

Course 3. Schema Refinement

Good designs / bad designs

Data represented by schemas generally have application-dependent constraints relating to attribute values.

Example: Consider the following *MovieList* relation:

<i>Title</i>	<i>Director</i>	<i>Cinema</i>	<i>Phone</i>	<i>Time</i>
The Hobbit	Jackson	Cinema City	441111	11:30
The Lord of the Rings 3	Jackson	Cinema City	441111	14:30
Adventures of Tintin	Spielberg	Odeon	442222	11:00
The Lord of the Rings 3	Jackson	Odeon	442222	14:00
War Horse	Spielberg	Odeon	442222	16:30

Figure 3.1 *MovieList* relation instance

Data stored by this relation respect the following constraints:

- Each movie has one director
- Each cinema has one phone number
- Each cinema screens one movie at a time

Common problems if a design is bad:

- **Insertion anomaly:** We can't store information about a new movie if the screening place and time are not known
- **Deletion anomaly:** If we delete all movies directed by Peter Jackson, we lose information about *Cinema City* cinema
- **Update anomaly:** If the phone number of a cinema changes, we have to be careful of inconsistent updates

Usually, we can refine a bad schema by *decomposing* it into multiple “good” ones.

<i>Movies</i>		<i>Cinema</i>	
<i>Title</i>	<i>Director</i>	<i>Cinema</i>	<i>Phone</i>
The Hobbit	Jackson	Cinema City	441111
The Lord of the Rings 3	Jackson	Cinema City	441111
Adventures of Tintin	Spielberg	Odeon	442222
War Horse	Spielberg	Odeon	442222

<i>Screens</i>		
<i>Cinema</i>	<i>Time</i>	<i>Title</i>
Cinema City	11:30	The Hobbit
Cinema City	14:30	The Lord of the Rings 3
Odeon	11:00	Adventures of Tintin
Odeon	14:00	The Lord of the Rings 3
Odeon	16:30	War Horse

Figure 3.2 Decomposition of *MovieList* relation

Refined schema allows:

- insertions of new movies without knowing their screening details
- deletions of movies without losing information about cinemas
- a single record to be updated to change a cinema's phone number

There are two main questions:

- How to determine whether a schema design is “good” or “bad”?
- How to transform a bad design into a good one?

The theory of *functional dependencies* provides a systematic approach to address these questions. This theory was introduced by E.F. Codd in: “A relational model for large shared data banks”, Com. of the ACM, 13(6), 1970, pp.377-387.

Functional dependencies

Functional dependencies (FDs) are constraints on schemas that specify that the values for a certain set of attributes determine unique values for another set of attributes

Let α and β denote subsets of attributes of a relational schema R . We use:

$$\alpha \rightarrow \beta$$

to denote that α functionally determines β (or β functionally depends on α)

In our previous example (*MovieList* relation) we can identify the following functional dependencies:

1. Title \rightarrow Director
2. Cinema \rightarrow Phone
3. Cinema, Time \rightarrow Title

Definition. The functional dependency $\alpha \rightarrow \beta$ holds on R if and only if for any relation instance of R , whenever two tuples t_1 and t_2 agree on the attributes α , they also agree on the attributes β .

That is,

$$\pi_{\alpha}(t_1) = \pi_{\alpha}(t_2) \Rightarrow \pi_{\beta}(t_1) = \pi_{\beta}(t_2)$$

Note: $\pi_{\alpha}(t)$ denote the projection of attributes α of tuple t

Let r be a relation instance of relation schema R

We are saying that r **satisfies FD** $\alpha \rightarrow \beta$ if for every pair of tuples t_1 and t_2 in r such that $\pi_{\alpha}(t_1) = \pi_{\alpha}(t_2)$, it is also true that $\pi_{\beta}(t_1) = \pi_{\beta}(t_2)$. Thus, a **FD f holds on R** if and only if for any relation instance r of R , r satisfies f

r is said to **violate** a FD f if r does not satisfy f . r is said to be a **legal instance of R** if r satisfies all FDs that hold on R .

A FD $\alpha \rightarrow \beta$ is a **trivial FD** if $\alpha \supseteq \beta$; otherwise it is a **non-trivial FD**

Example. Relation **Movie**(Title, Director, Composer). Let r be a legal relation instance of **Movie** as shown:

<i>Title</i>	<i>Director</i>	<i>Composer</i>
Schindler's List	Spielberg	Williams
Saving Private Ryan	Spielberg	Williams
North by Northwest	Hitchcock	Herrmann
Angela's Ashes	Parker	Williams
Vertigo	Hitchcock	Herrmann

Figure 3.3. **Movie** relation instance

The functional dependency $composer \rightarrow director$ does not hold on **Movie**. At the same time, r satisfies the FD $director \rightarrow composer$, but we cannot conclude that $director \rightarrow composer$ holds on **Movie**!

Conclusion: based on legal instances of R we can tell which FDs do not hold on R , but we can't deduce which non-trivial FDs hold.

Implication Problem: Given a set of functional dependencies F (that hold on R) and a functional dependency f , does f also hold on R ? F **logically implies (or implies)** f , denoted by $F \Rightarrow f$, if every relation instance r of R that satisfies the FDs F also satisfies the FD f

Example: In **MovieList**, we have the following predefined set of functional dependencies:

$$F = \{ \text{Title} \rightarrow \text{Director} \\ \text{Cinema} \rightarrow \text{Phone} \\ \text{Cinema, Time} \rightarrow \text{Title} \}$$

Does $\text{Cinema, Time} \rightarrow \text{Director}$ or $\text{Time} \rightarrow \text{Director}$ also hold?

Let F & G denote sets of functional dependencies, and f denote a functional dependency. More generally, $F \Rightarrow G$ if $F \Rightarrow g$ for each $g \in G$.

The **closure of F** (denoted by F^+) is the set of all functional dependencies implied by F ; that is,

$$F^+ = \{f \mid F \Rightarrow f\}$$

Two sets of functional dependencies, F and G , are **equivalent** (denoted by $F \equiv G$) if $F^+ = G^+$ (i.e., $F \Rightarrow G$ and $G \Rightarrow F$)

Axioms for Functional Dependencies

= a collection of formal rules used to derive a functional dependency from a set of functional dependencies

Armstrong's Axioms: Let $\alpha, \beta, \gamma \subseteq R$

Reflexivity: If $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$

Augmentation: If $\alpha \rightarrow \beta$, then $\alpha\gamma \rightarrow \beta\gamma$

Transitivity: If $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

Armstrong's Axioms are both *sound* and *complete*

Sound: Any derived FD is implied by F

Complete: All FDs in F^+ can be derived

Problem: Consider R(A, B, C, D, E) with 3 functional dependencies:

$$F = \{A \rightarrow C; B \rightarrow C; CD \rightarrow E\}.$$

Show that $F \Rightarrow AD \rightarrow E$

Solution:

1. $A \rightarrow C$ (given)
2. $AD \rightarrow CD$ (augmentation with (1))
3. $CD \rightarrow E$ (given)
4. $AD \rightarrow E$ (transitivity with (2) and (3))

Additional Inference Rules

Union: If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$, then $\alpha \rightarrow \beta\gamma$

Decomposition: If $\alpha \rightarrow \beta$, then $\alpha \rightarrow \beta'$ for any $\beta' \subseteq \beta$

Superkeys, Keys & Prime Attributes

A set of attributes α is a superkey of schema R (with FDs F) if $F \Rightarrow \alpha \rightarrow R$.

A set of attributes α is a key of schema R if

- (1) α is a superkey, and
- (2) no proper subset of α is a superkey
(i.e., for each $\beta \subset \alpha$, $\beta \rightarrow R \notin F^+$)

An attribute $A \in R$ is a prime attribute if A is contained in some key of R; otherwise, it is a nonprime attribute.

Example: Consider again **MovieList** (Title, Director, Cinema, Phone, Time) with functional dependencies set:

- (1) Cinema, Time \rightarrow Title
- (2) Cinema \rightarrow Phone
- (3) Title \rightarrow Director

{Cinema, Time} is the only key of **MovieList**.

Cinema and Time are the only prime attributes in **MovieList**.

Any superset of {Cinema; Time} in R is a superkey of **MovieList**.

Attribute Closure

Computing F^+ for a set of FDs F is not efficient as the size of F^+ could be exponentially large!

More efficient to compute the closure of a set of attributes

Let $\alpha \subseteq R$ and F be a set of FDs that hold on R . The closure of α (with respect to F), denoted by α^+ , is the set of attributes that are functionally determined by α with respect to F ; i.e.,

$$\alpha^+ = \{A \in R \mid F \Rightarrow \alpha \rightarrow A\}$$

Note that $F \Rightarrow \alpha \rightarrow \beta$ if and only if $\beta \subseteq \alpha^+$ (w.r.t. F)

Algorithm to compute attribute closure:

Input: α, F

Output: α^+ (w.r.t. F)

Compute a sequence of sets of attrs $\alpha_0, \alpha_1, \dots, \alpha_k, \alpha_{k+1}$ as follows:

$$\alpha_0 = \alpha$$

$$\alpha_{i+1} = \alpha_i \cup \gamma \text{ such that there is some FD}$$

$$\beta \rightarrow \gamma \in F \text{ and } \beta \subseteq \alpha_i$$

Terminate the computation once $\alpha_{k+1} = \alpha_k$ for some k . Return α_k

Problem: Given $F = \{A \rightarrow C; B \rightarrow C; CD \rightarrow E\}$, show that $F \Rightarrow AD \rightarrow E$.

Solution

<i>i</i>	α_i	<i>FD used</i>
0	AD	given input
1	ACD	$A \rightarrow C$
2	ACDE	$CD \rightarrow E$
3	ACDE	none

So $AD^+ = ACDE$. Since $E \in AD^+$, therefore $F \Rightarrow AD \rightarrow E$

Schema Decompositions

The **decomposition of schema R** is a set of schemas $\{R_1, R_2, \dots, R_n\}$ such that each $R_i \subseteq R$ and $R = \bigcup R_i$. If r is a relation of R , then r is decomposed into $\{r_1, r_2, \dots, r_n\}$, where each $r_i = \pi_{R_i}(r)$

Example:

{ (Cinema, Time, Title),
(Title, Director),
(Cinema, Phone)}

is a decomposition of: **MovieList**(Title, Director, Cinema, Phone, Time)

Properties of schema decomposition:

1. Decomposition must preserve information
 - o Data in original relation \equiv Data in decomposed relations

- Crucial for correctness!
- 2. Decomposition should preserve FDs
 - Functional dependencies in original schema \equiv functional dependencies in decomposed schemas
 - Facilitates checking of functional dependency violations

Lossless - Join Decomposition

It is important that a decomposition preserves information; i.e., we can reconstruct r from joining its projections $\{r_1, r_2, \dots, r_n\}$. Note that if $\{R_1, R_2, \dots, R_n\}$ is a decomposition of R , then for any relation r of R , it is always true that

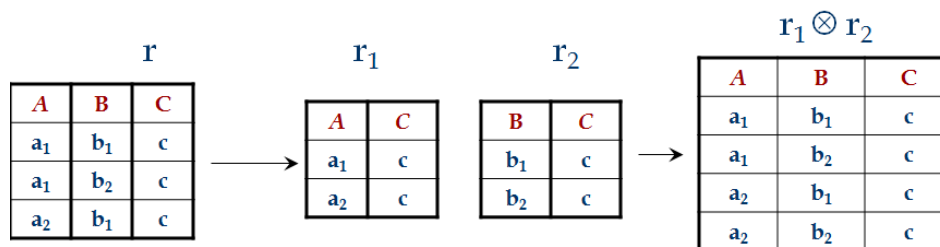
$$r \subseteq \pi_{R_1}(r) \otimes \pi_{R_2}(r) \otimes \dots \otimes \pi_{R_n}(r)$$

A decomposition of R (with functional dependencies set F) into $\{R_1, R_2, \dots, R_n\}$ is a lossless-join decomposition with respect to F if

$$\pi_{R_1}(r) \otimes \pi_{R_2}(r) \otimes \dots \otimes \pi_{R_n}(r) = r$$

for every relation r of R that satisfies F .

Example. Consider the decomposition of $R(A,B,C)$ into $\{R_1(AC), R_2(BC)\}$



Since $r \subset r_1 \otimes r_2$, the above decomposition is not lossless-join. A decomposition that is not lossless-join is called a lossy decomposition.

How to determine if $\{R_1, R_2\}$ is a lossless-join decomposition of R ?

Theorem: The decomposition of R (with FDs F) into $\{R_1, R_2\}$ is lossless with respect to F if and only if:

$$F \Rightarrow R_1 \cap R_2 \rightarrow R_1 \quad \text{or} \quad F \Rightarrow R_1 \cap R_2 \rightarrow R_2$$

How to decompose R into $\{R_1, R_2\}$ such that it is a lossless-join decomposition?

Corollary: If $\alpha \rightarrow \beta$ holds on R and $\alpha \cap \beta = \emptyset$, then the decomposition of R into $\{R - \beta, \alpha\beta\}$ is a lossless-join decomposition.

Example. Consider $R(A,B,C)$ with FDs $F = \{A \rightarrow B\}$

The decomposition $\{AB, AC\}$ has a lossless join since $AB \cap AC = A$ and $A \rightarrow AB$.

The decomposition $\{AB, BC\}$ is not lossless join w.r.t. F since $AB \cap BC = B$ and neither $B \rightarrow AB$ nor $B \rightarrow BC$ holds on R .

Theorem: If $\{R_1, R_2\}$ is a lossless-join decomposition of R , and if $\{R_{1,1}, R_{1,2}\}$ is a lossless-join decomposition of R_1 , then $\{R_{1,1}, R_{1,2}, R_2\}$ is a lossless-join decomposition of R .

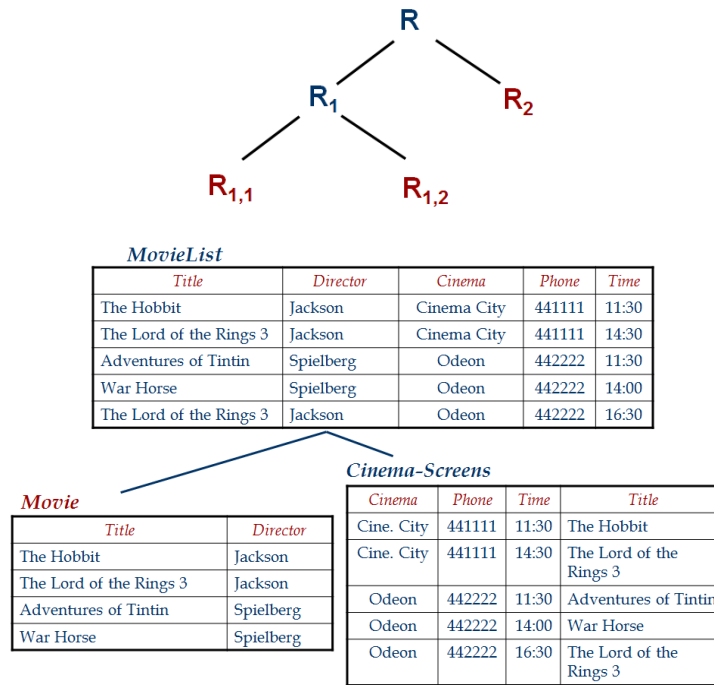


Figure 3.4 First step of decomposing **MovieList** relation, based on $Title \rightarrow Director$ functional dependency

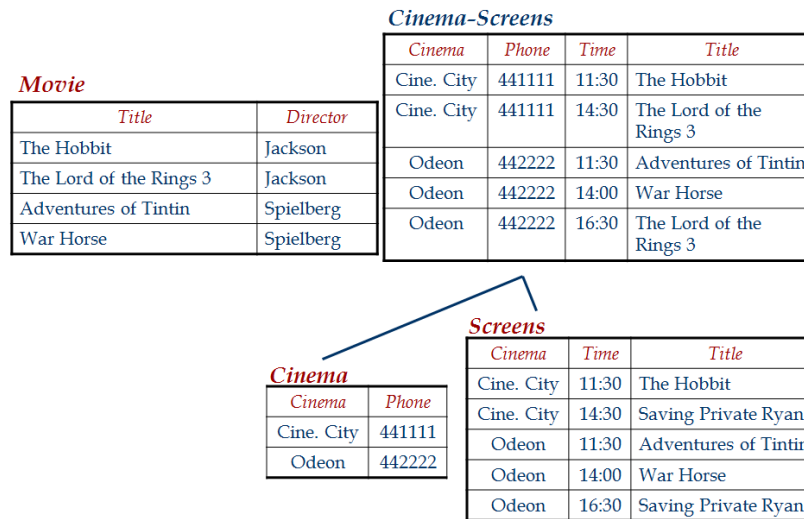


Figure 3.5 Last step of decomposing **MovieList** relation, based on $Cinema \rightarrow Phone$ functional dependency

Preserve Functional Dependencies

The projection of F on α (denote by F_α) is the set of FDs in F^+ that involves only attributes in α ; i.e., $F_\alpha = \{ \beta \rightarrow \gamma \in F^+ \mid \beta\gamma \subseteq \alpha \}$

Computing FD Projections:

Input: α, F

Output: F_α

result = \emptyset ;

for each $\beta \subseteq \alpha$ do

$T = \beta^+$ (w.r.t. F)

 result = result $\cup \{ \beta \rightarrow T \cap \alpha \}$

```
return result
```

If R is decomposed into X , Y and Z , and we enforce the FDs that hold on X , on Y and on Z , then all FDs that were given to hold on R must also hold.

Definition. The decomposition $\{R_1, R_2, \dots, R_n\}$ of R is dependency preserving if

$(F_{R_1} \cup F_{R_2} \cup \dots \cup F_{R_n})$ and F are equivalent, i.e.:

$(F_{R_1} \cup F_{R_2} \cup \dots \cup F_{R_n}) \Rightarrow F$ and $F \Rightarrow (F_{R_1} \cup F_{R_2} \cup \dots \cup F_{R_n})$