---

## ✅ RDBMS Constraints & Keys

---

### 🟩 What are Constraints?

Constraints are **rules applied to columns** to **restrict data** and maintain **data integrity** in the database.

- They enforce **valid, consistent, and accurate** data entry.

---

### 🔶 Types of Constraints

| Constraint | Purpose |
|---|---|
| PRIMARY KEY | Uniquely identifies each row |
| FOREIGN KEY | Establishes relationship between tables |
| UNIQUE | Ensures all values are unique |
| NOT NULL | Ensures a column must have a value |
| CHECK | Ensures values meet specific conditions |
| DEFAULT (clause) | Sets default value when none provided |

---

### ✅ PRIMARY KEY

- **What:**
  - A column (or combination) that **uniquely identifies a record** in a table.
- **Rules:**
  - Cannot be NULL
  - Cannot have duplicate values
  - Only **one PRIMARY KEY** per table
  - **Unique index** is auto-created
  - Cannot be of **TEXT or BLOB**
- **Syntax:**

```
CREATE TABLE student (
 roll_no INT PRIMARY KEY,
 name VARCHAR(50)
);
```

🔸 **Composite Primary Key:**

- Use when **one column isn't enough**

```
CREATE TABLE marks (
 student_id INT,
 subject_code CHAR(5),
 marks INT,
 PRIMARY KEY(student_id, subject_code)
);
```

---

✅ **COMPOSITE PRIMARY KEY**

- Combines multiple columns to form a **unique identifier**.
- Useful in **transaction or junction tables**.

```
PRIMARY KEY(col1, col2)
```

---

✅ **CANDIDATE KEY**

🔸 **What:**

- Any column(s) that **can potentially become a primary key**.

Primary Key is **chosen** from Candidate Keys

Example:

In Employee Table:

- EmpID (unique)

- Email (unique)

- PAN No (unique)

→ All are candidate keys

---

✅ **ALTERNATE KEY**

🔸 **What:**

- A **Candidate Key** that is **not selected** as the Primary Key.
- Usually enforced using **UNIQUE + NOT NULL**

---

✅ **SUPER KEY**

🔸 **What:**

- **Any combination of columns** that uniquely identifies a row.

All candidate keys are super keys, but **not all super keys are candidate keys**.

---

✅ **FOREIGN KEY**

🔸 **What:**

- A column in one table that **references the primary key** in another table.
- Establishes **referential integrity**.

🔸 **Rules:**

- Can allow NULL
- Can allow duplicates
- Parent column must be **PK or UNIQUE**
- Supports **ON DELETE CASCADE**, **ON UPDATE CASCADE**

🔸 **Example:**

```
CREATE TABLE department (
 dept_id INT PRIMARY KEY,
 dept_name VARCHAR(50)
);

CREATE TABLE employee (
```

```
 emp_id INT PRIMARY KEY,
 emp_name VARCHAR(50),
 dept_id INT,
 FOREIGN KEY (dept_id) REFERENCES department(dept_id)
);
```

---

🔸 **Self-Referencing FK**

FOREIGN KEY (mgr_id) REFERENCES employee(emp_id)

---

🔸 **Composite Foreign Key**

FOREIGN KEY (branch_code, order_no) REFERENCES order_mst(branch_code, order_no)

---

✅ **NOT NULL Constraint**

🔸 **What:**

- Column **must contain a value**, i.e., **no NULL allowed**.

🔸 **Example:**

ename VARCHAR(50) NOT NULL

---

✅ **UNIQUE Constraint**

🔸 **What:**

- Ensures all values in a column (or set) are **distinct**.

🔸 **Rules:**

- Allows **NULL values** (1 or more in MySQL)
- Creates **UNIQUE INDEX**
- You can have **multiple UNIQUE keys**

🔸 **Example:**

CREATE TABLE emp (

```
 emp_id INT,
 email VARCHAR(100) UNIQUE
);
```

Composite Unique:

UNIQUE (deptno, emp_id)

Drop Unique:

DROP INDEX email ON emp;

---

✅ **CHECK Constraint**

🔸 **What:**

- Enforces validation rules on column data.

🔸 **Example:**

sal FLOAT CHECK (sal > 5000 AND sal < 100000)

status CHAR(1) CHECK (status IN ('A','I','P'))

CHECK (sal + comm < 200000)

---

✅ **DEFAULT (Not a Constraint, but Clause)**

🔸 **What:**

- Automatically assigns a value when no value is given in INSERT.

🔸 **Example:**

status CHAR(1) DEFAULT 'T'

sal FLOAT DEFAULT 7000

---

🔶 **Constraint Levels**

| Type | Description |
| --- | --- |
| Column-level | Defined at column creation time |
| Table-level | Defined after columns are declared |

**Note:**

- NOT NULL → Always column-level
- Composite PK, FK, UNIQUE → Must be table-level

---

✅ **ALTER Constraints (Modify After Table is Created)**

🔸 **Add Primary Key**

```
ALTER TABLE emp ADD PRIMARY KEY(emp_id);
```

🔸 **Drop Primary Key**

```
ALTER TABLE emp DROP PRIMARY KEY;
```

🔸 **Add Foreign Key**

```
ALTER TABLE emp ADD FOREIGN KEY(dept_id) REFERENCES department(dept_id);
```

🔸 **Drop Foreign Key**

```
ALTER TABLE emp DROP FOREIGN KEY fk_emp_deptno;
```

🔸 **Add UNIQUE**

```
ALTER TABLE emp ADD CONSTRAINT uq_email UNIQUE (email);
```

---

🔶 **Difference Between Constraints**

| Constraint | Null Allowed | Duplicate Allowed | Purpose |
| --- | --- | --- | --- |
| PRIMARY KEY | ❌ No | ❌ No | Unique identification |
| UNIQUE | ✅ Yes | ❌ No | Alternate unique values |
| FOREIGN KEY | ✅ Yes | ✅ Yes | Reference other table |
| NOT NULL | ❌ No | ✅ Yes | Mandatory value |

| Constraint | Null Allowed | Duplicate Allowed | Purpose |
|---|---|---|---|
| CHECK | ✅ Yes | ✅ Yes | Enforce data rules |
| DEFAULT | ✅ Yes | ✅ Yes | Provide auto value |

---

### 📌 Quick Summary for Interview / Revision

| Key Type | Description |
|---|---|
| **Primary Key** | Unique + Not Null; only one per table |
| **Composite Key** | Primary key with multiple columns |
| **Candidate Key** | All potential primary keys |
| **Alternate Key** | Candidate key not selected as PK |
| **Foreign Key** | Links to PK in another table |
| **Super Key** | Superset of primary key |
| **Unique Key** | Unique values but can allow NULLs |
| **Not Null** | Ensures column must have value |
| **Check** | Validates data using condition |
| **Default** | Fills value when not specified in insert |

| Key Type | Description |
| --- | --- |

## ✅ What is a Surrogate Key?

**A Surrogate Key is a system-generated, artificial, and unique identifier used as the primary key in a table when no natural key is available or suitable.**

---

### 🔹 Why Surrogate Key?

**Sometimes:**

- **No column exists that can uniquely identify each row**
- **All natural keys are either:**
  - **Too long**
  - **Changeable over time (like email, phone number)**
  - **Or not guaranteed to be unique**

**👉 In such cases, we add an extra column (often id) to serve as a unique identifier → This is called a Surrogate Key**

---

### 🔹 Characteristics of Surrogate Key:

| Property | Description |
| --- | --- |
| Artificial/Generated | Not based on business data |
| No business meaning | Purely technical, doesn't convey info |
| Auto-incremented | Often uses AUTO_INCREMENT in MySQL, IDENTITY in SQL Server |
| Unique & Non-null | Always unique; cannot be null |
| Stable | Once assigned, doesn't change |
| Usually numeric | Easy indexing and fast access |

| Key Type | Description |
| --- | --- |

---

🔹 **Example (MySQL):**

```
sql
CopyEdit
CREATE TABLE employee (
 emp_id INT AUTO_INCREMENT PRIMARY KEY,
 name VARCHAR(50),
 email VARCHAR(50)
);
```

- emp_id is a surrogate key.
- It has no business meaning, but is used to uniquely identify each row.

---

🔹 **Surrogate Key vs Natural Key**

| Feature | Surrogate Key | Natural Key |
| --- | --- | --- |
| Meaningful? | ❌ No | ✅ Yes (based on real data) |
| Performance | ✅ Fast indexing | ❌ Slower with large text keys |
| Stability | ✅ Never changes | ❌ May change (like phone/email) |
| Simplicity | ✅ Simple integer | ❌ Often composite or long |

---

🔹 **Interview Tip:**

**Q: Why do we use a Surrogate Key instead of a Natural Key?**

✅ **Answer:**
We use surrogate keys because:

| Key Type | Description |
| --- | --- |

- **They are simpler, more efficient, and immune to data changes**
- **Avoid dependency on business data which might change**
- **Help maintain data integrity and performance**

---