# DDL

Grigorios Loukides
Email: grigorios.loukides@kcl.ac.uk

# Session Objectives





In this session, you will learn:

- How to translate ER models into tables in a database
- Structured Query Language (SQL)
- Data types supported by SQL standard
- How to create, alter and delete tables
- How to create and delete indexes using SQL

# Database Design Problem





Draw an ERD for next week:

*The club has a number of members who can practice different martial arts. For all members you need to register, in addition to the name and the membership number, which martial arts they practice (you can practice several) and which belt (or degree) they hold in the art in question. You also need to register information about their membership fee, namely the amount and the payment date. You must even register whether the member has a valid licence or not (you need a different licence for each art that you practice, so think about how to represent these!)*

# From ERD to Relational Model



| **ERD** | **Logical Model** |
|---|---|
| Entity | Create table |
| Attribute | Create column in table |
| Relationship | |
| –1:N | Primary key of table on "one" side as foreign key of table on "many" side |
| –1:1 | |
| (a) Mandatory on both sides | Combine entities into one table |
| (b) Mandatory on one side | Primary key of table on "optional" side as foreign key of table on "mandatory" side |
| (c) Optional on both sides | Arbitrary |
| –N:M relationship | Create a composite table. The primary keys from entities into the new table to act as primary and and foreign key |

# Introduction





SQL is a database language that:

- Allows you to create database and table structures, to perform data management tasks and to perform complex queries designed to transform the raw data into useful information.
- It is portable, it is a de facto standard SQL

# SQL





SQL functions fit into two broad categories:

- Data Definition Language (DDL): SQL includes commands to create database objects such as tables, indexes, and views, as well as commands to define access rights to those database objects
- It is a data manipulation language (DML): SQL includes commands to insert, update, delete, and retrieve data within the database tables

# Data Definition Language (DDL)





DDL allows to define database objects:

- `SCHEMA` is a collection of tables
- `TABLE` is a set of data elements (values) using a model of vertical columns (identifiable by name) and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows
- `INDEX` is a special lookup table that the DBMS can use to speed up data retrieval
- `VIEW` is a virtual table that does not necessarily exist in the database but can be produced upon request

# Writing SQL





SQL statement consists of *reserved words* and *user-defined* words:

- *Reserved words* are a fixed part of SQL and must be spelt exactly as required
- *User-defined words* are made up by user and represent names of various database objects such as relations, columns, views

# Literals in SQL





- Literals are constants that are used in SQL statements
- There are different forms of literals for every data type supported by SQL

# Writing Literals





A literal is a data value commonly used in variable assignments or comparisons:

- *Numeric Literals*: Integers are represented as a sequence of digits. Floats use . as a decimal separator
- *String Literals*: is a sequence of bytes or characters, enclosed within either single quote (') or double quote (") characters
- *Date and Time Literals*: Date and time values can be represented in several formats, such as quoted strings or as numbers, depending on the exact type of the value and other factors
- *Boolean Literals*: The constants TRUE and FALSE evaluate to 1 and 0, respectively. The constant names can be written in any lettercase.

# Syntax Notation





- Upper-case letters represent reserved words
- | indicates a choice among alternatives.
- Curly braces { indicate a required element.
- Square brackets [ indicate an optional element.
- ... indicates optional repetition (0 or more).

# Overview of some DDL Commands





```
CREATE SCHEMA   DROP SCHEMA
CREATE TABLE    DROP TABLE
       ALTER TABLE
CREATE INDEX    DROP INDEX
CREATE VIEW     DROP VIEW
```

# Creating a Database





`CREATE {DATABASE | SCHEMA} [IF NOT EXISTS]` *dbName*

Creates a database with the given name:

- To use this statement, you need the CREATE privilege for the database.
- `CREATE SCHEMA` is a synonym for `CREATE DATABASE`.
- An error occurs if the database exists and you did not specify `IF NOT EXISTS`.

# Deleting a Database





`DROP {DATABASE | SCHEMA} [IF EXISTS]` *dbName*

Deletes a database with the given name:

- Be very careful with this statement
- To use this statement, you need the DROP privilege for the database.

# Create Table





```
CREATE TABLE tableName
{(colName dataType [NOT NULL][DEFAULT defaultOption]
                   [AUTO_INCREMENT][,...]}
[PRIMARY KEY (listOfColumns),]
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES parentTableName [(listOfCKColumns)] [,]}
)}
```

Creates a table with the given name

# Data Types





MySQL uses many different data types, broken into three categories:

1. Numeric
2. Date and time
3. String types

# Numeric Data Types





- INT - A normal-sized integer that can be signed or unsigned
  - TINYINT, SMALLINT, MEDIUMINT, BIGINT
- FLOAT(M,D) - A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals)
  - DOUBLE (M,D), DECIMAL(M,D)

# Date and Time Types





- DATE - A date in YYYY-MM-DD format
- TIME - Stores the time in HH:MM:SS format
- DATETIME - A date and time combination in YYYY-MM-DD HH:MM:SS format

# String Types





- CHAR(M) - A fixed-length string between 1 and 255 characters in length, right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1
- VARCHAR(M) - A variable-length string between 1 and 255 characters in length. You must define a length when creating a VARCHAR field
- ENUM - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

# Exercise





Given the table below determine the data types of the different columns:

**Table name: EMPLOYEE**

| EMP_NUM | EMP_LNAME | EMP_FNAME | EMP_INITIAL | EMP_HIREDATE | JOB_CODE |
|---|---|---|---|---|---|
| 101 | News | John | G | 08-Nov-00 | 502 |
| 102 | Senior | David | H | 12-Jul-89 | 501 |
| 103 | Arbough | June | E | 01-Dec-96 | 503 |
| 104 | Ramoras | Anne | K | 15-Nov-87 | 501 |
| 105 | Johnson | Alice | K | 01-Feb-93 | 502 |

# Create Table





```
CREATE TABLE tableName
{(colName dataType [NOT NULL][DEFAULT defaultOption]
                   [AUTO_INCREMENT][,...]}
[PRIMARY KEY (listOfColumns),]
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES parentTableName [(listOfCKColumns)] [,]}
)}
```

- The optional `DEFAULT` clause can be specified to provide a default value for a particular column
- The `AUTO_INCREMENT` attribute can be used to generate a unique identity for new rows
  - If no value is specified for an `AUTO_INCREMENT` column, MySQL assigns sequence numbers automatically
  - If applies to integer and floating-point types

# Create Table





```
CREATE TABLE tableName
{(colName dataType [NOT NULL][DEFAULT defaultOption]
                   [AUTO_INCREMENT][,...]}
[PRIMARY KEY (listOfColumns),]
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES parentTableName [(listOfCKColumns)] [,]}
  )}
```

- The `PRIMARY KEY` clause specifies the column or columns that form the primary key
- The `FOREIGN KEY` clause specifies a foreign key

# Create Table





```
CREATE TABLE tableName
{(colName dataType [NOT NULL][DEFAULT defaultOption]
[,...]}
[PRIMARY KEY (listOfColumns),]
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES parentTableName (listOfCKColumns) [,]}
  )}
```

- The `FOREIGN KEY` clause specifies a foreign key
  - A `listOfFKColumns`, the column or columns from the table being created that form the foreign key.
  - A `REFERENCES` subclause, giving the parent table; that is, the table holding the matching candidate key and the list of columns referenced

# Exercise





Given the attributes below (which are a subset of the Employee table) create a SQL statement to create the table of a table named EMP_1

| ATTRIBUTE (FIELD) NAME | DATA DECLARATION |
|---|---|
| EMP_NUM | CHAR(3) |
| EMP_LNAME | VARCHAR(15) |
| EMP_FNAME | VARCHAR(15) |
| EMP_INITIAL | CHAR(1) |
| EMP_HIREDATE | DATE |
| JOB_CODE | CHAR(3) |

# Foreign Key





```
FOREIGN KEY (listOfFKColumns)
REFERENCES parentTable (listOfFKColumns)
      [ON DELETE option]
      [ON UPDATE option]

option: RESTRICT | CASCADE | SET NULL | SET DEFAULT
```

MySQL:

- Rejects any `INSERT` or `UPDATE` on the child table violating referential integrity
- An `UPDATE` or `DELETE` operation in a value in the parent table with matching rows in the child table is determined by the referential action

# Foreign Key





```
FOREIGN KEY (listOfFKColumns)
REFERENCES parentTable (listOfFKColumns)
      [ON DELETE option]
      [ON UPDATE option]
option: RESTRICT | CASCADE | SET NULL | SET DEFAULT
```

Referential Actions:

- `CASCADE`: Delete or update the row from the parent table, and automatically delete or update the matching rows in the child table
- `SET NULL`: Delete or update the row from the parent table, and set the foreign key column or columns in the child table to `NULL`
- `RESTRICT`: Rejects the operation for the parent table
- For an `ON DELETE` or `ON UPDATE` that is not specified, the default action is always `RESTRICT`

# Cloning a Table





`CREATE TABLE` *newTable* `LIKE` *oldTable*

Creates an empty table based on the definition of another table, including any column attributes and indexes defined in the original table

# Altering a Table





`ALTER TABLE` *tblName* **alterSpecification**

Changes the structure of a table

- These changes are indicated in the **alterSpecification**
- You can add or delete columns, keys, indexes, change the type of existing columns, or rename columns or the table itself..
- More information about its syntax:
  `http://dev.mysql.com/doc/refman/5.7/en/alter-table.html`

# Deleting a Table





`DROP TABLE [IF EXISTS]` *tblName*[,...]

DROP TABLE removes one or more tables (whose names are passed as a list):

- All table data and the table definition are removed, so be careful with this statement!
- If any of the tables named in the argument list do not exist, MySQL returns an error indicating by name which nonexisting tables it was unable to drop, but it also drops all of the tables in the list that do exist.

# Exercise





Table name: DIRECTOR — Database name: Ch03_Theater

| DIR_NUM | DIR_LNAME | DIR_DOB |
|---|---|---|
| 100 | Broadway | 12-Jan-65 |
| 101 | Hollywoody | 18-Nov-53 |
| 102 | Goofy | 21-Jun-62 |

Table name: PLAY

| PLAY_CODE | PLAY_NAME | DIR_NUM |
|---|---|---|
| 1001 | Cat On a Cold, Bare Roof | 102 |
| 1002 | Hold the Mayo, Pass the Bread | 101 |
| 1003 | I Never Promised You Coffee | 102 |
| 1004 | Silly Putty Goes To Washington | 100 |
| 1005 | See No Sound, Hear No Sight | 101 |
| 1006 | Starstruck in Biloxi | 102 |
| 1007 | Stranger In Parrot Ice | 101 |

1. What are the data types of the different columns?
2. What are the referential integrity constraints that should hold on the schema?
3. Write appropriate SQL DDL statements to define the database

# Index



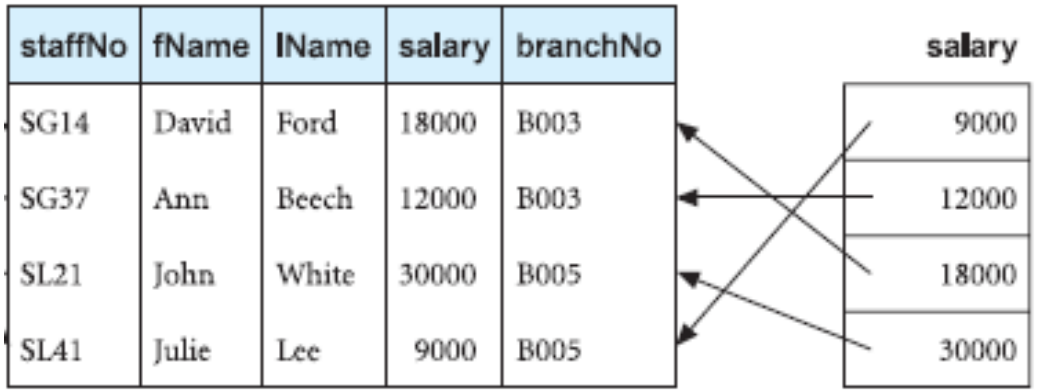


- Indexes are used to find rows with specific column values quickly.
- Without an index, a DBMS must begin with the first row and then read through the entire table to find the relevant rows.
- The larger the table, the more this costs.
- If the table has an index for the columns in question, DBMS can quickly determine the position to seek to in the middle of the data file without having to look at all the data.
- This is much faster than reading every row sequentially.

# Index Example







| staffNo | fName | lName | salary | branchNo |
|---|---|---|---|---|
| SG14 | David | Ford | 18000 | B003 |
| SG37 | Ann | Beech | 12000 | B003 |
| SL21 | John | White | 30000 | B005 |
| SL41 | Julie | Lee | 9000 | B005 |

| salary |
|---|
| 9000 |
| 12000 |
| 18000 |
| 30000 |

# Index





- An index is a structure that provides accelerated access to the rows of a table based on the values of one or more columns
- Indexes can be used to improve the efficiency of data retrievals
- However, since indexes may be updated by the system every time the underlying tables are updated, additional overheads may be incurred
- Indexes are usually created to satisfy particular search criteria after the table has been in use for some time and has grown in size.

# Create Index





`CREATE [UNIQUE] INDEX` *indexName* `ON` *tableName*
(*columnName* `[ASC| DESC][, . . . ]`)

- The specified columns constitute the index key and should be listed in major to minor order
- If the `UNIQUE` clause is used, uniqueness of the indexed column or combination of columns will be enforced by the DBMS
- For each column, we may specify that the order is ascending (`ASC`) or descending (`DESC`)

# Index Design





- It's not possible to make meaningful physical design decisions until you understand in detail the operations that have to be supported by the DBMS
- General Recommendations
  1. Do not index small tables
  2. Index the primary key (MySQL does it automatically)
  3. Add an index to any column that is heavily used for data retrieval
  4. Add a secondary index to a foreign key if there is frequent access based on it

# Drop Index





`DROP INDEX` *indexName* `ON` *tableName*

Drops the index named *indexName* from the table *tableName*

# Exercise 1





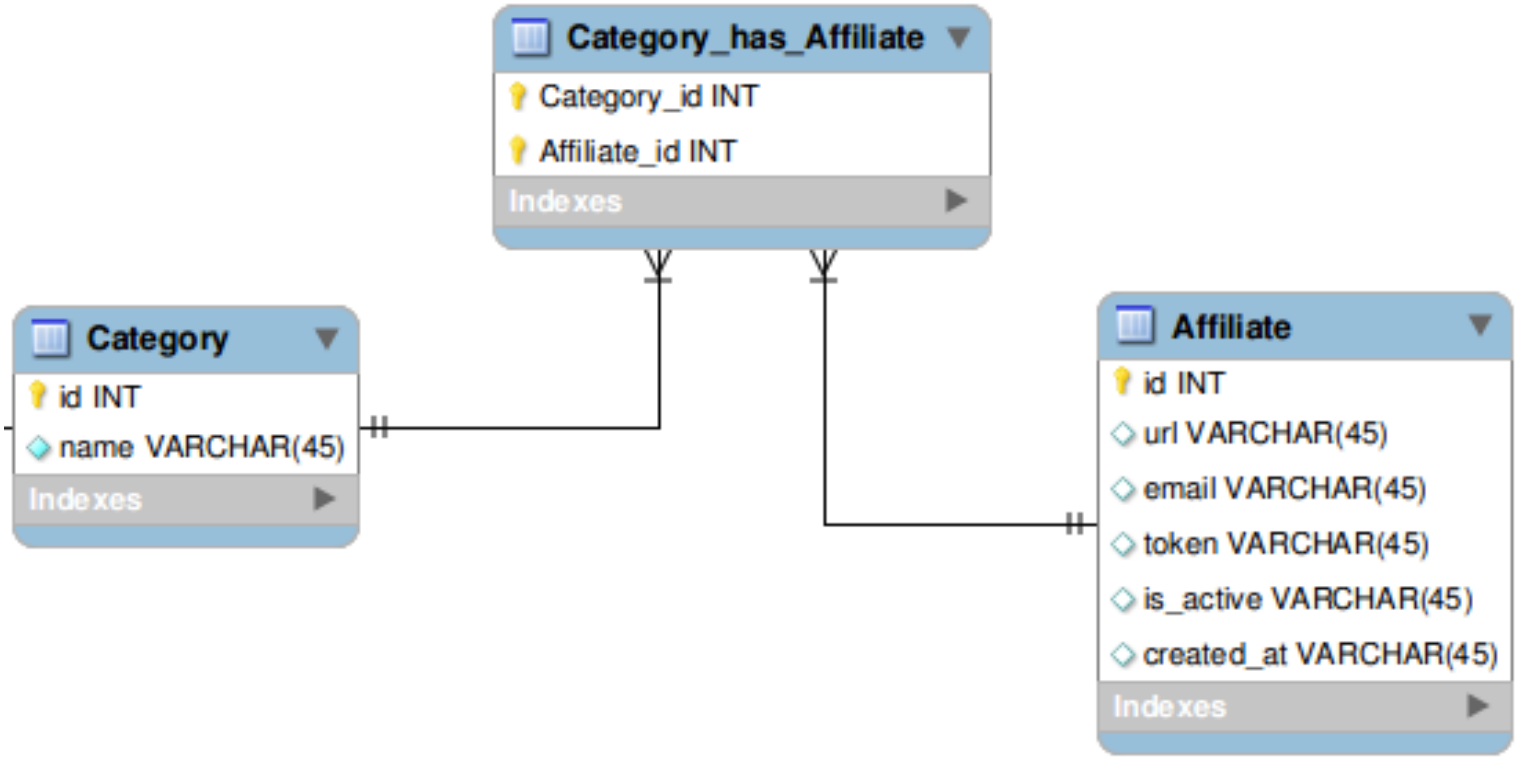
| | |
|---|---|
| Hotel | (hotelNo, hotelName, city) |
| Room | (roomNo, hotelNo, type, price) |
| Booking | (hotelNo, guestNo, dateFrom, dateTo, roomNo) |
| Guest | (guestNo, guestName, guestAddress) |

1. Create the Hotel table
2. Now create the Room, Booking, and Guest tables with the following constraints:
   - type must be one of Single, Double, or Family.

# Exercise 2








1. What are the referential integrity constraints that should hold on the schema?
2. Write appropriate SQL DDL statements to define the database

# Exercise 3





*A company needs to store information about employees (identified by ssn, with salary and phone), departments (identified by dno, with dname and budget), and children of employees (with name and age). Employees work in departments; each department is managed by an employee; a child must be identified uniquely by name when the parent (who is an employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company.*

1. Drawn an ERD for these requirements
2. Write appropriate SQL DDL statements to define the database

# Conclusion





In this session we have covered:

- Table creation, alteration and deletion
- Index creation and deletion

- **Feedback on KEATS** - this week for lecturers, next week for labs/tutorials.

# Lab Session





Next week's lab session more about DDL in MySQL

# Suggested Readings





- Chapters 5 of Fundamentals of Database Systems. Elmasri & Navathe.
- Chapter 6 of Database systems: a practical approach to design, implementation, and management. Connolly, Thomas M; Begg, Carolyn