DBMS

- DBMS, schema, catalog, metadata, data independence, pre-compiler

1. DBMS (Database Management System)

- **Definition:** Software that allows users to define, create, maintain, and control access to the database.
- **Examples:** MySQL, PostgreSQL, Oracle, SQL Server.
- **Role:** It sits between **You** (or your App) and the **Raw Data Files**. You speak to the DBMS (using SQL), and the DBMS speaks to the hard drive.

2. Schema (The Blueprint)

- **Concept:** The *description* of the database structure. It does not contain the actual data (the rows), just the rules.
- **Analogy:** The **Floor Plan** of the library. It says "Fiction is in Section A," but it doesn't list every single book title.
- **Three Schema Architecture:**
    - **External Schema (View Level):** What the **User** sees (e.g., a "Profile Page").
    - **Conceptual Schema (Logical Level):** The whole design (Tables, Columns, Relationships).
    - **Internal Schema (Physical Level):** How data is physically stored on the disk (Indexes, Block sizes).

3. Metadata & Data Dictionary (The Catalog)

- **Metadata:** "Data about Data."
    - *Example:* The data is "John Smith". The *metadata* is "Name: String, Max Length 50, Cannot be Null."
- **System Catalog (Data Dictionary):** A mini-database *inside* the main database that stores all the metadata. It tells the DBMS where everything is.
    - *Analogy:* The **Card Catalog** in the library. It tells you that "Harry Potter" exists and is located on Shelf 4, Row 2.

4. Data Independence (The Separation)

The ability to change the database structure at one level *without* breaking the levels above it.

- **Physical Data Independence:** You can change the **Hard Drive** (move from HDD to SSD, or change file organization) without changing the **Tables** (Logical Schema). The user doesn't know or care that the files moved.
- **Logical Data Independence:** You can change the **Tables** (add a new column like "Phone Number") without breaking the **Application** (External View). The app just ignores the new column until it's updated.

5. Pre-Compiler (The Translator)

Sometimes, programmers write SQL code inside other languages like C++ or Java (Host Languages). The regular C++ compiler doesn't understand SQL.

- **The Process:**
    - **Pre-Compiler:** Scans the code, pulls out the SQL commands, and translates them into DBMS function calls.
    - **Host Compiler:** Compiles the rest of the C++ code normally.

1. What is the primary function of a DBMS?

A. To provide an interface between the user and the database, managing data storage and retrieval.

✓ That's right!
It handles all the heavy lifting of data management so the user doesn't have to deal with raw files.

B. To design websites.

C. To protect the computer from viruses.

D. To compile Java code.

2. What is 'Metadata'?

A. Encrypted data.

B. Huge data (Big Data).

C. The actual data stored in the database (e.g., 'John Doe').

D. Data about data (e.g., 'Column Name = Name, Type = String').

✓ That's right!
It describes the structure and constraints of the data.

3. Where is Metadata stored in a DBMS?

A. In the RAM only.

B. In the Operating System logs.

C. In the System Catalog (Data Dictionary).

✓ That's right!
The Catalog is the repository for all schema definitions.

D. It is not stored.

4. What is 'Physical Data Independence'?

A. The ability to change the Physical Schema (Storage) without changing the Logical Schema.

✓ That's right!
You can defragment the disk or move files to a new drive without breaking the database tables.

B. Using wireless data.

C. The ability to change the Logical Schema (Tables) without changing the External View.

D. Storing data in the cloud.

5. What is a 'Database Schema'?

A. The user interface.

B. The actual content of the database at a specific moment.

✕ Not quite

That is a Database Instance (or State).

C. The hard drive.

D. The logical structure or blueprint of the database.

✓ Right answer

It defines the tables, fields, and relationships. It rarely changes.

6. What is the role of a Pre-compiler (DML Pre-processor)?

A. To speed up the internet.

B. To back up the data.

C. To design the database.

D. To convert SQL statements embedded in a host language (like C++) into executable code for the DBMS.

✓ That's right!

It extracts the SQL and prepares it so the main compiler doesn't get confused.

7. **Which level of the Three-Schema Architecture is closest to the physical storage?**

A. View Level

B. External Level

✕ Not quite

Closest to the User.

C. Internal Level

✓ Right answer

This level deals with data compression, encryption, and block storage on the disk.

8. **Why is Logical Data Independence harder to achieve than Physical Data Independence?**

A. Because of metadata.

B. Because hardware is cheap.

C. It isn't; it is easier.

D. Because applications rely heavily on the specific structure of tables (column names, types).

✓ That's right!

If you delete a column that an app expects, the app crashes. Moving a file on disk (Physical) is invisible to the app.

- Users: naïve, sophisticated, casual

Different people interact with the kitchen in different ways.

1. Naïve (Parametric) Users 😅

- **The Customer:** They don't know how to cook. They don't even know what the kitchen looks like. They just sit at a table, look at a **Menu** (The Application Interface), and point to "Burger."
- **In DBMS:** These are the vast majority of users (99%). They have **zero** technical knowledge.
- **How they interact:** They use **"Canned Transactions"** (pre-written programs with buttons and forms).
- **Examples:**
    - A Bank Teller checking your balance.
    - You booking a flight ticket on a website.
    - A supermarket cashier scanning barcodes.

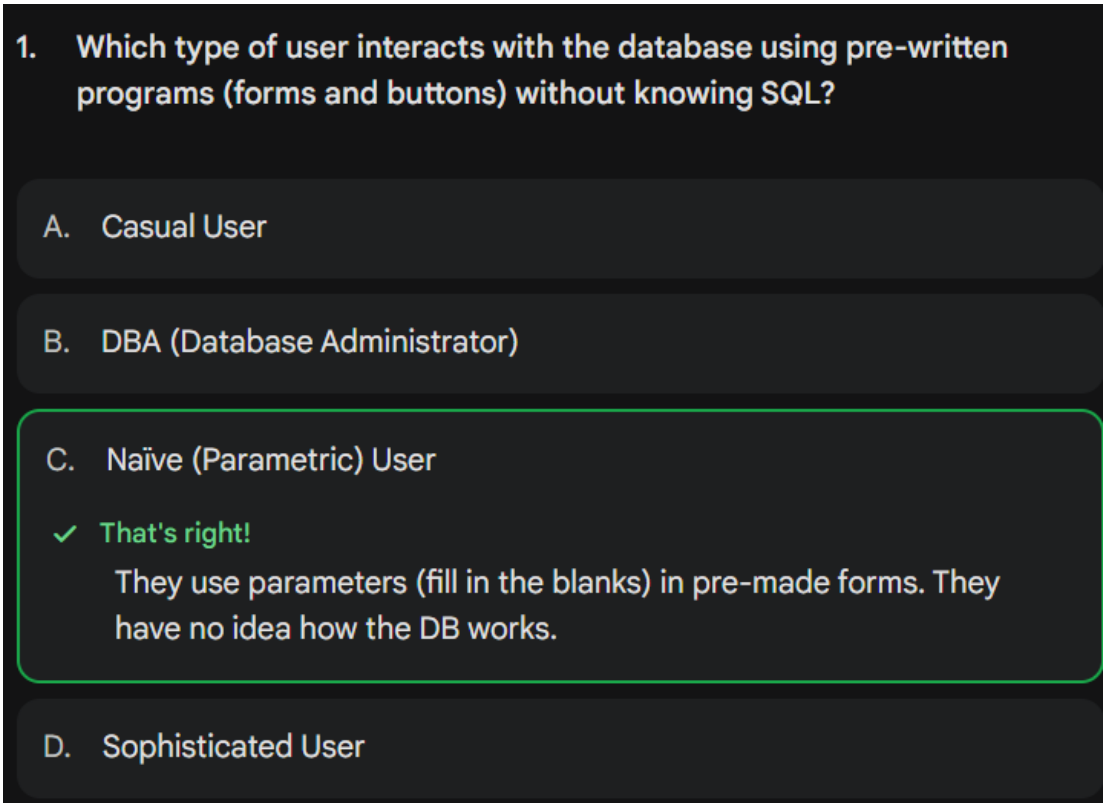
2. Sophisticated Users 🧐

- **The Food Critic / Chef:** They know exactly how food is made. They walk into the kitchen and say, "I want a medium-rare steak with a dash of thyme and garlic butter." They don't need a menu; they create their own orders.
- **In DBMS:** These are **Engineers, Scientists, and Analysts**. They understand the database structure and SQL.
- **How they interact:** They write **SQL Queries** directly to the database to get specific, complex answers. They don't use the simple buttons the Naïve users use.
- **Examples:**
    - A Data Analyst searching for "Sales trends in July for users under 30."
    - A Software Developer debugging the system.

3. Casual Users 🚶

- **The Restaurant Owner:** They don't cook, and they don't eat there every day. But once in a while, they walk in and ask, "How many burgers did we sell last month?" because they need that specific info right now.
- **In DBMS:** These are **Managers or Middle-Level Executives**. They access the database occasionally (not daily) but need different information each time.

- **How they interact:** They usually use high-level query interfaces that let them ask questions without writing complex code, but they are more flexible than Naïve users.
- **Examples:**
    - A Manager checking a quarterly report.



2. A Data Scientist writing complex SQL queries to analyze trends is an example of which user type?

A. Casual User

B. Sophisticated User

✓ That's right!
They have the technical sophistication to speak the database's language (SQL) directly.

C. Parametric User

D. Naïve User

3. Which user accesses the database only occasionally, usually to get new information each time?

A. Naïve User

B. DBA

C. Sophisticated User

D. Casual User

✓ That's right!
They check in 'casually' when they need specific info.

4. What is a 'Canned Transaction'?

A. A database virus.

B. A standard, pre-written SQL query wrapped in an easy-to-use interface.

✓ That's right!
Used by Naïve users (e.g., 'Withdraw Cash' button at an ATM).

C. A deleted file.

D. A backup.

5. Who defines the 'Canned Transactions' that Naïve users interact with?

A. The Operating System

B. The Naïve User

C. The Application Programmer / Sophisticated User

✓ That's right!
Programmers write the code for the ATM interface that the Naïve users then click on.

D. The Casual User

- ER model: entity, attributes, structural constraints

Imagine you are an **Architect** designing a house. You don't start by laying bricks (Coding). You start by drawing a **Blueprint** (ER Diagram).

- **The ER Model (Entity-Relationship Model)** is that blueprint. It describes *what* data we need to store and *how* it connects.

1. Entity (The Noun)

- **Definition:** A real-world object or concept that can be distinctly identified.
- **Symbol:** Represented by a **Rectangle**.
- **Examples:**
  - **Tangible:** `Student`, `Car`, `Employee`.
  - **Intangible:** `Course`, `Project`, `Account`.
- **Entity Set:** A collection of similar entities (e.g., All students in a university).

2. Attributes (The Adjectives)

- **Definition:** The properties or characteristics that describe an Entity.
- **Symbol:** Represented by an **Oval (Ellipse)**.
- **Types:**
  - **Simple Attribute:** Cannot be divided further (e.g., Age, `Gender`).
  - **Composite Attribute:** Can be broken down (e.g., `Address` -> `Street`, `City`, `Zip`).
  - **Multi-valued Attribute:** An entity can have multiple values for this (e.g., Phone `Number` - you might have 2 or 3). **Symbol:** Double Oval.
  - **Derived Attribute:** Not stored directly but calculated from other attributes (e.g., Age is derived from `Date of Birth`). **Symbol:** Dashed Oval.
  - **Key Attribute:** Unique identifier (e.g., `Student ID`, SSN). **Symbol:** Oval with text underlined.

3. Relationship (The Verb)

- **Definition:** How two entities interact.
- **Symbol:** Represented by a **Diamond**.

- **Example:** `Student` **Enrolls in** `Course`.

4. Structural Constraints (The Rules) 📏

These rules define *how many* times an entity can participate in a relationship.

A. Cardinality Ratio (Maximums)

- **One-to-One (1:1):** One Manager manages One Department.
- **One-to-Many (1:N):** One Customer places Many Orders.
- **Many-to-One (N:1):** Many Students belong to One Department.
- **Many-to-Many (M:N):** Many Students enroll in Many Courses.

B. Participation Constraint (Minimums)

- **Total Participation (Existence Dependency):** Every entity *must* be involved.
  - *Example:* Every `Employee` *must* work for a `Department`.
  - **Symbol:** Double Line.
- **Partial Participation:** Not every entity is involved.
  - *Example:* Not every `Employee` manages a `Department`.
  - **Symbol:** Single Line.

1. What shape is used to represent an Entity in an ER Diagram?

A. Diamond

B. Triangle

C. Rectangle

✓ That's right!
Standard notation for Entities.

D. Oval

2. Which type of attribute can have multiple values for a single entity (e.g., Skills: Java, Python, C++)?

A. Multi-valued Attribute

✓ That's right!
Represented by a double oval.

B. Simple Attribute

C. Derived Attribute

D. Composite Attribute

3. An attribute that is NOT stored in the database but calculated from another attribute (e.g., Age from DOB) is called:

A. Composite Attribute

B. Multi-valued Attribute

C. Derived Attribute

✓ That's right!
It is derived on the fly.

D. Key Attribute

4. What does a 'Double Line' between an Entity and a Relationship signify?

A. Many-to-Many

B. One-to-Many

C. Partial Participation

D. Total Participation (Mandatory)

✓ That's right!
It means every single instance of that entity MUST be involved in the relationship.

5. Which attribute uniquely identifies each entity in an entity set (e.g., Student ID)?

A. Composite Attribute

B. Simple Attribute

C. Key Attribute (Primary Key)

✓ That's right!
Underlined in the diagram.

D. Derived Attribute

6. If one Student can take many Courses, and one Course can have many Students, what is the Cardinality Ratio?

A. One-to-One (1:1)

B. Many-to-Many (M:N)

✓ Right answer
Many on both sides.

C. Total Participation

D. One-to-Many (1:N)

✕ Not quite
No.

7. An attribute that can be subdivided into smaller parts (e.g., Name -> First Name, Last Name) is called:

A. Composite Attribute

✓ That's right!
Composed of parts.

B. Derived Attribute

C. Key Attribute

D. Simple Attribute

8. In an ER diagram, what shape represents a Relationship?

A. Diamond

✓ That's right!
Connects two entities.

B. Oval

C. Rectangle

D. Triangle

- Relational model: constraints, relational algebra operations

The **Relational Model** is just a set of rules for keeping these tables clean, organized, and error-free.

1. Relational Constraints (The Rules)

Constraints are laws that the data *must* obey. If you try to break them, the DBMS rejects your data.

A. Domain Constraint (The Data Type Rule)

- **Rule:** Every value in a column must be of the same type and "atomic" (cannot be broken down).
- **Example:** If the "Age" column is defined as an Integer, you cannot type "Twenty-Five" (String) or "25, 26" (Multiple values).

B. Key Constraint (The Uniqueness Rule)

- **Rule:** Every table must have a **Primary Key** (a unique ID) that identifies each row distinctively.
- **Example:** You cannot have two students with the same `Student_ID`.

C. Entity Integrity Constraint (The "No-Null" Rule)

- **Rule:** The **Primary Key cannot be NULL** (Empty).
- **Reasoning:** If the ID is empty, how can we identify the row? It's like a book with no title.

D. Referential Integrity Constraint (The Relationship Rule)

- **Rule:** A **Foreign Key** in one table must point to a valid **Primary Key** in another table.
- **Example:** If `Student_Table` has a `Course_ID` column, you cannot enter Course ID 999 if that course doesn't exist in the `Course_Table`. No orphans allowed!

2. Relational Algebra (The Math)

This is the "procedural" language used to manipulate tables. It tells the DBMS *how* to get the data.

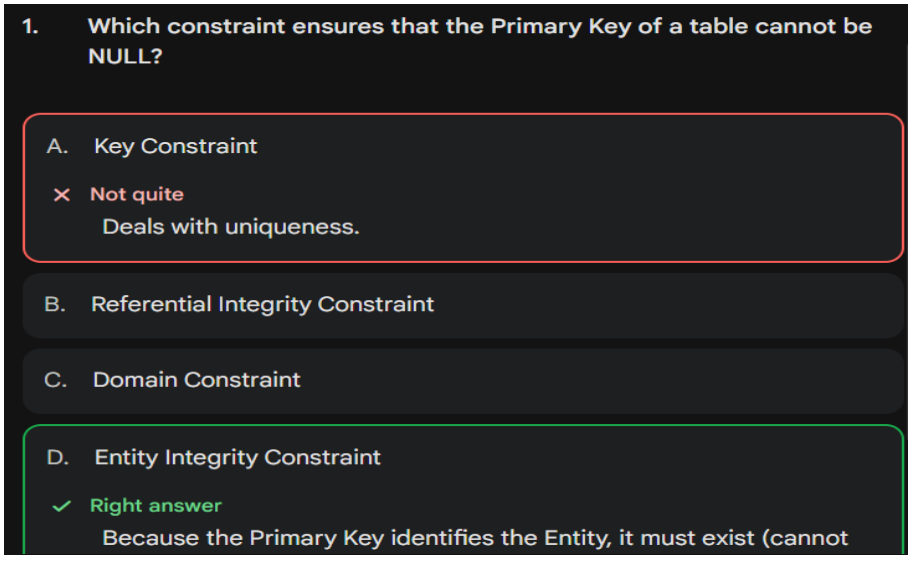
A. Unary Operations (Work on One Table)

- **Select (σ):** Filters **Rows** based on a condition.

    - *Symbol:* Sigma (σ)
    - *Analogy:* Using a Filter in Excel. "Show me only students with GPA > 3.5".
- **Project (π):** Filters **Columns**.
    - *Symbol:* Pi (π)
    - *Analogy:* Hiding columns in Excel. "Show me only the Names, hide the Phone Numbers."
- **Rename (ρ):** Changes the name of a table or column temporarily.

B. Binary Operations (Work on Two Tables)

- **Union ($\cup$):** Combines rows from Table A and Table B. (Must have same columns).
    - *Example:* "List all students in Math **OR** Science."
- **Intersection ($\cap$):** Finds common rows.
    - *Example:* "List students in Math **AND** Science."
- **Set Difference ($-$):** Rows in A that are NOT in B.
    - *Example:* "List students in Math **BUT NOT** in Science."
- **Cartesian Product ($\times$):** Combines **every** row of A with **every** row of B.
    - *Result:* If A has 5 rows and B has 4 rows, result has $5 \times 4 = 20$ rows. It's usually useless on its own.
- **Join ($\bowtie$):** A smarter Cartesian Product. It combines tables only where the data matches (e.g., matching `Student_ID`).



2. The 'Select' operation (σ) filters data by:

A. Databases

✕ Not quite

No.

B. Columns (Vertical)

C. Rows (Horizontal)

✓ Right answer

Select picks specific rows that meet a condition (e.g., Salary > 5000).

3. If Table R has 10 rows and Table S has 5 rows, how many rows will the Cartesian Product ($R \times S$) have?

A. 5 (Min)

B. 15 (10 + 5)

C. 10 (Max)

D. 50 (10 × 5)

✓ That's right!

Cartesian Product multiplies the row counts. Every row in R connects to every row in S.

4. Which operation is denoted by the Pi symbol (π)?

A. Project

✓ That's right!
Project filters columns.

B. Select

C. Join

D. Product

5. The Referential Integrity Constraint prevents:

A. Wrong data types.

B. Duplicate Primary Keys.

C. Invalid Foreign Keys (Orphans).

✓ That's right!
It ensures you cannot point to a record that doesn't exist.

D. Null values in Primary Keys.

6. Which operation finds rows that are in Table A but NOT in Table B?

A. Union (∪)

B. Intersection (∩)

C. Join (⋈)

D. Set Difference (−)

✓ That's right!

A minus B removes anything in A that is also found in B.

7. For Union (∪) and Intersection (∩) to work, the two tables must be:

A. Connected by a Foreign Key

✗ Not quite

No.

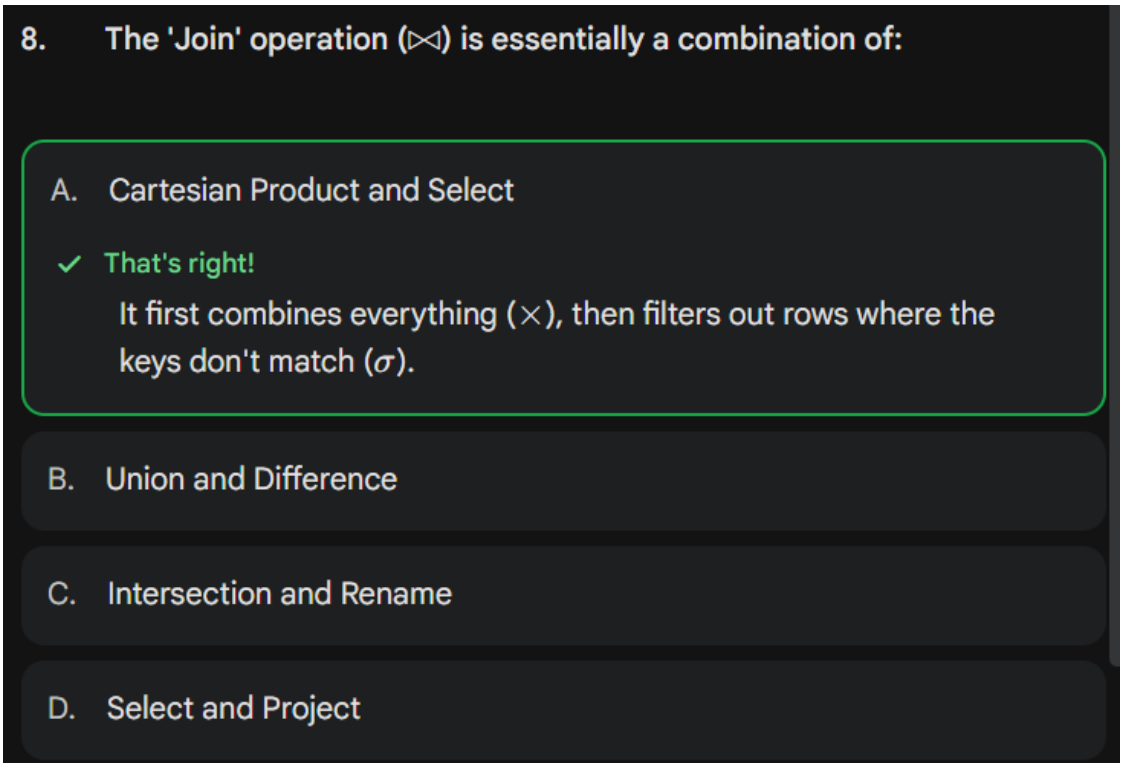
B. Sorted

C. The same size (rows)

D. Union Compatible

✓ Right answer

This means they must have the same number of attributes (columns) and compatible domains (data types). You can't Union a



- SQL: DDL, DML, TCL, DCL commands

Imagine you are **Building and Managing a House**.

- **DDL** is **Architecture**. (Building walls, adding rooms, destroying the house).
- **DML** is **Interior Design**. (Moving furniture, painting walls, throwing out trash).
- **DCL** is **Security**. (Giving keys to people).
- **TCL** is **Time Travel**. (Saving your game or undoing a mistake).

1. DDL (Data Definition Language) 🏗️

These commands define the **Structure** (Schema) of the database. They deal with the "Container," not the data inside.

- **Key Characteristic:** Auto-committed (You cannot undo them!).
- **Commands:**
  - `CREATE`: Builds a new table or database.
  - `ALTER`: Modifies the structure (e.g., add a new column).
  - `DROP`: Destroys the table **and** its structure completely (The Nuclear Option).

- `TRUNCATE`: Removes all data but keeps the structure (Like emptying a trash can).

2. DML (Data Manipulation Language)

These commands deal with the **Data** inside the tables.

- **Key Characteristic:** Not auto-committed (You can undo them using Rollback).
- **Commands:**
  - `INSERT`: Adds new rows.
  - `UPDATE`: Modifies existing data.
  - `DELETE`: Removes specific rows (can be undone).
  - `SELECT`: Retrieves data. *(Note: Sometimes called DQL - Data Query Language)*.

3. DCL (Data Control Language)

These commands deal with **Permissions** and security.

- **Commands:**
  - `GRANT`: Gives a user permission to access/modify data.
  - `REVOKE`: Takes away permission.

4. TCL (Transaction Control Language)

These commands manage **Transactions** (groups of DML commands).

- **Commands:**
  - `COMMIT`: Saves all changes permanently.
  - `ROLLBACK`: Undoes changes (Time travels back to the last Commit).
  - `SAVEPOINT`: Creates a checkpoint you can roll back to later.

1. Which command is used to remove a table completely from the database (Structure + Data)?

A. TRUNCATE

B. DROP

✓ That's right!
DROP removes the object itself. It's gone forever.

C. REMOVE

D. DELETE

2. Which of the following is a DML (Data Manipulation) command?

A. UPDATE

✓ That's right!
UPDATE modifies the actual data inside the rows.

B. CREATE

C. GRANT

D. ALTER

3. What is the difference between DELETE and TRUNCATE?

A. TRUNCATE deletes the table structure.

B. DELETE can be rolled back (DML); TRUNCATE cannot (DDL).

✓ That's right!

TRUNCATE is faster because it resets the table structure without logging individual row deletions.

C. They are the same.

D. DELETE is DDL, TRUNCATE is DML.

4. Which TCL command saves your changes permanently to the database?

A. GRANT

B. COMMIT

✓ Right answer

Commit makes the transaction permanent.

C. ROLLBACK

D. SAVEPOINT

✕ Not quite

Creates a checkpoint, doesn't finalize

5. The 'ALTER' command is part of which category?

A. TCL

B. DCL

C. DDL (Data Definition Language)

✓ That's right!
It modifies the definition/structure of the table (e.g., adding a column).

D. DML

6. Which command allows you to undo changes made since the last Commit?

A. BACKUP

B. ROLLBACK

✓ That's right!
Reverts the database state.

C. REVOKE

D. RETURN

7. Which category deals with permissions (GRANT/REVOKE)?

A. TCL

B. DDL

C. DML

D. DCL (Data Control Language)

✓ That's right!

It controls access.

8. If you want to add a new column 'Age' to an existing 'Students' table, which command do you use?

A. CREATE

B. INSERT

C. ALTER

✓ Right answer

ALTER TABLE Students ADD Age INT;

D. UPDATE

✗ Not quite

Updates data, not columns.

- Basic queries and top N queries

Imagine you have a **Spreadsheet** called `Students` with columns: `Name`, `Age`, `Grade`.

1. Basic Queries (The Questions) ❓

- **SELECT:** "Which columns do you want to see?"
  - `SELECT Name, Age` (Show me only Names and Ages).
  - `SELECT *` (Show me everything).
- **FROM:** "Which table are we looking at?"
  - `FROM Students`
- **WHERE:** "Filter the rows."
  - `WHERE Age > 18` (Only adults).
  - `WHERE Name = 'Alice'` (Only Alice).
- **ORDER BY:** "Sort the results."
  - `ORDER BY Age ASC` (Youngest to Oldest).
  - `ORDER BY Age DESC` (Oldest to Youngest).
- **GROUP BY:** "Group identical data."
  - `GROUP BY Grade` (Put all 'A' students in one bucket, 'B' in another). often used with **Aggregate Functions** like `COUNT()`, `AVG()`, `SUM()`.

Example:

SQL

```
SELECT Grade, COUNT(*)
FROM Students
WHERE Age > 18
GROUP BY Grade
ORDER BY Grade DESC;
```

(Translation: "Find all adult students, group them by grade, count how many are in each grade, and show me the grades from A to F.")

2. Top N Queries (The Leaderboard) 🏆

Sometimes you don't want *all* the results. You only want the **Top 5** or **Bottom 10**.

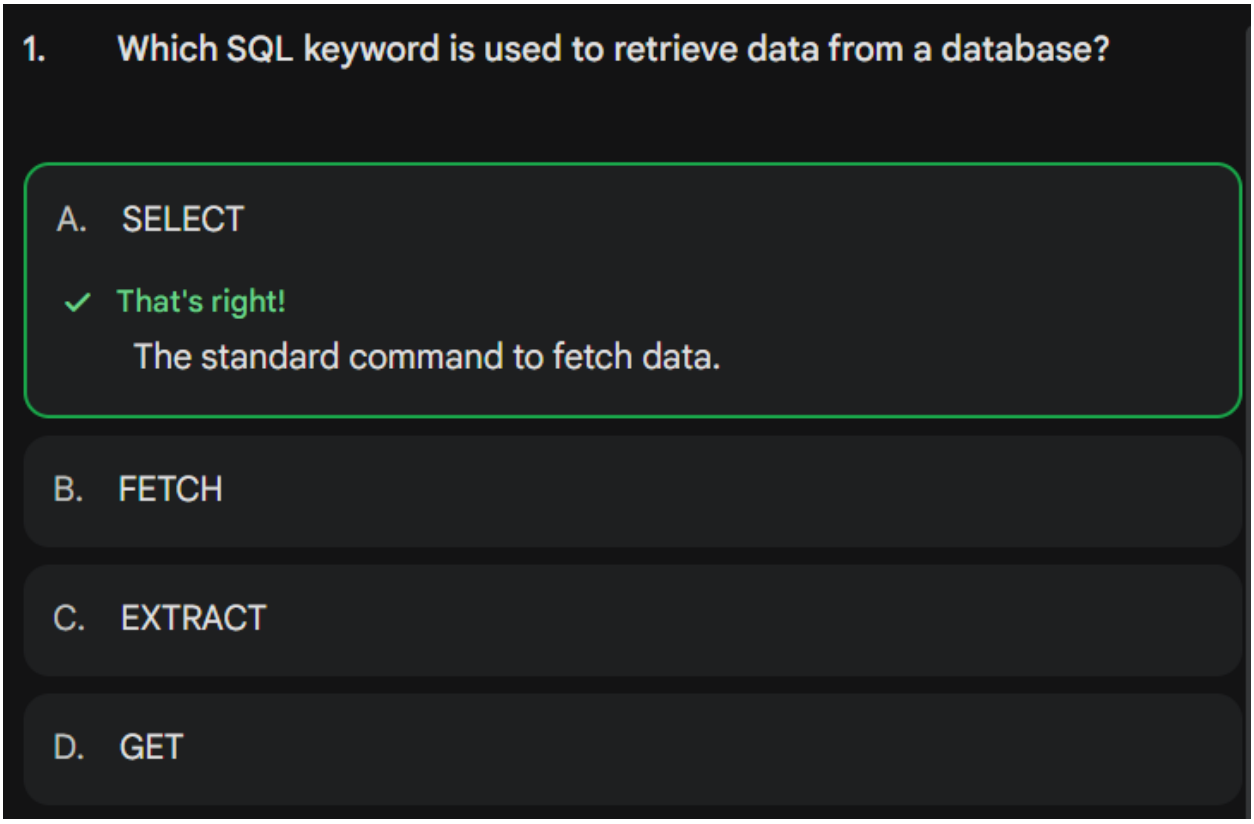
- **Why use it?** To find the "Highest Salary," "Most Recent Order," or "Best Student."
- **Syntax varies by Database:**

- **MySQL / PostgreSQL:** `LIMIT N`
- **SQL Server:** `TOP N`
- **Oracle:** `FETCH FIRST N ROWS ONLY`

Example (MySQL):

```
SQL
SELECT Name, Grade
FROM Students
ORDER BY Grade DESC
LIMIT 3;
```

(Translation: "Show me the top 3 students with the highest grades.")



2. Which clause is used to filter records?

A. LIMIT

B. GROUP BY

C. WHERE

✓ That's right!

It restricts the output to rows that meet a condition.

D. ORDER BY

3. In MySQL, how do you select the top 5 records from a table?

A. SELECT TOP 5 * FROM Table

B. SELECT * FROM Table LIMIT 5

✓ That's right!

This is standard MySQL/PostgreSQL syntax.

C. SELECT FIRST 5 * FROM Table

D. SELECT * FROM Table WHERE ROWNUM <= 5

4. Which keyword sorts the result in descending order?

A. SORT

B. ASC

C. ORDER

D. DESC

✓ That's right!
Descending (Z-A, 10-1).

5. What is the result of `SELECT COUNT(*) FROM Students` ?

A. It returns the sum of ages.

B. It deletes all students.

C. It returns the total number of rows (students) in the table.

✓ That's right!
COUNT(*) counts rows.

D. It returns the names of all students.

6. The `GROUP BY` statement is often used with which functions?

A. String functions (LEN, SUBSTR)

B. Date functions

C. None

D. Aggregate functions (COUNT, MAX, MIN, SUM, AVG)

✓ That's right!

You group rows to perform calculations on each group (e.g., Average salary per department).

7. How do you select all columns from a table named 'Users'?

A. SELECT ALL FROM Users

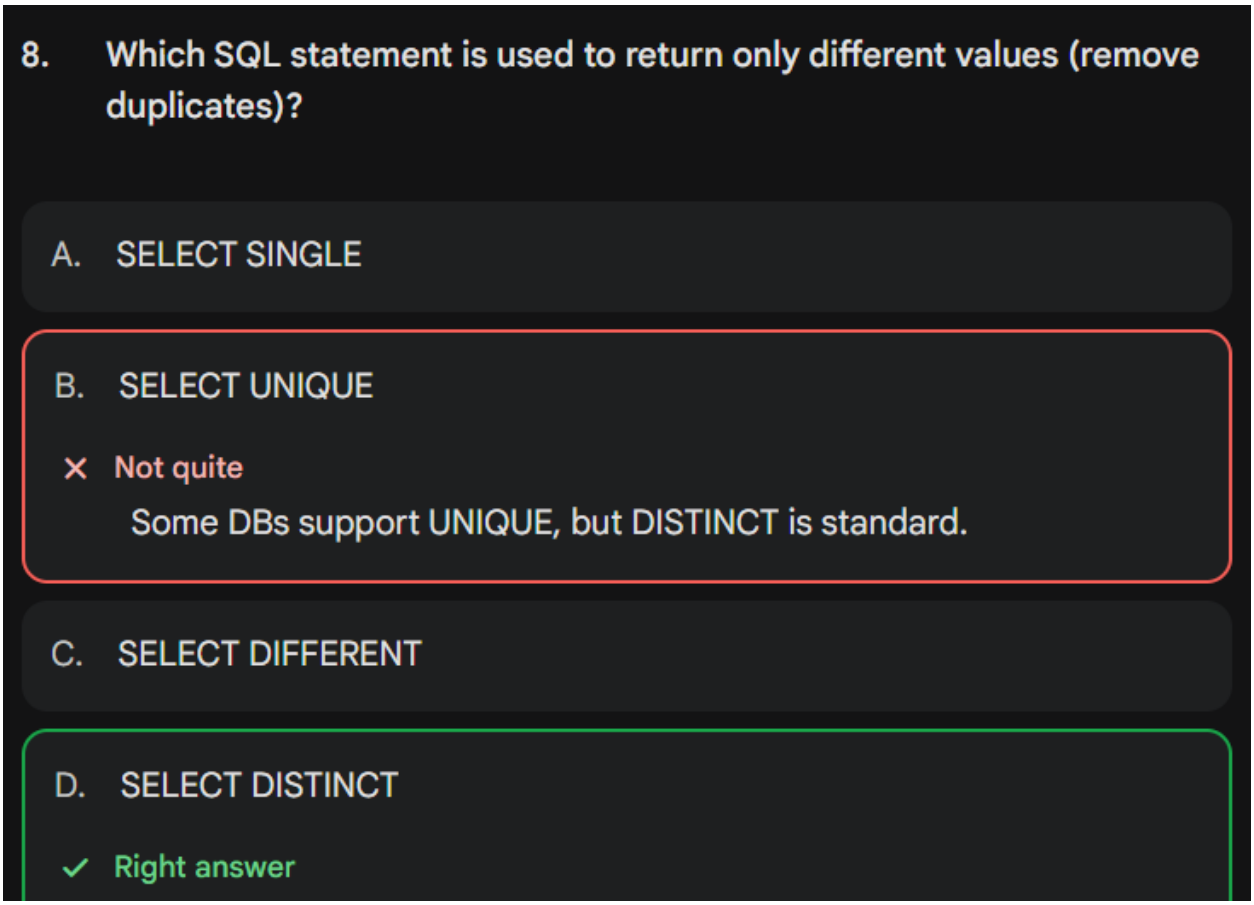
B. SELECT * FROM Users

✓ That's right!

The asterisk (*) is the wildcard for 'all columns'.

C. GET Users

D. SELECT Users



- Normalization: properties, 1NF, 2NF, 3NF, BCNF

Imagine a **Spreadsheet** where you store Student Grades.

- **Row 1:** `John Smith | Math, Science | Mr. A, Mrs. B` This is a mess. "Math, Science" is two things in one box. If you want to change Mr. A's name, you have to find every row where he appears.

Normalization is the process of organizing data to:

1. **Minimize Redundancy:** Store each fact only once.
2. **Avoid Anomalies:** Prevent errors when adding, updating, or deleting data.

1. The Anomalies (The Problems) ⚠

- **Insertion Anomaly:** You can't add a new Course unless you have a Student to enroll in it. (Because the Student ID is the key).

- **Deletion Anomaly:** If the only student in "Biology" drops the class, you accidentally delete the "Biology" course from the database entirely.
- **Update Anomaly:** If Mr. Smith changes his address, you have to update 100 rows. If you miss one, the data becomes inconsistent.

2. The Normal Forms (The Levels of Cleanliness) 🧹

A. First Normal Form (1NF) - The "Atomic" Rule

- **Rule:** Every cell must contain only **one single value**. No lists, arrays, or comma-separated values.
- **Fix:** Split the lists into new rows.
    - *Bad:* `John | Math, Science`
    - *Good:*`John | MathJohn | Science`

B. Second Normal Form (2NF) - The "Whole Key" Rule

- **Prerequisite:** Must be in 1NF.
- **Rule:** No **Partial Dependency**. All non-key attributes must depend on the **entire** Primary Key, not just part of it.
- **Scenario:** Key is `{Student_ID, Course_ID}`.
    - *Bad:* Storing `Course_Fee` in this table. `Course_Fee` depends only on `Course_ID`, not on `Student_ID`.
    - *Fix:* Move `Course_Fee` to a separate `Course` table.

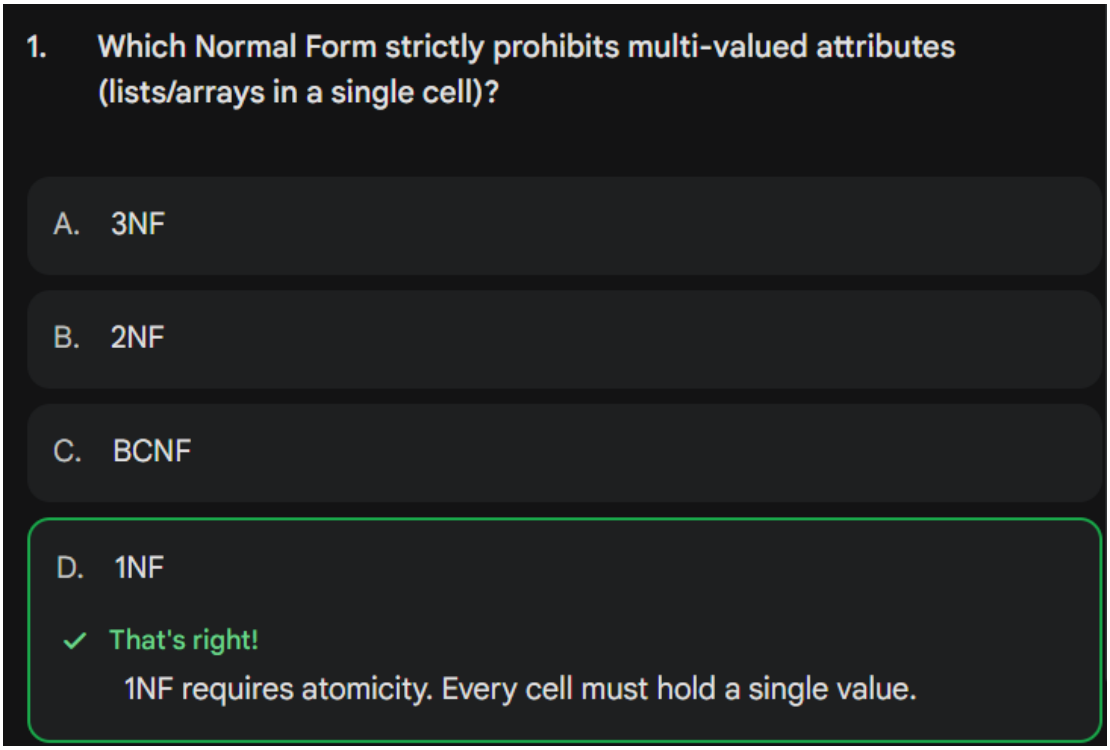
C. Third Normal Form (3NF) - The "Direct" Rule

- **Prerequisite:** Must be in 2NF.
- **Rule:** No **Transitive Dependency**. Attributes must depend **only** on the Key, not on other non-key attributes. ("The Key, the whole Key, and nothing but the Key").
- **Scenario:** Table has `{Student_ID, Zip_Code, City}`.
    - *Bad:* `City` depends on `Zip_Code`. `Zip_Code` depends on `Student_ID`. This is a chain (Transitive).
    - *Fix:* Move `{Zip_Code, City}` to a separate `Zip_Codes` table.

D. Boyce-Codd Normal Form (BCNF) - The "Super Key" Rule

- **Prerequisite:** Must be in 3NF.
- **Rule:** For every dependency $X \rightarrow Y$, **X must be a Super Key**.

- **Scenario:** A stricter version of 3NF that handles edge cases where a non-key attribute determines part of a composite key.



2. A table is in 2NF if it is in 1NF and contains no:

A. Transitive Dependencies

B. Partial Dependencies

✓ Right answer

Attributes must depend on the Whole Key, not just a part of it.

C. Multi-valued attributes

D. Join Dependencies

✗ Not quite

That is 5NF.

3. Which anomaly occurs when deleting a row inadvertently causes the loss of other valuable information?

A. Redundancy

B. Deletion Anomaly

✓ That's right!

Deleting the last student in 'Physics' also deletes the fact that 'Physics' exists as a course.

C. Insertion Anomaly

D. Update Anomaly

4. To achieve 3NF, a table must be in 2NF and remove all:

A. Repeating Groups

B. Primary Keys

C. Partial Dependencies

D. Transitive Dependencies

✓ That's right!

Non-key attributes depending on other non-key attributes (e.g., City depends on Zip Code).

5. BCNF (Boyce-Codd Normal Form) is a stricter version of which form?

A. 4NF

B. 3NF

✓ That's right!

BCNF handles cases where 3NF fails (often involving overlapping composite keys).

C. 1NF

D. 2NF

6. What is the primary goal of Normalization?

A. To make queries faster.

B. To increase redundancy.

C. To reduce data redundancy and improve data integrity.

✓ That's right!
Store it once, store it right.

D. To encrypt data.

7. Which property ensures that decomposing a table into smaller tables does not lose any information when joined back together?

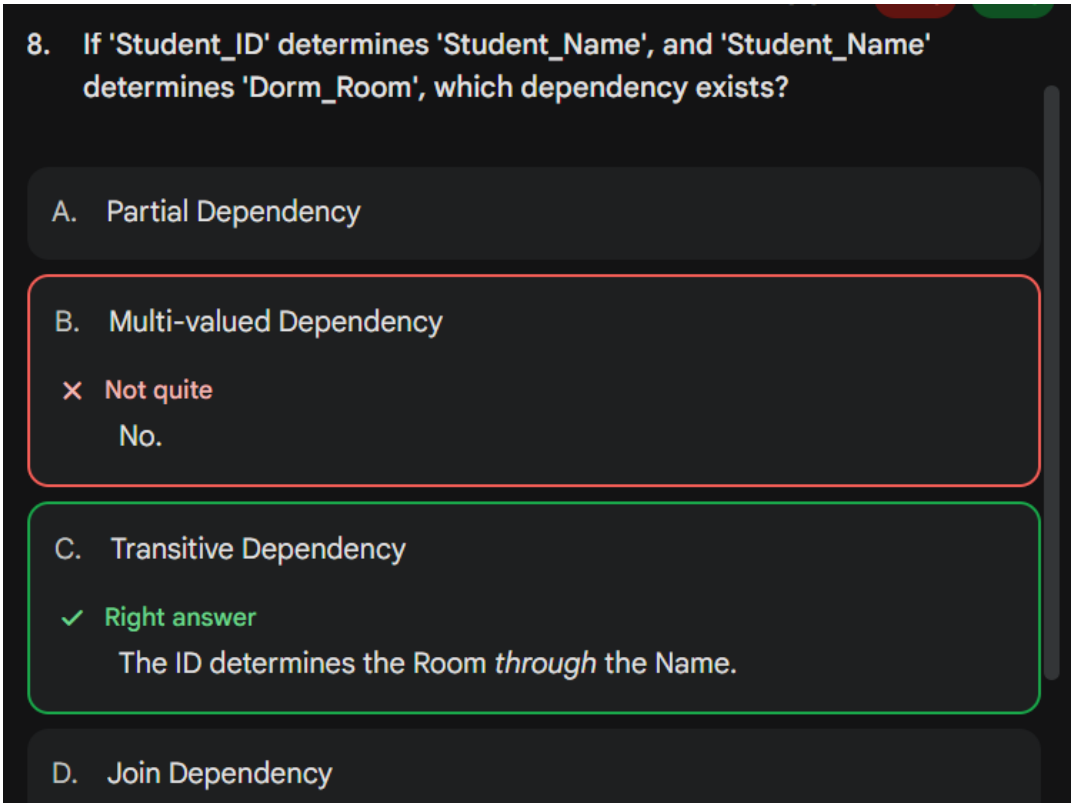
A. Atomicity

B. Consistency

✗ Not quite
No.

C. Lossless Join Property

✓ Right answer
The Join of the decomposed tables must yield the original table exactly.



- Indexing: different types, hash vs B-tree index

Indexing is a data structure that improves the speed of data retrieval operations on a database table.

1. Types of Indices 🗃

A. Primary (Clustered) Index

- **Concept:** The data in the table is **physically sorted** on the disk based on this key.
- **Rule:** You can only have **ONE** Clustered Index per table (because you can only sort the physical rows in one order).
- **Analogy:** A **Dictionary**. The words are physically arranged in alphabetical order. You don't need a separate list to find "Apple"; you just go to the 'A' section.

B. Secondary (Non-Clustered) Index

- **Concept:** A separate file that contains the Key and a Pointer (address) to the actual data row. The data itself is not sorted.
- **Rule:** You can have **MANY** Secondary Indices.

- **Analogy:** The **Index at the back of a Textbook**. The book chapters are ordered by topic (Physical), but the index is ordered alphabetically (Logical).

C. Dense vs. Sparse Index

- **Dense:** An index entry exists for **every single** search key value. (Faster search, takes more space).
- **Sparse:** An index entry exists only for **some** values (e.g., the first record of each block). (Slower search, saves space).

2. B-Tree Index (The All-Rounder) 🌲

Most databases (like MySQL, Oracle) use **B-Trees** (or B+ Trees) as the default.

- **Structure:** A balanced tree structure. It keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time.
- **Best For: Range Queries**.
    - *Example:* `SELECT * FROM Users WHERE Age > 18 AND Age < 30`.
    - Because the tree is sorted, the database finds "18" and just reads the next nodes until it hits "30".

3. Hash Index (The Specialist) #️⃣

- **Structure:** Uses a **Hash Function** to calculate the address (bucket) where the data is stored.
- **Best For: Exact Matches** (Equality).
    - *Example:* `SELECT * FROM Users WHERE ID = 542`.
    - It calculates `Hash(542)` and jumps instantly to that slot. It is faster than B-Tree for this specific job ($O(1)$ vs $O(\log N)$).
- **Worst For: Range Queries**.
    - Since hash values are random, "542" might be in Bucket 1 and "543" in Bucket 99. You can't just "read the next one."

1. What is the primary benefit of creating an Index on a database column?

A. It saves storage space.

B. It encrypts the data.

C. It speeds up data insertion.

D. It speeds up data retrieval (Select statements).

✓ That's right!
It avoids full table scans.

2. Which type of index physically reorders the table data on the disk?

A. Clustered (Primary) Index

✓ Right answer
Since data can only be physically sorted one way, there is only one Clustered Index.

B. Hash Index

C. Secondary Index

✕ Not quite
No.

3. How many Clustered Indices can a single table have?

A. Only One

✓ That's right!

You cannot physically sort a pile of books by Title AND by Author at the same time.

B. Unlimited

C. None

D. Up to 16

4. Which index structure is best suited for Range Queries (e.g., `WHERE Salary > 50000`)?

A. None

B. Hash Index

C. B-Tree (or B+ Tree) Index

✓ That's right!

B-Trees keep data sorted, making it easy to find a starting point and read sequentially.

D. Bitmap Index

5. Which index structure provides the fastest access ($O(1)$) for Exact Match queries (e.g., `WHERE ID = 101`)?

A. Clustered Index

B. Sparse Index

C. B-Tree Index

D. Hash Index

✓ That's right!
Hash calculates the address instantly.

6. What is a disadvantage of a Dense Index compared to a Sparse Index?

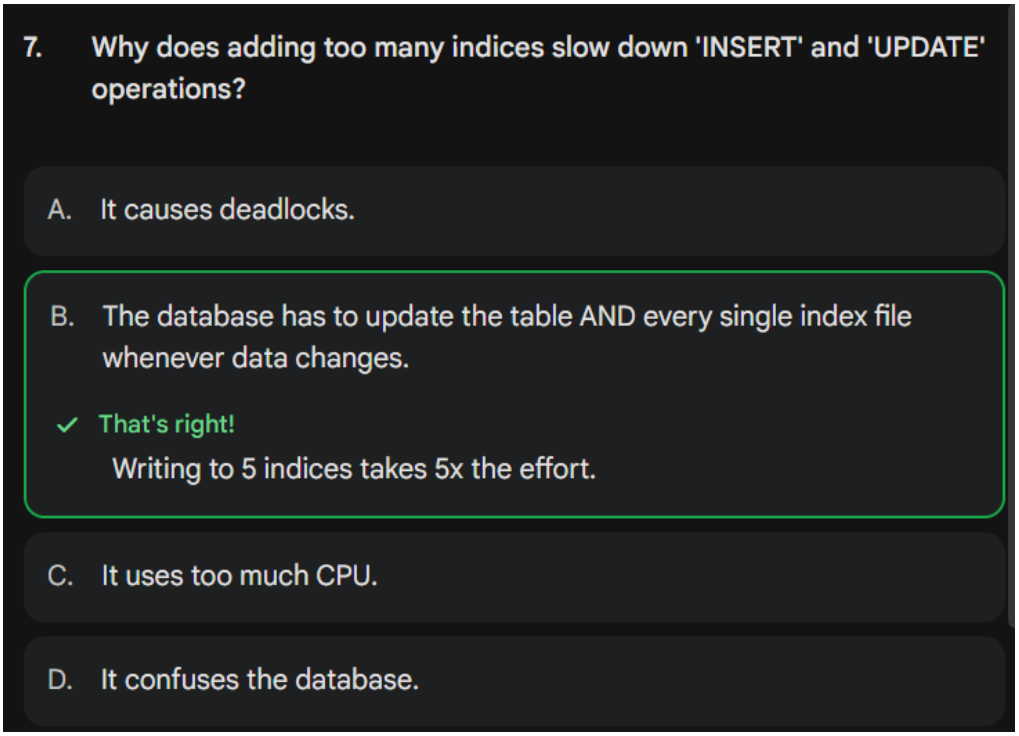
A. It is hard to create.

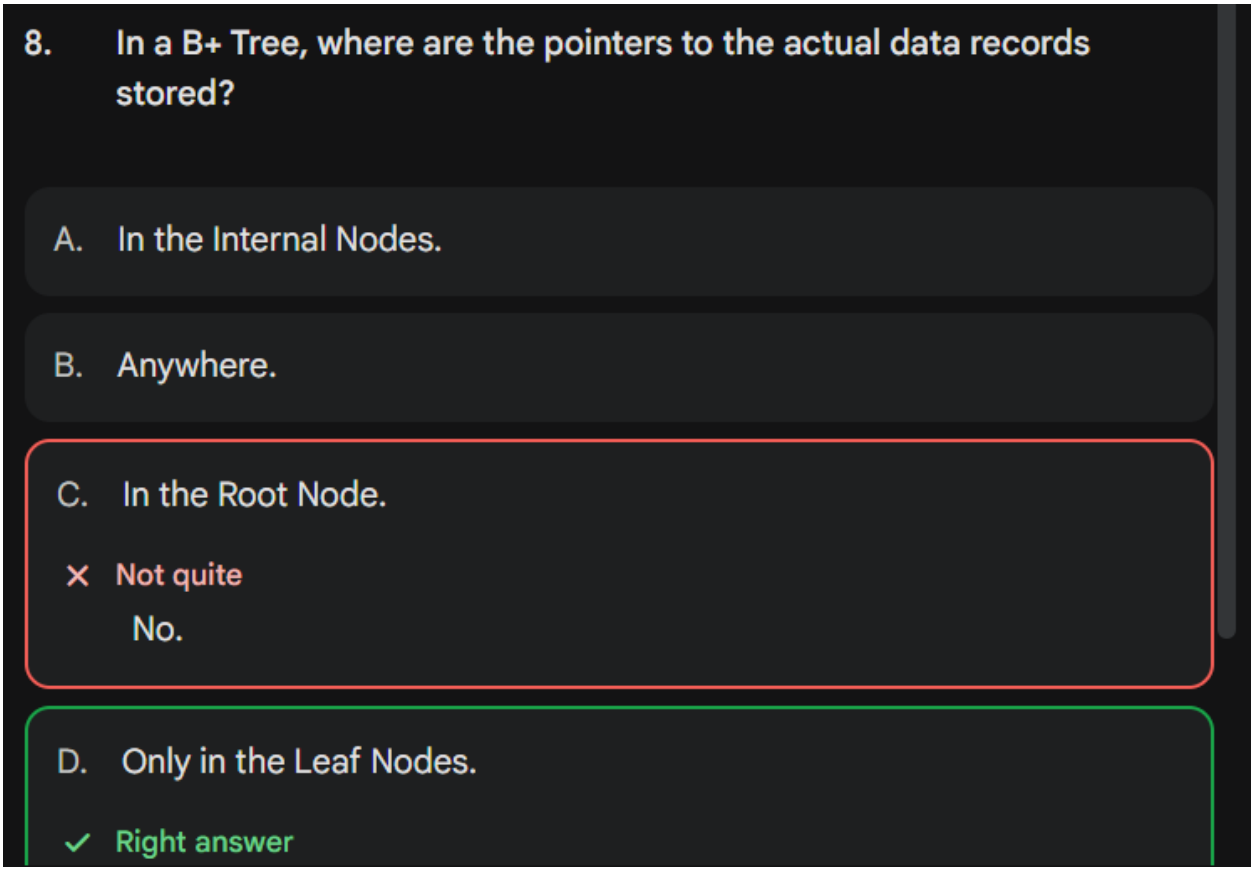
B. It cannot handle duplicates.

C. It requires more storage space.

✓ That's right!
Because it has an entry for *every* row, the index file is much larger.

D. It is slower to search.





- Transaction problems, concurrency control techniques, recovery methods

Imagine a **Bank Transfer**.

- You transfer $100 from Account A to Account B.
- This involves two steps:
    - Deduct $100 from A.
    - Add $100 to B.
- If the power goes out after Step 1 but before Step 2, money vanishes. This is a disaster.

A **Transaction** is a logical unit of work that must be treated as a whole. It follows the **ACID Properties**:

- **Atomicity:** All or Nothing. (If one part fails, the whole thing is undone).
- **Consistency:** The database moves from one valid state to another.
- **Isolation:** Transactions running at the same time shouldn't mess each other up.
- **Durability:** Once saved (committed), the changes are permanent, even if the power fails.

1. Transaction Problems (When Isolation Fails) 💥

When multiple users access the database at the same time (Concurrency), strange things can happen without proper controls.

A. Dirty Read

- **Scenario:** User A updates a value but hasn't committed yet. User B reads that uncommitted value. User A then cancels (Rolls back).
- **Result:** User B is working with data that never technically existed.

B. Lost Update

- **Scenario:**
    - User A reads "Balance = 100".
    - User B reads "Balance = 100".
    - User A writes "90" (withdraws 10).
    - User B writes "110" (deposits 10).
- **Result:** User B's write overwrites User A's write. The final balance is 110, but it should be 100. User A's withdrawal is lost.

C. Phantom Read

- **Scenario:** User A reads "All students with GPA > 3.0" and finds 10 rows. User B inserts a new student with GPA 4.0. User A runs the same query again and suddenly finds 11 rows.
- **Result:** A "phantom" row appeared out of nowhere.

2. Concurrency Control Techniques (The Traffic Police)

A. Locking Protocol (2-Phase Locking - 2PL)

- **Growing Phase:** The transaction acquires all the locks it needs (Read Lock or Write Lock). It cannot release any lock yet.
- **Shrinking Phase:** Once it starts releasing locks, it cannot acquire any new ones.
- **Pros:** Guarantees Serializability (Safety).
- **Cons:** Can cause **Deadlocks**.

B. Timestamp Ordering Protocol

- **Concept:** Every transaction gets a timestamp (e.g., T1 started at 10:00, T2 at 10:01).
- **Rule:** Older transactions always have priority. If a younger transaction tries to overwrite an older one's data, it gets rolled back and restarted.
- **Pros:** No Deadlocks.

3. Recovery Methods (The Backup Plan)

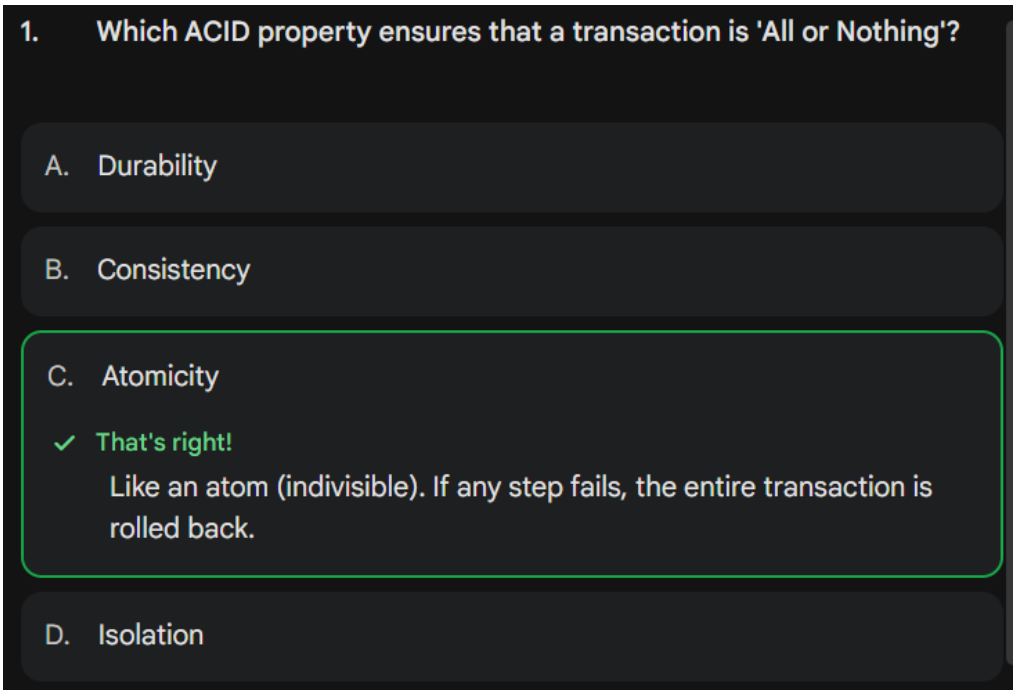
What happens if the system crashes?

A. Log-Based Recovery (The Black Box)

- The DBMS records every single action in a **Log File** *before* touching the actual data (Write-Ahead Logging - WAL).
- **Deferred Update:** Changes are written to the database only *after* the transaction commits. (No Undo needed).
- **Immediate Update:** Changes are written instantly. If a crash happens, the system uses the Log to **Undo** uncommitted changes and **Redo** committed ones.

B. Shadow Paging

- **Concept:** The database maintains two Page Tables (Current and Shadow).
- **Mechanism:** When writing, you don't overwrite the old page. You write to a *new* block and point the Current Page Table to it. The Shadow Page Table still points to the old (safe) data.
- **Crash:** If it crashes, discard the Current table and revert to the Shadow table.



2. What is a 'Dirty Read'?

A. Reading uncommitted data written by another transaction.

✓ That's right!
If the other transaction rolls back, you have read a value that never really existed.

B. Reading data without permission.

C. A slow read.

D. Reading corrupted data from a bad hard drive sector.

- 3. In the Two-Phase Locking (2PL) protocol, what happens in the 'Shrinking Phase'?

A. The transaction acquires locks.

B. The transaction commits.

C. The transaction releases locks and cannot acquire any new ones.

✓ That's right!
Once you start letting go, you can't grab anything else.

D. The database shrinks in size.

4. Which recovery technique uses a 'Shadow Page Table'?

A. Deferred Update

B. Log-Based Recovery

C. Checkpoints

D. Shadow Paging

✓ That's right!
It maintains a copy of the old page table (shadow) as a backup.

5. What is the 'Lost Update' problem?

A. When data is deleted.

B. When two transactions update the same item, and the second update overwrites the first one without knowing it.

✓ That's right!
T1 writes X. T2 writes X. T1's write is lost forever.

C. When a query returns zero rows.

D. When the hard drive crashes.

6. Which protocol uses the timestamps of transactions to decide priority?

A. None

B. Wait-Die Scheme

C. 2PL (Two-Phase Locking)

D. Timestamp Ordering Protocol

✓ That's right!
Older timestamp = Higher priority.

7. What is 'Write-Ahead Logging' (WAL)?

A. Writing the log entry to stable storage BEFORE the actual data is updated on the disk.

✓ That's right!
Ensures that if a crash happens during the data write, we have a record to recover from.

B. Writing only to RAM.

C. Writing logs after the transaction commits.

D. Writing data to the disk before writing to the log.

8. **Which concurrency control problem involves a query returning a different set of rows when executed twice in the same transaction?**

A. Dirty Read

B. Non-Repeatable Read

C. Lost Update

D. Phantom Read

✓ That's right!

A 'phantom' row appears because another transaction inserted data that matches the query condition.