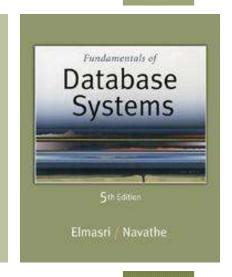


5th Edition

Elmasri / Navathe

Chapter 10

Functional Dependencies and Normalization for Relational Databases





Chapter Outline

- 1 Informal Design Guidelines for Relational Databases
 - 1.1Semantics of the Relation Attributes
 - 1.2 Redundant Information in Tuples and Update Anomalies
 - 1.3 Null Values in Tuples
 - 1.4 Spurious Tuples
- 2 Functional Dependencies (FDs)
 - 2.1 Definition of FD
 - 2.2 Inference Rules for FDs
 - 2.3 Equivalence of Sets of FDs
 - 2.4 Minimal Sets of FDs

Chapter Outline

- 3 Normal Forms Based on Primary Keys
 - 3.1 Normalization of Relations
 - 3.2 Practical Use of Normal Forms
 - 3.3 Definitions of Keys and Attributes Participating in Keys
 - 3.4 First Normal Form
 - 3.5 Second Normal Form
 - 3.6 Third Normal Form
- 4 General Normal Form Definitions (4NF For Multiple Keys)
- 5 BCNF (Boyce-Codd Normal Form)

1 Informal Design Guidelines for Relational Databases (1)

- What is relational database design?
 - The grouping of attributes to form "good" relation schemas
- Two levels of relation schemas
 - The logical "user view" level
 - The storage "base relation" level
- Design is concerned mainly with base relations
- What are the criteria for "good" base relations?

Informal Design Guidelines for Relational Databases (2)

- We first discuss informal guidelines for good relational design
- Then we discuss formal concepts of functional dependencies and normal forms
 - 1NF (First Normal Form)
 - 2NF (Second Normal Form)
 - 3NF (Third Normal Form)
 - BCNF (Boyce-Codd Normal Form)
- Additional types of dependencies, further normal forms, relational design algorithms by synthesis are discussed in Chapter 11

1.1 Semantics of the Relation Attributes

- GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).
 - Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) 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.
- Bottom Line: Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.

Figure 10.1 A simplified COMPANY relational database schema

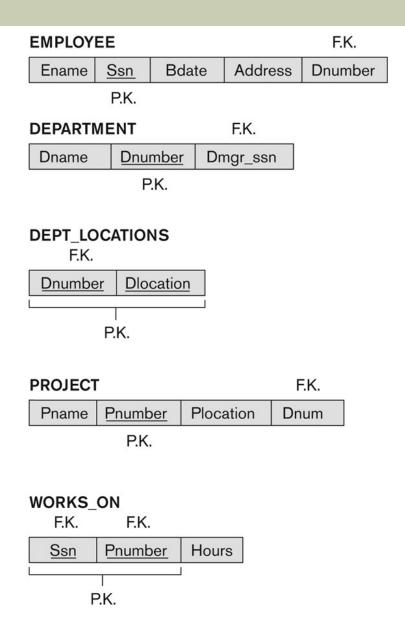


Figure 10.1
A simplified COMPANY relational database schema.

1.2 Redundant Information in Tuples and Update Anomalies

- Information is stored redundantly
 - Wastes storage
 - Causes problems with update anomalies
 - Insertion anomalies
 - Deletion anomalies
 - Modification anomalies

EXAMPLE OF AN UPDATE ANOMALY

- Consider the relation:
 - EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)
- Update Anomaly:
 - Changing the name of project number P1 from "Billing" to "Customer-Accounting" may cause this update to be made for all 100 employees working on project P1.

EXAMPLE OF AN INSERT ANOMALY

- Consider the relation:
 - EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)
- Insert Anomaly:
 - Cannot insert a project unless an employee is assigned to it.
- Conversely
 - Cannot insert an employee unless an he/she is assigned to a project.

EXAMPLE OF AN DELETE ANOMALY

- Consider the relation:
 - EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)
- Delete Anomaly:
 - When a project is deleted, it will result in deleting all the employees who work on that project.
 - Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

Example of Anomalies

Insertion anomaly:

- If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not allow inserting in referencing relation.
- For Example, If we try to insert a record in STUDENT_COURSE with STUD_NO =7, it will not allow.

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNT RY	STUD_AG E
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Course

	0
Course_Id	C_name
C1	DBMS
C2	CN
C3	Java
C4	Python

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME		
1	C1	DBMS		
2	C2	Computer Networks		
1	C2	Computer Networks		

Deletion and Update anomaly:

- If a tuple is deleted or updated from referenced relation and referenced attribute value is used by referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation.
- For Example, If we try to delete a record from STUDENT with STUD_NO =1, it will not allow.

Figure 10.3 Two relation schemas suffering from update anomalies

Figure 10.3

Two relation schemas suffering from update anomalies.

- (a) EMP_DEPT and
- (b) EMP_PROJ.

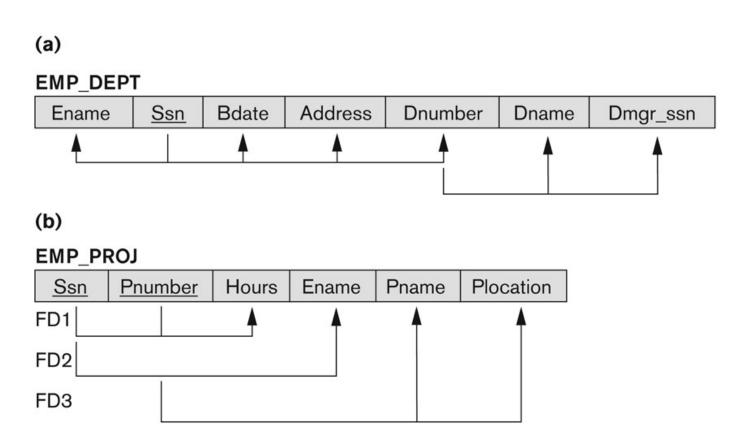


Figure 10.4 Example States for EMP_DEPT and EMP_PROJ

Redundancy

EMP_DEPT

Ename	San	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Guideline to Redundant Information in Tuples and Update Anomalies

GUIDELINE 2:

- Design a schema that does not suffer from the insertion, deletion and update anomalies.
- If there are any anomalies present, then note them so that applications can be made to take them into account.

1.3 Null Values in Tuples

■ GUIDELINE 3:

- 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)
- It causes confusion in COUNT and SUM operation.
- If NULL value comes in comparision with SELECT or JOIN operation, the result will unpredictable.

Slide 10-18

Reasons for nulls:

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

1.4 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(LOS

GUIDELINE 4:

The relations should be designed to satisfy the lossless join condition i.means No spurious tuples should be generated by doing a natural-join of any relations.

Spurious Tuples (2)

- There are two important properties of decompositions:
 - a) Non-additive or losslessness of the corresponding join
 - b) Preservation of the functional dependencies.

Note that:

- Property (a) is extremely important and cannot be sacrificed.
- Property (b) is less stringent and may be sacrificed. (See Chapter 11).

Example:



- The relation EMP_PROJ is decomposed into two relations: EMP_LOCS and EMP_PROJ1.
- by re-joining of these two relations will generate some spurious tuples.
- So the join operation will not get original EMP_Proj relation.

Example

- Spurious Tuples Example
- Consider the poorly designed relations EMP PROJ1

EMPNO	PNO	ENAME	PNAME	PLOC
1	P1	A	UNIX	BLORE
1	P2	A	C++	CHENNAI
2	Pl	В	UNIX	BLORE
2	P2	В	C++	CHENNAI
3	P1	C	UNIX	BLORE

 This poorly designed table is split on the basis of ploc(non PK)We get EMP-LOC AND EMP-PROJ

EMP-LOC		EMP-PR	OJ		
ENAME	PLOC	EMPNO	PNO	PNAME	PLOC
A	BLORE	1	P1	UNIX	BLORE
A	CHENNAI	1	P2	C++	CHENNAI
В	BLORE	2	P1	UNIX	BLORE
В	CHENNAI	2	P2	C++	CHENNAI
C	BLORE	3	P1	UNIX	BLORE

- NOW IF EMP-LOC AND EMP-PROJ JOINED
 ON THE BASIS OF PLOC, will not get proper output
- We will not get original table.

eno	pno	ename	pname	ploc	eno	pno	ploc
3	p1	C	Unix	Blore	1	p1	Blore
2	p1	В	Unix	Blore	1 1	p1	Blore
2	p2	В	C++	Chennai	1	p2	Chennai
1	p2	A	C++	Chennai	1 1	p2	Chennai
3	p1	C	Unix	Blore	2	p1	Blore
2	p1	В	Unix	Blore	2	p1	Blore
2	p2	В	C++	Chennai	2	p2	Chennai
1	p2	A	C++	Chennai	2	p2	Chennai
3	p1	C	Unix	Blore	3	p1	Blore
2	p1	В	Unix	Blore	3	p1	Blore

2.1 Functional Dependencies (1)

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

Keys:

- 1. Superkey:
- All possible combination of keys are called superkeys.
- Uniquely identify record.
- Superkey is a superset, we can derive other keys from superkey.

- {ID},{SSN},
- {ID,Name},{Name,SSN}
- {ID,SSN},{ID,phone},
- ID,email
- Name,email
- Name,ssn,email
- Id,SSN,Phone
- {Name,Salary}
- Here, {Name,Salary} can't become a superkey.



Candidate key

- Minimal set of superkey.
- From the following set of superkeys:

```
{ID},{SSN},{ID,Name},{Name,SSN},{ID,SSN},{ID,phone},
{ID,email},{Name,email},{Name,ssn,email},{Id,SSN,Phone},
{Name,Salary}
```

- We can select candidate key (minimal set)
- Such as,
- {ID} and {SSN} can be a separate candidate keys.
- {email}, {Name,Phone}
- {ID,Name} cant be candidate key, bcz ID is a part of another candidate key.
 Slide 10- 28

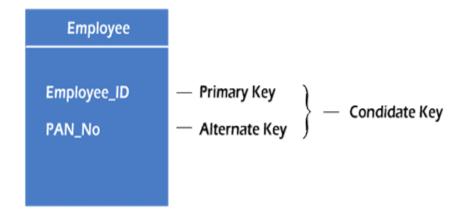
Primary key

- Primary key should be unique and not null
- Example:

```
Candidate keys in above examples are: {ID},{SSN},{Email},{Name,phone} Primary key is :{ID} or {SSN} Not both. {ID} is preferred.
```

Alternate Keys

The candidate key other than primary key are called alternate keys.



Unique key & Composite key

- It should contain unique values, but accept NULL values.
- Example:
- {Name,Phone} and {email} can be a unique keys.

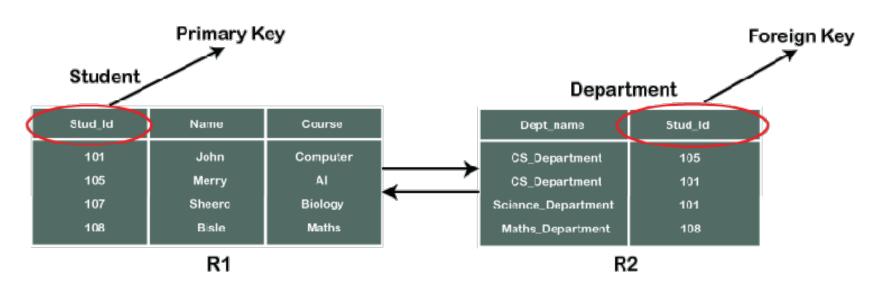
- Composite key:
- Combination of Two or More attribute
- Example:
- {Name,Phone}
- {ID,Phone}
- {ID,Name}.. etc.

Foreign key

- A foreign key in DBMS is a field that establishes and maintains a link between two separate tables.
- It acts as a constraint ensuring data and referential integrity within relational databases.
- Using foreign keys ensures that the data stored in different tables are consistent, helping maintain the relationship between the two tables while preventing any unauthorized modifications.
- In addition, they can also be used to create connections among multiple entities to organize information for better retrieval from the database.

Example

- This foreign key relationship ensures referential integrity between the two tables.
- It means that every value in the Stud_id column of the Department table must correspond to a valid Stud_id in the Student table.
- For example, you can't insert a record into the Department table with a Stud_id that doesn't exist in the Courses table. This helps to maintain consistency



Proper subset:

- A proper subset of a set A is a subset of A that is not equal to A.
- In other words, if B is a proper subset of A, then all elements of B are in A but A contains at least one element that is not in B.
- if $A=\{1,3,5\}$ then $B=\{1,5\}$ is a proper subset of A.
- The set C={1,3,5} is a subset of A, but it is not a proper subset of A since C=A.
- Example:
- Let us consider in relation R(A,B,C,D),
- If {A,B,C} is a super key, then proper subset of this super key are:
- · {A},
- · {B}
- · {C}
- · {A,B}
- {A,C}
- {B,C}

Functional Dependencies (2)

- X -> Y holds if whenever two tuples have the same value for X, they must have the same value for Y
 - For any two tuples t1 and t2 in any relation instance r(R): If t1[X]=t2[X], then t1[Y]=t2[Y]
- X -> Y in R specifies a constraint on all relation instances
 r(R)
- Written as X -> Y; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow:).
- FDs are derived from the real-world constraints on the attributes

Examples of FD constraints (1)

- Social security number determines employee name
 - SSN -> ENAME
- Project number determines project name and location
 - PNUMBER -> {PNAME, PLOCATION}
- Employee ssn and project number determines the hours per week that the employee works on the project
 - {SSN, PNUMBER} -> HOURS

Examples of FD constraints (2)

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

functional dependency

- The functional dependency is a relationship that exists between two attributes, where one set can accurately determine the value of other sets.
- It is denoted as X → Y, where X is a set of attributes that is capable of determining the value of Y.
- X is called **Determinant**, while on the right side, Y is called the **Dependent**.

Functional Dependency
A -> B

- B functionally dependent on A
- A determinant set
- B dependent attribute

Example

- Assume we have an employee table with attributes:
 - Emp_Id, Emp_Name, Emp_Address.
- Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.
- Functional dependency can be written as:
 - Emp_Id → Emp_Name
- We can say that Emp_Name is functionally dependent on Emp_Id.

Example

FD: X -> Y

X	Y
1	1
2	1
3	2
4	3

In this example,

Y	X	Γ
1	1	Ī
1	2	t1
2	3	ı
3	4	ı
5	2	t2
	2	t2

- FD: if t1.X = t2.X
- then t1.Y = t2.Y
- Here,
- + if t1.x = 1 t2.x = 2 false

Consider the following table:

- · RNO -> Name
- · FD
- Name -> RNO
- FD doesn't exist
- RNO -> Marks
- FD exist
- Name –Marks
- FD exist
- Dept -> Course
- FD doesn't exist
- Marks -> Dept
- FD doesn't exist
- {RNO, Name} -> Marks
- FD exist
- · {Name, Marks} -> Dept
- FD doesn't exist

RNO	Name	Marks	Dept	Course
1	Α	78	CS	C1
2	В	60	EX	C1
3	Α	78	CS	C2
4	В	60	ES	C3
5	С	80	IT	C3
6	D	80	EC	C2

2.3 Equivalence of Sets of FDs

- Two sets of FDs F and G are equivalent if:
 - Every FD in F can be inferred from G, and
 - Every FD in G can be inferred from F
 - Hence, F and G are equivalent if F⁺ =G⁺
- Definition (Covers):
 - F covers G if every FD in G can be inferred from F
 - (i.e., if G⁺ subset-of F⁺)
- F and G are equivalent if F covers G and G covers F
- There is an algorithm for checking equivalence of sets of FDs

2.4 Minimal Sets of FDs (1)

- A set of FDs is minimal if it satisfies the following conditions:
 - Every dependency in F has a single attribute for its RHS.
 - We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.
 - We cannot replace any dependency X -> A in F with a dependency Y -> A, where Y propersubset-of X (Y subset-of X) and still have a set of dependencies that is equivalent to F.

Minimal Sets of FDs (2)

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs
- To synthesize a set of relations, we assume that we start with a set of dependencies that is a minimal set
 - E.g., see algorithms 11.2 and 11.4

2.2 Inference Rules for FDs (1)

- Given a set of FDs F, we can infer additional FDs that hold whenever the FDs in F hold
- Armstrong's inference rules:
 - IR1. (**Reflexive**) If Y *subset-of* X, then X -> Y

```
If X \subseteq Y then X \rightarrow Y
```

Example:

```
{Rno,Name} -> Name
Here , name is a subset of {rno, name}
```

- IR2. (Augmentation) If X -> Y, then XZ -> YZ
 - (Notation: XZ stands for X U Z)

- IR3. (**Transitive**) If X -> Y and Y -> Z, then X -> Z
- IR1, IR2, IR3 form a sound and complete set of inference rules
 - These are rules hold and all other rules that hold can be deduced from these

Inference Rules for FDs (2)

- Some additional inference rules that are useful:
 - Decomposition: If X -> YZ, then X -> Y and X -> Z
 - Union: If X -> Y and X -> Z, then X -> YZ
 - Psuedotransitivity: If X -> Y and WY -> Z, then WX -> Z
- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

Inference Rules for FDs (3)

Closure of a set F of FDs is the set F+ of all FDs that can be inferred from F

Closure of a set of attributes X with respect to F is the set X+ of all attributes that are functionally determined by X

X⁺ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

Types of Functional dependencies

- 1.Full dependency
- 2. Partial dependancy
- 3. Trivial functional dependency
- 4. Non-Trivial functional dependency
- 5. Multivalued functional dependency
- 6. Transitive functional dependency

Fully functional dependency

An attribute is fully functional dependent on X and if Y is FD on X but not FD on any proper subset of X.

E_id	E_name	Department
101	A	Marketing
102	В	Production
103	С	Quality
101	В	Quality



Here, Eid -> Dept and E_name -> Dept, these 2 functional dependencies doesn't exist.

- Hence,
- Dept is completely depend on {Eid,Ename}.

Partial functional dependency

An attribute is fully functional dependent on X and if Y is FD on X but not FD on any

pro	E_id		E_name	Department	
•	101		A	Marketing	
	102		В	Production	
	103		С	Quality	



- Here, Eid -> Dept and E_name -> Dept, these 2 functional dependencies exist (individually).
- That means either Eid or Ename can determine Dept.
- Hence,
- Dept is partially depend on {Fid.Fname}.

Trivial Functional Dependency:

- In , a dependent is always a subset of the determinant.
 i.e. If X → Y and Y is the subset of X, then it is called trivial functional dependency
- For example,
- Here, {roll_no, name} → name is a trivial functional dependency, since the dependent name is a subset of determinant set {roll_no, name}
 Similarly, roll_no → roll_no is also an example of trivial functional dependency. 42 abc 17

Non-trivial_Functional_Dependency

- In Non-trivial functional dependency, the dependent is strictly not a subset of the determinant.
- i.e. If X → Y and Y is not a subset of X, then it is called Non-trivial functional dependency.
- For example,
- Here, roll_no → name is a non-trivial functional dependency, since the dependent name is not a subset of determinant roll no
- Similarly, {roll_no, name} → age is also a n trivial functional dependency, since age is no⁴³ pqr 18 subset of {roll_no, name}

abc

Multivalued Functional Dependency

- In Multivalued functional dependency, attributes 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, then it is called a multivalued functional dependency.
- Here, roll_no → {name, age} is a multivalued functional dependency, since the dependency are not dependence
- each other(i.e. name → age or age → name doesexist!)

Transitive Functional Dependency

For example,

In transitive functional dependency, dependent is indirectly dependent on determinant.
 i.e. If a → b & b → c, then according to axiom of transitivity, a → c. This is a transitive functional dependency

enrol_no	name	dept	building_no
42	abc	CO	4
43	pqr	EC	2
44	хух	IT	1
45	abc	EC	2

- Example:
- Here, enrol_no → dept and dept → building_no,
- Hence, according to the axiom of transitivity, enrol_no
 → building_no is a valid functional dependency.
- This is an indirect functional dependency, hence called Transitive functional dependency.

Find superkeys

- If we have 'N' attributes with one candidate key then the number of possible superkeys is 2^(N-1).
- Example:
- Let's say we have a relation R with attributes {a1, a2, a3} and a1 is the candidate key.
- Then, any superset of a1 is the super key.
- Here n=3, thus, we can have $2^{(3-1)}=2^{2-4}$,
- four possible super keys: {a1, a1 a2, a1 a3, a1 a2 a3}.

Examples

- Steps to Find the Attribute Closure:
- Given the FD set of a Relation R, The attribute closure set S is the set of Attribute Closure A.
 - Add A to S.
 - Recursively add attributes that can be functionally determined from attributes of the set S until done.
- From Table 1, FDs are
- Given R (E-ID, E-NAME, E-CITY, E-STATE)

 FDs = { E-ID->E-NAME, E-ID->E-CITY, E-ID->L-STATE }

 >E-STATE }

E-ID	E-NAME	E-CITY	E-STATE
E001	John	Delhi	Delhi
E002	Mary	Delhi	Delhi
E003	John	Noida	U.P.

- The attribute closure of E-ID can be calculated as:
- Add E-ID to the set {E-ID}
- Add Attributes that can be derived from any attribute of the set.
- As there is one other attribute remaining in relation to be derived from E-ID.
- So the result is:
- (E-ID)+ = {E-ID, E-NAME, E-CITY, E-STATE }

- Similarly,
- **(E-NAME)**+ = {E-NAME}
- **(E-CITY)**+ = {E-CITY, E_STATE}
- (E-STATE)+ = { E_STATE}

Example

Find the attribute closures of (B)+,given
FDs R(ABCDE) = {AB->C, B->D, C->E, D->A}.

Attributes Added in Closure	FD used
{B}	Triviality
{B, D}	B->D
{B, D, A}	D->A
{B, D, A, C}	AB->C
{B, D, A, C, E}	C->E

Closure of FD EXAMPLE'S

Ex1. To find Candidate keys

- R(ABCD)
- FD: $\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$
- Ans:
- Closure of A means what A determines.
- Find closure of all attributes:
- A+= {A,B,C,D} S.K
- $B^{+}=\{B,C,D\}$
- $C^{+}=\{C,D\}$
- $D^{+}=\{D\}$
- AB+={ABCD} S.K
- AC+ = {ABCD} S.K
- AD+ = ABC+ = ABD+ = ACD+ ... S.K
- CK={A}

PRIME ATTRIBUTE:

The set of attributes makes candidate key are called prime attributes.

Here,

Prime attributes are : {A}

Non –prime: {B,C,D}

Hence, candidate key is A.

Example

- Given Relation R(A,B,C,D,E) & FD: {A→B, D→E}
- $ABCDE^+ = \{A,B,C,D,E\} S.K$
- $ACDE^+ = \{A,B,C,D,E\} S.K$
- $ACD^+ = \{A,B,C,D,E\} S.K$
- How to check ACD is only the candidate key.
- Prime attributes: {A,C,D}
- Check any of the prime attribute is available in R.H.S of Functional Dependency i.e A,C & D
- No any prime attribute is present in RHS. So we can say that is the only candidate key {ACD}.