Notes on "Introduction to DBMS"

Notes on "Introduction to DBMS"

Key Definitions

- **Database**: A shared collection of logically interrelated data designed to meet the information needs of an organization.
- **DBMS (Database Management System)**: Software that facilitates defining, constructing, manipulating, and sharing databases among users and applications. Examples: Oracle, MySQL.

Applications of Databases

- **Enterprise Information**: Sales, accounting, human resources.
- **Banking and Finance**: Customer accounts, transactions.
- **Telecommunication**: Call records, billing, network usage.
- **Social Media**: Online advertisements and user data.
- **Document Databases**: Storing structured documents.

Why Database Systems?

Database systems address the limitations of file-processing systems:

1. Data redundancy and inconsistency.
2. Difficulty in accessing data.
3. Data isolation across formats.
4. Integrity problems with constraints buried in code.
5. Atomicity issues during updates.
6. Concurrent access conflicts.
7. Security challenges.

Levels of Data Abstraction

1. **Physical Level**: How data is physically stored (low-level details).
2. **Logical Level**: Structure of the database (e.g., relationships between data).
3. **View Level**: Simplified views for specific users or applications.

Instances and Schemas

- **Schema**: Overall database design.
  - Physical Schema: Design at the physical level.
  - Logical Schema: Design at the logical level.
  - Subschemas: Views for specific users.
- **Instance**: Data stored at a particular moment.

Data Independence

- Ability to modify one schema level without affecting others:
  - **Physical Data Independence**: Changes in physical schema don't affect logical schema.
  - **Logical Data Independence**: Changes in logical schema don't affect views.

Data Models

Conceptual tools for describing:

- Data
- Relationships
- Semantics
- Constraints

Types:

1. Relational Model
2. Entity-Relationship Model
3. Object-Based Model
4. Semi-Structured Model

Normalization

A method to design schemas with minimal redundancy using functional dependencies.

Database Languages

1. **Data Manipulation Language (DML)**:
    - Procedural: Specifies how to retrieve data.
    - Non-Procedural (Declarative): Specifies what data to retrieve (e.g., SQL).
2. **Data Definition Language (DDL)**:
    - Defines schema and constraints (e.g., domain constraints, referential integrity).

System Structure of DBMS

1. **Storage Manager**:
    - Handles storage and retrieval efficiently.
    - Components:
        - Authorization Manager
        - Transaction Manager
        - File Manager
        - Buffer Manager
    - Data Structures:
        - Data files
        - Metadata dictionary
        - Indices for fast access.
2. **Query Processor**:
    - Components:
        - DDL Interpreter
        - DML Compiler (optimizes queries)
        - Query Evaluation Engine.

Transaction Management

- Ensures consistency despite failures or concurrent transactions.
- Components:
    - Concurrency-Control Manager
    - Recovery Manager

Database Users

1. Naive Users: Use pre-written programs.
2. Application Programmers: Write application-specific programs.
3. Sophisticated Users: Use query languages or analysis tools.
4. Specialized Users: Develop non-traditional applications like expert systems.

Database Architectures

1. **2-Tier Architecture**:
    - Client interacts directly with the database server.
2. **3-Tier Architecture**:
    - Layers:
        1. Client Layer
        2. Business Logic Layer
        3. Data Layer (database)

This summary captures the key concepts from Unit 1 on DBMS, including its purpose, structure, and functionality.

⁂