NOVI SAD J. MATH.
VOL. 30, NO. 3, 2000, 111-122

EXTENDED REFERENTIAL INTEGRITY

Pavle Mogin, Ivan Luković, Miro Govedarica
Faculty of Technical Sciences, University of Novi Sad
trg D. Obradovica 7, 21000 Novi Sad, Yugoslavia
e-mail: {mogin, ivan,miro} @iis.ns.ac.yu

Abstract

Referential integrity, or foreign key constraint, is one of the basic relational data model constraints. It's objective is to maintain data of two base relations in consistent state during tuple insertion, deletion, and modification. In the course of third normal form database schema design, there arise situations where it is not possible to define a referential integrity between two relation schemes, although the real system business rules dictate the definition of an appropriate database constraint. The problem arises because of the fact that the referencing relation scheme contains only the proper subset of the referenced relation scheme primary key. To overcome the described problem, the notion of extended referential integrity was introduced in the paper. The *extended referential integrity* is a constraint that spans more than two relation schemes. Accordingly, checking the satisfaction of that constraint involves the relations over all the included relation schemes. All the properties of the extended referential integrity, as described in the paper, correspond to the ANSI SQL/2 standard.

AMS Mathematics Subject Classification (1991): 68P15
Key words and phrases: Third normal form database schema, inclusion dependency, referential integrity, extended referential integrity, SQL standards.

1. Introduction

Referential integrity is a special case of inclusion dependency. Unirelational inclusion dependencies were introduced by Casanova et al. in [Ca84]. A slight extension of their definition redefines inclusion dependencies as the interrelation (database) constraints. Codd was one of the first authors who introduced the referential integrity as an interrelation constraint [C79]. The referential integrity actions: restricted, cascade, set null, and the properties of referential integrity in the presence of null values, are, among others, considered in [D81], [D92], and [H92]. Implementation features of the constraint are covered through SQL standards [D94].

In all these papers, referential integrity is treated as a constraint between at most two base relations. In the course of third normal form database schema design, there arise situations where it is not possible to define a referential integrity between two relation schemes, although the real system business rules dictate the definition of an appropriate database constraint. The problem arises because of the fact that the referencing relation scheme contains only the proper subset of the referenced relation scheme primary key. To overcome the described problem, the notion of extended referential integrity is introduced here. The *extended referential integrity* is a constraint that spans more than two relation schemes. Accordingly, checking satisfaction of that constraint involves the relations over all the included relation schemes. All the properties of the extended referential integrity correspond to ANSI SQL/2 standard. In that sense, the extended referential integrity type can be either full, partial, or one that is neither of them. Also, the "on delete" action can be one of the following: "no action", "cascade", "set null" and "set default".

Apart from this Introduction, the paper contains two other parts and Conclusion. In the part two, some necessary concepts are given, under assumption that the reader is familiar with basic concepts of the relational data model and corresponding notation. The definition and properties of the extended referential integrity are given in part three.

2. Preliminaries

Let $S = \{N_i(R_i, K_i) |\ i = 1, ..., n\}$ be a set of the third normal form relation schemes. Further, let $N_i(R_i, K_i)$ and $N_j(R_j, K_j)$ be two relation

schemes in S and $X = \{A_1, ..., A_n\}$ and $Y = \{B_1, ..., B_n\}$ such sets of attributes that: $X \subseteq R_i$, $Y \in K_j$ and $(\forall k \in \{1, ..., n\})$ $(dom(A_k) \subseteq dom(B_k))$. The attribute sets X and Y are *domain compatible*, denoted as $dom(X) \subseteq dom(Y)$, where $dom(X) \subseteq dom(A_1) \times \ldots \times dom(A_n)$ and $dom(Y) \subseteq dom(B_1) \times \ldots \times dom(B_n)$.

Supposing that the attributes X and Y have no null values in the instances $r(N_i)$ and $r(N_j)$ of the relation schemes N_i and N_j, the referential integrity is

$$i : N_i[X] \subseteq N_j[Y],$$

where i is the optional referential integrity name, N_i and N_j the names of the relations schemes $N_i(R_i, K_i)$, $N_j(R_j, K_j) \in S$, and the attribute sets X and Y are considered as attribute arrays $(A_1, ..., A_n)$ and $(B_1, ..., B_n)$, respectively. The referential integrity $i : N_i[X] \subseteq N_j[Y]$ is satisfied if

$$(1) \qquad (\forall u \in r(N_i))(\exists v \in r(N_j))(u[X] = v[Y]).$$

According to (1), the tuples of the relation $r(N_i)$ over the relation scheme (R_i, K_i) existentially depend on the tuples of the relation $r(N_j)$ over the relation scheme (R_j, K_j). For $Z \in \{X, Y\}$, $N[Z]$ is the abbreviated notation for the restriction $r(N)[Z]$ or the projection $\pi_Z(r(N))$ of the instance $r(N)$ onto the attribute set Z. If $X = Y = K_p(R_j)$, where $K_p(R_j)$ is the primary key of the relation scheme $N_j(R_j, K_j)$, then the referential integrity is called *foreign key constraint*, too.

2.1. Referential integrity and null values

As the attribute set $Y = \{B_1, \ldots, B_n\}$ represents the key of the relation scheme N_j, attributes B from Y can not have null values in the relation $r(N_j)$. On the other hand, if it is not otherwise declared for the relation scheme N_i, the attributes A from $X = \{A_1, \ldots, A_n\}$ set can have null values in the relation $r(N_i)$. The relational data language SQL standards [D94], pay considerable attention to the referential integrity and its database consistency checking mechanisms.

2.2. Referential integrity and SQL

SQL offers the possibilities of declaring one of three types of referential integrity for data consistency control of the instances of the relation schemes

N_i and N_j. The first one could be named as implicit, second as partial, and the third one as full.

Implicit referential integrity is not explicitly declared. It is a referential integrity that is neither partial nor full. It is fulfilled if for each tuple u from $r(N_i)$:

- each attribute of the X has a non-null value and there is at least one tuple v inside the relation $r(N_j)$ with the same Y value, or
- tuple u has null value for at least one attribute in X.

Partial referential integrity is fulfilled if for each tuple u from $r(N_i)$ exists at least one tuple v in relation $r(N_j)$, for which every non-null value within X equals the value of the corresponding domain compatible attribute of Y. Partial referential integrity is declared explicitly within relation scheme description.

Full referential integrity is fulfilled if for every tuple u from $r(N_i)$:

- every attribute of X has a non-null value and there is one tuple v in the relation $r(N_j)$ with the same Y value or
- every attribute of the set X has a null value.

Full referential integrity is declared explicitly in the relation scheme description.

The standards also contain the procedures for handling the deletion and the key value modification of the tuple v of the relation $r(N_j)$. These procedures are called *actions*. For each referential integrity, one of the following four deletion actions is defined: restrict, cascade deletion, set null, and set default.

2.3. The closure graph

One of the frequent remarks on the relational data model is its lack of a schematic technique for the database schema structure presentation. This can be improved by introducing the closure graph.

Definition 1. Closure graph *is the basic graph* $\mathcal{G} = (S, \varphi)$, *where* $S = \{N_i(R_i, K_i) | \ i = 1, \ldots, n\}$ *and the relation* φ *is defined in* S^2 *as:*

$$\varphi = \{((R_i, K_i), (R_j, K_j)) \in S^2 | R_j \subset (R_i)^+_{\mathcal{F}} \wedge (\forall (R_k, K_k) \in (S \setminus \{(R_i, K_i), (R_j, K_j)\}))(R_j \not\subset (R_k)^+_{\mathcal{F}} \vee R_k \not\subset (R_i)^+_{\mathcal{F}})\},$$

where $\mathcal{F} = \{X \to A | (\exists (R_i, K_i) \in S)(A \in (R_i \setminus X) \wedge X \in K_i)\}$.

The relation φ establishes the natural order of the set S, where the relation scheme (R_j, K_j) follows the relation scheme (R_i, K_i) if $R_i \to R_j \in \mathcal{F}^+$. The closure graph concept makes sense only if the relation scheme set S is at least in third normal form. It is a suitable instrument for visual presentation of that natural structure on the relation scheme set. When considering the foreign key constraint, the relation scheme N_i directly precedes the relation scheme N_j, so that the referential integrity of this class is easily identified by using the closure graph (S, φ).

3. Extended foreign key constraint

There are certain relation scheme sets for which the referential integrity can not be directly defined for all pairs of $(N_i, N_j) \in \varphi$. Those are the pairs wich satisfy the conditions: $X = Y = K_p(R_j)$ and $K_p(R_j) \not\subseteq R_i$, as illustrated through the following example.

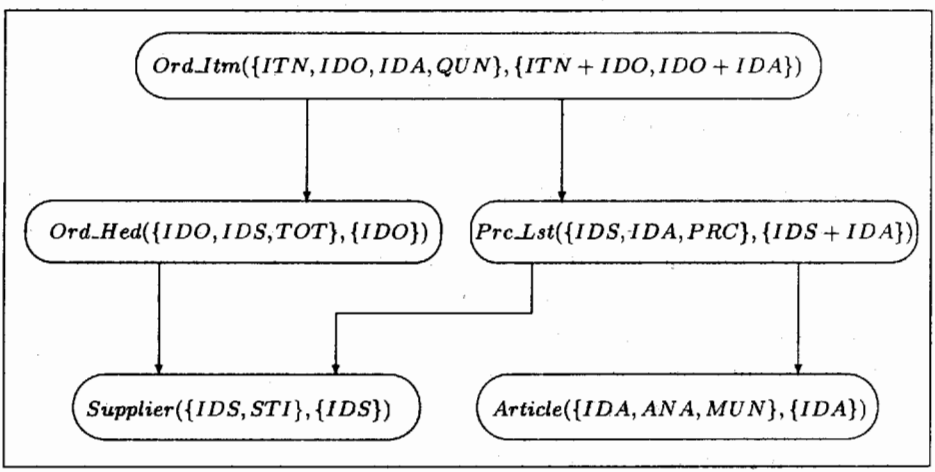

Figure 1.

Example 1. Figure 1 shows the closure graph of the relation scheme set presenting a model of orders, suppliers, articles, and suppliers price lists. The mnemonics of the attributes have the following meanings: *ITN* - item number, *IDO* - id number of the order, *TOT* - total price of the ordered

articles, IDA - id number of the articles, QUN - quantity of the articles within the item, IDS - suppliers id number, STI - suppliers title, PRC - item price, ANA - article name and MUN - measurement unit.

For the pair of the relation schemes $(Ord\_Itm, Ord\_Hed) \in \varphi$ it is $X = Y = (IDO)$. The condition $X \subseteq R_i$ is fulfilled, so between the relation schemes Ord_Itm and Ord_Hed the basic referential integrity can be defined.

For the pair of the relation schemes $(Ord\_Itm, Prc\_Lst) \in \varphi$ it is $X = Y = (IDS, IDA)$ and $X \cap R_i = (IDA)$. The condition $X \subseteq R_i$ is not fulfilled because $\{IDS, IDA\} \not\subseteq \{ITN, IDO, IDA, QUN\} = R_{Ord\_Itm}$. However, between the relation schemes Ord_Itm and Prc_Lst a referential integrity should be defined in order to secure that orders contain only the articles of the supplier to whom the order is being sent. □

The following lemma is given without proof.

Lemma 1. *Let $N_i(R_i, K_i)$ and $N_j(R_j, K_j)$ be such relation schemes in S, that $(N_i, N_j) \in \varphi$ and $K_p(R_j) \not\subseteq R_i$. Than $K_p(Rj) \cap R_i \neq \emptyset$ and*

$$(\exists N_k(R_k, K_k) \in S)\,(R_k \subset (R_i)^+_{\mathcal{F}} \wedge R_k \not\subseteq (R_j)^+_{\mathcal{F}} \wedge ((K_p(R_j) \setminus R_i) \cap (R_k \setminus K_p(R_k)) \neq \emptyset)),$$

where $\mathcal{F} = \{X \to A |\ (\exists (R_l, K_l) \in S)(X \in K_l \wedge A \in R_l \setminus X)\}$.

When the basic referential integrity can not be defined, the definition of multiple inclusion dependencies with partial arrays of $K_p(R_j) = (B_1, ..., B_n)$, used as the arguments, would not be a good solution. The solution is not good because those inclusion dependencies are not based on the whole value of the relation scheme (R_j, K_j) primary key, so they do not establish existential dependency between the tuples $r(N_i)$ and tuples of $r(N_j)$. That is why the notion of the extended foreign key constraint is introduced.

Let S be a relation scheme set and $(N_{i_1}, N_j) \in \varphi$. Let $\{N_{i_1}, ..., N_{i_n}\}$ be a non-empty subset of S. The sets S_r and S_p are introduced, with the aim of defining the set $\{N_{i_1}, ..., N_{i_n}\}$. The set S_r satisfies the following condition

$$K_p(R_j) \subseteq (R_{i_1} \cup \bigcup_{N_k \in S_r} R_k) \wedge (\forall N_l \in S_r)(N_l \in S \wedge R_l \subseteq (R_{i_1})^+_{\mathcal{F}} \wedge R_l \not\subseteq (R_j)^+_{\mathcal{F}} \wedge K_p(R_j) \cap (R_l \setminus K_p(R_l)) \neq \emptyset \wedge K_p(R_j) \not\subseteq (R_{i_1} \cup \bigcup_{N_k \in S_r \setminus \{N_l\}} R_k)).$$

The set S_p satisfies the following condition

$$(\forall N_k \in S_r)(R_k \subseteq (R_{i_1})^+_{\mathcal{G}}) \wedge (\forall N_l \in S_p)(N_l \in S \wedge (\exists N_k \in S_r)(R_k \not\subseteq (R_{i_1})^+_{\mathcal{G}\setminus\mathcal{G}_l})),$$

where $\mathcal{G} = \{X \to A | (\exists (R_i, K_i) \in S_p)(X \in K_i \wedge A \in (R_i \setminus X))\}$, and $\mathcal{G}_l = \{X \to A | X \in K_l \wedge A \in (R_l \setminus X)\}$.

Finally, the relation scheme set $\{N_{i_1}, ..., N_{i_n}\}$ is defined as

$$\{N_{i_1}, ..., N_{i_n}\} = \{N_{i_1}\} \cup S_r \cup S_p. \tag{2}$$

Let $\mathcal{P}(\{N_{i_1}, ..., N_{i_n}\}, N_j)$ be a predicate meaning "$\{N_{i_1}, ..., N_{i_n}\}$ and the relation scheme N_j fulfils the condition (2)".

Definition 2. *Let (S, φ) be the closure graph, $(N_{i_1}, N_j) \in \varphi$, $K_p(R_j) \not\subseteq R_{i_1}$ and let the predicate $\mathcal{P}(\{N_{i_1}, ..., N_{i_n}\}, N_j)$ gets true. The* **extended foreign key constraint**, *denoted as:*

$$\triangleright\triangleleft \{N_{i_1}, ..., N_{i_n}\}[K_p(R_j)] \subseteq N_j[K_p(R_j)],$$

is a constraint that is satisfied as:

- *an* **implicit** *referential integrity if:*

$$\begin{aligned}(\forall u \in r(N_{i_1}))(\exists v \in r(N_j))((u[K_p(R_j) \cap R_{i_1}] = v[K_p(R_j) \cap R_{i_1}]) \wedge \\ (\exists t \in \overset{n}{\underset{k=1}{\triangleright\triangleleft}} r(N_{i_k}))(u = t[R_{i_1}] \Rightarrow t[K_p(R_j)] = v[K_p(R_j)])) \vee \\ (\exists A \in (K_p(R_j) \cap R_{i_1}))(u[A] = \omega) \vee \\ ((\exists t \overset{n}{\underset{k=1}{\triangleright\triangleleft}} r(N_{i_k}))(u = t[R_{i_1}]) \Rightarrow (\exists A \in K_p(R_j))(t[A] = \omega)).\end{aligned} \tag{3}$$

- *a* **partial** *referential integrity if:*

$$\begin{aligned}(\forall u \in r(N_{i_1}))(\exists v \in r(N_j))((\forall A \in R_i \cap K_p(R_j))(u[A] = \omega \vee \\ u[A] = v[A]) \wedge (\exists t \in \overset{n}{\underset{k=1}{\triangleright\triangleleft}} r(N_{i_k}))(u = t[R_{i_1}] \Rightarrow \\ (\forall A \in K_p(R_j))(t[A] = \omega \vee t[A] = v[A]))),\end{aligned} \tag{4}$$

- *a* **full** *referential integrity if:*

$$
\begin{aligned}
&(\forall u \in r(N_{i_1}))((\exists v \in r(N_j))(u[K_p(R_j) \cap R_{i_1} = v[K_p(R_j) \cap R_{i_1}] \wedge \\
&\qquad (\exists t \underset{k=1}{\overset{n}{\triangleright\!\triangleleft}} r(N_{i_k}))(u = t[R_{i_1}] \Rightarrow t[K_p(R_j)] = v[K_p(R_j)]))) \vee \\
&\qquad\qquad ((\forall A \in (K_p(R_j) \cap R_{i_1})(u[A] = \omega) \wedge \\
&((\exists t \in \underset{k=1}{\overset{n}{\triangleright\!\triangleleft}} r(N_{i_k}))(u = t[R_{i_1}]) \Rightarrow (\forall A \in K_p(R_j))(t[A] = \omega))).
\end{aligned} \tag{5}
$$

The extended foreign key constraint is a referential integrity between the relation scheme $(\bigcup_{k=1}^{n} R_{i_k}, K_i)$ and the relation scheme (R_j, K_j) and an inclusion dependency of the form $N_{i_1}[K_p(R_j) \cap R_{i_1}] \subseteq N_j[K_p(R_j) \cap R_{i_1}]$, too. The natural join $\underset{k=1}{\overset{n}{\triangleright\!\triangleleft}} r(N_{i_k})$ represents the instance of the relation scheme $(\bigcup_{k=1}^{n} R_{i_k}, K_i)$. If, for a tuple u from $r(N_{i_1})$, there is no tuple t in $\underset{k=1}{\overset{n}{\triangleright\!\triangleleft}} r(N_{i_k})$ such that $u = t[R_{i_1}]$, than the inclusion dependency $N_{i_1}[K_p(R_j) \cap R_{i_1}] \subseteq N_j[K_p(R_j) \cap R_{i_1}]$ connects the tuple u with $r(N_j)$. The basic foreign key constraint $N_{i_1}[K_p(R_j)] \subseteq N_j[K_p(R_j)]$ represents the special case of the extended foreign key constraint for $i_1 = i_n$, i.e. $S_r = S_p = \emptyset$

Example 2. In the case of the closure graph presented in Figure 1, the extended foreign key constraint between the relation *Ord_Itm* and *Prc_Lst* would be

$$\triangleright\triangleleft (Ord\_Itm, Ord\_Hed)[(IDS, IDA)] \subseteq Prc\_Lst[(IDS, IDA)].$$

The semantics of this constraint is that the order items can contain only the articles of the supplier to whom order is sent. If instead of that constraint the inclusion dependencies $Ord\_Itm[IDA] \subseteq Prc\_Lst[IDA]$ and $Ord\_Hed[IDS] \subseteq Prc\_Lst[IDS]$ were defined, each item of a single order can contain articles of some other supplier under the condition that it is present in the price list. □

The extended foreign key constraint has three tasks. The first one is to prevent insertion of the tuple u into the relation $r(N_{i_1})$, if after that insertion, the database state would not be consistent, according to the chosen type of extended foreign key constraint. The second task is to prevent such

modification of the attribute A from X in a tuple of $r(N_{i_1})$, or in a tuple of any relation $r(N_l)$, where $N_l \in S_r$, if, after that modification tuple becomes invalid, due to the chosen type of the extended foreign key constraint.

The third task of the extended foreign key constraint is the handling of the consequences of the deletion of the tuple v from the relation $r(N_j)$. These handling procedures are called actions and are going to be analyzed only in the case of partially extended foreign key constraint. Actions are to be applied to those tuples u from the relation $r(N_{i_1})$, for which the following condition holds

$$((\forall t \in (r(N_j) \setminus \{v\}))(\exists A \in R_i \cap K_p(R_j))(u[A] \neq \omega \wedge u[A] \neq t[A]) \vee$$

$$(6) \qquad ((\exists s \in \overset{n}{\underset{k=1}{\bowtie}} r(N_{i_k}))(u = s[R_{i_1}] \wedge (\forall t \in (r(N_j) \setminus \{v\}))$$
$$(\exists A \in K_p(R_j))(s[A] \neq \omega \wedge s[A] \neq t[A]))).$$

In other words, the tuples u of $r(N_{i_1})$ that are to be handled, are the ones that would remain unconnected to any tuple of $r(N_j)$ after the deletion of the tuple v from $r(N_j)$.

If the *restrict* action is declared, then the deletion of the tuple v from relation $r(N_j)$ is not to be performed, if relation $r(N_{i_1})$ contains at least one tuple u satisfying (6).

If the *cascade deletion* action is declared, then the deletion of the tuple v from the relation $r(N_j)$ leads to the deletion of all the tuples u from the relation $r(N_{i_1})$ satisfying (6).

If the *set null* action is declared, then the deletion of the tuple v from the relation $r(N_j)$ leads to insertion of null values for all attributes $A \in (X \cap R_{i_1})$ of all tuples u of the relation $r(N_{i_1})$ satisfying (6).

If the *set default* action is declared, then the deletion of the tuple v from the relation $r(N_j)$ leads to the insertion of the corresponding default value into each attribute $A \in (X \cap R_{i_1})$ of all the tuples u of the relation $r(N_{i_1})$ satisfying (6).

The database designer specifies the actions according to: the real system business rules and the structures of relation schemes N_{i_1} and N_j. All four deletion handling activities require detection of those tuples $u \in r(N_{i_1})$ that fulfil the condition (6). This detection requires the access to all the tuples

of the relation $r(N_{i_1})$ and to every tuple of the natural join $\overset{n}{\underset{k=1}{\triangleright\!\triangleleft}} r(N_{i_k})$.

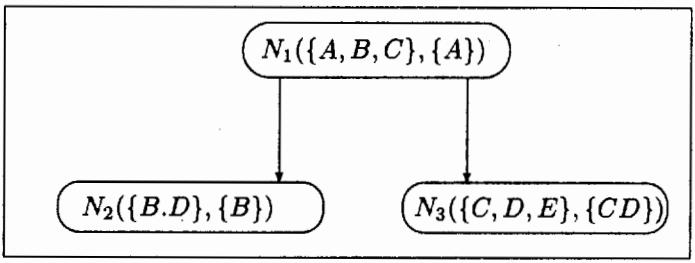

Figure 2

Example 3. Figure 3 represents an instance of the database scheme whose closure graph is presented in Figure 2, with the assumption that the attribute C within the relation scheme N_1, and the attribute D within the relation scheme N_2, are declared as that may have null values. It is easily proved that the tuples (a_1, b_1, c_1), $(a_5, b_4, c_3) \in r_1$ are satisfy the extended foreign key constraint $\triangleright\triangleleft (N_1, N_2)[(C, D)] \subseteq N_3[(C, D)]$. The remaining tuples satisfy that constraint according to the following:

- tuple (a_2, b_2, c_1), because $r_1 \triangleright\triangleleft r_2$ contains the tuple (a_2, b_2, c_1, ω), and c_1 is in $r_3[C]$,

$r_1 =$

A	B	C
a_1	b_1	c_1
a_2	b_2	c_1
a_3	b_3	ω
a_4	ω	c_2
a_5	b_4	c_3

$r_2 =$

B	D
b_1	d_1
b_2	ω
b_3	d_3
b_4	d_4
b_5	d_5

$r_3 =$

C	D	E
c_1	d_1	e_1
c_1	d_2	e_1
c_2	d_2	e_2
c_1	d_3	e_3
c_3	d_4	e_3

Figure 3.

- tuple (a_3, b_3, ω), because $r_1 \triangleright\triangleleft r_2$ contains the tuple (a_3, b_3, ω, d_3), and d_3 is in $r_3[D]$,

- tuple (a_4, ω, c_2), based on the fact that $r_1 \triangleright\triangleleft r_2$ does not contain such tuple t for which $t[ABC] = (a_4, \omega, c_2)$ is satisfied, but $(R_1 \cap K_p(R_3)(t)) = (C)$, and the relation r_3 contains the tuple v, for which $v[C] = c_2$.

Further on, only a few aspects of the database instance of Figure 3 updates are illustrated. Regarding the insertion of new tuples, the extended foreign key constraint $\triangleright\triangleleft (N_1, N_2)[(C, D)] \subseteq N_3[(C, D)]$ allows insertion of the tuple (a_6, ω, ω) into the relation r_1. But it does not allow insertion of the tuple (a_7, b_3, c_2) into the relation r_1, because there is no tuple v, such that $v[CD] = (c_2, d_3)$ in the relation r_3. Also, it does not allow insertion of the tuple (a_7, ω, c_4) into the relation r_1 as well, because the relation r_3 does not contain a tuple v, for which $v[C] = c_4$.

Let for the foreign key constraint $\triangleright\triangleleft (N_1, N_2)[(C, D)] \subseteq N_3[(C, D)]$ the deletion action for the relation r_3 be defined as set null. Then deleting the tuple (c_3, d_4, e_3) from r_3 leads to modification of the tuple (a_5, b_4, c_3) of r_1 to (a_5, b_4, ω). If the action of the cascade deletion were defined, then deleting the tuple (c_3, d_4, e_3) from r_3 would lead to the deletion of the tuple (a_5, b_4, c_3) from r_1.

Modifications of the attribute C in the relation $r(N_1)$, and of the attribute D in the relation $r(N_2)$ are handled in a similar way as insertions.

4. Conclusion

The extended referential integrity, as an interrelation constraint, is introduced. In addition to a precise definition of that constraint, it is pointed out that it can be either partial, full, or implicit, i.e. one that is neither partial nor full. The tasks of the extended referential integrity in maintaining the database in the consistent state are discussed, as well.

Present database management systems do not provide the possibilities for declarative specification of the extended referential integrity. Its specification has to be done procedurally, by means of database triggers.

References

[C79] Codd, E.F., Extending the Relational Database Model to Capture More Meaning, ACM TODS Vol. 4, No. 4 (1979), 397-434.

[Ca84] Casanova, M. A., Fagin, R., Papadimitriu, C. H., Inclusion Dependencies and Their Interaction with Functional Dependencies, Journal of Computer and System Sciences, Vol. 28, No. 1 (1984), 29-59.

[D81] Date, C. J., Referential Integrity, Proc. 7th International Conference on Very Large Databases, Canes, France, September 1981.

[D92] Date, C. J., Composite Foreign Keys and Nulls, In C. J. Date and Hugh Drawen, Relational Database Writings 1989-1991, Reading, Mass., Addison-Wesley, 1992.

[D94] Date, C. J., Drawen, H., A Guide to the SQL Standard, Third edition, Reading, Mass., Addison-Wesley, 1994.

[H92] Horowitz, B. M., Run-Time Execution Model for Referential Integrity Mainte- nance, Proc. 8th International Data Engineering Conference, Phoenix, Ariz., February 1992.