Functional Dependency

Swarup Kr Ghosh

Introduction

- Functional Dependency (FD) is a *many to one relationship* from one set of attributes to another in a given relation
- Two aspects of dependency
  - Case a: the values of a given relation at a given point in time
  - Case b: the set of all possible values that the given relation may assume at different time

FIRST

s#	status	city	p#	qty
s1	20	London	p1	100
s1	20	London	p2	100
s2	10	Paris	p1	200
s2	10	Paris	p2	200
s3	10	Paris	p2	300
s4	20	London	p2	400
s4	20	London	p4	500
s4	20	London	p5	600
s5	20	Kolkata	p1	200

Functional Dependency

- Let r be a relation and let X and Y be arbitrary subsets of the set of attributes of r.
- Then we say Y is functionally dependent on X- in symbols

$$X \rightarrow Y$$

  - (or X functionally determines Y)- if and only if, each X value in r has associated with it precisely one Y value in r
  - Whenever two tuples of r agree in their X value, they also agree on their Y value

Examples of FDs

$\{s\#\} \rightarrow \{city\}$

$\{city\} \rightarrow \{status\}$

$\{s\#,p\#\} \rightarrow \{qty\}$

$\{s\#,p\#\} \rightarrow \{city\}$

$\{s\#,p\#\} \rightarrow \{s\#\}$

$\{s\#,p\#\} \rightarrow \{s\#,p\#,city,qty\}$

FIRST

s#	status	city	p#	qty
s1	20	London	p1	100
s1	20	London	p2	100
s2	10	Paris	p1	200
s2	10	Paris	p2	200
s3	10	Paris	p2	300
s4	20	London	p2	400
s4	20	London	p4	500
s4	20	London	p5	600
s5	20	Kolkata	p1	200

Functional Dependency

- Determinant and Dependent of FD
- Singleton set
- FD holds for all time
- Integrity Constraint

$X \rightarrow Y$

$s\# \rightarrow city$

$\{s\#, p\#\} \rightarrow qty$

Functional Dependency

- If X is a candidate key of a relation R then all attributes Y of R is functionally dependent on X
- p # ⓪ { p # , pname, color, weight, city}
- If Relation R satisfies A ⓪ B and the B is not *trivial* and A is not candidate key then R will involve some REDUNDANCY

FIRST

s#	status	city	p#	qty
s1	20	London	p1	100
s1	20	London	p2	100
s2	10	Paris	p1	200
s2	10	Paris	p2	200
s3	10	Paris	p2	300
s4	20	London	p2	400
s4	20	London	p4	500
s4	20	London	p5	600
s5	20	Kolkata	p1	200

$s\# \rightarrow city$

Trivial and Nontrivial FDs

Trivial FD: R.H.S is the subset of LHS

$$\{s\#,p\#\} \rightarrow \{s\#\}$$

Nontrivial FD

$$\{s\#,p\#\} \rightarrow \{city\} \qquad \{s\#,p\#\} \rightarrow \{s\#,p\#,city,qty\}$$

$$\{s\#\} \rightarrow \{city\} \qquad \{s\#,p\#\} \rightarrow \{qty\}$$

Transitive Dependency

- Let us have R(A,B,C)
- FDs: A→B and B→C holds true for R
- Then we say that A→C also holds for R
- The FD A→C is a transitive FD….. C is transitively dependent on A via B

Transitive Dependency

FIRST

s#	status	city	p#	qty
s1	20	London	p1	100
s1	20	London	p2	100
s2	10	Paris	p1	200
s2	10	Paris	p2	200
s3	10	Paris	p2	300
s4	20	London	p2	400
s4	20	London	p4	500
s4	20	London	p5	600
s5	20	Kolkata	p1	200

Closure of a Set of Dependencies

❖ Some FDs might imply other

$$\{s\#,p\#\} \rightarrow \{city\} \qquad \{s\#,p\#\} \rightarrow \{qty\}$$

$$\{s\#,p\#\} \rightarrow \{city,qty\}$$

❖ The set of all FDs that are implied by a given set F of FDs is called the closure of F

Written as F^+

F^+ is a superset of *F*

Inference Rules (Armstrong's Axioms)

❖Let A,B,C, W be arbitrary subsets of the set of attributes of the given relation R

❖**IR1: REFLEXIVITY:** If B is a subset of A Then A→B holds true

FIRST

s#	status	city	p#	qty
s1	20	London	p1	100
s1	20	London	p2	100
s2	10	Paris	p1	200
s2	10	Paris	p2	200
s3	10	Paris	p2	300
s4	20	London	p2	400
s4	20	London	p4	500
s4	20	London	p5	600
s5	20	Kolkata	p1	200

Inference Rules (Armstrong's Axioms)

❖Let A,B,C, W be arbitrary subsets of the set of attributes of the given relation R

❖**IR1: REFLEXIVITY:** If B is a subset of A Then A→B holds true

❖**IR2: AUGMENTATION** If A→B then AC→BC

(Notation: AC stands for A U C)

❖**IR3: TRANSITIVE**: If A→B and B→C, then A→C

❖These rules are

- **complete** (Given a set S of FDs, all FDs implied by S can be derived from S using the rules)
- **sound** (No additional FDs can be derived)

Inference Rules (Armstrong's Axioms)

<u>Some additional inference rules :</u>

- **Self Determination:** A→A
- **Decomposition:** If A → BC, then A → B and A → C
- **Union:** If A → B and A → C, then A → BC
- **Composition:** If A→B and C→D , Then AC→ BD
- **Psuedotransitivity:** If A → B and WB→C, then WA → C

These inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

Inference Rules (Armstrong's Axioms)

- Example:
- Consider a relation R with attributes A,B,C,D,E , F and the FDs
  - A→BC
  - B→E
  - CD→EF
- Show that FD AD→F holds for R and is thus a member of the closure of the given set.

F Closure (F^+)

❖ Closure of set of Functional Dependencies

- Given a set F set of functional dependencies, there are certain other functional dependencies that are logically implied by F
  - For e.g.: If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that $A \rightarrow C$
- The set of **all** functional dependencies logically implied by F is the **closure** of F
- We denote the *closure* of F by F^+

F Closure example

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- Some members of F^+
  - $A \rightarrow H$
    - by transitivity from $A \rightarrow B$ and $B \rightarrow H$
  - $AG \rightarrow I$
    - by augmenting $A \rightarrow C$ with G, to get $AG \rightarrow CG$ and then transitivity with $CG \rightarrow I$
  - $CG \rightarrow HI$
    - by augmenting $CG \rightarrow I$ to infer $CG \rightarrow CGI$, and augmenting of $CG \rightarrow H$ to infer $CGI \rightarrow HI$, and then transitivity

F Closure

- To compute the closure of a set of functional dependencies F:

```
F + = F
repeat
      for each functional dependency f in F+
            apply reflexivity and augmentation rules on f
            add the resulting functional dependencies to F +
      for each pair of functional dependencies f1 and f2 in F +
            if f1 and f2 can be combined using transitivity
                  then add the resulting functional dependency to F +
until F + does not change any further
```

NOTE: We shall see an alternative procedure for this task later

Closure of a set of Attributes

- **Closure** of a set of attributes Z/α with respect to F is the set Z+/ α+ of all attributes that are functionally determined by Z/ α
- Z^+ can be calculated by repeatedly applying IR1, IR2, IR3

using the FDs in F

- Given a relation R, a set Z of attributes R, and a set F of FDs that

hold for R→ can be used to determine the set of all attributes of R(Z^+)

that are functionally dependent on Z→ Closure Z^+ of Z under F.

Closure of Attribute Sets(α^+)

- Given a relation R, and 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 X → Y **in** *F* **do**
      begin
        if X ⊆ *result* **then** *result* := *result* ∪ Y
      end

Example of Attribute Set Closure

- *R = (A, B, C, G, H, I)*
- *F* = {*A* → *B* ,*A* → *C*, *CG* → *H*, *CG* → *I*, *B* → *H*}
- Find *(AG)*$^+$ of the set of attributes {AG} under the set of FDs
  1. *result = AG*
  2. *result = ABG* (*A* → *B*)
  3. *result = ABCG* (*A* → *C*)
  3. *result = ABCGH* (*CG* → *H* and *CG* ⊆ *AGBC*)
  4. *result = ABCGHI* (*CG* → *I* and *CG* ⊆ *AGBCH*)

Is *AG* a candidate key?

Example of Attribute Set Closure

- **Is *AG* a candidate key?**
  1. **Is AG a super key?**
     1. Does $AG \rightarrow R$? == Is $(AG)^+ \supseteq R$
  2. **Is any subset of AG a superkey?**
     1. Does $A \rightarrow R$? == Is $(A)^+ \supseteq R$
     2. Does $G \rightarrow R$? == Is $(G)^+ \supseteq R$

Uses of Attribute Closure

There are several uses of the attribute closure algorithm:

- **Testing for Superkey:**
  - To test if X is a superkey, we compute X^+, and check if X^+ contains all attributes of *R*.
- **Testing Functional Dependencies**
  - To check if a functional dependency $\alpha \rightarrow \beta$ holds (or, in other words, is in F^+), just check if $\beta \subseteq \alpha^+$.
  - That is, we compute α^+ by using attribute closure, and then check if it contains β.
  - Is a simple and cheap test, and very useful
- **Computing closure of F**
  - For each $\gamma \subseteq R$, we find the closure γ^+, and for each $S \subseteq \gamma^+$, we output a functional dependency $\gamma \rightarrow S$.

Example of Attribute Set Closure

- *Example:*
- $R = (A, B, C, D, E, F)$
- $F = \{A \rightarrow BC, E \rightarrow CF, B \rightarrow E, CD \rightarrow EF\}$
- Compute the closure $(AB)^+$ of the set of attributes {AB} under the set of FDs

Cover

- Let S1 and S2 be two set of FDs.
- If every FD implied by S1 is implied by S2 – i.e., $S1^+$ is a subset of $S2^+$ → S2 is a cover of S1
- If S2 is a cover of S1 and S1 is a cover of S2 →

  $S1^+ = S2^+$ → S1 and S2 are equivalent

Canonical Cover

❖ Sets of functional dependencies may have redundant dependencies that can be inferred from the others

- ✓ For example: $A \rightarrow C$ is redundant in: $\{A \rightarrow B, \ B \rightarrow C\}$
- ✓ Parts of a functional dependency may be redundant
  - ✓ E.g.: on RHS: $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow CD\}$ can be simplified to $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow D\}$
  - ✓ E.g.: 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 functional dependencies equivalent to F, having no redundant dependencies or redundant parts of dependencies

Canonical Cover: RHS

- $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow CD\}$ ➔ $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow D\}$
  - (1) $A \rightarrow CD$ ➔ $A \rightarrow C$ and $A \rightarrow D$ (2) $A \rightarrow B, \ B \rightarrow C$ ➔ $A \rightarrow C$
  - $A+ = ABCD$
- $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow D\}$ ➔ $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow CD\}$
  - $A \rightarrow B, \ B \rightarrow C$ ➔ $A \rightarrow C$
  - $A \rightarrow C, A \rightarrow D$ ➔ $A \rightarrow CD$
  - $A+ = ABCD$

Canonical Cover: LHS

- $\{A \to B,\ \ B \to C,\ \ AC \to D\} \Rightarrow \{A \to B,\ \ B \to C,\ \ A \to D\}$
  - $A \to B,\ \ B \to C \Rightarrow A \to C \Rightarrow A \to AC$
  - $A \to AC$, $AC \to D \Rightarrow A \to D$
  - $A+ = ABCD$
- $\{A \to B,\ \ B \to C,\ \ A \to D\} \Rightarrow \{A \to B,\ \ B \to C,\ \ AC \to D\}$
  - $A \to D \Rightarrow AC \to D$
  - $AC+ = ABCD$

Extraneous Attributes

- Consider a set *F* of functional dependencies and the functional dependency $\alpha \to \beta$ in *F*.
  - Attribute A is extraneous in α if $A \in \alpha$ and *F* logically implies $(F - \{\alpha \to \beta\}) \cup \{(\alpha - A) \to \beta\}$.
  - Attribute *A* is extraneous in β if $A \in \beta$ and the set of functional dependencies $(F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - A)\}$ logically implies *F*.
- *Note:* implication in the opposite direction is trivial in each of the cases above, since a "stronger" functional dependency always implies a weaker one
  - Example: Given $F = \{A \to C, AB \to C\}$
    B is extraneous in $AB \to C$ because $\{A \to C, AB \to C\}$ logically implies $A \to C$ (I.e. the result of dropping *B* from $AB \to C$).
  - Example: Given $F = \{A \to C, AB \to CD\}$
    C is extraneous in $AB \to CD$ since $AB \to C$ can be inferred even after deleting *C*

Testing if an Attribute is Extraneous

- Consider a set F of functional dependencies and the functional dependency $\alpha \to \beta$ in F.
- To test if attribute $A \in \alpha$ is extraneous in α
  1. compute $(\{\alpha\} - A)^+$ using the dependencies in F
  2. check that $(\{\alpha\} - A)^+$ contains β; if it does, A is extraneous in α
- To test if attribute $A \in \beta$ is extraneous in β
  1. compute α^+ using only the dependencies in
     $F' = (F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - A)\}$,
  2. check that α^+ contains A; if it does, A is extraneous in β

Canonical Cover

- A *canonical cover* for F is a set of dependencies F_c such that
  - F logically implies all dependencies in F_c, and
  - F_c logically implies all dependencies in F, and
  - No functional dependency in F_c contains an extraneous attribute, and
  - Each left side of functional dependency in F_c is unique.

- Minimal Set of FDs
- Irreducible set of FDs

- To compute a canonical cover for F:

repeat

Use the union rule to replace any dependencies in F

$\alpha_1 \to \beta_1$ and $\alpha_1 \to \beta_2$ with $\alpha_1 \to \beta_1 \beta_2$

Find a functional dependency $\alpha \to \beta$ with an extraneous attribute either in α or in β

If an extraneous attribute is found, delete it from $\alpha \to \beta$

until F does not change

- Note: Union rule may become applicable after some extraneous attributes have been deleted, so it has to be re-applied

Computing a Canonical Cover

- $R = (A, B, C)$
  $F = \{A \rightarrow BC$
  $B \rightarrow C$
  $A \rightarrow B$
  $AB \rightarrow C\}$
- Combine $A \rightarrow BC$ and $A \rightarrow B$ into $A \rightarrow BC$
  - Set is now $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$
- A is extraneous in $AB \rightarrow C$
  - Check if the result of deleting A from $AB \rightarrow C$ is implied by the other dependencies
    Yes: in fact, $B \rightarrow C$ is already present!
  - Set is now $\{A \rightarrow BC, B \rightarrow C\}$
- C is extraneous in $A \rightarrow BC$
  - Check if $A \rightarrow C$ is logically implied by $A \rightarrow B$ and the other dependencies
    Yes: using transitivity on $A \rightarrow B$ and $B \rightarrow C$.
    Can use attribute closure of A in more complex cases
- The canonical cover is: $A \rightarrow B$
  $B \rightarrow C$

Irreducible set of Functional Dependencies

- R(p#, pname, color, wt, city)
- The set of FDs:
  - p # →pname
  - p # →color
  - p # →wt
  - p # →city

Irreducible set of Functional Dependencies

- A set F of FD is said to be irreducible if and only if it satisfies the following
  - The RHS (the dependent) of every FD in F involves just one attribute ,i.e., it is a singleton set
  - The LHS (the determinant) of every FD in F is irreducible in turn- meaning that no attribute can be discarded from the determinant without changing the closure F^+ → Such an FD is called **Left irreducible**
  - No FD in F can be discarded from F without changing the closure F^+ ,i.e., without converting F into some set not equivalent to F

Irreducible set of Functional Dependencies

- The RHS (the dependent) of every FD in F involves just one attribute ,i.e., it is a singleton set

p#→ {pname, color}	p#→ pname
p#→ color	p#→ color
p#→ wt	p#→ wt
p#→ city	p#→ city

Irreducible set of Functional Dependencies

- The LHS (the determinant) of every FD in F is irreducible in turn-meaning that no attribute can be discarded from the determinant without changing the closure F^+ → Such an FD is called **Left irreducible**

{p#, pname → color	p# → pname
p# → pname	p# → color
p# → wt	p# → wt
p# → city	p# → city

NOT LEFT-IRREDUCIBLE

Irreducible set of Functional Dependencies

- No FD in F can be discarded from F without changing the closure F+ ,i.e., without converting F into some set not equivalent to F

p#, → p#	p# → pname
p#, → color	p# → color
p# → pname	p# → wt
p# → wt	p# → city
p# → city	

Irreducible set of Functional Dependencies

- Example: Consider a relation R(A, B, C, D) and FDs
- $A \rightarrow BC$, $B \rightarrow C$, $A \rightarrow B$, $AB \rightarrow C$, $AC \rightarrow D$
- Compute an irreducible set of FDs that is equivalent to this given set

Solution:
$A \rightarrow B$
$B \rightarrow C$
$A \rightarrow D$

Equivalence of Sets of Functional Dependencies

- Let F & G are two functional dependency sets.
  - These two sets F & G are equivalent if $F^+ = G^+$
  - Equivalence means that every functional dependency in F can be inferred from G, and every functional dependency in G an be inferred from F
- F and G are equal only if
  - F covers G: Means that all functional dependency of G are logically numbers of functional dependency set $F \Rightarrow F \supseteq G$.
  - G covers F: Means that all functional dependency of F are logically members of functional dependency set $G \Rightarrow G \supseteq F$

Practice problem

- **Find if a given functional dependency is implied from a set of Functional Dependencies:**
  1. For: A → BC, CD → E, E → C, D → AEH, ABH → BD, DH → BC
     a. Check: BCD → H
     b. Check: AED→C
  2. For: AB → CD, AF → D, DE → F, C → G, F → E, G → A
     a. Check: CF → DF
     b. Check: BG → E
     c. Check: AF → G
     d. Check: AB → EF
  3. For: A → BC, B → E, CD → EF
     a. Check: AD → F
- **Find Candidate Key using Functional Dependencies:**
  1. Relational Schema R(ABCDE). Functional dependencies: AB → C, DE → B, CD → E
  2. Relational Schema R(ABCDE). Functional dependencies: AB → C, C → D, B → EA

3NF Decomposition

- R= ABCDEFGH
- FD={ A→B, ABCD→E, EF→GH, ACDF→EG }

- Compute Canonical cover:
  - Make RHS a single attribute: { A→B, ABCD→E, EF→G, EF→H, ACDF→E, ACDF→G }
  - Minimize LHS: ACD→E instead of ABCD→E
  - Eliminate redundant FDs:
    - ✓ Can ACDF→E be removed?
    - ✓ Can ACDF→G be removed?

- Canonical cover= { A→B, ACD→E, EF→G, EF→H }

- Super key= ACDF

- 3NF decomposition: { AB, ACDE, EFG, EFH, } U {ACDF}

3NF Decomposition

- R= CSJDPQV
- FD={C→CSJDPQV, SD→P, JP→C,J→S }

- Compute Canonical cover:
  - Make RHS a single attribute:
  - Minimize LHS:
  - Eliminate redundant FDs:
    - ✓ Can be removed?
    - ✓ Can be removed?

- Canonical cover=

- Super key=

- 3NF decomposition: {CJDQV, JPC, JS, SDP}

@SKG 11/5/2020 41

BCNF Decomposition

- To check if a non-trivial dependency $\alpha \rightarrow$ causes a violation of BCNF
  1. compute α^+ (the attribute closure of α), and
  2. verify that it includes all attributes of *R*, that is, it is a super key of *R*.

- Simplified test: To check if a relation schema *R* is in BCNF, it suffices to check only the dependencies in the given set *F* for violation of BCNF, rather than checking all dependencies in F^+.
  - ❖ If none of the dependencies in *F* causes a violation of BCNF, then none of the dependencies in F^+ will cause a violation of BCNF either.

- However, simplified test using only *F* is incorrect when testing a relation in a decomposition of R
  - Consider $R = (A, B, C, D, E)$, with $F = \{ A ì B, BC ì D\}$
    - Decompose *R* into $R_1 = (A,B)$ and $R_2 = (A,C,D, E)$
    - Neither of the dependencies in *F* contain only attributes from
    - (A,C,D,E) so we might be mislead into thinking R_2 satisfies BCNF.
    - In fact, dependency $AC \rightarrow D$ in F^+ shows R_2 is not in BCNF.

@SKG 11/5/2020 42

BCNF Decomposition Algorithm

1. For all dependencies A → B in F+, check if A is a superkey
   - By using attribute closure
2. If not, then
   - Choose a dependency in F+ that breaks the BCNF rules, say A → B
   - Create R1 = A B
   - Create R2 = A (R – (B – A))
   - Note that: R1 ∩ R2 = A and A → AB (= R1), so this is lossless decomposition
3. Repeat for R1, and R2
   - By defining F1+ to be all dependencies in F that contain only attributes in R1
   - Similarly F2+

BCNF Decomposition Algorithm

result := {*R* };
done := false;
compute F^+;

while (not *done)* **do**
if (there is a schema R_i in *result* that is not in BCNF)
then begin
let $\alpha \rightarrow$ be a nontrivial functional dependency that holds on R_i
such that $\alpha \rightarrow R_i$ is not in F^+,
and $\alpha \cap \quad = \varnothing$;
result := (*result* – R_i) ∪ (R_i –) ∪ (α,);
end
else *done* := **true;**

Note: each R_i is in BCNF, and decomposition is lossless-join.

BCNF Decomposition Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B$
  $B \rightarrow C\}$
- Key = $\{A\}$
- R is not in BCNF ($B \rightarrow C$ but B is not super key)
- Decomposition
  $R_1 = (B, C)$
  $R_2 = (A,B)$

BCNF Decomposition Example

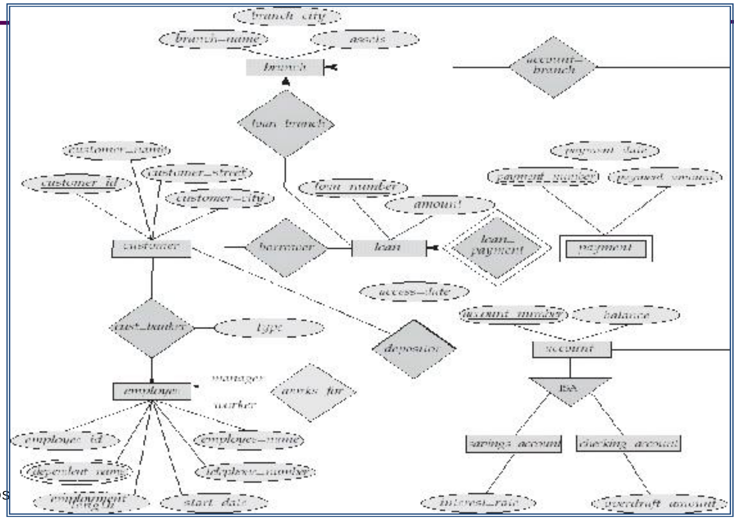
- Original relation R and functional dependency F
  R = (*branch_name, branch_city, assets, customer_name, loan_number, amount*)
  F = {*branch_name* → *assets, branch_city*
  loan_number → *amount, branch_name* }

  Key = {*loan_number, customer_name*}
- Decomposition
  - R_1 = (*branch_name, branch_city, assets*)
  - R_2 = (*branch_name, customer_name, loan_number, amount*)
  - R_3 = (*branch_name, loan_number, amount*)
  - R_4 = (*customer_name, loan_number*)
- Final decomposition
  R_1, R_3, R_4

Practice problem: BCNF Decomposition

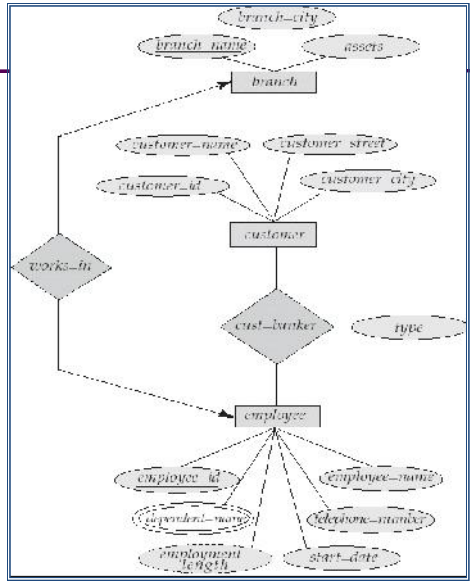
- R = ABCDE. F = {A → B, BC → D}
- R = ABCDE. F = {A → B, BC → D}
- R = ABCDEH. F = {A → BC, E → HA}
- R = CSJDPQV. F = {C →CSJDPQV, SD →P, JP →C,J → S}
- R = ABCD. F = {C→D, C→A, B→C}

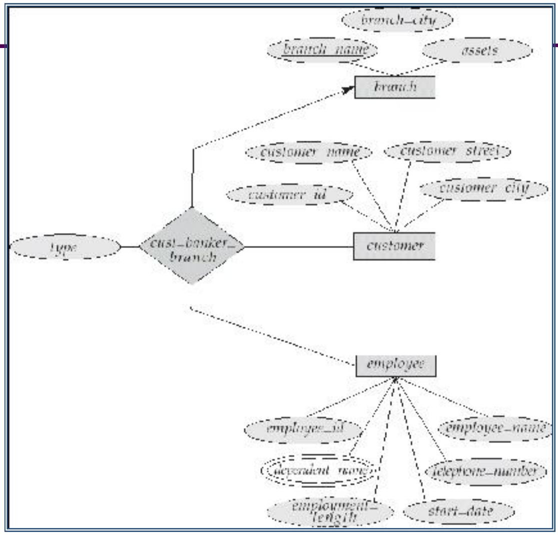
ERD vs NF



ERD vs NF

- 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
- However, in a real (imperfect) design, there can be functional dependencies from non-key attributes of an entity to other attributes of the entity
  - Example: an *employee* entity with attributes *department_name* and *building*, and a functional dependency *department_name* $\rightarrow$ *building*
  - Good design would have made department an entity
- Functional dependencies from non-key attributes of a relationship set possible, but rare --- most relationships are binary




branch_city
branch_name
assets
branch
customer_name
customer_street
customer_id
customer_city
type
cust_banker_branch
customer
employee
employee_id
employee_name
dependent_name
telephone_number
employment_length
start_date

Thank You

?