Why Normalization?

- To formally evaluate if our designed relations are good.
- To improve the design by addressing the discovered design problem.

Informal Design Guidelines

Semantic of the Relation Attributes

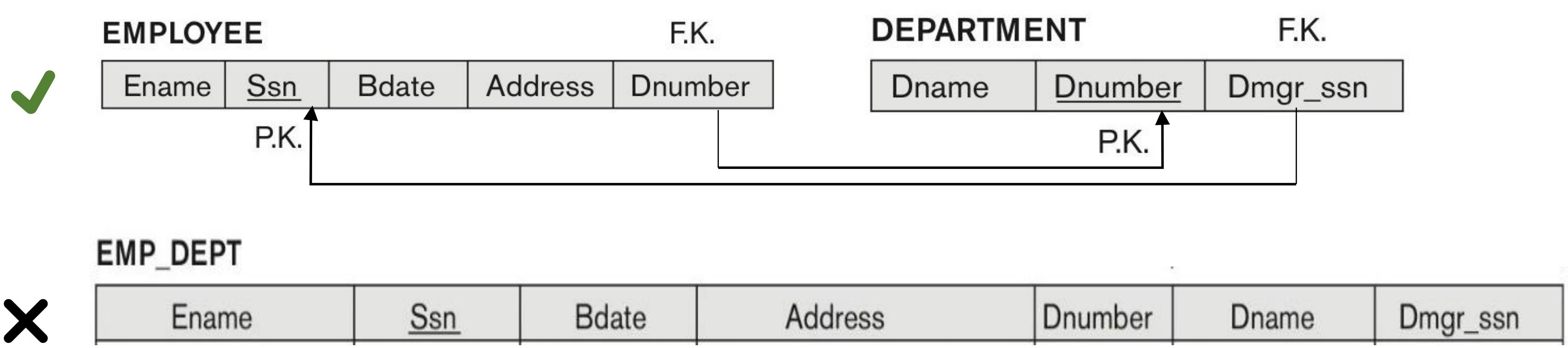
Redundant Information in Tuples and Update Anomalies

Null Values in Tuples

Spurious Tuples

Semantics of the Relation Attributes

- Informally, each tuple in a relation should represent one entity or relationship instance.
  - Attributes of different entities should not be mixed in the same relation
  - Only foreign keys should be used to refer to other entities
  - Entity and relationship attributes should be kept apart as much as possible.



Redundant Information and Anomalies

EMPLOYEE

F.K.

Ename	<u>Ssn</u>	Bdate	Address	Dnumber

P.K.

DEPARTMENT

F.K.

Dname	<u>Dnumber</u>	Dmgr_ssn

P.K.

Redundancy

EMP_DEPT

Ename	<u>Ssn</u>	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Bacher, Red, A	666666613	5/21/80	196 Elm Street, Miami, FL	8	Sales	555555500
Ball, Nandita, K	555555501	4/29/89	222 Howard, Sacramento, CA	6	Software	111111100
Bays, Bonnie, S	444444401	5/6/73	111 Hollow, Milwaukee, WI	7	Hardware	444444400
Bender, Bob, B	666666600	3/24/84	8794 Garfield, Chicago, IL	8	Sales	555555500
Best, Alec, C	444444402	11/10/87	233 Solid, Milwaukee, WI	7	Hardware	444444400
English, Joyce, A	453453453	7/31/83	5631 Rice Oak, Houston, TX	5	Research	333445555
Jabbar, Ahmad, V	987987987	3/29/79	980 Dallas, Houston, TX	4	Administration	987654321
Narayan, Ramesh, K	666884444	9/15/52	971 Fire Oak, Humble, TX	5	Research	333445555

A poorly designed database causes **anomalies:**

update anomaly

delete anomaly

insert anomaly

Update Anomaly

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Bacher, Red, A	666666613	5/21/80	196 Elm Street, Miami, FL	8	Sales	555555500
Ball, Nandita, K	555555501	4/29/89	222 Howard, Sacramento, CA	6	Software	111111100
Bender, Bob, B	666666600	3/24/84	8794 Garfield, Chicago, IL	8	Sales	555555500
Best, Alec, C	444444402	11/10/87	233 Solid, Milwaukee, WI	7	Hardware	444444400
English, Joyce, A	453453453	7/31/83	5631 Rice Oak, Houston, TX	5	Research	333445555
Jabbar, Ahmad, V	987987987	3/29/79	980 Dallas, Houston, TX	4	Administration	987654321
Narayan, Ramesh, K	666884444	9/15/52	971 Fire Oak, Humble, TX	5	Research	333445555

Updating a value of attribute that has redundant information may cause inconsistency

Example: Change the name of the Research department to R&D must be applied to all employees that work in the Research department otherwise the new state of EMP_DEPT has anomalies

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Bacher, Red, A	666666613	5/21/80	196 Elm Street, Miami, FL	8	Sales	555555500
Ball, Nandita, K	555555501	4/29/89	222 Howard, Sacramento, CA	6	Software	111111100
Bender, Bob, B	666666600	3/24/84	8794 Garfield, Chicago, IL	8	Sales	555555500
Best, Alec, C	444444402	11/10/87	233 Solid, Milwaukee, WI	7	Hardware	444444400
English, Joyce, A	453453453	7/31/83	5631 Rice Oak, Houston, TX	5	R&D	333445555
Jabbar, Ahmad, V	987987987	3/29/79	980 Dallas, Houston, TX	4	Administration	987654321
Narayan, Ramesh, K	666884444	9/15/52	971 Fire Oak, Humble, TX	5	Research	333445555

Delete & Insert Anomaly

Deleting employees may cause to losing the information about the departments.

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Bacher, Red, A	666666613	5/21/80	196 Elm Street, Miami, FL	8	Sales	555555500
Ball, Nandita, K	555555501	4/29/89	222 Howard, Sacramento, CA	6	Software	111111100
Bender, Bob, B	666666600	3/24/84	8794 Garfield, Chicago, IL	8	Sales	555555500
Best, Alec, C	444444402	11/10/87	233 Solid, Milwaukee, WI	7	Hardware	444444400
English, Joyce, A	453453453	7/31/83	5631 Rice Oak, Houston, TX	5	Research	333445555
Jabbar, Ahmad, V	987987987	3/29/79	980 Dallas, Houston, TX	4	Administration	987654321
Narayan, Ramesh, K	666884444	9/15/52	971 Fire Oak, Humble, TX	5	Research	333445555

We cannot add new a department without assigning an employee to it

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Bacher, Red, A	666666613	5/21/80	196 Elm Street, Miami, FL	8	Sales	555555500
Ball, Nandita, K	555555501	4/29/89	222 Howard, Sacramento, CA	6	Software	111111100
Bender, Bob, B	666666600	3/24/84	8794 Garfield, Chicago, IL	8	Sales	555555500
Best, Alec, C	444444402	11/10/87	233 Solid, Milwaukee, WI	7	Hardware	444444400
English, Joyce, A	453453453	7/31/83	5631 Rice Oak, Houston, TX	5	Research	333445555
Narayan, Ramesh, K	666884444	9/15/52	971 Fire Oak, Humble, TX	5	Research	333445555

Insert(NULL **NULL** NULL NULL 1 Headquarters NULL)

- The primary key of EMP_DEPT relation is SSN

Null Values In Tuples

- Relations should be designed such that their tuples will have as few NULL values as possible
- Attributes that are NULL frequently could be placed in separate relations (with the primary key)

Reasons for nulls:

- Attribute not applicable or invalid
- Attribute value unknown (may exist)
- Value known to exist, but unavailable

SSN	Ename	Dname	Location	Office#
888665555	Borg, James, E	Headquarters	Houston	NULL
987654321	Wallace, Jennifer, S	Administration	Stafford	451
987987987	Jabbar, Ahmad, V	Administration	Stafford	372
999887777	Zelaya, Alicia, J	Administration	Stafford	NULL
123456789	Smith, John, B	Research	Bellaire	NULL
333445555	Wong, Franklin, T	Research	Bellaire	NULL
453453453	English, Joyce, A	Research	Bellaire	NULL

SSN	Ename	Dname	Location
888665555	Borg, James, E	Headquarters	Houston
987654321	Wallace, Jennifer, S	Administration	Stafford
987987987	Jabbar, Ahmad, V	Administration	Stafford
999887777	Zelaya, Alicia, J	Administration	Stafford
123456789	Smith, John, B	Research	Bellaire
333445555	Wong, Franklin, T	Research	Bellaire
453453453	English, Joyce, A	Research	Bellaire

SSN	Office#
987654321	451
987987987	372

Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The "lossless join" property is used to guarantee meaningful results for join operations
- The relations should be designed to satisfy the lossless join condition.
- No spurious tuples should be generated by doing a natural-join of any relations.

SSN	Pnumber	Pname	Hours	Location
123456789	1	ProductX	32.5	Boston
123456789	2	ProductY	7.5	Sugarland
333445555	2	ProductY	10	Sugarland
333445555	3	ProductZ	10	Houston
453453453	4	ProductW	20	Boston

SSN	Hours	Location
123456789	32.5	Boston
123456789	7.5	Sugarland
333445555	10	Sugarland
333445555	10	Houston
453453453	20	Boston

$\bowtie$

Pnumber	Pname	Location
1	ProductX	Boston
2	ProductY	Sugarland
3	ProductZ	Houston
4	ProductW	Boston

SSN	Pnumber	Pname	Hours	Location
123456789	1	ProductX	32.5	Boston
123456789	2	ProductY	7.5	Sugarland
333445555	2	ProductY	10	Sugarland
333445555	3	ProductZ	10	Houston
453453453	4	ProductW	20	Boston
123456789	4	ProductW	32.5	Boston
453453453	1	ProductX	20	Boston

Functional Dependencies

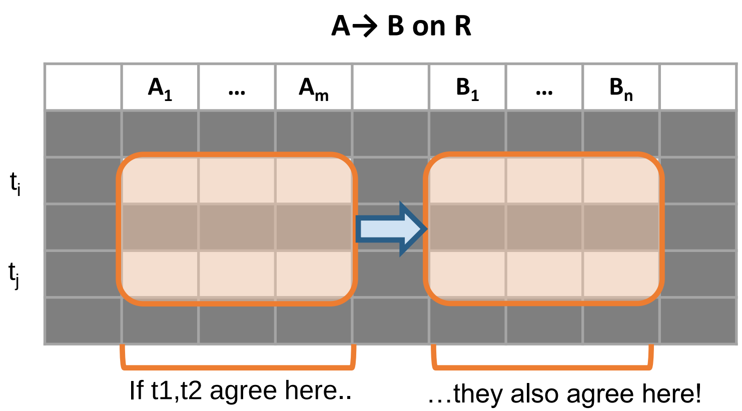
- Functional dependencies (FDs)
  - Are used to specify *formal measures* of the "goodness" of relational designs
  - And keys are used to define **normal forms** for relations
  - Are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes

Definition: Let A,B be *sets* of attributes

We write A → B or say A ***functionally determines*** B if, for any tuples t_1 and t_2:

$$t_1[A] = t_2[A] \text{ implies } t_1[B] = t_2[B],$$

and we call A → B a **functional dependency.**



A->B means that "whenever two tuples agree on A then they agree on B"

Examples of FDs

First Name	Last Name	SSN	DOB	Address	Sex	Salary
Ahmad	Jabbar	987987987	12/3/79	980 Dallas, Houston, TX	M	25000
Jared	James	111111100	5/12/65	123 Peachtree, Atlanta, GA	M	85000
John	James	555555500	5/12/65	7676 Bloomington, Sacramento, CA	M	81000
Jill	Jarvis	666666601	10/1/93	6234 Lincoln, Chicago, IL	F	36000
Jon	Jones	111111101	11/4/86	111 Allgood, Atlanta, GA	M	45000
Kate	King	666666602	7/3/92	1976 Boone Trace, Chicago, IL	F	44000
Alice	King	666666604	7/3/92	556 Washington, Chicago, IL	F	38000
Sara	Riazi	334241992	11/4/86	5201 Fairview Rd., Charlotte, NC	F	83200

$$\text{SSN} \rightarrow \text{Address}$$

Two distinguished tuples cannot have different addresses while having the same SSN

$$\text{Last Name} \not\rightarrow \text{Address}$$

Can we say the same thing for address and last name? No

Examples of FDs

First Name	Last Name	SSN	DOB	Address	Sex	Salary
Ahmad	Jabbar	987987987	12/3/79	980 Dallas, Houston, TX	M	25000
Jared	James	111111100	5/12/65	123 Peachtree, Atlanta, GA	M	85000
John	James	555555500	5/12/65	7676 Bloomington, Sacramento, CA	M	81000
Jill	Jarvis	666666601	10/1/93	6234 Lincoln, Chicago, IL	F	36000
Jon	Jones	111111101	11/4/86	111 Allgood, Atlanta, GA	M	45000
Kate	King	666666602	7/3/92	1976 Boone Trace, Chicago, IL	F	44000
Alice	King	666666604	7/3/92	556 Washington, Chicago, IL	F	38000
Sara	Riazi	334241992	11/4/86	5201 Fairview Rd., Charlotte, NC	F	83200

$$\text{SSN} \rightarrow \text{Address}$$

Two distinguished tuples cannot have different addresses while having the same SSN

$$\text{Last Name} \not\rightarrow \text{Address}$$

Can we say the something for address and last name? No

$$\text{Last Name} \not\rightarrow \text{DOB}$$

We cannot infer function dependencies from a snapshot of data

Armstrong's Axioms

Reflexivity/Trivial

$$\text{If } Y \subseteq X \text{ then } X \rightarrow Y$$

First Name, Last Name → Last Name

Augmentation

$$\text{If } X \rightarrow Y \text{ and } Z \subseteq R \text{ then } XZ \rightarrow YZ$$

First Name, Last Name, Address → Last Name, Address

Transitivity

$$\text{If } X \rightarrow Y \text{ and } Y \rightarrow Z \text{ then } X \rightarrow Z$$

SSN → Dnumber and
Dnumber → Dname then
SSN → Dname

Union

$$\text{If } X \rightarrow Y \text{ and } X \rightarrow Z \text{ then } X \rightarrow YZ$$

SSN → Dnumber and SSN → Last Name
then SSN → Dnumber, Last Name

Augmentation

$$X \rightarrow Y \Longrightarrow X \rightarrow YX$$

$$X \rightarrow Z \Longrightarrow YX \rightarrow YZ$$

$$\Longrightarrow X \rightarrow YZ$$

Augmentation

Transitivity

Inferring New FDs

Department	Category	Brand	Name	Color	Price
Appliance	Refrigerator	LG	LFDS22520S	White	1,549.0
Appliance	Refrigerator	LG	LFDS22520S	Metallic	1,749.0
Electronics	Tablet	Apple	2021 iPad 10.2-inch	Gray	449.0
Electronics	Headphone	Sony	WH-1000XM4	Silver	298.0
Electronics	Headphone	Sony	WH-1000XM4	Black	278.0

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

Reflexivity

$$\text{If } Y \subseteq X \text{ then } X \rightarrow Y$$

Augmentation

$$\text{If } X \rightarrow Y \text{ and } Z \subseteq R \text{ then } XZ \rightarrow YZ$$

Transitivity

$$\text{If } X \rightarrow Y \text{ and } Y \rightarrow Z \text{ then } X \rightarrow Z$$

Union

$$\text{If } X \rightarrow Y \text{ and } X \rightarrow Z \text{ then } X \rightarrow YZ$$

Inferring New FDs

Department	Category	Brand	Name	Color	Price
Appliance	Refrigerator	LG	LFDS22520S	White	1,549.0
Appliance	Refrigerator	LG	LFDS22520S	Metallic	1,749.0
Electronics	Tablet	Apple	2021 iPad 10.2-inch	Gray	449.0
Electronics	Headphone	Sony	WH-1000XM4	Silver	298.0
Electronics	Headphone	Sony	WH-1000XM4	Black	278.0

Reflexivity

If $Y \subseteq X$ then $X \rightarrow Y$

Augmentation

If $X \rightarrow Y$ and $Z \subseteq R$ then $XZ \rightarrow YZ$

Transitivity

If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

Union

If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name} → {department}

Inferring New FDs

Department	Category	Brand	Name	Color	Price
Appliance	Refrigerator	LG	LFDS22520S	White	1,549.0
Appliance	Refrigerator	LG	LFDS22520S	Metallic	1,749.0
Electronics	Tablet	Apple	2021 iPad 10.2-inch	Gray	449.0
Electronics	Headphone	Sony	WH-1000XM4	Silver	298.0
Electronics	Headphone	Sony	WH-1000XM4	Black	278.0

Reflexivity

$$\text{If } Y \subseteq X \text{ then } X \to Y$$

Augmentation

$$\text{If } X \to Y \text{ and } Z \subseteq R \text{ then } XZ \to YZ$$

Transitivity

$$\text{If } X \to Y \text{ and } Y \to Z \text{ then } X \to Z$$

Union

$$\text{If } X \to Y \text{ and } X \to Z \text{ then } X \to YZ$$

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name} → {department}

{name} → {category, brand, department}

Inferring New FDs

Department	Category	Brand	Name	Color	Price
Appliance	Refrigerator	LG	LFDS22520S	White	1,549.0
Appliance	Refrigerator	LG	LFDS22520S	Metallic	1,749.0
Electronics	Tablet	Apple	2021 iPad 10.2-inch	Gray	449.0
Electronics	Headphone	Sony	WH-1000XM4	Silver	298.0
Electronics	Headphone	Sony	WH-1000XM4	Black	278.0

Reflexivity

$$\text{If } Y \subseteq X \text{ then } X \to Y$$

Augmentation

$$\text{If } X \to Y \text{ and } Z \subseteq R \text{ then } XZ \to YZ$$

Transitivity

$$\text{If } X \to Y \text{ and } Y \to Z \text{ then } X \to Z$$

Union

$$\text{If } X \to Y \text{ and } X \to Z \text{ then } X \to YZ$$

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name} → {department}

{name} → {category, brand, department}

{name, color} → {category, brand, department, color}

Inferring New FDs

Department	Category	Brand	Name	Color	Price
Appliance	Refrigerator	LG	LFDS22520S	White	1,549.0
Appliance	Refrigerator	LG	LFDS22520S	Metallic	1,749.0
Electronics	Tablet	Apple	2021 iPad 10.2-inch	Gray	449.0
Electronics	Headphone	Sony	WH-1000XM4	Silver	298.0
Electronics	Headphone	Sony	WH-1000XM4	Black	278.0

Reflexivity

$$\text{If } Y \subseteq X \text{ then } X \to Y$$

Augmentation

$$\text{If } X \to Y \text{ and } Z \subseteq R \text{ then } XZ \to YZ$$

Transitivity

$$\text{If } X \to Y \text{ and } Y \to Z \text{ then } X \to Z$$

Union

$$\text{If } X \to Y \text{ and } X \to Z \text{ then } X \to YZ$$

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name} → {department}

{name} → {category, brand, department}

{name, color} → {category, brand, department, color}

{name} → {name}

Inferring New FDs

Department	Category	Brand	Name	Color	Price
Appliance	Refrigerator	LG	LFDS22520S	White	1,549.0
Appliance	Refrigerator	LG	LFDS22520S	Metallic	1,749.0
Electronics	Tablet	Apple	2021 iPad 10.2-inch	Gray	449.0
Electronics	Headphone	Sony	WH-1000XM4	Silver	298.0
Electronics	Headphone	Sony	WH-1000XM4	Black	278.0

Reflexivity

$$\text{If } Y \subseteq X \text{ then } X \rightarrow Y$$

Augmentation

$$\text{If } X \rightarrow Y \text{ and } Z \subseteq R \text{ then } XZ \rightarrow YZ$$

Transitivity

$$\text{If } X \rightarrow Y \text{ and } Y \rightarrow Z \text{ then } X \rightarrow Z$$

Union

$$\text{If } X \rightarrow Y \text{ and } X \rightarrow Z \text{ then } X \rightarrow YZ$$

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name} → {department}

{name} → {category, brand, department}

{name, color} → {category, brand, department, color}

{name} → {name}

{name, color} → {name, color}

Inferring New FDs

Department	Category	Brand	Name	Color	Price
Appliance	Refrigerator	LG	LFDS22520S	White	1,549.0
Appliance	Refrigerator	LG	LFDS22520S	Metallic	1,749.0
Electronics	Tablet	Apple	2021 iPad 10.2-inch	Gray	449.0
Electronics	Headphone	Sony	WH-1000XM4	Silver	298.0
Electronics	Headphone	Sony	WH-1000XM4	Black	278.0

Reflexivity

If $Y \subseteq X$ then $X \rightarrow Y$

Augmentation

If $X \rightarrow Y$ and $Z \subseteq R$ then $XZ \rightarrow YZ$

Transitivity

If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

Union

If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name} → {department}

{name} → {category, brand, department}

{name, color} → {category, brand, department, color}

{name} → {name}

{name, color} → {name, color}

?

Inferring New FDs

Department	Category	Brand	Name	Color	Price
Appliance	Refrigerator	LG	LFDS22520S	White	1,549.0
Appliance	Refrigerator	LG	LFDS22520S	Metallic	1,749.0
Electronics	Tablet	Apple	2021 iPad 10.2-inch	Gray	449.0
Electronics	Headphone	Sony	WH-1000XM4	Silver	298.0
Electronics	Headphone	Sony	WH-1000XM4	Black	278.0

Reflexivity

If $Y \subseteq X$ then $X \rightarrow Y$

Augmentation

If $X \rightarrow Y$ and $Z \subseteq R$ then $XZ \rightarrow YZ$

Transitivity

If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

Union

If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name} → {department}

{name} → {category, brand, department}

{name, color} → {category, brand, department, color}

{name} → {name}

{name, color} → {name, color}

{name, color} → {name, category, brand, department, color, price}

Superkey and Candidate key

Department	Category	Brand	Name	Color	Price
Appliance	Refrigerator	LG	LFDS22520S	White	1,549.0
Appliance	Refrigerator	LG	LFDS22520S	Metallic	1,749.0
Electronics	Tablet	Apple	2021 iPad 10.2-inch	Gray	449.0

R = {department, category, brand, name, color, price}

{name, color} $\rightarrow$ {department, category, brand, name, color, price}

{name, color} $\rightarrow$ R

=> {name, color} is a key

{name, color, category} $\rightarrow$ {department, category, brand, name, color, price}

{name, color, department} $\rightarrow$ {department, category, brand, name, color, price}

{name, color, department, category} $\rightarrow$ {department, category, brand, name, color, price}

Definition: For relation R , if we have $X \rightarrow R$ then X is a superkey of relation R .

X is minimal $\Rightarrow \nexists A \in R \text{ s.t. } X - \{A\} \rightarrow R$

Definition: For relation R , if we have $X \rightarrow R$ and X is minimal then X is a candidate key of relation R.

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

```
function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    C ← X
    repeat
        for f ∈ F do
            if LHS(f) ⊆ C then
                C ← C ∪ RHS(f)
            end if
        end for
    until C does not change
    return C
end function
```

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

$\{name, color\}^+$ = CLOSURE({name, color}, F)

function CLOSURE(X, F)
  //F is the set of FDs
  //X is the set of input attributes
  $C \leftarrow X$
  repeat
    for $f \in F$ **do**
      if LHS(f) $\subseteq C$ **then**
        $C \leftarrow C \cup \text{RHS}(f)$
      end if
    end for
  until C does not change
  return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C =

{name, color}

function CLOSURE(X, F)
  //F is the set of FDs
  //X is the set of input attributes
  $C \leftarrow X$
  repeat
    for $f \in F$ **do**
      if LHS(f) $\subseteq C$ **then**
        $C \leftarrow C \cup \mathrm{RHS}(f)$
      end if
    end for
  until C does not change
  return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C =

{name, color}

function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    $C \leftarrow X$
    repeat
        for $f \in F$ **do**
            if $\mathrm{LHS}(f) \subseteq C$ **then**
                $C \leftarrow C \cup \mathrm{RHS}(f)$
            end if
        end for
    until C does not change
    return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C =

{name, color, category}

function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    $C \leftarrow X$
    repeat
        for $f \in F$ **do**
            if LHS(f) $\subseteq C$ **then**
                $C \leftarrow C \cup \mathrm{RHS}(f)$
            end if
        end for
    until C does not change
    return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C =

{name, color, category}

function CLOSURE(X, F)
  //F is the set of FDs
  //X is the set of input attributes
  $C \leftarrow X$
  repeat
    for $f \in F$ **do**
      if LHS(f) $\subseteq C$ **then**
        $C \leftarrow C \cup$ RHS(f)
      end if
    end for
  until C does not change
  return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C = {name, color, category, department }

function CLOSURE(X, F)
  //F is the set of FDs
  //X is the set of input attributes
  $C \leftarrow X$
  repeat
    for $f \in F$ **do**
      if LHS(f) $\subseteq C$ **then**
        $C \leftarrow C \cup \mathrm{RHS}(f)$
      end if
    end for
  until C does not change
  return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C = {name, color, category, department }

```
function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    C ← X
    repeat
        for f ∈ F do
            if LHS(f) ⊆ C then
                C ← C ∪ RHS(f)
            end if
        end for
    until C does not change
    return C
end function
```

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C = {name, color, category, department, price }

function CLOSURE(X, F)
 //F is the set of FDs
 //X is the set of input attributes
 $C \leftarrow X$
 repeat
  for $f \in F$ **do**
   if LHS(f) $\subseteq C$ **then**
    $C \leftarrow C \cup \text{RHS}(f)$
   end if
  end for
 until C does not change
 return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C = {name, color, category, department, price }

function CLOSURE(X, F)
  //F is the set of FDs
  //X is the set of input attributes
  $C \leftarrow X$
  repeat
    for $f \in F$ **do**
      if LHS(f) $\subseteq C$ **then**
        $C \leftarrow C \cup \mathrm{RHS}(f)$
      end if
    end for
  until C does not change
  return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C = {name, color, category, department, price, brand }

function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    $C \leftarrow X$
    repeat
        for $f \in F$ **do**
            if LHS(f) $\subseteq C$ **then**
                $C \leftarrow C \cup \text{RHS}(f)$
            end if
        end for
    until C does not change
    return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name, color}$^+$ = CLOSURE({name, color}, F)

C = {name, color, category, department, price, brand }

{name, color} is a superkey

```
function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    C ← X
    repeat
        for f ∈ F do
            if LHS(f) ⊆ C then
                C ← C ∪ RHS(f)
            end if
        end for
    until C does not change
    return C
end function
```

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

$\{name\} \rightarrow \{category\}$
$\{category\} \rightarrow \{department\}$
$\{name, color\} \rightarrow \{price\}$
$\{name\} \rightarrow \{brand\}$

$\{name\}^+$ = CLOSURE($\{name\}$, F)

function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    $C \leftarrow X$
    repeat
        for $f \in F$ **do**
            if LHS(f) $\subseteq C$ **then**
                $C \leftarrow C \cup \text{RHS}(f)$
            end if
        end for
    until C does not change
    return C
end function

LHS({color, name} $\rightarrow$ {price}) = {color, name)

RHS({color, name} $\rightarrow$ {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name}$^+$ = CLOSURE({name}, F)

C =

{name}

```
function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    C ← X
    repeat
        for f ∈ F do
            if LHS(f) ⊆ C then
                C ← C ∪ RHS(f)
            end if
        end for
    until C does not change
    return C
end function
```

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X⁺.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name}⁺ = CLOSURE({name}, F)

C =

{name}

function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    $C \leftarrow X$
    repeat
        for $f \in F$ **do**
            if LHS(f) $\subseteq C$ **then**
                $C \leftarrow C \cup$ RHS(f)
            end if
        end for
    until C does not change
    return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

$\{name\}^+$ = CLOSURE({name}, F)

C =

{name, category}

function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    $C \leftarrow X$
    repeat
        for $f \in F$ **do**
            if LHS(f) $\subseteq C$ **then**
                $C \leftarrow C \cup \mathrm{RHS}(f)$
            end if
        end for
    until C does not change
    return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

$\{name\}^+$ = CLOSURE({name}, F)

C =

{name, category}

function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    $C \leftarrow X$
    repeat
        for $f \in F$ **do**
            if $\mathrm{LHS}(f) \subseteq C$ **then**
                $C \leftarrow C \cup \mathrm{RHS}(f)$
            end if
        end for
    until C does not change
    return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X⁺.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name}⁺ = CLOSURE({name}, F)

C = {name, category, department}

function CLOSURE(X, F)
  //F is the set of FDs
  //X is the set of input attributes
  $C \leftarrow X$
  repeat
    for $f \in F$ **do**
      if LHS(f) $\subseteq C$ **then**
        $C \leftarrow C \cup \text{RHS}(f)$
      end if
    end for
  until C does not change
  return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

$\{name\}^+$ = CLOSURE({name}, F)

C = {name, category, department}

```
function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    C ← X
    repeat
        for f ∈ F do
            if LHS(f) ⊆ C then
                C ← C ∪ RHS(f)
            end if
        end for
    until C does not change
    return C
end function
```

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name}$^+$ = CLOSURE({name}, F)

C = {name, category, department}

function CLOSURE(X, F)
  //F is the set of FDs
  //X is the set of input attributes
  $C \leftarrow X$
  repeat
    for $f \in F$ **do**
      if LHS(f) $\subseteq C$ **then**
        $C \leftarrow C \cup$ RHS(f)
      end if
    end for
  until C does not change
  return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

{name}$^+$ = CLOSURE({name}, F)

C = {name, category, department, brand}

function CLOSURE(X, F)
  //F is the set of FDs
  //X is the set of input attributes
  $C \leftarrow X$
  repeat
    for $f \in F$ **do**
      if LHS(f) $\subseteq C$ **then**
        $C \leftarrow C \cup \text{RHS}(f)$
      end if
    end for
  until C does not change
  return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X⁺.

F =

{name} $\rightarrow$ {category}
{category} $\rightarrow$ {department}
{name, color} $\rightarrow$ {price}
{name} $\rightarrow$ {brand}

{name}⁺ = CLOSURE({name}, F)

C = {name, category, department, brand}

```
function CLOSURE(X, F)
    //F is the set of FDs
    //X is the set of input attributes
    C ← X
    repeat
        for f ∈ F do
            if LHS(f) ⊆ C then
                C ← C ∪ RHS(f)
            end if
        end for
    until C does not change
    return C
end function
```

LHS({color, name} $\rightarrow$ {price}) = {color, name)

RHS({color, name} $\rightarrow$ {price}) = {price)

Closure

Definition: Given a set of FDs F and set of attributes X, we define closure of X under F as the set of all attributes that are functionally dependent on X. Closure of X is denoted as X^+.

F =

{name} → {category}
{category} → {department}
{name, color} → {price}
{name} → {brand}

$\{name\}^+$ = CLOSURE({name}, F)

C =

{name, category, department, brand}

{name} is not a key

function CLOSURE(X, F)
  //F is the set of FDs
  //X is the set of input attributes
  $C \leftarrow X$
  repeat
    for $f \in F$ **do**
      if $\mathrm{LHS}(f) \subseteq C$ **then**
        $C \leftarrow C \cup \mathrm{RHS}(f)$
      end if
    end for
  until C does not change
  return C
end function

LHS({color, name} → {price}) = {color, name)

RHS({color, name} → {price}) = {price)

Normalization and Normal Forms

- **Normalization**
  - The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

- **Normal forms:**
  - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

1NF

Test over atomicity of attributes

2NF

Test over fully functional dependencies on the candidate keys

3NF

Test over transitive dependencies on the candidate keys

BCNF

Test over dependency on the candidate keys

1st Normal Form (1NF)

Constraint: The attributes of a relation MUST be atomic

We cannot have composite or multivalued attributes

Example: Assume in our design the departments may have multiple locations, then DEPT_LOC is not 1NF

DEPT_LOC

Dname	Dnumber	Dmgr_ssn	Dlocations
Headquarters	1	888665555	{Houston}
Administration	4	987654321	{Stafford}
Research	5	333445555	{Bellaire, Houston, Sugarland}
Software	6	111111100	{Atlanta, Sacramento}
Hardware	7	444444400	{Milwaukee}

Approaches to normalize DEPT_LOC into 1NF:

1) Column expansion
2) Row expansion
3) Break down the original relation

Normalizing to 1st NF (Column expansion)

If the maximum number of values that multivalued attribute can take is fixed, construct multiple attributions to describe it

Dname	Dnumber	Dmgr_ssn	Dlocations
Headquarters	1	888665555	{Houston}
Administration	4	987654321	{Stafford}
Research	5	333445555	{Bellaire, Houston, Sugarland}
Software	6	111111100	{Atlanta, Sacramento}
Hardware	7	444444400	{Milwaukee}

Problem: We may have many NULL values.

Dname	Dnumber	Dmgr_ssn	Dloc1	Dloc2	Dloc3
Headquarters	1	888665555	Houston	**NULL**	**NULL**
Administration	4	987654321	Stafford	**NULL**	**NULL**
Research	5	333445555	Bellaire	Houston	Sugarland
Software	6	111111100	Atlanta	Sacramento	**NULL**
Hardware	7	444444400	Milwaukee	**NULL**	**NULL**

Normalizing to 1st NF (Row expansion)

Create new row for each value of the multivalued attribute

Dname	Dnumber	Dmgr_ssn	Dlocations
Headquarters	1	888665555	{Houston}
Administration	4	987654321	{Stafford}
Research	5	333445555	{Bellaire, Houston, Sugarland}
Software	6	111111100	{Atlanta, Sacramento}
Hardware	7	444444400	{Milwaukee}

Problem: We will have redundancy in data

$$\text{Dnumber} \nrightarrow \text{Dlocation}$$

Dname	Dnumber	Dmgr_ssn	Dlocation
Headquarters	1	888665555	Houston
Administration	4	987654321	Stafford
Research	5	333445555	Bellaire
Research	5	333445555	Houston
Research	5	333445555	Sugarland
Software	6	111111100	Atlanta
Software	6	111111100	Sacramento
Hardware	7	444444400	Milwaukee

Normalizing to 1st NF (Break down the original relation)

Decompose the original relation: one relation for other attributes, one relation for the key of the original relation and multivalued attribute

Dname	Dnumber	Dmgr_ssn	Dlocations
Headquarters	1	888665555	{Houston}
Administration	4	987654321	{Stafford}
Research	5	333445555	{Bellaire, Houston, Sugarland}
Software	6	111111100	{Atlanta, Sacramento}
Hardware	7	444444400	{Milwaukee}

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Houston
5	Sugarland
6	Atlanta
6	Sacramento
7	Milwaukee

Dname	Dnumber	Dmgr_ssn
Headquarters	1	888665555
Administration	4	987654321
Research	5	333445555
Software	6	111111100
Hardware	7	444444400

Prime Attribute

Definition: Attribute A of relation R is called a **prime attribute** if it is part of a candidate key of R

Dep-Loc

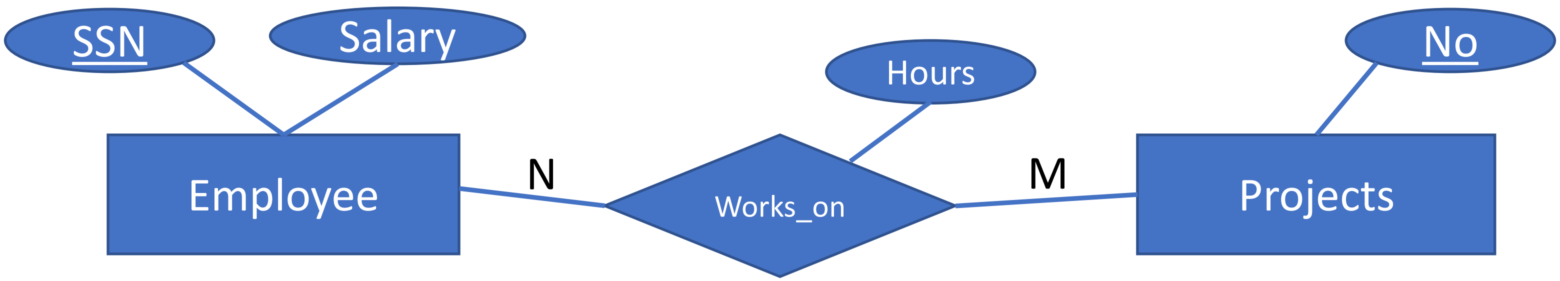
Dname	Dnumber	Dmgr_ssn	Dlocation
Headquarters	1	888665555	Houston
Administration	4	987654321	Stafford
Research	5	333445555	Bellaire
Research	5	333445555	Houston
Research	5	333445555	Sugarland
Software	6	111111100	Atlanta
Software	6	111111100	Sacramento
Hardware	7	444444400	Milwaukee

Candidate key: <u>Dnumber, Dlocation</u>

Prime attributes: Dnumber and Dlocation

Non-Prime attributes: Dname and Dmgr_ssn

Fully Functionally Dependency



Works_on

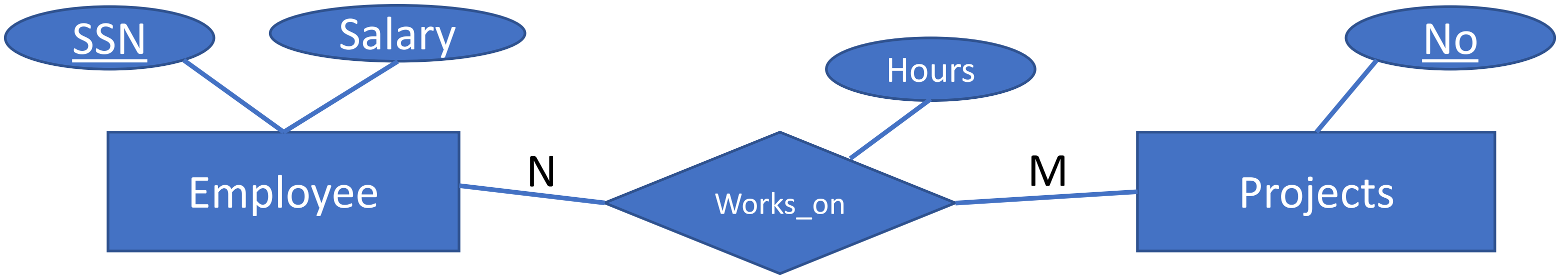
Essn	Pno	Hours
111111100	61	40
123456789	1	32.5
123456789	2	7.5
333445555	2	10
333445555	3	10

FDs:

Essn $\nrightarrow$ Hours

Pno $\rightarrow$ Hours

Fully Functionally Dependency



Works_on

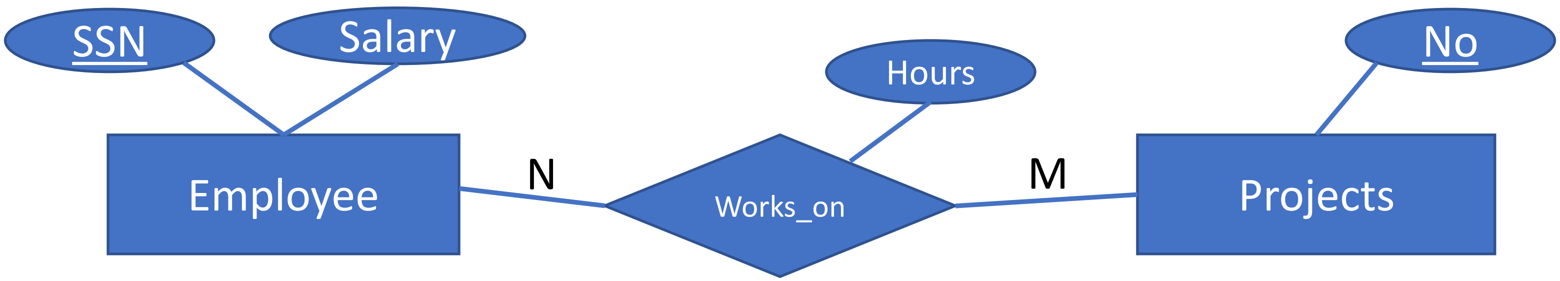
Essn	Pno	Hours
111111100	61	40
123456789	1	32.5
123456789	2	7.5
333445555	2	10
333445555	3	10

FDs:

$Essn \not\rightarrow Hours$

$Pno \not\rightarrow Hours$

Fully Functionally Dependency



Works_on

Essn	Pno	Hours
111111100	61	40
123456789	1	32.5
123456789	2	7.5
333445555	2	10
333445555	3	10

FDs:

Essn $\not\to$ Hours

Pno $\not\to$ Hours

Essn, Pno $\longrightarrow$ Hours

Definition: If $X \to Y$ then Y is **fully functionally dependent** on X if $\nexists A \in X \text{ s.t. } X - \{A\} \to Y$

Works_on

Essn	Pno	Hours	Salary
111111100	61	40	54000
123456789	1	32.5	78200
123456789	2	7.5	78200
333445555	2	10	67400
333445555	3	10	67400

FDs:

Essn, Pno $\longrightarrow$ Hours

Essn, Pno $\longrightarrow$ Salary

Essn $\longrightarrow$ Salary

Definition: If $X \to Y$ then Y is **Partially dependent** on X if $\exists A \in X \text{ s.t. } X - \{A\} \to Y$

2nd Normal Form (2NF)

Requirement: Each department has at most one office in each location

{Dnumber, Dlocation} is a candidate key of the relation

Dnumber, Dlocation → Dname

Dnumber → Dname

What does happen if a part of key identifies an attribute that is not prime?

Dep-Loc

Dname	Dnumber	Dmgr_ssn	Dlocation
Headquarters	1	888665555	Houston
Administration	4	987654321	Stafford
Research	5	333445555	Bellaire
Research	5	333445555	Houston
Research	5	333445555	Sugarland
Software	6	111111100	Atlanta
Software	6	111111100	Sacramento
Hardware	7	444444400	Milwaukee

We have redundancy in non-prime attributes that partially depend on a candidate key

Definition: A relation R is in 2NF if every nonprime attribute A in R **is fully functionally dependent** on **every** candidate key of R

Alternative Definition : A relation R is in 2NF if every nonprime attribute A in R is not **partially dependent** on any candidate key of R

Normalization into 2NF

Assume A,C → B and A → B and {A,C} is a candidate key and B is not a prime attribute. Construct two R1 and R2 such that

R1: A^+

R2: $R - A^+ \cup A$

Dep-Loc

Dname	Dnumber	Dmgr_ssn	Dlocation
Headquarters	1	888665555	Houston
Administration	4	987654321	Stafford
Research	5	333445555	Bellaire
Research	5	333445555	Houston
Research	5	333445555	Sugarland
Software	6	111111100	Atlanta
Software	6	111111100	Sacramento
Hardware	7	444444400	Milwaukee

Dnumber → Dname

R1: $\{Dnumber\}^+$

Dname	Dnumber	Dmgr_ssn
Headquarters	1	888665555
Administration	4	987654321
Research	5	333445555
Software	6	111111100
Hardware	7	444444400

$R - \{Dnumber\}^+ \cup \{Dnumber\}$

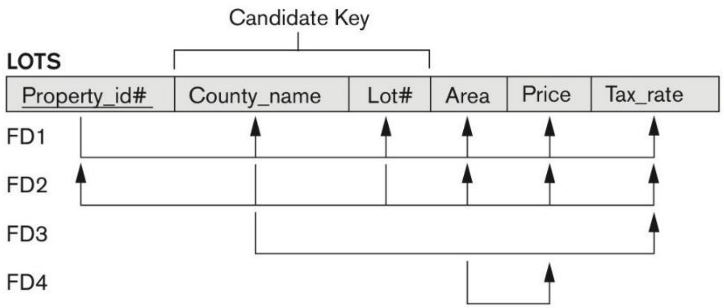
Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Houston
5	Sugarland
6	Atlanta
6	Sacramento
7	Milwaukee

Practice

Do we have a partial dependency on any key?

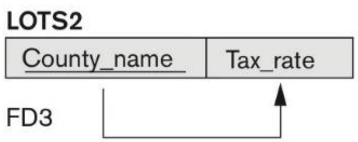
County_name, Lot# → Tax_rate

County_name → Tax_rate

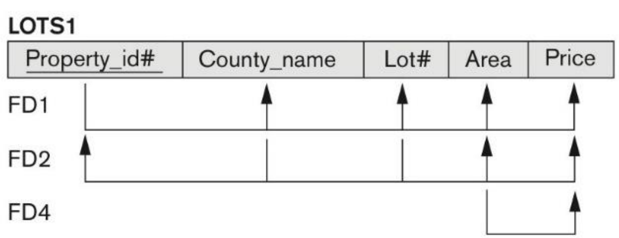


How to normalize into 2NF?

Closure(County_name) = {county_name, tax_rate}

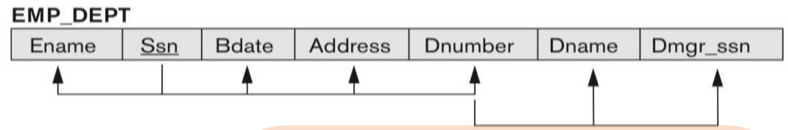


R – Closure(County_name) ∪ {County_name}



Third Normal Form

Transitive functional dependency: a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$



EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Bacher, Red, A	666666613	5/21/80	196 Elm Street, Miami, FL	8	Sales	555555500
Ball, Nandita, K	555555501	4/29/89	222 Howard, Sacramento, CA	6	Software	111111100
Bender, Bob, B	666666600	3/24/84	8794 Garfield, Chicago, IL	8	Sales	555555500
Best, Alec, C	444444402	11/10/87	233 Solid, Milwaukee, WI	7	Hardware	444444400
English, Joyce, A	453453453	7/31/83	5631 Rice Oak, Houston, TX	5	Research	333445555
Jabbar, Ahmad, V	987987987	3/29/79	980 Dallas, Houston, TX	4	Administration	987654321
Narayan, Ramesh, K	666884444	9/15/52	971 Fire Oak, Humble, TX	5	Research	333445555

SSN is the primary key and

SSN→ Dname

Can be derived from

SSN→ Dnumber and Dnumber→ Dname

Therefore, DName is transitively dependent on the primary key.

Definition: R is in 3NF if it is in 2NF and has **no transitive functional dependencies** on the **primary key**.
Key -> Non_Key -> Non_key

General Definition of 3NF

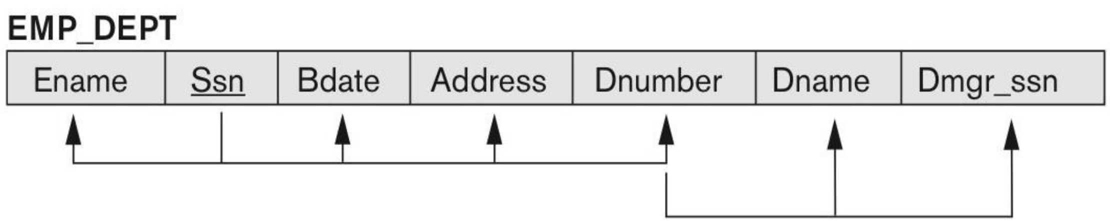
Definition: A relation R is in 3NF if

for **all FDs** A → B ∈ F that **hold on R** at least **one** of the following conditions is **true:**

- A → B is trivial (B ⊆ A),
- A is a **superkey** for R (Closure(A) == R)
- B is a **prime attribute** of table (B is part of candidate-key)

A ⟶ B

Super key Or **Prime attribute**



✓ SSN⟶ Ename — SSN is Superkey

✓ SSN⟶ Dnumber — SSN is Superkey

✗ Dnumber⟶ Dname — Dnumber is not Superkey, Dname is not prime

General definition of 3NF

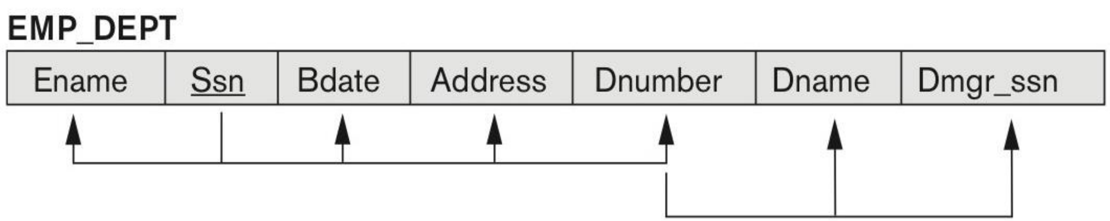
Definition: A relation R is in 3NF if

for **all FDs** A → B ∈ F that **hold on R** at least **one** of the following conditions is **true:**

- A → B is trivial (B ⊆ A),
- A is a **superkey** for R (Closure(A) == R)
- B is a **prime attribute** of table (B is part of candidate-key)

A ⟶ B

Super key Or **Prime attribute**



✓	SSN⟶ Ename	SSN is Superkey
✓	SSN⟶ Dnumber	SSNis Superkey
✗	Dnumber⟶ Dname	Dnumber is not Superkey, Dname is not prime

Practice 3NF

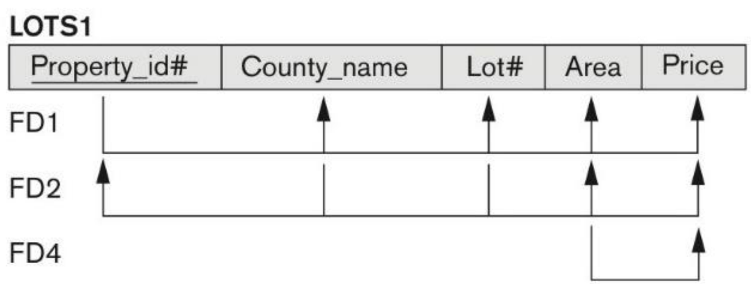
Definition: A relation R is in 3NF if

for **all FDs** A → B ∈ F that **hold on R** at least one of the following conditions is **true:**

- A → B is trivial (B ⊆ A),
- A is a superkey for R (Closure(A) == R)
- B is a **prime attribute** of table (B is part of candidate-key)

A ⟶ B

Super key Or **Prime attribute**

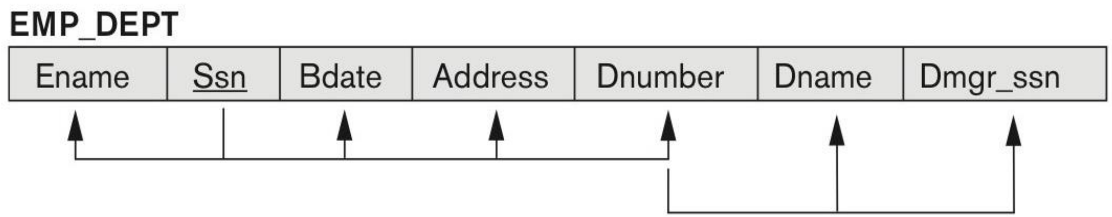


✓ County_name, Lot# ⟶ Price — {County_name, Lot#} is Superkey

✓ Property_id#⟶ County_name — Property_id# is Superkey and County_name is prime

✗ Area⟶ Price — Area is **not** Superkey and Price is **not** prime

Decomposition into 3NF

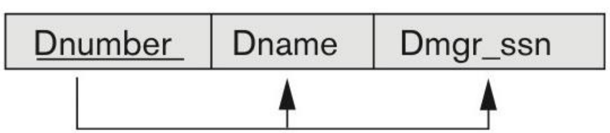


✗ Dnumber→ Dname Violating FD

R1: {Dnumber}$^+$

R2: R - R1 ∪ {Dnumber}

✓ 3NF

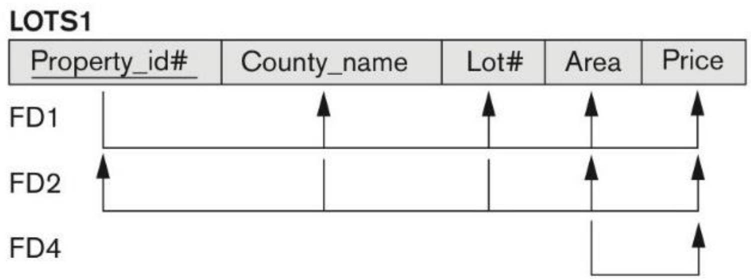


✓ 3NF

Assume A →B violates 3NF.

1) Construct two R1 and R2 such that R1: A$^+$ and R2: R – R1 ∪ A
2) Check R1 and R2 for 3NF

Decomposition into 3NF



Area→ Price

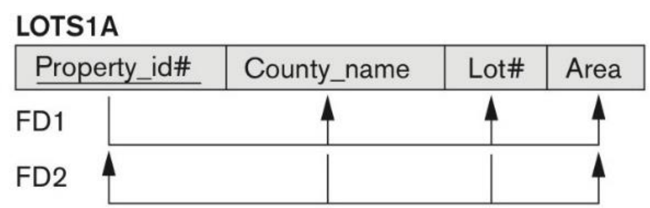
Violating FD

R1: Closure({Area})

R2: R – R1 ∪ {Area}

3NF

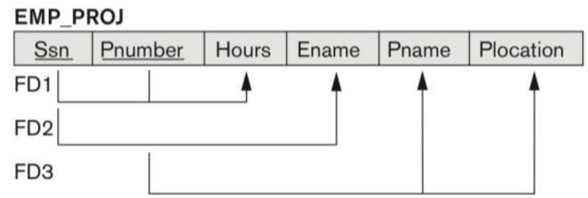
3NF



Assume A →B violates 3NF.

1) Construct two R1 and R2 such that R1: Closure(A) and R2: R – R1 ∪ A
2) Check R1 and R2 for 3NF

Practice

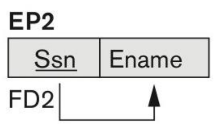


✓ SSN, Pnumber→ Hours {SSN, Pnumber} is Superkey

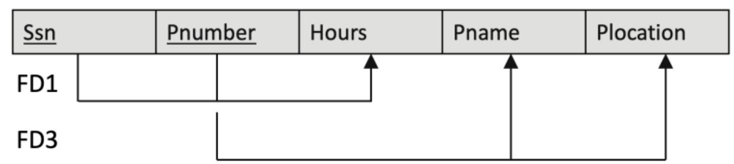
✗ SSN→ Ename SSN is not Superkey

R1: Closure({SSN})

✓ 3NF



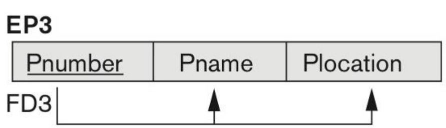
R2: R - R1 ∪ {SSN}



✗ Pnumber → Pname Violating FD

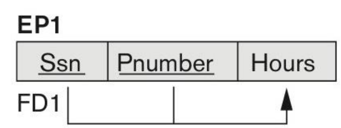
R3: Closure({Pnumber})

✓ 3NF



R4: R2- R3 ∪ {Pnumber}

✓ 3NF



Assume A →B violates 3NF.

1) Construct two R1 and R2 such that R1: Closure(A) and R2: R – R1 ∪ A
2) Check R1 and R2 for 3NF

BCNF

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

{Student, Course} is superkey

✓ Student, Course → Instructor

{Student, Course} is superkey

✓ Instructor → Course

{Instructor} is not superkey but course is a prime attribute

The TEACH is in 3NF but still we have redundancy

BCNF

Definition: A relation R is in ~~3NF~~ if

for **all FDs** A → B ∈ F that **hold on R** at least one of the following conditions is **true:**

- A → B is trivial (B ⊆ A),
- A is a **superkey** for R (Closure(A) == R)
- ~~B is a **prime attribute** of table (B is part of candidate-key)~~

A → B

Super key Or ~~**Prime attribute**~~

BCNF Decomposition Algorithm

Main idea: define "good" and "bad" FDs as follows:

- $X \rightarrow A$ is a "*good FD" if X is a (super) key*
  I.e., A is the set of all attributes
- Else, $X \rightarrow A$ is a *"bad FD"*
  I.e., X functionally determines *some* attributes; other attributes can be duplicate/anomalies
- We will try to eliminate the "bad" FDs!

BCNFDecomp(R):
For each FD $X \rightarrow A$ in R
$X^+ \leftarrow$ the closure set of X;
If $X^+ \neq X$ and $X^+ \neq$ [all attributes]

Check for non-trivial "bad" FDs, i.e. is not a superkey, using closures

Then decompose R into $\mathbf{R_1}$ **:(X^+)** and $\mathbf{R_2}$**:($X \cup (R - R_1)$)**

R2: Rest of attributes not in R1 plus X

If (not found) **Then Return** R

If no "bad" FDs found, in BCNF!

Return BCNFDecomp(R_1), BCNFDecomp(R_2)

Check R1 and R2 for being in BCNF

Example

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

Student, Course → Instructor

Instructor → Course

R: {Student, Course, Instructor}

$\{Student, Course\}^+ =$ {Student, Course, Instructor} = R ✓

$\{Instructor\}^+ =$ {Course, Instructor} ✗

R1: {Course, Instructor}

R2: {Instructor} ∪ (R – R1) = {Student, Instructor}

R1: {Instructor, Course}

Instructor	Course
Mark	Database
Navathe	Database
Ammar	Operating Systems
Schulman	Theory
Ahamad	Operating Systems
Omiecinski	Database

Instructor → Course

$\{Instructor\}^+ =$ {Course, Instructor} ✓

BCNF

R2: {Student, Instructor}

Student	Instructor
Narayan	Mark
Smith	Navathe
Smith	Ammar
Smith	Schulman
Wallace	Mark
Wallace	Ahamad
Wong	Omiecinski
Zelaya	Navathe
Narayan	Ammar

BCNF

We lost
Student, Course → Instructor
But since instructor is the primary key of the other relation, we won't generate spurious tuples

Normalizing into BCNF

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield
Joe	987-65-4321	908-555-1234	Westfield

R = {SSN, Name, City, PhoneNumber}

{SSN,PhoneNumber} $\rightarrow$ {SSN, Name, City, PhoneNumber}
{SSN} $\rightarrow$ {Name, City}

{SSN,PhoneNumber}$^+$ =
{SSN, Name, City, PhoneNumber} = R ✓

{SSN}$^+$ = {SSN, Name, City} $\neq$ R ✗

$\Longrightarrow$ **Not** in BCNF

R1 = {SSN}$^+$ = {SSN, Name, City}

SSN	Name	City
123-45-6789	Fred	Seattle
987-65-4321	Joe	Madison

{SSN} $\rightarrow$ {Name, City}

{SSN}$^+$ = {SSN, Name, City} = R1 =>Superkey

BCNF

R2 = {SSN} ∪ (R-R1) = {SSN, PhoneNumer}

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

{SSN,PhoneNumber} $\rightarrow$ {SSN, PhoneNumber}

{SSN,PhoneNumber}$^+$ = {SSN, PhoneNumber}

BCNF

Practice: Is This Relation In BCNF?

R = {Property_id#, County_name, Lot#, Area, Price, Tax_rate}

{County_name}$^+$ = {County_name, Tax_rate} => violation

Property_id# ⟶ {County_name, Lot#, Area, Price, Tax_rate}

County_name, Lot# ⟶ {Property_id#, Area, Price, Tax_rate}

County_name ⟶ Tax_rate

Area ⟶ County_name

R1 = {County_name, Tax_rate}

County_name ⟶ Tax_rate

{County_name}$^+$ = R1 => superkey

BCNF

R2 = {County_name, Property_id#, Lot#, Area, Price }

Property_id# ⟶ {County_name, Lot#, Area, Price}

County_name, Lot# ⟶ {Property_id#, Area, Price}

Area ⟶ County_name

{Property_id#}$^+$ = R2 => superkey

{County_name, Lot#}$^+$ = R2 => superkey

{Area}$^+$ = {Area, County_name} => violation

R3 = {Area, County_name }

Area ⟶ County_name

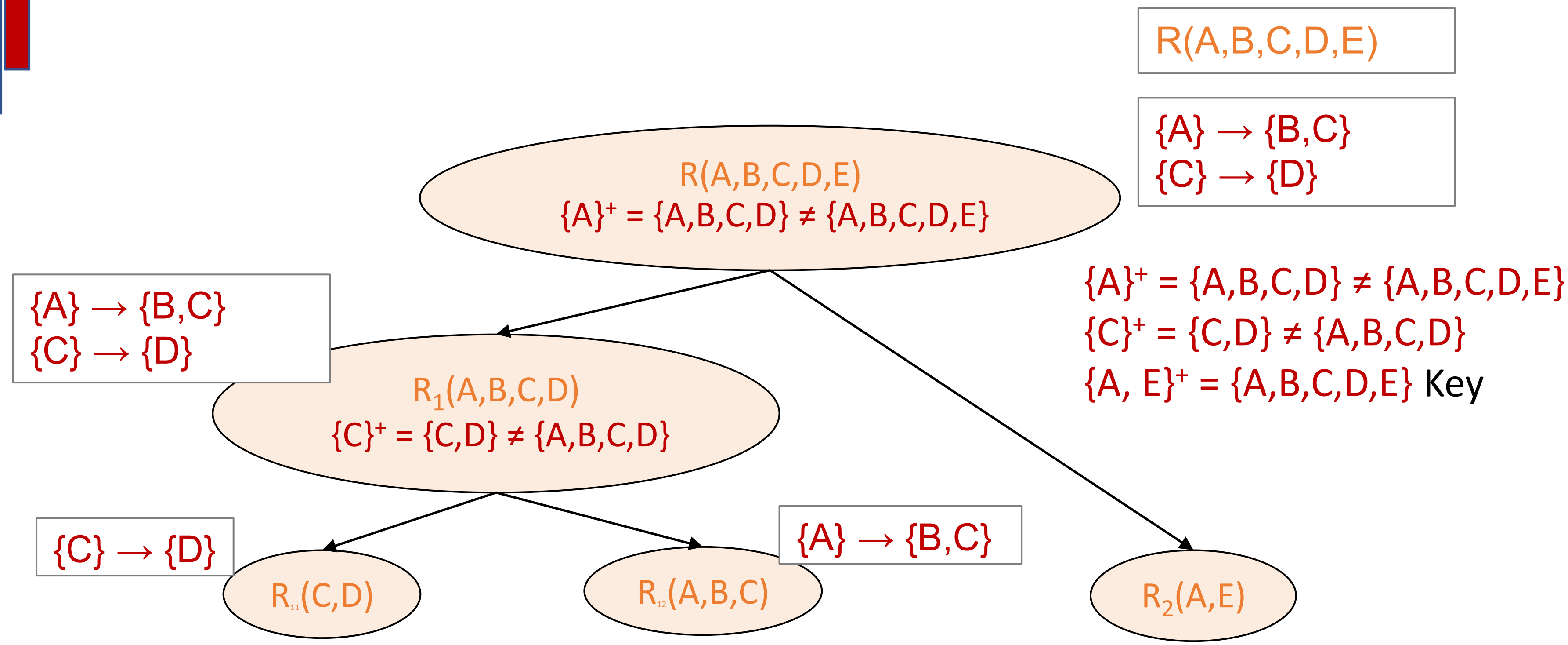
BCNF

R4 = {Area, Property_id#, Lot#, Price }

Property_id# ⟶ {Lot#, Area, Price}

BCNF

Practice BCNF



A Problem with BCNF

Unit	Company	Product
…	…	…

Unit	Company
…	…

{Unit} → {Company}

Unit	Product
…	…

{Unit} → {Company}
{Company,Product} → {Unit}

$\{\text{Unit}\}^+$ = {Unit, Company}

$\{\text{Company, Product}\}^+$ = {Unit, Product,Company}

We do a BCNF decomposition on a "bad" FD:
$\{\text{Unit}\}^+$ = {Unit, Company}

We lose the FD {Company,Product} → {Unit}!!

A Problem with BCNF

Original:

Unit	Company	Product
Galaxy	Samsung	S20
iPhone	Apple	iPhone 14

{Company,Product} → {Unit}

$\{Company, Product\}^{+}$ = {Unit, Product,Company}

New tuple <Vision, Apple, iPhone 14> violates key

Decomposed:

<u>Unit</u>	Company
Galaxy	Samsung
iPhone	Apple

Unit	Product
iPhone	iPhone 14
Galaxy	S20

No problem so far.
All *local* FD's are satisfied.

{Unit} → {Company}

Unit	Company	Product
Galaxy	Samsung	S20
iPhone	Apple	iPhone 14
Vision	Apple	iPhone 14

Violates the FD {Company,Product} → {Unit}!!

Checking the lost FD by joining the decomposed relations is expensive, thus we have a trade-off between redundancy and performance.

Denormalization

Storing relations in denormalized form can be considered for performance purposes.

- Infrequent updates
- Complex queries over multiple tables

Denormalized relations can speed up queries by avoiding join, however, it should be only considered where most of the queries are read rather than update.

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Houston
5	Sugarland
6	Atlanta
6	Sacramento
7	Milwaukee

Dname	Dnumber	Dmgr_ssn
Headquarters	1	888665555
Administration	4	987654321
Research	5	333445555
Software	6	111111100
Hardware	7	444444400

2NF

Dname	Dnumber	Dmgr_ssn	Dlocation
Headquarters	1	888665555	Houston
Administration	4	987654321	Stafford
Research	5	333445555	Bellaire
Research	5	333445555	Houston
Research	5	333445555	Sugarland
Software	6	111111100	Atlanta
Software	6	111111100	Sacramento
Hardware	7	444444400	Milwaukee

1NF