databases

A comprehensive cheat sheet covering database concepts, ER modeling, table creation, data population, and querying using SQL. From understanding databases to crafting advanced queries, this guide provides a quick reference for database design and manipulation.

Database Fundamentals & ER Modeling

What is a Database?

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. Databases are used to store, retrieve, modify, and delete data.

Databases enable efficient data management, ensuring data integrity, consistency, and security. They are crucial for applications ranging from simple contact lists to complex inventory management systems.

ER Model Elements

Entity: A real-world object or concept (e.g., Customer, Product, Order). Represented as a rectangle.

Attribute: A characteristic of an entity (e.g., Customer Name, Product Price, Order Date). Represented as an oval.

Relationship: An association between entities (e.g., a Customer places an Order). Represented as a diamond.

Cardinality: Defines the numerical relationship between entities (e.g., one-to-many, many-to-many).

Cardinalities

One-to-One (1:1): Each entity in one set is associated with at most one entity in the other set. Example: One person has one passport.

One-to-Many (1:N): One entity in one set is associated with multiple entities in the other set. Example: One customer can place many orders.

Many-to-Many (M:N): Multiple entities in one set are associated with multiple entities in the other set. Example: Many students can enroll in many courses.

ER Diagrams to Tables & Normalization

ER to Table Conversion

Entities become tables. Attributes become columns.

Primary keys are chosen to uniquely identify each row in a table. Foreign keys are used to represent relationships between tables.

For 1:1 relationships, the primary key of one table can be added as a foreign key to the other table.

For 1:N relationships, the primary key of the 'one' side is added as a foreign key to the 'many' side.

For M:N relationships, a new table (junction table) is created with foreign keys referencing the primary keys of both tables.

Normalization

Normalization is the process of organizing data to reduce redundancy and improve data integrity. It involves dividing databases into tables and defining relationships between the tables.

Common normalization forms include 1NF, 2NF, and 3NF. Each form addresses specific types of redundancy.

By ensuring each table represents a single entity and minimizing redundant data, normalization helps prevent update anomalies and ensures data consistency.

Creating Tables in SQL

CREATE TABLE Syntax

```
CREATE TABLE table_name (
    column1 datatype constraints,
    column2 datatype constraints,
    ...
    PRIMARY KEY (column1)
);
```

Common Data Types

INT: Integer numbers.

VARCHAR(size): Variable-length string (max size).

DATE: Date values (YYYY-MM-DD).

BOOLEAN: True/False values.

DECIMAL(precision, scale): Exact number with precision digits, scale digits after the decimal point.

TIMESTAMP: Date and time values.

Constraints

PRIMARY KEY: Unique identifier for the table.

NOT NULL: Column cannot contain NULL values.

FOREIGN KEY: Establishes a relationship with another table.

UNIQUE: Ensures all values in a column are unique.

CHECK: Specifies a condition that must be true for values in a column.

DEFAULT: Sets a default value for a column if none is specified.

Data Manipulation & Querying

INSERT Statements

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

```
INSERT INTO table_name VALUES (value1, value2, ...); -- All columns
```

SELECT Statements

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

```
SELECT * FROM table_name; -- Select all columns
```

WHERE Clause

Comparison Operators: `=`, `!=`, `>`, `<`, `>=`, `<=`

Logical Operators: `AND`, `OR`, `NOT`

BETWEEN: `column BETWEEN value1 AND value2`

LIKE: `column LIKE pattern` (e.g., `LIKE 'A%'`)

IN: `column IN (value1, value2, ...)`

IS NULL / IS NOT NULL: Check for NULL values.

Joins

Cartesian Product: Combines each row of the first table with each row of the second table.

```
SELECT * FROM table1, table2;
```

NATURAL JOIN: Joins tables based on columns with the same name and data type.

```
SELECT * FROM table1 NATURAL JOIN
table2;
```

INNER JOIN: Returns rows only when there is a match in both tables.

```
SELECT * FROM table1 INNER JOIN table2
ON table1.column = table2.column;
```