Week 5: Relational Database Design, ORM, and Migrations

Session 1: Theory

1. Relational Database Design Principles

1.1 What is Relational Database Design?

- A **relational database** stores data in structured tables with rows and columns.
- Uses **primary keys and foreign keys** to establish relationships between tables.
- The goal is to **store data efficiently, eliminate redundancy, and ensure data integrity.**

1.2 Database Normalization

Normalization is the **process of organizing database tables to reduce redundancy** and **improve data integrity**.

Real-Life Example of Normalization

Imagine a **library system** where book records are stored. Without normalization, a single table may contain repeated author names and book categories.

- If an author has multiple books, their name is stored multiple times (data redundancy).
- If a category changes, it must be updated in multiple places (consistency issue).

Normalization would:

1. Create a **Books table** (book details, category ID, and author ID).
2. Create an **Authors table** (unique author details).
3. Create a **Categories table** (unique book categories).

Now, updates apply to only **one place** instead of multiple records.

Normalization Forms:

Normal Form	Description
1NF (First Normal Form)	Ensure each column contains atomic (indivisible) values and each row has a unique identifier (primary key).
2NF (Second Normal Form)	Achieve 1NF and remove partial dependencies (fields must be fully dependent on the primary key).
3NF (Third Normal Form)	Achieve 2NF and remove transitive dependencies (non-key attributes must depend only on the primary key).
BCNF (Boyce-Codd Normal Form)	Achieve 3NF and ensure that all determinants are candidate keys.

Advantages of Normalization:

- Reduces data redundancy.
- Ensures data integrity.
- Simplifies database maintenance.

Industry Leader Spotlight: Edgar F. Codd

- Edgar F. Codd, a **computer scientist at IBM**, developed **relational database theory** and **normalization principles** in the 1970s.
- His work led to the creation of **SQL-based relational database systems** like MySQL, PostgreSQL, and Oracle.
- The **ACID properties (Atomicity, Consistency, Isolation, Durability)** in modern databases are a result of Codd's innovations.

1.3 Indexing in Databases

Indexing **improves the speed of data retrieval** in a relational database.

Daily Life Example of Indexing

Think of an **index in a book**: instead of reading every page, you check the **index** to quickly find a topic. Similarly, a database index speeds up data retrieval.

Benefits of Indexing:

- **Faster queries**: Indexing reduces search time.
- **Efficient sorting**: Helps in ORDER BY queries.
- **Speeds up joins**: Enhances relational query performance.

Industry Leader Spotlight: Larry Ellison

- **Larry Ellison**, co-founder of **Oracle Corporation**, revolutionized relational databases by commercializing **SQL-based relational database systems**.
- Oracle's indexing strategies improved database query performance, influencing enterprise data management worldwide.

2. ORM (Object-Relational Mapping)

2.1 What is ORM?

- ORM allows developers to **interact with a database using code instead of SQL queries**.
- Converts **Python (Django ORM) or JavaScript (Sequelize) objects into database records**.

Real-Life Example of ORM

Imagine ordering food online:

- **Without ORM**: You need to write manual instructions (SQL queries) for each step: checking menu items, inserting orders, retrieving past orders.
- **With ORM**: You interact naturally with objects (e.g., Order.create()) without needing to write SQL manually.

Industry Leader Spotlight: David Heinemeier Hansson

- Creator of **Ruby on Rails**, which popularized **ActiveRecord ORM**.
- His ideas influenced Django ORM, Sequelize, and other ORMs used today.

3. Managing Migrations

3.1 What are Migrations?

- Migrations **track and apply changes to a database schema**.
- Allows version control for **schema changes** (adding/removing columns, modifying tables).

Daily Life Example of Migrations

Think of **building a house**:

- **Blueprints** (Migrations) define changes over time.
- You may add a new **room** (column), modify a **window** (data type change), or remove a **door** (column deletion).
- Instead of rebuilding the house, migrations **track incremental changes**.

Benefits of Migrations:

- Keep track of database changes over time.
- Allow rollbacks to previous versions.
- Help in team collaboration, ensuring consistency.

Industry Leader Spotlight: Guido van Rossum

- The creator of **Python** and contributor to Django, Guido van Rossum, helped shape **Django migrations**, making database schema management simpler and automated.

Conclusion

Database normalization reduces redundancy and improves integrity.
Indexing speeds up database queries.
ORMs (Django ORM, Sequelize) simplify database interactions.
Migrations ensure database schema consistency.

Industry Leaders Who Shaped These Principles

- **Edgar F. Codd** (Relational Database & Normalization)
- **Larry Ellison** (Oracle & Indexing Innovations)
- **David Heinemeier Hansson** (ORM Concepts in Rails, Django, and Sequelize)
- **Guido van Rossum** (Python & Django Migrations)

Understanding these concepts **helps in designing efficient, scalable, and maintainable databases**.