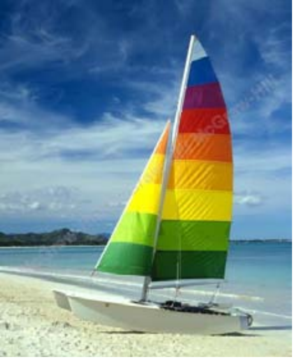


Appendix B: Advanced Relational Database Design

Database System Concepts, 6th Ed.

Appendix B: Advanced Relational Database Design

- Reasoning with MVDs
- Higher normal forms
  - Join dependencies and PJNF
  - DKNF

Theory of Multivalued Dependencies

- Let D denote a set of functional and multivalued dependencies. The closure D^+ of D is the set of all functional and multivalued dependencies logically implied by D.
- Sound and complete inference rules for functional and multivalued dependencies:

1. **Reflexivity rule**. If α is a set of attributes and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ holds.
2. **Augmentation rule**. If $\alpha \rightarrow \beta$ holds and γ is a set of attributes, then $\gamma \alpha \rightarrow \gamma \beta$ holds.
3. **Transitivity rule**. If $\alpha \rightarrow \beta$ holds and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds.

Theory of Multivalued Dependencies (Cont.)

4. **Complementation rule.** If $\alpha \rightarrow\rightarrow \beta$ holds, then $\alpha \rightarrow\rightarrow R - \beta - \alpha$ holds.
5. **Multivalued augmentation rule.** If $\alpha \rightarrow\rightarrow \beta$ holds and $\gamma \subseteq R$ and $\delta \subseteq \gamma$, then $\gamma\,\alpha \rightarrow\rightarrow \delta\,\beta$ holds.
6. **Multivalued transitivity rule**. If $\alpha \rightarrow\rightarrow \beta$ holds and $\beta \rightarrow\rightarrow \gamma$ holds, then $\alpha \rightarrow\rightarrow \gamma - \beta$ holds.
7. **Replication rule.** If $\alpha \rightarrow \beta$ holds, then $\alpha \rightarrow\rightarrow \beta$.
8. **Coalescence rule.** If $\alpha \rightarrow\rightarrow \beta$ holds and $\gamma \subseteq \beta$ and there is a δ such that $\delta \subseteq R$ and $\delta \cap \beta = \emptyset$ and $\delta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ holds.

Simplification of the Computation of D^+

- We can simplify the computation of the closure of *D* by using the following rules (proved using rules 1-8).
  - **Multivalued union rule.** If $\alpha \rightarrow\rightarrow \beta$ holds and $\alpha \rightarrow\rightarrow \gamma$ holds, then $\alpha \rightarrow\rightarrow \beta\gamma$ holds.
  - **Intersection rule.** If $\alpha \rightarrow\rightarrow \beta$ holds and $\alpha \rightarrow\rightarrow \gamma$ holds, then $\alpha \rightarrow\rightarrow \beta \cap \gamma$ holds.
  - **Difference rule.** If If $\alpha \rightarrow\rightarrow \beta$ holds and $\alpha \rightarrow\rightarrow \gamma$ holds, then $\alpha \rightarrow\rightarrow \beta - \gamma$ holds and $\alpha \rightarrow\rightarrow \gamma - \beta$ holds.

Example

- $R = (A, B, C, G, H, I)$
  $D = \{A \twoheadrightarrow B$
  $B \twoheadrightarrow HI$
  $CG \rightarrow H\}$
- Some members of D^+:
  - $A \twoheadrightarrow CGHI$.
    Since $A \twoheadrightarrow B$, the complementation rule (4) implies that $A \twoheadrightarrow R - B - A$.
    Since $R - B - A = CGHI$, so $A \twoheadrightarrow CGHI$.
  - $A \twoheadrightarrow HI$.
    Since $A \twoheadrightarrow B$ and $B \twoheadrightarrow HI$, the multivalued transitivity rule (6) implies that $B \twoheadrightarrow HI - B$.
    Since $HI - B = HI$, $A \twoheadrightarrow HI$.

Example (Cont.)

- Some members of D^+ (cont.):
  - $B \rightarrow H$.
    Apply the coalescence rule (8); $B \rightarrow\rightarrow HI$ holds.
    Since $H \subseteq HI$ and $CG \rightarrow H$ and $CG \cap HI = \emptyset$, the coalescence rule is satisfied with α being B, β being HI, δ being CG, and γ being H. We conclude that $B \rightarrow H$.
  - $A \rightarrow\rightarrow CG$.
    $A \rightarrow\rightarrow CGHI$ and $A \rightarrow\rightarrow HI$.
    By the difference rule, $A \rightarrow\rightarrow CGHI - HI$.
    Since $CGHI - HI = CG$, $A \rightarrow\rightarrow CG$.

Normalization Using Join Dependencies

- Join dependencies constrain the set of legal relations over a schema *R* to those relations for which a given decomposition is a lossless-join decomposition.
- Let *R* be a relation schema and R_1 , R_2 ,..., R_n be a decomposition of *R*. If $R = R_1 \cup R_2 \cup \ldots. \cup R_n$, we say that a relation *r*(*R*) satisfies the *join dependency* $*(R_1 , R_2 ,..., R_n)$ if:

  $$r = \prod_{R1} (r) \bowtie \prod_{R2} (r) \bowtie \ \ldots\ldots \bowtie \prod_{Rn}(r)$$

  A join dependency is *trivial* if one of the R_i is *R* itself.
- A join dependency $*(R_1, R_2)$ is equivalent to the multivalued dependency $R_1 \cap R_2 \rightarrow\rightarrow R_2$. Conversely, $\alpha \rightarrow\rightarrow \beta$ is equivalent to $*(\alpha \cup (R - \beta), \alpha \cup \beta)$
- However, there are join dependencies that are not equivalent to any multivalued dependency.

Project-Join Normal Form (PJNF)

- A relation schema R is in PJNF with respect to a set D of functional, multivalued, and join dependencies if for all join dependencies in D^+ of the form

  $*(R_1, R_2, ..., R_n)$ where each $R_i \subseteq R$

  and $R = R_1 \cup R_2 \cup ... \cup R_n$

  at least one of the following holds:
  - $*(R_1, R_2, ..., R_n)$ is a trivial join dependency.
  - Every R_i is a superkey for R.
- Since every multivalued dependency is also a join dependency,

  every PJNF schema is also in 4NF.

Example

- Consider *Loan-info-schema* = (*branch-name, customer-name, loan-number, amount*).
- Each loan has one or more customers, is in one or more branches and has a loan amount; these relationships are independent, hence we have the join dependency
- *(=(*loan-number, branch-name*), (*loan-number, customer-name*), (*loan-number, amount*))
- *Loan-info-schema* is not in PJNF with respect to the set of dependencies containing the above join dependency. To put *Loan-info-schema* into PJNF, we must decompose it into the three schemas specified by the join dependency:
  - (*loan-number, branch-name*)
  - (*loan-number, customer-name*)
  - (*loan-number, amount*)

Domain-Key Normal Form (DKNY)

- **Domain declaration**. Let A be an attribute, and let **dom** be a set of values. The domain declaration $A \subseteq$ **dom** requires that the *A* value of all tuples be values in **dom**.
- **Key declaration**. Let *R* be a relation schema with $K \subseteq R$. The key declaration **key** (*K*) requires that *K* be a superkey for schema *R* ($K \rightarrow R$). All key declarations are functional dependencies but not all functional dependencies are key declarations.
- **General constraint**. A general constraint is a predicate on the set of all relations on a given schema.
- Let **D** be a set of domain constraints and let **K** be a set of key constraints for a relation schema *R*. Let **G** denote the general constraints for *R*. Schema *R* is in DKNF if **D** $\cup$ **K** logically imply **G**.

Example

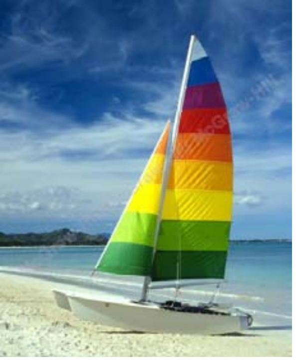
- Accounts whose *account-number* begins with the digit 9 are special high-interest accounts with a minimum balance of 2500.
- General constraint: ``If the first digit of *t* [*account-number*] is 9, then *t* [*balance*] ≥ 2500.''
- DKNF design:

  Regular-acct-schema = (*branch-name, account-number, balance*)

  Special-acct-schema = (*branch-name, account-number, balance*)
- Domain constraints for {*Special-acct-schema*} require that for each account:
  - The account number begins with 9.
  - The balance is greater than 2500.

DKNF rephrasing of PJNF Definition

- Let $R = (A_1, A_2, \ldots, A_n)$ be a relation schema. Let dom(A_i) denote the domain of attribute A_i, and let all these domains be infinite. Then all domain constraints **D** are of the form $A_i \subseteq$ **dom** (A_i).
- Let the general constraints be a set **G** of functional, multivalued, or join dependencies. If F is the set of functional dependencies in **G**, let the set **K** of key constraints be those nontrivial functional dependencies in F^+ of the form $\alpha \rightarrow R$.
- Schema R is in PJNF if and only if it is in DKNF with respect to **D**, **K**, and **G**.



End of Appendix B

Figure B.01

r_1:

A	B
a_1	b_1
a_2	b_1

r_2:

C	G	H
c_1	g_1	h_1
c_2	g_2	h_2

r_3:

A	I
a_1	i_1
a_2	i_2

r_4:

A	C	G
a_1	c_1	g_1
a_2	c_2	g_2

Figure B.02

A	B	C	G	H	I
a_1	b_1	c_1	g_1	h_1	i_1
a_2	b_1	c_2	g_2	h_2	i_2

Figure B.03

	$R_1 - R_2$	$R_1 \cap R_2$
$\Pi_{R_1}(t_1)$	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$
$\Pi_{R_1}(t_2)$	$b_1 \ldots b_i$	$a_{i+1} \ldots a_j$

	$R_1 \cap R_2$	$R_2 - R_1$
$\Pi_{R_2}(t_1)$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
$\Pi_{R_2}(t_2)$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$

Figure B.04

	$R_1 - R_2$	$R_1 \cap R_2$	$R_2 - R_1$
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$b_1 \ldots b_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$b_1 \ldots b_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$

Figure B.05

A	B	C
a_1	b_1	c_2
a_2	b_1	c_1
a_1	b_2	c_1
a_1	b_1	c_1

Figure B.06

A	B	C
a_1	b_1	c_2
a_2	b_1	c_1
a_1	b_2	c_1
a_1	b_1	c_1

Figure B.07

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3