Interacting with Databases

Express and MySQL

Why a Database?

- During interaction(s) between a client and a server:
  - The client might need data from the server,
  - The client might have data for the server, and/or
  - The server might have data for the client.
- Data must be persistent.

What is a Database?

- A database is an organized collection of data, so that it can be easily accessed and managed.
- The main purpose of the database is to operate a large amount of information by storing, retrieving, updating, and managing data.

Which DBMS?

- There are many databases' systems available.
- Examples:
  - **MySQL**,
  - Sybase,
  - Oracle,
  - MongoDB,
  - Informix,
  - PostgreSQL,
  - SQL Server,
  - MariaDB
  - …

Two Groups

- **SQL DB**
- NoSQL DB
  - Not only SQL

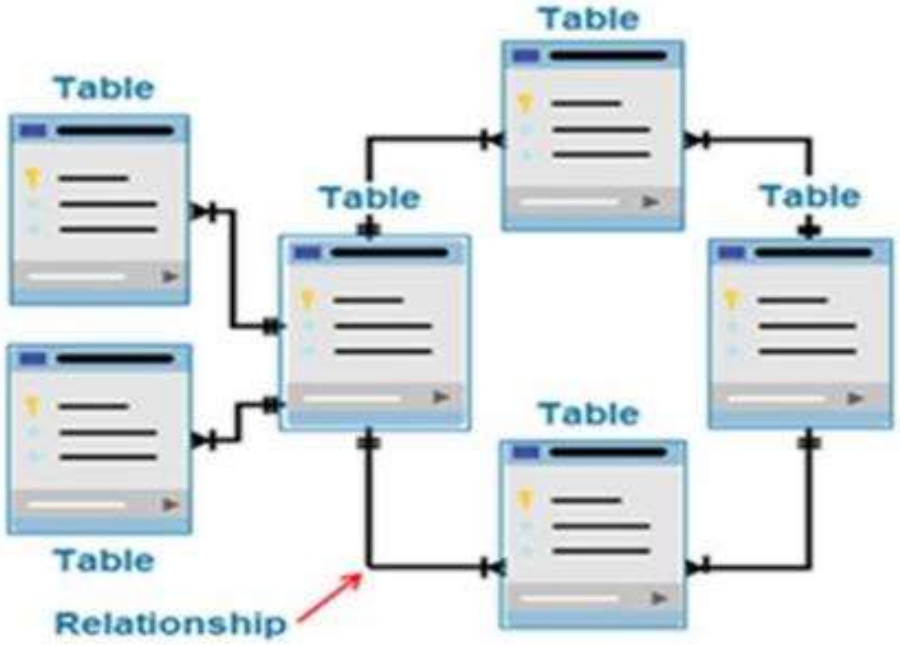
RDBMS

- A relational database is a type of database that stores and provides access to data points that are related to one another.
- Relational databases are based on the relational model, an intuitive, straightforward way of representing data in **tables**.
- In a relational database, each row in the table is a record with a unique ID called the **key**.
- The columns of the table hold attributes of the data, and each record usually has a value for each attribute.

Tables

- Tables are database objects that contain all the data in a database.
- In tables, data is logically organized in a row-and-column format like a spreadsheet.
- Each row represents a unique record, and each column represents a field in the record.
- For example, a table that contains employee data for a company might contain a row for each employee and columns representing employee information such as employee number, name, address, job title, and telephone number.

Database and tables



RDBMS

Relational Databases

Example of a table

SALES			
purchase_number	date_of_purchase	customer_id	item_code
1	03/09/2016	1	A_1
2	02/12/2016	2	C_1
3	15/04/2017	3	D_1
4	24/05/2017	1	B_2
5	25/05/2017	4	B_2
6	06/06/2017	2	B_1
7	10/06/2017	4	A_2
8	13/06/2017	3	C_1
9	20/07/2017	1	A_1
10	11/08/2017	2	B_1

CRUD: Operations on data

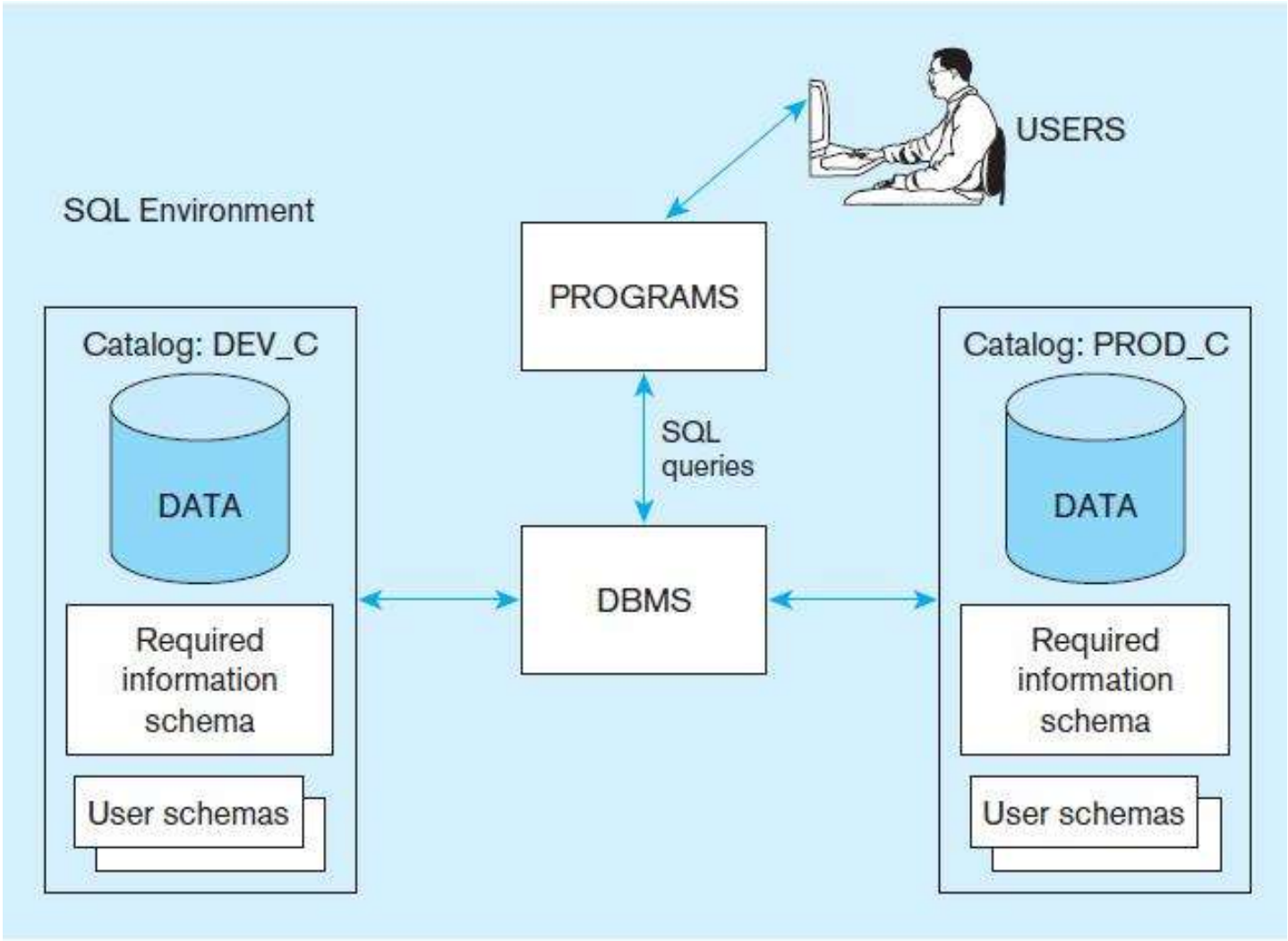
- **C**reate
- **R**ead
- **U**pdate
- **D**elete

Structured **Q**uery **L**anguage

- SQL
- Programming language
- The standard for relational database management systems (RDBMS)

Structured Query Language (2)

- SQL statements can:
  - Create/delete a database
  - Create/alter/delete a table
  - Read data (**Select** statement)
  - Add/Insert data (**Insert** statement)
  - Update data (**Update** statement)
  - Delete data (**Delete** statement)
- SQL can be used from within other programming languages (e.g. Java) using appropriate drivers (e.g., JDBC)


USERS
SQL Environment
PROGRAMS
Catalog: DEV_C
DATA
SQL
queries
Catalog: PROD_C
DATA
DBMS
Required
information
schema
Required
information
schema
User schemas
User schemas

Example: SELECT Statement options

- Used for queries on single or multiple tables
- Clauses of the SELECT statement:
  - SELECT
    - List the columns (and expressions) to be returned from the query
  - FROM
    - Indicate the table(s) or view(s) from which data will be obtained
  - WHERE
    - Indicate the conditions under which a row will be included in the result
  - GROUP BY
    - Indicate categorization of results
  - HAVING
    - Indicate the conditions under which a category (group) will be included
  - ORDER BY
    - Sorts the result according to specified criteria

MySQL

- MySQL is a well-known DBMS
  - Powering big websites (Facebook…)
- Free, open-source
- Cross-platform
- Maintained by Oracle

Install MySQL

- While we just need MySQL, XAMPP provides additional tools
  - Mainly: phpMyAdmin to create/edit/delete/browse databases, tables, and/or data.
- Search for "XAMPP installer"
- XAMPP is available for Windows, macOS, and Linux

Creating database and tables

- This can be done in:
  - phpMyAdmin or
  - JavaScript
- We will use phpMyAdmin

Create Database

- Using phpMyAdmin
- From XAMPP control center:
  - Start MySQL
  - Start apache
  - Go to localhost/phpmyadmin
  - Select Database tab
  - Give a name to your database
  - Click "Create"

Create Table

- We will create 3 tables:
  - Students: id, name, email
    - id is the primary key
  - Courses: id, code, title
    - id is the primary key
  - Grades: studentID, courseID, grade
    - Any of these can be a key?
    - Maybe you need to add a key

Node/Express and MySQL

- Express can interact with MySQL
- Use `mysql` module
  - Install with: npm install mysql

Modules

```
const express = require("express");
const mysql = require("mysql");

const app = express();
```

Connect to DB

```
const db = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "soen287_Demo" // use the name of your DB
});

db.connect((err) => {
  if (err) {
    console.log("Error connecting to DB");
  } else {
    console.log("Connected");
  }
});
```

Add student

```
app.get("/addstudent", (request, response) => {
  let student = {
    name: "Name 1",
    email: "email1@gmail.com",
  };

  let sql = "INSERT INTO Students SET ?";

  let query = db.query(sql, student, (err, result) => {
    if (err)
      response.send("Could not insert new record!");
    else
      response.send("Record inserted successfully!");
  });
});
```

Get all students (JSON)

```
app.get("/getstudents", (request, response) => {

  let sql = "SELECT * FROM STUDENTS";

  let query = db.query(sql, (err, result) => {
    if (err)
      response.send("Could not retrieve data from table!");
    else
      response.send(result);
  });
});
```

Get all students (formatted)

```
app.get("/getstudents", (request, response) => {

  let sql = "SELECT * FROM STUDENTS";

  let query = db.query(sql, (err, result) => {
    if (err)
       response.send("Could not retrieve data from table!");
    else {
      let content = "";
      for (let i = 0; i < result.length; i++) {
        content = content + result[i].name + "  " +
                                  result[i].email + "<br>";
      }
      response.send(content);
    }
  });
});
```

Add title and CSS

- Put CSS formatting in a file styles.css
- Add a link to this file in the response

CSS

```
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}

th,td {
    background-color: #96D4D4;
}

h1{
    color: blue;
}
```

With CSS

```
app.use(express.static("public")); // styles.css should be in this folder

app.get("/getstudents", (request, response) => {
  let sql = "SELECT * FROM STUDENTS";
  let query = db.query(sql, (err, result) => {
    if (err) response.send("Could not retrieve data from table!");
    else {
      let content = `<head>
                        <title>List of students</title>
                        <link rel="stylesheet" href="styles.css" />
                      </head>`;
      for (let i = 0; i < result.length; i++) {
        content =
          content +
          "<h1>" +
          result[i].name +
          "  " +
          result[i].email +
          "</h1>" +
          "<br>";
      }
      response.send(content);
    }
  });
});
```

REST API

localhost:5000/route/param1/param2/

- Replace param1, param2 by the actual values you want to pass to the API
- See next examples

Update a student

```
app.get("/updatestudent/:id", (request, response) => {
  let newEmail = "newEmail@gmail.com";

  let sql = `UPDATE Students SET email = '${newEmail}'
WHERE id = ${request.params.id}`;

  let query = db.query(sql, (err, result) => {
    if (err) response.send("Could not update the student
\n" + err);
    else response.send(result);
  });
});
```

To update student with id 1, use this link in the browser: localhost:5000/updatestudent/1

Delete a student

```
app.get("/deletestudent/:id", (request, response) =>
{
  let sql = `DELETE FROM Students WHERE id =
${request.params.id}`;
  let query = db.query(sql, (err, result) => {
    if (err)
      response.send(
        `Could not delete the student with ID =
${request.params.id} \n ${err}`
      );
    else response.send(result);
  });
});
```

To delete student with id 2, use this link in the browser:
localhost:5000/deletestudent/2

Exercise 1

- Add a few students
- Add a few courses
- Add a few grades of students in few courses
  - For example:
    - Student with id 1 got A+ in course with id 2
    - Student with id 2 got B in course with id 2
    - Students with id 3 got B+ in course with id 1

Exercise 2

- In a table, display information of all students
  - Names of students have a **hyperlink**
- Whenever a name of a student is clicked, display that student's information from the database

Exercise 3

- In a table, display information of all students
- Names of students have a **hyperlink**
  - Whenever a name of a student is clicked, display that student's information from the database
- The last column of the table has a hyperlink "Delete" for each row
  - Whenever "Delete" is clicked, delete the student of that row from the database

Exercise 4

- In a table, display grades of all students in all courses
- Each row has:
  - Student ID
  - Student names
  - Course code
  - Grade