Addition Notes:

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

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 $Y \rightarrow Z$ in F do
5. if $X^+ \supseteq Y$ **then** $X^+ := X+ \cup Z$;
6. until ($X^+ = oldX^+$);

Closure Of Functional Dependency : Introduction

- **The Closure Of Functional Dependency means the complete set of all possible attributes that can be functionally derived from given functional dependency using the inference rules known as Armstrong's Rules.**
- **If "F" is a functional dependency then closure of functional dependency can be denoted using "$\{F\}^+$".**
- **There are three steps to calculate closure of functional dependency. These are:**

Step-1 : Add the attributes which are present on Left Hand Side in the original functional dependency.

Step-2 : Now, add the attributes present on the Right Hand Side of the functional dependency.

Step-3 : With the help of attributes present on Right Hand Side, check the other attributes that can be derived from the other given functional dependencies. Repeat this process until all the possible attributes which can be derived are added in the closure.

Closure Of Functional Dependency : Examples

Example-1 : Consider the table student_details having (Roll_No, Name,Marks, Location) as the attributes and having two functional dependencies.

FD1 : Roll_No → Name, Marks

FD2 : Name →Marks, Location

Now, We will calculate the closure of all the attributes present in the relation using the three steps mentioned below.

Step-1 : Add attributes present on the LHS of the first functional dependency to the closure.

$\{Roll\_no\}^+ = \{Roll\_No\}$

Step-2 : Add attributes present on the RHS of the original functional dependency to the closure.

$\{Roll\_no\}^+ = \{Roll\_No, Marks\}$

Step-3 : Add the other possible attributes which can be derived using attributes present on the RHS of the closure. So Roll_No attribute cannot functionally determine any attribute but Name attribute can determine other attributes such as Marks and Location using 2nd Functional Dependency(Name [icon name="long-arrow-right" class="" unprefixed_class=""] Marks, Location).

Therefore, complete closure of Roll_No will be :

$\{Roll\_no\}^+ = \{Roll\_No, Marks, Name, Location\}$

Similarly, we can calculate closure for other attributes too i.e "Name".

Step-1 : Add attributes present on the LHS of the functional dependency to the closure.

$\{Name\}^+ = \{Name\}$

Step-2 : Add the attributes present on the RHS of the functional dependency to the closure.

$\{Name\}^+ = \{Name, Marks, Location\}$

Step-3 : Since, we don't have any functional dependency where "Marks or Location" attribute is functionally determining any other attribute , we cannot add more attributes to the closure. Hence complete closure of Name would be :

$\{Name\}^+ = \{Name, Marks, Location\}$

NOTE : We don't have any Functional dependency where marks and location can functionally determine any attribute. Hence, for those attributes we can only add the attributes themselves in their closures. Therefore,

$\{Marks\}^+ = \{Marks\}$

and

$\{Location\}^+ = \{ Location\}$

Example-2 : Consider a relation R(A,B,C,D,E) having below mentioned functional dependencies.

FD1 : A→ BC

FD2 : C→ B

FD3 : D → E

FD4 : E → D

Now, we need to calculate the closure of attributes of the relation R. The closures will be:

$\{A\}^+ = \{A, B, C\}$

$\{B\}^+ = \{B\}$

$\{C\}^+ = \{B, C\}$

$\{D\}^+ = \{D, E\}$

$\{E\}^+ = \{E\}$

Closure Of Functional Dependency : Calculating Candidate Key

- **"A Candidate Key of a relation is an attribute or set of attributes that can determine the whole relation or contains all the attributes in its closure."**
- **Let's try to understand how to calculate candidate keys.**

Example-1 : Consider the relation R(A,B,C) with given functional dependencies :

FD1 : A → B

FD2 : B→ C

Now, calculating the closure of the attributes as :

$\{A\}^+ = \{A, B, C\}$

$\{B\}^+ = \{B, C\}$

$\{C\}^+ = \{C\}$
Clearly, "A" is the candidate key as, its closure contains all the attributes present in the relation "R".

Example-2 : Consider another relation R(A, B, C, D, E) having the Functional dependencies :

FD1 : A → BC

FD2 : C→ B

FD3 : D → E

FD4 : E → D

Now, calculating the closure of the attributes as :

$\{A\}^+ = \{A, B, C\}$

$\{B\}^+ = \{B\}$

$\{C\}^+ = \{C, B\}$

$\{D\}^+ = \{E, D\}$

$\{E\}^+ = \{E, D\}$
In this case, a single attribute is unable to determine all the attribute on its own like in previous example. Here, we need to combine two or more attributes to determine the candidate keys.

$\{A, D\}^+ = \{A, B, C, D, E\}$

$\{A, E\}^+ = \{A, B, C, D, E\}$
Hence, "AD" and "AE" are the two possible keys of the given relation "R". Any other combination other than these two would have acted as extraneous attributes.

NOTE : Any relation "R" can have either single or multiple candidate keys.

Closure Of Functional Dependency : Key Definitions

1. **Prime Attributes : Attributes which are indispensable part of candidate keys. For example : "A, D, E" attributes are prime attributes in above example-2.**
2. **Non-Prime Attributes : Attributes other than prime attributes which does not take part in formation of candidate keys. For example.**
3. **Extraneous Attributes : Attributes which does not make any effect on removal from candidate key.**

For example : Consider the relation R(A, B, C, D) with functional dependencies :

FD1 : A → BC

FD2 : B → C

FD3 : D→ C

Here, Candidate key can be "AD" only. Hence,

Prime Attributes : A, D.

Non-Prime Attributes : B, C

Extraneous Attributes : B, C(As if we add any of the to the candidate key, it will remain unaffected). Those attributes, which if removed does not affect closure of that set.

https://minigranth.in/dbms-tutorial/canonical-cover-of-functional-dependency

Consider the following Supplier Database schema: Solve using SQL and relational algebra
SUPPLIERS (sid: integer, sname: string, address: string)
PARTS (pid: integer, pname: string, color: string)
CATALOG (sid: integer, pid: integer, cost: real)

1. Insert at least 5 values for each table.
2. Retrieve all supplier names (sname) along with their corresponding addresses.
3. Find the names and colors of all parts.
4. List the suppliers who supply a part with a specific pid (e.g., pid = 101)
5. Retrieve the names of all parts supplied by a specific supplier (e.g., sid = 1).
6. Display the catalog entries along with the corresponding supplier names and part names
7. Retrieve the part names (pname) and their corresponding colors for parts supplied by a specific supplier (e.g., sid = 2)

8. List the suppliers who supply parts in a specific color (e.g., color = 'Red').

```
INSERT INTO CATALOG (sid, pid, cost) VALUES (1, 101, 10.50), (1, 102, 15.75),(1, 103, 20.00),
(1, 104, 12.25), (1, 105, 18.50);

SELECT sname, address FROM SUPPLIERS;

SELECT pname, color FROM PARTS;

SELECT s.sname FROM SUPPLIERS s JOIN CATALOG c ON s.sid = c.sid WHERE c.pid = 101;

SELECT p.pname FROM PARTS p JOIN CATALOG c ON p.pid = c.pid WHERE c.sid = 1;

SELECT s.sname, p.pname, c.cost FROM CATALOG c JOIN SUPPLIERS s ON c.sid = s.sid
JOIN PARTS p ON c.pid = p.pid;

SELECT p.pname, p.color  FROM PARTS p JOIN CATALOG c ON p.pid = c.pid WHERE c.sid = 2;

SELECT DISTINCT s.sname FROM SUPPLIERS s  JOIN CATALOG c ON s.sid = c.sid
JOIN PARTS p ON c.pid = p.pid WHERE p.color = 'Red';
```

```
SUPPLIERS := { (1, 'Supplier A', 'Address A')};

π(sname, address)(SUPPLIERS)

π(pname, color)(PARTS)

π(sname)((σ(pid=101)(CATALOG)) ⋈ SUPPLIERS)

π(pname)((σ(sid=1)(CATALOG)) ⋈ PARTS)

(π(sid, sname, pid, pname, cost)(CATALOG ⋈ SUPPLIERS ⋈ PARTS))

π(pname, color)((σ(sid=2)(CATALOG)) ⋈ PARTS)

π(sname)((σ(color='Red')(CATALOG)) ⋈ SUPPLIERS)
```

Given the schema and write the SQL queries

EMP (Fname, Lname, SSN, Bdate, Address, Gender, Salary, Super SSN, Dno)

DEPT (Dname, Dnumber, Mgr_SSN, Mgrstartdate)

DEPT_LOC (DNumber, Dloc)

PROJECT (Pname, Pnumber, Ploc, Dnum)

WORKS_ON (WSSN, Pno, Hours)

DEPENDENT (ESSN, Dept_name, Gender, Bdate, relation).

1. Select all employees' first and last names:

SELECT Fname, Lname FROM EMP;

2. Select the names of employees who have a salary greater than $50,000:

SELECT Fname, Lname FROM EMP WHERE Salary > 50000;

3. Select the project name, number, and location for each project:

SELECT Pname, Pnumber, Ploc FROM PROJECT;

4. Select the first and last names of employees who work on project number '12345':

SELECT EMP.Fname, EMP.Lname FROM EMP INNER JOIN WORKS_ON ON EMP.SSN = WORKS_ON.WSSN WHERE WORKS_ON.Pno = '12345';

5. Insert a new department with name 'Research', number '5', manager SSN '123456789', and manager start date '2024-01-01':

INSERT INTO DEPT (Dname, Dnumber, Mgr_SSN, Mgrstartdate) VALUES ('Research', 5, '123456789', '2024-01-01');

6. Update the salary of employee with SSN '987654321' to $60,000:

UPDATE EMP SET Salary = 60000 WHERE SSN = '987654321';

7. Delete the project with number '56789':

DELETE FROM PROJECT WHERE Pnumber = '56789';

Relational Algebra:

Let's define the following symbols:

- π: Projection
- σ: Selection
- ρ: Renaming
- ⋈: Natural Join

1. Select all employees' first and last names:

π(Fname, Lname)(EMP)

2. Select the names of employees who have a salary greater than $50,000:

σ(Salary > 50000)(EMP)

3. Select the project name, number, and location for each project:

π(Pname, Pnumber, Ploc)(PROJECT)

4. Select the first and last names of employees who work on project number '12345':

π(EMP.Fname, EMP.Lname)(EMP ⋈ (σ(Pno = '12345')(WORKS_ON)))

5. Insert a new department with name 'Research', number '5', manager SSN '123456789', and manager start date '2024-01-01':

There is no direct relational algebra representation for insert operations. Relational algebra deals with querying existing data rather than modifying it.

6. Update the salary of employee with SSN '987654321' to $60,000:

There is no direct relational algebra representation for update operations.

7. Delete the project with number '56789':

There is no direct relational algebra representation for delete operations.

These SQL queries and their relational algebra representations demonstrate common operations that can be performed on the given schema.

SAILORS (sid, sname, rating, age)

BOATS (bid, bname, color)

RESERVES (sid, bid, day)

1) Find the names of sailors who have reserved the boat number '103'.

2)Find the names of sailors who have reserved a 'red' and a 'green' boat.

3)Find the names of sailors who have reserved atleast one boat.

4) Find the names of sailors with age over 20 years, who have not reserved a red boat.

5) Find the names of sailors who have reserved a red boat.

6) Retrieve all sailors' names who have not reserved any boat.

SQL Queries:

1. **Find the names of sailors who have reserved the boat number '103':**

SELECT DISTINCT S.sname FROM SAILORS S INNER JOIN RESERVES R ON S.sid = R.sid WHERE R.bid = '103';

2. **Find the names of sailors who have reserved a 'red' and a 'green' boat:**

SELECT DISTINCT S.sname FROM SAILORS S INNER JOIN RESERVES R ON S.sid = R.sid INNER JOIN BOATS B ON R.bid = B.bid WHERE B.color IN ('red', 'green') GROUP BY S.sid HAVING COUNT(DISTINCT B.color) = 2;

3. **Find the names of sailors who have reserved at least one boat:**

SELECT DISTINCT S.sname FROM SAILORS S INNER JOIN RESERVES R ON S.sid = R.sid;

4. **Find the names of sailors with age over 20 years, who have not reserved a red boat:**

SELECT DISTINCT S.sname FROM SAILORS S LEFT JOIN RESERVES R ON S.sid = R.sid LEFT JOIN BOATS B ON R.bid = B.bid AND B.color = 'red' WHERE S.age > 20 AND B.bid IS NULL;

5. **Find the names of sailors who have reserved a red boat:**

SELECT DISTINCT S.sname FROM SAILORS S INNER JOIN RESERVES R ON S.sid = R.sid INNER JOIN BOATS B ON R.bid = B.bid WHERE B.color = 'red';

6. **Retrieve all sailors' names who have not reserved any boat:**

SELECT DISTINCT S.sname FROM SAILORS S LEFT JOIN RESERVES R ON S.sid = R.sid WHERE R.sid IS NULL;

Relational Algebra:

Let's define the following symbols:

- π: Projection
- σ: Selection
- ⋈: Natural Join
- ρ: Renaming
- -: Set Difference

1. **Find the names of sailors who have reserved the boat number '103':**

π(sname)(σ(bid = '103')(RESERVES ⋈ SAILORS))

2. **Find the names of sailors who have reserved a 'red' and a 'green' boat:**

π(sname)(σ(color = 'red' ∨ color = 'green')(BOATS ⋈ RESERVES ⋈ SAILORS) - π(sname)(σ(color = 'red' ∧ color = 'green')(BOATS ⋈ RESERVES ⋈ SAILORS)))

3. **Find the names of sailors who have reserved at least one boat:**

π(sname)(SAILORS ⋈ RESERVES)

4. **Find the names of sailors with age over 20 years, who have not reserved a red boat:**

π(sname)((SAILORS - (π(sname)(σ(color = 'red')(BOATS ⋈ RESERVES ⋈ SAILORS)))) ⋈ σ(age > 20)(SAILORS))

5. **Find the names of sailors who have reserved a red boat:**

π(sname)(σ(color = 'red')(BOATS ⋈ RESERVES ⋈ SAILORS))

6. **Retrieve all sailors' names who have not reserved any boat:**

π(sname)(SAILORS - (RESERVES ⋈ SAILORS))

These SQL queries and relational algebra expressions provide solutions to the given scenarios based on the provided schema.