ملاحظة: المكثف بهدف لتلخيص اهم النقاط لقواعد البيانات, بس ما بظمنلك تجيب العلامة الكاملة, رح تستفيد من هاظ المكثف اكثر لو درست المادة قبل تدرس هان, و انا مش مسؤول عن اي حد ما جاب العلامة الي بدو اياها

Chapter 1

Database : A Collection of related data or Related info that can be recorded and has meaning i.e Amazon.

Mini world (universe of discourse (UoD)): Represents some Real World data, that's logically coherent and built for a specific purpose.

Database management system (DBMS): Programs that enable users to create and maintain data.

To Define a database, you need to specify the data types, structures, and constraints of the data to be stored.

Meta-data: descriptive information Stored by the DBMS in the form of a database catalog or dictionary.

To Manipulate a database we use queries.

Types of databases

1. Traditional databases: Text or Numbers
2. Multimedia databases: images, audio clips, and video streams
3. Geographic information systems (GIS): maps, weather data, and satellite images
4. Data warehouses and online analytical processing (OLAP) systems: useful business information from very large databases , which supports decision making
5. Real-time and active database technology: Control industrial and manufacturing processes

Components of a database system

1. Users
2. Applications programs, queries
3. The DBMS, to process queries, programs, and to access stored data.
4. The Data Base, including meta data, and the stored data base

Databases can be shared (used by more than 1 user), and the data can be accessed using queries, and can be maintained (updated overtime), and are protected by system and

security protection.

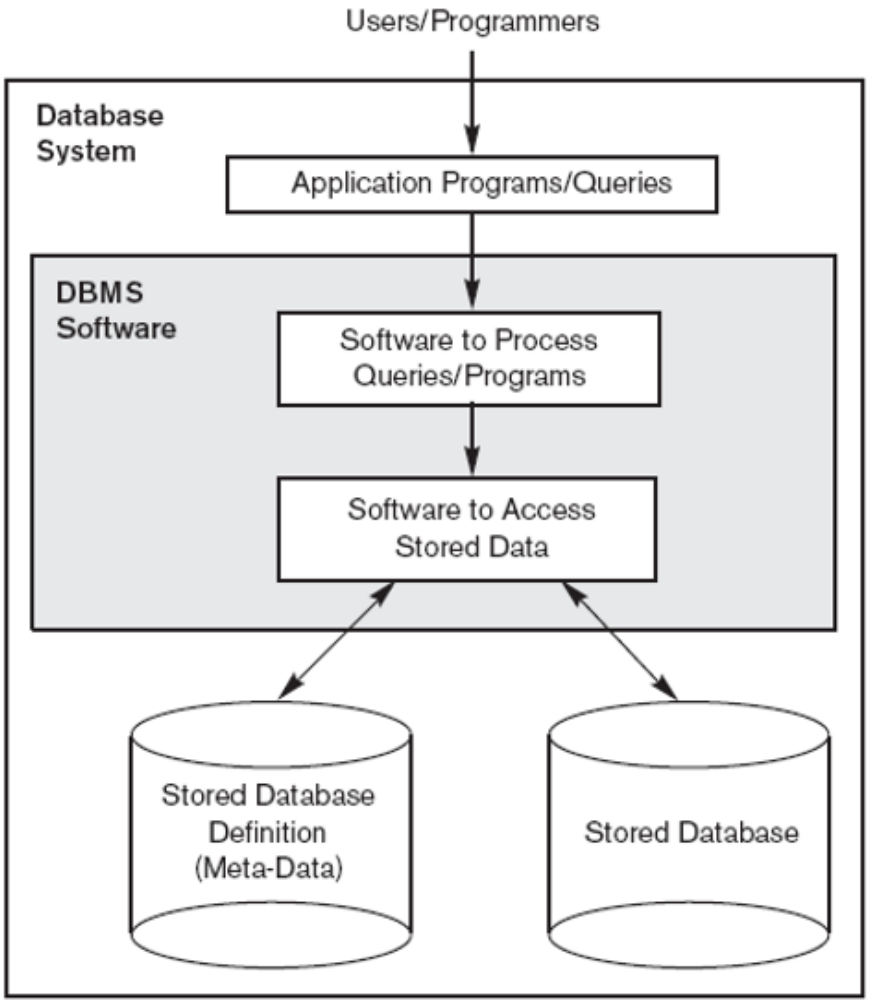

Figure 1.1
A simplified database system environment.

Phases for designing a database

1. Requirements specification and analysis
2. Conceptual design
3. Logical design
4. Physical design

Characteristics of the Database Approach

1. Traditional file processing: Each user defines and implements the files needed for a specific software application

2. Contain a Database approach: Single repository maintains data that is defined once and then accessed by various users

Main characteristics of database approach

A. Self-describing nature of a database system

as in it contains complete definition of structure and constraints, with a catalogue used DBMS software and the user.

B. Insulation Between Programs and Data

Structure of data files is stored in DBMS catalog separately from access programs, where operations are specified in two parts:

1. Interface includes operation name and data types of its arguments
2. Implementation can be changed without affecting the interface

Data abstraction: Facilitates independence between programs and data, as well as between programs and operations.

Conceptual representation of data: Excludes specifics regarding data storage or operational implementation, focusing purely on conceptual aspects.

Data model: A form of data abstraction employed to offer a conceptual representation.

C. Support of Multiple Views of the Data

View: A subset of the database that presents data derived from the database files but isn't physically stored, typically created to meet specific needs or requirements.

Multiuser DBMS: Supports multiple users with diverse applications by offering tools to define and manage various views tailored to their respective needs.

D. Sharing of Data and Multiuser Transaction Processing

Allow multiple users to access the database at the same time

this characteristic usually includes

1. Concurrency Control Mechanisms: Software designed to regulate updates to shared data among multiple users, ensuring correctness and integrity of results.

2. Online Transaction Processing (OLTP) Applications: Systems optimized for managing and processing a large number of transactions in real-time.

3. Transaction: Core component of database applications, comprising a program or process that interacts with one or more databases.

4. Isolation Property: Ensures that each transaction appears to execute independently of other concurrent transactions, maintaining data consistency.

5. Atomicity Property: Guarantees that either all database operations within a transaction are executed successfully or none of them are, preserving the integrity of the database.

Types of Users

1. Workers behind the Scene:

A. Database administrators (DBA) are responsible for:

- Authorizing access to the database
- Coordinating and monitoring its use
- Acquiring software and hardware resources

B. Database designers are responsible for:

- Identifying the data to be stored
- Choosing appropriate structures to represent and store this data

C. System analysts:

- Determine requirements of end users

D. Application programmers:

- Implement these specifications as programs

E. DBMS System Designers and Implementers:

- Design and implement the DBMS modules and interfaces as a software package.

F. Tool Developers:

- Design and implement tools.

G. Operators and Maintenance Personnel:

- Responsible for running and maintaining the hardware and software environment for the database system.

2- Actors on the Scene

End users: Those needing database access for their work

Types:

Casual users
Naive or parametric users
Sophisticated users
Standalone users

Advantages of Using the DBMS

1. Controlling redundancy: rough controlled redundancy via data normalization and occasional denormalization for query efficiency.
2. Restricting unauthorized access: through a dedicated security and authorization subsystem with privileged software components.
3. Facilitating permanent storage of program objects, where complex objects in C++ can be stored in object-oriented DBMS, addressing the impedance mismatch problem with compatibility offered by object-oriented database systems.
4. Enhancing query processing efficiency: through storage structures like indexes, along with buffering, caching, and optimization techniques to improve performance.
5. Providing backup and recovery : through the DBMS's Backup and recovery subsystem
6. Providing multiple user interfaces: using a GUI.
7. Representing complex relationships among data: May include numerous varieties of data that are interrelated in many ways.
8. Enforcing integrity constraints: where every record must be related to a course record, and be unique.
9. Permitting inferencing and actions using rules: Deductive database systems enable inferencing, action, and rule enforcement through triggers (Rule activated by updates to the table) and stored procedures (More involved procedures to enforce rules).
10. Additional implications of using the database approach: they can reduce development time, offer flexibility, access to real-time data, and economies of scale.

A Brief History of Database Applications

Database evolution began with hierarchical and network systems, moving to relational databases for data abstraction and flexibility. Object-oriented applications demanded more complex databases, especially in specialized fields. XML became crucial for interchanging data online. Database capabilities expanded for ERP and CRM. Information retrieval (IR) deals with content like books and manuscripts separately from traditional databases.

When Not to Use a DBMS

1. Simple, static database applications.

2. Real-time needs where database overhead might be an issue.

3.Regular files are suitable for embedded systems with limited storage.

4. When There's no requirement for multiple-user access to data.

Chapter 2

Classification of Database Management

Based On				
Classification	Hierarchical	Network	Object Relational	Relational
User	Single User	Multi User		
Site	Centralized	Client-Server	Distributed	
Cost	Open source	Different types of licensing		
Use	General	Special-purpose		

---

1. Based on Data model

Data Model:

- Database structure description
- Enables data abstraction

Data Model Operation:

- Basic operations: Retrieve and update database
- Dynamic behavior definition: Specifies valid operations on objects

Categories of Data Models

1. High-level or Conceptual Data Models: - Represent user-centric perspective through abstract data representation.

2. Low-level or Physical Data Models:

- Provide detailed technical depiction of how data physically resides on storage media.

3. Representational Data Models:

- Offer user-friendly representation resembling data organization for easy comprehension.

4. Physical Data Models:

- Define file-based data depiction, including storage format and access mechanisms.

Access Path:

- Efficient data retrieval route enabling optimized record searching.

Index:

- Serve as search enhancement mechanism facilitating direct data access through keyword or index term.

Components of Data Models

Entity: Represents a tangible object or abstract concept.

Attribute: Describes a specific characteristic or property of an entity.

Relationship: Indicates an association between two or more entities.

Entity-Relationship Model: Illustrates entities, attributes, and relationships in a structured format.

Relational Data Model: Predominantly employed in traditional commercial database management systems.

Object Data Model: A newer category of data models, aligning closely with conceptual representations.

Schemas, Instances, and Database State

Database Schema: Description of database structure.

Schema Diagram: Visual display of schema aspects.

Schema Construct: Schema object.

Database State: Current data snapshot.

Data Abstraction: Simplifying data representation.

Schema Evolution: Adaptation of schema to accommodate changing application needs.

Schemas steps

Define New Database: Establishing a fresh database instance.

Initial State: Database loaded with initial data.

Valid State: Database conforms to schema structure and constraints.

Three-Schema Architecture, levels

Internal Level: Describes physical storage structure of the database.

Conceptual Level: Describes overall structure of the database for users.

External or View Level: Describes specific parts of the database that are relevant to users.

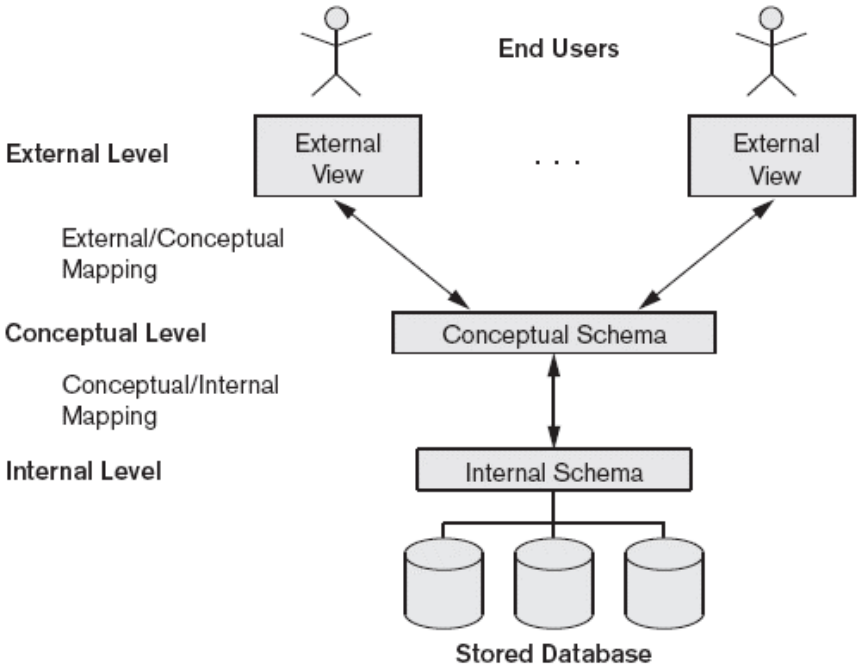

Figure 2.2
The three-schema architecture.

Data Independence

Data Independence: Ability to change one level's schema without altering the next higher level's schema.

Types:

- Logical
- Physical

---

2. Based on Number of users

•Single-User (MS Access)

•Multiuser (My SQL, Oracle)

---

3. Based on Number of sites

•Centralized

•Distributed (includes):
1. Homogeneous
2.
3. Heterogeneous

Site: Centralized and Distributed

(Client/Server Architectures for DBMSs)

Centralized DBMS Architecture: All functions on one machine.
(like application program execution, and user interface processing)

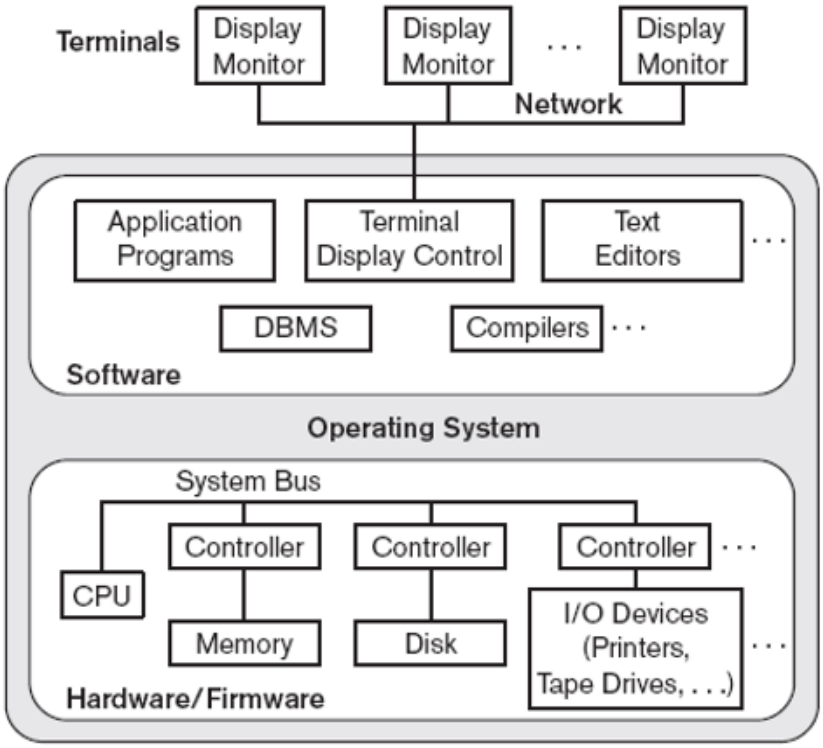

Figure 2.4
A physical centralized architecture.

Basic Client/Server Architectures

Client: - User machine with interface and local processing.

Server:

- System providing hardware and software services to clients.
- Services include file access, printing, archiving, and database access.

Specialized Servers:

1. File Server: Manages client files.
2. Printer Server: Directs print requests to printers.
3. Web or Email Servers.

Client Machines:

- Utilize server interfaces.
- Run local applications.

DBMS Example:

- Server manages queries, updates, transactions.
- Client handles user interface and applications.

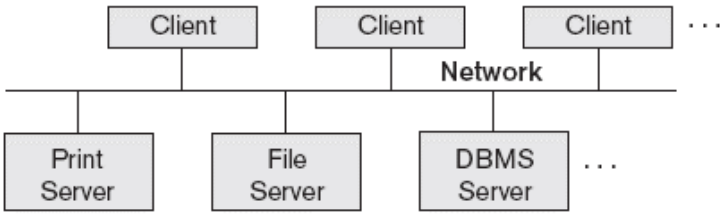

Figure 2.5
Logical two-tier client/server architecture.

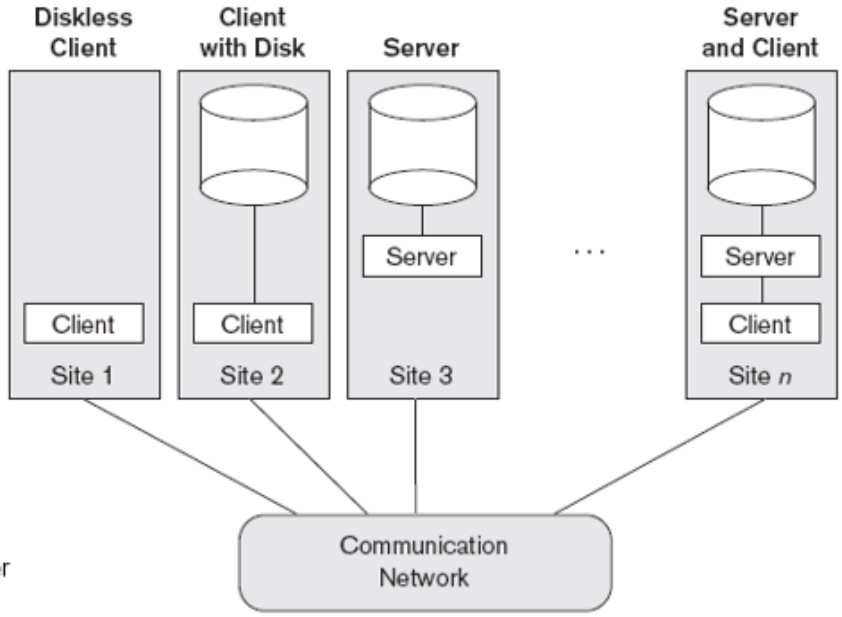

Figure 2.6
Physical two-tier client/server architecture.

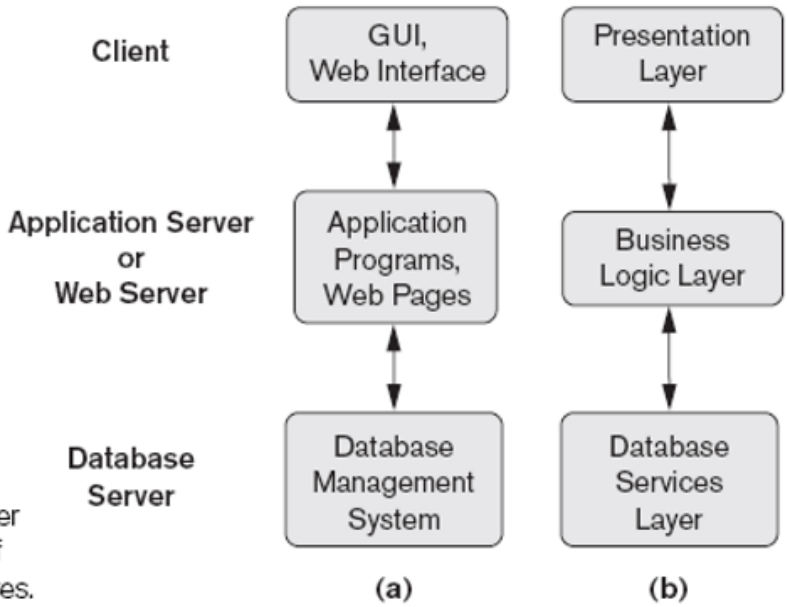

Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

4. Based on Cost

•Open source

•Different types of licensing

5. Based on Access path options

- General - Special Purpose

DBMS Languages

Data Definition Language (DDL):

- Defines both schemas.

Storage Definition Language (SDL):

- Specifies the internal schema.

View Definition Language (VDL):

- Defines user views and mappings to the conceptual schema.

Data Manipulation Language (DML):

- Enables retrieval, insertion, deletion, and modification.

DML types

High-level or nonprocedural DML: - Specifies complex database operations concisely. - Set-at-a-time or set-oriented.

Low-level or procedural DML:

- Embedded in a general-purpose programming language.
- Operates on a record-at-a-time basis.

DBMS Interfaces

- Menu-based interfaces for web clients or browsing - Forms-based interfaces - Graphical user interfaces - Natural language interfaces - Speech input and output - Interfaces for parametric users - Interfaces for the DBA

The Database System Environment (DBMS component modules)

DBMS component modules

- Buffer Management - Data Manager - DDL Compiler - Query Interface - Query Compiler - Optimizer - Precompiler - Runtime Processor - System Catalog - Concurrency Control - Backup System

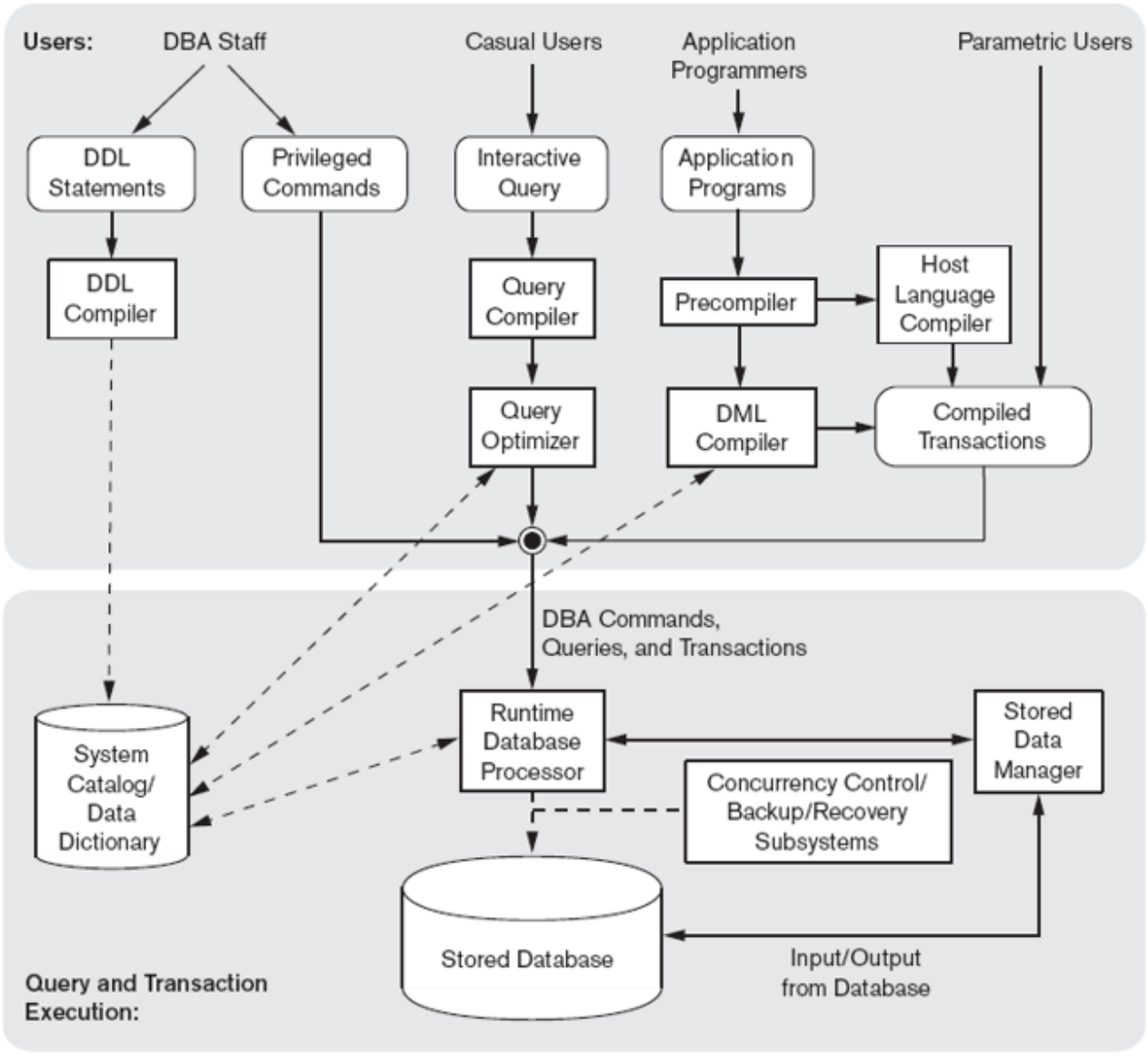

Figure 2.3
Component modules of a DBMS and their interactions.

Database System Utilities

- Loading: Load existing data files - Backup: Create database backup - Storage Reorganization: Rearrange database files - Performance Monitoring: Track database usage

DBMS Facilities

CASE tools

1. Data dictionary (data repository) system: stores design decisions, usage standards, application program descriptions, and user information. 2. Application development environments. 3. Communications software.

Chapter 3

Relational model: First available in the early 80s, and has been implemented in many large commercial systems, it was preceded by the hierarchical and network models.

Relational Model Concepts

Table

Represents data as a collection of relations, and contains

1. Rows(tuples, records): a collection of related data values, typically corresponds to a real-world entity or relationship
2. Column (Attribute): Interpret the meaning of the values in each row attribute
3. **Entity instance (relation instance):** A specific entry in a table, like a row (tuple, record), holding unique data for each category member while sharing the same set of attributes.

Type of attributes

1. Simple: single value like gender, id.
2. Composite: several value for same attribute like name (first, second, family name).
3. §Multi: multi value for same attribute but we take just one (choices) like (car color: red or blue or…..).
4. derived attributes:

Domains, Attributes, Tuples, and Relations

Domain: Set of atomic values,that contain name, data type, format

or A domain is the set of Atomic values that an attribute belonging to that domain can take.

Atomic: Each value indivisible (not composite)

Specifying a domain: Data type specified for each domain(int, varchar,…..)

Any attribute in a relationship has a domain.

The well-known Data types (int, char,...) can be considered domains for attributes. A domain can also be defined and have a name, type, and format, this is called a logical definition of domain.

Example: The domain for the attribute StdNo in the Student relationship
Domain name: Student Number, Type: char(5), format: dd-dd-dd .

Relation schema R(table schema witch contain the attribute, its domain, constraint). Denoted by $R(A_1, A_2, ..., A_n)$, Made up of a relation name R and a list of attributes, $A_1, A_2, ..., A_n$.

Attribute A_i: Name of a role(attribute) played by some domain D in the relation schema R(table name

Degree (or arity) of a relation: Number of attributes n of its relation schema

Relation (or relation state/ instance): a set of n-tuples, each comprising an ordered list of values from domains or a special NULL value, forming a state or instance of the relation

Current relation state: Relation state at a given time, reflects only the valid tuples that represent a particular state of the real world

Cardinality: Total number of values in domain

Initial state : before entering data

Attribute names: different roles, or interpretations, for the domain.

Characteristics of Relations

1. Ordering of tuples in a relation
    - Relation defined as a set of tuples
    - Elements have no order among them
2. Ordering of values within a tuple and an alternative definition of a relation
    - Order of attributes and values is not that important
    - As long as correspondence between attributes and values maintained
3. Alternative definition of a relation
    - Tuple considered as a set of (`<attribute>, <value>`) pairs

- Each pair gives the value of the mapping from an attribute A_i to a value v_i from $dom(A_i)$

4. Use the first definition of relation
    - Attributes and the values within tuples are ordered
    - Simpler notation(addressing the values according the data entry)
5. Values and NULLs in tuples
    - Each value in a tuple is atomic
    - Flat relational model: Composite and multivalued attributes not allowed. First normal form is assumed.
    - Multivalued attributes (Must be represented by separate relations)
    - Composite attributes (Represented by simple component attributes in basic relational model)
6. NULL values
    - Represent the values of attributes that may be unknown or may not apply to a tuple
    - Meanings for NULL values
        1. Value unknown
        2. Value exists but is not available
        3. Attribute does not apply to this tuple (also known as value undefined)
7. Interpretation (meaning) of a relation
    - Assertion: Each tuple in the relation is a fact or a particular instance of the assertion
    - Predicate: Values in each tuple interpreted as values that satisfy predicate

---

Relational Model Notation

1. Relation schema R of degree n
    - Denoted by $R(A_1, A_2, ..., A_n)$
2. Uppercase letters Q, R, S
    - Denote relation names
3. Lowercase letters q, r, s
    - Denote relation states
4. Letters t, u, v
    - Denote tuples
5. Name of a relation schema: STUDENT
    - Indicates the current set of tuples in that relation
6. Notation: STUDENT(Name, Ssn, ...)
    - Refers only to relation schema

7. Attribute A can be qualified with the relation name R to which it belongs using the dot notation.
8. Notation of n-tuple in a Relation:
    - Denoted by $t = <v_1, v_2, ..., v_n>$
    - v_i is the value corresponding to attribute A_i
9. **Component Values of Tuples:**
    - $t[A_i]$ and $t.A_i$ refer to the value v_i in t for attribute A_i
    - $[A_u, A_w, ..., A_z]$ and $t.(A_u, A_w, ..., A_z)$ refer to the subtuple of values $<v_u, v_w, ..., v_z>$ from t corresponding to the attributes specified in the list

---

Part two: Relational Model Constraints

Constraints: Restrictions on the actual values in a database state, Derived from the rules in the database's miniworld.

Type of Constraints:

1. Inherent model-based constraints or implicit constraints

- Inherent in the data model

Domain Constraints, they include:

1. Numeric data types for integers and real numbers
2. Characters
3. Booleans
4. Fixed-length strings
5. Variable-length strings
6. Date, time, timestamp
7. Money
8. Other special data types

Key Constraints and Constraints on NULL Values

No two tuples can have the same combination of values for all their attributes

Superkey: No two distinct tuples in any state r (relation state, values) of R can have the same value for SK(it must be unique)

Key: Superkey of R, Removing any attribute A from K leaves a set of attributes K that is not a superkey of R any more

Key satisfies two properties:
1. Two distinct tuples in any state of relation cannot have identical values for (all) attributes in key (unique rule)
2. Minimal superkey: Cannot remove any attributes and still have uniqueness constraint in above condition hold

Candidate key: Relation schema may have more than one key

Primary key of the relation: Designated among candidate keys(its one of candidate key), represented as a Underline attribute.

Other candidate keys are designated as unique keys.

---

2. Schema-based Constraints vs Explicit Constraints

Schema-based constraints are those directly expressed within the data model schemas.

Relational Database Schema (S):
Set of relation schemas: S={R1,R2,...,Rm}
Set of integrity constraints: IC

Relational Database State (DB): Set of relation states: DB={r1,r2,...,rm}DB={r1,r2,...,rm}
Each ri is a state of Ri, ensuring that ri satisfies integrity constraints specified in IC.

Invalid State: A state that does not adhere to all the integrity constraints.

Valid State: A state that satisfies all constraints in the defined set of integrity constraints ICIC.

Integrity, Referential Integrity, and Foreign Keys

Relation (Entity) Integrity Constraint: No primary key value can be NULL.

Referential Integrity Constraint (Foreign Key): Specified between two relations, it maintains consistency among tuples in two relations.

Foreign Key Rules:

1. The attributes in a Foreign Key (FK) have the same domain(s) as the primary key attributes (PK).

2. The value of FK in a tuple $t1$ of the current state $r1(R1)$ either occurs as a value of PK for some tuple $t2$ in the current state $r2(R2)$or is NULL.

Diagrammatic Representation of Referential Integrity Constraints: Directed arcs from each foreign key to the relation it references.

All integrity constraints should be specified on the relational database schema.

3- Application-based or semantic constraints or business rules

Cannot be directly expressed in schemas, Expressed and enforced by application program

Semantic integrity constraints

1. May have to be specified and enforced on a relational database
2. Use triggers and assertions
3. More common to check for these types of constraints within the application programs
4. Cannot be directly expressed in schemas, Expressed and enforced by application program

Functional dependency constraint

Establishes a functional relationship among two sets of attributes X and Y, Value of X determines a unique value of Y

State constraints

Define the constraints that a valid state of the database must satisfy

Transition constraints

Define to deal with state changes in the database

Populated D.B state

Update Operations, Transactions, and Dealing with Constraint Violations

Operations of the relational model can be categorized into:

1. retrievals

2. updates Transaction

Basic operations that change the states of relations in the database:

1. Insert:
   Provides a list of attribute values for a new tuple t that is to be inserted into a relation R, Can violate any of the three types of constraints, If an insertion violates one or more constraints, Default option is to reject the insertion
2. Delete:
   Can violate only referential integrity, If tuple being deleted is referenced by foreign keys from other tuples
   Restrict
   •Reject the deletion
   Cascade
   •Propagate the deletion by deleting tuples that reference the tuple that is being deleted
   Set null or set default
   •Modify the referencing attribute values that cause the violation
3. Update (or Modify)
   Necessary to specify a condition on attributes of relation: Select the tuple (or tuples) to be modified
   If attribute not part of a primary key nor of a foreign key, Usually causes no problems
   Updating a primary/foreign key: Similar issues as with Insert/Delete

---

The Transaction Concept

Transaction: Executing program, Includes some database operations, Must leave the database in a valid or consistent state

Online transaction processing (OLTP) systems: Execute transactions at rates that reach several hundred per second

---

Chapter 4

SQL (Structured Query Language): one of the major reasons for the commercial success of relational databases, Statements for data definitions, queries, and updates (both DDL and DML), Core specification ,Plus specialized extensions.

Terminology: Table, row, and column used for relational model terms relation, tuple, and attribute

The main SQL command for data definition is CREATE

Schema and Catalog Concepts in SQL

SQL schema: Identified by a schema name, Includes an authorization identifier and descriptors for each element.

Schema elements include:
1.Tables
2. constraints
3. views
4. domains
5. other constructs

Each statement in SQL ends with a semicolon

example: CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’;

Catalog: Named collection of schemas in an SQL environment

SQL environment: Installation of an SQL-compliant RDBMS on a computer system

Create

Specify a new relation

need to Provide name and Specify attributes and initial constraints

Can optionally specify schema (example):

CREATE TABLE COMPANY.EMPLOYEE ... or CREATE TABLE EMPLOYEE ...

Base tables (base relations): Relation and its tuples are actually created and stored as a file by the DBMS

Virtual relations: Created through the CREATE VIEW statement

Some foreign keys may cause errors if they Specified either via:
1. Circular references
2. Or because they refer to a table that has not yet been created

---

Attribute Data Types and Domains in SQL

Basic data types

1. Numeric data types (includes): 1. Integer numbers: INTEGER, INT, and SMALLINT 2. Floating-point (real) numbers: FLOAT or REAL, and DOUBLE PRECISION

2. Character-string data types (includes):
    1. Fixed length: CHAR(n), CHARACTER(n)
    2. Varying length: VARCHAR(n), CHAR VARYING(n), CHARACTER VARYING(n)
3. Bit-string data types
    1. Fixed length: BIT(n)
    2. Varying length: BIT VARYING(n)
4. Boolean data type
    1. Values of TRUE or FALSE or NULL
5. DATE data type
    1. Ten positions
    2. Components are YEAR, MONTH, and DAY in the form YYYY-MM-DD

Additional data types

1. Timestamp data type (TIMESTAMP)
    1. Includes the DATE and TIME fields
    2. Plus a minimum of six positions for decimal fractions of seconds
    3. Optional WITH TIME ZONE qualifier
2. INTERVAL data type
    1. Specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp

Domain

Domain: Name used with the attribute specification

is used to make it easier to change the data type for a domain that is used by numerous attributes and Improves schema readability.

example: CREATE DOMAIN SSN_TYPE AS CHAR(9);

Specifying Constraints, Attribute Constraints and Attribute Defaults in SQL

Basic constraints:
1. Key and referential integrity constraints
2. Restrictions on attribute domains and NULLs
3. Constraints on individual tuples within a relation

NOT NULL: NULL is not permitted for a particular attribute

Default value: DEFAULT `<value>`

CHECK clause: Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);

Specifying Key and Referential Integrity Constraints

PRIMARY KEY clause

Specifies one or more attributes that make up the primary key of a relation

example: Dnumber INT PRIMARY KEY;

UNIQUE clause

Specifies alternate (secondary) keys

Dname VARCHAR(15) UNIQUE;

FOREIGN KEY clause

Default operation: reject update on violation

Attach referential triggered action clause
•Options include SET NULL, CASCADE, and SET DEFAULT

Action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE and ON UPDATE

CASCADE option suitable for “relationship” relations

Giving Names to Constraints

Keyword CONSTRAINT: Name a constraint, Useful for later altering

Specifying Constraints on Tuples Using CHECK

CHECK clauses at the end of a CREATE TABLE statement

Apply to each tuple individually

CHECK (Dept_create_date <= Mgr_start_date);

Basic Retrieval Queries in SQL

SELECT statement: One basic statement for retrieving information from a database

SQL allows a table to have two or more tuples that are identical in all their attribute values, Unlike relational model. (Multiset or bag behavior)

Basic form of the SELECT statement: SELECT `<attribute list>` FROM `<table list>` WHERE `<condition>`

The SELECT-FROM-WHERE Structure of Basic SQL Queries

Logical comparison operators
1. =
2. <
3. <=
4. >=
5. <>

Projection attributes: Attributes whose values are to be retrieved

Selection condition: Boolean condition that must be true for any retrieved tuple

Ambiguous Attribute Names

Same name can be used for two (or more) attributes, as long as the attributes are in different relations

Must qualify the attribute name with the relation name to prevent ambiguity

Aliasing, Renaming, and Tuple Variables

Aliases or tuple variables
Declare alternative relation names E and S
EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno)

Unspecified WHERE Clause

and Use of the Asterisk

Missing WHERE clause: Indicates no condition on tuple selection

CROSS PRODUCT: All possible tuple combinations

Specify an asterisk (*) to Retrieve all the attribute values of the selected tuples

Tables as Sets in SQL

SQL does not automatically eliminate duplicate tuples in query results

Use the keyword DISTINCT in the SELECT clause

Only distinct tuples should remain in the result

Set operations
UNION, EXCEPT (difference), INTERSECT
Corresponding multiset operations: UNION ALL, EXCEPT ALL, INTERSECT ALL)

Substring Pattern Matching and Arithmetic Operators

LIKE comparison operator
Used for string pattern matching
% replaces an arbitrary number of zero or more characters
underscore (_) replaces a single character

Standard arithmetic operators:
Addition (+), subtraction (–), multiplication (*), and division (/)

BETWEEN comparison operator

Ordering of Query Results

Use ORDER BY clause 1. Keyword DESC to see result in a descending order of values 2. Keyword ASC to specify ascending order explicitly

example: ORDER BY D.Dname DESC, E.Lname ASC, E.Fname AS

INSERT, DELETE, and UPDATE Statements in SQL

The INSERT Command

Specify the relation name and a list of values for the tuple

example:

INSERT INTO EMPLOYEE
VALUES
('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
Oak Forest, Katy, TX', 'M', 37000, '653298653',4);

INSERT INTO WORKS_ON_INFO (Emp_name, Proj_name,
Hours_per_week)

```
SELECT E.Lname, P.Pname, W.Hours
FROM PROJECT P, WORKS_ON W, EMPLOYEE E
WHERE P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

The DELETE Command

Removes tuples from a relation
Includes a WHERE clause to select the tuples to be deleted

example:

```
DELETE FROM EMPLOYEE
WHERE Lname='Brown';

DELETE FROM EMPLOYEE
WHERE Ssn='123456789';

DELETE FROM EMPLOYEE
WHERE Dno=5;

DELETE FROM EMPLOYEE;
```

The UPDATE Command

Modify attribute values of one or more selected tuples

Additional SET clause in the UPDATE command
Specifies attributes to be modified and new values

example:

```
UPDATE PROJECT
SET Plocation = 'Bellaire', Dnum = 5
WHERE Pnumber=10;
```

Additional Features of SQL

1. Techniques for specifying complex retrieval queries 2. Writing programs in various programming languages that include SQL statements 3. Set of commands for specifying physical database design parameters, file structures for relations, and access paths 4. Transaction control commands 5. Specifying the granting and revoking of privileges to users

6. Constructs for creating triggers 7. Enhanced relational systems known as object-relational 8. New technologies such as XML and OLAP

SQL Chapter 1

What is SQL?

Stands for Structured Query Language

Lets you access and manipulate databases

Is an ANSI (American National Standards Institute) standard

What Can SQL do?

1. Can execute queries against a database 2. Can retrieve data from a database 3. Can insert records in a database 4. Can update records in a database 5. Can delete records from a database 6. Can create new databases 7. Can create new tables in a database 8. Can create stored procedures in a database 9. Can create views in a database 10. Can set permissions on tables, procedures, and views

Using SQL in Your Web Site

To build a web site that shows data from a database, you will need:
1. An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
2. To use a server-side scripting language, like PHP or ASP
3. To use SQL to get the data you want
4. To use HTML / CSS

RDBMS

Stands for Relational Database Management System.

The basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access

The data in RDBMS is stored in database objects called tables.

Table: a collection of related data entries and it consists of columns and rows.

SQL (Structured Query Language) is a syntax for executing queries, But the SQL language also includes a syntax to update, insert, and delete records.

There two part of SQL
1. Data Manipulation Language (DML)

2. Data Definition Language (DDL)

Data Manipulation Language (DML)

These query and update commands together form the Data Manipulation Language (DML) part of SQL:

1. SELECT - extracts data from a database table
2. UPDATE - updates data in a database table
3. DELETE - deletes data from a database table
4. INSERT INTO - inserts new data into a database table

Data Definition Language (DDL)

The Data Definition Language (DDL) part of SQL permits database tables to be created or deleted.

We can also define indexes (keys), specify links between tables, and impose constraints between database tables.

The most important DDL statements in SQL are:

1. CREATE TABLE - creates a new database table
2. ALTER TABLE - alters (changes) a database table
3. DROP TABLE - deletes a database table
4. CREATE INDEX - creates an index (search key)
5. DROP INDEX - deletes an index

CREATE DATABASE Statement

Used to create a database.

Syntax: CREATE DATABASE dbname;

example: The following SQL statement creates a database called “Company":

CREATE DATABASE Company;

Database tables can be added with the CREATE TABLE statement.

SQL CREATE TABLE Statement

The CREATE TABLE statement is used to create a table in a database.

Tables are organized into rows and columns; and each table must have a name.

qSQL CREATE TABLE Syntax:

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);
```

The SQL INSERT INTO Statement

Used to insert new records in a table.

Syntax:vINSERT INTO table_name
VALUES (value1,value2,value3,...);

It is possible to write the INSERT INTO statement in two forms.

The first form does not specify the column names where the data will be inserted, only their values:

The second form specifies both the column names and the values to be inserted:

Syntax: INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);

SQL UPDATE Statement

Used to update records in a table.

Syntax:
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;

SQL DELETE Statement

Used to delete rows in a table, or delete all data.

Syntax:
DELETE FROM table_name
WHERE some_column=some_value;

It is possible to delete all rows in a table without deleting the table.

This means that the table structure, attributes, and indexes will be intact:

SQL DELETE Syntax(delete all data)
DELETE *
FROM table_name;

Note: Be very careful when deleting records. You cannot undo this statement!

SQL Chapter 2

SQL/ SELECT Statement

is used to select data from a database, the result is stored in a result table, called the result-set.

four types of select:

1. select all:
   Syntax:
   SELECT *
   FROM table_name;
2. SELECT some columns (specify some columns)
   Syntax:

SELECT column_name,column_name
FROM table_name;

3. The SQL SELECT DISTINCT Statement
   Used to return only distinct (different) values.
   Syntax:

SELECT DISTINCT column_name,column_name
FROM table_name;

4. The SQL WHERE Clause
   Used to extract only those records that fulfill a specified criterion.

Syntax:

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

SQL /LIKE Operator

Used in a WHERE clause to search for a specified pattern in a column.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

adding an '%' before the letter returns people with the start with that letter

ex: '%s' returns people that start with s

adding an '%' after the letter returns people with the end with that letter

ex: '%s' returns people that end with s

AND & OR Operators

AND: Displays a record if both the first condition AND the second condition are true.

EX:

```
SELECT * FROM Customers
WHERE Country='Germany'
AND City='Berlin';
```

OR: Displays a record if either the first condition OR the second condition is true.

EX:

```
SELECT * FROM Customers
WHERE Country='Germany'
OR City='Berlin';
```

ORDER BY Keyword

Used to sort the result-set by one or more columns.

sorts the records in ascending order by default.

you can use the DESC keyword to sort in a descending order

We can order by one or several columns

Syntax

```
SELECT column_name,column_name
FROM table_name
ORDER BY column_name,column_name ASC|DESC;
```

JOIN

Used to combine rows from two or more tables, based on a common field between them.

Types:

1. INNER JOIN: Returns all rows when there is at least one match in BOTH tables (most common)
   EX: SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
   FROM Orders
   INNER JOIN Customers
   ON Orders.CustomerID=Customers.CustomerID;

---

2. LEFT JOIN: Return all rows from the left table, and the matched rows from the right table, result is NULL in the right side when there is no match.
   Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN (or LEFT OUTER JOIN) table2
ON table1.column_name=table2.column_name;
```

---

3. RIGHT JOIN: Return all rows from the right table, and the matched rows from the left table, result is NULL in the left side when there is no match.
   Syntax:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN (or RIGHT OUTER JOIN) table2
ON table1.column_name=table2.column_name;
ON table1.column_name=table2.column_name;
```

---

4. FULL JOIN: Return all rows when there is a match in ONE of the tables, combines the result of both LEFT and RIGHT joins
   Syntax:

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

SQL Chapter 3

AVG()

returns the average value of a numeric column.

Syntax:

```
SELECT AVG(column_name)
FROM table_name
```

COUNT()

Returns the number of rows that matches a specified criteria.

types:

1. COUNT(*)
2. COUNT DISTINCT

COUNT(column_name) function returns the number of values (NULL values will not be counted) of the specified column.

Syntax:

```
SELECT COUNT(column_name)
FROM table_name;
```

Count(*)

Returns the number of records in a table:

Syntax:

```
SELECT COUNT( * )
FROM table_name;
```

COUNT(DISTINCT column_name)

Returns the number of distinct values of the specified column:

Syntax:

```
SELECT COUNT(DISTINCT column_name)
FROM table_name;
```

MAX()

Returns the largest value of the selected column.

SQL MAX() Syntax:

```
SELECT MAX(column_name)
FROM table_name;
```

MIN()

Returns the smallest value of the selected column.

SQL MIN() Syntax:

```
SELECT MIN(column_name)
FROM table_name;
```

SUM()

SUM() function returns the total sum of a numeric column.

SQL SUM() Syntax:

```
SELECT SUM(column_name)
FROM table_name;
```

DROP

To drop table or database.

DROP TABLE: used to delete a table.

Syntax:

```
DROP TABLE table_name
```

DROP DATABASE: is used to delete a database.

Syntax:

```
DROP DATABASE database_name
```

PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a database table.

Primary keys must contain unique values.

A primary key column cannot contain NULL values.

Most tables should have a primary key, and each table can have only ONE primary key.

Two ways to create primary key:
1. on CREATE TABLE
2. by alter the table

Constraint on CREATE TABLE

Example:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,LastName varchar(255) NOT NULL,FirstName varchar(255),
Address varchar(255),
City varchar(255),
```

```
PRIMARY KEY (P_Id)
)
```

Or:

```
CREATE TABLE Persons
(
P_Id int NOT NULL PRIMARY KEY,LastName varchar(255) NOT NULL,FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```