UNIVERSITY *of* WASHINGTON

Week 3: Intro to SQL

AGENDA

- Homework Overview
- MariaDB
- Introduction to SQL
  - Creating/Altering/Dropping Tables
  - Inserting Tuples
  - Selection, Joins

HOMEWORK 1

Overall good job!

I will have grades posted this weekend.

Let’s talk about solutions.

UNIVERSITY of WASHINGTON

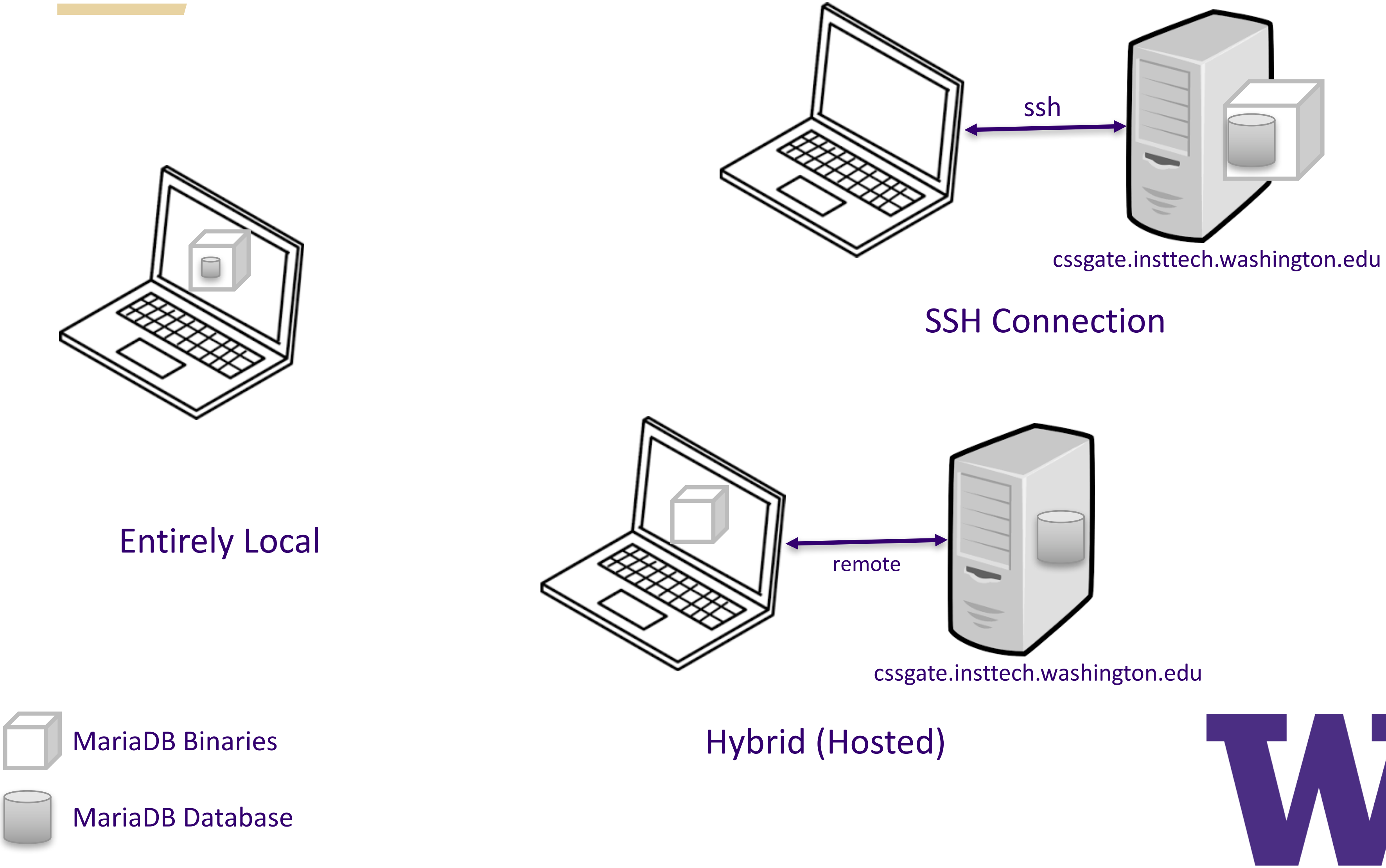
Introduction to RDBMs: MariaDB

Which RDBMS will we use?

- MariaDB: Drop-in replacement of MySQL
  - Open Source
  - ”Compatible” with MySQL GUIs such as MySQL Workbench, Navicat, Toad, etc.
- ”Created” when Oracle acquired MySQL

https://mariadb.com/kb/en/mariadb/documentation/
Google: “Maria DB Documentation”

Client/Server vs Local



ssh

cssgate.insttech.washington.edu

SSH Connection

Entirely Local

remote

cssgate.insttech.washington.edu

Hybrid (Hosted)

MariaDB Binaries

MariaDB Database

Entirely Local

- Install MariaDB
- Configure your root level access
- Act as your own DBA, create schemas, databases etc, play with DBMS level settings

- To connect: localhost (127.0.0.1)
- Make sure your DB Server is running.

Run entirely from cssgate

- cssgate.insttech.washington.edu runs a version of MariaDB and allows for remote connections
- Can control via Terminal or PUTTY vis SSH
- ssh uwnetid@cssgate.insttech.washington.edu

- Or connect via GUI (explained later)

Hybrid

- Install MariaDB
- Connect to cssgate as a remote server

```
mysql -u uwnet -h cssgate.insttech.washington.edu –p
```

- Use your sql password (obtained from ~/.pw file)

MariaDB GUI / IDE

> Many options for your IDE
  - TOAD, MySQL Workbench, NaviCat to name a few…

> We are going to use JetBrains: IntelliJ iDEA
  - Free for Students

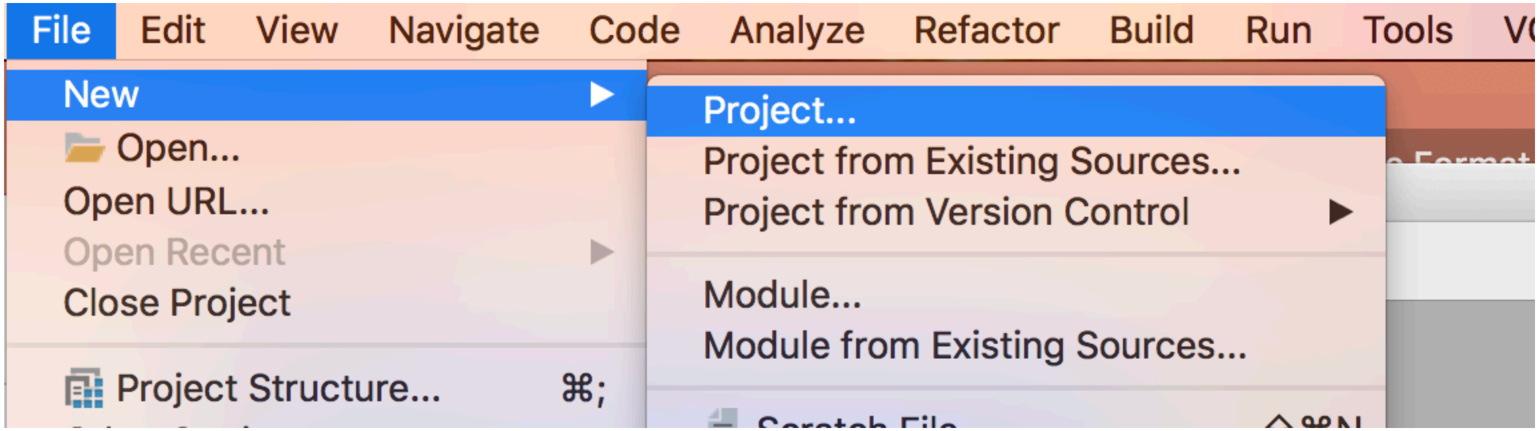
If you don’t already have a student license…

https://www.jetbrains.com/student/

Apply for License
Confirm Email Address
Activate Account
Create Username
Download and Install IntelliJ IDEA Ultimate

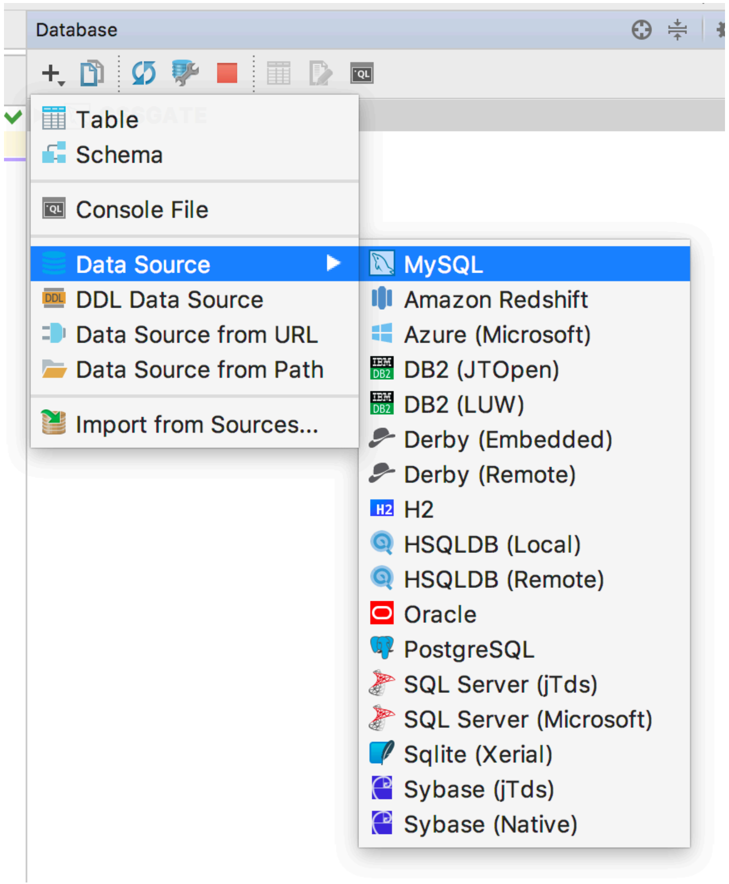
Configuring your GUI

- Create a new Project for TCSS445
  - If not prompted:

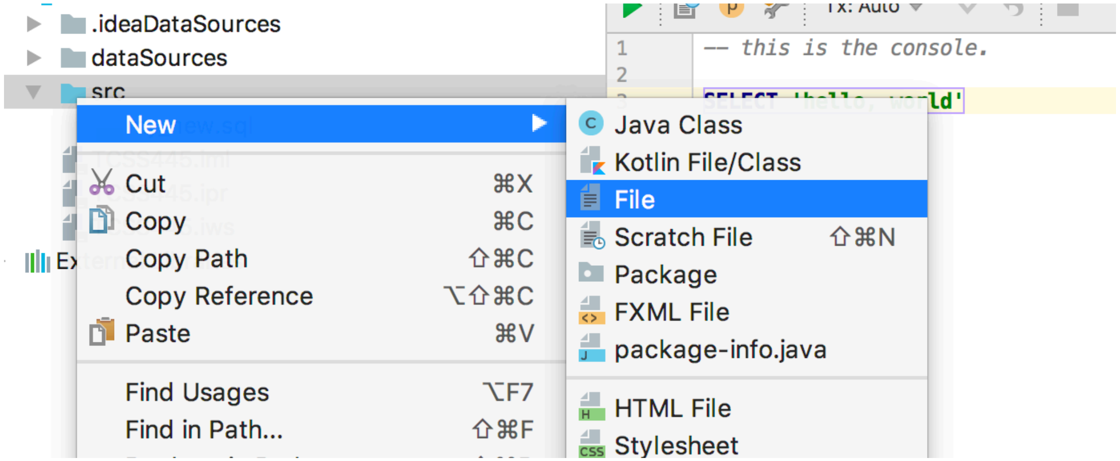


Configuring your GUI

> Tool Window > Database

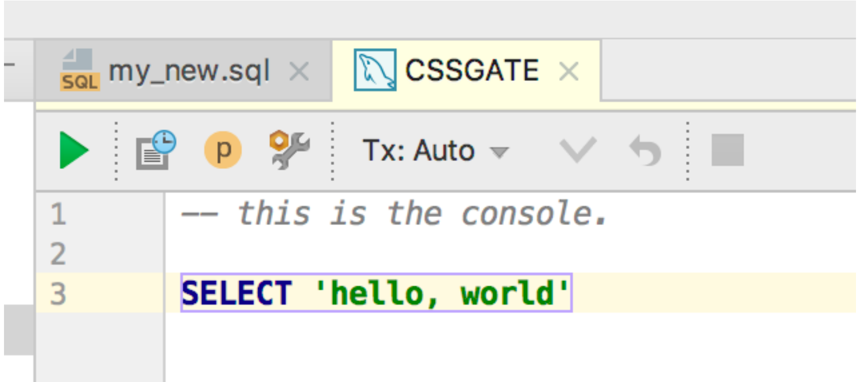


Configuring your GUI



Save queries/scripts as .sql files.

Interact without saving files using the console…



```
-- this is the console.

SELECT 'hello, world'
```

UNIVERSITY of WASHINGTON

INTRO TO SQL

Introduction to SQL

- Based on Relational Algebra
- “SQL” rooted from a ’standard’ (current SQL-11)
- SQL standard also referred to as ANSI-SQL
- SQL Server, Oracle, MariaDB/MySQL, all are different dialects of ANSI SQL

Types of SQL Statements

- DDL (Data Definition Language)
  - “Create, Alter, Drop”
  - Defines the schema and functional dependencies
- DML (Data Manipulation Language)
  - “Select, Insert, Update, Delete”
  - Retrieves manipulates data within constraints of the schema. DML cannot function outside of what is defined by DDL.
- DCL (Data Control Language) *not in your book*
  - “Grant, Revoke”
  - Provides (or denies) users rights to data

Relations, Tables, Views, Temp

- RDBMSs implement **relations** in several ways:
  - **Tables**: Persisted data. Arranged in a set way on a disk.
  - **Views**: Persisted “Queries.” Relation is created when requested.
  - **Temp Tables**: Cached Data (in memory). Can be stored globally, per session, or per query (behavior varies between RDBMS.

Creating Tables

```
CREATE TABLE <name> (
   <colname> <datatype> <options>
  ,<colname> <datatype> <options>
  ,...
  ,PRIMARY KEY(<colname>,<colname>)
);
```

For complete syntax reference for MariaDB:
https://mariadb.com/kb/en/library/create-table/

Tips for Table Naming

- Choose a convention and stick to it!
  - Most common casings:
    - MACRO_CASE
    - PascalCase
    - snake_case
  - Rarely:
    - kebab-case
    - camelCase
  - Unless you're using MS Access avoid Hungarian Notation!
    - tblData

> *"The wonderful thing about standards is that there are so many of them."*
> -Grace Murray Hopper

- Avoid Abbreviations unless completely obvious! (Unless using Oracle)
  - Employee (not Emp)
  - Transactions (not Txs)
  - DATE_DIM (DIM for dimension is common/acceptable)

- Singular Name, not Plural
  - It's assumed that there will be multiple Employees in the Employee table.

More on Singular Vs Plural

> In your text: Movies and MovieStar are both tables in the same schema.

> If you knew you had a movie table and a movie star table, you might try:
  select * from Movie;   or
  select * from MovieStars

> Some notable exceptions: People is a common table (people is plural, it isn't the Person table usually

Back to Creating Tables…

```
CREATE TABLE Employee (
    <colname> <datatype> <options>
   ,<colname> <datatype> <options>
   ,...
   ,PRIMARY KEY(<colname>,<colname>,...)
);
```

General Tips for Column Names

- Stick to a convention!
  - TableNameId is a common Primary Key as is “ID”
- Common Casings:
  - PascalCase
  - MACRO_CASE
  - snake_case
- Be specific, Abbreviations are more acceptable than in Table names, but avoid if possible.
- Avoid Reserved words like “SELECT” and “KEY”

Column Names, how descriptive?

- Which is "better?"
  - Transaction.processing_time
  - Transaction.processing_time_ms

- Which is "better?"
  - Product.height
  - Product.height_inches

- Alternatives?

Column Names, how descriptive?

- Transaction.processing_time
- Transaction.processing_time_unit

- Product.height
- Product.height_unit

Back to Creating Tables...

```
CREATE TABLE Employee (
   EmployeeId <datatype> <options>
  ,<colname> <datatype> <options>
  ,...
  ,PRIMARY KEY(<colname>,<colname>,...)
);
```

Common Data Types

TINYINT (-128 to 127)	1 byte
BOOLEAN (0 & 1)	1 byte
INT (-2^{31} to $2^{31}-1$)	4 bytes

FLOAT(M,D)
M = total digits,
D = digits after the decimal point
	M <= 25 ? (4bytes) : (8bytes)

CHAR(M)	M * bytes in char set
VARCHAR(M) $M < 2^{16} - 1$	M * bytes per char + 2 bytes
TIME (microsecond precision)	3 bytes
DATE (1000-01-01 to 9999-12-31)	3 bytes
DATETIME (1000-01-01 to 9999-12-31)	8 bytes

Full list: https://mariadb.com/kb/en/mariadb/data-types/

DataTypes

- Size matters
  - Storage is cheap? Why does it matter?

- Usage matters
  - How is this attribute going to be used?
  - 20170121 (3bytes MED INT)
  - 2017-01-21 12:00: AM (4bytes, datetime)

SOME OF OUR OPTIONS

```
CREATE TABLE Employee (
   EmployeeId INT <options>
  ,FirstName VARCHAR(50) <options>
  ,PRIMARY KEY(<colname>,<colname>...)
);
```

DEFAULT VALUES

- Make the database do some of the heavy lifting
- Can speed up INSERT statements

```
CREATE TABLE Employee (
   EmployeeId INT
  ,FirstName VARCHAR(50) DEFAULT 'Unknown'
  ,RecordCreateDate TIMESTAMP
     DEFAULT CURRENT_TIMESTAMP
  ,PRIMARY KEY(<colname>,<colname>,...)
);
```

NULL VS NOT NULL

```
CREATE TABLE Employee (
   EmployeeId INT
  ,FirstName VARCHAR(50) NOT NULL
  ,RecordCreateDate TIMESTAMP
     DEFAULT CURRENT_TIMESTAMP
  ,PRIMARY KEY(<colname>,<colname>,...)
);
```

Primary Keys

```
CREATE TABLE Employee (
   EmployeeId INT
  ,FirstName VARCHAR(50) NOT NULL
  ,RecordCreateDate TIMESTAMP
     DEFAULT CURRENT_TIMESTAMP
  ,PRIMARY KEY(<colname>,<colname>,...)
);
```

PRIMARY KEYS

> Must be unique!
> Cannot be NULL
> Often how the table is sorted on disk
> Smaller the better
> “Natural vs Generated|Artificial Keys?”
> CHAR|VARCHAR|INT|BIGINT?!
> AUTO_INCREMENT

UNIQUE

- UNIQUES MUST BE UNIQUE
- BUT CAN BE NULL

CREATED TABLE

```
CREATE TABLE Employee (
   EmployeeId INT
  ,FirstName VARCHAR(50) NOT NULL
  ,RecordCreateDate TIMESTAMP
     DEFAULT CURRENT_TIMESTAMP
  ,PRIMARY KEY(EmployeeId)
);
```

CREATED TABLE

```
CREATE TABLE Employee (
   EmployeeId INT PRIMARY KEY AUTO_INCREMENT
  ,FirstName VARCHAR(50) NOT NULL
  ,RecordCreateDate TIMESTAMP
     DEFAULT CURRENT_TIMESTAMP
);
```

DROPPING A TABLE

- DROP TABLE <tablename>;

ALTERING A TABLE

```
ALTER TABLE <tablename>
  ADD col_name col_def
    FIRST | AFTER col_name

ALTER TABLE Employee
  ADD LastName VARCHAR(50) DEFAULT 'unknown'
    AFTER FirstName
```

https://mariadb.com/kb/en/mariadb/alter-table/

INSERTING RECORDS (BASIC)

```
INSERT INTO <TABLENAME>(<COLNAME>,...) VALUES
  (TUPLE1, ...)
, (TUPLE2, ...)
```

```
INSERT INTO Employee VALUES
  (1, 'Jane', 'Doe', CURRENT_TIMESTAMP)
, (2, 'John', 'Doe', CURRENT_TIMESTAMP)
```

```
INSERT INTO Employee (EmployeeId, FirstName, LastName) VALUES
  (1, 'Jane', 'Doe')
, (2, 'John', 'Doe')
```

INSERTING WITH DEFAULTS AND AUTO_INCREMENT

```
INSERT INTO Employee (FirstName, LastName) VALUES
  ('Homer','Simpson')
, ('Marge','Simpson')
, ('Maggie','Simpson')
, ('Chief','Wiggum')
```

```
SELECT * FROM Employee
```

EmployeeId	FirstName	LastName	RecordCreateDate
1	Homer	Simpson	2017-01-20 22:10:00 AM
2	Marge	Simpson	2017-01-20 22:10:00 AM
3	Maggie	Simpson	2017-01-20 22:10:00 AM
4	Chief	Wiggum	2017-01-20 22:10:00 AM

UNIVERSITY *of* WASHINGTON

DML: SELECTION

INTRO TO SELECTION

SELECT	[COLUMNS & EXPRESSIONS]
FROM	[TABLE|VIEW]
JOIN	[TABLE|VIEW]
ON	[CONDITIONS]
WHERE	[CONDITIONS]
GROUP BY	[COLUMNS & EXPRESSIONS]
HAVING	[CONDITIONS]
ORDER BY	[COLUMNS & EXPRESSIONS]

INTRO TO SELECTION

SELECT	[COLUMNS]	π
FROM	[TABLE]	(Relation)
JOIN	[TABLE]	$\bowtie$
WHERE	[CONDITIONS]	σ
GROUP BY	[COLUMNS]	γ
HAVING	[CONDITIONS]	σ
ORDER BY	[COLUMNS]	τ

SELECT (Projection)

```
SELECT <COLUMNS|EXPRESSIONS>

SELECT CURRENT_TIMESTAMP

SELECT 10*100.0

SELECT *
```

Case Sensitivity

- In your book, it states “SQL is Case Insensitive”
  - Not true in all implementations...

- Case sensitive file systems can result in case sensitive objects and values.

- The keywords are NOT generally case-sensitive:

```
CREATE TABLE Employee != CREATE TABLE EMPLOYEE
creATE TabLE employee = CREATE TABLE employee
```

Dealing with Case Sensitivity

- If you are using a case sensitive Database choose:
  - MACRO_CASE
  - snake_case
- Even if not, try and keep your queries with matching case
  - I got burned on this once.

FROM (The relation(s) to be queried)

SELECT c1
FROM t1

Is the same as:
$\pi_{c1}(t1)$

WHERE (Condition/Criteria)

```
SELECT c1
FROM t1
WHERE c1 >= 10
```

Same as:

$\pi_{c1}(\sigma_{c1>=10}(t1))$

Exercise: Write DDL to create the following Tables

Employee (EmpId, Fname, Lname, Gender, MgrId, LocId, EmailAddress)

Location (LocId, LocName, City, State)

Class (ClassId, Cname, InstructorId, LocId, TotHours)

```
CREATE TABLE <name> (
   <colname> <datatype> <options>
  ,<colname> <datatype> <options>
  ,...
  ,PRIMARY KEY(<colname>,<colname>)
);
```

Exercise: Insert Tuples from HW1

- E.g. Employee
  - 106, Petrina, Tillman, F, NULL, 2, 106@company.com
  - 112, Alec, Wilhoit, M, 106, 1, 112@company.com
- E.g. Location
  - 1, CoffeeTree, Seattle, WA
  - 2, Evergreen, Tacoma, WA
- E.g Class
  - 1003, Conflict Management, 112, 1, 10
  - 1005, Management Essentials, 112, 1, 40

Write simple select statements:

> Get first and last names of Female Employees

> Get location name, city and state for Locations in Washington

> Get class names where total hours are more than 20

WHERE: PATTERN MATCHING WITH ‘LIKE’

```
% = Any number of characters (including zero)
_ = Any Single Character

Ex:  Cname LIKE ‘Management%’
Ex:  Urgency LIKE ‘P_ Resolution’
```

WHERE: PATTERN MATCHING WITH ‘LIKE’

> Find courses that start with the letter “P”

Find courses that are at least 3 words?

WHERE: PATTERN MATCHING WITH ‘LIKE’

> Find courses that start with the letter “P”

```
WHERE Cname LIKE ‘P%’
```

Find courses that are at least 3 words?

```
WHERE Cname LIKE ‘% % %’
```

MARIA DB SUPPORTS REGULAR EXPRESSIONS

Look up all classes that end in 1xx

(could use LIKE '%1__' but that would match 'Some class 1ab'

```
WHERE CName REGEXP '^.*1[0-9]{2}$';
```

(translated Starts with any character, any number of times, followed by a 1, directly followed by exactly 2 numeric characters, followed directly by the end of the string)

UNIVERSITY *of* WASHINGTON

JOINS!

Implicit Joins (eew)

```
SELECT *
FROM Class, Location
```

Returns all columns from both tables, matching on their common attributes (Natural Join)

Implicit Joins (eew)

```
SELECT name

FROM Movies, MovieExec

WHERE title = ‘Star Wars’
  AND ProducerC# = cert#
```

Explicit Joins: KA preferred.

```
SELECT MovieExec.name


FROM Movies
INNER JOIN MovieExec
  ON Movies.ProducerC# = MovieExec.cert#

WHERE title = 'Star Wars'
```

Aliases and Disambiguation

> Columns and Tables can (and should!) be aliased

```
SELECT
  C.Cname    as ClassName
, L.LocName as LocationName
, L.City

FROM Class C
JOIN Location L
  on C.LocId = L.LocId
```

ClassName	LocationName	City
Customer Service 101	Mile High	Denver
Conflict Management	CoffeeTree	Seattle

Aliases and Disambiguation

> Remember our self join in Homework 1?

```
SELECT
  E.Fname    as EmpFirstName
, E.Lname    as EmpLastName
, MGR.Fname  as MgrFirstName
, MGR.Lname  as MgrLastName

FROM Employee E
JOIN Employee MGR
  on E.MgrId = MGR.EmpId
```

UNIVERSITY *of* WASHINGTON

Unions, Intersections, and Difference

Set Operations

- SQL provides corresponding operators that apply to the results of queries, provided those queries produce relations with the same list of attributes and attribute types.
- The keywords used are UNION, INTERSECT, and EXCEPT for U, ∩, and -, respectively.
- There is a bag union called "UNION ALL" that will not remove duplicates but rather stick one relation on top of the other.
- *Not all databases support these keywords.*

Set Operations

```
MovieStar (name, address, gender, birthdate)
MovieExec (name, address, cert#, netWorth)
Movies (title, year, length, genre, studioName, producerC#)
StarsIn (movieTitle, movieYear, starName)
```

```sql
SELECT name, address
  FROM MovieStar
 WHERE gender = 'F'
INTERSECT
SELECT name, address
  FROM MovieExec
 WHERE netWorth > 10000000;
```

```sql
SELECT name, address
  FROM MovieStar
EXCEPT
SELECT name, address
  FROM MovieExec;
```

```sql
SELECT title, year
  FROM Movies
UNION
SELECT movieTitle AS title
     , movieYear  AS year
FROM StarsIn;
```