# DASC 5333 Database Systems for Data Science

## 2/6/2024 (self - annotation)

Welcome!

I am Bun Yue, the instructor.

**Database Basics**

by K. Yue

**1. Introduction**

* It is important to be familiar with the basic terms and concepts of databases in this course.
* A database system is built by using a Database Management System (DBMS).
* One DB ranking: <http://db-engines.com/en/ranking>.
* We focus on Relational DBMS (RDBMS).
* Examples of Relational DBMS:
  1. Access: most popular 'personal' DB
  2. Oracle: Most popular commercial DBMS
  3. MS SQL server: Likely second most popular commercial DBMS
  4. MySQL: most popular open source DBMS
  5. MariaDB: highly compatible to MySQL (more open source than MySQL)
  6. Postgres: popular open source DBMS known for innovation and functionality.
  7. SQLite: most popular portable DB engine.
* Relational DBMS basically use the relational model (with extensions).
* There are many other models. Examples:
  1. Object-Oriented Database (OODB): e.g., db4o, Gemstone, etc.
  2. Big Data:
     + Document DB: e.g., MongoDB, CouchDB
     + Key-Value DB: e.g., Redis, LevelDB
     + Wide Column DB: optimized over large dataset; store columns together, not rows. E.g. Cassandra and HBase.
     + Graphical DB: e.g., Neo4J

**2. Users**

* Users drive requirements. It is always important to find out the types of users.
* There are many types of users in a RDB. Three major kinds:
  1. End users: usually do not use SQL to access the database directly. Examples:
     + front-end users
     + managers and staff
     + domain experts
  2. DB application developers and data analysts
     + Develop DB solutions using SQL.
     + With various levels of access privileges.
  3. DB administrators
     + Manage the entire DB, such as:
       - conceptual and physical database design and implementation
       - security
       - user account management
       - backup and recovery
       - performance tuning
* The likely role of most of you currently: DB and application developers and data analysts.
* You are likely not the end users of the DB system you built.
* Thus, do *not* build the database for yourself.

**3. DB Development Phases**

* The classical waterfall software development life cycle can be useful as a basis to understand the various phases of database development.
  1. Requirement: conceptual modeling.
     1. Planning
     2. Analysis
  2. Design:
     1. Logical modeling and design
     2. Physical Design
  3. Implementation
  4. Testing
  5. Maintenance
* There are many other software lifecycle models.
* DB development is a kind of software development.

A diagram of a software development process

Description automatically generated

**3.1 Data Modeling**

General conceptual model. See, for example: <https://en.wikipedia.org/wiki/Conceptual_model_(computer_science)>.

1. Capture domain knowledge and requirements from the business and application perspectives.
2. Driven by requirements.
3. Construct a conceptual model iteratively.

See, for example: <https://en.wikipedia.org/wiki/Conceptual_schema>.

Diagram of a model architecture

Description automatically generated

**Identify and capture user requirements:**

1. Likely the most tedious and difficult parts for many traditional applications.
2. Collect documents of existing systems.
3. Study documents of existing systems.
4. Talk with domain experts and end users.
5. Model the problem using a modeling language, such as UML, ER, etc.
6. Document the captured requirements: e.g., modeling, requirement specifications, data dictionary, etc.
7. Iteratively refine and correct the model until enough details are captured.

**3.2 Design database solutions**

1. Select the appropriate data model of the database.
2. Select the appropriate DBMS
3. Design the *logical* model
4. Design the architecture of the DB system
5. Design the physical database
6. Design external views
7. Design individual components

**3.3. Implementation and testing**

1. Implement and test design
2. Optimize performance

**4. The Three-Layered DB Architecture**

* The three layered database architecture is well known and you can get a lot of information about it from the Web. Examples:
  + A simple one: <https://www.tutorialspoint.com/Three-Level-Architecture-of-Database>
  + A more nuanced one: <http://jcsites.juniata.edu/faculty/rhodes/dbms/dbarch.htm>
  + Consult Figure 2.4 of Ricardo.
* Use the *layer* pattern to manage complexity.The *layer pattern* is an important concept in Computer Science and software architecture.
* Three levels:
  + External or view level: Describes a part of the database for a particular user group; provide the right level of abstraction and security control.
  + Logical level: Describes logical structure of the entire database
    1. Some practitioners call the 'logical level' as the 'conceptual level'. This can cause confusion as other distinguish between 'conceptual model' and 'logical model'.
  + Internal/Physical level: Describes physical storage structure of the database
* Provide data independence
  + Logical data independence:
    1. between logical database and external views.
    2. changes in the logical database may not affect the external views.
  + Physical data independence:
    1. between logical database and physical database.
    2. changes in the physical database do not affect the correctness of the logical database.
* The logical level is the main focus.

A diagram of a data base

Description automatically generated

**A First Look at SQL**

by K. Yue

**1. Introduction**

* The basic relational data model in layman and more theoretical terms can be seen again here: [RelationalModel.html](https://dcm.uhcl.edu/yue/courses/joinDB/Spring2024/notes/Intro/RelationalModel.html)

**2. Very Simple SQL**

Install XAMPP: which contains MariaDB (MySQL) and Apache (Web server). See Section 3.1 below.

MySQL creation script for creating toyu: [Createtoyu.sql.txt](https://dcm.uhcl.edu/yue/courses/joinDB/Spring2024/notes/query/Createtoyu.sql.txt) (may remove .txt while saving). You can execute it with the source command in MySQL command line prompt:

1. Save the script file as createtoyu.sql in your working directory for this class.
2. Open a commnad line terminal in your working directory
3. Start mysql command prompt.
4. Run "source createtoyu.sql"

* A simple SELECT statement in *SQL* an be used to retrieve data from a table.

***Example:*** in MySQL, executing:

SELECT \* FROM grade;  
SELECT \* FROM school;  
SELECT \* FROM department;  
SELECT \* FROM faculty;  
SELECT \* FROM course;  
SELECT \* FROM class;  
SELECT \* FROM student;  
SELECT \* FROM enroll;

Result: (MariaDB is compatible to MySQL).

MariaDB [toyu]> SELECT \* FROM grade;  
+-------+------------+  
| grade | gradePoint |  
+-------+------------+  
| A     |     4.0000 |  
| A-    |     3.6667 |  
| B     |     3.0000 |  
| B+    |     3.3333 |  
| B-    |     2.6667 |  
| C     |     2.0000 |  
| C+    |     2.3333 |  
| C-    |     1.6667 |  
| D     |     1.0000 |  
| D+    |     1.3333 |  
| D-    |     0.6667 |  
| F     |     0.0000 |  
| IP    |       NULL |  
| P     |       NULL |  
| WX    |       NULL |  
+-------+------------+  
15 rows in set (0.004 sec)  
  
MariaDB [toyu]> SELECT \* FROM school;  
+------------+-------------------------------+  
| schoolCode | schoolName                    |  
+------------+-------------------------------+  
| BUS        | Business                      |  
| EDU        | Education                     |  
| HSH        | Human Sciences and Humanities |  
| CSE        | Science and Engineering       |  
+------------+-------------------------------+  
4 rows in set (0.003 sec)  
  
MariaDB [toyu]> SELECT \* FROM department;  
+----------+------------------------------+------------+----------+  
| deptCode | deptName                     | schoolCode | numStaff |  
+----------+------------------------------+------------+----------+  
| ACCT     | Accounting                   | BUS        |       10 |  
| ARTS     | Arts                         | HSH        |        5 |  
| CINF     | Computer Information Systems | CSE        |        5 |  
| CSCI     | Computer Science             | CSE        |       12 |  
| ENGL     | English                      | HSH        |       12 |  
| ITEC     | Information Technology       | CSE        |        4 |  
| MATH     | Mathematics                  | CSE        |        7 |  
+----------+------------------------------+------------+----------+  
7 rows in set (0.000 sec)  
  
MariaDB [toyu]> SELECT \* FROM faculty;  
+-------+----------+----------+----------+---------------------+  
| facId | fname    | lname    | deptCode | rank                |  
+-------+----------+----------+----------+---------------------+  
|  1011 | Paul     | Smith    | CSCI     | Professor           |  
|  1012 | Mary     | Tran     | CSCI     | Associate Professor |  
|  1013 | David    | Love     | CSCI     | NULL                |  
|  1014 | Sharon   | Mannes   | CSCI     | Assistant Professor |  
|  1015 | Daniel   | Kim      | CINF     | Professor           |  
|  1016 | Andrew   | Byre     | CINF     | Associate Professor |  
|  1017 | Deborah  | Gump     | ITEC     | Professor           |  
|  1018 | Art      | Allister | ARTS     | Assistant Professor |  
|  1019 | Benjamin | Yu       | ITEC     | Lecturer            |  
|  1020 | Katrina  | Bajaj    | ENGL     | Lecturer            |  
|  1021 | Jorginlo | Neymar   | ACCT     | Assistant Professor |  
+-------+----------+----------+----------+---------------------+  
11 rows in set (0.001 sec)  
  
MariaDB [toyu]> SELECT \* FROM course;  
+----------+--------+--------+-------------------------------------+---------+  
| courseId | rubric | number | title                               | credits |  
+----------+--------+--------+-------------------------------------+---------+  
|     2000 | CSCI   | 3333   | Data Structures                     |       3 |  
|     2001 | CSCI   | 4333   | Design of Database Systems          |       3 |  
|     2002 | CSCI   | 5333   | DBMS                                |       3 |  
|     2020 | CINF   | 3321   | Introduction to Information Systems |       3 |  
|     2021 | CINF   | 4320   | Web Application Development         |       3 |  
|     2040 | ITEC   | 3335   | Database Development                |       3 |  
|     2041 | ITEC   | 3312   | Introduction to Scripting           |       3 |  
|     2060 | ENGL   | 1410   | English I                           |       4 |  
|     2061 | ENGL   | 1311   | English II                          |       3 |  
|     2080 | ARTS   | 3311   | Hindu Arts                          |       3 |  
|     2090 | ACCT   | 3333   | Managerial Accounting               |       3 |  
+----------+--------+--------+-------------------------------------+---------+  
11 rows in set (0.000 sec)  
  
MariaDB [toyu]> SELECT \* FROM class;  
+---------+----------+----------+------+-------+------+  
| classId | courseId | semester | year | facId | room |  
+---------+----------+----------+------+-------+------+  
|   10000 |     2000 | Fall     | 2019 |  1011 | D241 |  
|   10001 |     2001 | Fall     | 2019 |  1011 | D242 |  
|   10002 |     2002 | Fall     | 2019 |  1012 | D136 |  
|   10003 |     2020 | Fall     | 2019 |  1014 | D241 |  
|   10004 |     2021 | Fall     | 2019 |  1014 | D241 |  
|   10005 |     2040 | Fall     | 2019 |  1015 | D237 |  
|   10006 |     2041 | Fall     | 2019 |  1019 | D217 |  
|   10007 |     2060 | Fall     | 2019 |  1020 | B101 |  
|   10008 |     2080 | Fall     | 2019 |  1018 | D241 |  
|   11000 |     2000 | Spring   | 2020 |  1011 | D241 |  
|   11001 |     2001 | Spring   | 2020 |  1012 | D242 |  
|   11002 |     2002 | Spring   | 2020 |  1013 | D136 |  
|   11003 |     2020 | Spring   | 2020 |  1016 | D217 |  
|   11004 |     2061 | Spring   | 2020 |  1018 | B101 |  
+---------+----------+----------+------+-------+------+  
14 rows in set (0.001 sec)  
  
MariaDB [toyu]> SELECT \* FROM student;  
+--------+-----------+----------+-------+-------+------+---------+  
| stuId  | fname     | lname    | major | minor | ach  | advisor |  
+--------+-----------+----------+-------+-------+------+---------+  
| 100000 | Tony      | Hawk     | CSCI  | CINF  |   40 |    1011 |  
| 100001 | Mary      | Hawk     | CSCI  | CINF  |   35 |    1011 |  
| 100002 | David     | Hawk     | CSCI  | ITEC  |   66 |    1012 |  
| 100003 | Catherine | Lim      | ITEC  | CINF  |   20 |    NULL |  
| 100004 | Larry     | Johnson  | ITEC  | NULL  |   66 |    1017 |  
| 100005 | Linda     | Johnson  | CINF  | ENGL  |   13 |    1015 |  
| 100006 | Lillian   | Johnson  | CINF  | ITEC  |   18 |    1016 |  
| 100007 | Ben       | Zico     | NULL  | NULL  |   16 |    NULL |  
| 100008 | Bill      | Ching    | ARTS  | NULL  |   90 |    NULL |  
| 100009 | Linda     | King     | ARTS  | CSCI  |  125 |    1018 |  
| 100111 | Cathy     | Johanson | NULL  | NULL  |    0 |    1018 |  
+--------+-----------+----------+-------+-------+------+---------+  
11 rows in set (0.001 sec)  
  
MariaDB [toyu]> SELECT \* FROM enroll;  
+--------+---------+-------+----------+  
| stuId  | classId | grade | n\_alerts |  
+--------+---------+-------+----------+  
| 100000 |   10000 | A     |        0 |  
| 100001 |   10000 | NULL  |     NULL |  
| 100002 |   10000 | B-    |        3 |  
| 100000 |   10001 | A     |        2 |  
| 100001 |   10001 | A-    |        0 |  
| 100000 |   10002 | B+    |        1 |  
| 100002 |   10002 | B+    |        2 |  
| 100000 |   10003 | C     |        0 |  
| 100002 |   10003 | D     |        4 |  
| 100004 |   10003 | A     |        0 |  
| 100005 |   10003 | NULL  |     NULL |  
| 100000 |   10004 | A-    |        1 |  
| 100004 |   10004 | B+    |     NULL |  
| 100005 |   10004 | A-    |        0 |  
| 100006 |   10004 | C+    |     NULL |  
| 100005 |   10005 | A-    |        0 |  
| 100006 |   10005 | A     |     NULL |  
| 100005 |   10006 | B+    |     NULL |  
| 100007 |   10007 | F     |        4 |  
| 100008 |   10007 | C-    |        0 |  
| 100007 |   10008 | A-    |        0 |  
| 100000 |   11001 | D     |        4 |  
+--------+---------+-------+----------+  
22 rows in set (0.000 sec)

***Example:***

Consider the following *instance* of the table department:

+----------+------------------------------+------------+----------+  
| deptCode | deptName                     | schoolCode | numStaff |  
+----------+------------------------------+------------+----------+  
| ACCT     | Accounting                   | BUS        |       10 |  
| ARTS     | Arts                         | HSH        |        5 |  
| CINF     | Computer Information Systems | CSE        |        5 |  
| CSCI     | Computer Science             | CSE        |       12 |  
| ENGL     | English                      | HSH        |       12 |  
| ITEC     | Information Technology       | CSE        |        4 |  
| MATH     | Mathematics                  | CSE        |        7 |  
+----------+------------------------------+------------+----------+  
7 rows in set (0.00 sec)

* The name of the table is 'department'.
* There are currently seven rows in the relation*instance*.
* Each row in the table 'department' has the same columns: deptCode, deptName, schoolCode and numStaff. This is the structure of the relation: the relation *schema*.
* The relation schema for department: department(deptCode, deptName, schoolCode, numStaff).
* Each column has a specific data type. Examples:
  1. deptCode: CHAR(4)
  2. deptName: VARCHAR(30)
  3. schoolCode: CHAR(3)
  4. numStaff: TINYINT
* We said the *domain* of the column deptCode is CHAR(4).
* A domain may be understood as the values allowed by the *data type*.
* Thus, the relation schema and column domains form the 'structure' of the database.
* The structures usually do not change much, just like the structure of a building.
* However, the structure can be changed (just like the structure of a building: remodeling).
* On the other hand, the content of a table (the relation *instance*) can be changed from time to time.

MariaDB [toyu]> desc department;  
+------------+-------------+------+-----+---------+-------+  
| Field      | Type        | Null | Key | Default | Extra |  
+------------+-------------+------+-----+---------+-------+  
| deptCode   | char(4)     | NO   | PRI | NULL    |       |  
| deptName   | varchar(30) | NO   | UNI | NULL    |       |  
| schoolCode | char(3)     | YES  | MUL | NULL    |       |  
| numStaff   | tinyint(4)  | YES  |     | NULL    |       |  
+------------+-------------+------+-----+---------+-------+  
4 rows in set (0.005 sec)  
  
Note that 'KEY' and 'INDEX' have the same meaning in MySQL.

* Key: PRI; primary key/index
* Key: UNI; unique key/index
* key: MUL; multiple value index.

***Example:***

After inserting a new row:

INSERT INTO department VALUES ('PHYS', 'Physics','CSE',3);

MariaDB [toyu]> INSERT INTO department VALUES ('PHYS', 'Physics','CSE',3);  
Query OK, 1 row affected (0.012 sec)

The new relation instance of the table department:

MariaDB [toyu]> SELECT \* from department;  
+----------+------------------------------+------------+----------+  
| deptCode | deptName                     | schoolCode | numStaff |  
+----------+------------------------------+------------+----------+  
| ACCT     | Accounting                   | BUS        |       10 |  
| ARTS     | Arts                         | HSH        |        5 |  
| CINF     | Computer Information Systems | CSE        |        5 |  
| CSCI     | Computer Science             | CSE        |       12 |  
| ENGL     | English                      | HSH        |       12 |  
| ITEC     | Information Technology       | CSE        |        4 |  
| MATH     | Mathematics                  | CSE        |        7 |  
| PHYS     | Physics                      | CSE        |        3 |  
+----------+------------------------------+------------+----------+  
8 rows in set (0.000 sec)  
  
After:

DELETE FROM department WHERE deptCode = 'PHYS';

MariaDB [toyu]> DELETE FROM department WHERE deptCode = 'PHYS';  
Query OK, 1 row affected (0.009 sec)

The relation instance reverts back.

MariaDB [toyu]> SELECT \* from department;  
+----------+------------------------------+------------+----------+  
| deptCode | deptName                     | schoolCode | numStaff |  
+----------+------------------------------+------------+----------+  
| ACCT     | Accounting                   | BUS        |       10 |  
| ARTS     | Arts                         | HSH        |        5 |  
| CINF     | Computer Information Systems | CSE        |        5 |  
| CSCI     | Computer Science             | CSE        |       12 |  
| ENGL     | English                      | HSH        |       12 |  
| ITEC     | Information Technology       | CSE        |        4 |  
| MATH     | Mathematics                  | CSE        |        7 |  
+----------+------------------------------+------------+----------+  
7 rows in set (0.000 sec)

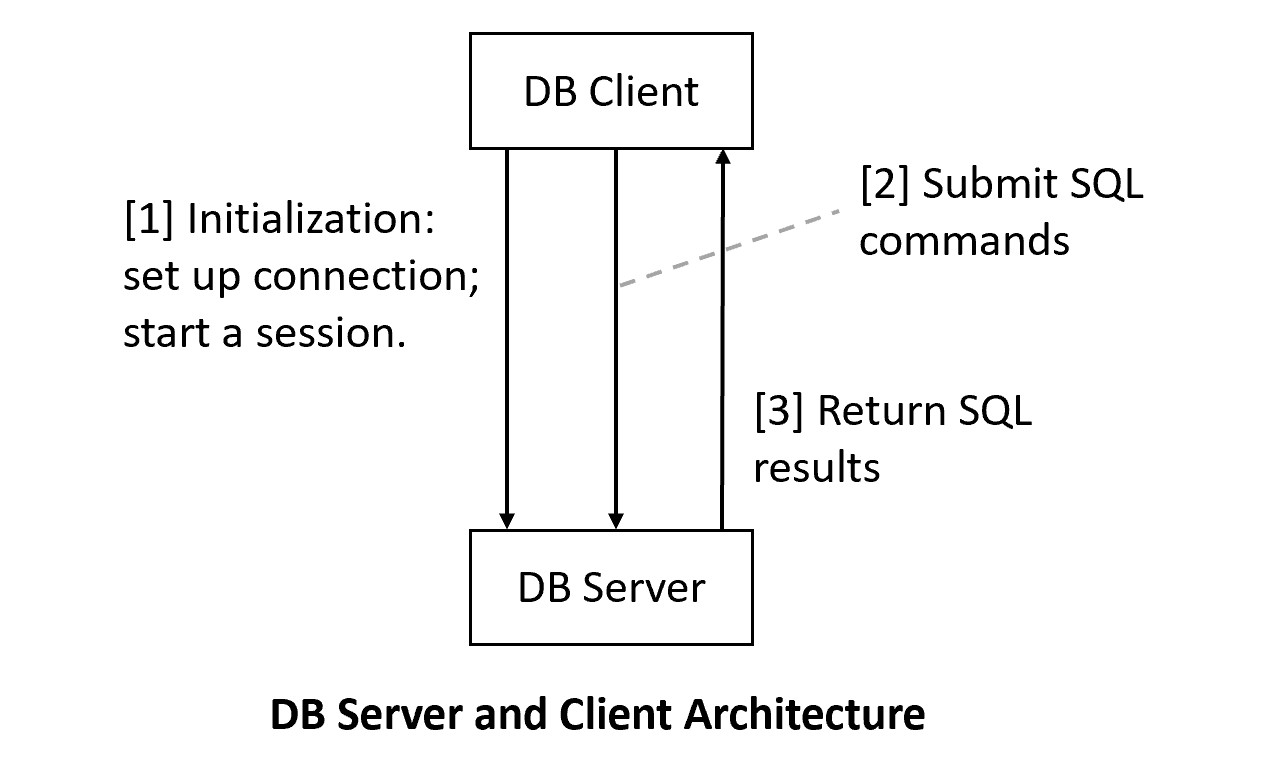
* Note the terms table and relation can be ambiguous. They may mean:
  1. the relation schema: the 'structure' of the table.
  2. the relation instance: actual data in the relation at a specific time.
* Note that relations are a logical concept to reason with.
* Relations must eventually be stored in the file systems to ensure persistence.
* For example, a row *may* be stored as a record, in which a column is a *field* of the record. A row is also known as a *tuple* in the relational model.
* Conceptually, a row may be used to represent (or store information about) an entity, a relationship, or a concept.
  1. A student record may represent a student (entity)
  2. An enrollment record may represent a relationship: a student taking a class.
* Tables may be 'connected' through unique identifiers, known asforeign keys. Example:
  1. The enroll table is related to the student table through stuId:

MariaDB [toyu]> SELECT \* FROM student;  
+--------+-----------+----------+-------+-------+------+---------+  
| *stuId*  | fname     | lname    | major | minor | ach  | advisor |  
+--------+-----------+----------+-------+-------+------+---------+  
| 100000 | Tony      | Hawk     | CSCI  | CINF  |   40 |    1011 |  
| 100001 | Mary      | Hawk     | CSCI  | CINF  |   35 |    1011 |  
| 100002 | David     | Hawk     | CSCI  | ITEC  |   66 |    1012 |  
| 100003 | Catherine | Lim      | ITEC  | CINF  |   20 |    NULL |  
| 100004 | Larry     | Johnson  | ITEC  | NULL  |   66 |    1017 |  
| 100005 | Linda     | Johnson  | CINF  | ENGL  |   13 |    1015 |  
| 100006 | Lillian   | Johnson  | CINF  | ITEC  |   18 |    1016 |  
| 100007 | Ben       | Zico     | NULL  | NULL  |   16 |    NULL |  
| 100008 | Bill      | Ching    | ARTS  | NULL  |   90 |    NULL |  
| 100009 | Linda     | King     | ARTS  | CSCI  |  125 |    1018 |  
| 100111 | Cathy     | Johanson | NULL  | NULL  |    0 |    1018 |  
+--------+-----------+----------+-------+-------+------+---------+  
11 rows in set (0.001 sec)  
  
MariaDB [toyu]> SELECT \* FROM enroll;  
+--------+---------+-------+----------+  
| *stuId*  | classId | grade | n\_alerts |  
+--------+---------+-------+----------+  
| 100000 |   10000 | A     |        0 |  
| 100001 |   10000 | NULL  |     NULL |  
| 100002 |   10000 | B-    |        3 |  
| 100000 |   10001 | A     |        2 |  
| 100001 |   10001 | A-    |        0 |  
| 100000 |   10002 | B+    |        1 |  
| 100002 |   10002 | B+    |        2 |  
| 100000 |   10003 | C     |        0 |  
| 100002 |   10003 | D     |        4 |  
| 100004 |   10003 | A     |        0 |  
| 100005 |   10003 | NULL  |     NULL |  
| 100000 |   10004 | A-    |        1 |  
| 100004 |   10004 | B+    |     NULL |  
| 100005 |   10004 | A-    |        0 |  
| 100006 |   10004 | C+    |     NULL |  
| 100005 |   10005 | A-    |        0 |  
| 100006 |   10005 | A     |     NULL |  
| 100005 |   10006 | B+    |     NULL |  
| 100007 |   10007 | F     |        4 |  
| 100008 |   10007 | C-    |        0 |  
| 100007 |   10008 | A-    |        0 |  
| 100000 |   11001 | D     |        4 |  
+--------+---------+-------+----------+  
22 rows in set (0.000 sec)

**3.SQL and MySQL**

* The standard query language for RDBMS is Structured Query Language (SQL).
* We use MySQL (or MariaDB) in this class.

DBMS mostly uses a client-server architecture.



**3.1 MySQL Server Setup**

We will use MariaDB that is a part of XAMPP. Do not recommend installing standalone MySQL.

[1] Install XAMPP, which contains many server software configured to work together for development purpose. For XAMPP, we will use MySQL/Maria DB and Apache (Web server).

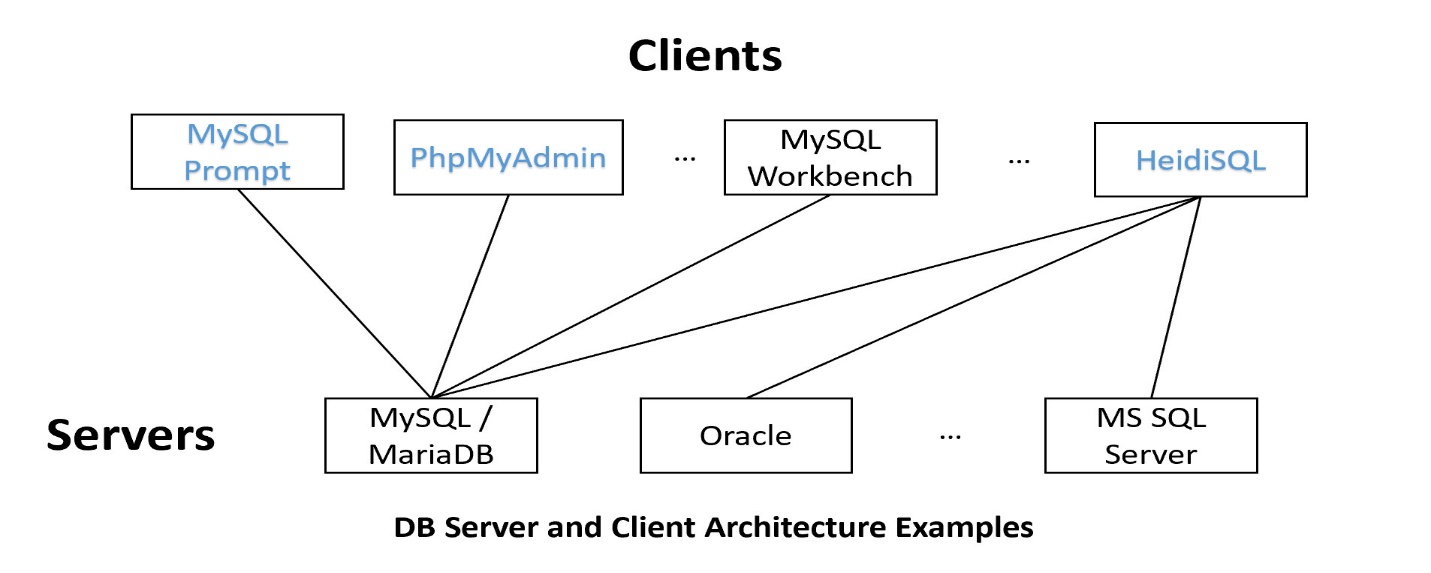
1. Recommended to install XAMPP in the*top* level: c:\xampp (likely the default).
2. Set up development accounts immediately using phpMyAdmin after installation.
3. Change the root password (optional but recommended): a secure step that requires tinkering.
4. To ensure that PHPMyAdmin will work on a new admin account (optional):
   1. Use PHPMyAdmin to create a new admin account "frog\_ad", with the password "a\_new\_prince" for both hostname '%' and 'localhost'
   2. PhpMyAdmin uses the default root account (with no initial password) via localhost.
   3. Thus, you will need to supply the new username and password to start up PhpMyAdmin by editing the file c:\xampp\phpMyAdmin\config.inc.php, search change the line to, for example:
      1. $cfg['Servers'][$i]['user'] = 'frog\_ad';
      2. $cfg['Servers'][$i]['password'] = 'a\_new\_prince';

**MySQL Clients Setup:**

It is common to use multiple clients to connect to the backend database server. In this course, we will use three clients in our classes. You may use your own favorite clients (e.g., MySQL Workbench). However, I may not be as helpful in these clients.

Working directory:

C:\xampp\mysql\bin\mysql.exe: MySQL command line prompt



HeidiSQL client:

A screenshot of a computer

Description automatically generated

MysqL server:

A screenshot of a computer

Description automatically generated

[1] MySQL Command-Line Prompt: will be used in this class.

1. Come with (1) XAMPP/MariaDB or (2) MySQL 8.x. (Note that the two versions of mysql prompt are somewhat different.)
   1. MariaDB mysql: <https://mariadb.com/kb/en/mysql-command-line-client/>
   2. MySQL 8.x mysql: <https://dev.mysql.com/doc/refman/8.0/en/mysql.html>
2. A command line text-based MySQL-specific client.
3. You may set the PATH variable so you can call mysql prompt anywhere, such as by adding "c:\xampp\mysql\bin" in the PATH system environment variable.

***Example:***

**mysql –h host -u user -p**

or

**mysql –h host -u user -p -P port\_number**

[2] PhPMyAdmin

1. A Web-based GUI client focused on DB administration.
2. After starting both MySQL and Apache in XAMPP, go to localhost in your browser.
3. MySQL specific.

[3] HeidiSQL: will be used in this class

1. A general Windows GUI SQL client

[4] MySQL Workbench:

1. A GUI MySQL client that comes with MySQL 8.x (but not XAMPP)

**SQL**

* Note that different DBMS support different versions of SQL. They may not fully support the standards and may include extensions.
* For example, older versions of MySQL do not support EXCEPT, which is in ANSI SQL 92.
* When developing databases in a given DBMS, portability and backward compatibility issues are a consideration for your choice of SQL statements.
* MySQL is not based on a pure relational model. For example:
  + Query results may contain duplicate rows.
  + Order may be important.
* SQL is easy to start with. Not so easy to become an expert.
* SQL is a rich and essentially declarative but with some procedural constructs.
* For the time being, we only need to know the basic form of the SELECT statement to get some taste of SQL and RDBMS.

SELECT DISTINCT <<result\_columns>>  
FROM <<source\_tables>> -- conceptually joined to form a large table  
WHERE <<conditions\_for\_inclusion>>

**Declarative Analysis**

1. <<source\_tables>>: the source tables to gather the result data
2. <<conditions\_for\_inclusion>>: the conditions to be satisfied for results to be included and the conditions the tables should be connected together.
3. <<result\_columns>>: the result columns or expressions desired to be displayed.

***Examples:***

Using toyu, executing the SQL:

-- Department codes and their names  
SELECT DISTINCT deptCode, deptName  
FROM department;  
  
-- Faculty information from the department 'CSCI'  
SELECT DISTINCT \*  
FROM faculty  
WHERE deptCode = 'CSCI';  
  
-- Faculty names from the department 'CSCI'  
SELECT DISTINCT fname, lname  
FROM faculty  
WHERE deptCode = 'CSCI';  
  
-- Associate professor names from the department 'CSCI'.  
SELECT DISTINCT fname, lname  
FROM faculty  
WHERE deptCode = 'CSCI'  
AND `rank` = 'Associate Professor';  
  
-- Department names and numbers of faculty with the numbers  
SELECT DISTINCT deptName, numStaff  
FROM department  
WHERE numStaff >= 10;  
  
-- Names of all faculty members and their  
-- department names and ranks.  
SELECT DISTINCT faculty.fName, faculty.lname,  
   department.deptName, faculty.`rank`  
FROM department, faculty  
WHERE faculty.deptCode = department.deptCode;  
  
SELECT DISTINCT faculty.fName, faculty.lname,  
   department.deptName, faculty.`rank`  
FROM department INNER JOIN faculty ON (faculty.deptCode = department.deptCode);

Result:  
  
MariaDB [toyu]> -- Department codes and their names  
MariaDB [toyu]> SELECT DISTINCT deptCode, deptName  
    -> FROM department;  
+----------+------------------------------+  
| deptCode | deptName                     |  
+----------+------------------------------+  
| ACCT     | Accounting                   |  
| ARTS     | Arts                         |  
| CINF     | Computer Information Systems |  
| CSCI     | Computer Science             |  
| ENGL     | English                      |  
| ITEC     | Information Technology       |  
| MATH     | Mathematics                  |  
+----------+------------------------------+  
7 rows in set (0.000 sec)  
  
MariaDB [toyu]>  
MariaDB [toyu]> -- Faculty information from the department 'CSCI'  
MariaDB [toyu]> SELECT DISTINCT \*  
    -> FROM faculty  
    -> WHERE deptCode = 'CSCI';  
+-------+--------+--------+----------+---------------------+  
| facId | fname  | lname  | deptCode | rank                |  
+-------+--------+--------+----------+---------------------+  
|  1011 | Paul   | Smith  | CSCI     | Professor           |  
|  1012 | Mary   | Tran   | CSCI     | Associate Professor |  
|  1013 | David  | Love   | CSCI     | NULL                |  
|  1014 | Sharon | Mannes | CSCI     | Assistant Professor |  
+-------+--------+--------+----------+---------------------+  
4 rows in set (0.000 sec)  
  
MariaDB [toyu]>  
MariaDB [toyu]> -- Faculty names from the department 'CSCI'  
MariaDB [toyu]> SELECT DISTINCT fname, lname  
    -> FROM faculty  
    -> WHERE deptCode = 'CSCI';  
+--------+--------+  
| fname  | lname  |  
+--------+--------+  
| Paul   | Smith  |  
| Mary   | Tran   |  
| David  | Love   |  
| Sharon | Mannes |  
+--------+--------+  
4 rows in set (0.000 sec)  
  
MariaDB [toyu]>  
MariaDB [toyu]> -- Associate professor names from the department 'CSCI'  
MariaDB [toyu]> SELECT DISTINCT fname, lname  
    -> FROM faculty  
    -> WHERE deptCode = 'CSCI'  
    -> AND `rank` = 'Associate Professor';  
+-------+-------+  
| fname | lname |  
+-------+-------+  
| Mary  | Tran  |  
+-------+-------+  
1 row in set (0.000 sec)  
  
MariaDB [toyu]>  
MariaDB [toyu]> -- Department names and numbers of faculty with the numbers  
MariaDB [toyu]> SELECT DISTINCT deptName, numStaff  
    -> FROM department  
    -> WHERE numStaff >= 10;  
+------------------+----------+  
| deptName         | numStaff |  
+------------------+----------+  
| Accounting       |       10 |  
| Computer Science |       12 |  
| English          |       12 |  
+------------------+----------+  
3 rows in set (0.002 sec)  
  
MariaDB [toyu]>  
MariaDB [toyu]> -- Names of all faculty members and their  
MariaDB [toyu]> -- department names and ranks.  
MariaDB [toyu]> SELECT DISTINCT faculty.fName, faculty.lname,  
    -> department.deptName, faculty.`rank`  
    -> FROM department, faculty  
    -> WHERE faculty.deptCode = department.deptCode;  
+----------+----------+------------------------------+---------------------+  
| fName    | lname    | deptName                     | rank                |  
+----------+----------+------------------------------+---------------------+  
| Paul     | Smith    | Computer Science             | Professor           |  
| Mary     | Tran     | Computer Science             | Associate Professor |  
| David    | Love     | Computer Science             | NULL                |  
| Sharon   | Mannes   | Computer Science             | Assistant Professor |  
| Daniel   | Kim      | Computer Information Systems | Professor           |  
| Andrew   | Byre     | Computer Information Systems | Associate Professor |  
| Deborah  | Gump     | Information Technology       | Professor           |  
| Art      | Allister | Arts                         | Assistant Professor |  
| Benjamin | Yu       | Information Technology       | Lecturer            |  
| Katrina  | Bajaj    | English                      | Lecturer            |  
| Jorginlo | Neymar   | Accounting                   | Assistant Professor |  
+----------+----------+------------------------------+---------------------+  
11 rows in set (0.000 sec)  
  
MariaDB [toyu]>  
MariaDB [toyu]> SELECT DISTINCT faculty.fName, faculty.lname,  
    -> department.deptName, faculty.`rank`  
    -> FROM department INNER JOIN faculty ON (faculty.deptCode = department.deptCode);  
+----------+----------+------------------------------+---------------------+  
| fName    | lname    | deptName                     | rank                |  
+----------+----------+------------------------------+---------------------+  
| Paul     | Smith    | Computer Science             | Professor           |  
| Mary     | Tran     | Computer Science             | Associate Professor |  
| David    | Love     | Computer Science             | NULL                |  
| Sharon   | Mannes   | Computer Science             | Assistant Professor |  
| Daniel   | Kim      | Computer Information Systems | Professor           |  
| Andrew   | Byre     | Computer Information Systems | Associate Professor |  
| Deborah  | Gump     | Information Technology       | Professor           |  
| Art      | Allister | Arts                         | Assistant Professor |  
| Benjamin | Yu       | Information Technology       | Lecturer            |  
| Katrina  | Bajaj    | English                      | Lecturer            |  
| Jorginlo | Neymar   | Accounting                   | Assistant Professor |  
+----------+----------+------------------------------+---------------------+  
11 rows in set (0.000 sec)

***Classroom Demonstration and Exercises:***

***Example***. Show all student names.

Expected Result:

+---------+-----------+  
| lname   | fname     |  
+---------+-----------+  
| Hawk    | Tony      |  
| Hawk    | Mary      |  
| Hawk    | David     |  
| Lim     | Catherine |  
| Johnson | Larry     |  
| Johnson | Linda     |  
| Johnson | Lillian   |  
| Zico    | Ben       |  
| Ching   | Bill      |  
| King    | Linda     |  
+---------+-----------+  
10 rows in set (0.00 sec)

Declarative Analysis:

[1] Sources: student  
[2] Conditions: none  
[3] Output fields: lname, fname

SELECT DISTINCT s.fname, s.lname  
FROM student AS s -- s is the alias of student

***Example.*** List the last names and first names of students minoring in CINF and having 1011 as faculty advisor.

+-------+-------+  
| lname | fname |  
+-------+-------+  
| Hawk  | Tony  |  
| Hawk  | Mary  |  
+-------+-------+  
2 rows in set (0.001 sec)

[Analysis]

[1] Source tables: student

[2] Conditions:

1. minor = 'CINF'
2. advisor = 1011

[3] Output columns:

1. lname
2. fname

[suggested solution of sample question]  
  
SELECT DISTINCT s.fname, s.lname  
FROM student AS s -- s is the alias of student  
WHERE s.minor = 'CINF'  
AND s.advisor = 1011;

***Classroom examples:***

1. All student names and the major codes.  
2. All student names and the major department names.  
3. All student names enrolled in the class with id 10003.  
4. Show all information of students majoring in ‘MATH’.  
5. Show the names and credits of students majoring in 'CSCI'.  
6. Show the names and credits of students majoring in 'CSCI' and having 40 or more ach credits.  
7. Show the id of students enrolled in the course CSCI 4333.  
8. Show the code of departments with faculty in the rank of 'Professor'.  
9. Show the names of departments with faculty in the rank of 'Professor'.  
10. Show the names of students who have enrolled in the course CSCI 4333.  
11. Show the names and major names of every student.  
12. Show the names, major names, and advisor names of every student.  
11. Show the student names and their major names for all students who have received a grade A in a class offered by a faculty from the CSCI department.

Query

(a) Show the stuId, names and advisorId of all students in the following format. The expected result:

A screenshot of a computer

Description automatically generated with medium confidence

Declarative Analysis (in contrast algorithmic analysis):

[1] Output columns (label: value)

1. stuId: student.stuId
2. student: student.fname & ‘ ‘ & student.lname (expression: +: string concatenation.)
3. major: student.major
4. AdvisorId: student.advsior

[2] Source tables:

1. student

[3] Conditions: no condition

SQL:

SELECT DISTINCT   
-- columns  
FROM  
-- source tables  
WHERE   
-- condition

HW #2, Fall 2023:

[2] List the names of all departments together with their faculty members' names and ranks of the School 'Science and Engineering' in the following format. You should not use 'CSE' in your query.

+---------------+---------------------+------------------------------+

| faculty | rank | department |

+---------------+---------------------+------------------------------+

| Paul Smith | Professor | Computer Science |

| Mary Tran | Associate Professor | Computer Science |

| David Love | NULL | Computer Science |

| Sharon Mannes | Assistant Professor | Computer Science |

| Daniel Kim | Professor | Computer Information Systems |

| Andrew Byre | Associate Professor | Computer Information Systems |

| Deborah Gump | Professor | Information Technology |

| Benjamin Yu | Lecturer | Information Technology |

+---------------+---------------------+------------------------------+

8 rows in set

Declarative Analysis:

[1] Column:

1. CONCAT(f.fname, ‘ ‘, f.lname) AS faulty
2. f.rank
3. d.deptName AS department

[2] Source:

1. faculty AS f
2. department AS d
3. school AS s

[3] Conditions:

1. Join conditions:
   1. (FK) f.deptCode = d.deptCode (PK)
   2. (FK) d.schoolCode = s.schoolCode (PK)
2. Problem conditions: the School 'Science and Engineering' in the following format. You should not use 'CSE' in your query.
   1. S.schoolName = 'Science and Engineering'