Principles of Software Engineering and Data Bases

Davide Yi Xian Hu
Email: davideyi.hu@polimi.it

Date: 5 November 2024

Exercise Lecture: 02 - SQL

POLITECNICO
MILANO 1863

SQLite

- 👉 Serverless Architecture
- 👉 Zero Configuration
- 👉 Single File Database
- 👉 Widely Used in Mobile and Embedded Systems
- 👉 **SQL Compliance (most of SQL-92)**
- 👉 Flexible Typing System
- 👉 Data Size Limitations

Exercise 1
Exams

Exercise 1 - Exams

Database Schema:

- 👉 **STUDENT** (<u>Student-ID</u>, Name, City, Degree Program)
- 👉 **EXAM** (<u>Student-ID</u>, <u>Course-Code</u>, Date, Grade)
- 👉 **COURSE** (<u>Course-Code</u>, Title, Professor)

Exercise 1 - Exams

Naming Conventions

- 👉 **Table:** Camel Case
  - 🔎 Student, StudentExam, Course, CourseDate
- 👉 **Column:** Snake Case
  - 🔎 student, student_exam, course, course_date
- 👉 **Keywords:** Uppercase
  - 🔎 STUDENT, STUDENTEXAM, COURSE, COURSEDATE

Exercise 1 - Exams

👉 **STUDENT** (<u>Student-ID</u>, Name, City, Degree Program)

```
CREATE TABLE Student (
    student_id INTEGER PRIMARY KEY, -- OK student IDs
    name TEXT NOT NULL,    -- Name of the student
    city TEXT,             -- City of the student
    degree_program TEXT    -- Degree program
);
```

Exercise 1 - Exams

👉 **COURSE** (<u>Course-Code</u>, Title, Professor)

```
CREATE TABLE Course (
    course_code INTEGER PRIMARY KEY,  -- PK course codes
    title TEXT NOT NULL, -- Course title
    professor TEXT -- Name of the professor
);
```

Exercise 1 - Exams

👉 **EXAM** (Student-ID, Course-Code, Date, Grade)

```
CREATE TABLE Exam (
    student_id INTEGER,   -- Foreign key referring to STUDENT
    course_code INTEGER,  -- Foreign key referring to COURSE
    date TEXT NOT NULL,   -- Date of the exam
    grade INTEGER,        -- Grade the student received
    …
```

Exercise 1 - Exams

👉 **EXAM** (<u>Student-ID</u>, <u>Course-Code</u>, Date, Grade)

```
    ...
    PRIMARY KEY (student_id, course_code),
    FOREIGN KEY (student_id) REFERENCES Student (student_id),
    FOREIGN KEY (course_code) REFERENCES Course (course_code)
);
```

Exercise 1 - Exams

```
INSERT INTO Student
  (student_id, name, city, degree_program) VALUES

(1, 'Alice Johnson', 'Milan', 'Health IT'),
(2, 'Bob Smith', 'Milan', 'Management'),
(3, 'Charlie Brown', 'Rome', 'Computer Science'),
(4, 'Diana Prince', 'Milan', 'Engineering'),
(5, 'Eve White', 'Milan', 'Health IT');
```

Exercise 1 - Exams

```
INSERT INTO Course
  (course_code, title, professor) VALUES

(101, 'Mathematics', 'Dr. Alan Turing'),
(102, 'Informatics', 'Dr. Grace Hopper'),
(103, 'Physics', 'Dr. Marie Curie'),
(104, 'Data Structures', 'Dr. Edsger Dijkstra'),
(105, 'Health Informatics', 'Dr. Rosalind Franklin');
(106, 'Mathematics', 'Dr. Banana Fruit');
```

Exercise 1 - Exams

```
INSERT INTO Exam
  (student_id, course_code, date, grade) VALUES

(1, 101, '2024-03-10', 30),
(2, 101, '2024-03-15', 18),
(3, 102, '2024-04-01', 30),
(3, 104, '2024-05-10', 28),
(4, 102, '2024-04-01', 26),
(4, 101, '2024-05-20', 30),
(5, 105, '2024-06-01', 25),
(5, 102, '2024-06-15', 30),
(5, 101, '2024-06-20', 24);
```

Exercise 1 - Exams - Query 1

? Find the names of students enrolled in the Health IT degree program in Milan.

Exercise 1 - Exams - Query 1

```
SELECT name
FROM Student
WHERE degree_program = 'Health IT'
    AND city = 'Milan';
```

Exercise 1 - Exams - Query 2

? Find the names of students who have at least one grade of 30.

Exercise 1 - Exams - Query 2

```
SELECT DISTINCT Student.name
FROM Student
JOIN Exam
     ON Student.student_id = Exam.student_id
WHERE Exam.grade = 30;
```

Exercise 1 - Exams - Query 2

```
SELECT DISTINCT s.name
FROM Student s
JOIN Exam e
     ON s.student_id = e.student_id
WHERE e.grade = 30;
```

Exercise 1 - Exams - Query 3

? Find the "*Informatics*" courses in which at least one student not enrolled in the Management degree program has scored 30.

Exercise 1 - Exams - Query 3

```
SELECT DISTINCT Course.course_code
FROM Course
JOIN Exam
     ON Course.course_code = Exam.course_code
JOIN Student
     ON Exam.student_id = Student.student_id
WHERE Course.title = 'Informatics'
  AND Student.degree_program ≠ 'Management'
  AND Exam.grade = 30;
```

Exercise 1 - Exams - Query 4

? Find the professors of Mathematics courses in which no student has scored 30.

Exercise 1 - Exams - Query 4

```
SELECT Course.professor
FROM Course
WHERE Course.title = 'Mathematics'
  AND Course.course_code NOT IN (
      SELECT Exam.course_code
      FROM Exam
      WHERE Exam.grade = 30
  );
```

Exercise 1 - Exams - Query 5

? Find the title of courses where
<u>no</u> student has scored 18 and
<u>no</u> student has scored 30.

Exercise 1 - Exams - Query 5

```
SELECT DISTINCT Course.title
FROM Course
WHERE Course.course_code NOT IN (
      SELECT Exam.course_code
      FROM Exam
      WHERE Exam.grade = 18
  )
  AND Course.course_code NOT IN (
      SELECT Exam.course_code
      FROM Exam
      WHERE Exam.grade = 30
);
```

Exercise 1 - Exams - Query 6

Find the names of students
who have received at least one grade of 30 and
have never received a grade lower than 24.

Exercise 1 - Exams - Query 6

```
SELECT DISTINCT Student.name
FROM Student
JOIN Exam ON Student.student_id = Exam.student_id
WHERE Student.student_id IN (
      SELECT student_id
      FROM Exam
      WHERE grade = 30
  )
  AND Student.student_id NOT IN (
      SELECT student_id
      FROM Exam
      WHERE grade < 24
  );
```

Exercise 1 - Exams - Query 6

```
SELECT DISTINCT Student.name
FROM Student
JOIN Exam ON Student.student_id = Exam.student_id
WHERE Exam.grade = 30
  AND Student.student_id NOT IN (
      SELECT student_id
      FROM Exam
      WHERE grade < 24
  );
```

Exercise 1 - Exams - Query 6

```
SELECT DISTINCT Student.name
FROM Student
JOIN Exam ON Student.student_id = Exam.student_id
GROUP BY Student.student_id, Student.name
HAVING MAX(Exam.grade) = 30
   AND MIN(Exam.grade) ⩾ 24;
```

Exercise 2
Flights

Exercise 2 - Flights

Database Schema:

👉 **FLIGHTS** (Departure, Arrival)

FLIGHTS	
Departure	Arrival
Milano Linate	London Heathrow
Milano Linate	London Gatwick
Milano Malpensa	London Heathrow
London Heathrow	New York JFK
London Gatwick	New York JFK
...	...

Exercise 2 - Flights

👉 **FLIGHTS** (Departure, Arrival)

```
CREATE TABLE Flights (
    departure TEXT NOT NULL,
    arrival TEXT NOT NULL,
    PRIMARY KEY (departure, arrival)
);
```

Exercise 2 - Flights

👉 **FLIGHTS** (Departure, Arrival)

```
INSERT INTO Flights (departure, arrival) VALUES
('Milano Linate', 'London Heathrow'),
('Milano Linate', 'London Gatwick'),
('Milano Malpensa', 'London Heathrow'),
('London Heathrow', 'New York JFK'),
('London Gatwick', 'New York JFK');
```

FLIGHTS	
Departure	Arrival
Milano Linate	London Heathrow
Milano Linate	London Gatwick
Milano Malpensa	London Heathrow
London Heathrow	New York JFK
London Gatwick	New York JFK
...	...

Exercise 2 - Flights - Query 1

? Determine all possible connections between two airports where at most two flights are required.

Exercise 2 - Flights - Query 1

```
SELECT DISTINCT f1.departure, f1.arrival
FROM Flights f1
UNION
-- Connections with two flights
SELECT DISTINCT f1.departure, f2.arrival
FROM Flights f1
JOIN Flights f2 ON f1.arrival = f2.departure
WHERE f1.departure ≠ f2.arrival;
```

Exercise 2 - Flights - Query 2

? Determine all possible connections between two airports where at most three flights are required.

Exercise 2 - Flights - Query 2

```
SELECT DISTINCT f1.departure, f1.arrival
FROM Flights f1 UNION
-- Connections with two flights
SELECT DISTINCT f1.departure, f2.arrival
FROM Flights f1
JOIN Flights f2 ON f1.arrival = f2.departure
WHERE f1.departure ≠ f2.arrival UNION
-- Connections with three flights
SELECT DISTINCT f1.departure, f3.arrival
FROM Flights f1
JOIN Flights f2 ON f1.arrival = f2.departure
JOIN Flights f3 ON f2.arrival = f3.departure
WHERE f1.departure ≠ f3.arrival;
```

Exercise 2 - Flights - Query 3

? Determine all possible connections between two airports.

Exercise 2 - Flights - Query 3

? Determine all possible connections between two airports.

Solution

👉 **It's not possible:**
SQLite does not support recursive calls.
(at least few years ago)

Exercise 2 - Flights - Query 3

```
WITH RECURSIVE FlightPaths AS (
    SELECT departure, arrival, 1 AS num_flights
    FROM Flights
    UNION ALL
    SELECT fp.departure,
           f.arrival,
           fp.num_flights + 1
    FROM FlightPaths fp
    JOIN Flights f ON fp.arrival = f.departure
    WHERE fp.num_flights < 10 )
SELECT DISTINCT departure, arrival
FROM FlightPaths;
```

Exercise 2 - Flights+

👉 **FLIGHTS** (Departure, Arrival)

```
CREATE TABLE Flights (
    departure TEXT NOT NULL,
    arrival TEXT NOT NULL,
    departure_time DATETIME NOT NULL,
    arrival_time DATETIME NOT NULL,
    price REAL NOT NULL,
    PRIMARY KEY (departure, arrival, departure_time)
);
```

Exercise 2 - Flights+

```
INSERT INTO Flights (departure, arrival, departure_time, arrival_time, price) VALUES

('Milano Linate', 'London Heathrow', '2024-03-10 08:00:00', '2024-03-10 10:00:00', 150.00),
('Milano Linate', 'London Gatwick', '2024-03-11 09:00:00', '2024-03-11 11:00:00', 140.00),
('Milano Malpensa', 'London Heathrow', '2024-03-12 07:30:00', '2024-03-12 09:45:00', 160.00),
('London Heathrow', 'New York JFK', '2024-03-15 14:00:00', '2024-03-15 21:00:00', 500.00),
('London Gatwick', 'New York JFK', '2024-03-16 13:45:00', '2024-03-16 20:45:00', 480.00),
('New York JFK', 'Los Angeles', '2024-03-20 10:15:00', '2024-03-20 16:15:00', 300.00),
('Los Angeles', 'Tokyo Narita', '2024-03-25 23:30:00', '2024-03-26 11:30:00', 800.00),
('Milano Linate', 'Paris Charles de Gaulle', '2024-03-17 06:30:00', '2024-03-17 08:15:00', 120.00),
('Paris Charles de Gaulle', 'London Heathrow', '2024-03-17 09:30:00', '2024-03-17 10:30:00', 100.00),
('London Heathrow', 'Berlin Tegel', '2024-03-17 12:00:00', '2024-03-17 14:00:00', 200.00),
('Berlin Tegel', 'Tokyo Narita', '2024-03-18 10:00:00', '2024-03-19 05:30:00', 750.00),
('Tokyo Narita', 'Sydney', '2024-03-20 20:00:00', '2024-03-21 08:00:00', 900.00),
('Sydney', 'Auckland', '2024-03-22 15:30:00', '2024-03-22 20:30:00', 250.00),
('Auckland', 'Los Angeles', '2024-03-25 23:00:00', '2024-03-26 14:00:00', 700.00),
('Los Angeles', 'London Heathrow', '2024-03-27 11:00:00', '2024-03-28 06:30:00', 600.00),
('London Heathrow', 'Milano Linate', '2024-03-29 09:00:00', '2024-03-29 11:00:00', 160.00),
('Milano Linate', 'Rome Fiumicino', '2024-03-30 07:00:00', '2024-03-30 08:15:00', 90.00),
('Rome Fiumicino', 'Madrid Barajas', '2024-03-30 09:30:00', '2024-03-30 11:45:00', 130.00),
('Madrid Barajas', 'New York JFK', '2024-03-31 13:00:00', '2024-03-31 20:00:00', 450.00),
('New York JFK', 'Paris Charles de Gaulle', '2024-04-01 17:00:00', '2024-04-02 06:00:00', 500.00);
```

Exercise 2 - Flights+ - Query 1

❔ Find all flights departing from 'Milano Linate'.

Exercise 2 - Flights+ - Query 1

```
SELECT * FROM Flights
WHERE departure = 'Milano Linate';
```

Exercise 2 - Flights+ - Query 2

- ? List all unique arrival airports.
- ? List all unique arrival airports and count how many flights arrive at each airport.

Exercise 2 - Flights+ - Query 2

```
SELECT DISTINCT arrival
FROM Flights;
```

```
SELECT f.arrival, COUNT(f.arrival) AS number_of_flights
FROM Flights f
GROUP BY f.arrival
ORDER BY number_of_flights DESC;
```

Exercise 2 - Flights+ - Query 3

? Determine all possible connections (with price) between two airports where at most two flights are required.

Exercise 2 - Flights+ - Query 3

```
SELECT DISTINCT * FROM Flights f1
UNION
SELECT DISTINCT
  f1.departure, f2.arrival,
  f1.departure_time, f2.arrival_time,
  (f1.price + f2.price) AS total_price
FROM Flights f1
JOIN Flights f2 ON f1.arrival = f2.departure
WHERE f2.departure_time > f1.arrival_time;
```

Exercise 2 - Flights+ - Query 4

? Find the cheapest flight for each connection.

Where a connection is defined as:
two airports are connected if you can reach it with at most two flights.

Exercise 2 - Flights+ - Query 4

```
WITH FlightConnections AS (
    SELECT f1.departure, f1.arrival, f1.departure_time,
f1.arrival_time, f1.price
    FROM Flights f1
    UNION ALL
    SELECT f1.departure, f2.arrival, f1.departure_time,
f2.arrival_time, (f1.price + f2.price) AS total_price
    FROM Flights f1
    JOIN Flights f2 ON f1.arrival = f2.departure
    WHERE f2.departure_time > f1.arrival_time
)
SELECT fc.departure, fc.arrival, MIN(fc.price) AS cheapest_price
FROM FlightConnections fc
GROUP BY fc.departure, fc.arrival;
```

Exercise 2 - Flights+ - Query 5

❔ Find the top 3 departure airports with the most connections (both direct and with one transfer), along with the average price of these connections (for each airport).

Exercise 2 - Flights+ - Query 5

```
SELECT fc.departure, COUNT(*) AS total_connections,
       AVG(fc.price) AS avg_price
FROM (
    SELECT f1.departure, f1.arrival,
           f1.departure_time, f1.arrival_time, f1.price
    FROM Flights f1
    UNION ALL
    SELECT f1.departure, f2.arrival, f1.departure_time,
           f2.arrival_time, (f1.price + f2.price) AS total_price
    FROM Flights f1
    JOIN Flights f2 ON f1.arrival = f2.departure
    WHERE f2.departure_time > f1.arrival_time
) AS fc
GROUP BY fc.departure
HAVING COUNT(*) > 1 ORDER BY total_connections DESC
LIMIT 5;
```

Exercise 3
Diagnoses

Exercise 3 - Diagnoses

Database Schema:

👉 **Patients** (patient_id, name, date_of_birth, gender, address)

👉 **Visits** (visit_id, patient_id, visit_date,
doctor_name, visit_notes)

👉 **Diagnoses** (diagnosis_id, visit_id, diagnosis_code,
diagnosis_description)

Exercise 3 - Diagnoses

👉 **Patients** (patient_id, name, date_of_birth, gender, address)

```
CREATE TABLE Patients (
    patient_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    date_of_birth DATE,
    gender TEXT CHECK (
           gender IN ('Male', 'Female', 'Other')
    ),
    address TEXT
);
```

Exercise 3 - Diagnoses

👉 **Patients** (patient_id, name, date_of_birth, gender, address)

```
INSERT INTO Patients (patient_id, name, date_of_birth, gender, address) VALUES

(1, 'John Doe', '1985-06-15', 'Male', '123 Elm Street, Springfield'),
(2, 'Jane Smith', '1990-09-25', 'Female', '456 Oak Street, Springfield'),
(3, 'Michael Johnson', '1978-02-11', 'Male', '789 Pine Avenue, Springfield'),
(4, 'Emily Davis', '2001-12-05', 'Female', '321 Maple Lane, Springfield'),
(5, 'William Brown', '1965-03-30', 'Male', '654 Birch Road, Springfield');
```

Exercise 3 - Diagnoses

👉 **Visits** (<u>visit_id</u>, patient_id, visit_date, doctor_name, visit_notes)

```
CREATE TABLE Visits (
    visit_id INTEGER PRIMARY KEY,
    patient_id INTEGER,
    visit_date DATETIME NOT NULL,
    doctor_name TEXT,
    visit_notes TEXT,
    FOREIGN KEY (patient_id) REFERENCES Patients(patient_id)
);
```

Exercise 3 - Diagnoses

👉 **Visits** (visit_id, patient_id, visit_date, doctor_name, visit_notes)

```
INSERT INTO Visits (visit_id, patient_id, visit_date, doctor_name, visit_notes) VALUES

(1, 1, '2024-01-10 09:30:00', 'Dr. Adams', 'Routine check-up'),
(2, 1, '2024-03-15 14:00:00', 'Dr. Baker', 'Follow-up for hypertension'),
(3, 2, '2024-02-20 11:00:00', 'Dr. Clark', 'Annual physical exam'),
(4, 3, '2024-04-05 15:30:00', 'Dr. Adams', 'Consultation for back pain'),
(5, 4, '2024-03-01 10:45:00', 'Dr. Clark', 'Routine check-up'),
(6, 5, '2024-02-10 08:00:00', 'Dr. Adams', 'Chronic condition management'),
(7, 2, '2024-03-10 13:15:00', 'Dr. Baker', 'Skin rash consultation');
```

Exercise 3 - Diagnoses

👉 **Diagnoses** (diagnosis_id, visit_id, diagnosis_code, diagnosis_description)

```
CREATE TABLE Diagnoses (
    diagnosis_id INTEGER PRIMARY KEY,
    visit_id INTEGER,
    diagnosis_code TEXT NOT NULL,
    diagnosis_description TEXT,
    FOREIGN KEY (visit_id) REFERENCES Visits(visit_id)
);
```

Exercise 3 - Diagnoses

👉 **Diagnoses** (diagnosis_id, visit_id, diagnosis_code, diagnosis_description)

```
INSERT INTO Diagnoses (diagnosis_id, visit_id, diagnosis_code,
diagnosis_description) VALUES

(1, 1, 'I10', 'Essential (primary) hypertension'),
(2, 2, 'I10', 'Essential (primary) hypertension - follow-up'),
(3, 3, 'Z00.00', 'General medical examination without abnormal findings'),
(4, 4, 'M54.5', 'Low back pain'),
(5, 5, 'Z00.00', 'General medical examination without abnormal findings'),
(6, 6, 'E11.9', 'Type 2 diabetes mellitus without complications'),
(7, 7, 'L30.9', 'Dermatitis, unspecified');
```

Exercise 3 - Diagnoses - Query 1

? Find the name of all patients who were visited by Dr. Adams and their diagnoses.

Exercise 3 - Diagnoses - Query 1

```
SELECT p.name, d.diagnosis_code
FROM Patients p
JOIN Visits v
     ON p.patient_id = v.patient_id
JOIN Diagnoses d
     ON v.visit_id = d.visit_id
WHERE v.doctor_name = 'Dr. Adams';
```

Exercise 3 - Diagnoses - Query 2

❔ List the most common diagnoses codes or descriptions made during visits.

Exercise 3 - Diagnoses - Query 2

```
SELECT d.diagnosis_description,
       COUNT(*) AS diagnosis_count
FROM Diagnoses d
GROUP BY d.diagnosis_description
ORDER BY diagnosis_count DESC;
```

Exercise 3 - Diagnoses - Query 3

- ? Find patients who have had more than one visit and list their most recent diagnosis.

Exercise 3 - Diagnoses - Query 3

```sql
SELECT p.name, MAX(v.visit_date) AS most_recent_visit,
       d.diagnosis_description
FROM Patients p
JOIN Visits v
     ON p.patient_id = v.patient_id
JOIN Diagnoses d
     ON v.visit_id = d.visit_id
GROUP BY p.patient_id, p.name
HAVING COUNT(v.visit_id) > 1;
```

Exercise 3 - Diagnoses - Query 3

```
SELECT Patients.name, MAX(Visits.visit_date) FROM Patients
JOIN Visits
     ON Visits.patient_id = Patients.patient_id
JOIN Diagnoses
     ON Diagnoses.visit_id = Visits.visit_id
WHERE Patients.patient_id IN (
     SELECT patient_id FROM Visits
     GROUP BY patient_id
     HAVING COUNT(patient_id) > 1
)
GROUP BY Patients.name;
```

Exercise 3 - Diagnoses - Query 4

❔ Find patients who have visited at least three different doctors and show the count of unique diagnoses they received.

Exercise 3 - Diagnoses - Query 4

```
SELECT name, unique_diagnosis FROM (
  SELECT DISTINCT Patients.patient_id, Patients.name, doctor_name
FROM Patients
  JOIN Visits ON Patients.patient_id = Visits.patient_id
) AS UniquePatientsDoctorPairs
JOIN (
  SELECT DISTINCT Patients.patient_id, COUNT(diagnosis_code) as
unique_diagnosis FROM Patients
  JOIN Visits ON Patients.patient_id = Visits.patient_id
  JOIN Diagnoses ON Visits.visit_id = Diagnoses.visit_id
  GROUP BY Patients.patient_id
) AS Counter ON Counter.patient_id =
UniquePatientsDoctorPairs.patient_id
GROUP BY name
HAVING COUNT(*) ⩾ 3;
```

Exercise 3 - Diagnoses - Query 5

? Find patients who have visited at least three different doctors and received unique diagnoses, <u>excluding those who have ever had a "General medical examination without abnormal findings" diagnosis.</u>

Exercise 3 - Diagnoses - Query 5

```
…
EXCEPT
SELECT p.name, COUNT(DISTINCT v.doctor_name),
       COUNT(DISTINCT d.diagnosis_description)
FROM Patients p
JOIN Visits v
     ON p.patient_id = v.patient_id
JOIN Diagnoses d
     ON v.visit_id = d.visit_id
WHERE d.diagnosis_description = 'General medical examination
without abnormal findings'
GROUP BY p.patient_id, p.name;
```

Exercise 3 - Diagnoses - Query 6

- Find all patients who have visited more than one doctor and list their most recent visit with the doctor they saw.

Exercise 3 - Diagnoses - Query 6

```
SELECT p.name, v.doctor_name,
       MAX(v.visit_date) AS most_recent_visit
FROM Patients p
JOIN Visits v
     ON p.patient_id = v.patient_id
GROUP BY p.name, v.doctor_name
HAVING COUNT(v.visit_id) > 1
ORDER BY most_recent_visit DESC;
```

Exercise 3 - Diagnoses - Query 7

❔ List all patients and the total number of different diagnoses they have received.

Exercise 3 - Diagnoses - Query 7

```
SELECT p.name, COUNT(DISTINCT d.diagnosis_description) AS
       total_unique_diagnoses
FROM Patients p
JOIN Visits v
     ON p.patient_id = v.patient_id
JOIN Diagnoses d
     ON v.visit_id = d.visit_id
GROUP BY p.name
ORDER BY total_unique_diagnoses DESC;
```