Managing Data & Databases

Session 4
Saving your lists

Parts of SQL

- DDL
  - Data Definition Language
- DML
  - Data Manipulation Language
- DCL
  - Data Control Language
- TCL
  - Transaction Control Language

- We mainly cover the first two

Some Concepts

- **Field:** Is the unit of data in databases. Ideally each field should contain only one piece of information. The fields can also be completely empty (Null).
- **Null Value:** Nothing. Zilch. This field has been emptied, or nothing has ever been assigned to it.
  - ***Attention***: NULL does not mean zero or empty and is not treated as such. See the examples below.
  - SELECT 2 + 0 => 2
  - SELECT 2 + NULL => NULL
  - SELECT CONCAT("Brel", "") => "Brel"
  - SELECT CONCAT("Brel", NULL) => NULL
- **Table:** Is the unit of data structure in RDBMSs. Tables are composed of rows (records) and columns. The intersection of each column and row is a cell called field.
- **Column:** Each column of a table has a definite data type, and may impose a number of constraints on the data that it captures.
- **Row:** Is a combination of data units called field. Each row should ideally include data about only one entity.

Some Concepts

- **Constraints**
  - **Primary Key:** Is the unique identifier of each row.
  - **Foreign Key:** Tells the database that the value of a field should be looked up in the primary key field of another table. Foreign keys are used to link separate tables in RDBMSs.
  - **Unique:** Ensures no repetitive values are entered in a column (or a combination of columns).
  - **Not Null:** Ensures no row is added with no value attributed to the field affected by "not null" constraint.
  - **Default:** Allows the creator of a column to choose a default value for the fields of the column. If the user fails to enter a value for the column while creating a new row, the default value will take over.
  - **Check:** Ensures that the value entered by the user in a certain field falls within an acceptable range of values. Unfortunately MySQL does not support checks.
- **Auto-increment:** Is a numeric field whose value is automatically generated by the database engine. Each time the user enters a new row of data, the db engine takes care of filling in the auto-increment field with a fresh unique value specific to that row. Auto-increment fields are usually used to create reliable primary keys.
  - **Attention:** Auto-increment guaranties order but does not guaranty continuity. Newer rows always take larger numbers in a single auto-increment field, but they won't necessarily take consecutive numbers. (e.g. 3, 6, 17, 20, …)

Today's Dose of SQL
DDL

- Create
- Alter
- Drop

SQL Command
CREATE

- Use case
  - CREATE is used to "create" different types of objects in a database. The one we learn about today is "table".
- Syntax
  - CREATE TABLE `table_name` (`column 1` data type for column 1 [column 1 constraint(s)], ... , [table constraint(s)]);
  - You can also create a table by right-clicking "Tables" in Workbench's sidebar and then selecting "Create Table".
- Example
  - CREATE TABLE people (`id` int(11) NOT NULL AUTO_INCREMENT, `first_name` varchar(45) NOT NULL, `last_name` varchar(45) NOT NULL, `birth_date` date DEFAULT NULL, PRIMARY KEY (`id`), UNIQUE KEY `id_UNIQUE` (`id`));

Some naming conventions (1)

- .
  - x.y can be read as "y" in "x"
  - Let's say table "people" in database "class": class.people
  - Let's say field "first_name" in table "people": people.first_name
  - Let's combine both: class.people.first_name
  - MySQL's query parser will guess what object you are referring to by looking at the context in which you are making the call. Sometimes it is not necessary to give the full address of an object when calling it, but it is generally recommended to do so in order to avoid confusion.

Some naming conventions (2)

- `
  - Backticks are optional unless you have used some "special" characters when naming your database objects. MySQL Workbench is just being conservative when putting every name between backticks.
  - Don't use space nor any sign (underscore "_" is an exception), or you will have to use backticks when calling your objects all the time.
- ' or "
  - Single or double quotes describe "text". They tell the query parser that what it sees does not need any interpretation, and it is actually simple text. The validity of these two as a text marker varies from installation to installation. Use the one that works on your DB, or have a look here for a deeper explanation.
- Define a naming convention for yourself and abide by it. Mine is: Always small letters; words separated by underscore.

SQL Command
ALTER

- Use case
  - ALTER is used to "alter" different types of objects in a database. We don't cover it in class today, but you may have to learn it in order to add a field to your table, or remove a constrain, etc…
- Syntax
  - ALTER TABLE `table_name` [alter specification];
  - You can also alter a table by right-clicking its name in Workbench's sidebar and selecting "Alter Table".
- Example
  - ALTER TABLE people ADD mcgill_id VARCHAR(9),
    ADD email VARCHAR(45),
    ADD CONSTRAINT UNIQUE(mcgill_id);

SQL Command DROP

- Use case
  - DROP is used to "remove" different types of objects in a database. Here we focus on dropping tables.
- Syntax
  - DROP TABLE `table_name`;
  - You can as well drop a table by right-clicking its name in Workbench's sidebar and selecting "Drop Table".
  - Attention: If you don't know whether a table exists and you want to make sure you have dropped it without receiving a database error, you can use DROP TABLE IF EXISTS
- Example
  - DROP TABLE people;

Today's Dose of SQL DML

- Insert
- Select
- Update
- Delete

SQL Commands
INSERT INTO

- Use case
  - INSERT INTO is used to "insert" data in a table.
- Syntax
  - INSERT INTO "table_name" [("column1", ...)] VALUES ("value1", ...);
  - You have probably noticed that when you view your table contents you can actually edit the fields.
- Example
  - INSERT INTO people VALUES (NULL, 'Mahmood', 'Zargar', '1982-01-15');
  - INSERT INTO people (first_name, last_name, birth_date) VALUES ("Mahmood", "Zargar", "1982-01-15");

SQL Commands
SELECT

- Use case
  - SELECT is used to "fetch" data from tabular data containers (tables, etc…) in a database. Here we focus on selecting from tables.
- Simplified Syntax
  - SELECT `field1_name`, … FROM `table_name`;
  - You can as well select data from a table by right-clicking its name in MySQL's sidebar and selecting "Select Rows".
- Example
  - SELECT * FROM people;
  - SELECT id, last_name FROM people;

SQL Commands
WHERE Clause

- Use case
  - WHERE is used to "limit" the domain of reference of many of SQL commands that deal with fetching or manipulating data.
- Simplified Form
  - WHERE `field_name` condition;
- Example (The list of conditions is not exhaustive)
  - SELECT * FROM people WHERE first_name = "Mahmood";
  - SELECT * FROM people WHERE first_name like "%oo%";
    - Fetch all info on people with a first name containing double O.
  - SELECT * FROM people WHERE birth_date IS NOT NULL;
  - SELECT last_name FROM people WHERE id BETWEEN 1 AND 3;
    - BETWEEN is inclusive

SQL Commands
UPDATE

- Use case
  - UPDATE is used to "update" the data in a range of rows in a table. Works exclusively with a WHERE clause.
- Syntax
  - UPDATE "table_name" SET "column_1" = [some value] WHERE "condition";
- Example
  - UPDATE people SET email = "mahmood.shafeiezargar@mcgill.ca", mcgill_id = "260403725" WHERE first_name = "Mahmood" AND last_name = "Zargar";

SQL Commands
DELETE FROM

- Use case
  - DELETE FROM is used to "delete" a range of rows from a table. In most databases it necessarily requires a WHERE clause to work.
- Syntax
  - DELETE FROM "table_name" WHERE "condition";
- Example
  - DELETE FROM people WHERE birth_date IS NULL;
  - If you encounter an error when trying to delete, see "Why MySQL doesn't let me delete my records?" from our newborn FAQ.
- Bonus
  - If you want to empty a table you don't have to delete every record. All you have to do is to "truncate" the table.