UNIVERSITY DATABASE MANAGEMENT SYSTEM

Group Members

(Section 2)

Archit Agrawal (202051213)

Kalash Singh Jadoun (202051097)

Sarang Nagar (202051168)

Sahil Kumar Rai (202051167)

Sushil Kumar Patel (202051188)

Reporting TA : Amit Dwivedi

INTRODUCTION

University Database Management System is a MySQL database designed to handle the university records. The University Database Management System creates, manages and performs all the activities related to the database of a given university.

The database consists of information about the university, branches, students, faculties, courses, library, clubs etc. The main aim of this project is to manage the database in such a way that information can be retrieved and modified in an efficient way.

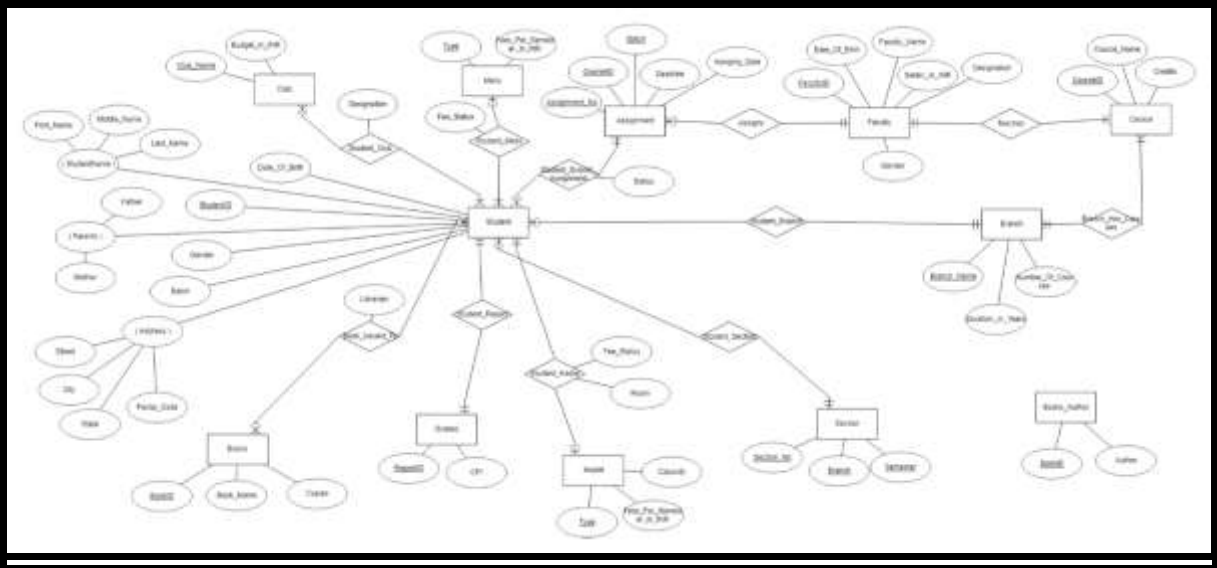
⇨ **Submitted in Previous Report**:

- ER Diagram
- Conversion from ER to Relational Model
- MySQL Database
- SQL Queries on the database
- Normal Forms and their Justification

⇨ **This Report Includes**:

- Modification in ER Diagram
- Schema's of all the database tables
- Functional Dependencies
- Closure of Functional Dependencies
- Minimal Cover of Functional Dependencies
- Verification of Minimal Cover of Functional Dependencies
- Verification of Lossless Join and Dependency Preserving Decomposition

Entity-Relationship Diagram



(The ER diagram is also submitted separately in the zip file as it is not clearly visible here.)

We have increased the number of entities and relations according to our needs as we proceeded with the project.

MySQL Database

This report deals mainly with the functional dependencies and their closures and minimal cover. Hence, instead of pasting screenshots of all the database tables along with its data, we have just pasted the schemas of all the tables (there are 26 tables).

List of all tables in the database:

```
mysql> SHOW TABLES;
+--------------------------------+
| Tables_in_University_Database |
+--------------------------------+
| Assignment                     |
| Assigns                        |
| Book_Issued_To                 |
| Books                          |
| Books_Author                   |
| Branch                         |
| Branch_Has_Course              |
| Club                           |
| Course                         |
| Faculty                        |
| Grades                         |
| Hostel                         |
| Mess                           |
| Section                        |
| Student                        |
| Student_Address                |
| Student_Branch                 |
| Student_Club                   |
| Student_Hostel                 |
| Student_Mess                   |
| Student_Name                   |
| Student_Parents                |
| Student_Report                 |
| Student_Section                |
| Student_Submits_Assignment     |
| Teaches                        |
+--------------------------------+
26 rows in set (0.00 sec)
```

As you can see, there are 26 tables overall. The description of all of the tables is pasted below.

```
mysql> DESC Assignment;
+-----------------+----------+------+-----+---------+-------+
| Field           | Type     | Null | Key | Default | Extra |
+-----------------+----------+------+-----+---------+-------+
| Assignment_No   | int      | NO   | PRI | NULL    |       |
| CourseID        | int      | NO   | PRI | NULL    |       |
| Batch           | year     | NO   | PRI | NULL    |       |
| Assigning_Date  | date     | YES  |     | NULL    |       |
| Deadline        | datetime | YES  |     | NULL    |       |
+-----------------+----------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql> DESC Assigns;
+---------------+------+------+-----+---------+-------+
| Field         | Type | Null | Key | Default | Extra |
+---------------+------+------+-----+---------+-------+
| FacultyID     | int  | NO   | PRI | NULL    |       |
| Assignment_No | int  | NO   | PRI | NULL    |       |
| CourseID      | int  | NO   | PRI | NULL    |       |
| Batch         | year | NO   | PRI | NULL    |       |
+---------------+------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> DESC Book_Issued_To;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StudentID | int         | NO   | PRI | NULL    |       |
| BookID    | int         | NO   | PRI | NULL    |       |
| Librarian | varchar(50) | NO   |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DESC Books;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| BookID    | int          | NO   | PRI | NULL    |       |
| Book_Name | varchar(100) | NO   |     | NULL    |       |
| Copies    | int          | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> DESC Books_Author;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| BookID | int          | NO   | PRI | NULL    |       |
| Author | varchar(100) | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> DESC Branch;
+-------------------+--------------+------+-----+---------+-------+
| Field             | Type         | Null | Key | Default | Extra |
+-------------------+--------------+------+-----+---------+-------+
| Branch_Name       | varchar(100) | NO   | PRI | NULL    |       |
| Duration_in_Years | int          | NO   |     | NULL    |       |
| Number_Of_Courses | int          | NO   |     | NULL    |       |
+-------------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> DESC Branch_Has_Course;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Branch_Name | varchar(100) | NO   | PRI | NULL    |       |
| CourseID    | int          | NO   | PRI | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> DESC Club;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Club_Name     | varchar(20) | NO   | PRI | NULL    |       |
| Budget_in_INR | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> DESC Course;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| CourseID    | int          | NO   | PRI | NULL    |       |
| Course_Name | varchar(100) | NO   |     | NULL    |       |
| Credits     | int          | YES  |     | 0       |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

```
mysql> DESC Faculty;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| FacultyID     | int          | NO   | PRI | NULL    |       |
| Faculty_Name  | varchar(100) | NO   |     | NULL    |       |
| Date_Of_Birth | date         | NO   |     | NULL    |       |
| Salary_in_INR | int          | YES  |     | NULL    |       |
| Designation   | varchar(50)  | NO   |     | NULL    |       |
| Gender        | varchar(20)  | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> DESC Grades;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| ReportID | int         | NO   | PRI | NULL    | auto_increment |
| CPI      | double(8,6) | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> DESC Hostel;
+--------------------------+--------------+------+-----+---------+-------+
| Field                    | Type         | Null | Key | Default | Extra |
+--------------------------+--------------+------+-----+---------+-------+
| Type                     | varchar(100) | NO   | PRI | NULL    |       |
| Fees_Per_Semester_in_INR | int          | YES  |     | NULL    |       |
| Capacity                 | int          | YES  |     | NULL    |       |
+--------------------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DESC Mess;
+--------------------------+-------------+------+-----+---------+-------+
| Field                    | Type        | Null | Key | Default | Extra |
+--------------------------+-------------+------+-----+---------+-------+
| Type                     | varchar(50) | NO   | PRI | NULL    |       |
| Fees_Per_Semester_in_INR | int         | YES  |     | NULL    |       |
+--------------------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> DESC Section;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Section_No  | int          | NO   | PRI | NULL    |       |
| Branch_Name | varchar(100) | NO   | PRI | NULL    |       |
| Semester    | int          | NO   | PRI | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DESC Student;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| StudentID     | int         | NO   | PRI | NULL    |       |
| Date_Of_Birth | date        | NO   |     | NULL    |       |
| Gender        | varchar(20) | NO   |     | NULL    |       |
| Batch         | year        | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> DESC Student_Address;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| StudentID   | int         | NO   | PRI | NULL    |       |
| Street      | varchar(50) | YES  |     | NULL    |       |
| City        | varchar(30) | NO   |     | NULL    |       |
| State       | varchar(30) | NO   |     | NULL    |       |
| Postal_Code | varchar(10) | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> DESC Student_Branch;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| StudentID   | int          | NO   | PRI | NULL    |       |
| Branch_Name | varchar(100) | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
mysql> DESC Student_Club;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| StudentID   | int         | NO   | PRI | NULL    |       |
| Club_Name   | varchar(20) | NO   | PRI | NULL    |       |
| Designation | varchar(30) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DESC Student_Hostel;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| StudentID  | int          | NO   | PRI | NULL    |       |
| Type       | varchar(100) | YES  | MUL | NULL    |       |
| Fee_Status | varchar(20)  | YES  |     | NULL    |       |
| Room       | int          | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> DESC Student_Mess;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| StudentID  | int         | NO   | PRI | NULL    |       |
| Type       | varchar(50) | YES  | MUL | NULL    |       |
| Fee_Status | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DESC Student_Name;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| StudentID   | int         | NO   | PRI | NULL    |       |
| First_Name  | varchar(30) | NO   |     | NULL    |       |
| Middle_Name | varchar(20) | YES  |     | NULL    |       |
| Last_Name   | varchar(20) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> DESC Student_Parents;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StudentID | int         | NO   | PRI | NULL    |       |
| Father    | varchar(50) | YES  |     | NULL    |       |
| Mother    | varchar(50) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DESC Student_Report;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| StudentID | int  | NO   | MUL | NULL    |       |
| ReportID  | int  | NO   | PRI | NULL    |       |
+-----------+------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> DESC Student_Section;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| StudentID  | int  | NO   | PRI | NULL    |       |
| Section_No | int  | YES  |     | NULL    |       |
| Semester   | int  | NO   | PRI | NULL    |       |
+------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DESC Student_Submits_Assignment;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| StudentID     | int         | NO   | PRI | NULL    |       |
| Assignment_No | int         | NO   | PRI | NULL    |       |
| CourseID      | int         | NO   | PRI | NULL    |       |
| Batch         | year        | NO   | PRI | NULL    |       |
| Status        | varchar(20) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> DESC Teaches;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| FacultyID | int  | NO   | PRI | NULL    |       |
| CourseID  | int  | NO   | PRI | NULL    |       |
+-----------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Closure and Minimal Cover of Functional Dependencies

In this section, we have mentioned all the functional dependencies that are present in our database. We have formed their closures and minimal covers. Since, there are a large number of schemas, we have shown how to find closure and minimal cover for some schemas.

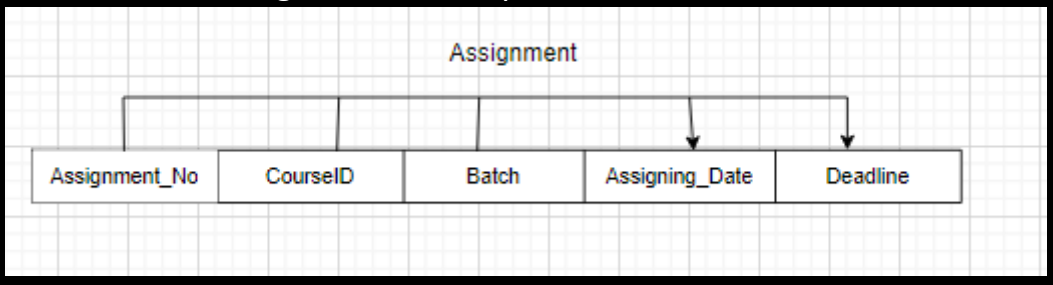
- ⇨ The **closure of a set of functional dependency** F, denoted by F^+, is the set of all functional dependencies that can be derived from F using the Armstrong's Rules.
  Note: Reflexive Functional Dependencies on a schema $\{A, B, C, D, ..\}$ such as $\{A \rightarrow A, AB \rightarrow A,\ ABC \rightarrow AB\}$ etc. are also a part of F^+, but these dependencies are trivial and hence will not be written in F^+. We have mainly written the transitive, pseudo-transitive and union rule dependencies, as they can be non-trivial. The candidate keys will be highlighted in red.

- ⇨ The **minimal cover F_c, of a set F of functional dependencies** is the minimum set of dependencies such that $F^+ = F_c^+$.

  Steps to find Minimal Cover of set F of functional dependencies:

  - ❖ Using decomposition rule, replace all functional dependencies of the form $A \rightarrow BC$ by $A \rightarrow B$ and $A \rightarrow C$.
  - ❖ Check for any extraneous attribute in the new set of functional dependencies. If there are any such attributes, remove them.
  - ❖ Check for any redundant dependency. If there is any, remove them.

- **In Schema "Assignment":**
  We want the following functional dependencies to hold.



For the sake of writing, let's just give each attribute an easier name, according to the following table.

Original Name	Assignment_No	CourseID	Batch	Assigning_Date	Deadline
Given Name	A	B	C	D	E

Therefore, the set F of functional dependencies contains:

$$ABC \rightarrow D$$
$$ABC \rightarrow E$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A,\ ABC \rightarrow AB, ABC \rightarrow D, ABC \rightarrow E,$$
$$ABCE \rightarrow DE, ABCD \rightarrow DE, ABC \rightarrow DE, ABC \rightarrow ABCDE\}$$

- Functional dependencies $\{ABCE \rightarrow DE, ABCD \rightarrow DE\}$ are derived using the Augmentation Rule on the dependencies in F. More dependencies can be derived using Augmentation Rule (however, they will be trivial).
- No Transitive Dependency
- $\{ABC \rightarrow DE, ABC \rightarrow ABCDE\}$ is derived using Union Rule.

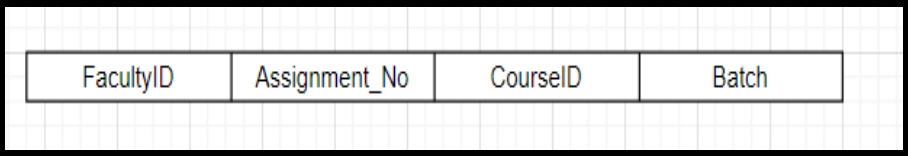
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency. This can be checked in the following way.
  - Remove the dependency $ABC \rightarrow D$ and find the closure of ABC using remaining dependencies in F. Since, $(ABC)^+ = \{A, B, C, E\}$, which does not contain D. Hence, $ABC \rightarrow D$ is not a redundant functional dependency.
  - Similarly, we can test $ABC \rightarrow E$ is not redundant.

Hence, $F_c = F$

- **In Schema "Assigns":**

  Only Trivial Functional Dependencies exists, therefore $F = \emptyset$.



Original Name	FacultyID	Assignment_No	CourseID	Batch
Given Name	A	B	C	D

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, ABCD \rightarrow ABCD\}$$

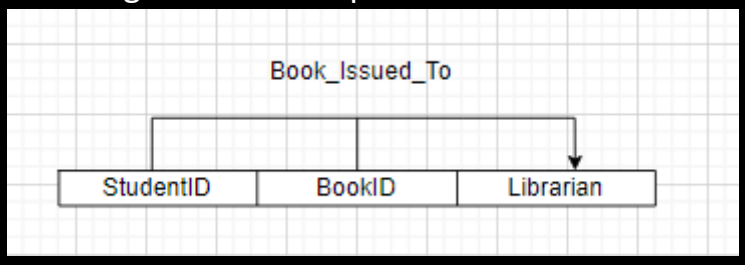
- Although $F = \emptyset$, but the closure of F contains trivial dependencies.
- Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.

➔ The minimal cover F_c for F is:

Since $F = \emptyset$, hence, $F_c = \emptyset$

- **In Schema "Book_Issued_To":**

We want the following functional dependencies to hold.



Original Name	StudentID	BookID	Librarian
Given Name	A	B	C

Therefore, the set F of functional dependencies contains:

$$AB \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, AB \rightarrow AB, ABC \rightarrow AB, AB \rightarrow C, ABC \rightarrow AC, AB \rightarrow ABC\}$$

- Functional dependencies $\{ABC \rightarrow AC\}$ are derived using the Augmentation Rule on the dependency $\{AB \rightarrow A\}$. More dependencies can be derived using Augmentation Rule (however, they will be trivial).
- No Transitive Dependency
- $\{AB \rightarrow AC, AB \rightarrow ABC\}$ is derived using Union Rule on $\{AB \rightarrow A, AB \rightarrow C\}$.

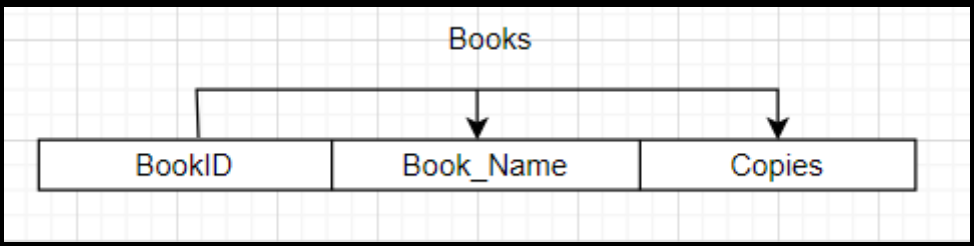
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Books":**

We want the following functional dependencies to hold.



Original Name	BookID	Book_Name	Copies
Given Name	A	B	C

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, A \rightarrow ABC\}$$

- Functional dependencies $\{AC \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BC, A \rightarrow ABC\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B, A \rightarrow C\}$.

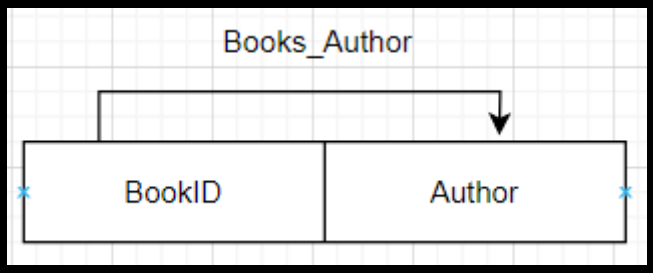
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Books_Author":**

We want the following functional dependencies to hold.



Original Name	BookID	Author
Given Name	A	B

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$

➔ The closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- No Transitive Dependency
- $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

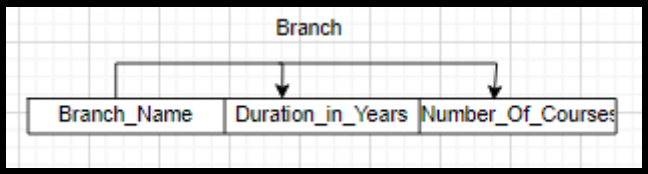
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Branch":**

We want the following functional dependencies to hold.



Original Name	Branch_Name	Duration_in_Years	Number_Of_Courses
Given Name	A	B	C

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow BC, A \rightarrow ABC\}$$

- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \text{ and } A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BC\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C\}$.

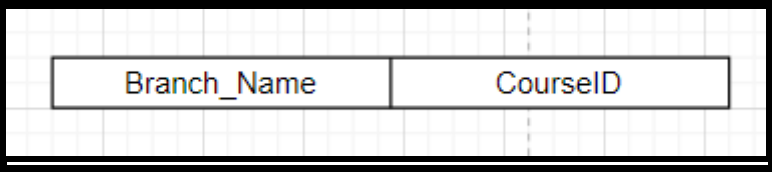
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Branch_Has_Course":**

  Only Trivial Functional Dependencies exists, therefore $F = \emptyset$.



Original Name	Branch_Name	CourseID
Given Name	A	B

➔ The closure of F, F^+, is:

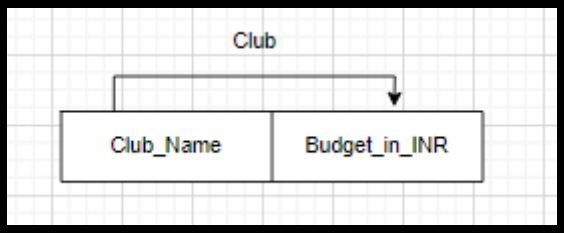
$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, AB \rightarrow AB\}$$

➢ Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.

➔ The minimal cover F_c for F is:
Since $F = \emptyset$, hence, $F_c = \emptyset$

- **In Schema "Club":**

We want the following functional dependencies to hold.



Original Name	Club_Name	Budget_in_INR
Given Name	A	B

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$

➔ The closure of F, F^+, is:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- No Transitive Dependency
- $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

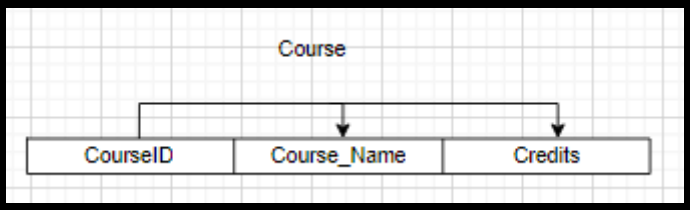
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Course":**

We want the following functional dependencies to hold.



Original Name	CourseID	Course_Name	Credits
Given Name	A	B	C

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow ABC\}$$

- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \text{ and } A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow ABC\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B, A \rightarrow C\}$.

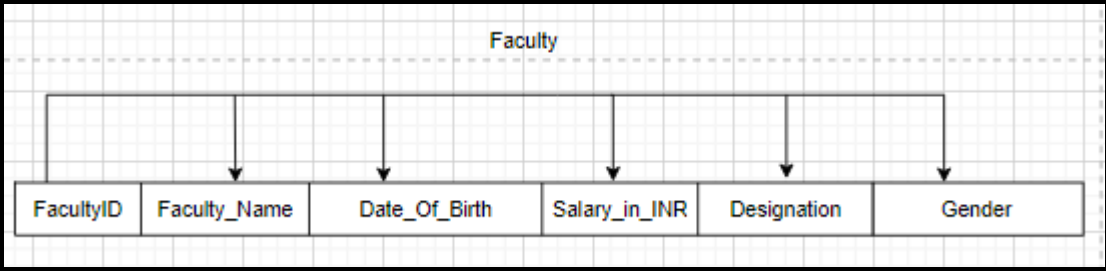
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Faculty":**

We want the following functional dependencies to hold.



Original Name	FacultyID	Faculty_Name	Date_Of_Birth	Salary_in_INR	Designation	Gender
Given Name	A	B	C	D	E	F

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$
$$A \rightarrow D$$
$$A \rightarrow E$$
$$A \rightarrow F$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E, A \rightarrow F, AC \rightarrow BC, AF \rightarrow BF, A \rightarrow BC, A \rightarrow ABCDEF\}$$

- Functional dependencies $\{AC \rightarrow BC, AF \rightarrow BF\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \text{ and } A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency

- $\{A \rightarrow BC, A \rightarrow ABCDEF\}$ are derived using Union Rule.

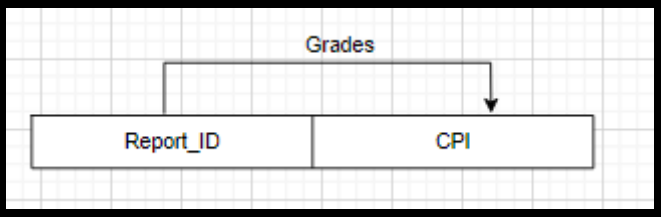
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency. This can be checked in the following way.
  - Remove the dependency $A \rightarrow B$ and find the closure of A using remaining dependencies in F. Since, $(A)^+ = \{A, C, D, E, F\}$, which does not contain B. Hence, $A \rightarrow B$ is not a redundant functional dependency.
  - Similarly, we can test for other functional dependencies in F.

Hence, $F_c = F$

- **<u>In Schema "Grades":</u>**

We want the following functional dependencies to hold.



Original Name	ReportID	CPI
Given Name	A	B

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$

➔ The closure of F, F^+, is:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- No Transitive Dependency
- $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

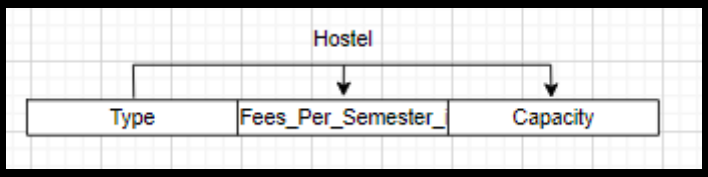
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Hostel":**

We want the following functional dependencies to hold.



Original Name	Type	Fees_Per_Semester_in_INR	Capacity
Given Name	A	B	C

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow BC, A \rightarrow ABC\}$$

- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \; and \; A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BC\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C\}$.

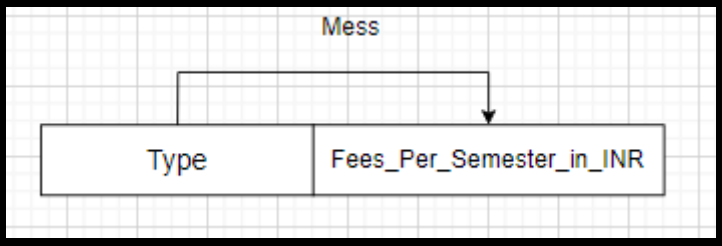
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Mess":**

  We want the following functional dependencies to hold.



Original Name	ReportID	CPI
Given Name	A	B

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$

➔ The closure of F, F^+, is:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- No Transitive Dependency
- $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

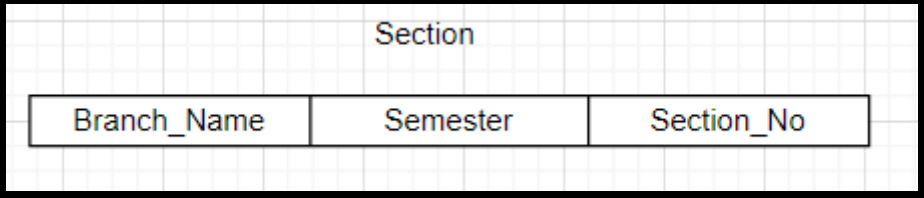
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Section":**

  Only Trivial Functional Dependencies exists, therefore $F = \emptyset$.



Original Name	Branch_Name	Semester	Section_No

Given Name	A	B	C

➔ Few dependencies from the closure of F, F^+, are:

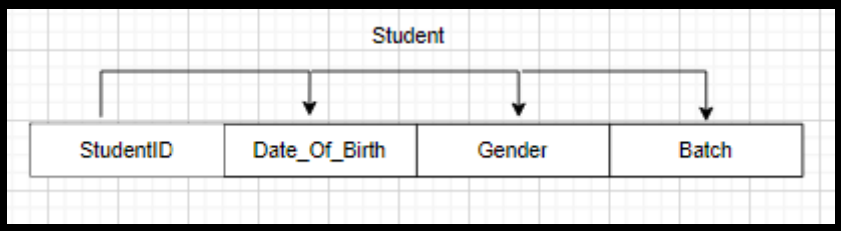
$$F^+ = \{A \rightarrow A, B \rightarrow B, ABC \rightarrow AB, ABC \rightarrow ABC\}$$

- Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.

➔ The minimal cover F_c for F is:
Since $F = \emptyset$, hence, $F_c = \emptyset$

- **In Schema "Student":**

We want the following functional dependencies to hold.



Original Name	StudentID	Date_Of_Birth	Gender	Batch
Given Name	A	B	C	D

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$
$$A \rightarrow D$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, A \rightarrow D, AC \rightarrow BC, AD \rightarrow BD, A \rightarrow BCD, A \rightarrow ABCD\}$$

- Functional dependencies $\{AC \rightarrow BC, AD \rightarrow BD\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency

- $\{A \rightarrow BCD, A \rightarrow ABCD\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C, A \rightarrow D\}$.

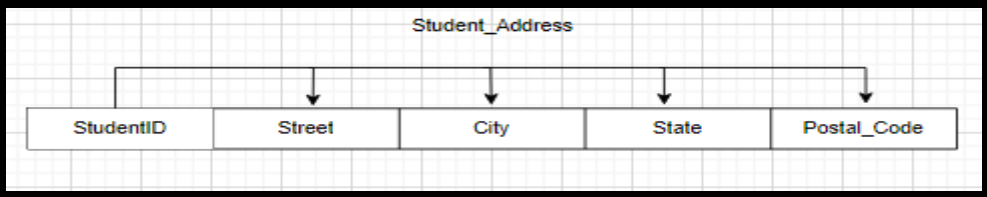
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency. This can be checked in the following way.
  - Remove the dependency $A \rightarrow B$ and find the closure of A using remaining dependencies in F. Since, $(A)^+ = \{A, C, D\}$, which does not contain B. Hence, $A \rightarrow B$ is not a redundant functional dependency.
  - Similarly, we can test for other functional dependencies in F.

Hence, $F_c = F$

- **<u>In Schema "Student_Address":</u>**

We want the following functional dependencies to hold.



Original Name	StudentID	Street	City	State	Postal_Code
Given Name	A	B	C	D	E

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$
$$A \rightarrow D$$
$$A \rightarrow E$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E, AC \rightarrow BC, AE \rightarrow BE, A \rightarrow BC, A \rightarrow ABCDE\}$$

- ➢ Functional dependencies $\{AC \rightarrow BC, AE \rightarrow BE\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \; and \; A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- ➢ No Transitive Dependency
- ➢ $\{A \rightarrow BC, A \rightarrow ABCDE\}$ are derived using Union Rule.

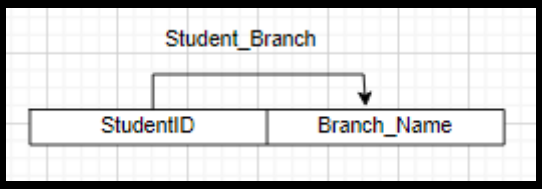
➔ The minimal cover F_c for F is:

- ➢ F does not contain any extraneous attribute.
- ➢ F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Student_Branch":**

We want the following functional dependencies to hold.



Original Name	StudentID	Branch_Name
Given Name	A	B

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$

➔ The closure of F, F^+, is:

$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, A \rightarrow B, A \rightarrow AB\}$$

- ➢ No Transitive Dependency
- ➢ $\{A \rightarrow AB\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B\}$.

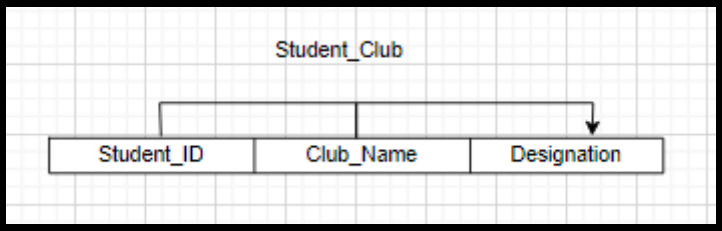
➔ The minimal cover F_c for F is:

- ➢ F does not contain any extraneous attribute.
- ➢ F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Student_Club":**

  We want the following functional dependencies to hold.



Original Name	StudentID	Club_Name	Designation
Given Name	A	B	C

Therefore, the set F of functional dependencies contains:

$$AB \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, AB \rightarrow C, AB \rightarrow AC, AB \rightarrow ABC\}$$

- More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{AB \rightarrow AC,\ AB \rightarrow ABC\}$ are derived using Union Rule.

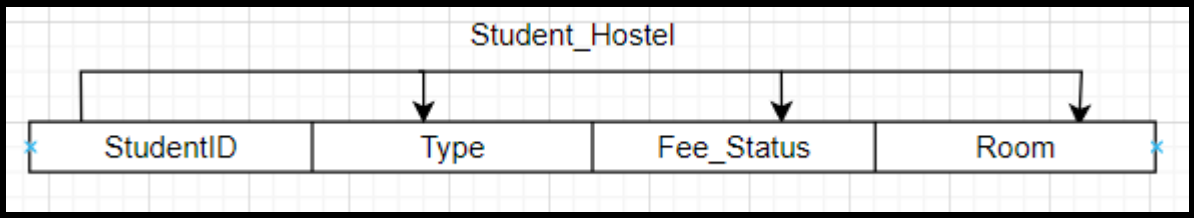
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Student_Hostel":**

  We want the following functional dependencies to hold.



Original Name	StudentID	Type	Fee_Status	Room
Given Name	A	B	C	D

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$
$$A \rightarrow D$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, A \rightarrow D, AC \rightarrow BC, AD \rightarrow BD, A \rightarrow BCD, A \rightarrow ABCD\}$$

- Functional dependencies $\{AC \rightarrow BC, AD \rightarrow BD\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BCD, A \rightarrow ABCD\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C, A \rightarrow D\}$.

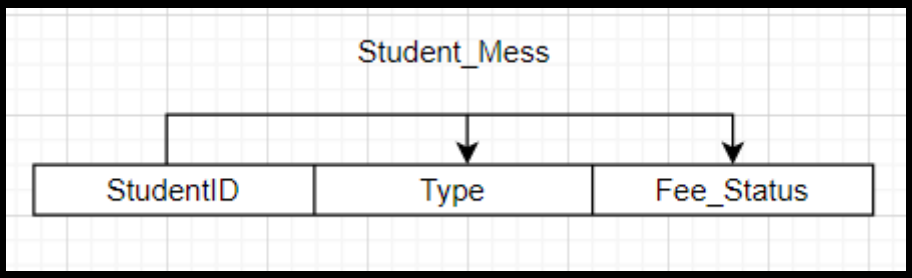
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency. This can be checked in the following way.
  - Remove the dependency $A \rightarrow B$ and find the closure of A using remaining dependencies in F. Since, $(A)^+ = \{A, C, D\}$, which does not contain B. Hence, $A \rightarrow B$ is not a redundant functional dependency.
  - Similarly, we can test for other functional dependencies in F.

Hence, $F_c = F$

- **In Schema "Student_Mess":**

We want the following functional dependencies to hold.



Original Name	StudentID	Type	Fee_Status
Given Name	A	B	C

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow BC, A \rightarrow ABC\}$$

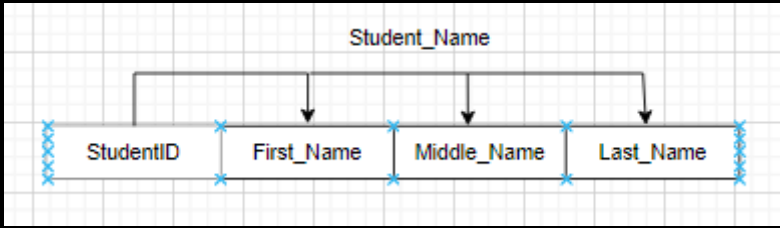
- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \; and \; A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow BC, A \rightarrow ABC\}$ is derived using Union Rule on $\{A \rightarrow A, A \rightarrow B, A \rightarrow C\}$.

➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **<u>In Schema "Student_Name":</u>**
  We want the following functional dependencies to hold.



Original Name	StudentID	First_Name	Middle_Name	Last_Name
Given Name	A	B	C	D

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$
$$A \rightarrow D$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, A \rightarrow D, AC \rightarrow BC, AD \rightarrow BD, A \rightarrow ABCD\}$$

- Functional dependencies $\{AC \rightarrow BC, AD \rightarrow BD\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow ABCD\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C, A \rightarrow D\}$.

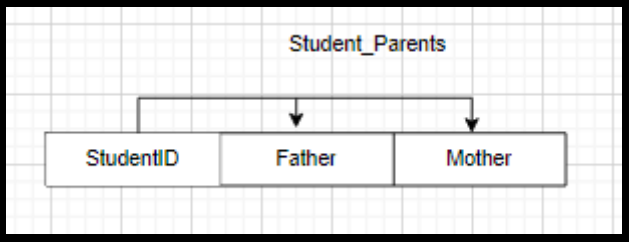
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency. This can be checked in the following way.
  - Remove the dependency $A \rightarrow B$ and find the closure of A using remaining dependencies in F. Since, $(A)^+ = \{A, C, D\}$, which does not contain B. Hence, $A \rightarrow B$ is not a redundant functional dependency.
  - Similarly, we can test for other functional dependencies in F.

Hence, $F_c = F$

- **In Schema "Student_Parents":**

  We want the following functional dependencies to hold.



Original Name	StudentID	Type	Fee_Status
Given Name	A	B	C

Therefore, the set F of functional dependencies contains:

$$A \rightarrow B$$
$$A \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, A \rightarrow B, A \rightarrow C, AC \rightarrow BC, AB \rightarrow BC, A \rightarrow ABC\}$$

- Functional dependencies $\{AC \rightarrow BC, AB \rightarrow BC\}$ are derived using the Augmentation Rule on the dependency $\{A \rightarrow B \text{ and } A \rightarrow C\}$. More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{A \rightarrow ABC\}$ is derived using Union Rule on $\{A \rightarrow B, A \rightarrow C\}$.

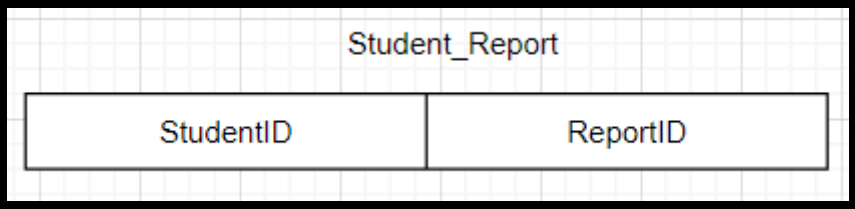
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Student_Report":**

  Only Trivial Functional Dependencies exists, therefore $F = \emptyset$.



Original Name	StudentID	ReportID
Given Name	A	B

➔ The closure of F, F^+, is:

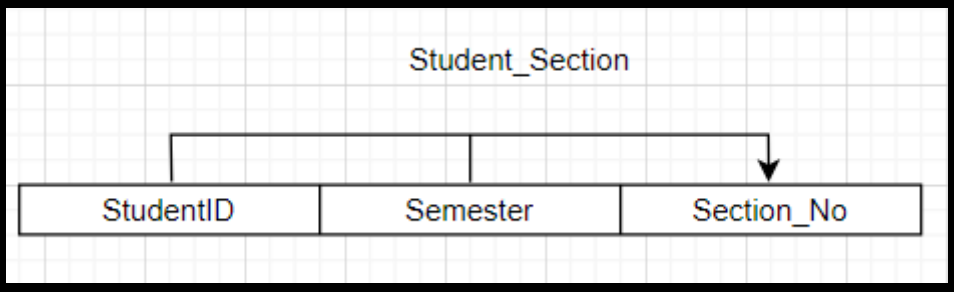
$$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, AB \rightarrow AB\}$$

- Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.

➔ The minimal cover F_c for F is:
Since $F = \emptyset$, hence, $F_c = \emptyset$

- **In Schema "Student_Section":**

We want the following functional dependencies to hold.



Original Name	StudentID	Semester	Section_No
Given Name	A	B	C

Therefore, the set F of functional dependencies contains:

$$AB \rightarrow C$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABC \rightarrow AB, AB \rightarrow C, AB \rightarrow AC, AB \rightarrow ABC\}$$

- More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{AB \rightarrow AC,\ AB \rightarrow ABC\}$ are derived using Union Rule.

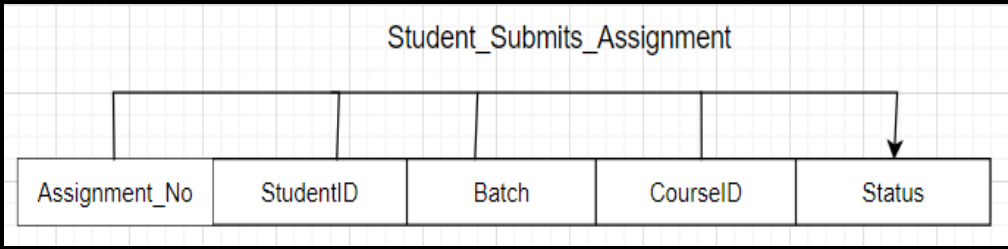
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Student_Submits_Assignment":**

We want the following functional dependencies to hold.



Original Name	Assignment_No	StudentID	Batch	CourseID	Status
Given Name	A	B	C	D	E

Therefore, the set F of functional dependencies contains:

$$ABCD \rightarrow E$$

➔ Few dependencies from the closure of F, F^+, are:

$$F^+ = \{A \rightarrow A, AB \rightarrow A, ABCD \rightarrow AB, ABCD \rightarrow E, ABCD \rightarrow ABCDE, ABCD \rightarrow ABCDE\}$$

- More dependencies can be derived using Augmentation Rule (however, most of them will be trivial).
- No Transitive Dependency
- $\{ABCD \rightarrow ABE\}$ are derived using Union Rule.

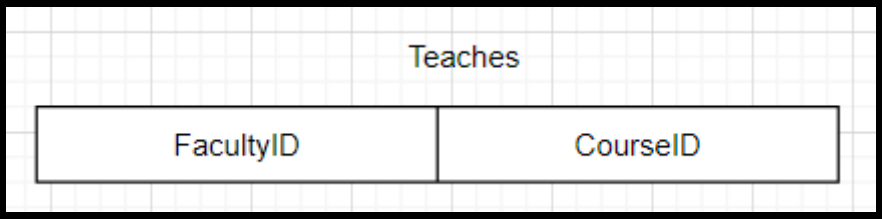
➔ The minimal cover F_c for F is:

- F does not contain any extraneous attribute. This can be checked in the following way.
  - Checking if B is extraneous attribute in $ABCD \rightarrow E$. Remove this dependency from F and add the dependency $ACD \rightarrow E$.
  - Now, form the attribute closure of each of A, C and D.
  - $A^+ = \{A\},\ C^+ = \{C\},\ D^+ = \{D\}$
  - Since, none of these attribute closures contains B, hence, B is not an extraneous attribute.
- F does not contain any redundant dependency.

Hence, $F_c = F$

- **In Schema "Teaches":**

Only Trivial Functional Dependencies exists, therefore $F = \emptyset$.



Original Name	FacultyID	CourseID
Given Name	A	B

➔ The closure of F, F^+, is:

$F^+ = \{A \rightarrow A, B \rightarrow B, AB \rightarrow A, AB \rightarrow B, AB \rightarrow AB\}$

- Applying reflexive and augmentation rule to the attributes of the schema, we will get several dependencies as are highlighted above.

➔ The minimal cover F_c for F is:

Since $F = \emptyset$, hence, $F_c = \emptyset$

Verifying the Minimal Cover

In this section, we have explained why the minimal covers found for each schema in the previous section are equivalent to the original set of functional dependencies.

- ⇨ To verify, if the minimal cover F_c is equivalent to the original set F of functional dependency, we need to show that,

  $$(F_c)^+ = F^+$$

  That is, the closure of F_c and F must be equal.

- ⇨ To simplify this, we can begin forming the closure of F_c and if at any point of time, we have $F \subseteq F_c'$, where F_c' is the derived set of dependencies from F_c. This is because when we have F as a subset of F_c', then we have all dependencies in F_c' that are originally present in F. Hence, F^+ can be formed from F_c'.
- ⇨ Since, for each schema in the database, the minimal cover F_c is equal to F. Hence, their closures will also be equal. Hence, the minimal cover is equivalent to the original set of functional dependencies for each schema.

Verifying the Lossless Join Property by Matrix Method

Steps to find if decomposition is Lossless and dependency preserving

- Fill $'\alpha'$ in each schema row, where the corresponding attribute is present in the schema.
- For each functional dependency in F, if two rows in the matrix have same value for a left attribute, then make the value of right attribute equal.
- If at least one row contains all $\alpha's$, then the decomposition is loseless – join and dependency preserving.

We decomposed only one schema in our previous assignment.

⇨ Schema Books(BookID, Book_Name, Author, Copies) into Books(BookID, Book_Name, Copies) and Books_Author(BookID, Author). For ease, let's use the following naming:

Original Name	Given Name
BookID	A
Book_Name	B
Author	C
Copies	D
Books(A,B,C,D)	R(A,B,C,D)
Books(A,B,D)	$R_1(A, B, D)$
Books_Author	$R_2(A, C)$

The functional dependencies on R were $F = \{A \rightarrow B,\ A \rightarrow C, A \rightarrow D,\ B \rightarrow C\}$.

The decomposed schemas are:

$$R_1(A, B, D)\ and\ R_2(A, C)$$

- Step 1

$$\begin{bmatrix} & A & B & C & D \\ R_1 & \alpha & \alpha & & \alpha \\ R_2 & \alpha & & \alpha & \end{bmatrix}$$

- Checking $A \rightarrow B$,

$$\begin{bmatrix} & A & B & C & D \\ R_1 & \alpha & \alpha & & \alpha \\ R_2 & \alpha & \alpha & \alpha & \end{bmatrix}$$

- Checking $A \rightarrow C$,

$$\begin{bmatrix} & A & B & C & D \\ R_1 & \alpha & \alpha & \alpha & \alpha \\ R_2 & \alpha & \alpha & \alpha & \end{bmatrix}$$

- Checking $A \rightarrow D$,

$$\begin{bmatrix} & A & B & C & D \\ R_1 & \alpha & \alpha & \alpha & \alpha \\ R_2 & \alpha & \alpha & \alpha & \alpha \end{bmatrix}$$

- Since, we have the first and second row, filled completely with $\alpha's$, hence our decomposition is lossless join and dependency preserving.