Relational Database Design Algorithms and Further Dependencies

Outline

- Designing a Set of Relations
- Properties of Relational Decompositions
- Algorithms for Relational Database Schema
- Multivalued Dependencies and Fourth Normal Form
- Join Dependencies and Fifth Normal Form
- Inclusion Dependencies
- Other Dependencies and Normal Forms

DESIGNING A SET OF RELATIONS (1)

- **The Approach of Relational Synthesis (Bottom-up Design):**
  - Assumes that all possible functional dependencies are known.
  - First constructs a minimal set of FDs
  - Then applies algorithms that construct a target set of 3NF or BCNF relations.
  - Additional criteria may be needed to ensure that the *set of relations* in a relational database are satisfactory

DESIGNING A SET OF RELATIONS (2)

- **Goals:**
  - Lossless join property (a must)
  - Dependency preservation property
  - Attribute Preservation property

1. Properties of Relational Decompositions (1)

- **Relation Decomposition and Insufficiency of Normal Forms:**
  - Universal Relation Schema:
    - A relation schema R = {A1, A2, …, An} that includes all the attributes of the database.
  - Universal relation assumption:
    - Every attribute name is unique.

Properties of Relational Decompositions (2)

- **Relation Decomposition and Insufficiency of Normal Forms (cont.):**
  - Decomposition:
    - The process of decomposing the universal relation schema R into a set of relation schemas D = {R1,R2, …, Rm} that will become the relational database schema by using the functional dependencies.
  - Attribute preservation condition:
    - Each attribute in R will appear in at least one relation schema Ri in the decomposition so that no attributes are "lost".

Properties of Relational Decompositions (2)

- Another goal of decomposition is to have each individual relation Ri in the decomposition D be in BCNF or 3NF.
- Additional properties of decomposition are needed to prevent from generating spurious tuples

Properties of Relational Decompositions (3)

- **Dependency Preservation Property of a Decomposition:**
  - Definition: Given a set of dependencies F on R, the **projection** of F on R_i, denoted by $\pi_{R_i}(F)$ where R_i is a subset of R, is the set of dependencies $X \rightarrow Y$ in F^+ such that the attributes in $X \cup Y$ are all contained in R_i.
  - Hence, the projection of F on each relation schema R_i in the decomposition D is the set of functional dependencies in F^+, the closure of F, such that all their left- and right-hand-side attributes are in R_i.

Properties of Relational Decompositions (4)

- **Dependency Preservation Property of a Decomposition (cont.):**
  - Dependency Preservation Property:
    - A decomposition D = {R1, R2, ..., Rm} of R is **dependency-preserving** with respect to F if the union of the projections of F on each Ri in D is equivalent to F; that is

$$((\pi_{R1}(F)) \cup \ldots \cup (\pi_{Rm}(F)))^+ = F^+$$

- Claim 1:
  - It is always possible to find a dependency-preserving decomposition D with respect to F such that each relation Ri in D is in 3nf.

Properties of Relational Decompositions (5)

- **Lossless (Non-additive) Join Property of a Decomposition:**
  - Definition: Lossless join property: a decomposition D = {R1, R2, ..., Rm} of R has the **lossless (nonadditive) join property** with respect to the set of dependencies F on R if, for *every* relation state r of R that satisfies F, the following holds, where * is the natural join of all the relations in D:

$$* (\pi_{R1}(r), \ldots, \pi_{Rm}(r)) = r$$

  - Note: The word loss in lossless refers to loss of information, not to loss of tuples. In fact, for "loss of information" a better term is "**addition of spurious information**"

Properties of Relational Decompositions (6)

- **Lossless (Non-additive) Join Property of a Decomposition**

- **Algorithm 11.1: Testing for Lossless Join Property**
  - **Input**: A universal relation R, a decomposition D = {R1, R2, ..., Rm} of R, and a set F of functional dependencies.

1. Create an initial matrix S with one row i for each relation Ri in D, and one column j for each attribute Aj in R.

2. Set S(i,j):=bij for all matrix entries. (* each bij is a distinct symbol associated with indices (i,j) *).

3. For each row i representing relation schema Ri

    {for each column j representing attribute Aj

        {if (relation Ri includes attribute Aj) then set S(i,j):= aj;};};

  - (* each aj is a distinct symbol associated with index (j) *)

- CONTINUED on NEXT SLIDE

Properties of Relational Decompositions (7)

4. **Repeat the following loop until a complete loop execution results in no changes to S**

{for each functional dependency X →Y in F

{for all rows in S *which have the same symbols* in the columns corresponding to attributes in X

{make the symbols in each column that correspond to an attribute in Y be the same in all these rows as follows:

If any of the rows has an "a" symbol for the column, set the other rows to that *same* "a" symbol in the column.

If no "a" symbol exists for the attribute in any of the rows, choose one of the "b" symbols that appear in one of the rows for the attribute and set the other rows to that same "b" symbol in the column ;};

};

};

5. **If a row is made up entirely of "a" symbols, then the decomposition has the lossless join property; otherwise it does not.**

Properties of Relational Decompositions (8)

Lossless (nonadditive) join test for *n*-ary decompositions.
(a) Case 1: Decomposition of EMP_PROJ into EMP_PROJ1 and EMP_LOCS fails test.
(b) A decomposition of EMP_PROJ that has the lossless join property.

(a)

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2\}$

R_1=EMP_LOCS={ENAME, PLOCATION}

R_2=EMP_PROJ1={SSN, PNUMBER, HOURS, PNAME, PLOCATION}

F={SSN→ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

(no changes to matrix after applying functional dependencies)

(b)

EMP

SSN	ENAME

PROJECT

PNUMBER	PNAME	PLOCATION

WORKS_ON

SSN	PNUMBER	HOURS

Properties of Relational Decompositions (8)

Lossless (nonadditive) join test for n-ary decompositions.
(c) Case 2: Decomposition of EMP_PROJ into EMP, PROJECT, and WORKS_ON satisfies test.

(c)

R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2, R_3\}$
R_1=EMP={SSN, ENAME}
R_2=PROJ={PNUMBER, PNAME, PLOCATION}
R_3=WORKS_ON={SSN, PNUMBER, HOURS}

F={SSN→{ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(original matrix S at start of algorithm)

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	$\cancel{b_{32}}$ a_2	a_3	$\cancel{b_{34}}$ a_4	$\cancel{b_{35}}$ a_5	a_6

(matrix S after applying the first two functional dependencies - last row is all "a" symbols, so we stop)

Properties of Relational Decompositions (9)

- **Testing Binary Decompositions for Lossless Join Property**
  - **Binary Decomposition:** Decomposition of a relation R into two relations.
  - **PROPERTY LJ1 (lossless join test for binary decompositions):** A decomposition D = {R1, R2} of R has the lossless join property with respect to a set of functional dependencies F on R *if and only if* either
    - The f.d. $((R1 \cap R2) \rightarrow (R1 - R2))$ is in F^+, or
    - The f.d. $((R1 \cap R2) \rightarrow (R2 - R1))$ is in F^+.

Properties of Relational Decompositions (10)

- **Successive Lossless Join Decomposition:**
  - **Claim 2 (Preservation of non-additivity in successive decompositions):**
    - If a decomposition D = {R1, R2, ..., Rm} of R has the lossless (non-additive) join property with respect to a set of functional dependencies F on R,
    - and if a decomposition Di = {Q1, Q2, ..., Qk} of Ri has the lossless (non-additive) join property with respect to the projection of F on Ri,
      - then the decomposition D2 = {R1, R2, ..., Ri-1, Q1, Q2, ..., Qk, Ri+1, ..., Rm} of R has the non-additive join property with respect to F.

2. Algorithms for Relational Database Schema Design (1)

- **Algorithm 11.2: Relational Synthesis into 3NF with Dependency Preservation (Relational Synthesis Algorithm)**
  - **Input: A universal relation R and a set of functional dependencies F on the attributes of R.**

1. Find a minimal cover G for F (use Algorithm 10.2);

2. For each left-hand-side X of a functional dependency that appears in G,

create a relation schema in D with attributes {X ∪ {A1} ∪ {A2} ... ∪ {Ak}},

where X → A1, X → A2, ..., X → Ak are the only dependencies in G with X as left-hand-side (X is the key of this relation) ;

3. Place any remaining attributes (that have not been placed in any relation) in a single relation schema to ensure the attribute preservation property.

  - **Claim 3: Every relation schema created by Algorithm 11.2 is in 3NF.**

Algorithms for Relational Database Schema Design (2)

- **Algorithm 11.3: Relational Decomposition into BCNF with Lossless (non-additive) join property**
  - **Input: A universal relation R and a set of functional dependencies F on the attributes of R.**

1. Set D := {R};

2. While there is a relation schema Q in D that is not in BCNF

do {

choose a relation schema Q in D that is not in BCNF;

find a functional dependency $X \rightarrow Y$ in Q that violates BCNF;

replace Q in D by two relation schemas (Q - Y) and ($X \cup Y$);

};

Algorithms for Relational Database Schema Design (3)

- **Algorithm 11.4 Relational Synthesis into 3NF with Dependency Preservation and Lossless (Non-Additive) Join Property**
  - **Input: A universal relation R and a set of functional dependencies F on the attributes of R.**

1. Find a minimal cover G for F (Use Algorithm 10.2).

2. For each left-hand-side X of a functional dependency that appears in G,

create a relation schema in D with attributes {X ∪ {A1} ∪ {A2} ... ∪ {Ak}},

where X → A1, X → A2, ..., X –>Ak are the only dependencies in G with X as left-hand-side (X is the key of this relation).

3. If none of the relation schemas in D contains a key of R, then create one more relation schema in D that contains attributes that form a key of R. *(Use Algorithm 11.4a to find the key of R)*

Algorithms for Relational Database Schema Design (4)

- **Algorithm 11.4a Finding a Key K for R Given a set F of Functional Dependencies**
  - **Input: A universal relation R and a set of functional dependencies F on the attributes of R.**

1. Set K := R;
2. For each attribute A in K {

    Compute (K - A)+ with respect to F;

    If (K - A)+ contains all the attributes in R,

        then set K := K - {A};

   }

Algorithms for Relational Database Schema Design (7)

- **Discussion of Normalization Algorithms:**
- Problems:
  - The database designer must first specify *all* the relevant functional dependencies among the database attributes.
  - It is not always possible to find a decomposition into relation schemas that preserves dependencies and allows each relation schema in the decomposition to be in BCNF (instead of 3NF as in Algorithm 11.4).

Algorithms for Relational Database Schema Design (8)

Table 11.1
Summary of the Algorithms Discussed in Sections 11.1 and 11.2

Algorithm	Input	Output	Properties/Purpose	Remarks
11.1	A decomposition *D* of *R* and a set *F* of functional dependencies	Boolean result: yes or no for nonadditive join property	Testing for nonadditive join decomposition	See a simpler test in Section 11.1.4 for binary decompositions
11.2	Set of functional dependencies *F*	A set of relations in 3NF	Dependency preservation	No guarantee of satisfying lossless join property
11.3	Set of functional dependencies *F*	A set of relations in BCNF	Nonadditive join decomposition	No guarantee of dependency preservation
11.4	Set of functional dependencies *F*	A set of relations in 3NF	Nonadditive join *and* dependency-preserving decomposition	May not achieve BCNF, but achieves *all* desirable properties and 3NF
11.4a	Relation schema *R* with a set of functional dependencies *F*	Key K of *R*	To find a key K (that is a subset of *R*)	The entire relation *R* is always a default superkey

3. Multivalued Dependencies and Fourth Normal Form (1)

(a) The EMP relation with two MVDs: ENAME —>> PNAME and ENAME —>> DNAME.
(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(a) **EMP**

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(b) **EMP_PROJECTS**

ENAME	PNAME
Smith	X
Smith	Y

EMP_DEPENDENTS

ENAME	DNAME
Smith	John
Smith	Anna

3. Multivalued Dependencies and Fourth Normal Form (1)

(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R1, R2, R3). (d) Decomposing the relation SUPPLY into the 5NF relations R1, R2, and R3.

(c) **SUPPLY**

SNAME	PARTNAME	PROJNAME
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(d) **R1**

SNAME	PARTNAME
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R2

SNAME	PROJNAME
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R3

PARTNAME	PROJNAME
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

Multivalued Dependencies and Fourth Normal Form (2)

Definition:

- A **multivalued dependency** (**MVD**) $X \twoheadrightarrow Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t_1 and t_2 exist in r such that $t_1[X] = t_2[X]$, then two tuples t_3 and t_4 should also exist in r with the following properties, where we use Z to denote $(R - (X \cup Y))$:
  - $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.
  - $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$.
  - $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$.
- An MVD $X \twoheadrightarrow Y$ in R is called a **trivial MVD** if (a) Y is a subset of X, or (b) $X \cup Y = R$.

Multivalued Dependencies and Fourth Normal Form (3)

- **Inference Rules for Functional and Multivalued Dependencies**:
  - IR1 (**reflexive rule for FDs**): If $X \supseteq Y$, then $X \rightarrow Y$.
  - IR2 (**augmentation rule for FDs**): $\{X \rightarrow Y\} \models XZ \rightarrow YZ$.
  - IR3 (**transitive rule for FDs**): $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$.
  - IR4 (**complementation rule for MVDs**): $\{X \twoheadrightarrow Y\} \models X \twoheadrightarrow (R - (X \cup Y))\}$.
  - IR5 (**augmentation rule for MVDs**): If $X \twoheadrightarrow Y$ and $W \supseteq Z$ then $WX \twoheadrightarrow YZ$.
  - IR6 (**transitive rule for MVDs**): $\{X \twoheadrightarrow Y, Y \twoheadrightarrow Z\} \models X \twoheadrightarrow (Z - Y)$.
  - IR7 (**replication rule for FD to MVD**): $\{X \rightarrow Y\} \models X \twoheadrightarrow Y$.
  - IR8 (**coalescence rule for FDs and MVDs**): If $X \twoheadrightarrow Y$ and there exists W with the properties that
    - (a) $W \cap Y$ is empty, (b) $W \rightarrow Z$, and (c) $Y \supseteq Z$, then $X \rightarrow Z$.

Multivalued Dependencies and Fourth Normal Form (4)

Definition:

- A relation schema *R* is in **4NF** with respect to a set of dependencies *F* (that includes functional dependencies and multivalued dependencies) if, for every *nontrivial* multivalued dependency $X \twoheadrightarrow Y$ in F^+, *X* is a superkey for R.
  - Note: F^+ is the (complete) set of all dependencies (functional or multivalued) that will hold in every relation state *r* of *R* that satisfies *F*. It is also called the **closure** of *F*.

Multivalued Dependencies and Fourth Normal Form (5)

Decomposing a relation state of EMP that is not in 4NF:
(a) EMP relation with additional tuples.
(b) Two corresponding 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(a) **EMP**

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John
Brown	W	Jim
Brown	X	Jim
Brown	Y	Jim
Brown	Z	Jim
Brown	W	Joan
Brown	X	Joan
Brown	Y	Joan
Brown	Z	Joan
Brown	W	Bob
Brown	X	Bob
Brown	Y	Bob
Brown	Z	Bob

(b) **EMP_PROJECTS**

ENAME	PNAME
Smith	X
Smith	Y
Brown	W
Brown	X
Brown	Y
Brown	Z

EMP_DEPENDENTS

ENAME	DNAME
Smith	Anna
Smith	John
Brown	Jim
Brown	Joan
Brown	Bob

Multivalued Dependencies and Fourth Normal Form (6)

Lossless (Non-additive) Join Decomposition into 4NF Relations:

- **PROPERTY LJ1'**
  - The relation schemas R_1 and R_2 form a lossless (non-additive) join decomposition of R with respect to a set F of functional *and* multivalued dependencies if and only if
    - $(R_1 \cap R_2) \twoheadrightarrow (R_1 - R_2)$
  - or by symmetry, if and only if
    - $(R_1 \cap R_2) \twoheadrightarrow (R_2 - R_1)$.

Multivalued Dependencies and Fourth Normal Form (7)

Algorithm 11.5: Relational decomposition into 4NF relations with non-additive join property

- **Input:** A universal relation R and a set of functional and multivalued dependencies F.

1. Set D := { R };
2. While there is a relation schema *Q* in *D* that is not in 4NF do {
   choose a relation schema *Q* in *D* that is not in 4NF;
   find a nontrivial MVD $X \rightarrow\!\!\!\rightarrow Y$ in *Q* that violates 4NF;
   replace *Q* in *D* by two relation schemas ($Q - Y$) and ($X \cup Y$);
   };

4. Join Dependencies and Fifth Normal Form (1)

Definition:

- A **join dependency** (**JD**), denoted by JD(R_1, R_2, ..., R_n), specified on relation schema R, specifies a constraint on the states r of R.
  - The constraint states that every legal state r of R should have a non-additive join decomposition into R_1, R_2, ..., R_n; that is, for every such r we have
  - $$* (\pi_{R1}(r), \pi_{R2}(r), ..., \pi_{Rn}(r)) = r$$

  ***Note**: an MVD is a special case of a JD where n = 2.*
- A join dependency JD(R_1, R_2, ..., R_n), specified on relation schema R, is a **trivial JD** if one of the relation schemas R_i in JD(R_1, R_2, ..., R_n) is equal to R.

Join Dependencies and Fifth Normal Form (2)

Definition:

- A relation schema R is in **fifth normal form** (**5NF**) (or **Project-Join Normal Form** (**PJNF**)) with respect to a set F of functional, multivalued, and join dependencies if,
  - for every nontrivial join dependency JD(R_1, R_2, ..., R_n) in F^+ (that is, implied by F),
    - every R_i is a superkey of R.

Relation SUPPLY with Join Dependency and conversion to Fifth Normal Form

Figure 11.4
Fourth and fifth normal forms.
(a) The EMP relation with two MVDs: Ename $\twoheadrightarrow$ Pname and Ename $\twoheadrightarrow$ Dname.
(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.
(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R_1, R_2, R_3).
(d) Decomposing the relation SUPPLY into the 5NF relations R_1, R_2, R_3.

(c) SUPPLY

Sname	Part_name	Proj_name
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(d) R_1

Sname	Part_name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R_2

Sname	Proj_name
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R_3

Part_name	Proj_name
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

Acknowledgement

Reference for this lecture is

- Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems,* Pearson Education.

The authors and the publishers are gratefully acknowledged.