Relational Model

In this Lecture you will Learn about:

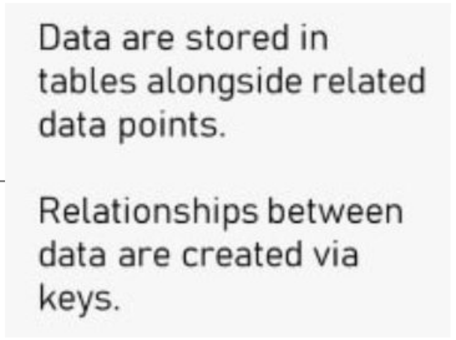
Relational Model

Basic Concepts

Relational Model

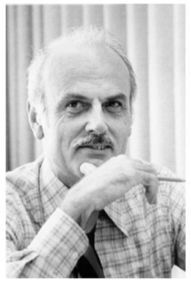
- **Relational Model (RM)** represents the database as a collection of relations.

- A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

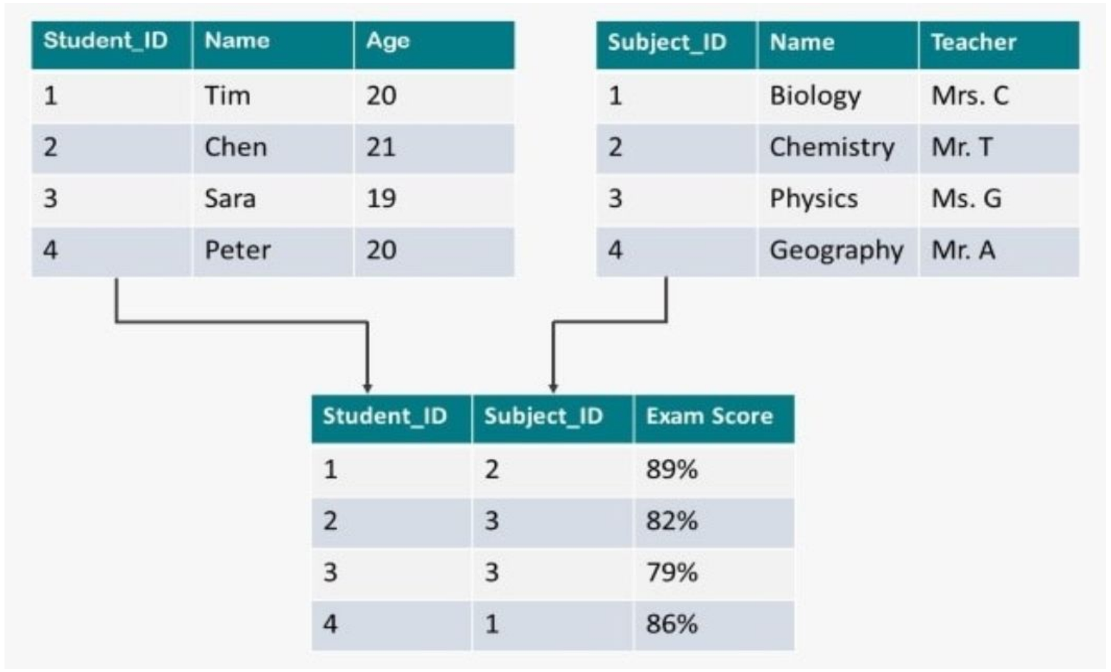

Data are stored in tables alongside related data points.

Relationships between data are created via keys.

Edgar F. Codd (1923-2003)



- Pilot in the Royal Air Force in WW2
- Inventor of the relational model and algebra while at IBM
- Turing Award, 1981



Student_ID	Name	Age
1	Tim	20
2	Chen	21
3	Sara	19
4	Peter	20

Subject_ID	Name	Teacher
1	Biology	Mrs. C
2	Chemistry	Mr. T
3	Physics	Ms. G
4	Geography	Mr. A

Student_ID	Subject_ID	Exam Score
1	2	89%
2	3	82%
3	3	79%
4	1	86%

Relational Model

- Database is a collection of Relations (or Tables)

- Each relation has a set of attributes (or Columns)

- Each attribute has a name and a domain (or Type)

- Atomic Values (no Set-valued attribute)

- Each relation contains a set of tuples (or rows)

  - Each tuple has value for each attribute of the relation

  - No duplicate tuples

-Note: Two tuples are duplicates if they agree on all attributes

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)

Basic Concepts

Employee_ID	Name	Department	Salary
001	David	HR	$50,000
002	Jane Smith	IT	$60,000
003	Mark Brown	Finance	$55,000

Basic Concepts

1. Attribute: Each column in the table represents an attribute. Attributes are properties that define the relation.

Example: In the "Employees" table, the attributes are "Employee_ID", "Name", "Department", and "Salary".

2. Tables: Relations in the relational model are stored in table format. A table consists of rows and columns. Rows represent records, and columns represent attributes.

Example: The "Employees" table is stored in a tabular format with rows and columns.

3. Tuple: A tuple is a single row of a table, containing a single record.

Example: (001, David, HR, $50,000) is a tuple representing one employee's information in the "Employees" table.

4.Relation Schema: The relation schema represents the name of the relation along with its attributes.

Example: The relation schema for the "Employees" table is Employees(Employee_ID, Name, Department, Salary).

Basic Concepts

5. Degree: The total number of attributes in a relation is called the degree of the relation.

Example: The degree of the "Employees" relation is 4, as it has 4 attributes.

6. Cardinality: The total number of rows present in the table is called the cardinality.

Example: The cardinality of the "Employees" table is 3, as it has 3 rows.

7. Column: Each column represents the set of values for a specific attribute.

Example: The "Name" column in the "Employees" table contains the set of values for the "Name" attribute.

8. Relation Instance: A relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.

Example: {(001, David, HR, $50,000), (002, Jane Smith, IT, $60,000), (003, Mark Brown, Finance, $5

Basic Concepts

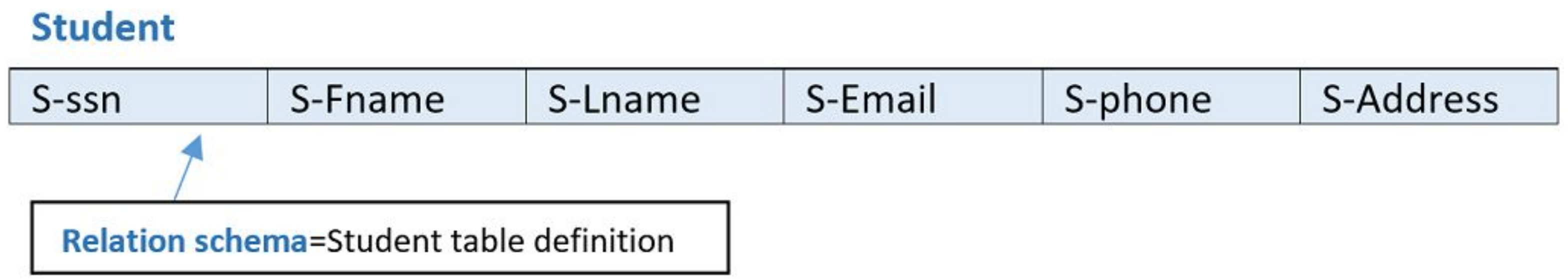
9. Relation Key: Every row has one or more attributes that uniquely identify it, which is called the relation key.

Example: In the "Employees" table, the "Employee_ID" attribute serves as the relation key as it uniquely identifies each employee.

10. Attribute Domain: Every attribute has a predefined set of values and scope, which is known as the attribute domain.

Example: The "Salary" attribute in the "Employees" table might have an attribute domain of positive integers representing the salary amount.

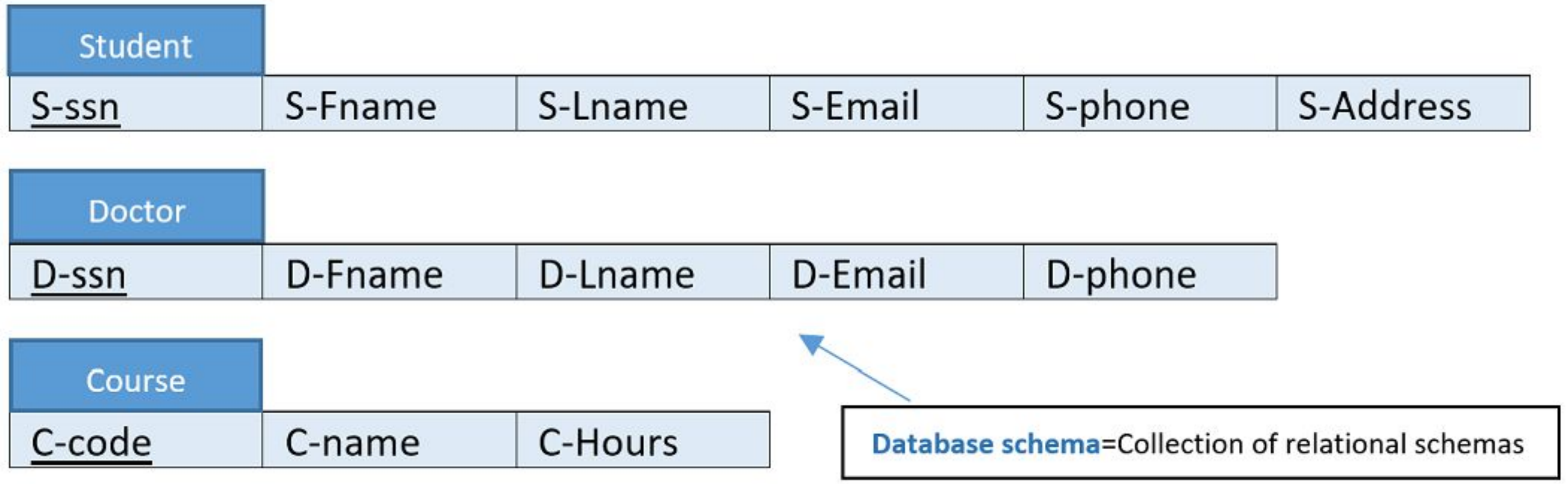
Basic Concepts



Student

S-ssn	S-Fname	S-Lname	S-Email	S-phone	S-Address

Relation schema=Student table definition



Student

S-ssn	S-Fname	S-Lname	S-Email	S-phone	S-Address

Doctor

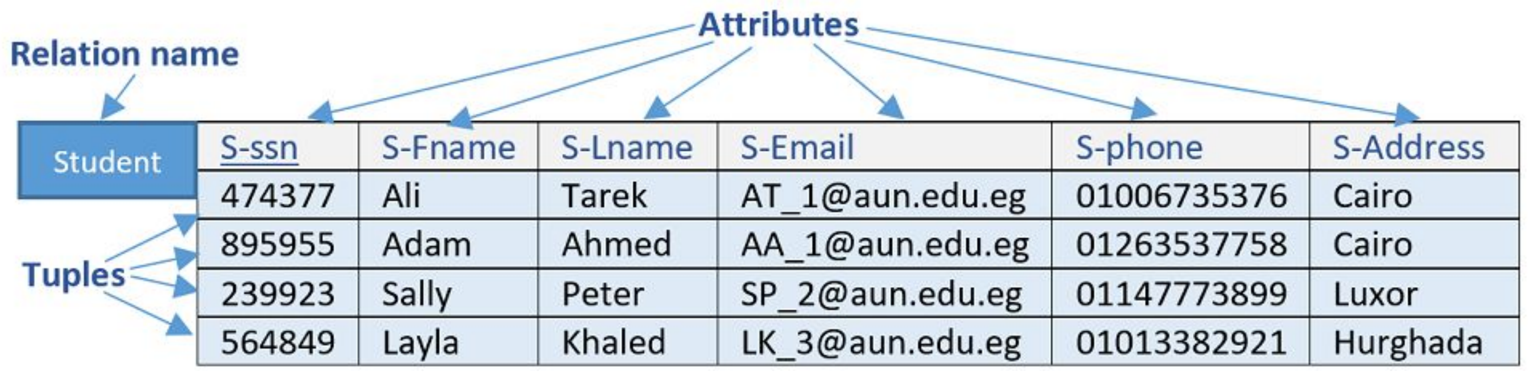
D-ssn	D-Fname	D-Lname	D-Email	D-phone

Course

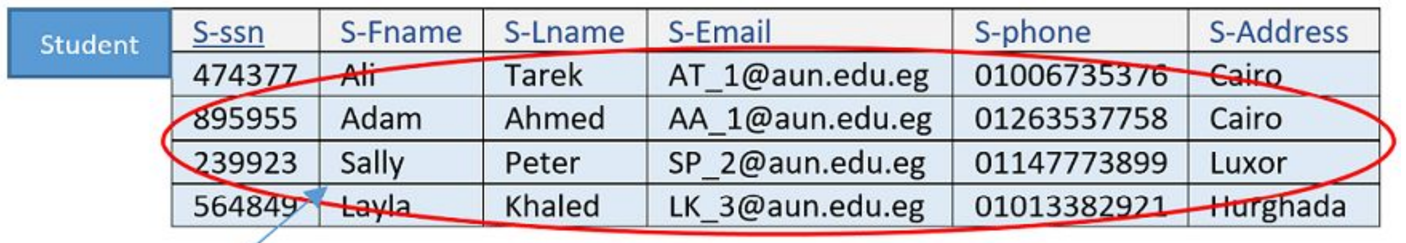
C-code	C-name	C-Hours

Database schema=Collection of relational schemas

Basic Concepts



S-ssn	S-Fname	S-Lname	S-Email	S-phone	S-Address
474377	Ali	Tarek	AT_1@aun.edu.eg	01006735376	Cairo
895955	Adam	Ahmed	AA_1@aun.edu.eg	01263537758	Cairo
239923	Sally	Peter	SP_2@aun.edu.eg	01147773899	Luxor
564849	Layla	Khaled	LK_3@aun.edu.eg	01013382921	Hurghada



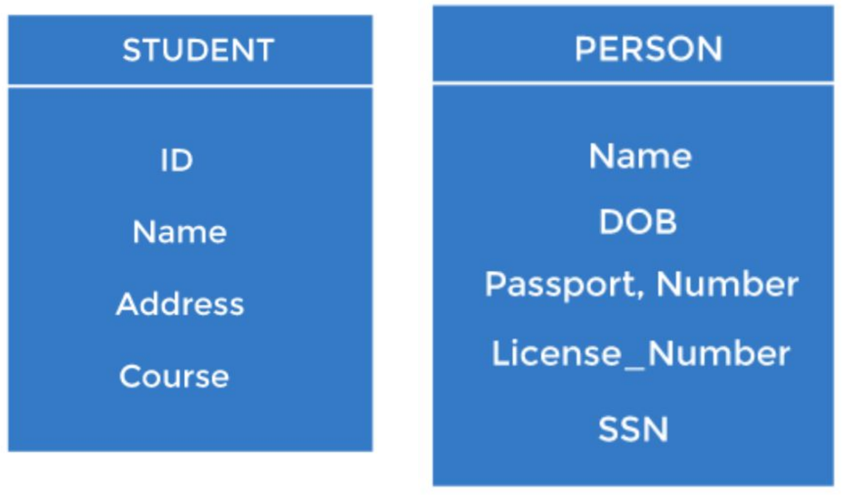
S-ssn	S-Fname	S-Lname	S-Email	S-phone	S-Address
474377	Ali	Tarek	AT_1@aun.edu.eg	01006735376	Cairo
895955	Adam	Ahmed	AA_1@aun.edu.eg	01263537758	Cairo
239923	Sally	Peter	SP_2@aun.edu.eg	01147773899	Luxor
564849	Layla	Khaled	LK_3@aun.edu.eg	01013382921	Hurghada

Database state= All student records' values at this point of time *(as values change frequently with each database operation like insert, update or delete)*

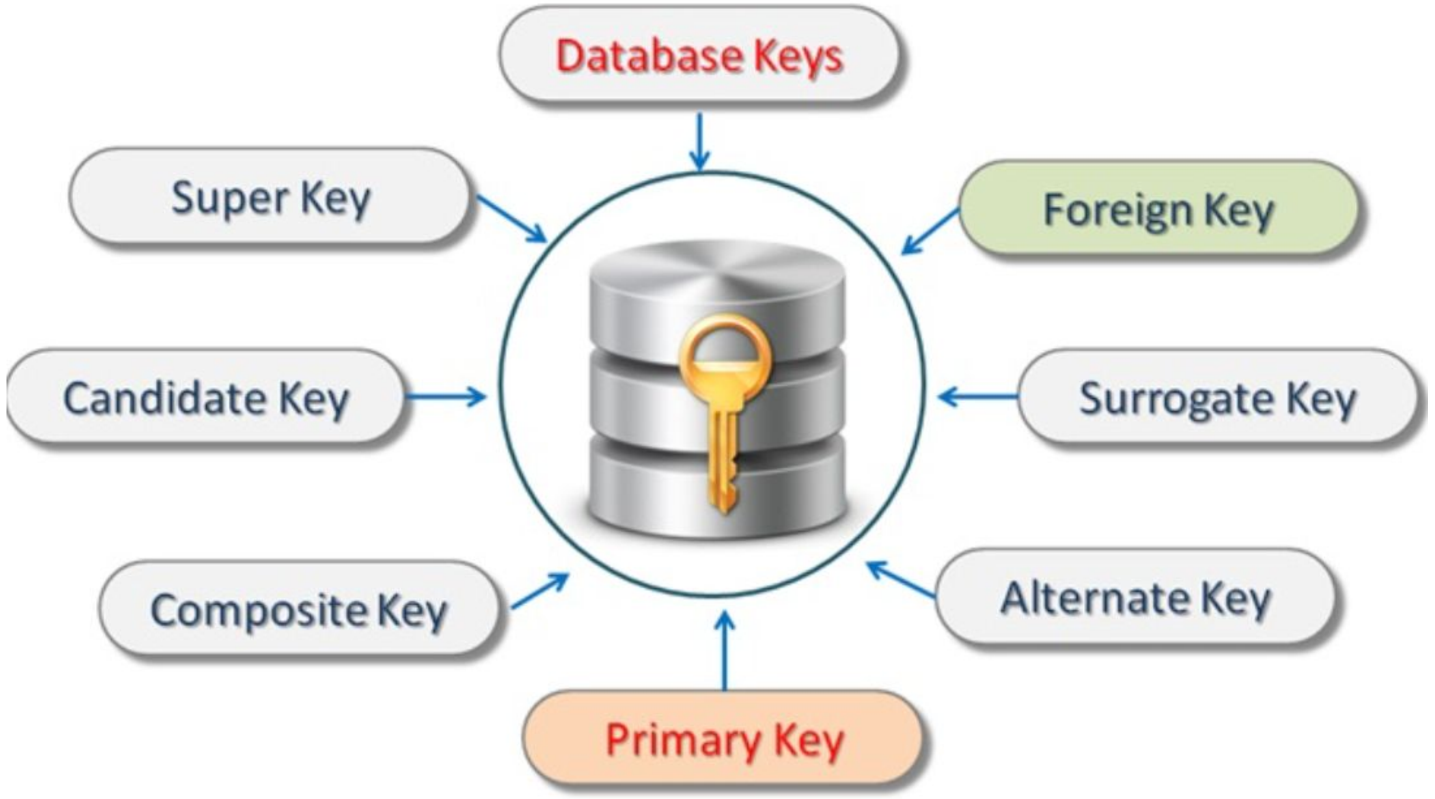
Note: At any time, **Database state** should be valid having correct values.

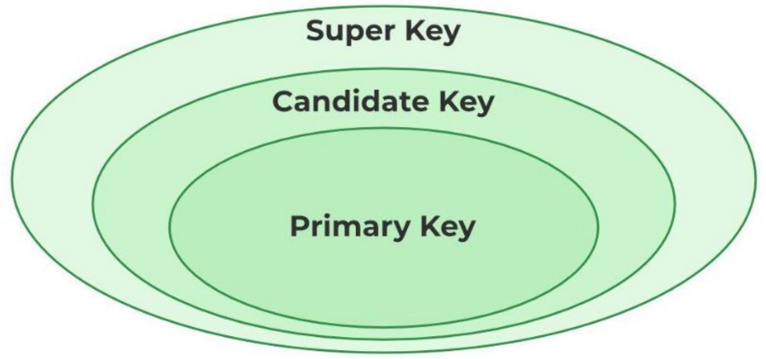
Database Keys

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.
- ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

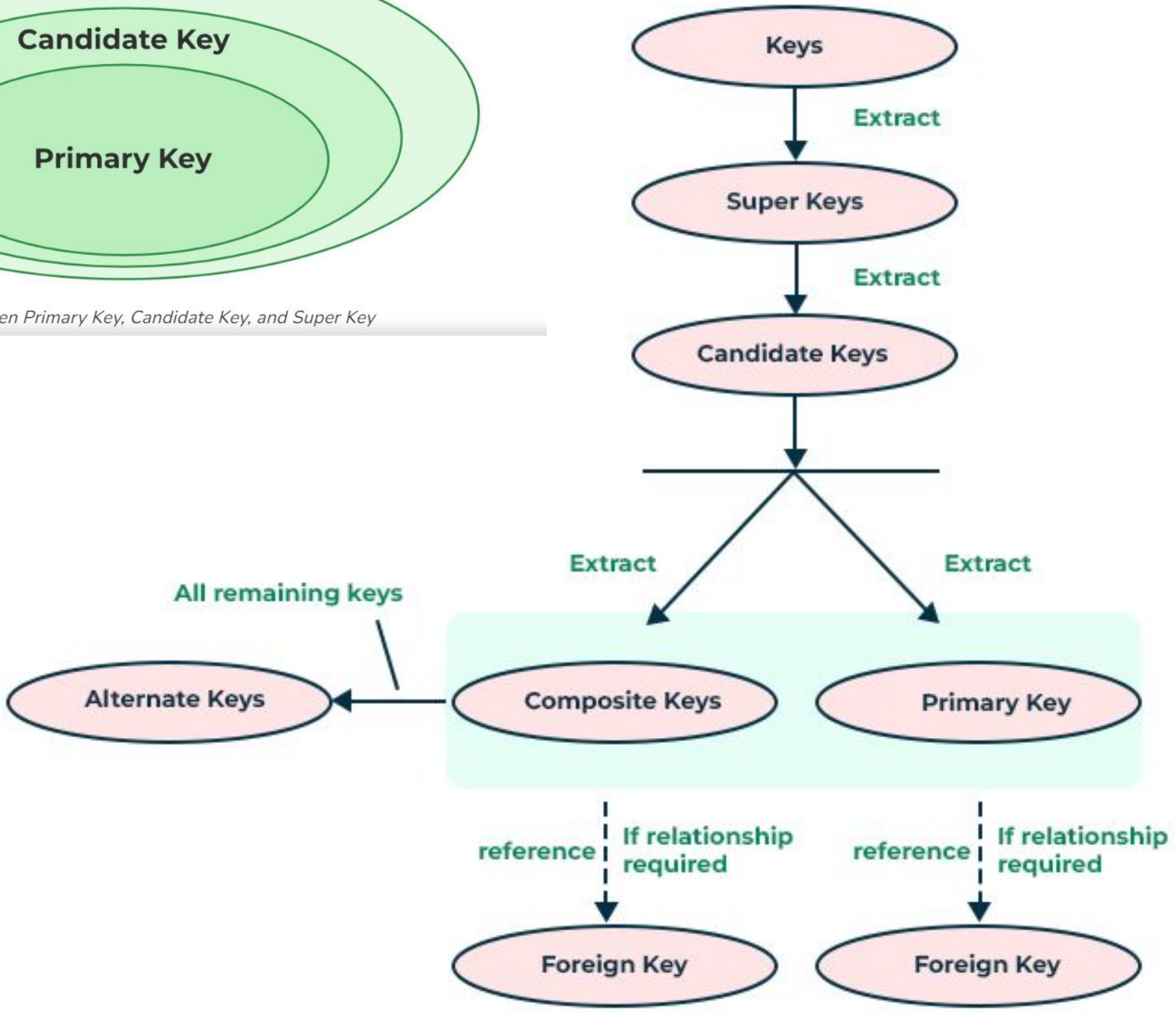


Database Keys



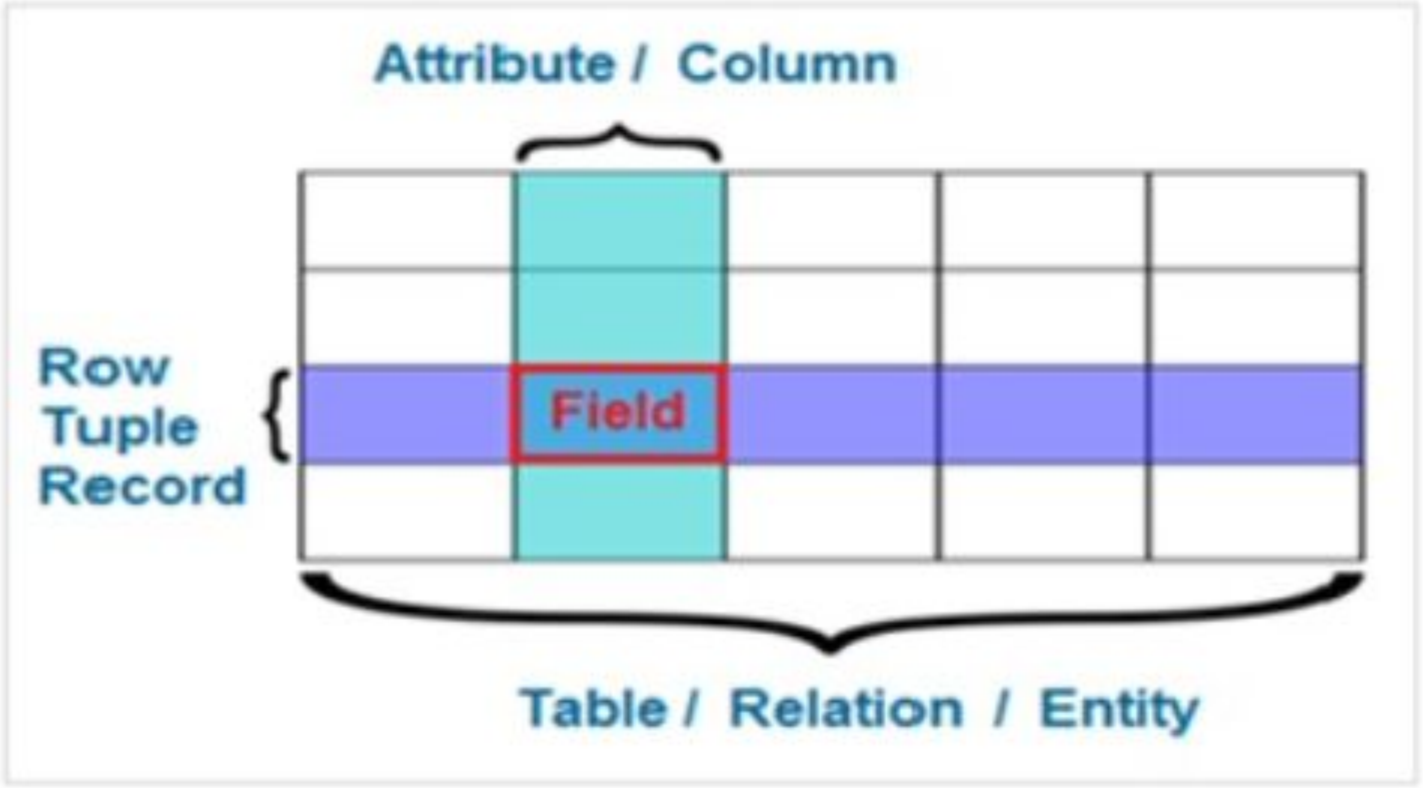


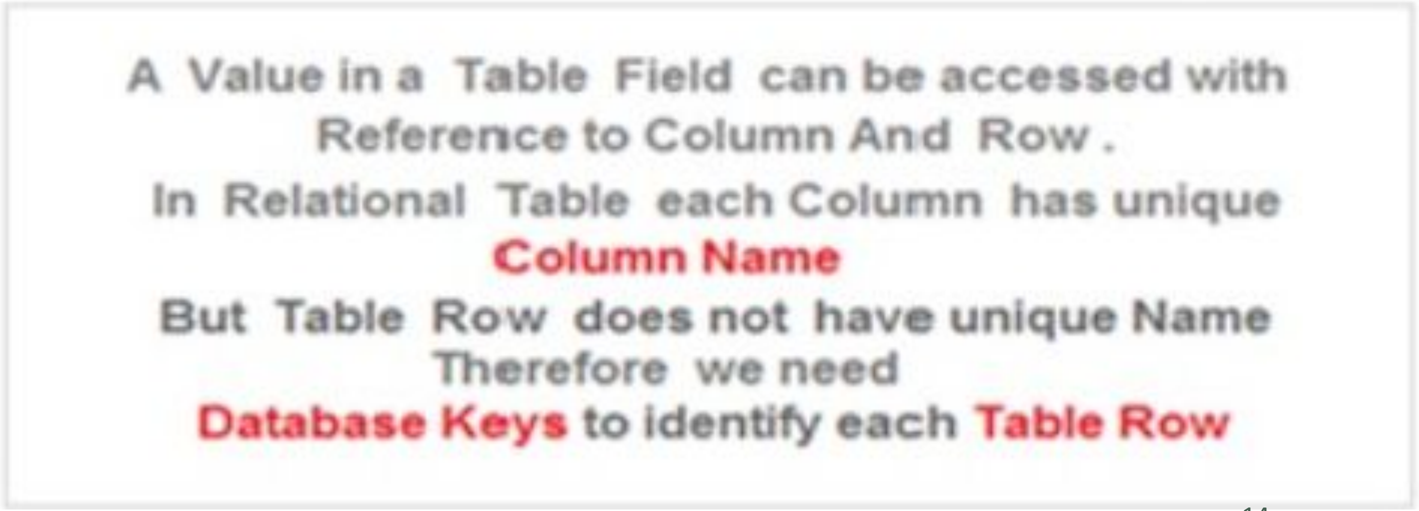
Relation between Primary Key, Candidate Key, and Super Key



Database Keys

Database Keys





A Value in a Table Field can be accessed with Reference to Column And Row .
In Relational Table each Column has unique **Column Name**
But Table Row does not have unique Name Therefore we need
Database Keys to identify each **Table Row**

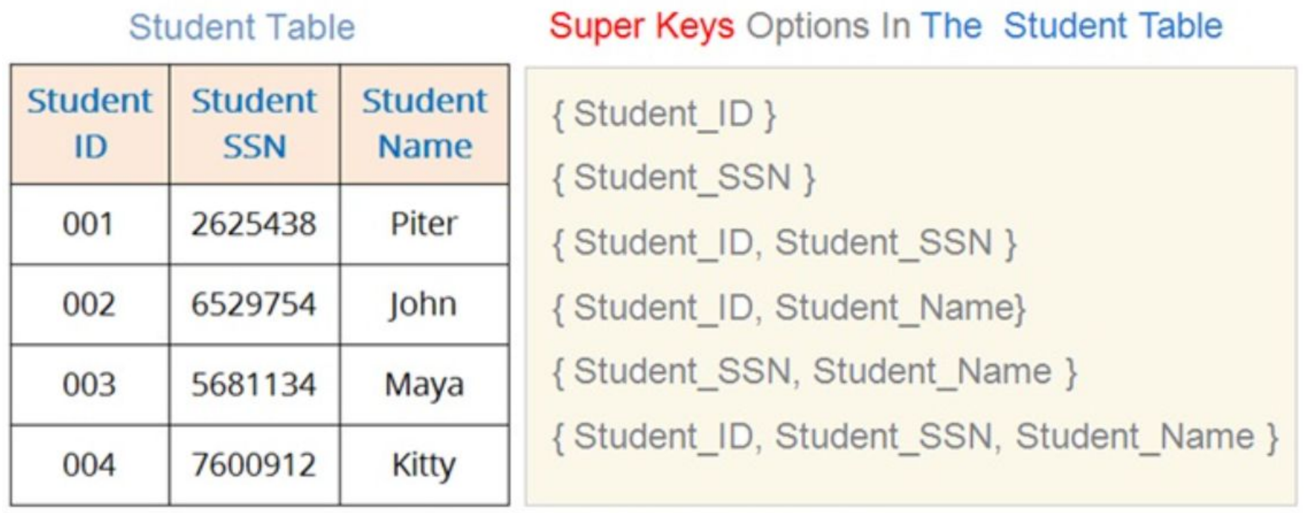
Database Keys

Super key

Super key is an attribute set that can uniquely identify a tuple.

A super key is a superset of a candidate key.

A super key doesn't necessarily have to be minimal. It can contain extra attributes that are not required for unique identification.



Student Table

Student ID	Student SSN	Student Name
001	2625438	Piter
002	6529754	John
003	5681134	Maya
004	7600912	Kitty

Super Keys Options In The Student Table

{ Student_ID }
{ Student_SSN }
{ Student_ID, Student_SSN }
{ Student_ID, Student_Name}
{ Student_SSN, Student_Name }
{ Student_ID, Student_SSN, Student_Name }

Database Keys

Super key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.

- Superset
- Uniquely identify the tuple.
- Can have null values.
- For example(Name) attribute cannot be super key.
- For example: (ID,Name)

Database Keys

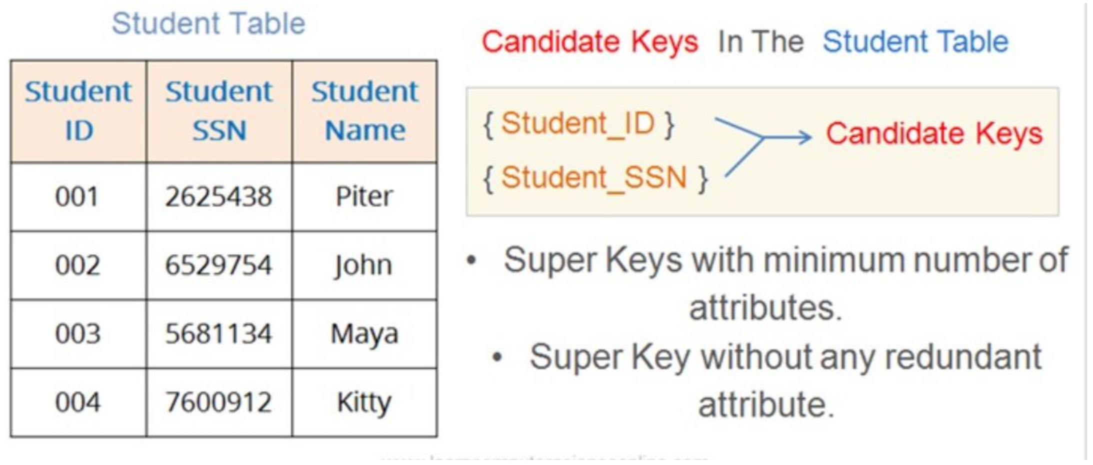
Candidate key

A candidate key is a minimal super key.

A unique identifier for a record in a table that could potentially be used as a primary key.

No proper subset of the attributes in the key can uniquely identify a record.

In other words, if you remove any attribute from a candidate key, it will no longer be a super key.



Student Table

Student ID	Student SSN	Student Name
001	2625438	Piter
002	6529754	John
003	5681134	Maya
004	7600912	Kitty

Candidate Keys In The Student Table

$$\begin{tikzcd} \{ \text{Student\_ID} \} \arrow[dr] & \\ & \text{Candidate Keys} \\ \{ \text{Student\_SSN} \} \arrow[ur] & \end{tikzcd}$$

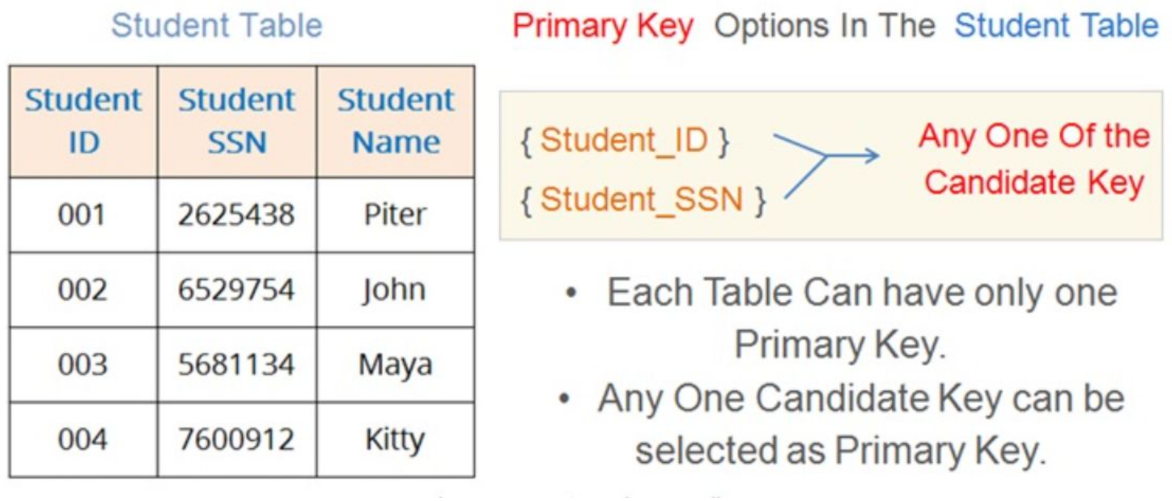
- Super Keys with minimum number of attributes.
- Super Key without any redundant attribute.

Database Keys

Primary key

A unique identifier for each record in a table. It cannot have a null value and must be unique across the entire table.

No duplication


Student Table

Student ID	Student SSN	Student Name
001	2625438	Piter
002	6529754	John
003	5681134	Maya
004	7600912	Kitty

Primary Key Options In The Student Table

$$\begin{tikzcd} \{\text{Student\_ID}\} \arrow[dr] & & \\ & {} \arrow[r] & \text{Any One Of the Candidate Key} \\ \{\text{Student\_SSN}\} \arrow[ur] & & \end{tikzcd}$$

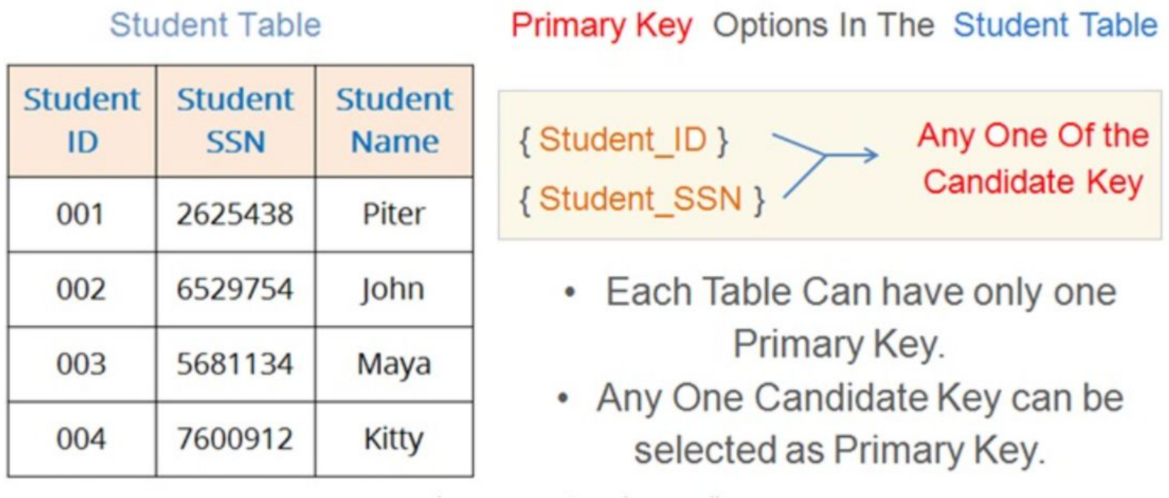
- Each Table Can have only one Primary Key.
- Any One Candidate Key can be selected as Primary Key.

Database Keys

Alternate key / Secondary Key

Candidate column other the Primary column, like if EmployeeID or StudentID is set for a PK then SSN would be the Alternate key.

secondary keys can also be used for uniquely identifying records



Student Table

Student ID	Student SSN	Student Name
001	2625438	Piter
002	6529754	John
003	5681134	Maya
004	7600912	Kitty

Primary Key Options In The Student Table

$$\begin{tikzcd} \{\text{Student\_ID}\} \arrow[dr] & & \\ & {} \arrow[r] & \text{Any One Of the Candidate Key} \\ \{\text{Student\_SSN}\} \arrow[ur] & & \end{tikzcd}$$

- Each Table Can have only one Primary Key.
- Any One Candidate Key can be selected as Primary Key.

Database Keys

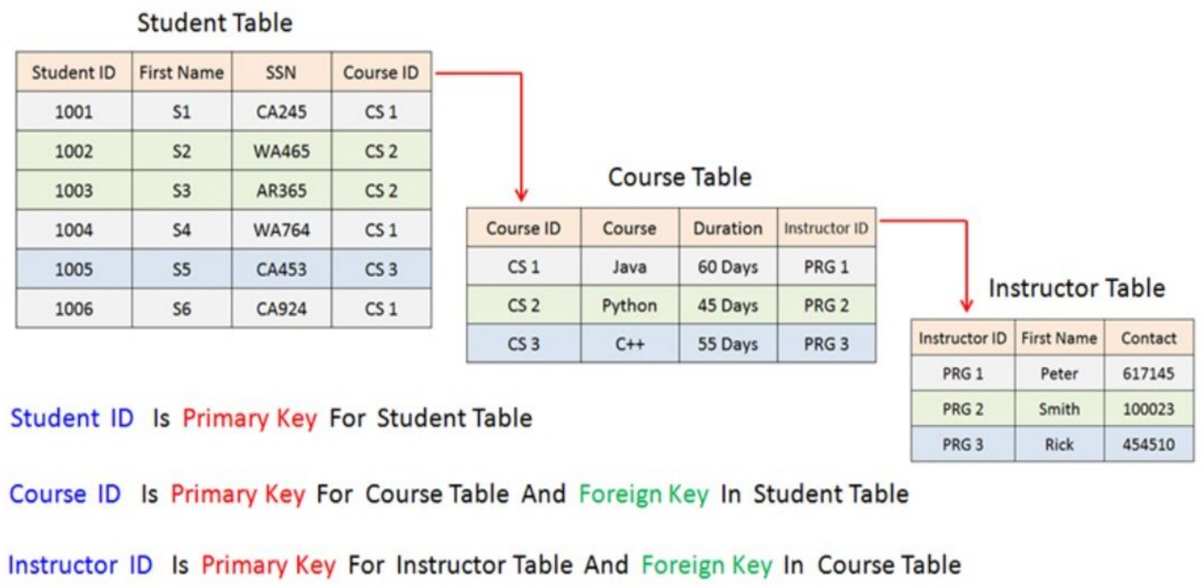
Unique Key:

- A unique key is a constraint ensuring the uniqueness of values within a column or set of columns.
- Similar to a primary key but allows null values, enforcing uniqueness among non-null values.
- **Example:** Using a "Username" column in a "Users" table with a unique constraint, ensuring that each username is distinct.
- (Name,Phone) can be unique key.

Database Keys

Foreign key

A field in a table that matches the primary key of another table. It creates a link between the two tables and is used to establish a relationship between them.



Student Table

Student ID	First Name	SSN	Course ID
1001	S1	CA245	CS 1
1002	S2	WA465	CS 2
1003	S3	AR365	CS 2
1004	S4	WA764	CS 1
1005	S5	CA453	CS 3
1006	S6	CA924	CS 1

Course Table

Course ID	Course	Duration	Instructor ID
CS 1	Java	60 Days	PRG 1
CS 2	Python	45 Days	PRG 2
CS 3	C++	55 Days	PRG 3

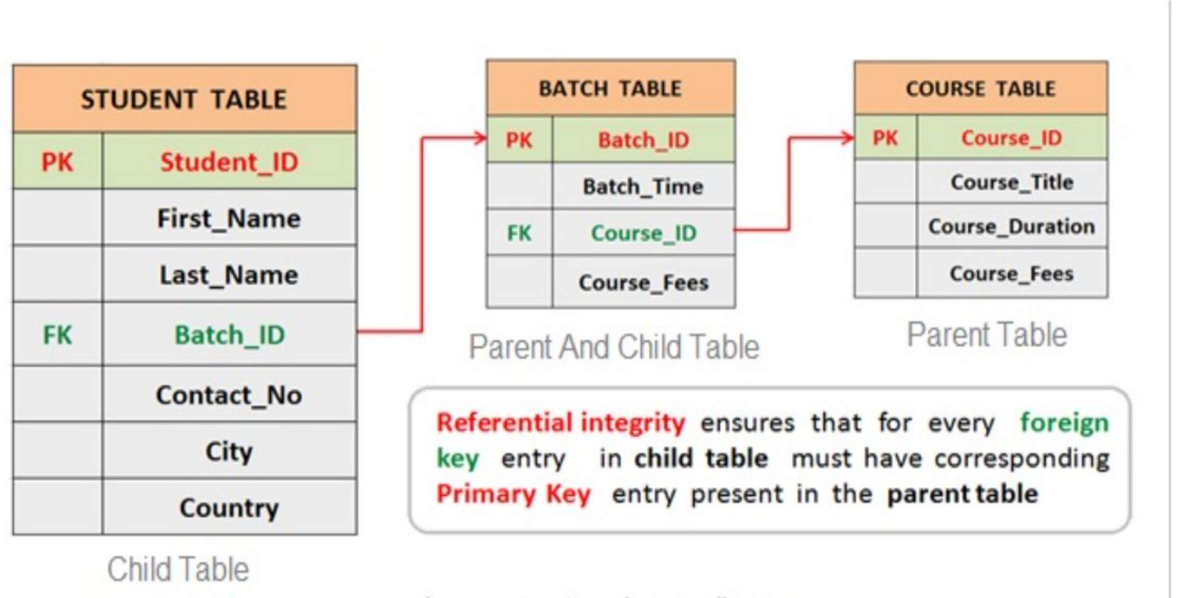
Instructor Table

Instructor ID	First Name	Contact
PRG 1	Peter	617145
PRG 2	Smith	100023
PRG 3	Rick	454510

Student ID Is Primary Key For Student Table

Course ID Is Primary Key For Course Table And Foreign Key In Student Table

Instructor ID Is Primary Key For Instructor Table And Foreign Key In Course Table



STUDENT TABLE

PK	Student_ID
	First_Name
	Last_Name
FK	Batch_ID
	Contact_No
	City
	Country

Child Table

BATCH TABLE

PK	Batch_ID
	Batch_Time
FK	Course_ID
	Course_Fees

Parent And Child Table

COURSE TABLE

PK	Course_ID
	Course_Title
	Course_Duration
	Course_Fees

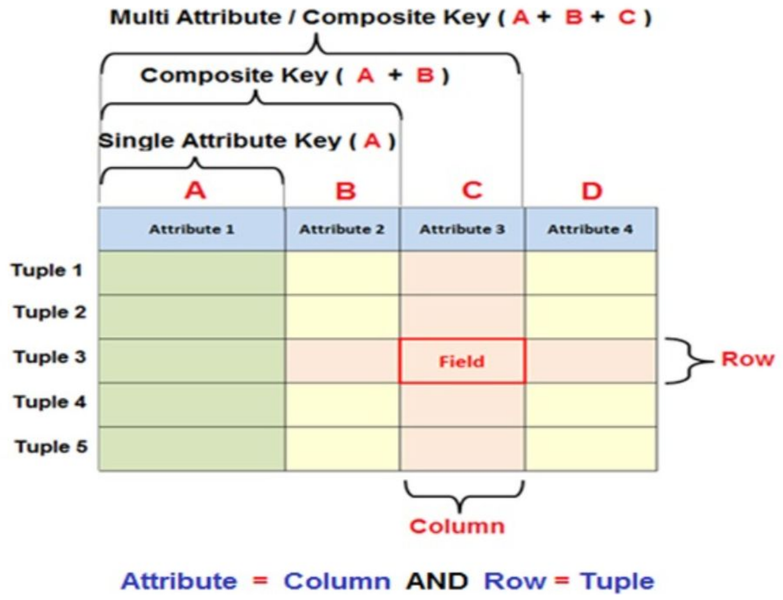
Parent Table

Referential integrity ensures that for every foreign key entry in **child table** must have corresponding Primary Key entry present in the **parent table**

Database Keys

Composite key

A combination of two or more columns used as a primary key. It is used when a single column is not sufficient to uniquely identify a record.



Database Keys

surrogate key

- A surrogate key is an artificially generated unique identifier assigned to each record to serve as its primary key.
- It's not derived from the actual data within the table and doesn't have any inherent business meaning.
- It is system-generated, lacks inherent meaning, provides stable identification.
- **Example:** An auto-incremented integer column in a "Customers" table, serving as a surrogate key to uniquely identify each customer record.

Tasks

Exercise: Identify Keys in a Database Schema

Consider the following database schema for a hypothetical online bookstore:

Book (ISBN, Title, Author, Publisher, Price)

Customer (CustomerID, Name, Email, Phone)

Order (OrderID, CustomerID, ISBN, OrderDate, Quantity)

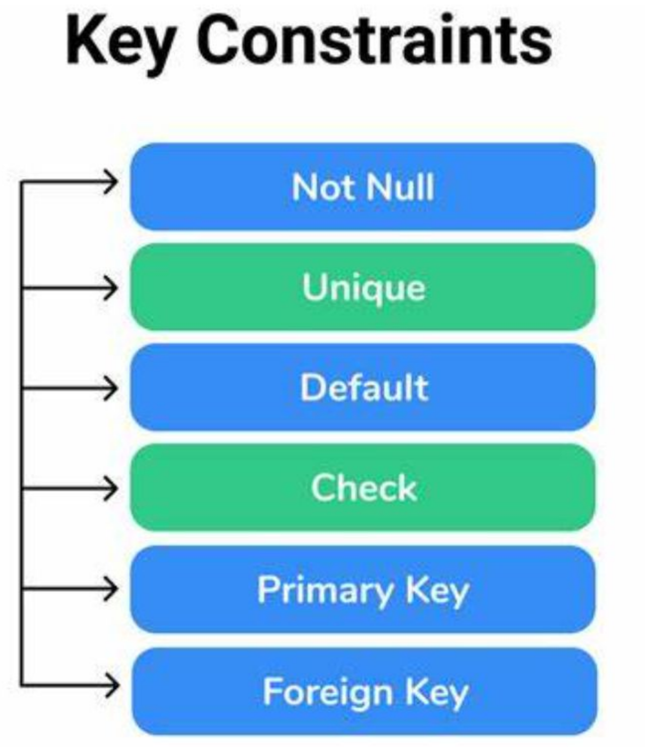
Integrity Constraints

- **Integrity:** In the context of database management systems (DBMS), integrity refers to the accuracy, consistency, and reliability of the data stored within the database.
- **Integrity constraints** are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the *data insertion, updating, and other processes* have to be performed in such a way ***that data integrity is not affected.***

Key Constraint

Key Constraint-

Constraint	Description
NOT NULL	values cannot be null
UNIQUE	values cannot match any older value
PRIMARY KEY	used to uniquely identify a row
FOREIGN KEY	references a row in another table
CHECK	validates condition for new value
DEFAULT	set default value if not passed



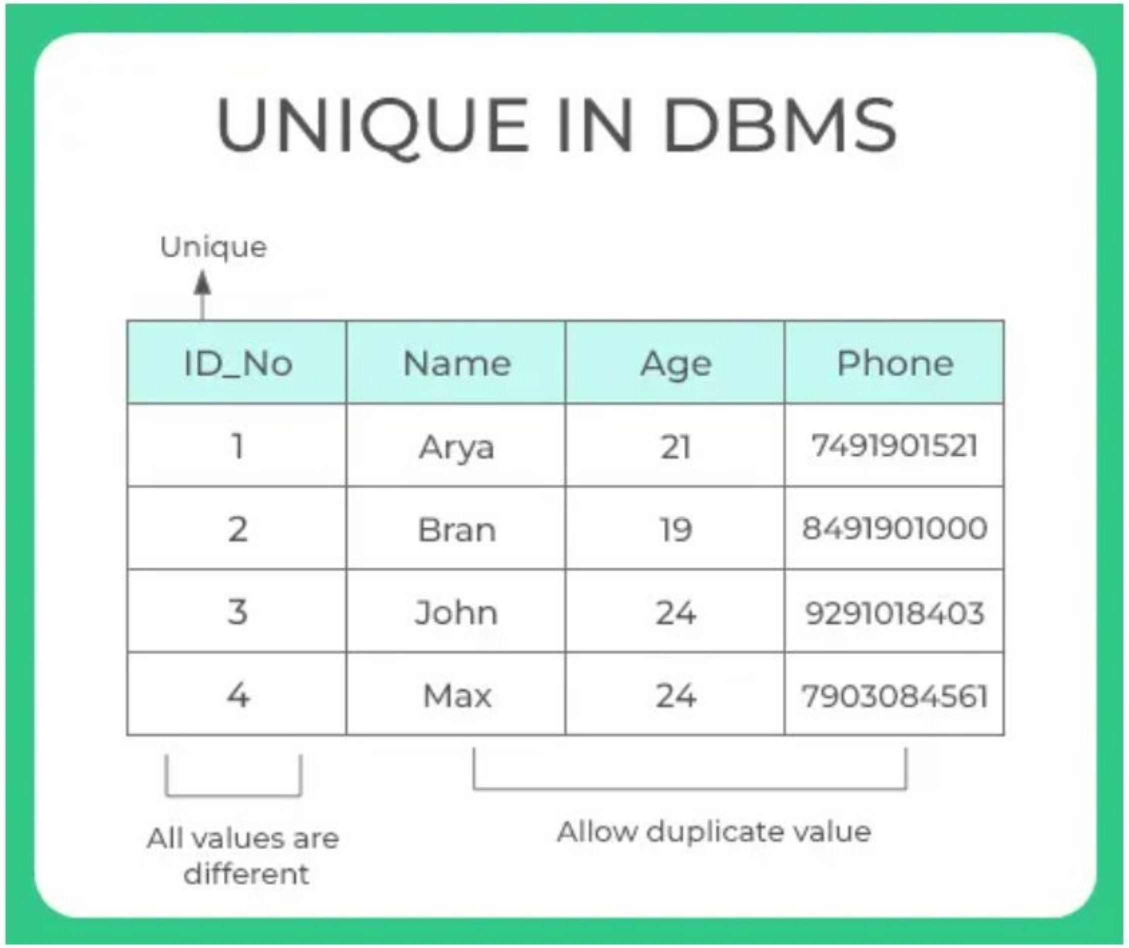
Key Constraint

NOT NULL Constraint

Null represents a record where data may be missing data or data for that record may be optional

Once not null is applied to a particular column, you cannot enter null values to that column and restricted to maintain only some proper value other than null

A not-null constraint cannot be applied at table level


UNIQUE IN DBMS
Unique
All values are different
Allow duplicate value

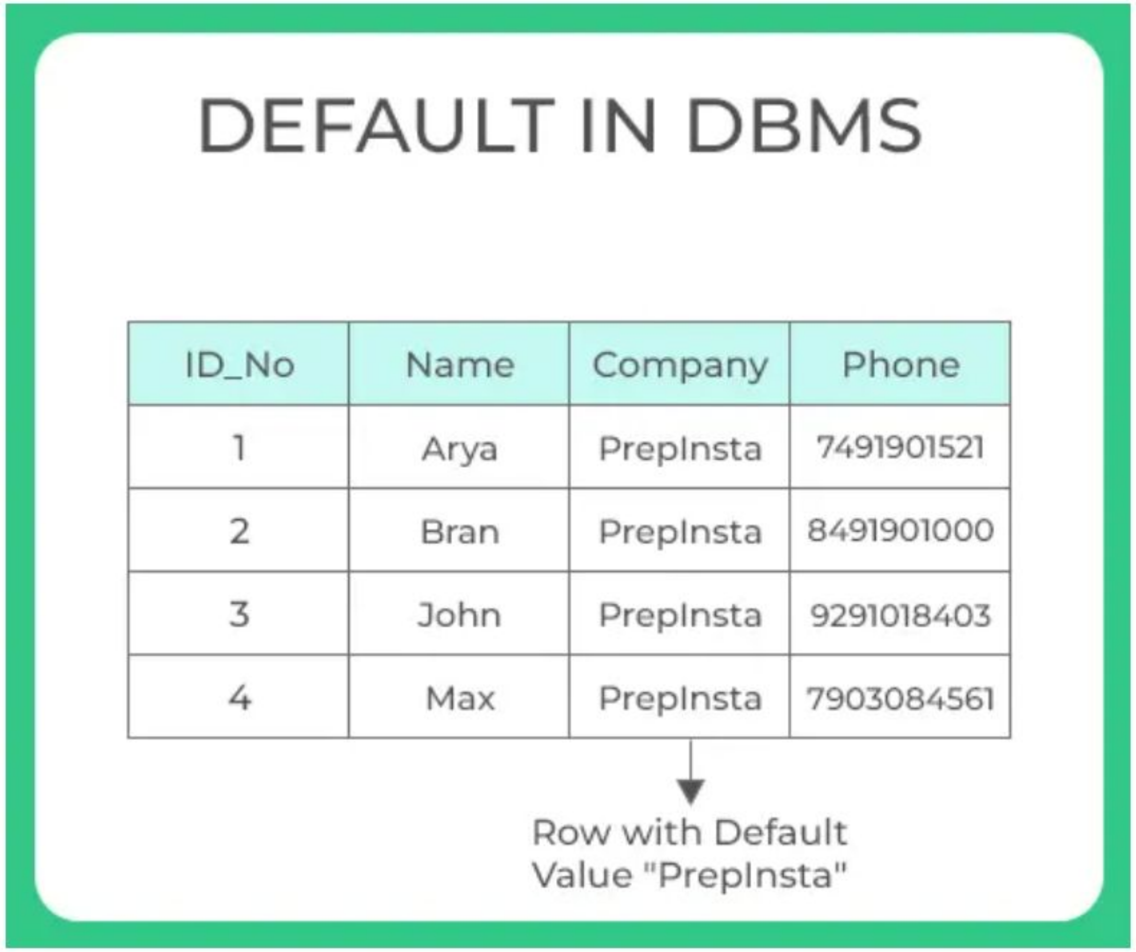
ID_No	Name	Age	Phone
1	Arya	21	7491901521
2	Bran	19	8491901000
3	John	24	9291018403
4	Max	24	7903084561

Key Constraint

Unique Constraint

Sometimes we need to maintain only unique data in the column of a database table, this is possible by using a unique constraint

Unique constraint ensures that all values in a column are unique



DEFAULT IN DBMS

ID_No	Name	Company	Phone
1	Arya	PrepInsta	7491901521
2	Bran	PrepInsta	8491901000
3	John	PrepInsta	9291018403
4	Max	PrepInsta	7903084561

Row with Default Value "PrepInsta"

Key Constraint

Default Constraint

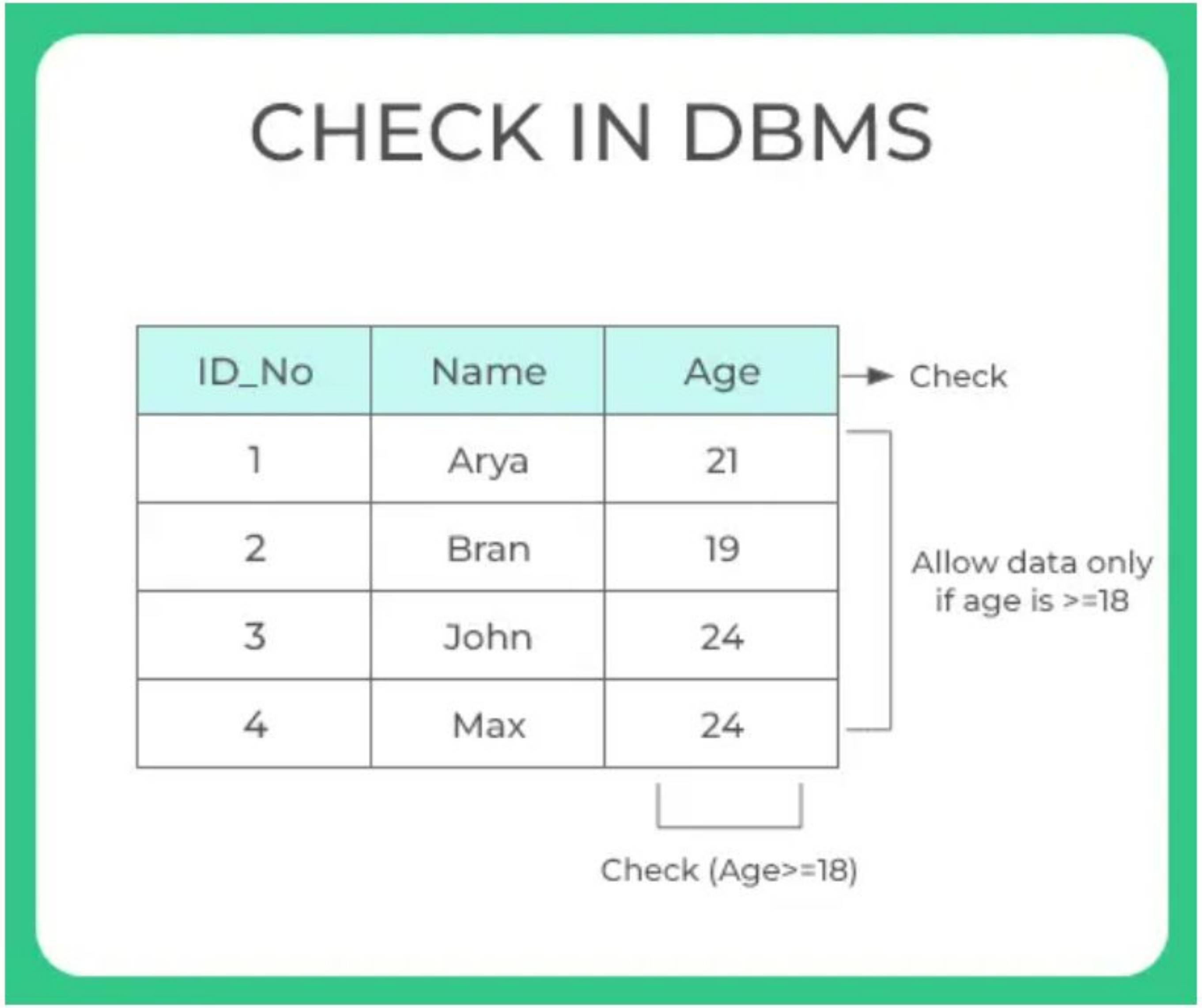
The DEFAULT Constraint is used to fill a column with a default and fixed value. The value will be added to all new records when no other value is provided.

Key Constraint

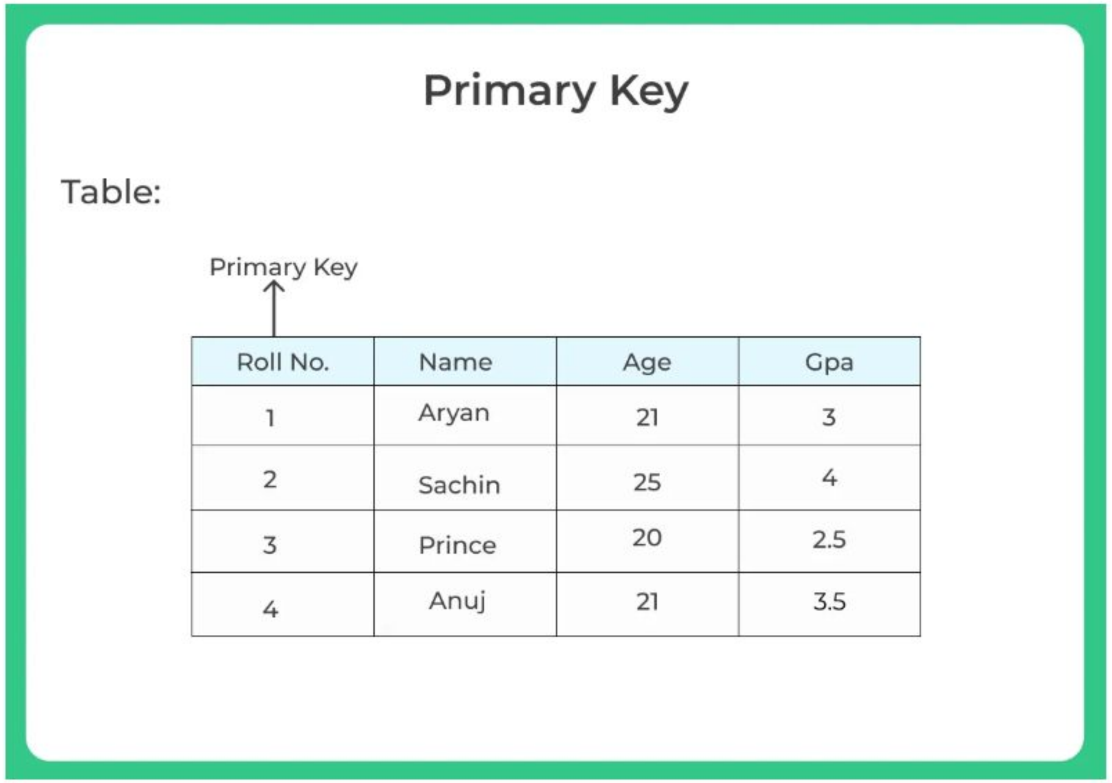
Check Constraint

It limits the values that a column can hold in a relation.

The CHECK constraint is used to limit the value range that can be placed in a column.



ID_No	Name	Age
1	Arya	21
2	Bran	19
3	John	24
4	Max	24



Primary Key

Table:

Primary Key

Roll No.	Name	Age	Gpa
1	Aryan	21	3
2	Sachin	25	4
3	Prince	20	2.5
4	Anuj	21	3.5

Key Constraint

primary key constraint

It depicts a key comprising one or more columns that will help uniquely identify every tuple/record in a table.

Properties :

- No duplicate values are allowed, i.e. Column assigned as primary key should have UNIQUE values only.
- NO NULL values are present in column with Primary key. Hence there is Mandatory value in column having Primary key.
- Only one primary key per table exist although Primary key may have multiple columns.
- No new row can be inserted with the already existing primary key.

Key Constraint

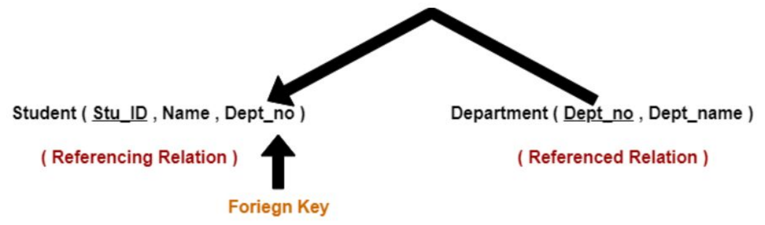
Foreign Key Constraint

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

The main purpose of the foreign key is only those values are allowed in the present table that will match the primary key column of another table.


Student (Stu_ID , Name , Dept_no)
(Referencing Relation)
Foriegn Key
Department (Dept_no , Dept_name)
(Referenced Relation)

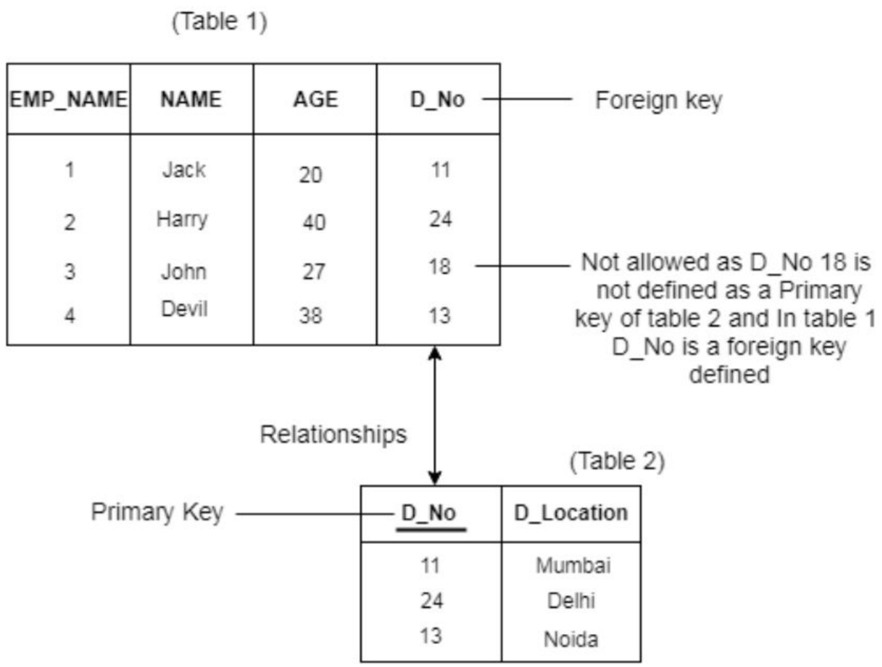
Referential integrity Constraint

A **referential integrity constraint** is specified between two tables.

In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be **null or be available** in Table 2.

Following are two important points:

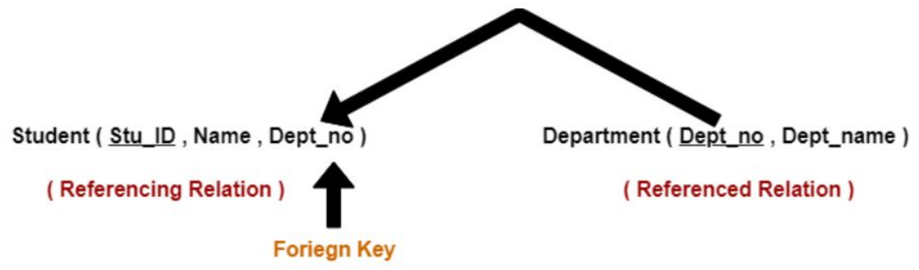
- We can not insert a record into a referencing/child relation if the corresponding record does not exist in the referenced/parent relation.
- We can not delete the record of referenced/parent relation if the corresponding record exists in the referencing/child relation.

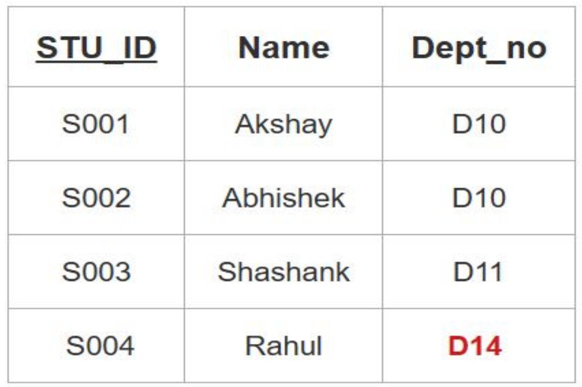


(Table 1)
Foreign key
Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1 D_No is a foreign key defined
Relationships
(Table 2)
Primary Key

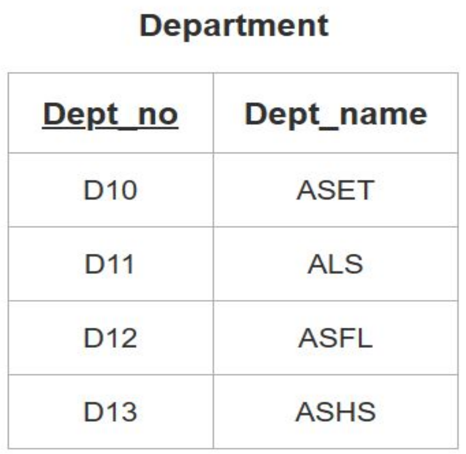
EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida


Student (Stu_ID , Name , Dept_no)
(Referencing Relation)
Foriegn Key
Department (Dept_no , Dept_name)
(Referenced Relation)



STU_ID	Name	Dept_no
S001	Akshay	D10
S002	Abhishek	D10
S003	Shashank	D11
S004	Rahul	D14



Department

Dept_no	Dept_name
D10	ASET
D11	ALS
D12	ASFL
D13	ASHS

Referential integrity Constraint

- The relation ‘Student’ does not satisfy the referential integrity constraint.
- This is because in relation ‘Department’, no value of primary key specifies department no. 14.
- **Thus, referential integrity constraint is violated.**

Referential integrity Constraint Violation

Referential Integrity Constraint Violation-

There are following three possible causes of violation of referential integrity constraint-

Cause-01: Insertion in a referencing relation

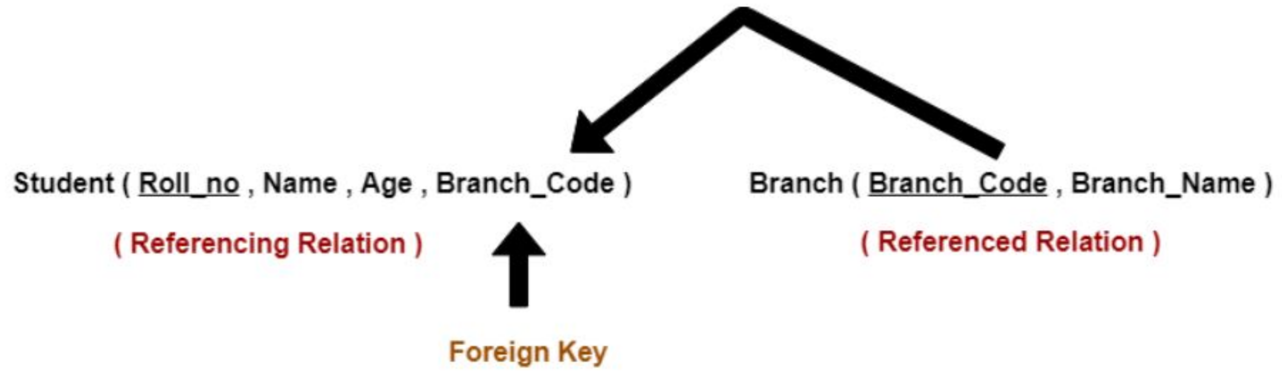
Cause-02: Deletion from a referenced relation

Cause-03: Updation in a referenced relation

Referential integrity Constraint Violation

Referential Integrity Constraint Violation-

Consider the following two schemas-


Student (Roll_no , Name , Age , Branch_Code)
(Referencing Relation)
Foreign Key
Branch (Branch_Code , Branch_Name)
(Referenced Relation)

Branch

Branch_Code	Branch_Name
CS	Computer Science
EE	Electronics Engineering
IT	Information Technology
CE	Civil Engineering

Student

Roll_no	Name	Age	Branch_Code
1	Rahul	22	CS
2	Anjali	21	CS
3	Teena	20	IT

Branch

Branch_Code	Branch_Name
CS	Computer Science
EE	Electronics Engineering
IT	Information Technology
CE	Civil Engineering

Student

Roll_no	Name	Age	Branch_Code
1	Rahul	22	CS
2	Anjali	21	CS
3	Teena	20	IT

Referential integrity Constraint Violation

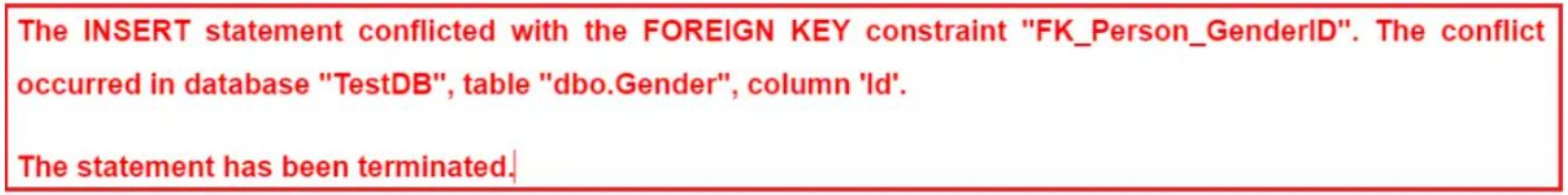
Cause-01: Insertion in a referencing relation

Here,

In relation “Student”, **we can not insert any student having branch code ME (Mechanical Engineering).**

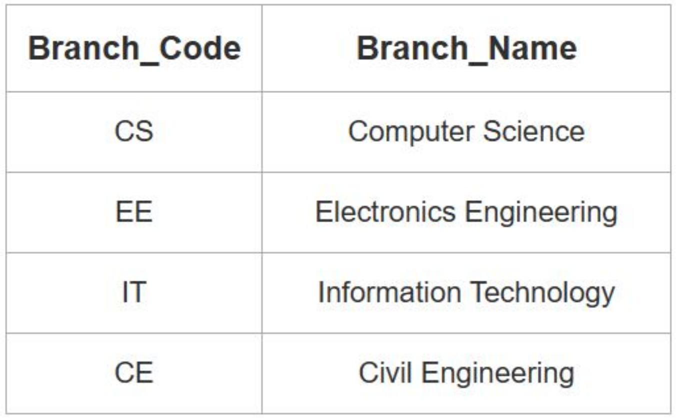
This is because branch code ME is not present in the relation “Branch”.

Example: Error MSG



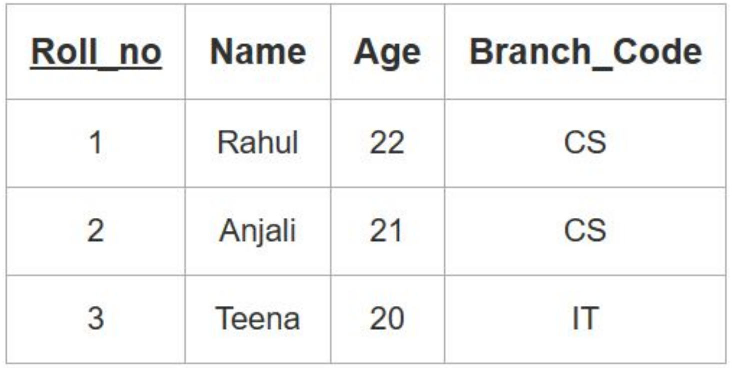
```
The INSERT statement conflicted with the FOREIGN KEY constraint "FK_Person_GenderID". The conflict
occurred in database "TestDB", table "dbo.Gender", column 'Id'.

The statement has been terminated.
```

Branch



Branch_Code	Branch_Name
CS	Computer Science
EE	Electronics Engineering
IT	Information Technology
CE	Civil Engineering

Student



Roll_no	Name	Age	Branch_Code
1	Rahul	22	CS
2	Anjali	21	CS
3	Teena	20	IT

Referential integrity Constraint Violation

Referential Integrity Constraint Violation-

Cause-02: Deletion from a referenced relation

- Consider the given two relations,
- **We can not delete a tuple from the relation “Branch” having branch code ‘CS’.**
- This is because the referencing attribute “Branch_Code” of the referencing relation “Student” references the value ‘CS’.
- **However, we can safely delete a tuple from the relation “Branch” having branch code ‘CE’.**
- This is because the referencing attribute “Branch_Code” of the referencing relation “Student” does not uses this value.

Referential integrity Constraint Violation

Referential Integrity Constraint Violation-

Cause-02: Deletion from a referenced relation

The violation caused due to a deletion from the referenced relation can be handled in the following three ways-

Method-01:

ON DELETE CASCADE constraint is used in MySQL to delete the rows from the child table automatically, when the rows from the parent table are deleted.

This method of handling the violation is called as **On Delete Cascade.**

Referential integrity Constraint Violation

Referential Integrity Constraint Violation-

Cause-02: Deletion from a referenced relation

Method-02:

This method involves **aborting or deleting the request for a deletion** from the referenced relation if the value is used by the referencing relation.

Method-03:

This method involves setting the value being deleted from the referenced relation **to NULL**.

Referential integrity Constraint Violation

Referential Integrity Constraint Violation-

Cause-03: Updation in a referenced relation

- It is not allowed to update a row of the referenced relation if the referencing attribute uses the value of the referenced attribute of that row.
- Such an updation violates the referential integrity constraint.
- The same methods will work with updation as explained previously.