KEYs in DBMS

Let's create a simple example using a **StudentCourse** table that includes all the key types: **Primary Key**, **Foreign Key**, **Candidate Key**, **Super Key**, **Composite Key**, **Alternate Key**, and **Artificial Key**.

Table: StudentCourse

StudentID	Email	CourseID	CourseName	EnrollmentDate
1	alice@email.com	C101	Math	2025-01-10
2	bob@email.com	C102	Science	2025-01-11
3	charlie@email.com	C101	Math	2025-01-12
4	diana@email.com	C103	History	2025-01-13

Table: Course

CourseID	CourseName	Instructor	Credits
C101	**Math**	**Dr. Smith**	**3**
C102	**Science**	**Dr. Johnson**	**4**
C103	**History**	**Prof. Brown**	**3**

Explanation of Key Types in the StudentCourse table:

1. **Primary Key**:
    - **Primary Key**: The **StudentID** it uniquely identifies each record in the table (a student can enroll in multiple courses, but this combination ensures uniqueness).
    - **Primary Key**: (StudentID).
2. **Foreign Key**:
    - **Foreign Key**: The **CourseID** is a **foreign key** that refers to a **Course** table (not shown here), where CourseID is the primary key in the **Course** table.
    - **Foreign Key**: CourseID (this links the StudentCourse table to the Course table).
3. **Candidate Key**:

- **Candidate Keys**:
  - **StudentID** — This could uniquely identify a student.
  - **Email** — This could also uniquely identify a student.
- Both **StudentID** and **Email** are candidate keys, as they can each uniquely identify a student, and either could be used as the primary key.

4. **Super Key**:
   - **Super Keys**: A super key is any set of attributes that uniquely identifies a record, and it may contain unnecessary columns.
     - (StudentID, Email) — This combination can uniquely identify the record, but adding Email is redundant because StudentID is sufficient.
     - (StudentID, CourseID, Email) — This is a super key as well, though it includes extra attributes (Email) that are unnecessary for uniqueness.
5. **Composite Key**:
   - **Composite Key**: A **composite key** involves multiple columns used together to uniquely identify a record.
     - (StudentID, CourseID) is a **composite key** because this combination of two columns uniquely identifies each record (each student-course enrollment).
6. **Alternate Key**:
   - **Alternate Key**: An alternate key is a candidate key that was not selected as the primary key. Since we chose (StudentID, CourseID) as the primary key, the following would be the alternate keys:
     - Email — Although Email could uniquely identify a student, it was not selected as the primary key, so it is an alternate key.
7. **Artificial Key**:
   - **Artificial Key**: An **artificial key** is a system-generated key with no business meaning. If we wanted to make the table simpler by adding an EnrollmentID that auto-increments, this would be an **artificial key**.
   - **Artificial Key**: EnrollmentID (auto-incremented integer, generated by the system).

Example of the StudentCourse table with all keys:

EnrollmentID	StudentID	Email	CourseID	CourseName	EnrollmentDate
1	1	alice@email.com	C101	Math	2025-01-10
2	2	bob@email.com	C102	Science	2025-01-11
3	3	charlie@email.com	C101	Math	2025-01-12
4	4	diana@email.com	C103	History	2025-01-13

Key Types in the StudentCourse Table:

Key Type	Key Example(s)	Description
Primary Key	(StudentID, CourseID)	A combination of StudentID and CourseID uniquely identifies each record.
Foreign Key	CourseID (references Course table)	CourseID is a foreign key that references the Course table.
Candidate Key	StudentID, Email	Both StudentID and Email can uniquely identify a student.
Super Key	(StudentID, Email), (StudentID, CourseID, Email)	Any set of attributes that uniquely identifies a record (e.g., adding Email to StudentID).
Composite Key	(StudentID, CourseID)	The combination of StudentID and CourseID is used to uniquely identify each record.
Alternate Key	Email	Email is a candidate key but not chosen as the primary key.
Artificial Key	EnrollmentID (auto-incremented)	An artificial key created for simplicity, uniquely identifying each enrollment.

Summary of Key Types in the Table:

1. **Primary Key**: The combination of StudentID and CourseID uniquely identifies each record.
2. **Foreign Key**: CourseID is a foreign key that links to another table (e.g., Course).
3. **Candidate Key**: StudentID and Email are candidate keys because they can uniquely identify a record.

4. **Super Key**: StudentID + Email or StudentID + CourseID + Email are examples of super keys (they can uniquely identify a record but may contain unnecessary columns).
5. **Composite Key**: StudentID + CourseID is a composite key, as it involves multiple columns to uniquely identify a record.
6. **Alternate Key**: Email is an alternate key because it is a candidate key that was not selected as the primary key.
7. **Artificial Key**: EnrollmentID is an artificial key because it is auto-generated and has no real-world significance.

This example demonstrates how different types of keys can be used in a relational database table for ensuring uniqueness, establishing relationships, and simplifying data management.

REFRENTIAL INTEGRITY CONSTRAINTS EX :

Insert Constraint:

The Insert Constraint ensures that a foreign key value in the child table must exist in the master table (the table containing the primary key). In other words, you cannot insert a record in the child table if the referenced value (foreign key) does not exist in the master table.

- Meaning: When inserting a record into the child table, the foreign key must reference an existing primary key in the master table. If the value does not exist in the master table, the insert operation is blocked.
- **Example:**

Suppose we have the following tables:

Department Table (**Master Table**):

DeptID	DeptName
D01	HR
D02	IT
D03	Finance

Employee Table (**Child Table**):

EmpID	EmpName	DeptID
E01	Alice	D01
E02	Bob	D02

Now, suppose you try to insert a new record into the **Employee** table:

INSERT INTO Employee (EmpID, EmpName, DeptID)

VALUES (E03, 'Charlie', 'D04');

Since **D04** (the department ID for Charlie) does not exist in the **Department** table (Master Table), the database will **reject** the insert and display an error like:

ERROR: Foreign key constraint violation. 'D04' does not exist in the 'Department' table.

2. Delete Constraint:

Meaning: When deleting a record from the master table, the database ensures that no record in the child table references the record you are trying to delete. If there are dependent records in the child table, the deletion is blocked.

Now, suppose you try to delete a department from the **Department** table:

```
DELETE FROM Department

WHERE DeptID = 'D01';
```

Since **D01** is referenced by **Alice** in the **Employee** table, the delete operation will **fail** and display an error like:

ERROR: Cannot delete 'D01' from the 'Department' table because it is being referenced in the 'Employee' table.