Normalization, Constraints and DDL, DML, DCL and TCL

PANA ACADEMY

Normal forms

Normal Forms in DBMS

- **1NF:** Each table cell has a single value; each column has a unique name.
- **2NF:** Each non-key attribute is fully dependent on the primary key. No partial dependency should be present.
- **3NF:** Non-key attributes are independent of each other. No transitive dependency should be present.
- **BCNF:** Every determinant is a candidate key.
- **4NF:** No multivalued dependencies in the table.
- **5NF:** Decompose tables to eliminate redundancy and ensure data integrity.

Objective of Normalization

Normalization is the process of minimizing **redundancy** from a relation or set of relations.

Eliminate Redundancy:

- Reduce data duplication by organizing data into related tables.
- Helps to save storage space and prevents inconsistencies by ensuring that each piece of data is stored only once.

Ensure Data Integrity:

- Maintain accuracy and consistency of data across the database.
- By eliminating anomalies (insertion, update, and deletion anomalies), normalization ensures that the data remains reliable and accurate.

Simplify Queries:

- Make the database structure clear and efficient.
- Simplifies the database design, making it easier to query, update, and maintain. This enhances the performance and usability of the database.

Functional Dependencies

A functional dependency, denoted as A→B means that if two tuples (rows) have the same value for attribute A, they must also have the same value for attribute B.

A is also called determinant.

B is also called dependent.

Functional dependencies are directional. A→B, does not imply B→A.

Types of Functional Dependencies:

1. **Trivial Functional Dependency:**
   - If B is a subset of A, then $A \rightarrow B$ is a trivial functional dependency. For example, $\{A, B\} \rightarrow A$.
2. **Non-Trivial Functional Dependency:**
   - If B is not a subset of A, then $A \rightarrow B$ is a non-trivial functional dependency. For example, $A \rightarrow B$.
3. **Full Functional Dependency:**
   - A functional dependency $A \rightarrow B$ is a full functional dependency if removal of any attribute from A means the dependency no longer holds. For example, if $\{A, B\} \rightarrow C$ holds but $A \rightarrow C$ does not hold, then $\{A, B\} \rightarrow C$ is a full functional dependency.
4. **Partial Functional Dependency:**
   - A functional dependency $A \rightarrow B$ is partial if some attribute can be removed from A and the dependency still holds. For example, if $\{A, B\} \rightarrow C$ holds and $A \rightarrow C$ also holds, then $\{A, B\} \rightarrow C$ is a partial dependency.
5. **Transitive Functional Dependency:**
   - If $A \rightarrow B$ and $B \rightarrow C$ hold, then $A \rightarrow C$ is a transitive dependency. For example, if $A \rightarrow B$ and $B \rightarrow C$ hold, then $A \downarrow C$ must also hold.

Integrity Constraints and Domain Constraints

Integrity constraints are rules that help to maintain the accuracy and consistency of data in a database.

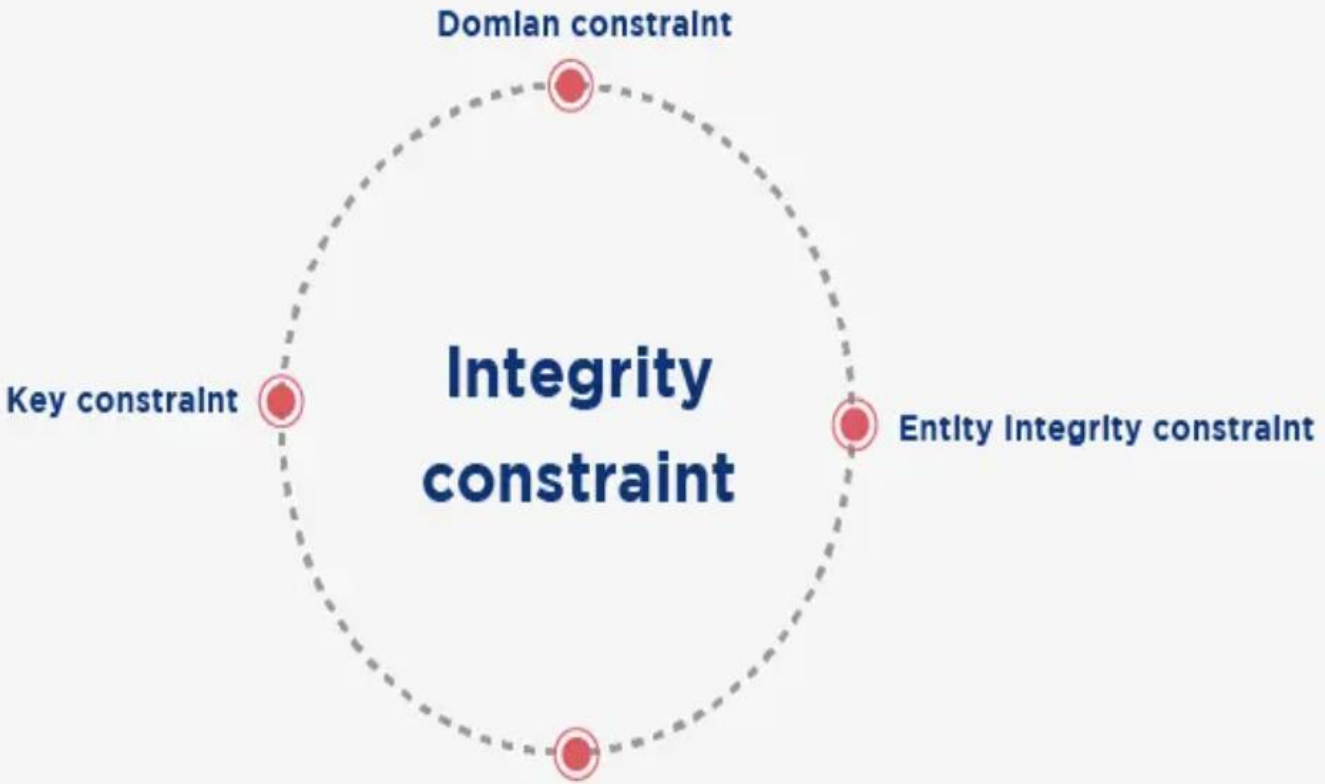
They can be used to enforce business rules or to ensure that data is entered correctly.

For example, a simple integrity constraint in DBMS might state that all customers must have a valid email address.

1. Domain Constraint

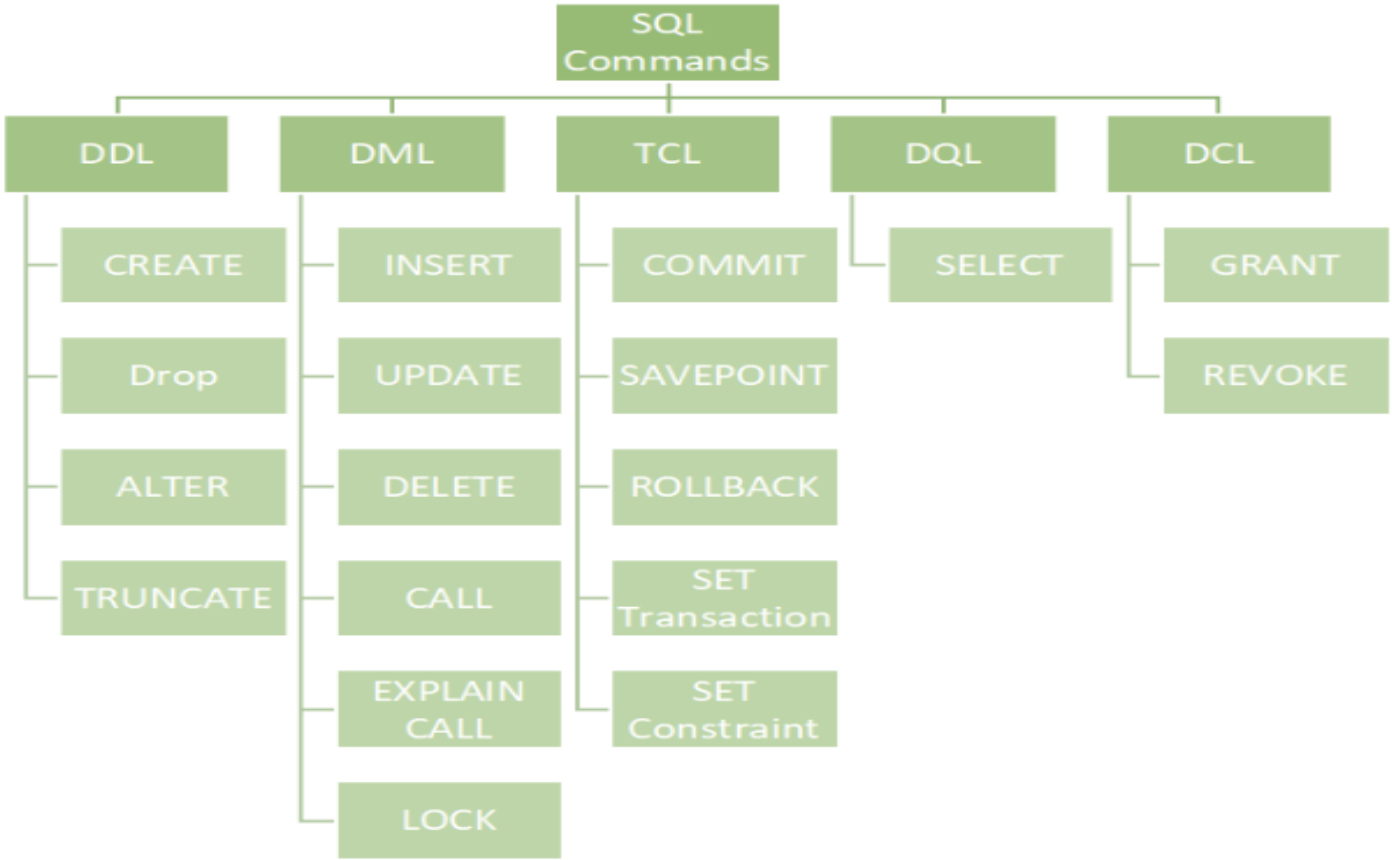
A domain constraint is a restriction on the values that can be stored in a column. For example, if you have a column for "age," domain integrity constraints in DBMS would ensure that only values between 1 and 120 can be entered into that column. This ensures that only valid data is entered into the database.

Types of Integrity Constraints in DBMS



SQL commands are categorized into 5 category.

1. **DDL** – Data Definition Language
2. **DQL** – Data Query Language
3. **DML** – Data Manipulation Language
4. **DCL** – Data Control Language
5. **TCL** – Transaction Control Language


SQL Commands
DDL
DML
TCL
DQL
DCL
CREATE
Drop
ALTER
TRUNCATE
INSERT
UPDATE
DELETE
CALL
EXPLAIN CALL
LOCK
COMMIT
SAVEPOINT
ROLLBACK
SET Transaction
SET Constraint
SELECT
GRANT
REVOKE

Assertions,Triggers and Views

Assertions: Enforce complex business rules at the database level.

Triggers: Automate actions in response to specific events.

Views: Provide simplified, secure, and customized data access.

Assertions are constraints that specify a condition that must be true for the database at all times. They are used to enforce rules at the database level.

- **Definition:** An assertion is a Boolean condition that the DBMS must ensure holds true.
- **Purpose:** Ensure data integrity and consistency.
- **Example:** Ensuring the total number of employees in a department does not exceed a certain limit.

```
CREATE ASSERTION check_employee_count

CHECK (SELECT COUNT(*) FROM Employees WHERE Department_ID = 'D001') <= 100;
```

Triggers

Triggers are procedures that are automatically executed in response to certain events on a particular table or view in a database.

CREATE TRIGGER update_salary_log AFTER UPDATE ON Employees FOR EACH ROW BEGIN INSERT INTO SalaryLog (EmployeeID, OldSalary, NewSalary, ChangeDate) VALUES (:OLD.EmployeeID, :OLD.Salary, :NEW.Salary, SYSDATE); END;

Syntax of trigger:

CREATE TRIGGER trigger_name { BEFORE | AFTER } { INSERT | UPDATE | DELETE } ON table_name FOR EACH ROW [WHEN (condition)] BEGIN -- SQL statements END;

Views

Views are virtual tables that provide a way to present data in a different format or from multiple tables. They do not store data themselves but fetch data from the underlying tables.

Syntax of creating View:

CREATE VIEW EmployeeDepartmentView AS SELECT Employees.EmployeeID, Employees.EmployeeName, Departments.DepartmentName FROM Employees JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;

Joined and Derived Relations

Joined Relations: Combine rows from two or more tables based on a related column, with various types like inner, outer (left, right, full), and cross joins.

Derived Relations (Views): Virtual tables created from SQL queries on base tables, simplifying complex queries, enhancing security, and providing data abstraction.

https://www.sanfoundry.com/oracle-database-mcqs-normalization/

https://www.sanfoundry.com/oracle-sql-mcqs-ddl-command/