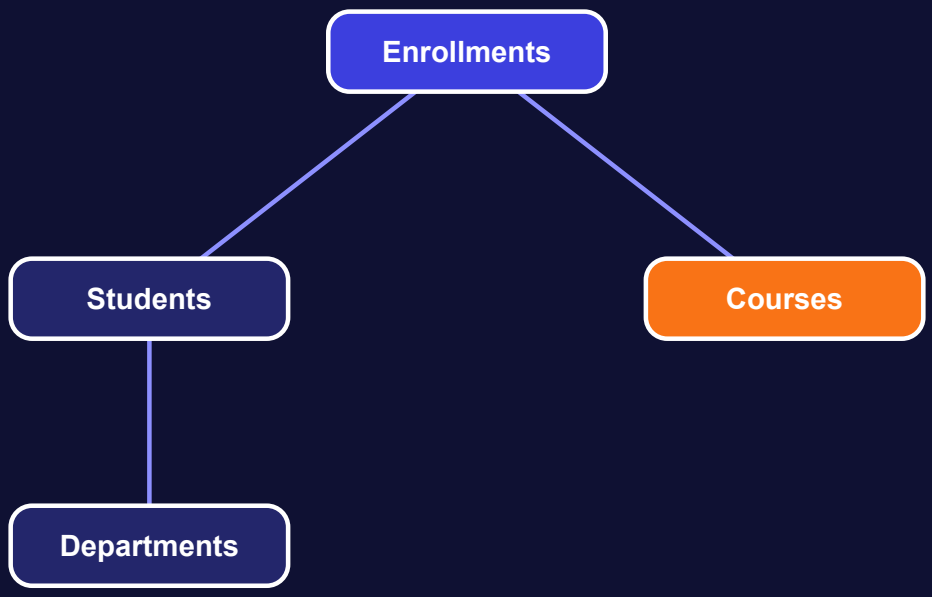
Normalization in DBMS

A worked register, SQL examples and practice questions



Code and Debug

codeanddebug.in/blog | YouTube: @codeanddebug

1. Two timetables for the same course

Asha brings you a timetable that says SQL Basics. Ravi has one that says SQL. Both students are enrolled in C1. You need to decide where the official title belongs.

Student	Course ID	Title on timetable
Asha	C1	SQL Basics
Asha	C2	Python
Ravi	C1	SQL

Work the desk with a pencil

Circle the two C1 rows. Correcting Ravi's copy today still leaves you two places to edit next time. Underline the last Python enrollment. Cross it out. Where did the course title go?

Request	Side effect in one flat register	Name
Rename C1	Two titles can disagree.	Update anomaly
Add C3 with zero students	An enrollment needs a student.	Insertion anomaly
Drop the last Python enrollment	The course record disappears.	Deletion anomaly

How many enrollments?

Before this register, Asha wrote both course IDs in one cell. Break that list into two rows. Add Ravi's SQL row. Two students now have three enrollments. Each course cell holds one ID. This is first normal form (1NF).

2. Find a home for each fact

Each student may enroll in a course once. S1 alone cannot identify one enrollment, and neither can C1. Together they can. This pair is the composite primary key.

Fact	Which ID determines it?	Table
Asha’s name	Student ID	Students
The title SQL	Course ID	Courses
Asha takes C1	Student ID and course ID	Enrollments
The name Computing	Department ID	Departments

2NF: test part of the key

Cover the student ID. C1 still tells you the course title. In the flat register the title depends on part of the key. Put it in Courses. Put names in Students. Keep the enrollment pair together.

3NF: follow the second lookup

S1 tells you Asha belongs to D1. D1 tells you the name Computing. Put that name in Departments and keep D1 in Students as a foreign key.

Table	Rows in this example
Students	S1 / Asha / D1; S2 / Ravi / D1
Departments	D1 / Computing
Courses	C1 / SQL; C2 / Python
Enrollments	S1 / C1; S1 / C2; S2 / C1

Trace Ravi before turning the page

Start at S2 / C1. Which student, course title and department will a join return? Answer: Ravi, SQL and Computing. Ravi has no C2 enrollment, so Python must be absent from his timetable.

3. Run the original register

Open an empty SQLite database. Run all four SQL blocks on pages 4 to 6 in order, using the same session. The sample starts with consistent course titles. The two names on page 2 illustrated what a missed edit would do.

SQL

```
PRAGMA foreign_keys = ON;
CREATE TABLE register (
  student_id TEXT NOT NULL, name TEXT NOT NULL,
  course_id TEXT NOT NULL, title TEXT NOT NULL,
  dept_id TEXT NOT NULL, dept_name TEXT NOT NULL,
  PRIMARY KEY (student_id, course_id)
);
INSERT INTO register VALUES
  ('S1','Asha','C1','SQL','D1','Computing'),
  ('S1','Asha','C2','Python','D1','Computing'),
  ('S2','Ravi','C1','SQL','D1','Computing');
SELECT name, title FROM register
ORDER BY student_id, course_id;
```

OUTPUT

```
name | title
Asha | SQL
Asha | Python
Ravi | SQL
```

Read the result

Two different students take SQL, and Asha also takes Python. The order comes from ORDER BY. The two IDs form the primary key, so inserting another S1 / C1 enrollment would violate the key.

Predict the split

How many rows will Courses need? Count the different IDs: C1 and C2. A repeated course enrollment does not create another course.

4. Split the register

Create the parent tables before inserting enrollment pairs. REFERENCES links a column to a parent key. SELECT DISTINCT copies each different fact once.

SQL

```
CREATE TABLE departments (
  dept_id TEXT PRIMARY KEY NOT NULL, dept_name TEXT NOT NULL
);
CREATE TABLE students (
  student_id TEXT PRIMARY KEY NOT NULL, name TEXT NOT NULL,
  dept_id TEXT NOT NULL REFERENCES departments(dept_id)
);
CREATE TABLE courses (
  course_id TEXT PRIMARY KEY NOT NULL, title TEXT NOT NULL
);
CREATE TABLE enrollments (
  student_id TEXT NOT NULL REFERENCES students(student_id),
  course_id TEXT NOT NULL REFERENCES courses(course_id),
  PRIMARY KEY (student_id, course_id)
);
INSERT INTO departments SELECT DISTINCT dept_id, dept_name FROM register;
INSERT INTO students SELECT DISTINCT student_id, name, dept_id FROM register;
INSERT INTO courses SELECT DISTINCT course_id, title FROM register;
INSERT INTO enrollments SELECT student_id, course_id FROM register;
SELECT course_id, title FROM courses ORDER BY course_id;
```

OUTPUT

```
course_id | title
C1 | SQL
C2 | Python
```

If the source already contains two titles for C1, check the correct title before migrating. DISTINCT can remove identical copies. It cannot decide which conflicting title is right.

5. Rebuild a timetable and rename C1

Read from Enrollments. Each JOIN looks up one matching ID. Compare all six values in each result row with the three original inserts on page 4.

SQL

```
SELECT s.student_id, s.name, c.course_id, c.title,
       d.dept_id, d.dept_name
FROM enrollments AS e
JOIN students AS s ON s.student_id = e.student_id
JOIN courses AS c ON c.course_id = e.course_id
JOIN departments AS d ON d.dept_id = s.dept_id
ORDER BY s.student_id, c.course_id;
```

OUTPUT

```
student_id | name | course_id | title | dept_id | dept_name
S1 | Asha | C1 | SQL | D1 | Computing
S1 | Asha | C2 | Python | D1 | Computing
S2 | Ravi | C1 | SQL | D1 | Computing
```

Now make the real edit

SQL

```
UPDATE courses SET title = 'SQL Basics'
WHERE course_id = 'C1';
SELECT s.name, c.title
FROM enrollments AS e
JOIN students AS s ON s.student_id = e.student_id
JOIN courses AS c ON c.course_id = e.course_id
WHERE c.course_id = 'C1'
ORDER BY s.student_id;
```

OUTPUT

```
name | title
Asha | SQL Basics
Ravi | SQL Basics
```

The SELECT reads the title twice, once per enrollment. The UPDATE wrote it once, in Courses.

6. Test your design

Cover the answers below. Make a choice and explain which fact or key supports it.

1. Asha has two course IDs in one Courses cell. Which repair makes counting enrollments easier?

Add a third ID / Give each enrollment a row / Sort the names

2. Which fact needs both S1 and C1?

Asha's name / The title of C1 / Whether Asha takes SQL

3. C1 is renamed. Where should you edit its title?

Courses / Students / Every enrollment

4. Ravi leaves the college. Should that delete Computing?

Yes / Only if he took SQL / No

5. Two different students have the same name. What should the join match?

Student name / Student ID / Course title

6. A student can take C1 in two terms. Which key fits one attempt per term?

Course ID / Student ID and course ID / Student ID, course ID and term

Answer key

1. Give each enrollment a row. One row now means one student on one course. COUNT can count those rows.

2. Whether Asha takes SQL. An enrollment describes the pair. Names and titles can each be looked up with one ID.

3. Courses. Each course has one row in Courses. A join reads that title for everyone.

4. No. A department can exist independently of Ravi. Its row belongs in Departments.

5. Student ID. The ID identifies one student. Matching names could attach the wrong enrollment.

6. Student ID, course ID and term. Adding term distinguishes the two attempts under that business rule.