Example for Super Key and candidate key

Emp\_SSN Emp\_Number Emp\_Name

--------- ---------- --------

123456789 226 Steve

999999321 227 Ajeet

888997212 228 Chaitanya

777778888 229 Robert

**Super keys**: The above table has following super keys. All of the following sets of super key are able to uniquely identify a row of the employee table.

* {Emp\_SSN}
* {Emp\_Number}
* {Emp\_SSN, Emp\_Number}
* {Emp\_SSN, Emp\_Name}
* {Emp\_SSN, Emp\_Number, Emp\_Name}
* {Emp\_Number, Emp\_Name}

**Candidate Keys** a candidate key is a minimal super key with no redundant attributes. The following two set of super keys are chosen from the above sets as there are no redundant attributes in these sets.

* {Emp\_SSN}
* {Emp\_Number}

[**Primary key**](https://beginnersbook.com/2015/04/primary-key-in-dbms/):  
A Primary key is selected from a set of candidate keys. This is done by database admin or database designer. We can say that either {Emp\_SSN} or {Emp\_Number} can be chosen as a primary key for the table Employee.

Functional Dependency and Attribute Closure

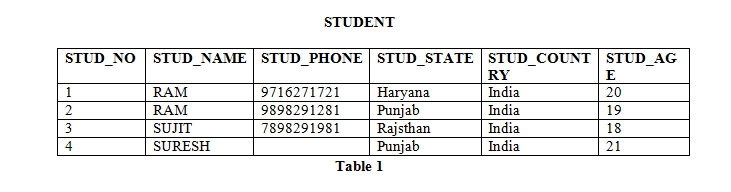
**Functional Dependency**

A functional dependency A->B in a relation holds if two tuples having same value of attribute A also have same value for attribute B. For Example, in relation STUDENT shown in table 1, Functional Dependencies

STUD\_NO->STUD\_NAME, STUD\_NO->STUD\_PHONE **hold**

but

STUD\_NAME->STUD\_ADDR **do not hold**



Functional Dependencies in a relation are dependent on the domain of the relation. Consider the STUDENT relation given in Table 1.

* We know that STUD\_NO is unique for each student. So STUD\_NO->STUD\_NAME, STUD\_NO->STUD\_PHONE, STUD\_NO->STUD\_STATE, STUD\_NO->STUD\_COUNTRY and STUD\_NO -> STUD\_AGE all will be true.
* Similarly, STUD\_STATE->STUD\_COUNTRY will be true as if two records have same STUD\_STATE, they will have same STUD\_COUNTRY as well.
* For relation STUDENT\_COURSE, COURSE\_NO->COURSE\_NAME will be true as two records with same COURSE\_NO will have same COURSE\_NAME.

Functional Dependency Set: Functional Dependency set or FD set of a relation is the set of all FDs present in the relation. For Example, FD set for relation STUDENT shown in table 1 is:

{ STUD\_NO->STUD\_NAME, STUD\_NO->STUD\_PHONE, STUD\_NO->STUD\_STATE, STUD\_NO->STUD\_COUNTRY, STUD\_NO -> STUD\_AGE, STUD\_STATE->STUD\_COUNTRY }

Attribute Closure: Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.

To find attribute closure of an attribute set:

Add elements of attribute set to the result set.

Recursively add elements to the result set which can be functionally determined from the elements of the result set.

**How to find Candidate Keys and Super Keys using Attribute Closure?**

* If attribute closure of an attribute set contains all attributes of relation, the attribute set will be super key of the relation.
* If no subset of this attribute set can functionally determine all attributes of the relation, the set will be candidate key as well. For Example, using FD set of table 1,

(STUD\_NO, STUD\_NAME)+ = {STUD\_NO, STUD\_NAME, STUD\_PHONE, STUD\_STATE, STUD\_COUNTRY, STUD\_AGE}

{STUD\_NO}-> {STUD-STATE,STUD-PIN}

{STUD-STATE , STUD-PIN}->{STUD\_COUNTRY}

{STUD-NO, STUD-PIN}->{STUD-COUNTRY}

(STUD\_NO)+ = {STUD\_NO, STUD\_NAME, STUD\_PHONE, STUD\_STATE, STUD\_COUNTRY, STUD\_AGE}

(STUD\_NO, STUD\_NAME) will be super key but not candidate key because its subset (STUD\_NO)+ is equal to all attributes of the relation. So, STUD\_NO will be a candidate key.

# Inference Rule (IR):

* The Armstrong's axioms are the basic inference rule.
* Armstrong's axioms are used to conclude functional dependencies on a relational database.
* The inference rule is a type of assertion. It can apply to a set of FD (functional dependency) to derive other FD.
* Using the inference rule, we can derive additional functional dependency from the initial set.

The Functional dependency has 6 types of inference rule:

**Reflexive Rule (IR1)**

In the reflexive rule, if Y is a subset of X, then X determines Y.

If X ⊇ Y then X → Y

Example:

X = {a, b, c, d, e}

Y = {a, b, c}

**Augmentation Rule (IR2)**

The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

If X    → Y then XZ   →   YZ

**Example:**

For R(ABCD),  **if** A   →   B then AC  →   BC

**Transitive Rule (IR3)**

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

If X → Y and Y → Z then X → Z

**Union Rule (IR4)**

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

If X    →  Y and X   →  Z then X  →    YZ

**Proof:**

1. X → Y (given)  
2. X → Z (given)  
3. X → XY (using IR2 on 1 by augmentation with X. Where XX = X)  
4. XY → YZ (using IR2 on 2 by augmentation with Y)  
5. X → YZ (using IR3 on 3 and 4)

**Decomposition Rule (IR5)**

Decomposition rule is also known as project rule. It is the reverse of union rule.

This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

If X   →   YZ then X   →   Y and X →    Z

**Proof:**

1. X → YZ (given)  
2. YZ → Y (using IR1 Rule)  
3. X → Y (using IR3 on 1 and 2)

**Pseudo transitive Rule (IR6)**

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

If X   →   Y and ZY   →   W then XZ   →   W

**Proof:**

1. X → Y (given)  
2. YZ->W (GIVEN)  
3. WX → WY (using IR2 on 1 by augmenting with W)  
4. WX → Z (using IR3 on 3 and 2)

**Properties of Functional Dependencies**

Let X, Y, and Z are sets of attributes in a relation R. There are several properties of functional dependencies which always hold in R also known as Armstrong Axioms.

1. **Reflexivity**: If *Y* is a subset of *X*, then *X* → *Y.*e.g.; Let X represents {E-ID, E-NAME} and Y represents {E-ID}.  {E-ID, E-NAME}->E-ID is true for the relation.
2. **Augmentation**: If *X* → *Y*, then *XZ* → *YZ.* e.g.; Let X represents {E-ID}, Y represents {E-NAME} and Z represents {E-CITY}. As {E-ID}->E-NAME is true for the relation, so { E-ID,E-CITY}->{E-NAME,E-CITY} will also be true.
3. **Transitivity**: If *X* → *Y* and *Y* → *Z*, then *X* → *Z.*e.g.; Let X represents {E-ID}, Y represents {E-CITY} and Z represents {E-STATE}. As {E-ID} ->{E-CITY} and {E-CITY}->{E-STATE}  is true for the relation, so { E-ID }->{E-STATE} will also be true.
4. **Attribute Closure:** The set of attributes that are functionally dependent on the attribute A is called Attribute Closure of A and it can be represented as A+.

**Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, N} and the set of functional dependencies {{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L}, K -> {M}, L -> {N} on R. What is the key for R?**

A. {E, F}  
B. {E, F, H}  
C. {E, F, H, K, L}  
D. {E}

**Answer:** Finding attribute closure of all given options, we get:  
{E,F}+ = {EFGIJ}  
{E,F,H}+ = {EFHGIJKLMN}  
{E,F,H,K,L}+ = {{EFHGIJKLMN}  
{E}+ = {E}  
{EFH}+ and {EFHKL}+ results in set of all attributes, but EFH is minimal. So it will be candidate key.

**Prime and non-prime attributes**

Attributes which are parts of any candidate key of relation are called as prime attribute, others are non-prime attributes. For Example, STUD\_NO in STUDENT relation is prime attribute, others are non-prime attribute.

**Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A–>B, BC–> D, E–>C, D–>A}. What are the candidate keys of R?**

(a) AE, BE  
(b) AE, BE, DE  
(c) AEH, BEH, BCH  
(d) AEH, BEH, DEH

**Answer:** (AE)+ = {ABECD} which is not set of all attributes. So AE is not a candidate key. Hence option A and B are wrong.  
(AEH)+ = {ABCDEH}  
(BEH)+ = {BEHCDA}  
(BCH)+ = {BCHDA} which is not set of all attributes. So BCH is not a candidate key. Hence option C is wrong.  
So correct answer is D.

Given a relation R= {A, B, C, D, E, H} AND HAVING THE FOLLOWING FD’S

A->BC

CD->E

E->C

D->AEH

ABH->BD

DH->BC

Find the key for relation R with FD

A-> {ABC}

CD-> {CDEAHB} {CD}-CANDIDATE KEY

{ABCDEH}

D-> {DAEHBC} = {ABCDEH} {D}-CANDIDATE KEY {D}

PRIME ATTRIBUTE {CD}

NON PRIME ATTRIBUTS {ABEH}