Mod 9: Relational Database Design

Normalization Theory

Designing relations

- What is the right structure for tables?
- How many tables ?
  - Have more smaller or a few large tables ?
- Criteria ?
  - Understandability ?
    - Driven by meaning of data
  - Relatedness of data within a table ?
  - Minimize redundancy ?
  - Enforceability of constraints ?
  - Performance ?
- Theoretical foundations ?

Not "good" relation

- Consider :

  std (rno, name, hno, sdept, head, warden, crdept, crno, grade)
- Has redundancy
- Leading to inconsistencies while making updates : called update 'anomalies'
- Develop theory : achieve separation of data and enforcement of constraints in easy way
- Define good 'normalized' forms for relations based on this theory

First Normal Form (1NF)

- 1NF : attributes are atomic
  - No composite values, no arrays, …
- Non-atomic values
  - complicates storage
  - query language becomes complex
  - Object-orientation is a possible answer

Goal — Devise a Theory

- Decide whether a relation R is in "good" form.
- If no, decompose R into $\{R_1, R_2, ..., R_n\}$ such that
  - each R_i is in good form
  - the decomposition is lossless-join
- Our theory is based on:
  - functional dependencies (FDs)
  - multivalued dependencies (MVDs)
    - They capture 'constraints'

Functional Dependencies (FD)

- Notation : $\alpha \rightarrow \beta$
- Means : attributes α determines uniquely value for β
  - If 2 tuples have same α, they must also have same β
- FD is a generalization of the notion of a *key*
- K is a key (or, superkey) for R if $K \rightarrow R$

- Consider :

  Std (rno, sname, dept, dname, head, hno, phone, cpi)

Some FDs here : rno $\rightarrow$ dept ….

Functional Dependencies (Cont.)

- FDs need to be explicitly stated : application specific
- Database D satisfies given F (set of FDs)
- Set F holds on D

  (note : D varies with time, so F hold for all D's)

Properties of Functional Dependencies

- Armstrong's Axioms:
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ (reflexivity; trivial FD)
  - if $\alpha \rightarrow \beta$, then $\gamma\alpha \rightarrow \gamma\beta$ (augmentation)
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ (transitivity)
- These rules are
  - sound (generate FDs that actually hold), and
  - complete (generate all FDs that hold).

Closure of a Set of FDs

- Given *F* : other FDs are logically implied by *F*.
- F^+
  - set of all functional dependencies logically implied by *F*
  - called closure of *F*

More properties of Fds

- given $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$, then $\alpha \rightarrow \beta\gamma$ (union)
- given $\alpha \rightarrow \beta\gamma$, then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ (decomposition)
- If $\alpha \rightarrow \beta$ and $\gamma\beta \rightarrow \delta$, then $\alpha\gamma \rightarrow \delta$ (pseudotransitivity)

The above rules can be inferred from Armstrong's axioms.

Example

- $R = (A, B, C, G, H, I)$

  $F = \{ A \rightarrow B \quad A \rightarrow C$

  $CG \rightarrow H \quad CG \rightarrow I$

  $B \rightarrow H\}$
- some members of F^+
  - $A \rightarrow H$ by transitivity from $A \rightarrow B$ *and* $B \rightarrow H$
  - $CG \rightarrow HI$
    - $CG \rightarrow I$ gives $CG \rightarrow CGI$,

      and use $CG \rightarrow H$ to infer $CGI \rightarrow HI$,

      and then transitivity

Boyce-Codd Normal Form

R is in BCNF wrt *F* if for all non-trivial FDs
$\alpha \rightarrow \beta$ in F+

- α is a (super)key for *R*

Consider :

emp_dept (ID, name, salary, dept_name, building, budget)

This is not in BCNF
because *dept_name→ building, budget* holds,
but *dept_name* is not a superkey (not a GOOD relation)

What is required to get it into BCNF ? What gets achieved ?

Decomposing a Schema into BCNF

- Given R and a FD $\alpha \rightarrow \beta$ which violates BCNF
  decompose R into:
  - R1 with attributes ($\alpha \cup \beta$)
  - R2 with attributes (R - (β - α)) (keep α in both)
- In our example,
  emp (*ID, name, salary, dept_name, building, budget*)
  we have *dept_name* → *building, budget*
  then *emp* is replaced by
  - emp1 (*dept_name, building, budget*)
  - emp2 (*ID, name, salary, dept_name*)

Lossless-join Decomposition

- Decomposition using $\prod$ and re-construction using $\bowtie$
  $$r = \prod_{R1}(r) \bowtie \prod_{R2}(r)$$
- A decomposition of R into R_1 and R_2 is lossless join if at least one of the following dependencies is in F^+:
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

- Consider addr(city, street, PIN)
  PIN → city
  city, street → PIN
  (key is city + street)
- addr not in BCNF but difficult to decompose (2nd FD will span across 2 tables then !)

BCNF and Dependency Preservation

- FDs are Constraints
  - Easy to enforce using keys
- goal *: dependency preserving* decomposition in BCNF
- achieving both BCNF decomposition and dependency preservation not always possible
- Define a weaker normal form, called *third normal form (3NF)*.

Third Normal Form

- A relation R is in third normal form (3NF) if for all non-trivial FDs $\alpha \rightarrow \beta$ in F^+
  at least one of the following holds:
  - α is a (super)key
  - β attributes are contained in a candidate key
- A BCNF relation is in 3NF
- 3NF provides a minimal relaxation of BCNF to ensure dependency preservation.

- Bank example:
  C - cust, B: branch, E: emp

  Given R(C,B,E)
  FDs present : E → B CB → E
  Candidate keys: CB, CE but not EB since E may serve many customers

-- not in BCNF, but in 3NF since B is part of candidate key

Also : addr(<u>city, street</u>, pin) is in 3NF
for pin → city, city is part of key

Multivalued Dependencies

- Suppose we record names of children, and phone numbers for employees in table:

  emp_info (ID, child, phone)

  - Example data:
    (99999, David, 1234)
    (99999, David, 4321)
    (99999, William, 1234)
    (99999, William, 4321)
- This relation is in BCNF – there are no non-trivial FDs
- Has redundancies; Can lead to insertion anomalies
- Better to decompose it into 2 relations

  emp1 (ID, child) emp2(ID, phone)
- Need new type of constraint and normal form

Multi-valued Dependency (MVD)

- Capture 1-to-many relationships :

  $ID \twoheadrightarrow child\_name$

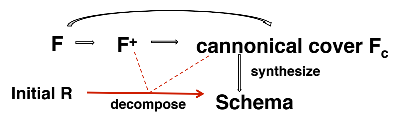
  $ID \twoheadrightarrow phone\_number$
- MVD captures the fact that two sets of data within a table are independent of each other.

Fourth Normal Form

- A relation R is in 4NF if for all non-trivial multivalued dependencies $\alpha \twoheadrightarrow \beta$, :
  - α is a (super)key for R *(implying FD* $\alpha \rightarrow \beta$)
- Our previous emp1, emp2 decompositions are in 4NF because the two MVDs are trivial
- If a relation is in 4NF it is in BCNF

Overall Theory

- Study properties of FDs
  - Which are redundant FDs ? Extraneous attributes ?
- develop algorithms to generate lossless decompositions (into BCNF and 3NF : analytical approach)
  - schema synthesis approach also possible
- develop algorithms to test if a decomposition is dependency-preserving



ER Model and Normalization

- When an E-R diagram is carefully designed, identifying all entities correctly, the tables generated from the E-R diagram should not need further normalization.

More Theory and algorithms

Closure of Attribute Sets

- Given a set of attributes α, define the *closure* of α under F (denoted by α^+) as the set of attributes that are functionally determined by α under F
- Algorithm to compute α^+, the closure of α under F

```
result := α;
while (changes to result) do
    for each β → γ in F do
      begin
        if β ⊆ result then  result := result ∪ γ
      end
```

We can define a BCNF relation on 'result'

Uses of Attribute Closure

- Testing for superkey:
  - To test if α is a superkey, we compute α^+, and check if α^+ contains all attributes of R.
- Testing functional dependencies
  - To check if a FD $\alpha \rightarrow \beta$ holds, just check if $\beta \subseteq \alpha^+$.
  - Is a simple and cheap test, and very useful

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B \quad A \rightarrow C$
  $CG \rightarrow H \quad CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. $result = AG$
  2. $result = ABCG$ $\quad$ ($A \rightarrow C$ and $A \rightarrow B$)
  3. $result = ABCGH$ $\quad$ ($CG \rightarrow H$)
  4. $result = ABCGHI$ $\quad$ ($CG \rightarrow I$)
- Is AG a candidate key?
  1. Is AG a super key?
     1. Does $AG \rightarrow R$? == Is $(AG)^+ \subseteq R$
  3. Is any subset of AG a superkey?
     1. Does $A \rightarrow R$? == Is $(A)^+ \subseteq R$
     2. Does $G \rightarrow R$? == Is $(G)^+ \subseteq R$

Canonical Cover

- Redundancies in a given F
  - An FD may be redundant :

  $A \rightarrow C$ is redundant in: $\{A \rightarrow B,\ B \rightarrow C, A \rightarrow C\}$
  - attributes may be redundant (extraneous)
    - on RHS: $\{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow CD\}$ can be simplified to $\{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow D\}$
    - on LHS: $\{A \rightarrow B,\ B \rightarrow C,\ AC \rightarrow D\}$ can be simplified to $\{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow D\}$
- Intuitively, a canonical cover of F is a "minimal" set of FDs equivalent to F, having no redundancies
  - No functional dependency in F_c contains an extraneous attribute, and
  - Each left side of functional dependency in F_c is unique.

Dependency Preservation

- Let F_i be the set of dependencies from F^+ that include only attributes in R_i.
  - A decomposition is dependency preserving, if

  $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$
  - If it is not, then checking updates for violation of functional dependencies may require computing joins, which is expensive.

Testing for Dependency Preservation

- To check if $\alpha \rightarrow \beta$ is preserved in a decomposition of R into $R_1, R_2, \ldots, R_n$ we apply the following test (with attribute closure done with respect to F)

```
result = α
while (changes to result) do
  for each Ri in the decomposition
        t = (result ∩ Ri)+ ∩ Ri
        result  =  result  ∪ t
```

whichever table contains α also β in its α+

If *result* contains all attributes in β, then the FD $\alpha \rightarrow \beta$ is preserved.

Example of BCNF Decomposition

class (*course_id*, *title*, *dept_name*, *credits*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)

- Functional dependencies:
  - *course_id*→ *title*, *dept_name*, *credits*
  - *building*, *room_number*→*capacity*
  - *course_id*, *sec_id*, *semester*, *year*→*building*, *room_number*, *time_slot_id*
- A candidate key {*course_id*, *sec_id*, *semester*, *year*}.

Example of BCNF Decomposition

class (*course_id*, *title*, *dept_name*, *credits*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)

- BCNF Decomposition:
  - *course_id*→ *title*, *dept_name*, *credits* holds
    - but *course_id* is not a superkey.
  - We replace *class* by:
    - *course*(*course_id*, *title*, *dept_name*, *credits*)
    - *class-1* (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)

Comparison of BCNF and 3NF

- It is always possible to decompose a relation into a set of relations that are in 3NF such that:
  - the decomposition is lossless
  - the dependencies are preserved
- It is always possible to decompose a relation into a set of relations that are in BCNF such that:
  - the decomposition is lossless
  - it may not be possible to preserve dependencies.

- SQL does not provide a direct way of specifying functional dependencies other than superkeys.

  Even if we had a dependency preserving decomposition, using SQL, we would not be able to efficiently test a functional dependency whose left hand side is not a key.

Multivalued Dependencies (MVDs)

- Let *R* be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The **multivalued dependency**

  $$\alpha \twoheadrightarrow \beta$$

  holds on *R* if in any legal relation *r(R),* for all pairs for tuples t_1 and t_2 in *r* such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in *r* such that:

  $$\begin{aligned} t_1[\alpha] = t_2[\alpha] &= t_3[\alpha] = t_4[\alpha] \\ t_3[\beta] &= t_1[\beta] \\ t_3[R - \beta] &= t_2[R - \beta] \\ t_4[\beta] &= t_2[\beta] \\ t_4[R - \beta] &= t_1[R - \beta] \end{aligned}$$

- Properties of MVDs defined

Theory of MVDs

The closure D^+ of *D* is the set of all functional and multivalued dependencies logically implied by *D.*

- We can compute D^+ from *D*, using the formal definitions of functional dependencies and multivalued dependencies.
- We can manage with such reasoning for very simple multivalued dependencies, which seem to be most common in practice
- For complex dependencies, it is better to reason about sets of dependencies using a system of inference rules

Example

- $R = (A, B, C, G, H, I)$
  $F = \{ A \twoheadrightarrow B$
  $B \twoheadrightarrow HI$
  $CG \twoheadrightarrow H \}$
- R is not in 4NF since $A \twoheadrightarrow B$ and A is not a superkey for R
- Decomposition
  a) $R_1 = (A, B)$ (R_1 is in 4NF)
  b) $R_2 = (A, C, G, H, I)$ (R_2 is not in 4NF, decompose into R_3 and R_4)
  c) $R_3 = (C, G, H)$ (R_3 is in 4NF)
  d) $R_4 = (A, C, G, I)$ (R_4 is not in 4NF, decompose …)

BCNF and Dependency Preservation

It is not always possible to get a BCNF decomposition that is dependency preserving

- $R = (J, K, L)$
  $F = \{JK \rightarrow L$
  $L \rightarrow K \}$
  Two candidate keys = JK and JL
- R is not in BCNF
- Any decomposition of R will fail to preserve
  $JK \rightarrow L$
  This implies that testing for $JK \rightarrow L$ requires a join

Third Normal Form: Motivation

- There are some situations where
  - BCNF is not dependency preserving, and
  - efficient checking for FD violation on updates is important
- Solution: define a weaker normal form, called Third Normal Form (3NF)
  - Allows some redundancy (with resultant problems; we will see examples later)
  - There is always a lossless-join, dependency-preserving decomposition into 3NF.

Redundancy in 3NF

- There is some redundancy in this schema
- Example of problems due to redundancy in 3NF
  - $R = (City, street, PIN)$
    $F = \{City, street \rightarrow PIN, PIN \rightarrow City\}$
  - It is in 3NF as City is part of Key (City, Street), and not BCNF because PIN is not its key
- repetition of information
  - PIN, City will repeat for multiple streets

Summary

- Dependencies capture constraints
- Can be used to structure relations to ensure maintaining constraints is possible
- Constraints have formal properties using which
  - We can decompose 'bad' relations
  - Synthesize 'good' relations
- Many normal forms defined to capture desirable structures
- Properly done ER models will produce relations in good normal form
- Theory not much used in practice but gives us formal and useful insight