Database Design - Explained from Two Standard Textbooks

i. Introduction to Database Design

Based on both Ramakrishnan and Elmasri, database design begins with understanding data through the ER model. Entities (like Student) and relationships (like Enrolls) are used to model the real world. Keys uniquely identify entities. Weak entities depend on others, and aggregation or class hierarchies help model complex relationships. Conceptual design is done using ER diagrams or UML, providing a blueprint before creating the database.

ii. Relational Model

The relational model (central in both books) represents data in tables (relations). SQL is used to create and modify these tables. Each table has constraints such as primary and foreign keys to ensure data integrity. Transforming ER models into relational schemas requires rules to convert entities and relationships into tables. Views are virtual tables that simplify access or restrict data visibility.

iii. Relational Algebra and Calculus

Ramakrishnan emphasizes relational algebra as a way to query data using operations like selection, projection, union, and joins. Division is used for queries like 'students who took all courses'. Elmasri introduces relational calculus (tuple and domain) as a declarative alternative to algebra, focusing on what to retrieve rather than how.

iv. SQL Queries

Both books detail SQL as the language to access and manipulate relational databases. Basic queries retrieve data. Nested queries are used inside other queries. Aggregate functions (SUM, AVG) summarize data. Joins link multiple tables. Advanced features include integrity constraints and triggers which automatically perform actions on data events.

v. Transaction Management

Elmasri and Ramakrishnan both describe transactions as sequences of operations that must follow ACID rules: Atomicity, Consistency, Isolation, Durability. Concurrency control (e.g., locks) ensures correctness when many transactions run at once. Deadlocks occur when transactions wait on each other; they can be prevented using timestamps or careful scheduling.

vi. Schema Refinement and Normal Forms

Schemas should avoid redundancy. Ramakrishnan introduces problems caused by bad design. Functional dependencies express relationships between attributes. Normalization is the process of organizing data into normal forms (1NF to BCNF). Elmasri also discusses how to decompose schemas without losing data (lossless join) or dependencies (dependency preservation).