Functional Dependency

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

X   →   Y

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

**For 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

## Types of Functional dependency

### 1. Trivial functional dependency

* A → B has trivial functional dependency if B is a subset of A.
* The following dependencies are also trivial like: A → A, B → B

**Example:**

1. Consider a table with two columns Employee\_Id and Employee\_Name.
2. {Employee\_id, Employee\_Name}   →    Employee\_Id is a trivial functional dependency as
3. Employee\_Id is a subset of {Employee\_Id, Employee\_Name}.
4. Also, Employee\_Id → Employee\_Id and Employee\_Name   →    Employee\_Name are trivial dependencies too.

### 2. Non-trivial functional dependency

* A → B has a non-trivial functional dependency if B is not a subset of A.
* When A intersection B is NULL, then A → B is called as complete non-trivial.

**Example:**

1. ID   →    Name,
2. Name   →    DOB

# 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:

## 1. Reflexive Rule (IR1)

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

1. If X ⊇ Y then X  →    Y

**Example:**

1. X = {a, b, c, d, e}
2. Y = {a, b, c}

## 2. 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.

1. If X    →  Y then XZ   →   YZ

**Example:**

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

## 3. Transitive Rule (IR3)

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

1. If X   →   Y and Y  →  Z then X  →   Z

## 4. Union Rule (IR4)

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

1. 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)

## 5. 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.

1. 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)

## 6. Pseudo transitive Rule (IR6)

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

1. If X   →   Y and YZ   →   W then XZ   →   W

**Proof:**

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

# Closure of an Attribute

**Closure of an Attribute:** Closure of an Attribute can be defined as a set of attributes that can be functionally determined from it.

OR

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 concerning F is the set X+ of all attributes that are functionally determined by X

### Pseudocode to find Closure of an Attribute?

Determine X+, the closure of X under functional dependency set F

X Closure : = will contain X itself;

Repeat the process as:

old X Closure     : = X Closure;

for each functional dependency P → Q in FD set do

if X Closure is subset of P then X Closure := X Closure U Q ;

Repeat until ( X Closure = old X Closure);

### Algorithm of Determining X+, the Closure of X under F

**Input:** A set F of FDs on a relation schema R, and a set of attributes X, which is a subset of R.

1. X+ := X;
2. repeat
3. oldX+ := X+ ;
4. **for** each functional dependency Y → Z in F do
5. if X+ ⊇ Y **then** X+ := X+ ∪ Z;
6. until (X+ = oldX+ );

### QUESTIONS ON CLOSURE SET OF ATTRIBUTE:

1) Given relational schema **R( P Q R S T U V)** having following attribute P Q R S T U and V, also there is a set of functional dependency denoted by **FD = { P->Q, QR->ST, PTV->V }.**

Determine Closure of **(QR)+ and (PR)+**

a) QR+ = QR (as the closure of an attribute or set of attributes contain same).

Now as per algorithm look into a set of FD that complete the left side of any FD contains either Q, R, or QR since in FD QR→ST has complete QR.

Hence QR+ = QRST

Again, trace the remaining two FD that any left part of FD contains any Q, R, S, T.

Since no complete left side of the remaining two FD{P->Q, PTV->V} contain Q, R, S, T.

**Therefore QR+ = QRST** (Answer)

# Closure of an Attribute

**Closure of an Attribute:** Closure of an Attribute can be defined as a set of attributes that can be functionally determined from it.

OR

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 concerning F is the set X+ of all attributes that are functionally determined by X

### Pseudocode to find Closure of an Attribute?

Determine X+, the closure of X under functional dependency set F

X Closure : = will contain X itself;

Repeat the process as:

old X Closure     : = X Closure;

for each functional dependency P → Q in FD set do

if X Closure is subset of P then X Closure := X Closure U Q ;

Repeat until ( X Closure = old X Closure);

### Algorithm of Determining X+, the Closure of X under F

**Input:** A set F of FDs on a relation schema R, and a set of attributes X, which is a subset of R.

1. X+ := X;
2. repeat
3. oldX+ := X+ ;
4. **for** each functional dependency Y → Z in F do
5. if X+ ⊇ Y **then** X+ := X+ ∪ Z;
6. until (X+ = oldX+ );

### QUESTIONS ON CLOSURE SET OF ATTRIBUTE:

1) Given relational schema **R( P Q R S T U V)** having following attribute P Q R S T U and V, also there is a set of functional dependency denoted by **FD = { P->Q, QR->ST, PTV->V }.**

Determine Closure of **(QR)+ and (PR)+**

a) QR+ = QR (as the closure of an attribute or set of attributes contain same).

Now as per algorithm look into a set of FD that complete the left side of any FD contains either Q, R, or QR since in FD QR→ST has complete QR.

Hence QR+ = QRST

Again, trace the remaining two FD that any left part of FD contains any Q, R, S, T.

Since no complete left side of the remaining two FD{P->Q, PTV->V} contain Q, R, S, T.

**Therefore QR+ = QRST** (Answer)

b) PR + = PR (as the closure of an attribute or set of attributes contain same)

Now as per algorithm look into a set of FD, and check that complete left side of any FD contains either P, R, or PR. Since in FD P→Q, P is a subset of PR, Hence PR+ = PRQ

Again, trace the remaining **two FD** that any left part of FD contains any P, R, Q, Since, in FD QR → ST has its complete left part QR in PQR

Hence PR+ = PRQST

Again trace the remaining one FD { PTV->V } that its complete left belongs to PRQST. Since complete PTV is not in PRQST, hence we ignore it.

**Therefore PR+ = PRQST ( Answer)**

2. Given relational schema R( P Q R S T) having following attributes P Q R S and T, also there is a set of functional dependency denoted by FD = { P->QR, RS->T, Q->S, T-> P }.

Determine Closure of ( T )+

T + = T (as the closure of an attribute or set of attributes contain same)

Now as per algorithm look into a set of FD that complete the left side of any FD contains T since, in FD T → P, T is in T, Hence T+ = TP

Again trace the remaining three FD that any left part of FD contain any TP, Since in FD P → QR has its complete left part P in TP, Hence T+ = TPQR

Again trace the remaining two FD { RS->T, Q->S } that any of its Complete left belongs to TPQR, Since in FD Q → S has its complete left part Q in TPQR, Hence T+ = TPQRS

Again trace the remaining one FD { RS->T } that its complete left belongs to TPQRS, Since in FD RS → T has its complete left part RS in TPQRS Hence T+ = TPQRS ( no changes, as T, is already in TPQRS)

**Therefore T+ = TPQRS ( Answer).**