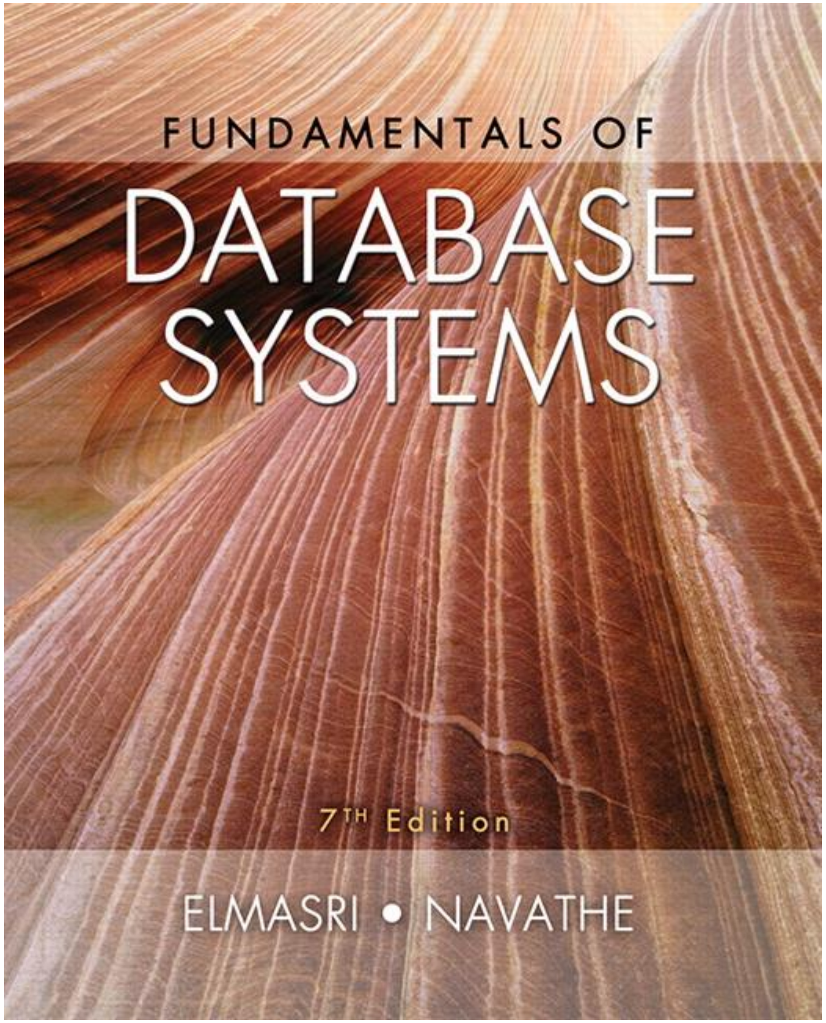
FUNDAMENTALS OF
DATABASE
SYSTEMS
7TH Edition
ELMASRI • NAVATHE

CHAPTER 15

Relational Database Design Algorithms and Further Dependencies

Chapter Outline

- 1. Further topics in Functional Dependencies
  - 1.1 Inference Rules for FDs
  - 1.2 Equivalence of Sets of FDs
  - 1.3 Minimal Sets of FDs
- 2. Properties of Relational Decompositions
- 3. Algorithms for Relational Database Schema Design
- 4. Nulls, Dangling Tuples, Alternative Relational Designs

Chapter Outline

- 5. Multivalued Dependencies and Fourth Normal Form – further discussion
- 6. Other Dependencies and Normal Forms
  - 6.1 Join Dependencies
  - 6.2 Inclusion Dependencies
  - 6.3 Dependencies based on Arithmetic Functions and Procedures
  - 6.2 Domain-Key Normal Form

FOR THIS LECTURE #10

We will only include:

SECTION 15.1 of Chapter 15:

- 15.1. Further topics in Functional Dependencies
  - 15.1.1 Inference Rules for FDs
  - 15.1.2 Equivalence of Sets of FDs
  - 15.1.3 Minimal Sets of FDs

Defining Functional Dependencies

- $X \rightarrow Y$ holds if whenever two tuples have the same value for X, they *must have* the same value for Y
  - For any two tuples t1 and t2 in any relation instance r(R): If t1[X]=t2[X], *then* t1[Y]=t2[Y]
- $X \rightarrow Y$ in R specifies a *constraint* on all relation instances r(R)
- Written as $X \rightarrow Y$; can be displayed graphically on a relation schema as in Figures in Chapter 14. (denoted by the arrow).
- FDs are derived from the real-world constraints on the attributes

1.1 Inference Rules for FDs (1)

- **Definition:** An FD $X \to Y$ is **inferred from** or **implied by** a set of dependencies F specified on R if $X \to Y$ holds in *every* legal relation state r of R; that is, whenever r satisfies all the dependencies in F, $X \to Y$ also holds in r.

- Given a set of FDs F, we can **infer** additional FDs that hold whenever the FDs in F hold

Inference Rules for FDs (2)

- Armstrong's inference rules:
  - IR1. (**Reflexive**) If Y *subset-of* X, then X → Y
  - IR2. (**Augmentation**) If X → Y, then XZ → YZ
    - (Notation: XZ stands for X U Z)
  - IR3. (**Transitive**) If X → Y and Y → Z, then X → Z
- IR1, IR2, IR3 form a **sound** and **complete** set of inference rules
  - Sound: Given a set F that holds in R, every dependency that can be inferred using the rules will hold in every state of R.
  - Complete: These rules and all other extended rules that hold can be applied to a set F of dependencies in R until no more dependencies can be inferred.

INFERENCE RULES -1

- **Armstrong's Axioms-1**

1. **Axiom of Reflexivity**
   - A set of attributes → A subset of the attributes

- **Example:**
  - `NRIC, Name → NRIC`
  - `StudentID, Name, Age → Name, Age`
  - `ABCD → ABC`
  - `ABCD → BCD`
  - `ABCD → AD`
  - `ABCD → ABCD`

INFERENCE RULES-2

- **Armstrong's Axioms-2**

2. Axiom of Augmentation

- If $A \rightarrow B$
- Then $AC \rightarrow BC$ *(for any C)*
- Also, $AA \rightarrow BA$ which means $A \rightarrow AB$

- **Example:** if NRIC → Name then
- NRIC, Age → Name, Age
- NRIC, Salary, Weight → Name, Salary, Weight
- NRIC, Address, Postal → Name, Address, Postal

INFERENCE RULES-3

- **Armstrong's Axioms-3**

3. **Axiom of Transitivity**
   - If $A \rightarrow B$ and $B \rightarrow C$
   - Then $A \rightarrow C$
- **Example:**
   - if NRIC → Address
   and Address → Postal
   - then NRIC → Postal

Inference Rules for FDs –contd.

- Some additional inference rules that are useful:
  - **Decomposition:** If X → YZ, then X → Y and X → Z
  - **Union:** If X → Y and X → Z, then X → YZ
  - **Psuedotransitivity:** If X → Y and WY → Z, then WX → Z

- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

Extended Armstrong Axioms -1

Armstrong's Axioms

1. **Reflexivity** $AB \rightarrow A$
2. **Augmentation** $A \rightarrow B \Rightarrow AC \rightarrow BC$
3. **Transitivity** $A \rightarrow B \;\&\; B \rightarrow C \Rightarrow A \rightarrow C$

- **Extended Armstrong's Axioms**

A. **Rule of Decomposition**

- If $A \rightarrow BC$
- Then $A \rightarrow B$ and $A \rightarrow C$

- **Proof:**

1. $A \rightarrow BC$ Given
2. $BC \rightarrow B$ Reflexivity $B \subseteq BC$

Hence, $A \rightarrow B$ Transitivity (1) and (2) ■

3. $BC \rightarrow C$ Reflexivity $C \subseteq BC$

Now, $A \rightarrow BC$ and $BC \rightarrow C$; Hence, $A \rightarrow C$, by Transitivity

Extended Armstrong Axioms -2

Armstrong's Axioms

1. **Reflexivity** $AB \rightarrow A$
2. **Augmentation** $A \rightarrow B \Rightarrow AC \rightarrow BC$
3. **Transitivity** $A \rightarrow B \;\&\; B \rightarrow C \Rightarrow A \rightarrow C$

- ## Extended Armstrong's Axioms

B. Rule of Union

- If $A \rightarrow B$ and $A \rightarrow C$
- Then $A \rightarrow BC$

- **Proof:**

1. $A \rightarrow B$ Given
2. $A \rightarrow C$ Given

Hence, $A \rightarrow AB$ Augmentation of (1) with A

3. $AB \rightarrow BC$ Augmentation of (2) with B

Now, $A \rightarrow AB$ and $AB \rightarrow BC$. Thus, $A \rightarrow BC$

Closure

Two types of CLOSURE

- **(1) Closure** of a set F of FDs is the set F^+ (which is called "Closure of F" or "F closure") of all FDs that can be inferred from F
- **(2) Closure** of a set of attributes X with respect to F is the set X^+ of all attributes (called Closure of X) that are functionally determined by X

- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

Algorithm to determine Closure

- **Algorithm 15.1.** Determining X^+, the Closure of X under F
- **Input:** A set F of FDs on a relation schema R, and a set of attributes X, which is a subset of R.

$X^+ := X$;

repeat

  old$X^+ := X^+$;

for each functional dependency $Y \rightarrow Z$ in F do

    if $X^+ \supseteq Y$ then $X^+ := X^+ \cup Z$;

until (X^+ = oldX^+);

Example of Closure (1)

- For example, consider the following relation schema about classes held at a university in a given academic year.

CLASS (Classid, Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity).

ASSUMPTION: The same instructor may offer the same course# in an assigned classroom on different days – these will get different sectionids. Different instructors may choose different texts for the same course.

- Let *F*, the set of functional dependencies for the above relation include the following f.d.s:

FD1: Classid $\rightarrow$ Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity;

FD2: Course# $\rightarrow$ Credit_hrs;

FD3: {Course#, Instr_name} $\rightarrow$ Text, Classroom;

FD4: Text $\rightarrow$ Publisher

FD5: Classroom $\rightarrow$ Capacity

SEMANTICS: *These f.d.s above represent the meaning of the individual attributes and the relationships among them and defines certain rules about the classes.*

Example of Closure (2)

- The closures of attributes or sets of attributes for some example sets:

  { Classid }$^+$ = { Classid , Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity } = CLASS

  { Course#}$^+$ = { Course#, Credit_hrs}

  { Course#, Instr_name }$^+$ = { Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity }

  Note that the above combination on LHS does NOT define Classid. (because same instructor can teach a course in the same classroom on different days). Hence it cannot be a key.

  Note that each closure above has an interpretation that is revealing about the attribute(s) on the left-hand-side.

  The closure of { Classid }$^+$ is the entire relation CLASS indicating that all attributes of the relation can be determined from Classid and hence it is a key.

  On that basis, Course# CANNOT be a key and even (Course#, Instr_name) cannot be the key.

Application of Inference Rules – Example 1

Armstrong's Axioms

1.	**Reflexivity**	$AB \to A$
2.	**Augmentation**	$A \to B \Rightarrow AC \to BC$
3.	**Transitivity**	$A \to B$ & $B \to C \Rightarrow A \to C$

Extended Armstrong's Axioms

A.	**Decomposition**	$A \to BC \Rightarrow A \to B$ & $A \to C$
B.	**Union**	$A \to B$ & $A \to C \Rightarrow A \to BC$

- ## Exercise #1
  - **Given:**
    1. $A \to B$
    2. $D \to C$
  - **Target (to show) :** $AD \to BC$
  - **Proof:**
    3. $AD \to BD$ Augmentation of (1) with D
    4. $AD \to B$ Decomposition of (3)
    5. $AD \to AC$ Augmentation (2) with A
    6. $AD \to C$ Decomposition of (5)
    7. $AD \to BC$ Union of (4),(6)

Type equation here.

Application of Inference Rules – Example 2

Armstrong's Axioms

1. **Reflexivity** $AB \rightarrow A$
2. **Augmentation** $A \rightarrow B \Rightarrow AC \rightarrow BC$
3. **Transitivity** $A \rightarrow B \;\&\; B \rightarrow C \Rightarrow A \rightarrow C$

Extended Armstrong's Axioms

A. **Decomposition** $A \rightarrow BC \Rightarrow A \rightarrow B \;\&\; A \rightarrow C$

B. **Union** $A \rightarrow B \;\&\; A \rightarrow C \Rightarrow A \rightarrow BC$

- ## Exercise #2
  - **Given:**
    1. $A \rightarrow C$
    2. $AC \rightarrow D$
    3. $AD \rightarrow B$
  - **Target (to show) :** $A \rightarrow B$
  - **Proof:**
    4. $A \rightarrow AC$ Augmentation (1) with A
    5. $A \rightarrow D$ Transitivity (4) and (2)
    6. $A \rightarrow AD$ Augmentation (5) with A
    7. $A \rightarrow B$ Transitivity (6) and (3)

1.2 Equivalence of Sets of FDs

- Two sets of FDs F and G are **equivalent** if:
  - Every FD in F can be inferred from G, and
  - Every FD in G can be inferred from F
  - Hence, F and G are equivalent if $F^+ = G^+$
- Definition (**Covers**):
  - F **covers** G if every FD in G can be inferred from F
    - (i.e., if G^+ *subset-of* F^+)
- F and G are equivalent if F covers G and G covers F

Functional Dependency Equivalence

- **Example**
  - **Example:** R(A, B, C, D, E)
    - F1 = {A → B, AB → C, D → AC, D → E}
    - F2 = {A → BC, D → AE}
    - ❖ Show F1 ≡ F2

1. Prove that F1 can be derived from F2
    - From F2, A → B and D → E can be derived easily using *decomposition rule*
    - $\{AB\}^+ = \{ABC\}$ so AB → C is implied by F2
    - $\{D\}^+ = \{DAE\}^+ = \{DAEBC\}$; so D → AC is implied by F2

∴ F1 can be derived from F2

Functional Dependency Equivalence

- **Example**
  - **Example:** `R(A, B, C, D, E)`
    - $F1 = \{A \rightarrow B,\ AB \rightarrow C,\ D \rightarrow AC,\ D \rightarrow E\}$
    - $F2 = \{A \rightarrow BC,\ D \rightarrow AE\}$
    - ❖ Show $F1 \equiv F2$

2. Prove that F2 can be derived from F1
    - $\{A\}^+ = \{ABC\}$ so $A \rightarrow BC$ is implied by F1
    - $\{D\}^+ = \{ABCDE\}$ so $D \rightarrow AE$ is implied by F1

using Closures

∴ F2 can be derived from F1

1.3 Finding Minimal Cover of F.D.s (1)

- Just as we applied inference rules to expand on a set *F* of FDs to arrive at F^+, its closure, it is possible to think in the opposite direction to see if we could shrink or reduce the set *F* to its *minimal form* so that the minimal set is still equivalent to the original set *F*.
- **Definition:** An attribute in a functional dependency (on LHS) is considered **extraneous attribute** if we can remove it without changing the closure of the set of dependencies. Formally, given F, the set of functional dependencies and a functional dependency $X \rightarrow A$ in *F* , attribute *set Y* is extraneous in *X* if *Y is a subset of X*, and *F* logically implies $(F - (X \rightarrow A) \cup \{ (X - Y) \rightarrow A \})$

Minimal Sets of FDs (2)

- A set of FDs is **minimal** if it satisfies the following conditions:
  1. Every dependency in F has a single attribute for its RHS.
  2. We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$, where Y is a proper-subset-of X and still have a set of dependencies that is equivalent to F.
  3. We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.

Minimal Sets of FDs (3)

- **Algorithm 15.2. Finding a Minimal Cover F for a Set of Functional Dependencies E**
  - **Input: A set of functional dependencies E.**

1. Se tF:=E.

breaking up the right hand side

2. Replace each functional dependency $X \rightarrow \{A1, A2, ..., An\}$ in F by the n functional dependencies $X \rightarrow A1$, $X \rightarrow A2$, ..., $X \rightarrow An$.
3. For each functional dependency $X \rightarrow A$ in F
   for each attribute B that is an element of X
   if $\{ \{F - \{X \rightarrow A\} \} \cup \{ (X - \{B\}) \rightarrow A\} \}$ is equivalent to F
   then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in F.

   (* The above constitutes a removal of the extraneous attribute B from X *)
4. For each remaining functional dependency $X \rightarrow A$ in F if $\{F - \{X \rightarrow A\} \}$ is equivalent to F, then remove $X \rightarrow A$ from F.

   (* The above constitutes a removal of the redundant dependency $X \rightarrow A$ from F *)

Computing the Minimal Sets of FDs (4)

We illustrate algorithm 15.2 with the following:
Let the given set of FDs be E : $\{B \rightarrow A, D \rightarrow A, AB \rightarrow D\}$.We have to find the minimum cover of E.

- All above dependencies are in canonical form; i.e., they have only one attribute on the RHS. So we have completed step 1

of Algorithm 15.2 and can proceed to step 2. In step 2 we need to determine if $AB \rightarrow D$ has any redundant attribute on the left-hand side; that is, can it be replaced by $B \rightarrow D$ or $A \rightarrow D$?

- Since B → A, by augmenting with B on both sides (IR2), we have $BB \rightarrow AB$, or $B \rightarrow AB$ (i). However, $AB \rightarrow D$ as given (ii).
- Hence by the transitive rule (IR3), we get from (i) and (ii), $B \rightarrow D$. Hence $AB \rightarrow D$ may be replaced by $B \rightarrow D$.
- We now have a set equivalent to original E , say E' : $\{B \rightarrow A, D \rightarrow A, B \rightarrow D\}$.

No further reduction is possible in step 2 since all FDs have a single attribute on the left-hand side.

- In step 3 we look for a redundant FD in E′. By using the transitive rule on $B \rightarrow D$ and $D \rightarrow A$, we derive $B \rightarrow A$. Hence $B \rightarrow A$ is redundant in E’ and can be eliminated.
- Hence the minimum cover of E is $\{B \rightarrow D, D \rightarrow A\}$.

Minimal Basis (Cover)

Conditions
1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes on LHS
4. No redundant FD

- **Example**
  - $F = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C\}$ $F_b = \{A \rightarrow B,\ B \rightarrow C\}$
  - Is F_b a minimal basis for F?
    1. $A \rightarrow C$ in F can be derived from F_b
       - F_b is F by removal of $A \rightarrow C$
    2. All FDs in F_b are non-trivial and decomposed
    3. For any FD in F_b, if we remove an attribute from left hand side, then the FD cannot be derived from F *(in fact, they have no left hand side!)*
    4. If any FD in F_b is removed, then some FD in F cannot be derived

    $\therefore$ F_b is a minimal basis (cover) for F

Minimal Basis (Cover)

Conditions

1. $F_b \equiv F$
2. Non-trivial and decomposed
3. No redundant attributes on LHS
4. No redundant FD

- **Example**
  - $F = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C\}$ $F_b = \{A \rightarrow B,\ AB \rightarrow C\}$
  - Is F_b a minimal basis for F?
    1. $B \rightarrow C$ in F can **NOT** be derived from F_b

$\therefore$ F_b is **NOT** equivalent to F and hence cannot be a minimal basis for F

Minimal Sets of FDs as a basis for design of relations

- Every set of FDs F has an equivalent minimal set
- There can be several equivalent minimal sets for a given set F of FDs.
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs. The process of Algorithm 15.2 is used until no further reduction is possible.
- *The synthesis approach to design a set of relations, which we will discuss in Lecture 12, starts with all possible F.D.s among a set of attributes that we wish to store, computes their minimal cover and then proceeds to design a set of relations in a specific Normal Form. We will discuss that in the algorithms of Ch.15.*

DESIGNING A SET OF RELATIONS

- **The Approach of Relational Synthesis (Bottom-up Design):**
  - Assumes that all possible functional dependencies are known. (impossible in real life)
  - 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 the the *set of relations* in a relational database are satisfactory (see Algorithm 15.3).

    companies use decomposition approach instead