Why not just use Excel / a single file?

Databases solve the problems files struggle with in real applications.

Collaboration

Many people editing the same data at the same time - safely.

Data integrity

Rules prevent invalid / messy data (e.g., unique IDs, required fields).

Performance

Fast search/filter at scale (no full-file scan every time).

MySQL: First Steps

Mental model + safety guarantees (ACID) + the 5 SQL verbs you'll use today.

MySQL

```
SELECT *
FROM students
WHERE marks > 70;
```

What is MySQL?

A relational database management system (RDBMS) for structured data + reliable querying.

Relational

Data is stored in tables (rows + columns).
Relationships are expressed using IDs.

Used via Structured Query Language (SQL)

SQL is the language you use to read/filter
and add/update/delete rows.

Mental model: system vs language vs tool

This separation prevents most beginner confusion.

MySQL Server (system)

Background service that stores data and executes queries.

SQL (language)

Commands you write: SELECT, INSERT, UPDATE, DELETE…

Client (tool)

Workbench or CLI sends SQL to the server and shows results.

Tables: rows + columns

A table stores many similar records in a predictable shape.

students

id	name	course	marks
1	Asha	DSA	82
2	Meera	DBMS	67
3	Ravi	DSA	91

- Row = one record (one student).
- Column = one attribute (name / marks / …).

Row vs Column

Add a student = new row. Add a new property = new column.

Row = a full record

Example: (id=2, name=Ravi, course=DBMS, marks=67)

Adding a student creates a new row.

Column = one field for everyone

Example column: marks for all students

Adding a field changes the table structure.

Primary Key (PK)

A column that uniquely identifies each row. Makes updates/deletes precise.

Why you need it

Names can repeat. IDs should not.

PK ensures every row has a unique identity.

How you use it

Update a row where the id = 2

Delete a row where the id = 2

You target exactly one row.

AUTO_INCREMENT

MySQL can generate new IDs automatically so you don’t manage them manually.

What it does

Insert a new row -> MySQL picks the next id (1, 2, 3…).

You usually omit id in INSERT statements.

Why it matters

Avoids duplicate IDs (especially in a class).

Keeps inserts simple and reliable.

SQL in 5 verbs

Most beginner workflows are just these actions on rows.

SELECT

Read rows

WHERE

Filter rows

INSERT

Add rows

UPDATE

Change rows

DELETE

Remove rows

Note: UPDATE/DELETE almost always need WHERE

SELECT = read

“Show me rows from a table.”

```
SELECT *
FROM students;
```

- SELECT reads data
- * means “all columns”
- FROM chooses the table

WHERE = filter

“Only rows that match a condition.”

```
SELECT *
FROM students
WHERE marks > 70;
```

- WHERE is the filter
- No WHERE = you see everything
- Operators:
  - = (equal),
  - > (greater than),
  - < (less than),
  - >= (greater than or equal to),
  - <= (less than or equal to),
  - != (not equal)

INSERT = add a row

“Create a new record.”

```
INSERT INTO students (name, course, marks)
VALUES ('Asha','DSA',82);
```

Note: if id is AUTO_INCREMENT-ed, so we usually omit it.

UPDATE = change existing rows

Always combine UPDATE with a WHERE so you target intended rows.

```
UPDATE students
SET marks = 85
WHERE id = 2;
```

Recommended: run the SELECT with the same WHERE first to verify you’re updating the right row.

DELETE = remove rows

DELETE is permanent in most setups - filter carefully.

```
DELETE FROM students
WHERE marks < 50;
```

Recommended: SELECT first, then DELETE with the same WHERE.

The safety rule

UPDATE/DELETE without WHERE can change everything.

Scary examples (don’t run)

```
UPDATE students SET marks = 0;
DELETE FROM students;
```

No WHERE -> affects ALL rows.

Safe habit

1) SELECT ... WHERE ...
2) If the rows are correct, run UPDATE/DELETE
3) SELECT again to verify

Read queries like English

If you can translate it, you can debug it.

```
SELECT name, marks
FROM students
WHERE course = "DSA";
```

English: “Show name and marks for students whose course is DSA.”

Storage Engine + InnoDB

A storage engine decides how tables are stored and how reads/writes behave.

Storage engine (what it means)

The component responsible for how table data is stored and retrieved.
Different engines -> different tradeoffs.

InnoDB (common default)

Supports transactions (for safety).
Designed for reliability and concurrency.
Often the default engine unless configured otherwise.

What is a transaction?

A group of changes treated as one unit: all succeed or all fail.

Example: money transfer

1) Subtract from account A

2) Add to account B

If step 2 fails, step 1 should not remain (no half-finished state).

ACID Properties

A simple checklist for “safe writes” in databases.

A

Atomicity

All or nothing

C

Consistency

Rules stay valid

I

Isolation

Concurrent users stay safe

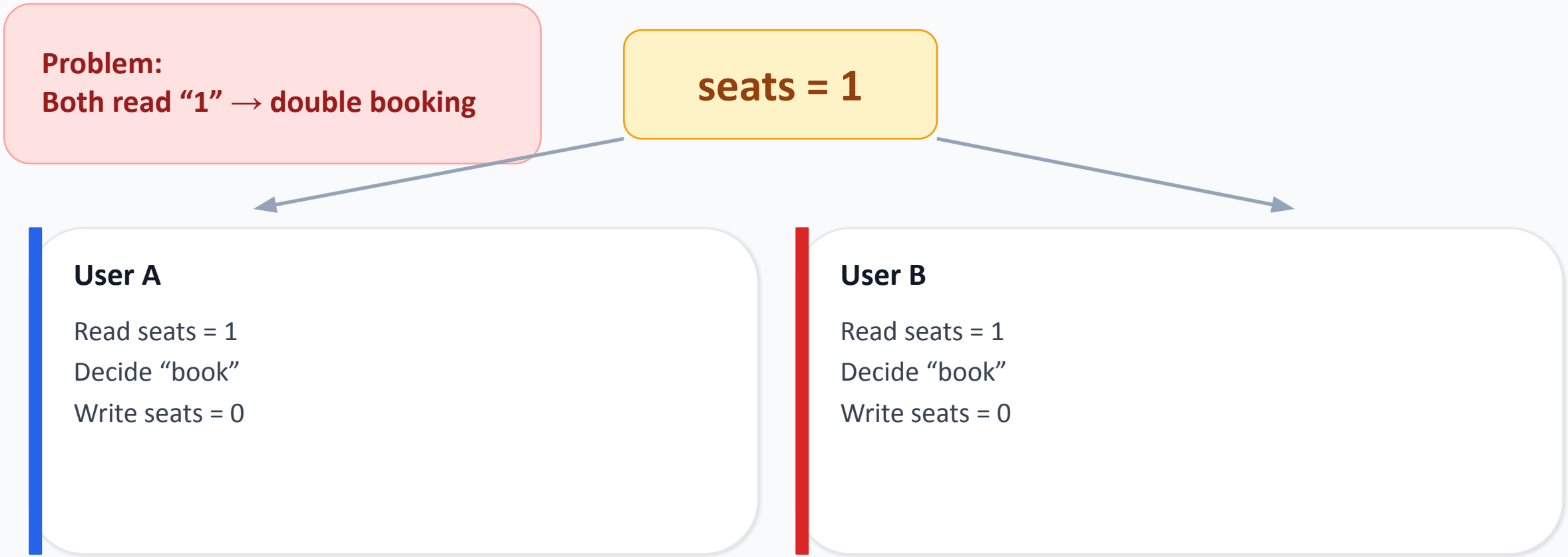
D

Durability

Committed data survives crashes

Why “Isolation” matters

Two users can read the same state before either write happens -> double booking.



Practice tasks

Complete a task, verify with SELECT & try the next one.

Tasks

1) Insert 2 more students
2) Fetch students with marks > 70
3) Update ONE student’s marks to 85 (by id)
4) Delete students with marks < 50

Verification habit

After INSERT: SELECT * (row count increased)

Before UPDATE/DELETE: SELECT with the same WHERE

After UPDATE/DELETE: SELECT again to confirm