DATABASE BASICS

Beginner-friendly notes covering SQL, relational databases, NoSQL, design and practical concepts

Concept	Simple meaning
Database	An organized system for storing, managing and retrieving data.
DBMS	Software that manages databases, users, queries, security and transactions.
SQL	Language commonly used to work with relational databases.
Table	A relational structure made of rows and columns.
Primary Key	A column or set of columns that uniquely identifies a row.
Foreign Key	A column that references a key in another table.
Index	A data structure that can make lookups faster, at the cost of storage and write overhead.

1. What is a Database?

Definition: A database is an organized collection of data that applications can store, update, search and retrieve.

Why databases are needed: Applications need persistent data for users, products, orders, payments, messages, logs and many other business operations.

Database vs DBMS: The database is the stored data; a DBMS is the software used to manage that data.

2. Types of Databases

Relational databases: Store structured data in tables and commonly use SQL. Examples include PostgreSQL, MySQL, MariaDB, Oracle Database and Microsoft SQL Server.

NoSQL databases: Designed for data models other than traditional relational tables. Common categories include document, key-value, wide-column and graph databases.

When to choose relational: Good when relationships, consistency, transactions and structured schemas are important.

When to choose NoSQL: Useful for particular high-scale, flexible-schema or specialized access patterns.

3. Relational Database Fundamentals

Table: Stores records of one logical entity, such as users or products.

Row: One record in a table.

Column: One attribute of a record, such as email or price.

Schema: Defines the structure of database objects, including tables, columns, relationships and constraints.

Constraint: A rule enforced by the database, such as NOT NULL, UNIQUE, PRIMARY KEY or FOREIGN KEY.

4. Keys

Primary Key: Uniquely identifies each row. It should be stable and non-duplicated.

Foreign Key: Connects rows between related tables and helps maintain referential integrity.

Composite Key: A key made from multiple columns.

Candidate Key: A column or set of columns that could uniquely identify a row.

Surrogate Key: An artificial identifier, often an integer or UUID, created primarily to identify a record.

5. Relationships

One-to-one: One row in table A corresponds to at most one row in table B.

One-to-many: One row in A can relate to many rows in B. Example: one customer can have many orders.

Many-to-many: Rows in both tables can relate to many rows in the other table. Usually implemented with a junction/link table.

6. Normalization

Purpose: Normalization reduces unnecessary duplication and helps keep data consistent.

1NF: Values should be atomic and repeating groups should be separated.

2NF: Be in 1NF and remove partial dependency on part of a composite key.

3NF: Be in 2NF and remove inappropriate transitive dependencies.

Practical note: Highly normalized schemas improve consistency, while selective denormalization can improve performance for specific workloads.

7. SQL Basics

SELECT: Reads data.

INSERT: Adds new rows.

UPDATE: Changes existing rows.

DELETE: Removes rows.

WHERE: Filters rows.

ORDER BY: Sorts results.

GROUP BY: Groups rows for aggregate calculations.

JOIN: Combines related rows from multiple tables.

8. Essential SQL Examples

```
Read rows
SELECT id, name, email FROM users;

Filter
SELECT * FROM users WHERE active = true;

Insert
INSERT INTO users (name, email) VALUES ('Ravi', 'ravi@example.com');

Update
UPDATE users SET active = false WHERE id = 10;

Delete
DELETE FROM users WHERE id = 10;

Aggregate
SELECT status, COUNT(*) FROM orders GROUP BY status;

Join
SELECT u.name, o.id FROM users u JOIN orders o ON o.user_id = u.id;
```

9. JOIN Types

INNER JOIN: Returns matching rows from both sides.

LEFT JOIN: Returns all rows from the left table and matching rows from the right table.

RIGHT JOIN: Returns all rows from the right table and matching rows from the left table.

FULL OUTER JOIN: Returns matching rows plus unmatched rows from both sides where supported.

Self JOIN: A table is joined to itself, useful for hierarchical relationships such as employee-manager data.

10. Transactions & ACID

Transaction: A logical unit of work that should be completed as a consistent operation.

Atomicity: All operations in a transaction succeed or the transaction is rolled back.

Consistency: A successful transaction leaves the database in a valid state according to its rules.

Isolation: Concurrent transactions are controlled so their intermediate states do not improperly interfere.

Durability: Committed changes survive failures according to the database's durability guarantees.

11. Indexes

What is an index?: An auxiliary structure that helps the database find rows without scanning every row.

Good candidates: Columns frequently used in WHERE, JOIN, ORDER BY or certain grouping operations.

Trade-off: Indexes consume storage and can slow INSERT, UPDATE and DELETE operations because indexes may also need updating.

Avoid over-indexing: Create indexes based on actual query patterns and execution plans.

12. Views, Stored Procedures & Triggers

View: A saved query presented like a virtual table.

Stored procedure/function: Database-side programmable logic that can encapsulate operations or calculations.

Trigger: Logic that automatically executes when certain database events occur. Use carefully because hidden side effects can make systems harder to reason about.

13. NoSQL Basics

Document database: Stores JSON-like documents. Useful when application data naturally maps to documents.

Key-value database: Stores values using unique keys. Often useful for caching and simple high-speed lookups.

Wide-column database: Uses flexible column-oriented structures for large-scale workloads.

Graph database: Stores nodes and relationships and is useful for highly connected data.

14. Database Security

Authentication: Verifies who or what is connecting.

Authorization: Determines what that identity is allowed to do.

Least privilege: Give applications and users only the permissions they actually need.

Encryption: Use encryption in transit and at rest where appropriate.

SQL injection: Never build SQL by directly concatenating untrusted user input. Use parameterized queries/prepared statements.

15. Backup & Recovery

Backup: A copy of data used to recover from accidental deletion, corruption or infrastructure failure.

RPO: Recovery Point Objective: how much recent data loss is acceptable.

RTO: Recovery Time Objective: how quickly the service should be restored.

Important: A backup is not proven until restoration has been tested.

16. Database Performance Basics

Measure first: Use query execution plans, database metrics and application telemetry rather than guessing.

Common problems: Missing indexes, inefficient joins, unnecessary columns, N+1 queries, excessive network round trips and poor data modeling.

Connection pooling: Reuses database connections instead of creating a new connection for every request.

Caching: Can reduce repeated database work for suitable read-heavy workloads.

17. Example Database Design

For a simple e-commerce application:

```
users (id, name, email)
products (id, name, price)
orders (id, user_id, order_date, status)
order_items (id, order_id, product_id, quantity, price)
```

Relationships: users 1→many orders; orders 1→many order_items; products 1→many order_items. The order_items table acts as the link between orders and products.

18. Database vs SQL vs DBMS

Term	Meaning	Example
Database	Stored collection of data	Application data
DBMS	Software that manages the data	PostgreSQL
SQL	Language used to query relational databases	SELECT ... FROM ...
Table	Structured data container inside a relational database	users
Query	Instruction sent to the database	SELECT * FROM users

19. Beginner Learning Path

• Understand tables, rows, columns and keys

• Learn SELECT, WHERE, ORDER BY and LIMIT

• Practice INSERT, UPDATE and DELETE

• Master JOINs and aggregate functions

• Learn normalization and relationships

• Understand indexes and query execution plans

• Learn transactions and ACID

• Practice PostgreSQL or MySQL

• Learn database security, backups and migrations

• Build a small application using a real database

20. Quick Interview Questions

Q: What is a primary key?
A: A key that uniquely identifies a row.

Q: What is a foreign key?
A: A column or set of columns that references a key in another table.

Q: DELETE vs TRUNCATE?
A: DELETE removes rows and can use filtering; TRUNCATE removes all rows from a table and has different transactional/identity behavior depending on the DBMS.

Q: WHERE vs HAVING?
A: WHERE filters rows before grouping; HAVING filters groups after aggregation.

Q: What is normalization?
A: A method of structuring relational data to reduce unnecessary duplication and anomalies.

Q: What is an index?
A: A structure used to speed up suitable data access patterns.

Q: What is a transaction?
A: A logical unit of database work governed by transactional guarantees.

Q: SQL vs NoSQL?
A: SQL databases generally use relational tables and SQL; NoSQL databases use other models suited to particular access and scaling patterns.

Study note: exact SQL syntax, transaction behavior, indexing features and limits vary between database engines. Check the documentation for the specific database you use.