DBMS

Chapter 1

1). Here are two notable disadvantages of using a database system:

**a).Complexity and Cost**: Implementing and maintaining a database system can be complex and costly. Designing a database schema, ensuring data integrity, and writing efficient queries require expertise.

**b).Performance Overhead**: While database systems are designed to improve data management, they can introduce performance overhead due to their inherent structure. The process of data normalization, which helps maintain data consistency and integrity, can lead to complex query joins and increased processing times

**c).Single Point of Failure.**

**2).** List five ways in which the type declaration system of a language such as Java or C++ differs from the data definition language used in a database.

**Data Manipulation vs. Data Storage**:

**Runtime vs. Compile Time**:

**Operations and Constraints Granularity of Definition**: **Purpose and Context**:

3).

1. **Requirement Gathering and Analysis:**
   * Understand data needs and goals.
   * Gather requirements from stakeholders.
2. **Database Design:**
   * Define schema, tables, relationships.
   * Design data models.
3. **Selecting Database Management System (DBMS):**
   * Choose suitable DBMS based on factors.
   * Consider scalability, security, performance.
4. **Physical Database Setup:**
   * Install and configure DBMS.
   * Set up authentication and permissions.
5. **Data Migration and Population:**
6. **Testing and Optimization**

**4).**

Certainly, here are key differences between keyword queries used in web search and database queries:

**Web Search Keyword Queries:**

1. **Natural Language:** Often, users input queries in natural language, using everyday language and phrases.
2. **Keywords:** Queries usually consist of a few relevant keywords that describe the user's information need.
3. **Less Structured:** Queries are generally less structured and more flexible in terms of grammar and wording.
4. **Intent-Based:** Focus is on capturing the user's intent behind the query rather than specific database structure.

**5).**

Certainly, here are four applications that most likely employ a database system to store persistent data:

1. **Online Shopping Platforms (e.g., Amazon, eBay):**
2. **Social Media Platforms (e.g., Facebook, Twitter):**.
3. **Banking and Financial Apps (e.g., PayPal, Mint**
4. **Customer Relationship Management (CRM) Systems (e.g., Salesforce, HubSpot):**

6). Certainly, here are four significant differences between a file-processing system and a Database Management System (DBMS):

**Data Redundancy and Data Independence:**

**Concurrent Access and Data Anomalies: Data Integrity and Security:**

**Data Structure and Organization:**

**7).** Explain the concept of physical data independence and its importance in database systems.

1. **Performance Optimization:** As data volumes and usage patterns change over time, you might need to reorganize or tune the physical storage for better performance. This could involve changes like creating or removing indexes, optimizing data placement, or partitioning large tables. With physical data independence, you can make these optimizations without disrupting the existing applications and queries.
2. **Efficient Maintenance:** Maintenance tasks like backup, restoration, and recovery are often related to the physical storage. Having physical data independence means that these tasks can be performed without impacting the logical or conceptual views of the data.
3. **Adaptation to New Technologies:** Over time, new storage technologies and platforms might become available that offer better performance, scalability, or cost efficiency. Physical data independence allows you to migrate your data to these new technologies seamlessly, ensuring that the applications and queries continue to work without changes.
4. **Reduced Complexity:** Developers and administrators can focus on optimizing the physical storage layer without needing to coordinate changes across the entire system. This separation of concerns simplifies development, maintenance, and troubleshooting processes.
5. **Long-Term Data Management:**

**8).**

List five responsibilities of a database-management system. For each responsibility, explain the problems that would arise if the responsibility were not discharged.

1. **Data Integrity and Constraints:**
   * **Responsibility:** Enforce data integrity rules and constraints, such as unique keys, foreign keys, and data type validations.
   * **Problem if Not Discharged:** Without enforcing constraints, data could become inconsistent and inaccurate. Duplicate records and invalid references might lead to unreliable information and errors in applications.
2. **Concurrency Control:**
   * **Responsibility:** Manage concurrent access to the database by multiple users or processes, ensuring data consistency during simultaneous operations.
   * **Problem if Not Discharged:** Lack of proper concurrency control can lead to data anomalies like lost updates and inconsistencies, as multiple users might modify the same data simultaneously without coordination.
3. **Data Security and Access Control:**
   * **Responsibility:** Implement security measures to control access to data, including user authentication, authorization, and privilege management.
   * **Problem if Not Discharged:** Unprotected data might be vulnerable to unauthorized access, breaches, and data theft. Confidential information could be exposed, leading to legal and financial consequences.
4. **Data Backup and Recovery:**
   * **Responsibility:** Provide mechanisms for regular data backups and recovery in case of hardware failures, software bugs, or data corruption.
   * **Problem if Not Discharged:** Without proper backups and recovery mechanisms, data loss due to system failures or errors could be irreversible, causing significant disruption and potential loss of business-critical information.
5. **Query Optimization and Performance:**
   * **Responsibility:** Optimize the execution of queries to ensure efficient data retrieval and manipulation, including utilizing indexes and choosing optimal execution plans.
   * **Problem if Not Discharged:** Poorly optimized queries could lead to slow performance, causing delays in application response times and user frustration. Over time, this might lead to reduced system usability and efficiency.

In essence, a DBMS's responsibilities ensure data accuracy, consistency, security, and efficient management. Failing to discharge these responsibilities can result in unreliable data, security breaches, inefficient operations, and significant disruptions to business processes.

Top of Form

9)

9). List at least two reasons why database systems support data manipulation using a declarative query language such as SQL, instead of just providing a library of C or C++ functions to carry out data manipulation.

a declarative query language like SQL provides a high-level, abstract, and user-friendly way to interact with databases. It enables data independence, optimization, security, reduced code complexity, and enhanced productivity for both developers and users.

**High-Level Abstraction:**Top of Form

**High-Level Abstraction:**

**Optimization and Efficiency:**

**10).**

The component of a database system that prevents both students from being given the last seat in the course is the **Concurrency Control** mechanism.

Concurrency control ensures that multiple transactions (in this case, registration attempts by the two students) can access and modify the database simultaneously without leading to data inconsistencies or conflicts. In the scenario described, concurrency control prevents both students from being assigned the last seat by coordinating their actions and ensuring that only one of them successfully registers for the course.

**11).**

Explain the difference between two-tier and three-tier application architectures. Which is better suited for web applications? Why?

ChatGPT

**Two-Tier Application Architecture:**

In a two-tier architecture, also known as a client-server architecture, the application is divided into two main components:

1. **Client Tier (Presentation Tier):** This tier includes the user interface (UI) and presentation logic. It's responsible for interacting with the user, gathering input, and displaying results.
2. **Server Tier (Data Tier):** This tier handles business logic and data management. It manages data processing, business rules, and interactions with the database.

The communication between the client and server tiers is direct and usually involves the client sending requests directly to the server for processing.

**Three-Tier Application Architecture:**

A three-tier architecture extends the two-tier model by introducing an additional tier:

1. **Client Tier (Presentation Tier):** This tier remains responsible for the user interface and interaction with the user.
2. **Application Tier (Business Logic Tier):** In this tier, the application's business logic is separated from the presentation logic. It handles processing and executing business rules and operations.
3. **Data Tier (Data Storage Tier):** This tier manages data storage and retrieval. It interacts with databases and manages data persistence.

In summary, while both two-tier and three-tier architectures have their uses, a three-tier architecture is generally better suited for web applications due to its scalability, modularity, security, flexibility, and maintenance advantages.

12).

Certainly, here are two features that were developed in the 2000s to help database systems handle data-analytics workloads:

1. **Columnar Storage:**

**2).Massively Parallel Processing (MPP) Databases:**

instructor ⋈instructor.ID=teaches.ID teaches.

Relational Algebra

1).

Consider the time slot relation. Given that a particular time slot can meet more than once in a week, explain why day and start time are part of the primary key of this relation, while end time is not.

ChatGPT

In a database schema, the choice of primary key attributes is crucial for uniquely identifying each record (or row) in a table. In the context of the "time slot" relation, where each time slot can meet more than once in a week, the primary key attributes are chosen to ensure the uniqueness of each time slot entry while reflecting the real-world constraints and requirements.

In this scenario, the primary key of the "time slot" relation consists of the "day" and "start time" attributes. The reason for including these attributes in the primary key while excluding the "end time" is based on how time slots are scheduled and managed:

1. **Uniqueness and Differentiation:** The combination of "day" and "start time" helps to uniquely identify individual time slots in the context of a week. Different classes, meetings, or events may start at the same time on different days, or they might occur on the same day but at different times. By including both "day" and "start time" in the primary key, you ensure that two time slots cannot have the same day and start time combination within a week.
2. **Overlapping Time Slots:** Time slots for different events or classes can start at the same time on the same day. This is why "start time" is part of the primary key. If you only used "day" as the primary key, you wouldn't be able to differentiate between overlapping time slots that start at the same time on the same day.
3. **Flexibility of End Time:** The "end time" attribute is not included in the primary key because it can vary based on the duration of the event or class. Different events might have different durations and therefore different end times. Including the "end time" in the primary key would unnecessarily complicate the key and might not accurately reflect the real-world constraints.

In summary, the primary key consisting of "day" and "start time" in the "time slot" relation ensures that each time slot can be uniquely identified within a week, considering the potential for overlapping time slots and the flexibility of end times for different events. Excluding the "end time" from the primary key allows for better flexibility in managing varying durations of events while still maintaining the uniqueness of time slots.

2).

In the instance of instructor , no two instructors have the same name. From this, can we conclude that name can be used as a superkey (or primary key) of instructor?

ChatGPT

Yes, if it's guaranteed that no two instructors have the same name, you can conclude that the "name" attribute can be used as a superkey, and potentially even as a primary key, of the "instructor" relation.

3). What is the result of first performing the Cartesian product of student and advisor, and then performing a selection operation on the result with the predicate s id = ID? (Using the symbolic notation of relational algebra, this query can be written as σs id=ID(student × advisor).)

student (s\_id, s\_name, major)

advisor (s\_id, i\_id)

σ(s\_id = ID)(student × advisor)

4).

Consider the employee database of Figure 2.17. Give an expression in the relational algebra to express each of the following queries: a. Find the name of each employee who lives in city “Miami”. b. Find the name of each employee whose salary is greater than $100000. c. Find the name of each employee who lives in “Miami” and whose salary is greater than $100000.

π(name)(σ(city = "Miami")(employee))

π(name)(σ(salary > 100000)(employee))

π(name)(σ(city = "Miami" AND salary > 100000)(employee))

5).

. **Find the ID and name of each employee who does not work for "BigBank":**

sqlCopy code

π(ID, name)(employee - σ(company = "BigBank")(employee))

b. **Find the ID and name of each employee who earns at least as much as every employee in the database:**

π(ID, name)(σ(∀e ∈ employee : salary ≤ e.salary)(employee))

Explanation: Apply the selection (σ) operation with a universal quantification (∀) condition that checks if the salary of each employee is less than or equal to their own salary. This effectively filters out employees who earn less than any other employee. Then apply the projection (π) operation to select only the "ID" and "name" attribute.

6)>

# The division operator of relational algebra, “÷”, is defined as follows. Let r(R) and s(S) be relations, and let S ⊆ R; that is, every attribute of schema S is also in schema R. Given a tuple t, let t[S] denote the projection of tuple t on the attributes in S. Then r ÷ s is a relation on schema R − S (that is, on the schema containing all attributes of schema R that are not in schema S). A tuple t is in r ÷ s if and only if both of two conditions hold: • t is in ΠR−S(r) • For every tuple ts in s, there is a tuple tr in r satisfying both of the following: a. tr[S] = ts[S] b. tr[R − S] = t Given the above definition: a. Write a relational algebra expression using the division operator to find the IDs of all students who have taken all Comp. Sci. courses. (Hint: project takes to just ID and course id, and generate the set of all Comp. Sci. course ids using a select expression, before doing the division.) b. Show how to write the above query in relational algebra, without using division. (By doing so, you would have shown how to define the division operation using the other relational algebra operations.)

A). ΠID, course\_id(σdept\_name = "Comp. Sci."(takes)) ÷ student

b). πID, course\_id(σdept\_name = "Comp. Sci."(takes)) ⨝ student ⨝ (ΠID, COUNT(course\_id)(σdept\_name = "Comp. Sci."(takes)) ÷ πcourse\_id(σdept\_name = "Comp. Sci."(course)))

7).

Describe the differences in meaning between the terms relation and relation schema.

ChatGPT

The terms "relation" and "relation schema" are fundamental concepts in the field of relational databases. They refer to different aspects of the structure and organization of data within a relational database model. Here's a description of the differences in meaning between these terms:

1. **Relation:**
   * A relation, also known as a table, is a two-dimensional data structure in a relational database that consists of rows and columns.
   * Each row in a relation represents a record or a tuple, and each column represents an attribute or a field.
   * A relation is composed of data values organized in a grid-like structure, where each cell contains a single value.
   * Each relation must have a unique name within the database, which is used to refer to it and perform operations on it.
2. **Relation Schema:**
   * A relation schema defines the structure and attributes of a relation, specifying the names of the attributes, their data types, and any constraints.
   * It serves as a blueprint or template for creating instances of relations (actual tables) in the database.
   * The relation schema includes information about the attributes that make up the relation, along with their associated data types (e.g., integer, string) and optional constraints (e.g., primary key, foreign key).
   * The relation schema does not hold actual data; it provides metadata that describes how the data should be structured and organized.

In summary, a relation is a concrete representation of data organized into rows and columns, while a relation schema is an abstract description of the attributes, data types, and constraints that define how data is stored in a relation. The relation schema provides the necessary information to create and manage instances of relations within a relational database.

Top of Form

Regenerate

8).

Consider the advisor relation shown in the schema diagram in Figure 2.9, with s id as the primary key of advisor. Suppose a student can have more than one advisor. Then, would s id still be a primary key of the advisor relation? If not, what should the primary key of advisor be?

ChatGPT

If a student can have more than one advisor, then the attribute "s\_id" alone would not be a suitable primary key for the "advisor" relation. In a primary key, each value must be unique across all rows to uniquely identify each tuple (row) in the relation. If a student can have multiple advisors, there will be multiple rows in the "advisor" relation with the same "s\_id," leading to a violation of the uniqueness requirement of a primary key.

In this scenario, you would need to include additional attributes in the primary key to ensure uniqueness. One approach could be to use a composite primary key consisting of both "s\_id" and an "advisor\_id" (assuming that each advisor is assigned a unique identifier). This way, you can uniquely identify each advisor-student pair in the relation.

Here's how the primary key could be defined:

mathematicaCopy code

Primary Key: (s\_id, advisor\_id)

This composite primary key would ensure that each combination of student and advisor is unique within the "advisor" relation. It would allow for the possibility of a student having multiple advisors while still maintaining data integrity and uniqueness within the relation.

9).

Here are the expressions in relational algebra to express each of the given queries based on the "employee," "works," and "company" relations:

Given:

* **employee (person\_name, street, city)**
* **works (person\_name, company\_name, salary)**
* **company (company\_name, city)**

a. **Find the ID and name of each employee who works for "BigBank":**

sqlCopy code

π(person\_name)(σ(company\_name = "BigBank")(works))

b. **Find the ID, name, and city of residence of each employee who works for "BigBank":**

sqlCopy code

π(person\_name, city)(σ(company\_name = "BigBank")(works ⨝ employee))

c. **Find the ID, name, street address, and city of residence of each employee who works for "BigBank" and earns more than $10000:**

sqlCopy code

π(person\_name, street, city)(σ(company\_name = "BigBank" AND salary > 10000)(works ⨝ employee))

d. **Find the ID and name of each employee in this database who lives in the same city as the company for which she or he works:**

sqlCopy code

π(person\_name)(σ(employee.city = company.city)(employee ⨝ works ⨝ company)

10).

Given the relations and attributes you've provided, here are the relational algebra expressions for each of the queries:

a. **Find each loan number with a loan amount greater than $10000:**

sqlCopy code

π(loan\_number)(σ(amount > 10000)(loan))

b. **Find the ID of each depositor who has an account with a balance greater than $6000:**

sqlCopy code

π(ID)(σ(balance > 6000)(depositor ⨝ account))

c. **Find the ID of each depositor who has an account with a balance greater than $6000 at the "Uptown" branch:**

sqlCopy code

π(ID)(σ(balance > 6000 AND branch\_name = "Uptown")(depositor ⨝ account))

11). Why null value introduce in DB?

**Missing Data:**

**Incomplete or Optional Attributes:** In some cases, attributes in a database might be optional, meaning they don't need to have a value for every record. If a record doesn't have a value for an optional attribute, it can be represented using a null value. For instance, a "middle name" attribute in a customer database might be optional, and if a customer doesn't have a middle name, the attribute for that customer would have a null value.

**Data Transformations and Conversions:**

**12).**

Discuss the relative merits of imperative, functional, and declarative languages.

ChatGPT

Imperative, functional, and declarative programming languages represent different programming paradigms, each with its own approach to solving problems and building software. Each paradigm has its relative merits depending on the context and the nature of the task at hand. Let's discuss the relative merits of these three programming paradigms:

**1. Imperative Programming:** Imperative programming focuses on describing a sequence of steps that the computer should follow to achieve a specific goal. It's based on giving explicit instructions and modifying the program's state through statements and commands.

**Merits:**

* **Control:** Imperative languages offer fine-grained control over how computations are performed and how data is manipulated, which can be important for optimizing performance.
* **Readability:** In some cases, imperative code can be more intuitive to read and understand, especially for tasks that closely mimic real-world procedures.
* **State Management:** Imperative languages work well for tasks that involve managing mutable state, such as user interfaces or simulations.

**2. Functional Programming:** Functional programming treats computation as the evaluation of mathematical functions and avoids changing state and mutable data. It emphasizes immutability and focuses on composing functions to create complex behaviors.

**Merits:**

* **Predictability:** In functional programming, functions do not have side effects, making code easier to reason about and debug.
* **Concurrency:** Functional languages often make it easier to reason about concurrent and parallel programming, as they avoid mutable state that can lead to race conditions.
* **Modularity and Reusability:** Functional programming encourages the creation of small, reusable functions, which can lead to more modular and maintainable code.

**3. Declarative Programming:** Declarative programming focuses on describing the desired outcome rather than specifying the exact steps to achieve it. It abstracts away the control flow and state management.

**Merits:**

* **Expressiveness:** Declarative languages can often express complex operations concisely by abstracting away implementation details, making code more readable and less error-prone.
* **Optimization:** The compiler or interpreter of a declarative language can optimize code automatically based on high-level descriptions, potentially leading to improved performance.
* **Domain-Specific Languages (DSLs):** Declarative languages are often used to create domain-specific languages that closely match the problem domain, enabling non-programmers to express complex logic.

13).

Given the university schema, here are the queries expressed in relational algebra:

**University Schema Relations (Assumed Schema):**

* **instructor (ID, name, department\_name)**
* **department (department\_name, building)**
* **student (ID, name)**
* **course (course\_id, department\_name)**
* **takes (ID, course\_id, year)**

a. **Find the ID and name of each instructor in the Physics department:**

sqlCopy code

π(ID, name)(σ(department\_name = "Physics")(instructor))

b. **Find the ID and name of each instructor in a department located in the building "Watson":**

sqlCopy code

π(ID, name)(σ(building = "Watson")(instructor ⨝ department))

c. **Find the ID and name of each student who has taken at least one course in the "Comp. Sci." department:**

sqlCopy code

π(ID, name)(σ(department\_name = "Comp. Sci.")(student ⨝ takes ⨝ course))

d. **Find the ID and name of each student who has taken at least one course section in the year 2018:**

sqlCopy code

π(ID, name)(σ(year = 2018)(student ⨝ takes))

e. **Find the ID and name of each student who has not taken any course section in the year 2018:**

sqlCopy code

π(ID, name)((student) - π(ID, name)(σ(year = 2018)(student ⨝ takes)))

14).

branch(branch name, branch city, assets) customer (ID, customer name, customer street, customer city) loan (loan number, branch name, amount) borrower (ID, loan number) account (account number, branch name, balance) depositor (ID, account number)

ChatGPT

Based on the assumption that branch names and customer names uniquely identify branches and customers, but loans and accounts can be associated with more than one customer, let's determine the appropriate primary keys and foreign keys for the provided relations.

**Appropriate Primary Keys:**

* **branch**: The primary key could be the **branch\_name** attribute.
* **customer**: The primary key could be the **ID** attribute.
* **loan**: The primary key could be the **loan\_number** attribute.
* **borrower**: Since loans can be associated with more than one customer, the primary key for this relation would be a composite key of both **ID** and **loan\_number**.
* **account**: The primary key could be the **account\_number** attribute.
* **depositor**: Since accounts can be associated with more than one customer, the primary key for this relation would be a composite key of both **ID** and **account\_number**.

**Appropriate Foreign Keys:**

* **loan**: The foreign key in the **loan** relation could be the **branch\_name** attribute, referencing the **branch** relation's **branch\_name** attribute.
* **borrower**: The foreign keys in the **borrower** relation could be both the **ID** and **loan\_number** attributes, referencing the **customer** and **loan** relations respectively.
* **account**: The foreign key in the **account** relation could be the **branch\_name** attribute, referencing the **branch** relation's **branch\_name** attribute.
* **depositor**: The foreign keys in the **depositor** relation could be both the **ID** and **account\_number** attributes, referencing the **customer** and **account** relations respectively.

In summary, the primary keys are determined based on unique identification criteria, and the foreign keys establish relationships between the tables by referencing the primary keys of other tables.

Sql notes;

1).  SQL is based on set and relational operations with certain modifications and enhancements

 This query is equivalent to the relational algebra expression.

A1, A2, ..., An(P (r1 x r2 x ... x rm))

 The result of an SQL query is a relation.

2).

 SQL allows duplicates in relations as well as in query results.

Can rename “salary/12” using the as clause:

select ID, name, salary/12 as monthly\_salary

3). operators <, <=, >, >=, =, and <>.

 The where clause specifies conditions that the result must satisfy.

The from clause lists the relations involved in the query.

 Cartesian product not very useful directly, but useful combined with where-

clause condition (selection operation in relational algebra).

4).

percent ( % ). The % character matches any substring.

 underscore ( \_ ). The \_ character matches any character.

5).

SQL supports a variety of string operations such as

 concatenation (using “||”)

 converting from upper to lower case (and vice versa)

 finding string length, extracting substrings, etc.

6).

Each of the above operations automatically eliminates duplicates

 To retain all duplicates use the

 union all,

 intersect all

 except all.

7).

 /\* erroneous query \*/

select dept\_name, ID, avg (salary)

from instructor

group by dept\_name;

8). Note: predicates in the having clause are applied after the formation of groups

whereas predicates in the where clause are applied before forming groups

9). Find courses offered in Fall 2017 and in Spring 2018.

Find the total number of (distinct) students who have taken course sections taught

by the instructor with ID 10101

10). The with clause provides a way of defining a temporary relation whose definition

is available only to the query in which the with clause occurs.

11). To avoid the danger of equating attributes erroneously, we can use the “using”

construct that allows us to specify exactly which columns should be equated.

12). Any relation that is not of the conceptual model but is made visible to a user as a

“virtual relation” is called a view.

13).

A relationship set is a mathematical relation among n ≥ 2 entities, each taken from

entity sets

{(e1, e2, ... en) | e1 ∈ E1, e2 ∈ E2, ..., en ∈ En}