Lecture 10

Functional Dependencies and Normalization for Relational Databases

There are two ways to design database

1. **Top-Down method**: By using Entity-Relationship diagram.
2. **Bottom-Up method**: By using Function dependency between attributes and normalization.

- **Fully normalized** term will be used to describe a collection of tables that are structured so that they can not contain redundant data

Functional Dependencies

- Functional dependencies (FDs)
  - Are used to specify *formal measures* of the "goodness" of relational designs
  - And keys are used to define **normal forms** for relations
  - Are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes
- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y

FD : If you have X , Can you get a unique Y.
X-> Y property is true when :
* X is PK (no duplicates): since PK can get any unique attribute.
* tuple1[X] = tuple2[X] & tuple1[Y] = tuple2[Y] : since you can still get unique Y from X.

Functional Dependencies

- $X \rightarrow Y$ holds if whenever two tuples have the same value for X, they *must have* the same value for Y

relation Instance : table state (populated data)`

  - 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)

i.e: The functional dependency should hold for all possible data in the attributes, for e.g If a functional dependency exist between a set of attributes in an instance (data state) but when the data changes they do are not dependent anymore, then there is no functional dependency between these attributes.

- Written as $X \rightarrow Y$; can be displayed graphically on a relation schema
- FDs are derived from the real-world constraints on the attributes

Examples of FD constraints

- Social security number determines employee name

  SSN → ENAME

- Project number determines project name and location

  PNUMBER → {PNAME, PLOCATION}

- Employee ssn and project number determines the hours per week that the employee works on the project

  {SSN, PNUMBER} → HOURS

Examples of FD constraints

- An FD is a property of the attributes in the schema R
- The constraint must hold on *every* relation instance r(R) i.e: All states (populated data) for the relation (table)
- If K is a key of R, then K functionally determines all attributes in R
  - (since we never have two distinct tuples with t1[K]=t2[K])

FD's are a property of the meaning of data and hold at all times: certain FD's can be ruled out based on a given state of the database

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Figure 10.7
A relation state of TEACH with a *possible* functional dependency TEXT → COURSE. However, TEACHER → COURSE is ruled out.

since t1[Teacher] = t1[Teacher] and t1[Course] != t2[Course]

When having 'TEXT' , we can get Unique value for "Course' . Rule : IF X (TEXT) has no duplicates then functional dependency exist (In this instance at lease , other data states can rule out the FD (break FD)

Deriving FDs from FDs :الاستدلال

Inference Rules for FDs

Armstrong's Axioms: These are the rules which you can use to define most of the FDs in a relation (Table)

- Given a set of FDs F, we can **infer** additional FDs that hold whenever the FDs in F hold
- Armstrong's inference rules:

e.g : {SSN,NAME} -> {Name} , Then : {SSN -> NAME}.
Also : X -> X (If you have an attribute X you can get unique attribute X) (Trivial and unnecessary)

  - IR1. (**Reflexive**) If Y *subset-of* X, then X → Y

Very Important

  - IR2. (**Augmentation**) If X → Y, then XZ → YZ

IF : SSN -> EMP_NAME
Then : {SSN , Phone} -> {EMP_NAME , Phone}

Very Important

    - (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
  - These are rules hold and all other rules that hold can be deduced from these

Inference Rules for FDs

- Some additional inference rules that are useful:
  - **IR4: Decomposition:** If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$ As long as 'determinant' is the same
  - **IR5: Union:** If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$ As long as 'determinant' is the
  - **IR6: Psuedotransitivity:** If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$

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

EXAMPLE

For a relation R with attributes A, B, C, D, E, F, and the FDs are:

$$A \rightarrow BC$$

$$B \rightarrow E$$

$$CD \rightarrow EF$$

Show that **$AD \rightarrow F$** holds also for R.

SOLUTION

1. $A \rightarrow BC$ (given)
2. $A \rightarrow C$ (decomposition of 1)
3. $AD \rightarrow CD$ (augmentation of 2)
4. $CD \rightarrow EF$ (given)
5. $AD \rightarrow EF$ (transitivity 3,4)
6. $AD \rightarrow F$ (decomposition of 5)

The goal of getting attribute Closures is to find all candidate keys since if a closure set of X is all attributes then it is a candidate key.

Inference Rules for FDs

Closure for an attribute X is a set of all other attributes which X determines {SSN} + = {NAME,Phone,DEPT, .. }

- **Closure** of a set F of FDs is the set F^+ of all FDs that can be inferred from F
- **Closure** of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X
- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

Super Key: set of attributes whose closure have all attributes

Example

- If:

SSN → ENAME

PNUMBER → PNAME , PLOCATION

{SSN, PNUMBER} → HOURS

- Then:

$\{SSN\}^{+} = \{SSN, ENAME\}$

$\{PNUMBER\}^{+} = \{PNUMBER, PNAME, PLOCATION\}$

$\{SSN, PNUMBER\}^{+} = \{SSN, ENAME, PNUMBER,$
$PNAME, PLOCATION, HOURS\}$

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
- There is an algorithm for checking equivalence of sets of FDs

Minimal Sets of FDs

Cononical Cover / Minimal Cover / Irreducable set of FDs

- A set of FDs is **minimal** if it satisfies the following conditions:
  1. Every dependency in F has a single attribute for its RHS. 1. RHS has to be 1 attribute
  2. We cannot remove any dependency from F and have a set of dependencies that is equivalent to F. 2. No Redundant **attributes** on left hand side. e.g AD -> C (can possibly be A -> C , D -> or even AD -> C if both must present)
  3. We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$, where Y proper-subset-of X (Y subset-of X) and still have a set of dependencies that is equivalent to F.

No Reundant FDS , i.e : the set of FDs should not be FDs that can be derived from each other, keep only the 'irreducable FDs'

Minimal Sets of FDs

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs
- To synthesize a set of relations, we assume that we start with a set of dependencies that is a minimal set

Computing the Minimal Sets of FDs

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.

1- All above dependencies are in canonical form (that is RHS is one attribute) ; so we have completed step 1

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 $A \rightarrow D$?

Computing the Minimal Sets of FDs

- Since B → A, (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 $B \rightarrow D$.

Computing the Minimal Sets of FDs

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

$$\{B \rightarrow D, D \rightarrow A\}.$$

Another example

For a relation R with attributes A, B, C, D and the FDs are:

1. $A \rightarrow BC$
2. $B \rightarrow C$
3. $A \rightarrow B$
4. $AB \rightarrow C$
5. $AC \rightarrow D$

Compute a minimal set of FDs.

Solution

- Apply rule 1 to FD no.1 :

1. $A \rightarrow B$
2. $A \rightarrow C$
3. $B \rightarrow C$
4. $A \rightarrow B$
5. $AB \rightarrow C$
6. $AC \rightarrow D$

FD 1 and 4 are the same, eliminate one.

- FD 6 : We can eliminate C from LHS as it is redundant for: $A \rightarrow C$, then $A \rightarrow AC$ (augmentation);

  $AC \rightarrow D$ (FD 6) Then $A \rightarrow D$ (transitivity)

Solution

- FD 5 : We can eliminate it as :

  $A \rightarrow C$, then $AB \rightarrow CB$ (augmentation);

  so $AB \rightarrow C$ (decomposition)

- From 1,3 there exist transitivity that lead to $A \rightarrow C$, which is the same as no.2, so eliminate FD 2.

- Then the final minimal set of FDs is:

$A \rightarrow B$

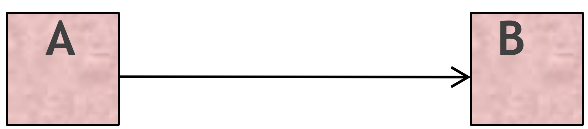
$B \rightarrow C$

$A \rightarrow D$

Determinancy Diagrams

- A simple way of showing determinants and the attributes that determine is to draw ***determinancy diagram*** (or ***functional dependency diagram***) as follow:

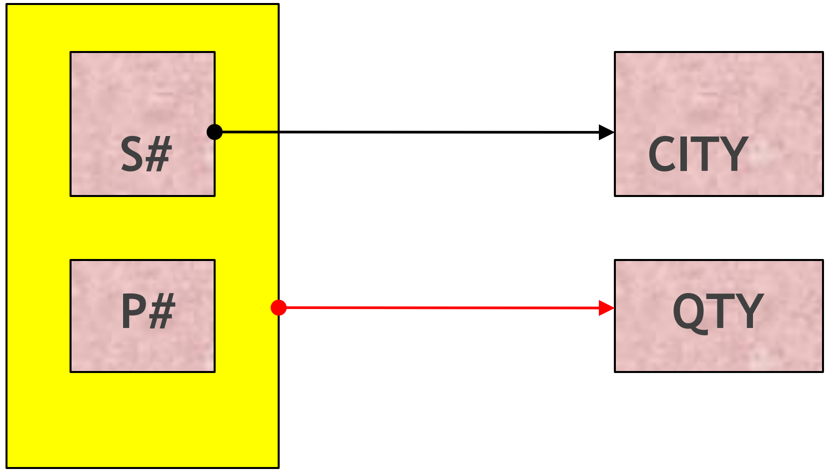
$$A \rightarrow B$$



For the SCP table; its FDs can be represented as:

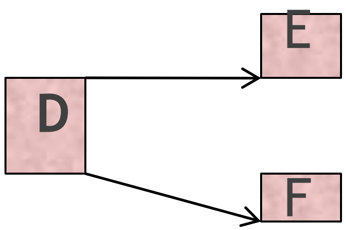
{S#} → {CITY}

{S#, P#} → {QTY}

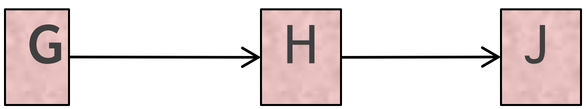


Identifier

- An identifier is an attribute or composite attributes that can never have duplicate values within a table occurrence, and whose value is sufficient to identify a row. None of the component attributes of an identifier may have NULL values.



a) D is the identifier



b) G is the identifier

c) (K,L) is the identifier