MTA SZTAKI Közlemények 37/1987 pp. 171-183

DESIGN TOOLS FOR LARGE RELATIONAL DATABASE SYSTEMS

B. THALHEIM

Technische Universität Dresden
Sektion Mathematik
GDR 8027 Dresden

INTRODUCTION

Today, database is a fashionable word. Two opposing research directions in databases were initiated in the early 70's, the introduction of the relational model and the development of semantic database models. The relational model revolutionized the field by dramatically separating data representation from underlying implementation. Significantly, the inherent simplicity in the relational model permitted the development of powerful, nonprozedural query languages and many useful theoretical results. Much effort is being devoted to establish a concrete foundation for database technology in order to design more efficient and transparent systems. The first and almost solved foundation step for database technology is the precise definition of data model. A database model is a collection of mathematically sound concepts defining the desired structural and behavioural properties of objects involved in a database application. In the axiomatic approach, database models are defined by the properties of its structures and operators using conventional mathematics and logic to define the structural and behavioural properties of objects within the database model. The second foundation step for database

This work was partially supported by the Hungarian National Scince Foundation, Grant No. 1066/ 1985

technology is the precise definition of semantics of data model. In terms of logic, the semantics of each database within the database model can be deduced precisely by the application of valid inference rules to sets of axioms. The semantics of a syntactically correct schema is given by the axioms which characterize the databases to be accepted, the "real world" databases.
One of the most important database models is the relational model. All data are seen as being stored in tables, with each row in the table having the same format. Each row in the table summarizes properties of some object or relationship in the real world. One benefit and aim of the relational model is to provide a methdological approach for design of schemata and databases. This benefit is based on a powerful theory whose kernel is the theory of dependencies and constraints. Database dependencies can be seen as a language for specifying the semantics of databases. They constitute an inherent property of database systems and express the different ways that data are associated with one another. Today we know at least five arreas of application of dependency theory: normalization for a more efficient storage, serch and modification; reduction of relations to subsets with the same information together with semantic constraints; utilization of dependencies for deriving new relations from basic relations in the view concept or in so-called deductive databases; verification of dependencies for a more powerful and user-friendly, nearly natural language design of databases; transformation of queries into more efficient search strategies.

In this paper some tools for database design with database dependencies are considered. More user-friendly functional dependencies are introduced in chapter 1. In chapter 2, the utilization of negative information is examined. A reduction principle is introduced in chapter 3. The horizontal decomposition with union constraints is analyzed in chapter 4.

Now the main basic conceptions are briefly introduced.

A (strong many-sorted relational database) preschema $S = (U , Rel , \preceq)$ is given by a finite set U of attributes, by a finite set $Rel = \{P_1, \ldots, P_k\}$ of predicates or predicate

names and by an arity function τ : Rel $\longrightarrow U^+$ which associates with every predicate P from Rel a string $\tau(P) = A_1 \ldots A_n$ of U . Let $V_U = (V_A / A \in U)$ be a family of variables. The set $L(S,V_U)$ of (S,V_U)-formulas is the smallest set whose elements are strings from $V_U \cup$ Rel meeting the following conditions:

(1) If $P \in$ Rel with $\tau(P) = A_1 \ldots A_k$ and $x_i \in V_{A_i}$ $(1 \leq i \leq k)$ then $P(x_1,\ldots,x_k) \in L(S,V_U)$ (or briefly $P(\tilde{x})$ or $P(\tilde{x},\tilde{y})$).

(2) If $x,y \in V_A$ then $x=y \in L(S,V_U)$.

(3) If α, $\beta \in L(S,V_U)$ and $x \in V_A$ for some $A \in U$ then $\{(\alpha \wedge \beta), (\neg\alpha), (\alpha \rightarrow \beta), \forall x \alpha, \exists x \alpha\} \subseteq L(S,V_U)$.

A relational database <u>schema</u> is a pair $S^* = (S,\Sigma)$ of a preschema S and a set of formulas Σ from $L(S,V_U)$ called integrity constraints.

For a given preschema $S = (U, Rel=\{P_1,\ldots,P_k\}, \tau)$ a S-database $M = (\underline{D}_U , R_1,\ldots,R_k)$ is given by the family $\underline{D}_U = (D_A / A \in U)$ of domains and by an Rel-indexed family $(R_1,\ldots,R_k)$ of relations, so that every predicate P_i Rel with $\tau(P_i) = A_1 \ldots A_m$ the relation R_i is a non-empty finite set of functions r from $\{A_1,\ldots,A_m\} = /\tau(P_i)/$ to $\bigcup_{j=1}^{m} D_{A_j}$ with $r(A_i) \in D_{A_i}$. We denote the function r by an m-tuple $(r(A_1),r(A_2),\ldots,r(A_m))$.

Now interpretations I_M of V_U on S-databases M can be introduced as mappings I_M: $V_U \longrightarrow \underline{D}_U$ with $I_M(x) \in D_A$ for $x \in V_A$. In the usual way, the (M,I_M)-satisfaction of formulas α , denoted by $\models_M \alpha[I_M]$, can be defined. A formula $\alpha \in L(S,V_U)$ is said to be valid in M , denoted $\models_M \alpha$, if $\models_M \alpha[I_M]$ for each interpretation I_M of V_U on M . M is said to be a model of a set Σ of formulas if any $\alpha \in \Sigma$ is valid in M .

A set of formulas Σ implies a formula α if for any model M

of Σ $\models_M \alpha$ (denoted by $\Sigma \models \alpha$) .

Let $S^* = (S, \Sigma)$ be a schema. A S-database is called S^*-database if M is a model of Σ .

A (Hilbert-style) formal system Γ for a set K of formulas consists of axioms $\subseteq K$ and inference rules $\frac{\Sigma}{\alpha}$ ($\Sigma \subseteq K$, $\alpha \in K$). A rule $\frac{\Sigma}{\alpha}$ is called sound if $\Sigma \models \alpha$. A derivation of a formula α from a set Σ is a sequence of formulas $\alpha_1, \ldots, \alpha_k = \alpha$ where each α_i is an axiom, or a member of Σ , or is inferred from preceding formulas in the sequence by an inference rule of Γ . If there is a derivation of α from Σ in Γ we write $\Sigma \vdash_\Gamma \alpha$. A formal system Γ is called sound iff from $\Sigma \vdash_\Gamma \alpha$ follows $\Sigma \models \alpha$ and is called complete for a class K iff for $\Sigma \subseteq K$, $\alpha \in K$ from $\Sigma \models \alpha$ follows $\Sigma \vdash_\Gamma \alpha$.

1. FUNCTIONAL DEPENDENCIES

One of the directions of further development of relational database theory and relational databases is the improvement of friendliness of user languages. Most of languages proposed only idiosyncratic versions of operations of the relational algebra or calculi and "adhocness" prevailed over some clearly understood notion of simplicity.

Generalized functional constraints are formulas from $L(S, V_U)$ with $S = (\{A_1, \ldots, A_n\}, \{P\}, \tau)$, $\tau(P) = A_1 \ldots A_n$

$\forall y_1 \ldots \forall y_m (P(\vec{x}) \wedge P(\vec{y}) \wedge \alpha \longrightarrow \beta)$ where α and β are quantifier-free, predicate-free formulas, the set of variables occuring in $P(\vec{x})$ and $P(\vec{y})$ is exactly the set $\{y_1, \ldots, y_m\}$ and a superset of the set of variables occuring in α and β .

They are called in (SDPF 81) Boolean dependencies and considered in (THAL 85).

Example 1. We consider $S=(U=\{$LECTURER, COURSE'UNIT, GROUP, ROOM, TIME$\}, \{$TIMETABLE$\}, \tau)$ with some restrictions, for instance:

1) Any room is reserved only for one group at the same time.
2) If there is a lecture given by more than one lecturer then participants are different.

These constraints can be formulated with the following formulas:

$$\alpha_1 = \forall x_1 \ldots \forall x'_5 (P(x_1,x_2,x_3,x_4,x_5) \wedge P(x'_1,x'_2,x'_3,x'_4,x'_5) \wedge x_4 = x'_4 \longrightarrow (x_3 = x'_3 \vee (\neg x_5 = x'_5)))$$

$$\alpha_2 = \forall x_1 \ldots \forall x'_5 (P(x_1,x_2,x_3,x_4,x_5) \wedge P(x'_1,x'_2,x'_3,x'_4,x'_5) \wedge x_2 = x'_2 (\neg x_1 = x'_1) \longrightarrow (\neg x_3 = x'_3)))$$

Generalized constraints are of importance for a more natural definition of dependencies of functional kind and unifies all these dependencies. They can be introduced in a more intuitive manner.

By T_1^i the class of all n-ary Boolean functions $f(x_1,\ldots,x_n)$ with $f(1,\ldots,1) = i$ for $i \in \{0,1\}$ is denoted.

A pair (f,g) of n-ary functions from $(T_1^1 \times T_1^1) \cup (T_1^0 \times T_1^0)$ is called <u>generalized functional dependency</u>.

Given a S-Database $M=(\underline{D}_U,R)$, $U=\{A_1,\ldots,A_n\}$. For a Boolean function f we can define a binary relation $\tilde{f}$ on R:

$r \tilde{f} r'$ iff $f(\sigma'_1(r,r'),\ldots,\sigma'_n(r,r')) = 1$

where $\sigma'_i(r,r')$ denotes the function

$$\sigma'_i(r,r') = \begin{cases} 0 & \text{if } r(A_i) \neq r'(A_i) \\ 1 & \text{if } r(A_i) = r'(A_i) \end{cases} \quad (1 \leq i \leq n).$$

Now we define $\models_M (f,g)$ for pairs of n-ary Boolean functions:

$\models_M (f,g)$ iff for any $r,r' \in R$ from $r \tilde{f} r'$ follows $r \tilde{g} r'$.

Corollary 1. If for a pair of n-ary Boolean functions f,g and a S-database $M = (\underline{D}_U,R)$ $\models_M (f,g)$ holds then (f,g) is a generalized functional dependency. For any generalized functional con-

straint $\propto$ which has a model there exists a generalized functional dependency (f,g) such that for any S-Database M $\models_M \propto$ iff $\models_M (f,g)$.

Example 1 (continued). There are Boolean functions $f_1(x_1,\ldots,x_5)$, $f_2(x_1,\ldots,x_5)$, $g_1(x_1,\ldots,x_5)$, $g_2(x_1,\ldots,x_5)$ such that (f_1,g_1) and (f_2,g_2) are equivalent to $_1$ and $_2$:
$f_1 = x_4$, $g_1 = x_3 \vee \bar{x}_5$, $f_2 = x_2 \wedge \bar{x}_1$, $g_2 = \bar{x}_3$.

Some special generalized functional dependencies are the strong functional dependency, dual functional dependency, weak functional dependency (DEGY 82), functional dependency (DEAB 85), monotonic functional dependency (THAL 87) and key dependency. The theory of all these special generalized functional dependencies can be unified and simplified by a theory of generalized functional dependencies which is based on the following main characterization theorem for implication.
For Boolean functions f,g the inequality $f \leq g$ holds if for any value $\tilde{\sigma}$ from $f(\tilde{\sigma}) = 1$ follows $g(\tilde{\sigma}) = 1$.

Theorem 2 (THAL 85). Let $(f_1,g_1),\ldots,(f_m,g_m)$, (f,g) be generalized functional dependencies. Then $\{(f_1,g_1),\ldots,(f_m,g_m)\} \models (f,g)$ holds iff $\bigwedge_{i=1}^{m} (f_i \rightarrow g_i) \leq f \rightarrow g$ holds.

Corollary 3. Let be (f_1,g_1), (f_2,g_2) generalized functional dependencies.

1. If $f_1 \geq f_2$ and $g_1 \leq g_2$ then $\{(f_1,g_1)\} \models (f_2,g_2)$.
2. $\{(f_1,g_1), (f_2,g_2)\} \models (f_1 \wedge f_2, g_1 \wedge g_2)$.
3. $\{(f_1,g_1), (f_2,g_2)\} \models (f_1 \vee f_2, g_1 \vee g_2)$.
4. If $g_1 \leq f_2$ then $\{(f_1,g_1),(f_2,g_2)\} \models (f_1,g_2)$.

Example 1 (continued). With theorem 2 the following generalized functional dependencies follows from $(f_1,g_1),(f_2,g_2)$:
$(x_4 \wedge x_5, x_3)$, $(x_2 \wedge x_3, x_1)$, $(\bar{x}_1 \wedge x_2 \wedge x_4, \bar{x}_3 \wedge \bar{x}_5)$, $(x_2 \wedge x_4 \wedge x_5, x_1 \wedge x_3)$.

Corollary 4. For any system Σ of generalized functional depen-

dencies there exists an equivalent functional dependency (f_{Σ}, g_{Σ}).

Corollary 5. Testing whether two sets of generalized functional dependencies are equivalent is NP-complete. Testing whether two sets of generalized functional dependencies imply the same set of key dependencies is NP-complete.

From theorem 2 the known axiomatizations of different classes of special generalized functional dependencies can be derived. Several kinds of covers for functional dependencies can be better studied and understood using generalized functional dependencies.

2. EXCLUDED FUNCTIONAL DEPENDENCIES

It is useful, for database logical design, normalization and effective algorithms, to utilize the full information on given relations. It is well known that functional dependencies are the favourite constraints used to decompose relation schemes. This privilege is certainly due to the simplicity of the concept of functional dependencies, and to their widespread appearance in the real world. However, in a great number of applications there is a requirement to allow violation of some functional dependencies, i.e. functional dependencies that are desired, but that do not hold in the relation. For instance if there is given a database $M = (\underline{D}_U, R)$ and it is of interest which formulas from a class of formulas holds in M the strategy of recognizing the validity of formulas will be faster if negative information is used in next steps.

The formula

$\forall\tilde{x}\forall\tilde{y}\forall\tilde{y}'\forall\tilde{z}\forall\tilde{z}'\ (P(\tilde{x},\tilde{y},\tilde{z}) \wedge P(\tilde{x},\tilde{y}',\tilde{z}') \longrightarrow \tilde{y} = \tilde{y}')$ is called functional dependency and for $\tilde{x} = x_{i_1},\ldots,x_{i_k}$, $\tilde{y}=x_{j_1},..,x_{j_l}$, $X = \{A_i \,/\, i \in \{i_1,\ldots,i_k\}\}$, $Y = \{A_j \,/\, j \in \{j_1,\ldots,j_l\}\}$ denoted by $X \longrightarrow Y$.

The formula

$\exists\tilde{x}\exists\tilde{y}\exists\tilde{y}'\exists\tilde{z}\exists\tilde{z}'\ (P(\tilde{x},\tilde{y},\tilde{z}) \wedge P(\tilde{x},\tilde{y}',\tilde{z}') \wedge (\neg(y = y')))$ is called

excluded functional dependency and for $\tilde{x} = x_{i_1},\ldots,x_{i_k}$, $\tilde{y} = x_{j_1},\ldots,x_{j_1}$, $X = \{A_i / i \in \{i_1,\ldots,i_k\}\}$, $Y = \{A_j / j \in \{j_1,..,j_1\}\}$ denoted by $X \not\rightarrow Y$.

It is easy to find sound inference rules for functional dependencies and excluded functional dependencies.

(1) $\dfrac{X \rightarrow Y \,,\; X \rightarrow Z}{X \rightarrow Y \cup Z}$ (2) $\dfrac{X \rightarrow Y \,,\; Y \rightarrow Z}{X \rightarrow Z}$ (3) $\dfrac{X \rightarrow Y \cup Z}{X \cup W \rightarrow Y}$

(4) $\dfrac{X \rightarrow Y \,,\; Y \rightarrow Z}{X \cup V \cup W \rightarrow Z \cup W}$ (5) $\dfrac{X \rightarrow Y \,,\; X \not\rightarrow Z}{X \cup Y \not\rightarrow Z}$

(6) $\dfrac{Y \rightarrow Z \,,\; X \not\rightarrow Z}{X \not\rightarrow Y}$ (7) $\dfrac{X \cup V \not\rightarrow Y}{X \not\rightarrow Y}$ (8) $\dfrac{X \not\rightarrow Y}{X \not\rightarrow Y \cup Z}$ if $Y \neq \emptyset$

(9) $\dfrac{X \rightarrow Y \,,\; X \cup V \cup W \not\rightarrow Z \cup W}{Y \not\rightarrow Z}$ (10) $\dfrac{X \rightarrow Y \,,\; X \not\rightarrow Y \cup Z}{X \not\rightarrow Z}$

(11) $\dfrac{Y \rightarrow Z \,,\; X \cup V \cup W \not\rightarrow Z \cup W}{X \not\rightarrow Y}$ if $Z \neq \emptyset$.

The formal system Γ_1 consists of the axiom $\{X \rightarrow X / X \subseteq U\}$ and the rules (4), (9) and (11).

Theorem 6. The formal system Γ_1 is sound and complete for the class of functional dependencies and excluded functional dependencies.

The theorem is based on the following property and theorem 2.

Lemma. Let $\Sigma \cup \{\alpha\}$ be a set of functional and excluded functional dependencies. Then $\Sigma \not\models \alpha$ iff there is a database $M = (\underline{D}_U, R)$ with $/R/ = 2$, $\models_M$ and $\not\models_M \alpha$.

3. DEDUCTIVE BASIS OF RELATIONS

The normalization of relations is one of the most important tools for database design. The concept of special kinds of dependencies

has been proved to be useful in the design and analysis of relational databases. Besides the normalization there is possible another "normalization" of relations, the reduction of relations to a necessary basis. This basis will be called deductive basis since using special formulas we get the entry relation from its deductive basis. In most cases, the deductive basis is more efficient than other known normal forms. During the query phase, the formulas are used to generate all possible derivations of facts and thereby make them again explicit in the database.

Let $S = (U,\{P\},\tau)$ be a preschema with $U = \{A_1,\ldots,A_n\}$ and let $L(S,V_U)$ be a set of formulas.

A formula from $L(S,V_U)$

$\forall x_{11}\ldots\forall x_{mn}(P(x_{11},..,x_{1n})\wedge\ldots\wedge P(x_{m1},..,x_{mn}) \rightarrow P(x_{01},..,x_{0n}))$

is called <u>template dependency</u> if $x_{0i} \in \{x_{1i},x_{2i},\ldots,x_{mi}\}$ for any i, $1 \leq i \leq n$, and is called <u>join dependency</u> if moreover for all i,j $(1 \leq i < j \leq m)$ and k $(1 \leq k \leq n)$ from $x_{ik} = x_{jk}$ follows $x_{ik} = x_{0k}$ (no hidden conditions in premise).

Given a set Σ of template dependencies, a S-database $M = (\underline{D}_U,R)$ and the set of interpretations $I(M)$ of V_U on M.

Then we define the Σ-closure of M:

$Cl_0(\Sigma,R) = R$,

$Cl_{k+1}(\Sigma,R) = \{ r \mid$ there is an $\alpha = \forall\tilde{y}(P(\tilde{x}_1)\wedge\ldots\wedge P(\tilde{x}_m)\rightarrow P(\tilde{x}_0))$ in Σ and $I \in I(M) : I(x_i) \in Cl_k(\Sigma,R)$ $(1 \leq i \leq m)$ and $I(\tilde{x}_0) = r \}$ $\cup\ Cl_k(\Sigma,R)$;

$CL(\Sigma,R) = \bigcup_{k=0}^{\infty} Cl_k(\Sigma,R)$;

$CL(\Sigma,M) = (\underline{D}_U, CL(\Sigma,R))$.

<u>Corollary 7.</u> Given a S-database $(\underline{D}_U,R)$ and a set of template dependencies Σ, there is a finite natural number i such that $CL(\Sigma,R) = Cl_i(\Sigma,R)$.

Corollary 8. For any S-database M and a set of template dependencies Σ $\quad \models_{CL(\Sigma,M)} \Sigma$.

Given a set of template dependencies Σ and a S-database $M = CL(\Sigma,M)$. A subset R' of R is called Σ-deductive subset if $R = CL(\Sigma,R')$. A Σ-deductive subset R' of R which is minimal is called Σ-deductive basis of M .

There are two main problems.

1) Given a Σ-deductive basis R of M . How many steps are needed for the construction of the set $CL(\Sigma,R)$? Given a set Σ . Are there limits for the construction of the Σ-closure of databases ?

2) Given R and Σ . How to construct a Σ-deductive basis of R ?

For the second problem there are some algorithms. The first problem is harder. If for a given Σ the construction of Σ-closure of databases is unlimited then the utilization of Σ-deductive bases is not effective. In (THAL 84) the following property is proven.

Corollary 9. There are two binary join dependencies α_1, α_2 . and a template dependency α_3 such that for any natural i a S-database $M_i = (\underline{D}_U, R_i)$ exists with

$Cl_i(\{\alpha_1, \alpha_2\}, R_i) \neq CL(\{\alpha_1, \alpha_2\}, R_i)$ and

$Cl_i(\{\alpha_3\}, R_i) \neq CL(\{\alpha_3\}, R_i)$.

A set Σ of join dependencies is called Sheffer-set if there is an equivalent join dependency α_Σ .

Theorem 10. Given a Sheffer-set Σ of join dependencies. For any S-database $M = (\underline{D}_U, R)$ it holds that

$CL(\Sigma,M) = (\underline{D}_U, Cl_1(\Sigma,R))$.

4. HORIZONTAL DECOMPOSITION WITH UNION CONSTRAINTS

The purpose of this chapter is to introduce the notion of union constraints which is a type of database constraints not previously discussed in literature and to show that there exists a sound and complete formal system. In the database literature, there is a number of results, both positive and negative, for the existence of finite formal systems. The class of union constraints is the first class of nodependencies which is known to be axiomatizable. By an union constraint it is stated that there exists a cover of the relation with possibilities of "forgetting" some attributes.

We are given a preschema $S = (U,\{P\},\tau)$ with $U = \{A_1,\ldots,A_n\}$, $\tau(P) = A_1\ldots A_n$ and $X,Y \subseteq U$, $X \cup Y = U$.

The pair $[X,Y]$ is called <u>union constraint</u>.

A S-database $M = (\underline{D}_U,R)$ satisfies this constraint if there are subsets R_1,R_2 of R such that $R_1 \cup R_2 = R$ and $R = R_1(X) + R_2(Y)$ (denoted by $\models_M [X,Y]$) where $R'(Z)$ is the projection of R' on Z and $R' + R''$ is the generalized union on $\underline{D}_U$ or the sum (DEAB 85).

Obviously, the constraint $[X \cup Z, Y \cup Z]$ can be described with the following formulas from $L(S,V_U)$ for disjoint X,Y,Z :

$$\forall\tilde{x}\forall\tilde{y}\forall\tilde{z}\forall\tilde{x}'\forall\tilde{y}'\,(\,P(\tilde{x},\tilde{y},\tilde{z}) \rightarrow P(\tilde{x},\tilde{y}',\tilde{z}) \vee P(\tilde{x}',\tilde{y},\tilde{z}))\,.$$

Now we can define the formal system Γ_2.

Formal system Γ_2.

Ax $\quad [U,U]$

RU (1) $\dfrac{[X,Y]}{[V,W]}$ if $X \subseteq V$, $Y \subseteq W$ or $X \subseteq W$, $Y \subseteq V$

(2) $\dfrac{[X_1,X_2],\ [Y_1,Y_2]}{[X_1 \cap Y_1, Y_2]}$ if $X_1 \cap X_2 \subseteq Y_1$, $X_2 \subseteq Y_2$.

Theorem 11. The system Γ_2 is sound and complete for the class of union constraints.

The proof is this theorem uses an equivalence between union constraints and binary join dependencies.

REFERENCES

DEAB 85 C. Delobel, M. Adiba, Relational database systems. North-Holland, Amsterdam 1985.

DEGY 82 J. Demetrovics, Gy. Gyepesi, Logical dependencies in relational databases. MTZSAKI Tanulmanyok 133, 1982, Budapest, 59 - 78.

SDPF 81 Y. Sagiv, C. Delobel, D.S. Parker, R. Fagin, An equivalence between relational data base dependencies and a subclass of propositional logic. Journal of the ACM, 1981, 3, 435-453.

THAL 85 B. Thalheim, Funktionale Abhängigkeiten in relationalen Datenstrukturen. EIK, 1985, 1/2, 23 -33.

THAL 87 B. Thalheim, Dependencies in relational databases. Teubner, Leipzig, 1987.

Design tools for large relational database systems

B. Thalheim

Summary

In the paper some tools for database design with database dependencies are considered.
More user-friendly functional dependencies are introduced. The utilization of negative information is examined. Besides the usual normalization another "normalization " of relations, the reduction of relations to a necessary basis is introduced. Finally, the horizontal decomposition with union constraints is analyzed.

Nagyméretü relációs adatbázis-rendszerek tervezésének eszközei

B. Thalheim

Összefoglaló

A szerző a különböző függőségi tipusokkal biró adatbázisok tervezéséhez szolgáló eszközöket vizsgálja. Bevezet ujfajta "felhasználó-barátságosabb" funkcionális függőségeket. Megvizsgálja a negativ információ hasznositásának kérdéseit. Bevezet egy uj redukciós elvet. Végül analizálja az egyesités feltételeknél a horizontális dekompoziciót.