Database system concepts

by Abraham Silberschatz, Henry F. Korth, S. Sudarshan · Language: EN

Summary by booksummaryai.com — https://booksummaryai.com/country/en/database-system-concepts

Overview

Database System Concepts by Silberschatz, Korth, and Sudarshan is a foundational textbook that provides a comprehensive and in-depth introduction to the principles and practices of database management systems (DBMS). The book systematically covers the entire spectrum of database technology, starting from the fundamental concepts of data modeling and database design, progressing through the intricacies of relational databases, SQL, and query processing, and culminating in advanced topics such as transaction management, concurrency control, recovery systems, and distributed databases. It aims to equip readers with a solid theoretical understanding necessary for designing, implementing, and managing efficient and reliable database systems, while also providing practical insights into real-world applications.

The main argument of the book is that a thorough understanding of database concepts is crucial for anyone involved in information technology, given the central role databases play in modern computing. It meticulously explains how data is structured, stored, retrieved, and protected, emphasizing the trade-offs involved in various design and implementation choices. The book's strength lies in its balanced approach, presenting both the theoretical underpinnings and the practical aspects of database systems, making complex topics accessible through clear explanations, examples, and exercises. It matters because it serves as an indispensable resource for students and professionals alike, providing the knowledge required to navigate the complexities of data management in an increasingly data-driven world, fostering the skills needed to build robust and scalable data solutions.

Key Takeaways

- Mastering the relational model and SQL is fundamental to understanding and interacting with most modern database systems.
- Effective database design, utilizing techniques like ER modeling and normalization, is crucial for data integrity and system performance.
- Understanding transaction management, including ACID properties, is essential for maintaining data consistency and reliability in multi-user environments.
- Concurrency control mechanisms are vital to ensure correct execution of concurrent transactions without data corruption.
- Database recovery techniques are necessary to restore the database to a consistent state after system failures.

• Optimizing query processing is key to achieving high performance in database applications, requiring knowledge of indexing and query plan evaluation.

• Distributed and parallel database systems offer solutions for scalability and availability, but introduce new challenges in data management.

• Security considerations must be integrated into database design and administration to protect sensitive information.

Chapter Summaries

Introduction to Database Systems

This foundational chapter introduces the core concepts and purpose of database systems, highlighting their advantages over traditional file processing systems. It covers fundamental data models, including the relational, network, and hierarchical models, and briefly touches upon object-oriented models. The chapter describes the architecture of a database system, including the three-schema architecture (external, conceptual, internal schemas), and explains data independence. It also outlines the roles of various database users, such as application programmers, sophisticated users, and naive users, as well as the responsibilities of the database administrator (DBA) in managing the system, ensuring security, and optimizing performance.

The Relational Model

This chapter dives into the theoretical underpinnings of the relational data model, which forms the basis for most modern database systems. It defines key concepts such as relations (tables), attributes (columns), tuples (rows), and domains. A significant portion is dedicated to explaining various types of keys—superkey, candidate key, primary key, and foreign key—and their crucial role in maintaining data integrity and establishing relationships between tables. Furthermore, the chapter introduces relational algebra, a procedural query language that provides a formal foundation for SQL, detailing operations like select, project, union, set difference, Cartesian product, and different types of join operations.

Introduction to SQL

This chapter provides a comprehensive introduction to the Structured Query Language (SQL), the standard language for interacting with relational databases. It covers Data Definition Language (DDL) commands for creating, altering, and dropping tables, specifying data types, and defining basic constraints. The core of the chapter focuses on Data Manipulation Language (DML) commands, particularly the SELECT statement for querying data, including filtering with WHERE clauses, sorting with ORDER BY, and using aggregate functions like COUNT, SUM, AVG, MIN, and MAX. It also introduces INSERT, UPDATE, and DELETE statements for modifying data within the database.

Advanced SQL

Building upon the basics, this chapter explores more sophisticated features and capabilities of SQL. It delves into complex query constructs such as nested subqueries, correlated subqueries, and the use of set operations (UNION, INTERSECT, EXCEPT). The concept of views is introduced as a way to simplify complex queries and provide data security. The chapter also covers advanced integrity constraints, including CHECK constraints and assertions, and discusses authorization mechanisms for granting and revoking user privileges. Furthermore, it introduces procedural extensions to SQL, such as stored procedures, functions, and triggers, which allow for encapsulating business logic within the database system.

Database Design using the E-R Model

This chapter introduces the Entity-Relationship (E-R) model as a high-level conceptual data model widely used for database design. It explains how to represent real-world entities, their attributes (simple, composite, multi-valued, derived), and the relationships between them. Key concepts covered include relationship sets, cardinality constraints (one-to-one, one-to-many, many-to-many), and participation constraints (total and partial). The chapter also discusses weak entity sets and generalization/specialization hierarchies. A significant part focuses on the systematic process of designing an E-R diagram and subsequently mapping it into a relational schema, providing a bridge between conceptual design and physical implementation.

Relational Database Design and Normalization

This chapter focuses on the principles of good relational schema design, aiming to minimize data redundancy and prevent update anomalies. It introduces the crucial concept of functional dependencies and provides Armstrong's axioms for inferring new dependencies. The core of the chapter explains various normal forms: First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), and Boyce-Codd Normal Form (BCNF). It details the conditions for each normal form and illustrates the decomposition process to achieve higher normal forms, ensuring that data is stored efficiently and consistently, thereby improving database integrity and maintainability.

Data Storage and File Structures

This chapter delves into the physical aspects of database implementation, explaining how data is stored on various storage media. It covers the storage hierarchy, from volatile memory to non-volatile secondary storage like magnetic disks and flash memory. Detailed discussions include the physical characteristics of disks, disk access times, and the use of RAID (Redundant Arrays of Independent Disks) for performance and reliability. The chapter then explores different file organization techniques, such as heap files, sequential files, and hash files, explaining their advantages and disadvantages in terms of data retrieval and modification efficiency.

Indexing and Hashing

This chapter focuses on techniques used to accelerate data retrieval from disk. It provides an in-depth explanation of various indexing structures, including primary, secondary, and clustering indices. A major emphasis is placed on tree-based indexing, particularly B-tree and B+-tree structures, detailing their construction, search algorithms, and methods for insertion and deletion of records while maintaining balance. The chapter also covers hashing techniques, including static hashing and dynamic hashing (e.g., extendable hashing, linear hashing), discussing how they map search keys to disk blocks for direct access and their performance implications.

Query Processing

This chapter outlines the internal mechanisms by which a database system executes user queries. It describes the various steps involved in query processing, starting from parsing and translation of the query into an internal representation (e.g., relational algebra expression). It then details different algorithms for performing fundamental relational operations, such as selection (linear scan, index scan), projection, and various join algorithms including nested-loop join, block nested-loop join, merge-join, and hash-join. The chapter also covers algorithms for set operations (union, intersection, difference) and aggregation, providing a foundation for understanding query performance.

Query Optimization

Building on query processing, this chapter focuses on the critical task of selecting the most efficient execution plan for a given query. It introduces the concept of query optimization and explains why it's essential for database performance. The chapter covers techniques for estimating the cost of different query plans, considering factors like disk I/O, CPU usage, and available memory. It discusses heuristic optimization strategies and, more importantly, cost-based optimization, which relies on statistical information about the data. The chapter also explores transformations of relational expressions and the role of the system catalog in providing necessary statistics for optimization decisions.

Transactions

This chapter introduces the fundamental concept of a transaction in a database system, defining it as a logical unit of work that accesses and possibly modifies the database. It thoroughly explains the ACID properties—Atomicity, Consistency, Isolation, and Durability—which are crucial for ensuring database reliability and integrity, especially in multi-user environments. The chapter details transaction states (active, partially committed, failed, aborted, committed) and the transitions between them. It also discusses the overall architecture of a transaction management system and the challenges posed by concurrent transactions and system failures, setting the stage for concurrency control and recovery.

Concurrency Control

This chapter addresses the complex problem of managing multiple concurrent transactions to ensure that their execution is correct and that the database remains consistent. It explains the concept of serializability as the correctness criterion for concurrent schedules. Various concurrency control protocols are discussed in detail, including lock-based protocols like two-phase locking (2PL) and its variations (strict 2PL, rigorous 2PL), timestamp-based protocols, and validation-based (optimistic) protocols. The chapter also covers issues such as deadlocks, starvation, and phantom phenomena, along with strategies for their detection and resolution, ensuring isolated execution of transactions.

Recovery System

This chapter focuses on how a database system recovers from various types of failures, including transaction failures, system crashes, and disk failures, to ensure data durability. It introduces the concept of a recovery manager and the use of a write-ahead log (WAL) for recording all database modifications. Detailed explanations are provided for log-based recovery schemes, specifically deferred database modification (NO-UNDO/REDO) and immediate database modification (UNDO/REDO), outlining the undo and redo operations. The chapter also covers checkpoints, which reduce recovery time, and discusses advanced recovery techniques like shadow paging and the ARIES recovery algorithm.

Object-Based Databases

This chapter explores extensions to the traditional relational model to handle more complex data types and relationships, leading to object-based databases. It introduces concepts from object-oriented programming, such as object identity, encapsulation, complex types, and inheritance, and discusses how these are integrated into database systems. The focus is primarily on object-relational databases, which combine relational features with object-oriented capabilities, allowing users to define abstract data types, nested relations, and methods. It illustrates how these extensions enable more natural modeling of complex real-world entities and behaviors, enhancing the expressive power of the database system.

XML and JSON

This chapter addresses semi-structured data models, specifically XML (eXtensible Markup Language) and JSON (JavaScript Object Notation), which are widely used for data exchange and representation on the web. It covers the hierarchical structure of XML documents, schema definition using DTDs (Document Type Definitions) and XML Schema, and querying XML data using XPath and XQuery. Similarly, it introduces JSON syntax, its use in web applications, and methods for parsing and manipulating JSON data. The chapter also discusses strategies for storing and querying XML and JSON data within traditional relational databases or specialized NoSQL document stores.

Data Warehousing and Data Mining

This chapter introduces the concepts and architectures related to data warehousing, which involves collecting and managing data from diverse sources for analytical purposes. It covers Online Analytical Processing (OLAP) operations like roll-up, drill-down, slice, dice, and pivot, and discusses multidimensional data models such as star schema and snowflake schema. The Extract, Transform, Load (ETL) process for populating data warehouses is explained. The chapter also provides an overview of data mining techniques, including association rules, classification, and clustering, illustrating how these methods can be applied to discover valuable patterns and insights from large datasets.

Big Data and NoSQL

This chapter addresses the challenges posed by "Big Data," characterized by its volume, velocity, and variety, and introduces alternative database paradigms. It explores NoSQL databases as a response to the limitations of traditional relational databases for certain large-scale, distributed, and schema-less applications. The chapter categorizes different types of NoSQL systems, including key-value stores, document databases (e.g., MongoDB), column-family stores (e.g., Cassandra), and graph databases. It discusses their respective strengths, weaknesses, and use cases, highlighting concepts like eventual consistency and horizontal scalability that are central to Big Data architectures.

Database Security and Authorization

This chapter focuses on protecting database systems and their data from unauthorized access, modification, or destruction. It covers various aspects of database security, including authentication (verifying user identity) and authorization (specifying what authenticated users can do). Discretionary Access Control (DAC) and Mandatory Access Control (MAC) models are discussed for managing user privileges. The chapter also explores common security threats to databases, such as SQL injection, denial-of-service attacks, and data breaches. It outlines mechanisms for ensuring data privacy, integrity, and availability, including encryption techniques and auditing.

Themes

- Data Modeling
- Query Optimization
- Transaction Management
- Data Integrity
- Concurrency Control
- Database Security

Frequently Asked Questions

What is Database system concepts about?

Database System Concepts is a comprehensive textbook that introduces the fundamental principles and practical aspects of database management systems. It covers data models, SQL, database design, storage, query processing, transaction management, concurrency control, and recovery, providing a holistic view of how databases are designed, implemented, and managed.

Is Database system concepts worth reading?

Yes, it is highly worth reading for anyone serious about understanding database systems. It is widely regarded as a standard reference and textbook in the field, offering a rigorous and detailed explanation of core concepts essential for both academic study and professional practice in database design, development, and administration.

Who should read Database system concepts?

This book is ideal for undergraduate and graduate students in computer science, information technology, and related fields. It is also highly beneficial for software developers, database administrators, and data architects who need a deep understanding of database theory and practical implementation to build and manage robust data-driven applications.

How long does it take to read Database system concepts?

Given its comprehensive nature and typical length of over 1000 pages, a thorough reading of Database System Concepts can take approximately 2000 to 2500 minutes, or about 33 to 42 hours, depending on the reader's pace and prior knowledge. Many readers will spend more time on specific chapters or exercises.

Best Quotes

“A database management system (DBMS) is a collection of interrelated data and a set of programs to access that data.”

Introduction

This foundational definition opens the discussion on what a DBMS is. It establishes the dual nature of a database system: both the structured data itself and the software tools that allow users to interact with, manage, and retrieve that data efficiently and securely. It's crucial for understanding the entire book's scope.

“The entity-relationship (ER) model is a high-level conceptual data model.”

The Entity-Relationship Model

This statement introduces the ER model as a primary tool for conceptual database design. It emphasizes its role in representing real-world entities and their relationships in an abstract,

human-understandable way, before translating it into a specific database schema. It's a cornerstone of logical database design.

“Atomicity requires that either all operations of the transaction are executed, or none are.”

Transactions

This is one of the four ACID properties, fundamental to ensuring data integrity in transactional systems. It means that a transaction is an indivisible unit of work; if any part fails, the entire transaction is rolled back, leaving the database unchanged. This prevents partial updates and maintains consistency.

“Consistency requires that any transaction must take the database from one consistent state to another.”

Transactions

Another ACID property, consistency ensures that a transaction, when executed, preserves the integrity constraints defined on the database. It means that the database always adheres to its rules (e.g., primary key uniqueness, foreign key references) before and after a transaction, assuming the transaction itself is correctly programmed.

“Isolation requires that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially.”

Concurrency Control

This ACID property is vital for multi-user environments. It guarantees that transactions appear to execute independently of each other, even if they are running concurrently. This prevents interference between transactions, ensuring that the intermediate results of one transaction are not visible to another until it commits.

“Durability ensures that once a transaction has committed, its effects remain in the database even in the event of system failures.”

Recovery System

The final ACID property, durability, is critical for data persistence. It means that committed changes are permanently stored and survive power outages, system crashes, or other failures. This is typically achieved through logging and recovery mechanisms, ensuring data integrity and reliability.

“Normalization is a process of organizing the columns and tables of a relational database to minimize data redundancy and improve data integrity.”

Relational Database Design

This definition introduces normalization as a key technique in relational database design. It explains its dual purpose: reducing duplicate data storage, which saves space and prevents update anomalies, and enhancing data integrity by ensuring that data dependencies are logically sound. It's central to good database schema design.

“A relation is a set of tuples.”

The Relational Model

This is a fundamental definition in the relational model. It precisely defines a table in a relational database as a collection of rows (tuples), where the order of rows does not matter. This mathematical precision underpins the entire relational algebra and calculus.

“A primary key is a minimal set of attributes that uniquely identifies each tuple in a relation.”

The Relational Model

This definition is crucial for understanding how individual records are identified within a table. The primary key serves as the unique identifier, ensuring that no two rows are identical and providing a mechanism for referencing specific records, which is essential for data integrity and relationships.

“A foreign key is a set of attributes in a relation that refers to the primary key of another relation.”

The Relational Model

This definition explains how relationships between different tables are established in a relational database. Foreign keys enforce referential integrity, ensuring that references between tables are valid and consistent, which is fundamental to maintaining the structural integrity of the database.

“Indexing is a data structure technique that allows for fast retrieval of records from a database table.”

Indexing and Hashing

This statement introduces indexing as a performance optimization technique. It highlights that indexes are specialized data structures (like B-trees) designed to speed up data access operations, particularly for queries involving search conditions or ordering, by avoiding full table scans. It's critical for efficient query processing.

“Concurrency control is the process of managing simultaneous operations on a database without having them interfere with one another.”

This definition outlines the core challenge and solution addressed by concurrency control mechanisms. It emphasizes the need to allow multiple users or applications to access and modify data concurrently while maintaining data consistency and integrity, preventing issues like lost updates or dirty reads.

Questions & Answers

What is the primary purpose of a Database Management System (DBMS)?

The primary purpose of a DBMS is to provide a systematic way to create, maintain, and access large amounts of data efficiently and securely. It abstracts the complexities of data storage and retrieval, offering features like data independence, concurrent access, data integrity, security, and recovery from failures. A DBMS allows users and applications to interact with data without needing to know the physical storage details, ensuring data consistency and reliability across various operations.

Explain the three-schema architecture in database systems.

The three-schema architecture, also known as ANSI/SPARC architecture, proposes three levels of data abstraction: the internal schema, the conceptual schema, and the external schema. The internal schema describes the physical storage structure of the database. The conceptual schema describes the entire database structure for a community of users, hiding the physical details. The external schema (or view schema) describes a part of the database relevant to a particular user group, further abstracting from the conceptual schema. This architecture promotes data independence.

What are the four ACID properties of transactions, and why are they crucial?

The four ACID properties are Atomicity, Consistency, Isolation, and Durability. Atomicity ensures transactions are all-or-nothing. Consistency guarantees transactions move the database from one valid state to another. Isolation ensures concurrent transactions appear to execute serially. Durability means committed changes are permanent. These properties are crucial because they collectively guarantee the reliability and integrity of data in a database system, especially in the face of concurrent access and system failures, preventing data corruption and ensuring business logic is correctly applied.

Describe the main differences between a file-processing system and a DBMS.

File-processing systems manage data in separate files, leading to data redundancy, inconsistency, and difficulty in accessing related data. They lack centralized control, data independence, and robust security or recovery mechanisms. A DBMS, in contrast, offers centralized data management, reducing redundancy and improving consistency. It provides data

independence, allowing schema changes without affecting applications, and includes sophisticated features for data integrity, security, concurrency control, and crash recovery, making it superior for complex data management needs.

What is data independence, and why is it important in database design?

Data independence refers to the ability to modify the schema at one level of a database system without affecting the schema at the next higher level. There are two types: physical data independence (changing physical storage without affecting conceptual schema) and logical data independence (changing conceptual schema without affecting external schemas). It's important because it reduces maintenance costs, allows for system evolution without rewriting applications, and simplifies database administration, making the system more flexible and adaptable to changing requirements.

How does the Entity-Relationship (ER) model help in database design?

The ER model helps in database design by providing a high-level conceptual view of the data. It allows designers to represent real-world entities, their attributes, and the relationships between them in a clear, graphical manner. This model facilitates communication between users and designers, ensuring that business requirements are accurately captured. It serves as a blueprint that can then be systematically translated into a relational schema, forming the basis for the logical design of the database, minimizing ambiguities and design flaws early on.

What is normalization, and what are the first three normal forms (1NF, 2NF, 3NF)?

Normalization is a process of organizing a relational database schema to minimize data redundancy and improve data integrity. 1NF requires that all attributes contain atomic values and there are no repeating groups. 2NF requires 1NF and that all non-key attributes are fully functionally dependent on the primary key, eliminating partial dependencies. 3NF requires 2NF and that all non-key attributes are non-transitively dependent on the primary key, removing transitive dependencies. These forms progressively reduce anomalies and improve schema quality.

Explain the concept of a transaction schedule and its role in concurrency control.

A transaction schedule is a sequence of operations (read, write, commit, abort) from a set of concurrent transactions, indicating the chronological order in which these operations are executed. Its role in concurrency control is to analyze and ensure that concurrent execution of transactions is correct, meaning it produces the same result as some serial execution of those transactions. Schedulers use techniques like locking or timestamping to order operations, preventing conflicts and maintaining the ACID properties, particularly isolation and consistency.

How do locks work to ensure concurrency control in a database system?

Locks are mechanisms used to control concurrent access to data items. When a transaction wants to read or write a data item, it first requests a lock on that item. Shared locks (S-locks)

allow multiple transactions to read concurrently but prevent writes. Exclusive locks (X-locks) allow only one transaction to access the item for reading or writing, blocking all other access. By acquiring and releasing appropriate locks, transactions prevent conflicts like dirty reads, lost updates, and unrepeatable reads, thereby ensuring serializability and data consistency.

What is the purpose of indexing in a database, and how does it improve query performance?

The purpose of indexing is to speed up data retrieval operations on database tables. An index is a data structure (like a B-tree or hash table) that stores a small, ordered subset of the data from a table, along with pointers to the full records. When a query searches for data based on an indexed column, the DBMS can quickly navigate the index to find the relevant records, avoiding a full table scan. This significantly reduces I/O operations and CPU usage, leading to faster query execution, especially for large tables.

Describe the role of a Database Administrator (DBA).

The Database Administrator (DBA) is responsible for the overall management and maintenance of a database system. Their roles include schema definition, storage structure and access method definition, schema and physical organization modification, granting user access and authorization, specifying integrity constraints, and performing backup and recovery. The DBA ensures the database's security, performance, availability, and integrity, acting as the central authority for all database-related operations and policies within an organization.

What are the different types of joins available in SQL, and when would you use each?

SQL offers several types of joins. An INNER JOIN returns only rows where there is a match in both tables, used when you need records with corresponding entries in both. A LEFT (OUTER) JOIN returns all rows from the left table and matching rows from the right, or NULLs if no match, useful for seeing all records from one table and their related data. A RIGHT (OUTER) JOIN is similar but prioritizes the right table. A FULL (OUTER) JOIN returns all rows from both tables, with NULLs where no match, used when you need all records from both tables regardless of matches.

How does a recovery manager ensure database durability after a system crash?

A recovery manager ensures durability by using logging and checkpointing mechanisms. Before any change is made to the database, a log record is written to stable storage, detailing the operation. In case of a crash, the recovery manager uses these logs to either undo uncommitted transactions (rollback) or redo committed transactions (rollforward) that might not have been fully written to disk. Checkpoints periodically flush modified buffer blocks to disk and record the state, reducing the amount of log scanning needed during recovery, thus restoring the database to a consistent state.

What is the difference between a data definition language (DDL) and a data manipulation language (DML) in SQL?

DDL (Data Definition Language) is used to define and manage the database schema. Commands like CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, and GRANT/REVOKE permissions fall under DDL. DML (Data Manipulation Language) is used for managing data within the schema. Commands like SELECT, INSERT, UPDATE, and DELETE are DML commands. DDL statements are typically executed by DBAs or designers to structure the database, while DML statements are used by applications and users to interact with the data stored within that structure.

Explain the concept of referential integrity and how foreign keys enforce it.

Referential integrity is a database concept that ensures that relationships between tables remain consistent. It dictates that if a foreign key in one table refers to a primary key in another table, then every value of the foreign key must either be NULL or exist as a primary key value in the referenced table. Foreign keys enforce this by preventing actions that would destroy these links, such as deleting a primary key record that is still referenced by a foreign key, or inserting a foreign key value that has no corresponding primary key.

What is the difference between a strong entity set and a weak entity set in the ER model?

In the ER model, a strong entity set is an entity set that has a primary key and can exist independently. Its existence does not depend on the existence of another entity set. A weak entity set, on the other hand, does not have sufficient attributes to form a primary key on its own. Its existence is dependent on a strong entity set, called its identifying or owner entity set. The primary key of a weak entity set is formed by combining its partial key with the primary key of its identifying strong entity set.

How does query optimization work in a DBMS?

Query optimization is the process of selecting the most efficient execution plan for a given SQL query. When a query is submitted, the query optimizer analyzes it, generates multiple possible execution plans (e.g., different join orders, index usage, access methods), and estimates the cost (CPU, I/O, network) for each plan using statistics about the data. It then chooses the plan with the lowest estimated cost. This process aims to minimize the resources consumed and maximize the speed of query execution, often transparently to the user.

What are the challenges of distributed databases compared to centralized ones?

Distributed databases present several challenges compared to centralized ones. These include increased complexity in design and management due to data fragmentation and replication across multiple sites. Concurrency control and recovery become more intricate, requiring distributed protocols to maintain global consistency and atomicity. Network delays and failures introduce performance and reliability issues. Security management is also more complex across multiple nodes. Additionally, achieving data transparency (location and fragmentation transparency) while maintaining performance is a significant hurdle.

What is the purpose of views in SQL?

Views in SQL are virtual tables based on the result-set of an SQL query. They do not store data themselves but rather provide a dynamic window into the underlying base tables. Their primary purposes include simplifying complex queries by pre-joining tables or aggregating data, enhancing security by restricting users to specific rows or columns of a table, and providing data independence by allowing changes to the base tables without affecting applications that access data through the view, as long as the view's definition remains valid.

Explain the concept of functional dependency in relational database theory.

A functional dependency (FD) is a constraint between two sets of attributes in a relation. It states that if two tuples have the same values for a set of attributes A, then they must also have the same values for a set of attributes B. This is written as A B, meaning A functionally determines B. FDs are fundamental to relational database design, particularly in normalization, as they help identify and eliminate redundancy and update anomalies by guiding the decomposition of relations into smaller, well-structured tables.