Introduction to Database Systems

CSE 444

Lecture #8
Jan 29 2001

Announcements

- **Mid Term Syllabus**
  - **Material in lectures**
  - **Textbook**
    - **Chapter 1.1, Chapter 2 (except 2.1 and ODL), Chapter 3 (except 3.2, 3.8), Chapter 4.1, 4.5, 4.6, Chapter 5 (except 5.10), Chapter 6.1, 6.2, 7.1, 7.3**
  - **Mid Term will be in class closed book exam**
- **Key Focus: Schema Design and SQL**
  - **Yes/No; Short answer; Multi-part question**
- **Extended Office Hours (this week only):**
  - **Surajit M,W: 4.50-5.50**
  - **Yana Thu 4.30-5.30 and usual hours**

2

Functional Dependencies

Reading: Chapter 3.5, 3.6, 3.7

Mapping ER Diagram to Relations

- Entity mapped to a relation
- Many-many relationship mapped to a relation
- Some columns will be NULL-able
- May be possible to combine relations
  - Many-to-one relationships
  - Danger of redundancy: delete/update inconsistencies

4

Example

- Drinker(name, addr) and Favorite(drinker, beer) combined as:
  - Drinker_info(name, addr, choice_beer)
- Can you combine Drinker(name, addr) and Likes(drinker, beer)?
  - Combination introduces redundancy

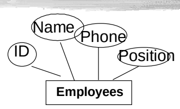
5

Need for Schema Refinement

- Resulting schema may have redundancy
  - Inaccurate E-R modeling
  - Inappropriate combination of relations during mapping
- Functional Dependency provides a mathematical tool to detect redundancy
- Decomposition to ensure that schema does not suffer from redundancy

6

Example



EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E1847	John	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	lawyer

Functional Dependencies

Definition:

If two tuples agree on the attributes

$A_1, A_2, \ldots A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots B_m$

Formally: $A_1, A_2, \ldots A_n \longrightarrow B_1, B_2, \ldots B_m$

Motivating example for the study of functional dependencies:

Name	Social Security Number	Phone Number

A Property of Functional Dependency

Splitting/Combining Lemma

$A_1, A_2, \ldots A_n \longrightarrow B_1, B_2, \ldots B_m$ Is equivalent to

$A_1, A_2, \ldots A_n \longrightarrow B_1$

$A_1, A_2, \ldots A_n \longrightarrow B_2$

…

$A_1, A_2, \ldots A_n \longrightarrow B_m$

Inference of Implied FD

- Important for
  - Schema Redesign
  - Identifying key
- Armstrong's axioms
  - Reflexivity If Y subset-of X, then X -> Y
  - Augmentation A->B implies AX -> BX
  - Transitivity A->B and B->C implies A->C
- A sound and complete inference system to obtain all implied functional dependencies

Example

$A \longrightarrow B$

$B\,C \longrightarrow D$

Example

$A\ B \longrightarrow C$

$A\ D \longrightarrow E$

$B \longrightarrow D$

$A\ F \longrightarrow B$

Lets infer a few FD-s…

Closure of a set of Attributes

Given a set of attributes A = {A1, …, An} and a set of dependencies S.

Closure(A) is the set of all attributes *B* such that: any relation which satisfies S also satisfies:

A1, …, An -> B

+

1. Closure(A) is a subset of all FDs implied
2. For a relation R(A) and a key B of R(A): What is the relationship between closure (B) and A?

13

Keys and SuperKeys

Product: name → price, manufacturer
Person: ssn → name, age
Company: name → stock price, president

Key of a relation is a set of attributes that:

- functionally determines all the attributes of the relation
- none of its subsets determines all the attributes.

Superkey: a set of attributes that contains a key

14

Example

⌘Drinkers (<u>name</u>, addr, <u>likesbeer</u>, manuf, favbeer)

⌘What are the
- Keys?
- Superkeys?

15

Closure Algorithm

Start with X={A1, …, An}.

Repeat until X doesn't change **do**:

if $B_1, B_2, \ldots B_n \longrightarrow C$ is in S, **and**
$B_1, B_2, \ldots B_n$ are all in X, **and**
C is not in X

then

add C to X.

16

Example

A B ⟶ C
A D ⟶ E
B ⟶ D
A F ⟶ B

Closure of {A,B}: X = {A, B, }

Closure of {A, F}: X = {A, F, }

17

Example

⌘AB -> C, C->D, D->A

⌘Any "interesting" consequences?

18

Why Is the Algorithm Correct ?

- Show the following by induction:
  - For every *B* in *X*:
    - $A1, \ldots, An \to B$
- Initially X = {A1, …, An} -- holds
- Induction step: B1, …, Bm in X
  - Implies $A1, \ldots, An \to B1, \ldots, Bm$
  - We also have $B1, \ldots, Bm \to C$
  - By transitivity we have $A1, \ldots, An \to C$
- This shows that the algorithm is *sound*; need to show it is *complete*

19

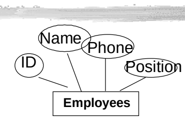
Relational Schema Design

Main idea:

- Start with initial relational schema
- Find out implied FD-s
- Use them to design a better relational schema

20

Example



EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E1847	John	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	lawyer

21

Example (Contd)

- What if:
  - All current *salespersons* resign
  - Can I update *Smith's* phone?
  - Can I add a salesperson *Roy* with phone *6923*?

22

Boyce-Codd Normal Form

A relation R is in BCNF if and only if:

Whenever there is a nontrivial dependency $A_1, A_2, \ldots A_n \to B$ for R, it is the case that $\{A_1, A_2, \ldots A_n\}$ a super-key for R.

In English (though a bit vague):

Whenever a set of attributes of *R* is determining another attribute, should determine ***all*** the attributes of *R*.

23

What is interesting about BCNF?

- No redundancy due to FD-s
- No update anomalies
  - Only one (unique) occurrence of a fact is updated
- No deletion anomalies

24

Relational Schema Design

Recall set attributes (persons with several phones):

Name	SSN	Phone Number
Fred	123-321-99	(201) 555-1234
Fred	123-321-99	(206) 572-4312
Joe	909-438-44	(908) 464-0028
Joe	909-438-44	(212) 555-4000

Problems:

- redundancy
- update anomalies
- deletion anomalies

Note: SSN is NOT a key here

25

Decompositions in General

Let R be a relation with attributes $A_1, A_2, \ldots A_n$

Create two relations *R1* and *R2* with attributes

$B_1, B_2, \ldots B_m$ $\quad$ $C_1, C_2, \ldots C_l$

Such that:

$B_1, B_2, \ldots B_m \cup C_1, C_2, \ldots C_l = A_1, A_2, \ldots A_n$

And

-- *R1* is the projection of R on $B_1, B_2, \ldots B_m$

-- *R2* is the projection of R on $C_1, C_2, \ldots C_l$

26

Incorrect Decomposition

⌘Sometimes it is incorrect:

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
DoubleClick	29.99	Camera

Decompose on : **Name, Category** and **Price, Category**

27

Incorrect Decomposition

Name	Category
Gizmo	Gadget
OneClick	Camera
DoubleClick	Camera

Price	Category
19.99	Gadget
24.99	Camera
29.99	Camera

When we put it back:

Cannot recover information

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
OneClick	29.99	Camera
DoubleClick	24.99	Camera
DoubleClick	29.99	Camera

28

Example

Name	SSN	Phone Number
Fred	123-321-99	(201) 555-1234
Fred	123-321-99	(206) 572-4312
Joe	909-438-44	(908) 464-0028
Joe	909-438-44	(212) 555-4000

What are the dependencies?

What are the keys?

Is it in BCNF?

29

And Now?

SSN	Name
123-321-99	Fred
909-438-44	Joe

SSN	Phone Number
123-321-99	(201) 555-1234
123-321-99	(206) 572-4312
909-438-44	(908) 464-0028
909-438-44	(212) 555-4000

30

What About This?

Name	Price	Category
Gizmo	$19.99	gadgets

Question:

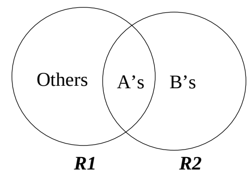
Find an example of a 2-attribute relation that is not in BCNF.

31

Decomposition Strategy for BCNF

Find a FD that violates the BCNF condition:

$A_1, A_2, \ldots A_n \longrightarrow B_1, B_2 \ldots B_m$



32

Example

- Movie (title, year, studio,president, pres_addr)
  - Title, year -> studio, studio -> president, president->pres_addr
- studio -> president, pres_addr
- Decompose: Studio(studio, president, pres_addr), Movie (title, year, studio)
- Decompose again?

33

Projecting FD

- Given F over R, what is the FD that must hold over S, where S is obtained by decomposition?
- Compute closure(X) for each subset X of S
- X-> B holds in S if
  - B in S
  - B in closure(X)
  - B not in X
- See Examples 3.39 and 3.40 in text

34

Decomposition Based on BCNF is Information Preserving

Attributes A, B, C. FD: $A \rightarrow C$

Relations R1[A,B] R2[A,C]

Tuples in R1: (a,b), (a,b')

Tuples in R2: (a,c), (a,c')

Tuples in the join of R1 and R2: (a,b,c), (a,b,c'), (a,b',c), (a,b',c')

Can (a,b,c') be a bogus tuple? What about (a,b',c') ?

35

Problem with BCNF

- Street, city -> zip, zip -> city
- Keys?
- Consider (Street, city) and (city, zip)
  - How to check street, city -> zip?
- 3NF
  - Allow FD if LHS is part of a key (prime)

36

Problems with Decompositions

- There are three potential problems to consider:
  - Some queries become more expensive.
    - e.g., find employee and department names
  - Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!
  - Checking some dependencies may require joining the instances of the decomposed relations.
    - BCNF decomposition example
- Tradeoff: Must consider these issues vs. redundancy.

Summary of Schema Refinement

- If a relation is in BCNF, it is free of redundancies that can be detected using FDs.
- If a relation is not in BCNF, we can try to decompose it into a collection of BCNF relations:
  - Lossless-join, *dependency preserving* decomposition into BCNF is not always possible
  - Lossless-join decomposition into BCNF ***is*** always possible
  - Lossless-join, dependency preserving decomposition into 3NF ***is*** always possible
  - Decompositions should be carried out and/or re-examined while keeping *performance requirements* in mind.
  - Various decompositions of a single schema are possible.