COMP 353 Databases

Design Theory for Relational Databases

Functional Dependencies

Schema Refinement (Decomposition)

Normal Forms

Functional Dependencies (FDs)

- A **functional dependency** (FD) is a kind of **constraint**
- Suppose **R** is a relation schema and **X,Y** $\subseteq$ **R**.

  A FD on **R** is a statement of the form **X** $\rightarrow$ **Y**, which asserts: "For every "legal/valid" instance **r** of **R**, and for all pairs of tuples t1 and t2 in **r**, if t1 and t2 **agree** on the values in **X**, then t1 and t2 **agree** also on the values in **Y**."

  In symbols: $\forall$ t1,t2 $\in$ **r**: t1[**X**] = t2[**X**] ➔ t1[**Y**] = t2[**Y**].
- We read **X** $\rightarrow$ **Y** as:

  X (functionally) determines Y (or Y is determined by X)
- We say that the FD: **X** $\rightarrow$ **Y** is *relevant* to **R** if **XUY** $\subseteq$ **R**.

Functional Dependencies

- Consider the relation schema:

  Star (name, SIN, street, city, postalCode, phone)
- Since we know the semantics of this relation from the design phase, we can answer the following question:
  - What are the functional dependencies on **Star**?
- Note that in general, FDs on a relation R may not be determined based on a given instance of R!

Functional Dependencies

- Consider the relation:

  Movie (title, year, length, filmType)
- What are the FD's on the **Movie** relation?

  We use the semantics of this relation to answer.

Keys

- The concept of FD generalizez the concept of key. How?
  - Let **X** $\subseteq$ **R**. Then **X** is a key of **R** iff **X** $\rightarrow$ **R**
- **X** is a **(candidate) key** of **R** (or a key, for short) if
  1. **X** $\rightarrow$ **R**. That is, attributes in **X** functionally determine **all** the attributes of **R**
  2. No proper subset of **X** is key, i.e., a candidate key must be **minimal**
- Is {title, year, filmType} a key for relation **Movie**?
- A set of attributes that contains a key is called a **superkey (that is, a superset of a key)**
  - Note that every key is a **superkey**, but not vice versa

Functional Dependencies

- **X** $\rightarrow$ **Y** is called a **functional dependency** because, in principle, there is a function that takes a list of values, one for each attribute in **X,** and returns at most one value (i.e., a *unique* value or no value at all) for the attributes in **Y**

Functional Dependencies

- Consider the relation:

  Movie (title, year, length, filmType, studioName, starName)
- What are the functional dependencies?

  {title, year} → length

  {title, year} → filmType

  {title, year} → studioName

  ➔ {title, year} → {length, filmType, studioName}

  Note: {title, year} → starName does not hold
- What is the key of the **Movie** relation?

Trivial FD's

- An FD **X** → **Y** is said to be **trivial** if **Y** ⊆ **X**.
  - For example: **{title, year}** → **title** is a trivial FD
- Otherwise, the FD is called **nontrivial**
  - For example: **{title, year}** → **length** is a nontrivial FD

Functional Dependencies

- Why are we interested in functional dependencies?

Redundancy Problem

- Redundancy – a "piece" of information is unnecessarily repeated in different tuples in a relation
- Recall that ***redundancy*** is the main source of problems:
  - Storage waste
    - Some information stored repeatedly
  - Update anomalies
    - If a copy of such information is updated, an inconsistency may arise unless all its copies are updated
  - Insertion anomalies
    - Unless we allow nulls, it may not be possible to store some information unless we have all the information to store
  - Deletion anomalies
    - Deleting some information may results in loosing some other information (which we don't want to loose)

Is this a good design for relation R?

Name	SSN	Phone
Fred	123-321-99	(201) 555-1234
Fred	123-321-99	(201) 572-4312
Joe	909-438-44	(908) 464-0028
Mary	938-401-54	(201) 555-1234

The only FD on R is: **SSN➔Name**

Therefore, the only key of R is: **{SSN, Phone}**

What about this design, replacing R with R1 and R2?

R1

SSN	Name
123-321-99	Fred
909-438-44	Joe
938-401-54	Mary

R2

SSN	Phone
123-321-99	(201) 555-1234
123-321-99	(201) 572- 4312
909-438-44	(908) 464-0028
938-401-54	(201) 555-1234

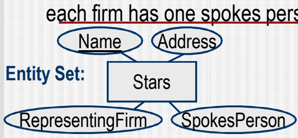
D = {R1(<u>SSN</u>, Name), R2(<u>SSN, Phone</u>)}

FD's on R1 and R2?

Another Example

- Suppose each star has a representing *firm* and each firm has one spokes person)

Entity Set:



Relation Schema:

Star (Name, Address, RepresentingFirm,SpokesPerson)

RepresentingFirm → SpokesPerson

Relation Instance:

Name	Address	RepresentingFirm	SpokesPerson
Carrie Fisher	123 Maple	Star One	Joe Smith
Harrison Ford	789 Palm rd.	Star One	Joe Smith
Mark Hamill	456 Oak rd.	Movies & Co	Mary Johns

Redundancy Problem

What is the role of FDs in detecting redundancy?

- Consider the relation scheme **R(A, B, C)**
  - Suppose no (nontrivial) FD holds on **R**
    - There is no redundancy in any instance **r** of **R**.
  - Now suppose FD: **A → B** holds on **R**
    - If several tuples have the same **A** value ➔ they must all have the same **B** value; otherwise this FD is violated
- Presence of some FDs in a relation suggests possibility of redundancy

Implications of FDs and Reasoning

- Consider relation **R(A, B,C)** with the set of FDs:
  F = {**A→B, B→C**}
- We can deduce from F that **A→C** also holds on **R.**
  How? Apply the definition…
- To detect possible data redundancy, is it necessary to consider "all" the FDs (implicit and explicit)?
  - As shown above, there might be some additional hidden (nontrivial) FDs "**implied**" by a given set of FD's

Implications of FDs

- Defn: If a relation instance **r** satisfies every FD in a given set **F** of FD's, then we say that **r** satisfies **F**.
  - In this case, we also say that **r** is a *legal/valid instance*.
- Given <**R**,**F**>, we say that **F implies** a FD **X → Y,** if *every* instance **r** of **R** that satisfies **F** also satisfies **X → Y.**
  Formally, we express this as: **F ⊨ X → Y**.
  We may also say that **X → Y** follows from **F**.
- To show **F ⊭ X → Y,** we may give a counter-example, i.e., an instance **r** of **R** that satisfies **F** but not **X → Y**.

FDs Implication (Cont'd)

- Consider **R**(A_1,A_2,A_3,A_4,A_5) with FDs:
  F = { $A_1 \to A_2$, $A_2 \to A_3$, $A_2A_3 \to A_4$, $A_2A_3A_4 \to A_5$ }
  Prove that **F** $\not\models A_5 \to A_1$
  Solution method: Provide a counter-example; give a relation instance **r** of R that satisfies every FD in **F** but not $A_5 \to A_1$
  A desired instance **r** of **R:**

	A_1	A_2	A_3	A_4	A_5
t1:	0	1	1	1	1
t2:	1	1	1	1	1

Closure of a set F of FDs

- Defn: The **closure of F**, denoted by **F⁺**, is the set of every FD: **X→ Y** that is implied by **F**.
- How can we determine **F⁺**?
  - Clearly, **F⁺** includes **F** and possibly some more FDs
  - To answer the question we need to *reason* about FDs

Equivalence of two sets of FD's

- Let **R** be a relation schema, and **S, T** be sets of FDs on **R**.
- Defn: we say **S** ***covers*** **T** (**S** $\models$**T**) if for every instance **r** of **R**, whenever **r** satisfies (every FD in) **S**, **r** also satisfies **T**.
- Defn: **T** and **S** are ***equivalent*** (**S** $\equiv$ **T**) iff **S** $\models$**T** and **T** $\models$ **S**.
- Note: **F** and **F⁺** are equivalent.

Example: Suppose **R** = {A,B,C}, and

S = {A $\rightarrow$ B, B $\rightarrow$ C, A $\rightarrow$ C}

T = {A $\rightarrow$ B, B $\rightarrow$ C}

We can show that **S** $\equiv$ **T**.

Armstrong's Axioms [1974]

- **R** is a relation schema, and **X**, **Y**, **Z** are subsets of **R**.
- **Reflexivity**
  - If **Y** $\subseteq$ **X**, then **X** $\rightarrow$ **Y** (trivial FDs)
- **Augmentation**
  - If **X** $\rightarrow$ **Y**, then **XZ** $\rightarrow$ **YZ**, for every **Z**
- **Transitivity**
  - If **X** $\rightarrow$ **Y** and **Y** $\rightarrow$ **Z**, then **X** $\rightarrow$ **Z**
- These are **sound** and **complete** inference rules for FDs

Additional rules / axioms

Other useful rules that follow from Armstrong Axioms:
Suppose **X, Y, Z,** and **W** are sets of attributes.

- **Union (Combining) Rule**
  - If **X** $\rightarrow$ **Y** and **X** $\rightarrow$ **Z**, then **X** $\rightarrow$ **YZ**
- **Decomposition (Splitting) Rule**
  - If **X** $\rightarrow$ **YZ**, then **X** $\rightarrow$ **Y** and **X** $\rightarrow$ **Z**
- **Pseudotransitivity Rule**
  - If **X** $\rightarrow$ **Y** and **WY** $\rightarrow$ **Z**, then **XW** $\rightarrow$ **Z**

Example – Discovering hidden FD's

- Consider **R** = {**A**, **B**, **C**, **G**, **H**, **I**} with the FDs:
  F = { **A** $\rightarrow$ **B**, **A** $\rightarrow$ **C**, **CG** $\rightarrow$ **H**, **CG** $\rightarrow$ **I**, **B** $\rightarrow$ **H** }
- Using Armstrong's rules, we can derive more FDs
  - Since **A** $\rightarrow$ **B** and **B** $\rightarrow$ **H**, then **A** $\rightarrow$ **H**, by transitivity
  - Since **CG** $\rightarrow$ **H** and **CG** $\rightarrow$ **I**, then **CG** $\rightarrow$ **HI**, by union
  - Since **A** $\rightarrow$ **C** then **AG** $\rightarrow$ **CG**, by augmentation
    Now, since **AG** $\rightarrow$ **CG** and **CG** $\rightarrow$ **I**, then **AG** $\rightarrow$ **I**, by transitivity (and in a similar way, we get **F** $\models$ **AG** $\rightarrow$ **H**)
  - Many trivial dependencies can be derived(!) by augmentation

Implication Problem

- Given a set **F** of FDs, does **X** $\rightarrow$ **Y** follow from **F**?
- In other words: is **X** $\rightarrow$ **Y** in the closure of **F**?

  (In symbols, does **F** $\models$ **X** $\rightarrow$ **Y** hold, or is

  X $\rightarrow$ **Y** $\in$ **F⁺** true?
- How to answer this question?
  - Compute the closure of **F** & check if it includes **X** $\rightarrow$ **Y**
- What is the problem with this approach?
  - Computing **F⁺** is *expensive*! Is there a better solution?

Closure of a Set of Attributes

- Given <**R, F**>; Let **X** $\subseteq$ **R.**
  The closure of **X** under **F** is the set of all attributes **Y** in **R** that are determined by **X**. This yields **X** $\rightarrow$ **Y**, i.e., every valid instance of **R** (that satisfies **F**) also satisfies **X** $\rightarrow$ **Y**
  - We denote the closure of a set of attributes **X** under **F** by X^+_F
    - When **F** is known , we simply write **X⁺** (and omit **F**)
    - Closure of **{A1, A2, ..., A_n}** is denoted **{A1, A2, ..., A_n}⁺**
  - Note that **X** $\subseteq$ **X⁺**, for any set **X** of attributes (because **X** $\rightarrow$ **X**)

Computing the Closure of Attributes

- Given a set **F** of FD's and a set **X** of attributes, how to compute the closure of **X** w.r.t. **F**?
  - Starting with set $X^+=X$, we repeatedly expand X^+ by adding the RHS Z for every FD: $W \rightarrow Z$ in **F**, if the LHD **W** is already in X^+.
  - This process terminates when X^+ could not be expanded further.
  - This process is expressed as an algorithm in the next slide.

25

An Algorithm to Compute X^+ under F

$X^+ \leftarrow X$ (initialization step)

repeat

for each FD $W \rightarrow Z$ in **F** *do*:

if $W \subseteq X^+$ *then*

$X^+ \leftarrow X^+ \cup Z$ // add **Z** to the result

until X^+ does not change

Complexity? In the worst case, how many times the "repeat" statement may be executed?

26

Examples

- Consider a relation schema **R** = { **A**, **B**, **C**, **D**, **E**, **H** } with the FD's **F** = { $AB \rightarrow C$, $BC \rightarrow AD$, $D \rightarrow E$, $CH \rightarrow B$ }
- Suppose **X**={**A**,**B**}. Compute X^+
- Execution result at each iteration:
  - Initially, X^+ = {A, B}
  - Using $AB \rightarrow C$, we get X^+ = {A, B, C}
  - Using $BC \rightarrow AD$, we get X^+ = {A, B, C, D}
  - Using $D \rightarrow E$, we get X^+ = {A, B, C,D, E}
  - No more change to X^+ is possible.

    ➔ $X^+ = \{A, B\}^+ = \{A, B, C, D, E\}$
- Does the order in which FD's appear in **F** affects the computation?

27

Implication Problem Revisited

- Given a set of FD's **F**, does an FD: $X \rightarrow Y$ follow from **F**?
  - That is, is FD $X \rightarrow Y$ in F^+?
- To answer this, we can compute X^+ under **F**, and check if **Y** is in X^+ or not
  - If yes, then the answer is positive! ($F \models X \rightarrow Y$ ☺)
  - Otherwise, it is negative ($F \not\models X \rightarrow Y$ ☹)

28

Example

- Consider <R,F> where **R** = { **A**, **B**, **C**, **D**, **E**, **H** } and **F** = { $AB \rightarrow C$, $BC \rightarrow AD$, $D \rightarrow E$, $CH \rightarrow B$ }
- Does $AB \rightarrow D$ follow from **F**?
- Two steps:
  1. Compute $\{A, B\}^+ = \{A, B, C, D, E\}$
  2. Check if $D \in \{A,B\}^+$
  - So, here we conclude that $AB \rightarrow D$ is implied by **F**

29

Example

- Consider a relation schema **R** = { **A**, **B**, **C**, **D**, **E**, **H** } with FDs: **F** = { $AB \rightarrow C$, $BC \rightarrow AD$, $D \rightarrow E$, $CH \rightarrow B$ }
- True/False: Does $D \rightarrow A$ follow from **F**?
- Two steps:
  1. Compute $X^+ = \{D\}^+ = \{D, E\}$
  2. Check if $A \in X^+$
  - Since $A \notin \{D, E\}$, the answer is NO, i.e., $F \not\models D \rightarrow A$

30

Closures and Keys

- Consider a case where X^+ includes **all** the attributes of a relation **R**
  - ➔ Clearly, **X** is a (super) **key** of **R**
- ➔ To check if **X** is a candidate key of **R**, we should check 2 things:
  1. If X^+ is a superkey **R,** i.e., when $X^+ = R$, and
  2. If **no** proper subset of **X** is a key, i.e., $\forall A \in X$: $(X - \{A\})^+ \neq R$
- To find the keys of a relation, we can use the algorithm on slide 26
- This would be exponential in the number of attributes! Can do better?
- Knowledge about keys is essential to understand “Normal forms.”

31