a database and $s_k^{up}$ is the result of applying a sequence of update operations $H$ on $s_k$.

However, a subschema instance is not materialized. A DBMS executes query and update commands on an instance over a database schema. Update operations that are initiated by a transaction program based on subschema concepts are transformed into operations expressed in terms of database schema concepts. This way, the sequence of operations $H$ generates a sequence of operations $CH$ that are expressed in terms of database schema concepts. By applying the composition of operations in $CH$ on an instance $s$ over a database schema, a set of relations $s^{op} = \{ r'(R_1), ..., r'(R_n) \}$ is generated, where $s^{op} = CH(s)$. The operations from $CH$ should alter only data that would belong to the hypothetical subschema instance $s_k$. Thus, the altered data belong to $P_\Sigma \times \Pi \Omega(s^{op}, P_j)$. Since $CH$ is made of transformed update operations from $H$, it may be concluded that $s_k^{up} = P_\Sigma \times \Pi \Omega(s^{op}, P_j)$ (Fig. 1). From the subschema point of view, the effect of an instance $s$ update by means of $CH$ is the same as it would be the effect of updating the hypothetical corresponding instance $s_k$ using $H$. The same claim is also expressed by the second principle of a database update using subschema concepts.
4. A Formal Definition of Database Update Principles

In this section of the paper the principles of a database update using subschema concepts will be formally defined at the abstraction level of instances.

Definition 1. Suppose a set of relation scheme $S_k$ of a subschema $P_k$ and a database schema $(S, I)$ satisfy the following condition:

$$\forall s \in SAT(S, I) (\exists! s_k \in SAT(S_k)) (\forall r^k \in s_k)(r^k = \sigma_{F_k}(\pi_{R_k}(r(Sr_{s_k}, N^k))))$$  \hspace{1cm} (1)

where $F_k$ is a select condition, whose aim is to select those tuples from $r$ that satisfy constraints embedded in $N^k \in S_k$.

Let $s \in SAT(S, I)$ be an instance over a database schema. A set of relations $s_k \in SAT(S_k)$ satisfying (1), is a corresponding set to the instance $s$ with regard to the set $S_k$, which will be denoted by $s_k = \Sigma\Pi(s, S_k)$.

The select condition $F_k$ is introduced in the formula (1) since subschema constraints may be more restrictive then database schema constraints. It should be also noted that the set $s_k$ that satisfies (1), may not satisfy subschema interrelation constraints $I_k$.

Now, the first update principle may be formally expressed in the following way.

Definition 2. A subschema $P_k(S_k, I_k)$ and a database schema $(S, I)$ satisfy the first update principle if the condition (1) is satisfied and:

$$\forall s \in SAT(S, I) (\exists! s_k \in SAT(S_k)(\forall r^k \in s_k)(r^k = \pi_{R_k}(\sigma_{F_k}(r(Sr_{s_k}, N^k))))),$$  \hspace{1cm} (2)

holds, where $F_k$ is a select condition that selects those tuples from Cartesian product of relations satisfying (1) that also satisfy subschema interrelation constraints $I_k$.

Now, we define the notion of a corresponding instance of a subschema.

Definition 3. Suppose a subschema $P_k(S_k, I_k)$ and a database schema $(S, I)$ satisfy conditions (1) and (2), and let $s \in SAT(S, I)$. An instance $s_k \in SAT(S_k, I_k)$ is a corresponding instance of the instance $s$ with regard to subschema $P_k(S_k, I_k)$, denoted $s_k = \Pi\Sigma\times\Sigma\Pi(s, P_k)$, if the following holds:

$$\forall r^k \in s_k)(r^k = \pi_{R_k}(\sigma_{F_k}(\times_{k \in R(P_k)}(s, S_k)))).$$  \hspace{1cm} (3)
Consider the database schema and subschemas given in Example 1. To select only the domestic orders for the relation scheme \( \text{Dom}_1\text{Shipped}_1\text{Order} \) of the subschema \( P_3 \), the selection condition of formula (1) will be \( F_{\text{Dom}_1\text{Shipped}_1\text{Order}} : \text{Origin} = d \). The selection condition \( F_i \) of formula (2) should provide only already shipped domestic orders. So,

\[
F_i : \text{ORDER}_1\text{OrdId} = \text{SHIPMENT}_1\text{OrdId}.
\]

Alternatively, to select those domestic orders that are only partially shipped, the corresponding selection condition would be:

\[
F_i : \text{ORDER}_1\text{OrdId} = \text{SHIPMENT}_1\text{OrdId} \land \text{ORDER}_1\text{Total} > \text{SHIPMENT}_1\text{ShipTotal}.
\]

Let \( s = \{ r_1, r_2, r_3 \} \in \text{SAT}(S, I) \) be an instance over a database schema, where \( r_1(\text{ORDER}) \), \( r_2(\text{SHIPMENT}) \), and \( r_3(\text{CUSTOMER}) \) are shown on Fig. 2. For \( F_{\text{Dom}_1\text{Shipped}_1\text{Order}} : \text{Origin} = d \), \( F_i : \text{ORDER}_1\text{OrdId} = \text{SHIPMENT}_1\text{OrdId} \) and a subschema \( P_i \) instance \( s_i = \{ r_1', r_2', \ldots, r_i' \} \in \text{SAT}(S, I) \), where \( r_1'(\text{Dom}_1\text{Shipped}_1\text{Order}) \), and \( r_i'(\text{Shipment}) \) are shown on Fig. 3. \( s_i = \Pi_\Sigma \times \Sigma_\Pi (s, P_i) \) holds. □

### Example 2

<table>
<thead>
<tr>
<th>( r_1(\text{CUSTOMER}) )</th>
<th>( r_2(\text{SHIPMENT}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustId</td>
<td>CustName</td>
</tr>
<tr>
<td>1</td>
<td>Alfa</td>
</tr>
<tr>
<td>2</td>
<td>Beta</td>
</tr>
<tr>
<td>3</td>
<td>Gama</td>
</tr>
<tr>
<td>4</td>
<td>Delta</td>
</tr>
</tbody>
</table>

**Fig. 2.** An instance over the database schema from Example 1.

<table>
<thead>
<tr>
<th>( r_1'(\text{Dom}_1\text{Shipped}_1\text{Order}) )</th>
<th>( r_2'(\text{Shipment}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrdId</td>
<td>Ordate</td>
</tr>
<tr>
<td>103</td>
<td>14.04.02</td>
</tr>
<tr>
<td>104</td>
<td>14.04.02</td>
</tr>
</tbody>
</table>

**Fig. 3.** An instance over subschema \( P_i \) from Example 1.

Let a set of relations \( s_k = \{ r(R^k_1), \ldots, r(R^k_n) \} \) be given, and suppose it is a corresponding instance of a database instance \( s \) with regard to subschema \( P_k \). A transaction program having subschema \( P_k \) as a part of its specification, may perform only the operations covered by the set \( \{ \text{Role}(P_k, N^i_1), \ldots, \text{Role}(P_k, N^i_n) \} \), where each \( \text{Role}(P_k, N^i) \) defines allowed operations on instances over \( N^i \). Each update of an instance over a relation scheme \( N^i \) may be represented as a composition of operations from \( \text{Role}(P_k, N^i) \).

**Definition 4.** Let a subschema \( P_1(S_1, I_1) \) and an instance \( s_k = \{ r(R^k_1), \ldots, r(R^k_n) \} \in \text{SAT}(S_k, I_k) \) be given. Let \( O_i \in \{ t, \lambda, m, r \} \cap \text{Role}(P_k, N^i) \), where \( N^i \in S_k \) be a database operation, \( t_i \) a tuple over the set of attributes \( R^k_i \) and \( P_k \) the primary key value of a tuple on which operation \( O_i \) is going to be applied. It is supposed that the following conditions are satisfied.

- If \( O_i = \lambda \), then \( P_k \) represents the primary key value of a tuple from \( r_i'(R^k_i) \in S_k \) that is intended for deleting.
- If \( O_i = m \), then \( t_i \) contains values that are intended to replace the old values of a tuple from \( r_i'(R^k_i) \in S_k \) with the primary key value \( P_k \).
- If \( O_i = t \), then \( t_i \) is a tuple intended for insertion into relation \( r_i'(R^k_i) \in S_k \). In this case, \( P_k \) represents the primary key value of \( t_i \).

A function \( h_r = h_{(P_k, \{ r_i \})} : \text{SAT}(R^k_i) \rightarrow \text{SAT}(R^k_i) \), defined in the following way:
Example 3. Let Example 1 and Example 2 be considered. Function \( h_{106,106,11.02.02,4,f,1200} \) is an allowed operation on an instance over the relation scheme \( \text{Order} \) of the subschema \( P_2 \), for which \( \{s_1 \setminus \{r_1^k\}\} \cup \{h_{106,106,11.02.02,4,f,1200}(r_1^k)\} \in \text{SAT}(S_s, I_s) \) holds. If the same allowed operation \( h_{106,106,11.02.02,4,f,1200} \) is applied on \( r_1^k(\text{Dom}_\text{Shipped(Order)}, \text{on Fig. 3}, \text{then} \{s_1 \setminus \{r_1^k\}\} \cup \{h_{106,106,11.02.02,4,f,1200}(r_1^k)\} \in \text{SAT}(S_s, I_s) \) does not hold, since \( t_i = (106, 11.02.02, 4,f,1200) \) represents a foreign order.

The operation \( h_{105,105,11.02.02,4,f,1200} \) is not allowed over the relation scheme \( \text{Order} \) of the subschema \( P_2 \), because \( m \not\in \text{Role}(P_{2}, N)^1 \). □

For the sake of simplicity and without any impact on further results, it is supposed in formula (4) that selection of database tuples to be deleted or modified is performed exclusively by means of a primary key value.

Definition 5. An update function \( h^k \) of a relation over a set of attributes \( R^k \) is produced by composing a sequence of allowed update operations \( h_m, ..., h_1 \):

\[
h^k = h_m \circ ... \circ h_1 = (h_m, ..., h_1),
\]

The operation \( \circ \) is applied according to the common mathematical definition for function composition. The set of all possible update functions that may be applied on an instance over \( R^k \) will be denoted by \( h^k \).

Each hypothetical instance \( s_k \) over a subschema \( P_k \) is updated by updating instances over its relation schemes.

Definition 6. The update function of a set of relations \( s_k = \{r(R_k^1), ..., r(R_k^n)\}, \) where \( S_k = \{N_k^1, ..., N_k^n\} \), denoted by \( H \), is any mapping \( H(s_k) = (h_k, ..., h_k)(s_k) = \{r'(R_k^1), ..., r'(R_k^n)\} \) such that:

\[
\forall i \in \{1,..., n\}, r'(R_k^i) = h_k(r(R_k^i)) \wedge h_k \in h^k
\]

is satisfied. The set of all possible update functions over \( \{R_k^1, ..., R_k^n\} \) will be denoted by \( H_k \). □

Definition 7. Let \( s_k = \{r(R_k^1), ..., r(R_k^n)\} \) be a set of relations, where \( S_k = \{N_k^1, ..., N_k^n\} \). The set of relations \( s_k^{up} = \{r'(R_k^1), ..., r'(R_k^n)\} \) is an updated instance of \( s_k \) if the following holds:

\[
(\exists H \in H_k)(s_k^{up} = H(s_k)).
\]

Since Definition 7 is strongly related to Definition 4, it follows that \( s_k^{up} = H(s_k) \) need not necessarily be an instance over \( P(S_s, I_s) \), even in the cases when \( s_k \in \text{SAT}(S_s, I_s) \) holds.

The next lemma introduce a consequence of condition (1) that is important for further considerations. The lemma is given without proof.

Lemma 1. Suppose a subschema \( P_0(S_0, I_0) \) and a database schema \( (S, I) \) satisfy condition (1). Then the following is satisfied:

\[
(\forall N^j \in S_0)(N_j) = S_r(P_0, N^j) \Rightarrow (R^k \subseteq R_j \wedge (\check{\cup} N^j \cup \check{\cup} \text{Un}(N^j)) \subseteq (\check{\cup} N_j \cup \check{\cup} \text{Un}(N_j)),
\]

where \( \check{\cup} \text{Un}(N) = \{X \mid \text{Unique}(N, X) \in \text{Uniq}(N)\} \). □

Since an instance \( s_k \) does not physically exist, each update operation \( h_k(r_1^k) \) and update function \( h^k \) must be transformed into an update operation and an update function on an instance \( r_1 \) over a relation scheme \( N_j \) of a database schema \( (S, I) \) that is corresponding to \( N^j \).
**Definition 8.** Suppose a subschema $P_1(S_1, T_1)$ and a database schema $(S, T)$ satisfy conditions (1) and (2). Let $h_i = h_{P_k, r_i, o_i}$ and $N_j = S\tau(P_k, N^k_i)$ be given. A **corresponding** operation to $h_i$ that should be executed on an instance $r_j$ over $N_j$ is a function $h_i = h_{P_k, r_i, o_i} : SAT(R_j, C_j) \rightarrow SAT(R_i)$ defined in the following way:

$$
Ch_i(r_i) = \begin{cases} 
 r_j \setminus \{t_1\}, & \text{if } O_r = \land t_1 \in r_j \land t_1[K_p(R^k_i)] = Pk_r, \\
(r_j \setminus \{t_1\}) \cup \{(t_1(R_i \setminus Mw(P_k, N^k_i)) \times \{t_1[R^k_i]\}), & \text{if } O_r = m \land t_1 \in r_j \land t_1[K_p(R^k_i)] = Pk_r, \\
(r_j \setminus \{t_1\}) \cup \{(t_1(R_i \setminus Mw(P_k, N^k_i)) \times \{t_1[R^k_i]\}), & \text{if } O_r = r \lor \\
r_j, & \text{if } O_r = r. 
\end{cases}
$$

(8)

where $P_k$ is a primary key value of a tuple from database relation $r_j \in s, t_i$ is a tuple over the set of attributes $R^k_i$, and $O_r \in \{i, \land, m, r\}$ is an operation that is performed on a tuple from $r_j \in s$. The term $\text{Qef}^\Delta T_{up}(N, W)$ is defined in the following way:

$$
\text{Qef}^\Delta T_{up}(N, W) = \{W \mid (\forall A \in W)(t[A] = \text{Def}(N, A))\},
$$

where $\text{Def}(N, A)$ is an attribute $A$ default value in the relation scheme $N$, if it is declared, otherwise it is a null value. □

**Definition 9.** The update function $Ch_i = (Ch_m, ..., Ch_1)$ that corresponds to the update function $h_i = (h_m, ..., h_1)$, is a composition of corresponding operations $Ch_m, ..., Ch_1$, where for each $j \in \{1, ..., m\}$, $Ch_j$ corresponds to an allowed operation $h_j$:

$$
Ch_i = Ch_m \circ ... \circ Ch_1 = (Ch_m, ..., Ch_1). \quad \square
$$

The set of all possible corresponding update functions induced by a relation scheme $N^k_i$ will be denoted by $\mathcal{C}_h_i$.

**Definition 10.** An update function $CH$ of a database schema instance $s$ that corresponds to the update function $H = (h_1^k, ..., h_s^k)$ of a subschema instance $s_k$, is a mapping $CH(s) = (f_1, ..., f_s)(s) = \{r_1'(R_1), ..., r_s'(R_s)\}$ such that:

$$
(\forall j \in \{1, ..., t\}) (r_j'(R_j) = f_j(t(R_j))
$$

holds, where:

$$
f_j = \begin{cases} 
 Ch_j^k, & \text{if } (\exists N^k_j \in S_k)(N_j = S\tau(P_k, N^k_j)) \\
\text{ident}, & \text{if } (\forall N^k_j \in S_k)(N_j \neq S\tau(P_k, N^k_j)). 
\end{cases}
$$

(10)

Each function $Ch_i \in \mathcal{C}_h_i$ corresponds to the update function $h_i^k$ from $H$, whereas ident is an identity function, i.e. $(\forall r_j \in s)(ident(r_j) = r_j)$. We will denote by $\mathcal{C}_h$ the set of all possible update functions of an instance over a database schema that are induced by the concepts of a subschema $P_i$. □

Since Definition 10 is strongly related to Definition 8, it follows that the updated instance $CH(s)$ need not necessarily be an instance over $(S, T)$, despite the fact that $s \in SAT(S, T)$ holds.

Let us introduce a term of successful update function. Then, a formal definition of the second update principle follows.
Definition 11. Let a subschema $P_d(S_k, I_k)$ and a database schema $(S, T)$ be given. Let $s$ be an instance over $(S, T)$ and $H \in H_k$ be an update function which is applied on subschema instance $s_k = \Pi_{x \in T}(s, P_k)$. CH denotes an update function of $s$, corresponding to $H$. Let a projection of database relations according to subschema $\Pi(s, P_k)$ be defined in the following way:

$$\Pi(s, P_k) = \{ r^{s_k}_j(R_j) \mid N_j \in S_k \land N_j = \delta r(P_k, N_j) \land r_j(R_j) \in s \land r_j^{s_k} = \pi_{R_j}(r_j) \}.$$  \hfill (11)

The update function $H(s_k)$ is successful, if the predicate $\text{Successful}(\Pi(s, P_k), s, H)$, defined in the following way:

$$H(s_k) \in \text{SAT}(S_k, I_k) \land H(\Pi(s, P_k))) = \Omega^r \cap \Omega^r|_{s_k}$$  \hfill (12)

holds, where $|$ denotes that a set of relations $H(\Pi(s, P_k))$ satisfy the set of constraints $\Omega^r \cap \Omega^r|_{s_k}$.

Definition 12. A subschema $P_d(S_k, I_k)$ and a database schema $(S, T)$ satisfy the second update principle if Conditions (1) and (2) are satisfied and the following holds:

$$(\forall s \in \text{SAT}(S, T))(\forall H \in H_k)(\text{Successful}(\Pi(s, P_k), s, H) \Rightarrow CH(s) \in \text{SAT}(S, T)), \hfill (13)$$

where $CH$ is an update function of $s$, corresponding to $H$. □

Conditions (1), (2) and (13) constitute a formal definition of the principles of a database update using subschema concepts. It follows from Definition 12 that the first update principle is a prerequisite for the second one. It is due to the fact that Conditions (1) and (2) are necessary to provide the existence of an instance $\Pi_{x \in T}(s, P_k)$, for each $s \in \text{SAT}(S, T)$.

By means of (1), (2) and (13) we express the fact that database updates, initiated by a transaction program based on the concepts of subschema, are safe.

Example 4. Let Example 1 and Example 2 be considered. Let us consider an update function $H(s_k) = (h_1^{s_k}, h_2^{s_k}(s_k))$, where $s_k$ is shown on Fig. 3. Let $h_1^{s_k} = (h_{106,106,14.06.02,6,4,1000,0})(r_1^{s_k})$ and $h_2^{s_k} = (h_{105,404,14.06.02,1000,0})(r_2^{s_k})$ be given operations. It is obvious that both of them are allowed operations over $s_k$, such that $H(s_k) \in \text{SAT}(S_k, I_k)$ holds. For $s$ shown on Fig. 2, a corresponding database update function is $CH(s) = (f_1, f_2, f_3)(s)$, where $f_1(r_1) = Ch_1^{s_k}(r_1) = (h_{106,106,14.06.02,6,4,1000,0})(r_1), f_2(r_2) = Ch_2^{s_k}(r_2) = (h_{404,404,14.06.02,1000,0})(r_2)$ and $f_3(r_3) = \text{idem}(r_3) = r_3$.

In this example, since $(106, 14.06.02, 6, d, 1000)$ represents a new order with a CustId = 6, which does not exist in database, the operation $Ch_1^{s_k}(r_1)$ will violate referential integrity $\text{ORDER}[\text{CustId}] \subseteq \text{CUSTOMER}[\text{CustId}]$. Therefore, $CH(s)$ will not be an instance over $(S, T)$. At the other hand, $\text{Successful}(\Pi(s, P_k), s, H)$ holds. This way, subschema $P_k$ and database schema $(S, T)$ violate the second update principle. Informally, it is a consequence of the fact that $i \in \text{Role}(P_k, \text{Dom}_\text{Shipped}_\text{Order})$, but $\text{ORDER}[\text{CustId}] \subseteq \text{CUSTOMER}[\text{CustId}]$ is not included subschema $P_k$.

If we considered in this example an operation $h_1^{s_k} = (h_{105,105,14.06.02,6,4,1000,0})(r_1^{s_k})$ instead of $h_1^{s_k} = (h_{106,106,14.06.02,6,4,1000,0})(r_1^{s_k})$, then the corresponding operation $Ch_1^{s_k}(r_1)$ would violate a key $\{\text{OrdId}\}$ of scheme ORDER, and $CH(s)$ would not be an instance over $(S, T)$, too. In this case, however, $\text{Successful}(\Pi(s, P_k), s, H)$ would not hold. □

5. The Formal Database Schema and Subschema Consistency

Definition 13. Let $\mathcal{O}$ be the set of database schema $(S, T)$ constraints. A constraint $\sigma \in \mathcal{O}^r$ is relevant for a subschema $P_k$, if it may be violated by some update function $CH \in H_k$. □

Example 5. Let us consider subschema $P_2$ in Example 1. The referential integrity $\text{ORDER}[\text{CustId}] \subseteq \text{CUSTOMER}[\text{CustId}]$ is a relevant constraint for $P_2$, since $i \in \text{Role}(P_2, \text{Order})$.

At the other hand, the inclusion dependency $\text{CUSTOMER}[\text{CustId}] \subseteq \text{ORDER}[\text{CustId}]$ is not a relevant constraint for $P_2$, since $\forall \sigma \in \text{Role}(P_2, \text{Customer}) \land \{i, m\} \cap \text{Role}(P_2, \text{Customer}) = \emptyset$. □

It is intuitively appealing that a subschema and a database schema will be formally consistent, if their concepts are in some way consistent. The concepts of a subschema and a database schema are formally consistent, if:
The set of attributes of each subschema relation scheme is a subset of the corresponding relation scheme attribute set;
2. The set of keys of each subschema relation scheme is a subset of the union of the corresponding relation scheme set of keys and the set of attribute sets with a unique property; and
3. All the constraints that can be inferred from the database schema and that are relevant for the subschema are embedded in the subschema.

Formally, the first and the second condition are expressed by formula (7). The third consistency condition can be expressed by the following logical implication:

\[
O_k \models O_{P_k}'
\]

where \(O_{P_k}'\) is the set of all database schema constraints that are relevant for subschema \(P_k\).

The most important components of a constraint \(\sigma \in \Omega\) specification are:
- A set of triples \(T(\sigma) = \{(N_j, \rho_j, At_j), ..., (N_{in}, \rho_{in}, At_{in})\}\); and
- A set of critical operations.

In a triple \((N_j, \rho_j, At_j)\), \(N_j\) is the name of a relation scheme that is spanned by \(\sigma\), \(\rho_j\) is the role of \(N_j\) in \(\sigma\) and \(At_j\) is a set or sequence of attributes from \(R_i\) that are relevant for \(\sigma\). An attribute \(A\) is relevant for \(\sigma\) if \(\sigma\) is used to check values of \(A\).

A critical operation is an operation that can violate a constraint. The constraint \(\sigma\) should belong to the set of subschema relevant constraints if the operation that might violate \(\sigma\) is allowed in the subschema \(P_k\).

There are two kinds of relevant constraints:
- The inclusive relevant constraints, denoted by \(\text{Ini}(\Omega, P_k)\); and
- The extensible relevant constraints, denoted by \(\text{Exi}(\Omega, P_k)\).

Suppose a constraint \(\sigma \in \Omega'\) is relevant for a subschema \(P_k\). The constraint \(\sigma\) belongs to \(\text{Ini}(\Omega, P_k)\) if and only if:

\[
(\forall (N_j, \rho_j, At_j) \in T(\sigma))(\exists N_j^t \in S_k)(\exists r(\sigma, N_j^t) = N_j \land At_j \subseteq R_i^t)
\]

holds.

A constraint \(\sigma\) belongs to \(\text{Exi}(\Omega, P_k)\) if and only if it is a relevant for \(P_k\), and \(\sigma \notin \text{Ini}(\Omega, P_k)\) holds.

**Example 6.** In Example 1, the referential integrity \(\text{ORDER}[\text{CustId}] \subseteq \text{CUSTOMER}[\text{CustId}]\) is an inclusive relevant constraint for subschema \(P_2\), since \(i \in \text{\text{Role}}(P_2, \text{Order})\), \(\text{ORDER} = \text{\text{sr}}(P_2, \text{Order})\), \(\text{CUSTOMER} = \text{\text{sr}}(P_2, \text{Customer})\) \(\text{CustId} \in R_{2j}^i\) and \(\text{CustId} \in R_{2j}^i\) holds.

Since \(\text{\text{Role}}(P_2, \text{Dom\_Shipped\_Order}) = \{i, r\}\) holds in Example 1, the referential integrity \(\text{ORDER}[\text{CustId}] \subseteq \text{CUSTOMER}[\text{CustId}]\) is an extending relevant constraint for subschema \(P_3\). It is relevant because \(\text{Dom\_Shipped\_Order} \in S_1\), \(\text{ORDER} = \text{\text{sr}}(P_3, \text{Dom\_Shipped\_Order})\) and \(i \in \text{\text{Role}}(P_3, \text{Dom\_Shipped\_Order})\) holds. It is an extending constraint, because for the referenced relation scheme \(\text{CUSTOMER}\) there is no corresponding relation scheme in \(P_1\). This implies that a relation scheme \(\text{CUSTOMER}\) should be included in \(P_1\). □

**Definition 14.** A subschema \(P_h(S_h, I_h)\) concepts are formally consistent with a database schema \((S, I)\) concepts if (7) holds and the following two conditions are satisfied:

\[
O_k \models \text{Ini}(\Omega, P_h),
\]

\[
\text{Exi}(\Omega, P_h) = \emptyset.
\]
Example 7. Suppose \( \text{Role}(P_2, \text{Customer}) = \{i, r\} \) in Example 1 holds. The constraints \( \text{ORDER}[\text{CustId}] \subseteq \text{CUSTOMER}[\text{CustId}] \) and \( \text{CUSTOMER}[\text{CustId}] \subseteq \text{ORDER}[\text{CustId}] \) are inclusive relevant constraints for subschema \( P_2 \). In this example, however, the condition \( O_2 \models \text{Ini}(O, P_2) \) is not satisfied, since \( \text{Customer}[\text{CustId}] \subseteq \text{Order}[\text{CustId}] \) is not included in the set of subschema constraints \( I_2 \). An insertion of a new customer not having any order would not violate the consistency of a hypothetical instance over \( P_2 \), whereas DBMS would reject this operation, since it would violate the consistency of a database schema instance. Besides, the predicate \( \text{Successful}(S, I), (S_2, I_2), s, H) \), for such an instance \( s \) and update function \( H \) would hold.

Since \( \text{Role}(P_1, \text{Done}_\text{Shipped}_\text{Order}) = \{i, r\} \) in Example 1 holds, the referential integrity \( \text{ORDER}[\text{CustId}] \subseteq \text{CUSTOMER}[\text{CustId}] \) is an extending relevant constraint for subschema \( P_1 \), which means that \( \text{Ex}(O, P_1) \neq \emptyset \) holds. An insertion of a new order with a non-existing \( \text{CustId} \) value would not violate the consistency of an instance over \( P_1 \), whereas DBMS would reject this operation, since it would violate the consistency of a database schema instance. In this case, the predicate \( \text{Successful} \) would be satisfied too.

6. Conclusion

In this paper, the notion of a subschema that represents the data definition part of a transaction program specification is introduced. A subschema is a structure, expressed by means of relational database schema concepts, extended by the specification of the allowed database update activities, and a mapping that uniquely bonds subschema relation schemes with the database relation schemes. The relationship between a database schema and a subschema is established by defining conditions of database update principles and the formal subschema and database schema consistency. It is claimed in the paper that the formal consistency implies database update principles. Accordingly, if a subschema is aimed to constitute the data definition component of a transaction program specification, its design process should adhere to formal consistency conditions. One of the main consequences of that statement is that the set of subschema constraints should imply all those database schema constraints that might be violated by the allowed database update operations.

A general solution of the implicational problem in the presence of different constraint types is very hard to find, if even possible. Testing the satisfaction of the formal consistency may be relaxed by considering implicational problem for various constraint types separately. We believe that our approach relaxed in that way, may lead to a good transaction program design practice.

We have studied subschemas where each relation scheme instance has been produced by applying relational project and select operations onto a base relation. That approach enables an easy transformation of database update operations expressed upon subschema concepts into database update operations expressed upon database schema concepts. However, sometimes it may appear preferable to allow defining instances of a subschema relation scheme by applying select and project operations onto joins of base relations. Therefore, a future work should study the effects of constructing a subschema relation scheme from more than one database schema relation scheme.

References