Lecture-13

Functional Dependencies - Determining Keys

CS211 - Introduction to Database

Two principles of database design

1. Capture all the information needed by the underlying application
   - Collect user requirements
   - Translates user requirements into a conceptual schema (ER-Model)

     -Entity

     -Attribute

     -Relationship

     -Key

2. Design the database with little redundancy/anomaly
   - Normalization

Different Keys and their meaning

R =

A K1 (1)	B K2(1)	C K3(2)	D	E	F	G	H K4(2)

- **Super Key** is the set of one or more column (ie attributes) which uniquely identifies a record.
  - (A, B, D, E)
  - (C, H, F, G)
- **Candidate key** is a minimal Super key.(it mean we cant remove any attributes from it otherwise it will not remain Super key anymore).
  - (A, B)
  - (C, H)
- **Primary Key** is a arbitrary selected Candidate key. There must be only One primary key. (A, B)
- **Alternate Keys** are the other candidate keys which are not chosen as the Primary key.
  - (C, H)
- If Primary Key have more then one column (or attributes) ,it is called **Composite Key**.
  - (A, B)

Keys

Employee {Emp_Code, Emp_Number,Emp_Name}

Super keys: All of the following sets are able to uniquely identify rows of the employee table.

{Emp_Code}
{Emp_Number}
{Emp_Code, Emp_Number}
{Emp_Code, Emp_Name}
{Emp_Code, Emp_Number, Emp_Name}
{Emp_Number, Emp_Name}

Candidate keys: They are the minimal super keys with no redundant attributes.

{Emp_Code}
{Emp_Number}

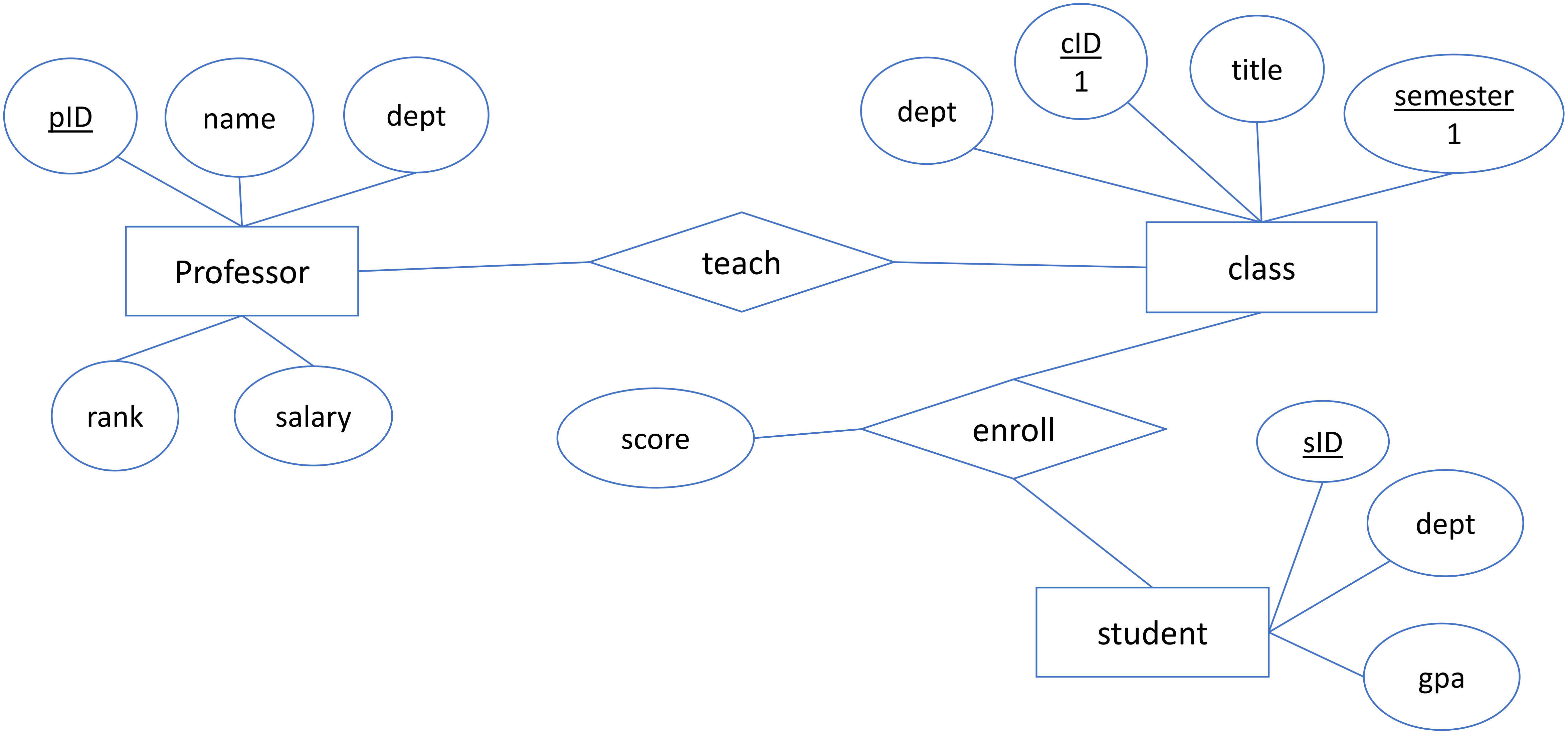
Primary key: The designer can select any candidate key as the primary key.

{Emp_Code}

Alternate keys: {candidate keys} - primary key → alternate keys.

{Emp_Number}

Database Redundancy



Redundancy and Anomalies

Student Relation

Sid is the Primary Key

Sid	Name	Credits	Dept	Building	Room_no	HOD	……….
1	John	5	CS	B1	101	-	
2	Adam	8	CS	B1	101	-	
3	Jiya	9	DS	B2	201	-	
4	Salim	9	DS	B2	201	-	
5	Xi	7	Civil	B1	110	-	
6	Chen	6	EC	B2	115	-	
7	Rahul	8	Civil	B1	120	-	
8	Allan	9	CS	B1	101	-	
NULL	NULL	NULL	ME	B2	120	-	

Redundant Storage

Update Anomaly

Deletion Anomaly

Insertion Anomaly

Functional Dependencies

Dependency

X->Y

X-determinant , Y-dependent

X	Y
5	25
2	4
3	9

$x \rightarrow y^2$

if X is unique, X->Y is always TRUE

What is Dependency?

X->Y   X-determinant   Y-dependent

X	Y
5	25
2	4
3	9
5 2	? ?

x -> y^2

Definition: Functional Dependencies

- A functional dependency (FD) has the form of X → Y, where X and Y are **sets of attributes**.

- FD means that whenever two tuples are identical on all the attributes in X, ie, t1[X]=t2[X], they must also be identical on all the attributes in Y ie, t1[Y]=t2[Y].
  - **Each possible value of X correspond to exactly one value of Y**

FD: $X \rightarrow Y$

if $t_1[X]=t_2[X]$

then $t_1[Y]=t_2[Y]$

A	B	C
1	2	3
4	2	3
5	3	3

FDs:

- $A \rightarrow B$
  YES
- $B \rightarrow A$
  NO
- $B \rightarrow C$
  YES
- $C \rightarrow B$
  NO

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b3	c2	d3
a3	b3	c2	d4

FDs:

- $A \rightarrow B$
  NO
- $A \rightarrow C$
  YES
- $A \rightarrow D$
  NO
- $B \rightarrow A$
  NO
- $B \rightarrow C$
  NO
- $B \rightarrow D$
  NO
- $D \rightarrow B$
  YES

FD: $X \rightarrow Y$

if $t_1[X]=t_2[X]$

then $t_1[Y]=t_2[Y]$

FD: X→Y

if $t_1[X]=t_2[X]$

then $t_1[Y]=t_2[Y]$

cID	Title	Semester	Dept
cs211	Database	2018su	cs
cs211	Database	2018s	cs
cs211	Database	2017s	ir

FDs:

- cID → title

  YES

- title → dept

  NO

- cID, semester → dept

  YES

- cID, semester → cID, dept

  YES

FD: X→Y

if $t_1[X]=t_2[X]$

then $t_1[Y]=t_2[Y]$

<u>sId</u>	Name	Marks	Dept	Course
1	a	78	CS	C1
2	b	60	EE	C1
3	a	78	CS	C2
4	b	60	EE	C3
5	c	80	IT	C3
6	d	80	EC	C2

- sId → Name
  YES
- Name → sId
  NO
- sId → Marks
  YES
- Dept→ Course
  NO
- Course → Dept
  NO
- Name→ Marks
  YES
- {sId, Name} → Dept
  YES
- {Name, Marks} → Dept
  YES
- {Name, Matrks} → {Dept, Course}
  No

Self Study

FD: X→Y

if $t1[X]=t2[X]$

then $t1[Y]=t2[Y]$

Check if FDs correct?

- sId → {Name, Marks}
- Name → Course
- {sId, Marks} → Dept
- {Dept, Course} →Name
- {Name, Matrks,Dept} → sId

sId	Name	Marks	Dept	Course
1	a	78	CS	C1
2	b	60	EE	C1
3	a	78	CS	C2
4	b	60	EE	C3
5	c	80	IT	C3
6	d	80	EC	C2

Propose possible FDs

customer(cID, cName, cGroup, phone, productID, productName, quantity, dateArrival)

$$cID \rightarrow (cName, cGroup, phone)$$

$$productID \rightarrow productName$$

$$(cID, productID, dateArrival) \rightarrow quantity$$

$$(cID, productID, dateArrival) \rightarrow (quantity, cName, cGroup, phone)$$

…

Propose possible FDs

employee(eID, eName, birthDate, phone, highestQualification, graduateSchool, dateTrain, courseTrain)

$$eID \rightarrow eName$$

$$eID \rightarrow birthDate$$

$$eID \rightarrow phone$$

$$eID \rightarrow (highestQulification, graduateSchool)$$

$$(eID, dateTrain) \rightarrow courseTrain$$

…

Facts of FD

- Ideally, we **do not want to miss** any FD, i.e., we want to obtain a complete set of FD that is as large as possible.

- However, in practice, FD collection is a **difficult process**. No one can guarantee always discovering all FDs.

- In practice, it is often the case that some FDs are **easier to see**, while others are more subtle and **harder to observe**.
    - Some of the subtle FDs can be derived from easy ones.
    - Some may not be discovered.

Uses of FDs

- FDs are a form of **integrity constraint** in relational databases.

- FDs promote **data correctness**, **consistency**, and **integrity**.

- FDs help to create "good" database schema where **data redundancy** (repetition of data) **is minimized**.

- FDs help in **Normalization** of database.

  → **Normalization is a data organization technique used to prevent redundancy, insertion, update and deletion anomalies.**

- FDs allows us to **identify poor designs**.

- FDs are used to determine **keys**.

Armstrong's Axioms (A set of inference rules used to infer all the functional dependencies on a relational database)

employee(eID, firstName, lastName, address, dept, position, salary)

Relation R={A1, A2, ... An}, and α→β is a FD in R

- A1 Reflexivity rule: X→Y if Y⊆X

$$(firstName, lastName) \to firstName$$

- A2 Augmentation rule: if X→Y, then XZ→YZ

$$position \to salary \quad \Longrightarrow \quad (position, eID) \to (salary, eID)$$

- A3 Transitivity rule: if X→Y and Y→Z, then X→Z

These rules are:

- sound (generate only functional dependencies that actually hold), and
- complete (generate all functional dependencies that hold).

Armstrong's Lemmas (Intermediate theorems: *It is possible to use Armstrong's axioms to prove that these rules are sound*)

Relation R={A1, A2, … An}, and $\alpha \rightarrow \beta$ is a FD in R

- Union rule: if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- Pseudo Transitivity: if $X \rightarrow Y$ and $YW \rightarrow Z$, then $XW \rightarrow Z$
- Decomposition rule: if $X \rightarrow Y$ and $Z \subseteq Y$, then $X \rightarrow Z$

Armstrong's Lemmas (Intermediate theorems: *It is possible to use Armstrong's axioms to prove that these rules are sound*)

- Union rule: if X→Y and X→Z, then X→YZ

$eID \rightarrow (firstName, lastName)$

$eID \rightarrow address$

$\Rightarrow$ $eID \rightarrow (firstName, lastName, address)$

- Pseudo Transitivity: if X→Y and WY→Z, then XW→Z

$eID \rightarrow (firstName)$

$(lastName, firstName) \rightarrow address$

$\Rightarrow$ $(lastname, eID) \rightarrow address$

- Decomposition rule: if X→Y and Z⊆Y, then X→Z

$eID \rightarrow (firstName, lastName)$

$(firstName) \subseteq (firstName, lastName)$

$\Rightarrow$ $(eID) \rightarrow firstName$

Determining the candidate Key

R(A B C D), F={A → C, B → D}, is {A, B} a candidate key?

Step1: Find if {A,B} is a Super key

$does\ AB \rightarrow ABCD\ ?$

$A \rightarrow C$

$B \rightarrow D$

$AAB \rightarrow ABC$

$BABC \rightarrow ABCD$

$AB \rightarrow ABC$

$ABC \rightarrow ABCD$

$AB \rightarrow ABCD$

Step2: Find if {A,B} is a Candidate key

$is\ A\ a\ superkey\ ?\ no$

$is\ B\ a\ superkey\ ?\ no$

$\boldsymbol{so}$

$\boldsymbol{AB\ is\ a\ candidate\ key}$

Determining the candidate Key

R(A B C D E), F={AB → C, AE → D, D → B}, prove {AE} is a candidate key.

$AE \rightarrow D$

$D \rightarrow B$

⇒

$AE \rightarrow B$

$AE \rightarrow D$

⇒

$AE \rightarrow BD$

$AB \rightarrow C$

⇒

$AEAE \rightarrow BDAE$

$ABDE \rightarrow CDE$

$AE \rightarrow ABDE$

$ABDE \rightarrow CDE$

⇒

$AE \rightarrow CDE$

$AE \rightarrow ABDE$

⇒

$\boldsymbol{AE \rightarrow ABCDE}$

Definition: Closure of attributes

- Let $\boldsymbol{\alpha}$ be a set of attributes.
- **We call the set of all attributes functionally determined by $\boldsymbol{\alpha}$ under a set *F* of functional dependencies the closure of $\boldsymbol{\alpha}$ under *F*.**
- We denote it by $\boldsymbol{\alpha}^+$.

Given F:

1. cID → title
2. title → dept
3. {cID, semester} → dept
4. {cID, semester} → {cID, semester]

$\boldsymbol{\alpha}$ ={cID}

$\boldsymbol{\alpha}^+$?

$$\boldsymbol{\alpha}^+ = \{cID\}$$
$$\boldsymbol{\alpha}^+ = \{cID, title\}$$
$$\boldsymbol{\alpha}^+ = \{cID, title, dept\}$$

Example of Closure

Given F:

1. cID → title
2. title → dept
3. {cID, semester} → dept
4. {cID, semester} → {cID, semester}

$\boldsymbol{\alpha}$ ={cID, semester}

$\boldsymbol{\alpha}^{+}$?

$$\boldsymbol{\alpha}^{+} = \{cID, semester\}$$

$$\boldsymbol{\alpha}^{+} = \{cID, semester, title\}$$

$$\boldsymbol{\alpha}^{+} = \{cID, semester, title, dept\}$$

Finding the Closure of attributes

Given F:

1. cID → title
2. title → dept
3. {cID, semester} → dept
4. {cID, semester} → {cID, semester}

$\boldsymbol{\alpha}$ ={cID, semester}

$\boldsymbol{\alpha}^{+}$?

$\boldsymbol{\alpha}^{+} = \{cID, semester\}$

$\boldsymbol{\alpha}^{+} = \{cID, semester, title\}$

$\boldsymbol{\alpha}^{+} = \{cID, semester, title, dept\}$

algorithm (**F**, $\boldsymbol{\alpha}$)

/* ***F*** is a set of FDs, and $\boldsymbol{\alpha}$ is an attribute set */

1. $\boldsymbol{\alpha}^{+} = \boldsymbol{\alpha}$

2. **while** *F* has a FD *A* → *B* such that $A \subseteq \boldsymbol{\alpha}^{+}$ **do**

    $\boldsymbol{\alpha}^{+} = \boldsymbol{\alpha}^{+} \cup B$

    remove *A* → *B* from *F*

3. return $\boldsymbol{\alpha}^{+}$ /* closure of $\boldsymbol{\alpha}$ */

Example

R = {A,B,C,D,E,G}

Given F:

{A,B}→{C}

{C}→{A}

{B,C}→{D}

{A,C,D}→{B}

{D}→{E,G}

{B,E}→{C}

{C,G}→{B,D}

{C,E}→{A,G}}

α ={B, D}, α^+ ?

$\alpha^+ = \{B, D\}$

$\alpha^+ = \{B, D, E, G\}$ since {D}→{E,G}

$\alpha^+ = \{B, C, D, E, G\}$ since {B,E}→{C}

$\alpha^+ = \{A, B, C, D, E, G\}$ since {C}→{A}

Candidate key

- So far we have been specifying candidate keys based on our preferences.
- In fact, candidate keys are not up to us at all.
- **Instead, they are uniquely determined by the set *F* of functional dependencies.**
- **A candidate key is a set *X* of attributes in *R* such that X^+ includes all the attributes in *R*. There is no proper subset *Y* of *X* such that Y^+ includes all the attributes in *R*.**

Example – Candidate Key

Consider a table R(A,B,C,D), and that F = {A → B, B → C}.

1. *A* is not a candidate key, because $A^+ = \{A,B,C\}$ which does not include *D*.
2. *AD* is a candidate key because $AD^+ = \{A,B,C,D\}$.
3. *ABD* is not a candidate key even though $ABD^+ = \{A,B,C,D\}$. This is because $AD^+ = \{A,B,C,D\}$.

To check for all the candidate keys of a relation *R* having *N* attributes, we need to find $\approx 2^n$ closures.

Identifying all the candidate Keys (Trick-1)

Consider a table R(A,B,C,D, E} and F = {A → B, D → E}.

Step -1: Determining the prime attributes (The attributes present in candidate keys are the prime attributes)

{ABCDE}$^+$ = {ABCDE} So, **[ABCDE]** is the **super-key**.

- Since A→B, we can remove B from the super-key. So, **[ACDE]** is the **super-key**.
- Since D→E, we can remove E from the super-key. So, **[ACD]** is the **super-key**.
- Now, the **proper subsets** of the super-key [ACD] are [A], [C], [D], [AC], [AD], [CD] and none of whose **closures** determine all the attributes {ABCDE} of R. **So, the super-key [ACD] is a candidate-key**.

The prime attributes are {A,C, D}

Step -2: Check if prime attributes are on the RHS of any FDs

- If NO, then we have found the only available candidate-key.

Identifying all the candidate Keys (Trick-2)

Consider a table R(A,B,C,D) and F = {A → B, B → C, C → A}.

Step -1: Determining the prime attributes

$\{ABCD\}^+ = \{ABCD\}$ So, **[ABCD]** is the **super-key**.

- Since A→B, we can remove B from the super-key. So, **[ACD]** is the **super-key**.
- Since B→C, using transitivity property (A → B, B → C So, A → C). We can remove C from the super-key. So, **[AD]** is the super-key.
- Now, the **proper subsets** of the super-key [AD] are [A] and [D], and none of whose **closures** determine all the attributes {ABCD} of R. **So, super-key [AD] is a candidate-key.**

The prime attributes are {A,D}

Step -2: Check if prime attributes are on the RHS of any FDs

- **YES.** The prime attribute A is on the RHS of C → A.
- Replace A with C in the existing candidate-key [AD] as [CD]. **It is a super-key now.**
- **Since none of the closures of the proper subsets of [CD] determine all the attributes {ABCD}, it is the new candidate-key.**

The prime attributes are now {A,D,C}

Identifying all the candidate Keys (Trick-2)

Consider a table R(A,B,C,D) and F = {A → B, B → C, C → A}.

- The candidate-keys so far are **[AD]** and **[CD]**
- The prime attributes so far are **{A,D,C}**

Step -3: Check if any unaccounted prime attributes are on the RHS of any FDs

- **YES.** The prime attribute C is on the RHS of B → C.
- Replace C with B in the existing candidate-key [CD] as [BD]. **It is a super-key now.**
- **Since none of the closures of the proper subsets of [BD] determine all the attributes {ABCD}, it is the new candidate-key.**

- The candidate-keys so far are **[AD], [CD] and [BD]**
- The prime attributes so far are **{A,B,C,D}**

Step -4: Check if any unaccounted prime attributes are on the RHS of any FDs

- **NO.** All the prime attributes are accounted for. So, we have found all the candidate-keys in R.

Quiz

Consider relation R(A,B,C,D,E,F) with functional dependencies:

$CDE \rightarrow B$, $ACD \rightarrow F$, $BEF \rightarrow C$, $B \rightarrow D$

Which of the following is a candidate key?

- BDF
- ABDF
- ACDE
- ADEF

Quiz

Consider relation R(A,B,C,D,E,F) with functional dependencies:

$CDE \rightarrow B$, $ACD \rightarrow F$, $BEF \rightarrow C$, $B \rightarrow D$

Which of the following is a candidate key?

- BDF
- ABDF
- **ACDE**
- ADEF

Types of Functional dependencies:

1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Transitive functional dependency
4. Multivalued functional dependency

1. Trivial functional dependency

In **Trivial FD**, a dependent is always a subset of the determinant.

i.e. If **X → Y** and **Y ⊆ X**.

roll_no	name	age
42	Alex	17
43	Eva	18
44	John	18

roll_no → roll_no

{roll_no, name} → name

2. Non-Trivial functional dependency

In **Non-trivial FD**, the dependent is **strictly not a subset** of the determinant.

i.e. If **X → Y** and **Y** $\nsubseteq$ **X**.

roll_no	name	age
42	Alex	17
43	Eva	18
44	John	18

roll_no → name

{roll_no, name} → age

3. Transitive functional dependency

In **transitive FD**, dependent is indirectly dependent on determinant.

i.e. If **a → b** & **b → c**, then according to axiom of transitivity, **a → c**.

roll_no	name	dept	building_no
42	Alex	CO	4
43	Eva	EC	2
44	John	IT	1
45	Jacob	EC	2

rolll_no → dept

dept → building_no

roll_no → building_no

4. Multivalued functional dependency

In **Multivalued FD**, entities of the dependent set are **not dependent on each other.**

i.e. If **a → {b, c}** and there exists **no functional dependency** between **b and c**.

roll_no	name	age
42	Alex	17
43	Eva	18
44	John	18

roll_no → {name, age}