Python and SQL: intro / SQL platforms

Instructor: Maciej Wilamowski, Ph.D. , mwilamowski@wne.uw.edu.pl

University of Warsaw, Faculty of Economic Sciences

Chair of Microeconomics

Instructor: Robert Wojciechowski, rwojciechowski@wne.uw.edu.pl

University of Warsaw, Faculty of Economic Sciences

Department of Quantitative Finance

Course content (part 1):Intro to SQL

1. **Relational model for database management.**
2. **SQL: Table manipulation and basic queries: create/drop table, select, where, insert, update**
3. **SQL: complex queries, joins, precuders**
4. **SQL: indexing, triggers**

Course content (part 2):Intro to Python

1. **Preparation of the envrironment (Ipython Notebook/PyCharm, data structures, debugging.**
2. **Flow control: if, for, while, iterators, error handling. Working with text files.**
3. **Functions and classes.**
4. **Linear algebra with NumPy**
5. **Data handling and wrangling with Pandas.**
6. **Visualisation with Seaborn and matplotlib.**
7. **Python in the web: using APIs, JSON, XML, we requests, very basic web applications.**
8. **Database manipulation with Python.**
9. **Presentations of projects.**

Grading:

- **Final project (60%) : written report + presentation (15 min)**
- **Written test (40%) [practical part +theoretical part]**
- **Activity (up to 10% extra)**

Grade	Total Score %	Description
5	+90%	very good
4+	+80%	better than good
4	+70%	good
3+	+60%	satisfactory
3	+50%	sufficient
2	Less than 51%	fail

Final project requirements:

Goal of the project:

- prepare project, which presents the use of Python Database programming

Possible project topics:

- Python and DBMS in business solutions (Warehouse, Airline reservations system,..)
- custom Python library (ex .text analyser) + simple registration website for clients who want to regsiter and download new library

Each project should :

- be prepared by 1-2 students
- contain codes written in Python + description and instruction (5 -6 pages)

Deadlines:

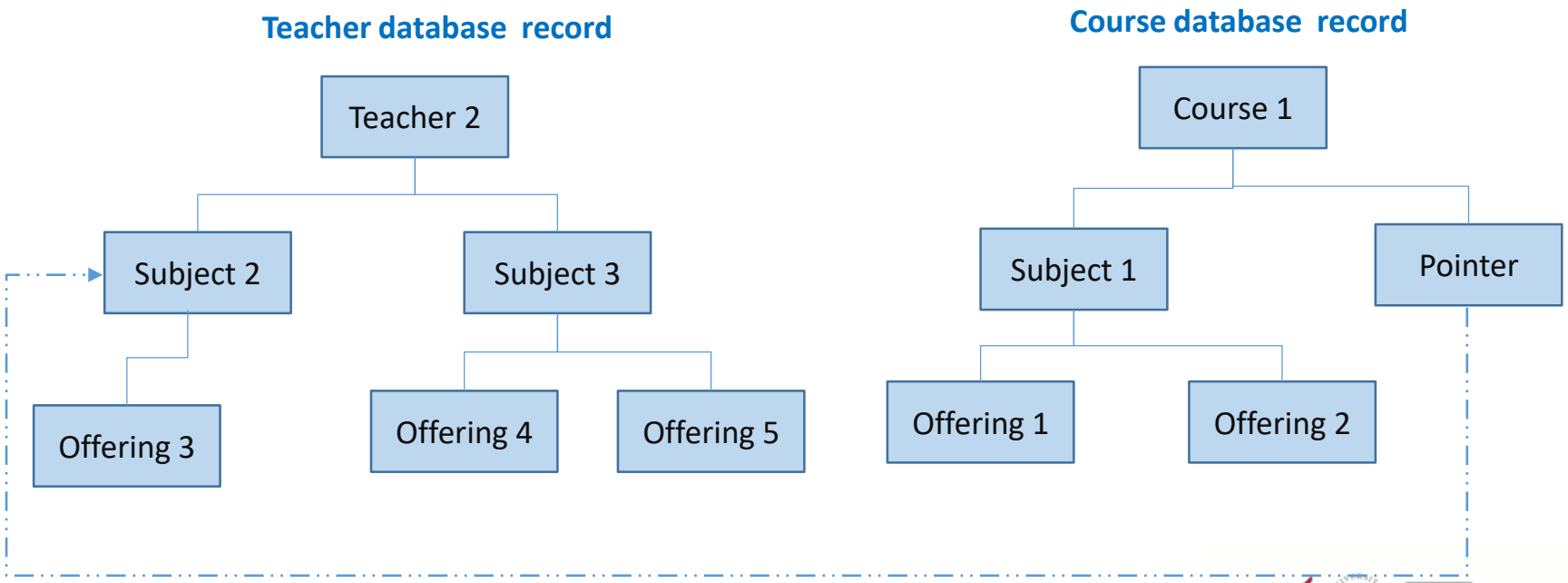
- Proposal of the project should be send instructor till the end of November 2017
- Individual defence of the project will be conducted during last labolatories (5-10 min per project)

Database Introduction

- **database model** - defines the logical design of data
- **database model** - describes the relationships between different part of data
- **most common database models**:
  - ❑Hierarchical Model
  - ❑Network Model
  - ❑Relational Model

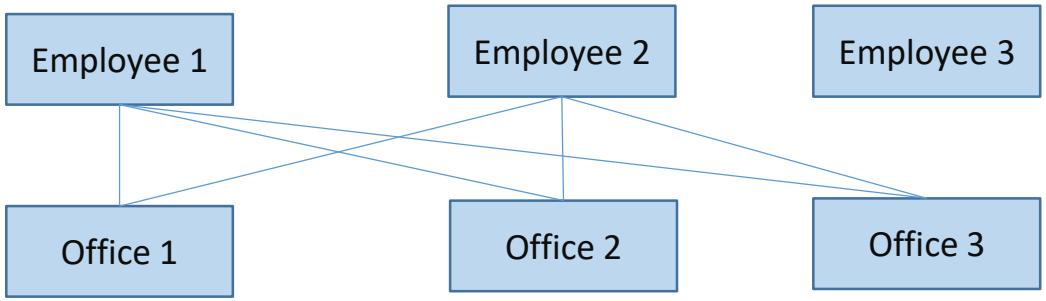
Hierarchical Model

- **oldest database model**
- **used mostly in banks**
- **hierarchical structure is implemented on tree**
- **each child node has only one parent node**
- **parent node can have many child nodes**
- **model efficient but difficult to implement because of its complex structure**



Network Model

- **more flexible than Hierarchical Model**
- **entities organized in graph**
- **no levels, record can have any number of owners**
- **number of links between records is high**
- **databases based on this model are slow**



Relational Model

- first introduced by E.F Codd (1970) in **"A Relational Model of data represented as set of tables"**
- **table** = collection of data elements organised in terms of rows and columns. Tables allow to represent relations
- **record (single row)** = represents set of related data in a table
- **attribute (column) =** set of value of a particular type in table
- example: table with 3 records (rows) and 4 attributes (columns)

Id	name	surname	salary
1	Tom	Smith	5000
2	Alex	Crank	6000
3	David	Gold	3000

Database Keys

- **Primary Key –** it is a key that uniquely identify each record in the table. Here {Id}

Id	Name	Surname	Age
0123	Tom	Gold	45
0124	Mark	Brown	21

- **Composite Key -** consist of two or more attributes that uniquely identify an entity occurance. Here {Name,Surname,DateOfBirth}

Name	Surname	DateOfBirth	NrOfCars
Tom	Gold	6.16.1965	2
Mark	Brown	6.16.1965	1
Adam	Gold	23.11.1975	2

Normalization of Database

- **Normalization –** process of organizing the columns and tables of relational database to reduce data redundancy and data integrity
- concept of normalization and all its four forms were introduced by Edgar F. Codd
- **forms of normalization**
  - ❑1NF (1970)
  - ❑2NF (1971)
  - ❑3NF (1971)
  - ❑BCNF (1974)

Data redundancy:
appears, when the same piece of data is held in two separate places

First Normal Form (1NF)

- each row should have a Primary Key that distinguishes it as unique
- row must not have a column in which more than one value is stored (ex. values:Toyota, Fiat in column {Car})

before

Surname	Age	Car
Gold	45	Toyota, Fiat
Brown	31	Honda, Renault

after

Surname	Age	Car
Gold	45	Toyota
Gold	45	Fiat
Brown	31	Honda
Brown	31	Renault

Second Normal Form (2NF)

- must be in 1NF
- 2NF requires that any non-key field be dependent on the entire **Primary Key**
- in the example below, the candidate compund Primary Key is {Surname,Car}, but {Age} attribute depends only upon {Surname} attribute
- solution: make single Primary Key by dividing input table into two tables, introduce Foreign Key {Surname} in one of the new tables

before (1NF)

Surname	Age	Car
Gold	45	Toyota
Gold	45	Fiat
Brown	31	Honda
Brown	31	Renault

after (2NF)

Surname	Age
Gold	45
Brown	31

ID_Surname_Car	Surname	Car
1	Gold	Toyota
2	Gold	Fiat
3	Brown	Honda
4	Brown	Renault

Third Normal Form (3NF)

Transitive Dependency :
exists, when any attribute in a table is dependent upon any other non-key attribute in that table

- must be in the 2NF (all attributes depend upon Primary Key)
- transitive dependency must be removed
- in given table, {Student_id} is Primary Key, but {Street}, {City} and {State} depend also upon {ZipCode}. The dependency between {Student_Id} and each of these fileds is called transitive dependency.

before (2NF)

Student_Id	Name	Surname	Street	City	State	ZipCode
23	Tom	Gold	Silver Street	Big City	Alabama	23-3472

after (3NF)

Student_Id	Name	Surname	ZipCode
23	Tom	Gold	23-3472

ZipCode	City	State	Street
23-3472	Big City	Alabama	Silver Street

Boyce and Codd Normal Form (BCNF)

- higher version of the Third Normal form
- table must be in 3NF form and does not have multiple overlapping candidate keys

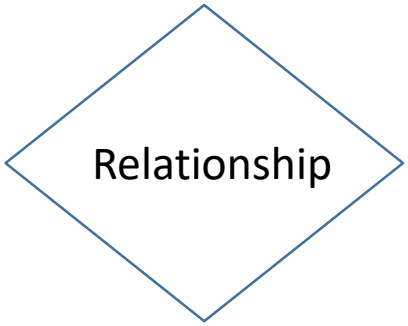
Summary:

[1NF] - data depends on the key
[2NF]- data depends the whole key
[3NF]- data depends on nothing but the key

Entity Relationship Diagram (ERD)

- visual representation of data that describes how data is related to each other
- typically used in computing in regard to the organization of data within databases or information systems
- **Entity** - object or concept about which you want to store information

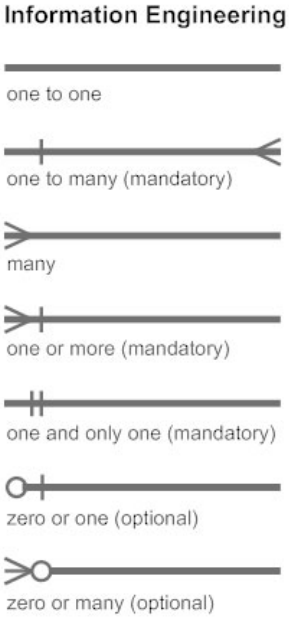
- **Action -** represented by diamond shape, show how two entities share information in the database

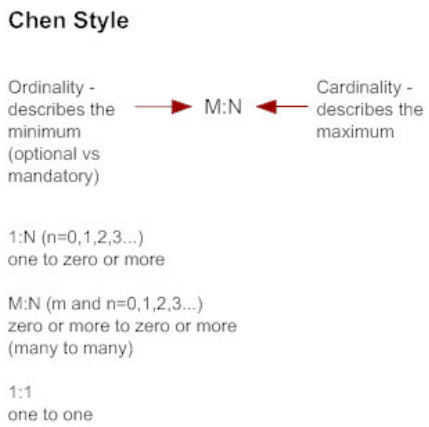


Entity Relationship Diagram

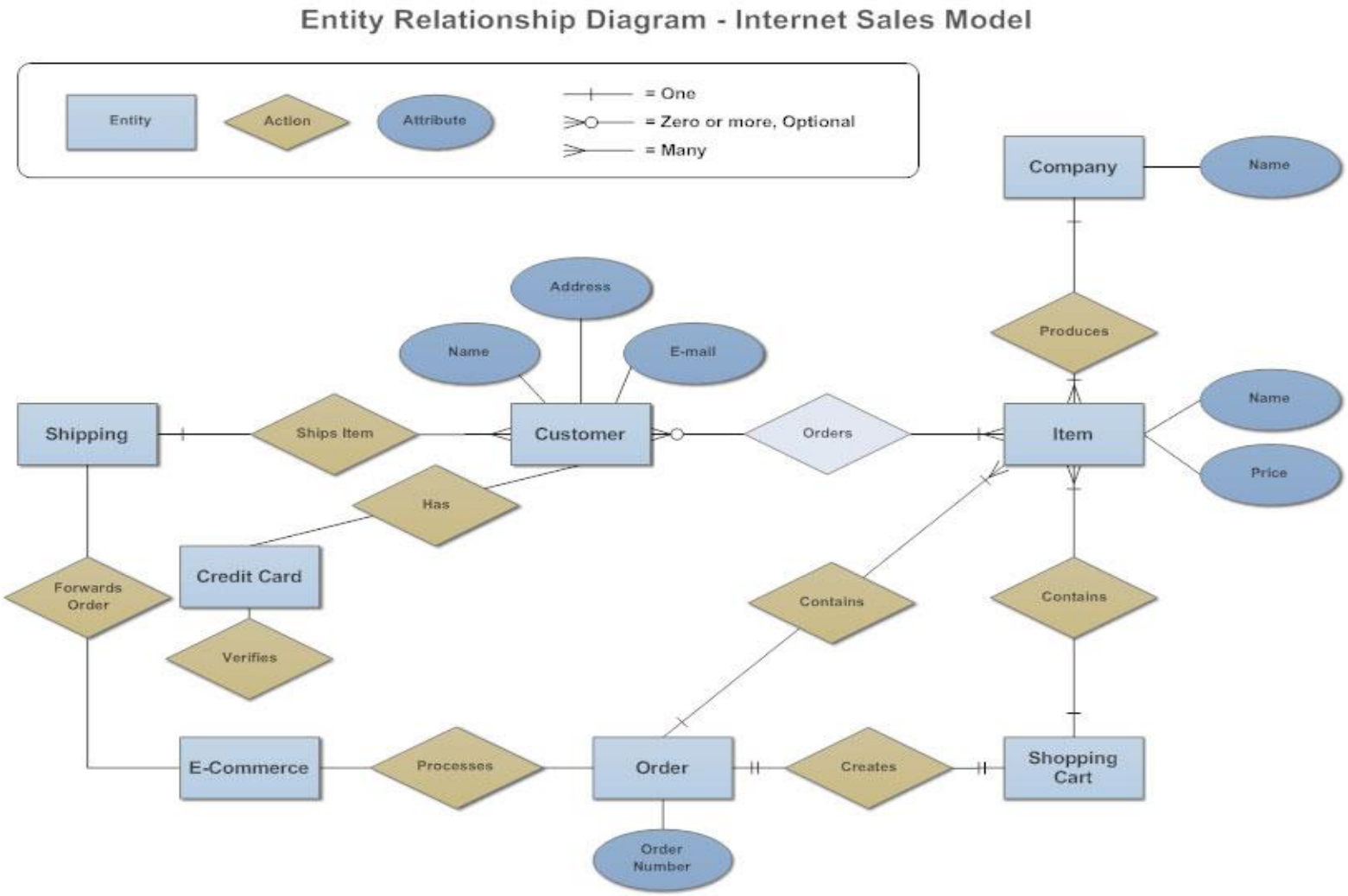
- **Attribute -** represented by oval. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee has attribute surname.

- **Relations –** in the context of cardinality, we should specifyy how many instances of an entity relate to one instance of another entity. There are many notation styles that express cardinality.





ERD Example


Entity Relationship Diagram - Internet Sales Model
Entity
Action
Attribute
= One
= Zero or more, Optional
= Many
Company
Name
Produces
Address
Name
E-mail
Shipping
Ships Item
Customer
Orders
Item
Name
Price
Has
Credit Card
Forwards Order
Verifies
Contains
Contains
E-Commerce
Processes
Order
Creates
Shopping Cart
Order Number

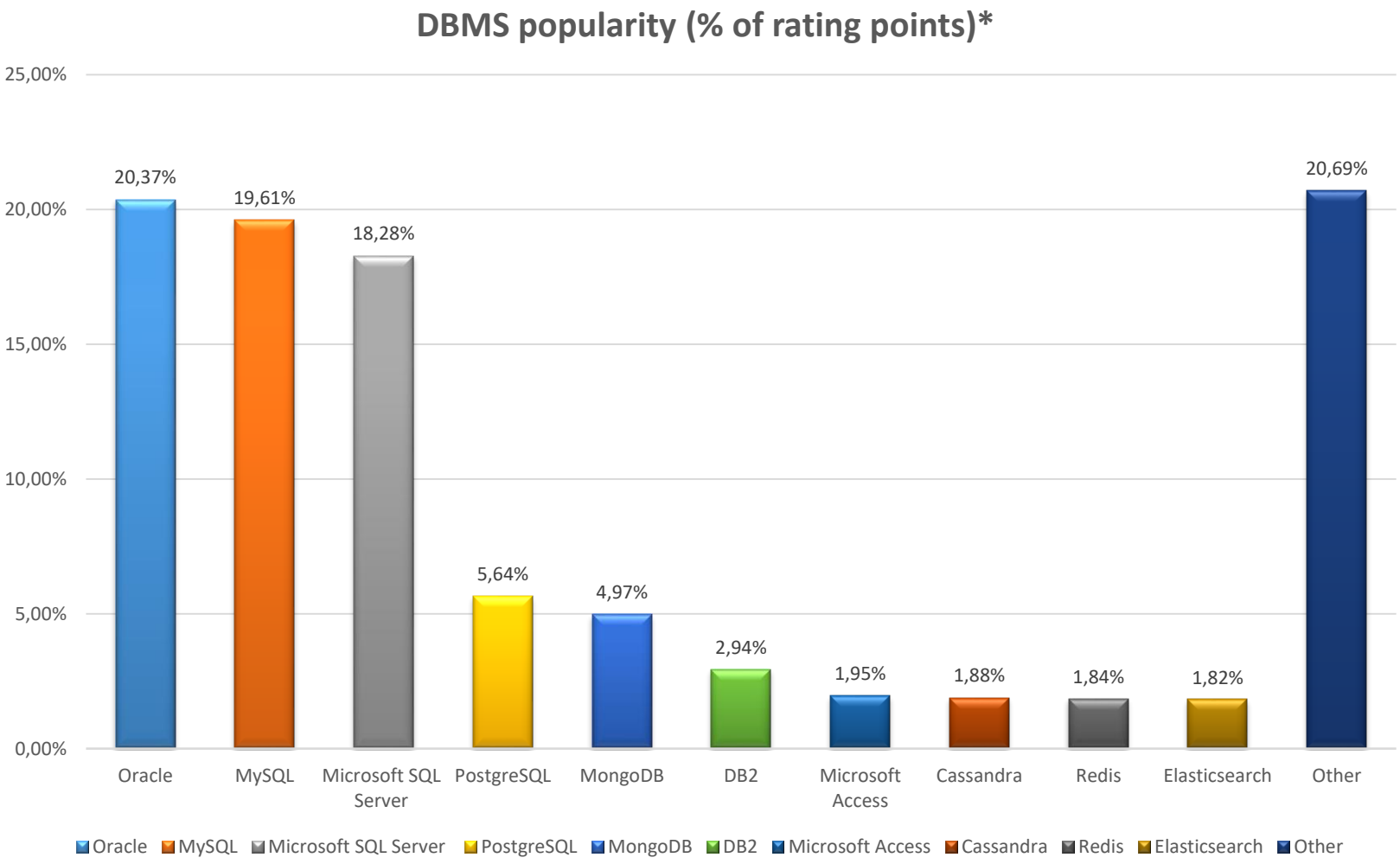
DBMS (Database Management Systems)

- **Functions of DBMS:**
  - Provides data Independence
  - Concurrency Control
  - Provides Recovery services
  - Provides Utility services
  - Provides a clear and logical view of the process that manipulates data

- **Advantages of DBMS:**
  - Segregation of applicaion program.
  - Minimal data duplicacy.
  - Easy retrieval of data.
  - Reduced development time and maintainance need.

- **Examples:** MySql, Oracle,Microsoft SQL Server, Microsoft Access.

DBMS overview



*based on data from https://db-engines.com

Oracle

- Developer: Oracle
- initial release 1980, current version 12
- commercial license
- Implementation language C,C++
- Server operating system : Linux,OS X,Solaris,Windows
- SQL support
- Partitioning: tables can be distributed across several files
- Other features : Concurrency, Transaction concepts, Triggers
- version: 11*g* Express Edition (max db size=11GB,max RAM=1GB,single CPU)

MySql

- relational DBMS
- Developer: Oracle
- initial release 1995, current version 5.7.19, July 2017
- Open Source
- implementation language C,C++
- Server operating system : Linux, Solaris, Windows, FreeBSD
- SQL support
- Partitioning: horizontal partitioning
- Other features : Concurrency, Transaction concepts, Triggers

SQL Server

- relational DBMS
- Developer: Microsoft
- initial release 1989, current version 2016
- commercial license
- Implementation language C++
- Server operating system : Windows
- SQL support
- Partitioning: tables can be distributed across several files
- Other features : Concurrency, Transaction concepts, Triggers
- Freeware version: Express Edition (max db size=10GB,max RAM=1GB,single CPU)

PostgreSQL

- relational DBMS
- Developer: PostgreSQL Global Development Group
- initial release 1989, current version 9.6.5, August 2017
- Open Source
- implementation language C
- Server operating system : Linux, Solaris, Windows
- SQL support
- Partitioning: no
- Other features : Concurrency, Transaction concepts, Triggers

MongoDB

- relational DBMS
- Developer: MongoDB, Inc.
- initial release 2009, current version 3.4.9, September 2017
- Open Source
- implementation language C++
- Server operating system : Linux, Solaris, Windows
- SQL support
- Partitioning: Sharding
- Other features : Concurrency

SQL Introduction

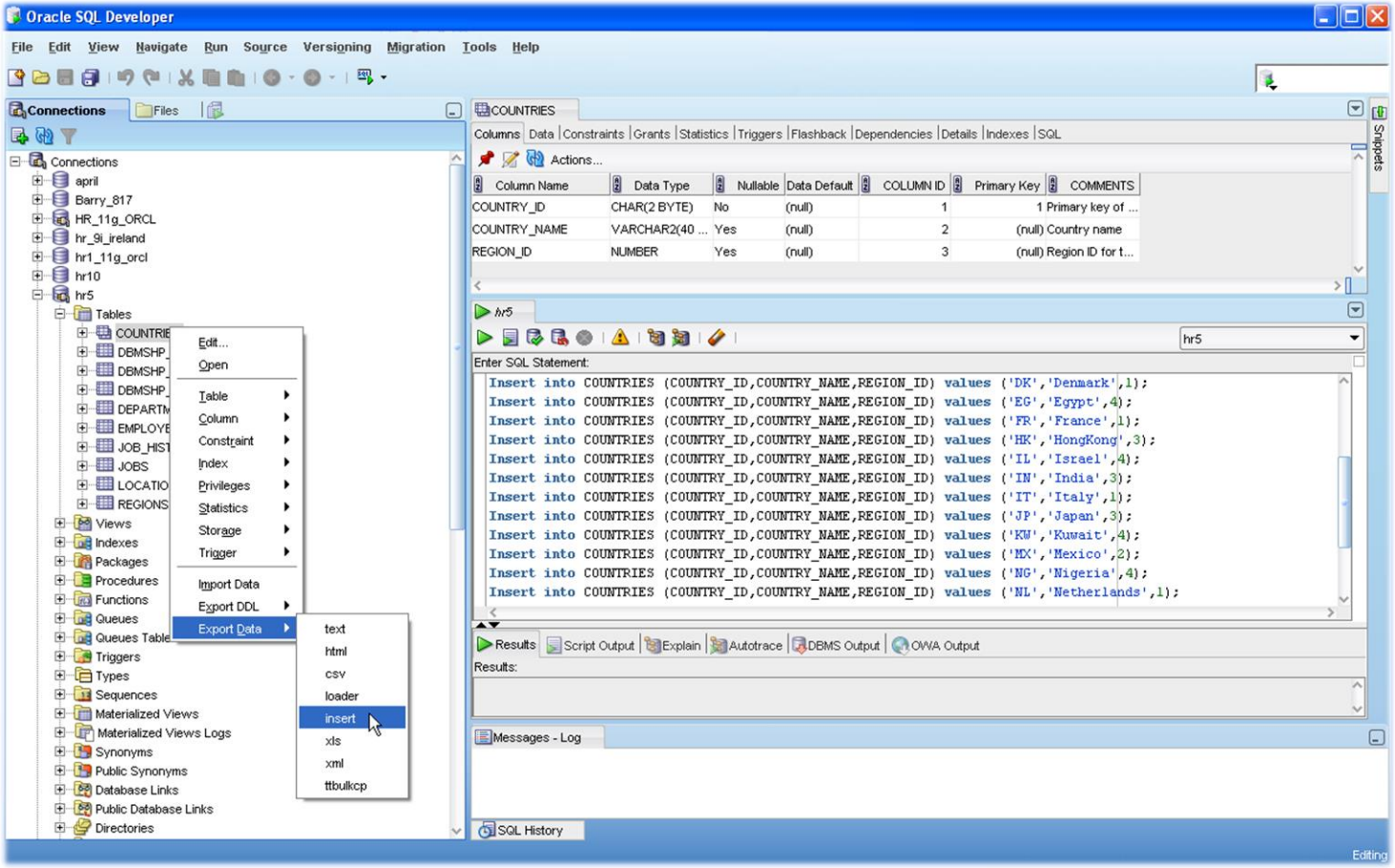
- **SQL -** Structure Query Language
- language for generating, manipulating, and retrieving data from a relational Database
- developed at IBM by Donald D. Chamberlin and Raymond F. Boyce (early 70's)
- initially known as Structured English Query Language (SEQUEL)
- in 1979 Oracle introduced the first commercially available implementation of SQL
- main features:
  - cross-platform (MySql,Oracle,MSQL,…)
  - easy to learn
  - allows the user to create, update, delete, and retrieve data from a database

Column Types

- When we design our database we need to decide what type of data will be stored in each column.
- Choosing appropriate data type is really important for optimized storage and speed of our database.
- Most data types are fixed and strict. Their narrow definition is what allows us to properly optimize storage and data manipulation.
- Additionally our columns can have following attributes:
  - Default value
  - Primary/Unique/Index/Fulltext
  - Auto Increment
  - And several other of lower importance.

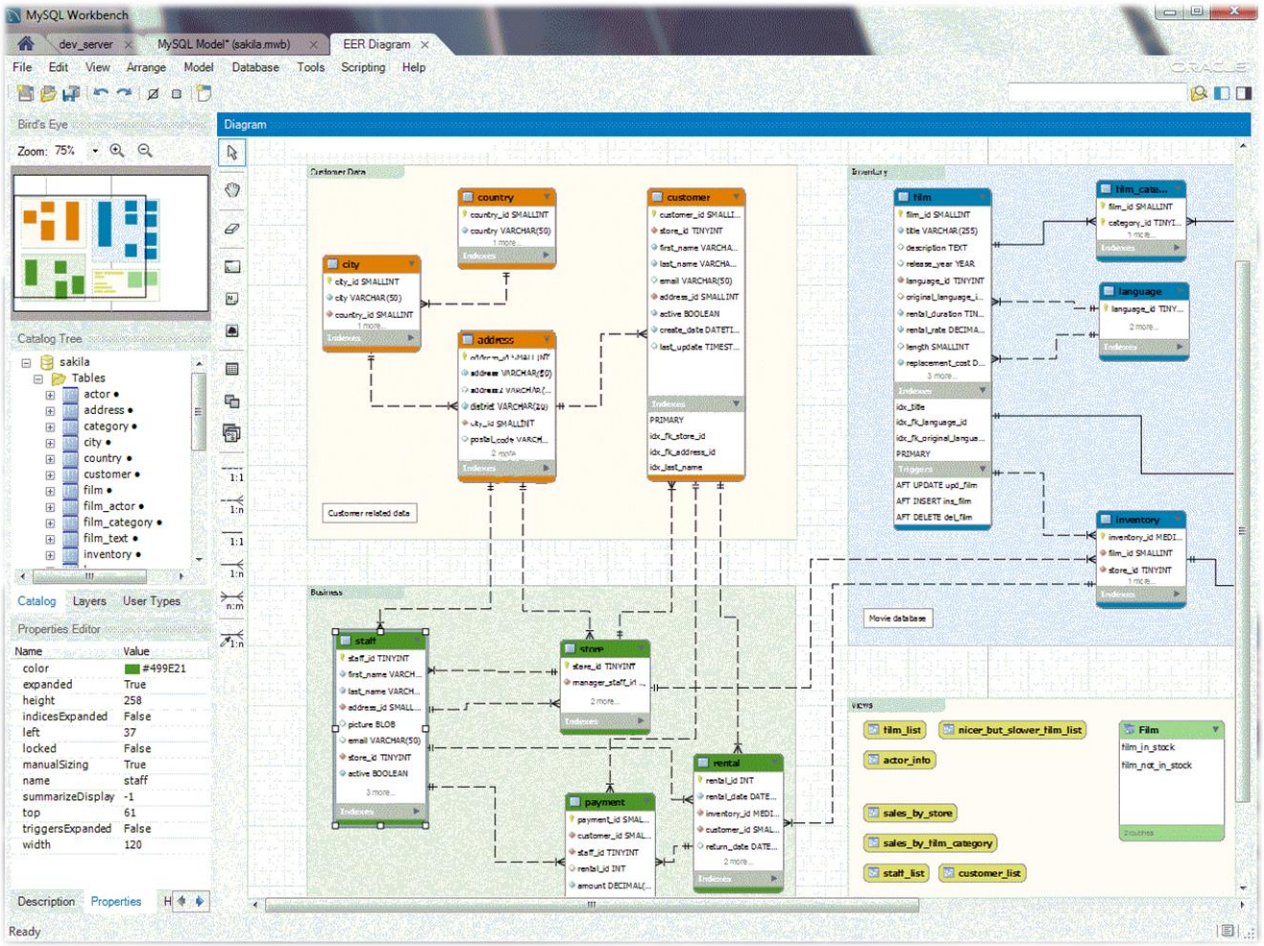
DB AdminTool: Oracle SQL Developer (for Oracle)

- http://www.oracle.com/
- freeware



DB AdminTool: MySQL Workbench (for MySQL)

- https://www.mysql.com/
- freeware

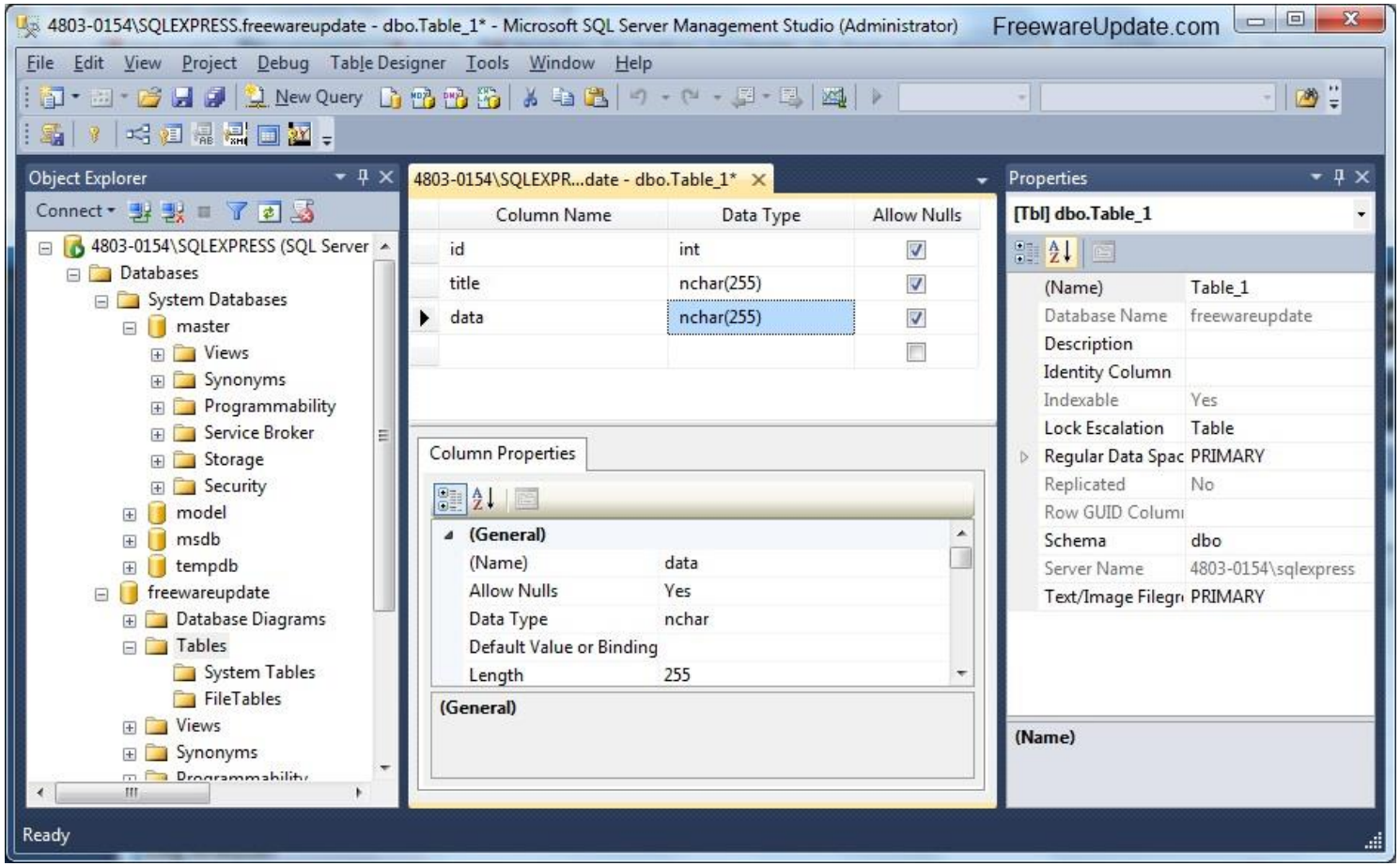


Extras:

-free MySQL host
https://www.db4free.net/signup.php
-free Workbench client
https://www.rollapp.com/app/mysqlworkbench

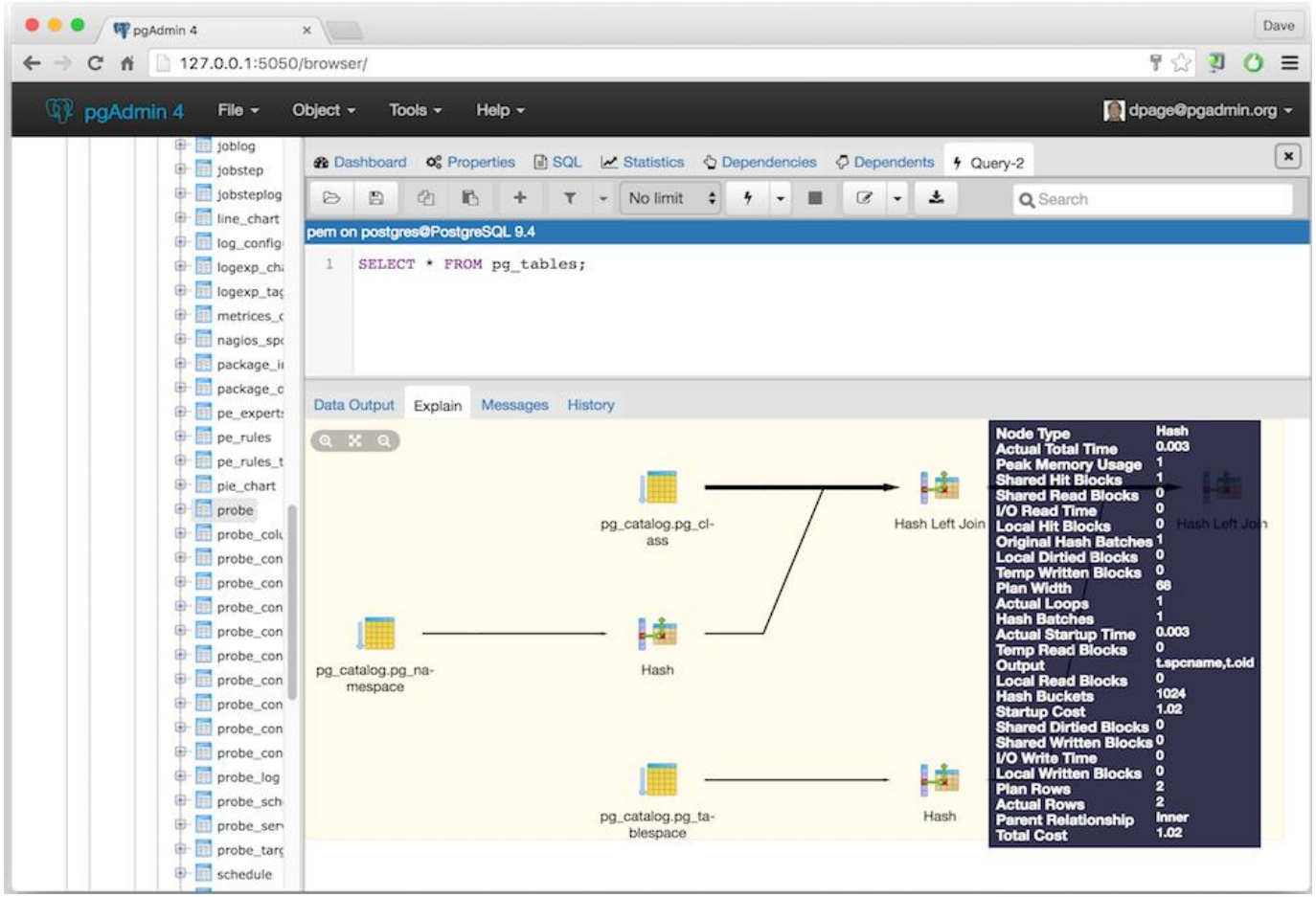
DB AdminTool: SQL Server Management Studio Express (for mSQL)

- https://www.microsoft.com
- freeware



DB AdminTool: pgAdmin (for PostgreSQL)

- https://www.postgresql.org
- freeware



Column Types - part1 (MySql)

ColumnType	Description	Example of use
int	Stores integer (whole number) values in the range –2,147,483,648 to 2,147,483,647.	ex.: id int
decimal	Stores a fixed-point number.	ex.: price decimal(4,2) values from -99,99 to 99,99 4- total number of digits 2- number of digits after comma
double	Stores floating-point numbers.	
float	Stores floating-point numbers.	
date	Stores and displays a date in the format YYYY-MM-DD for the range 1000-01-01 to 9999-12-31. Alternative formats: YYYY/MM/DD, YYYYMMDD	ex. : birth date
time	Stores a time in the format HHH:MM:SS for the range -838:59:59 to 838:59:59 Alternative formats: DD HH:MM:SS, HH:MM:SS, DD HH:MM, HH:MM,	

Column Types-part 2 (MySql)

ColumnType	Description	Example
Char	commonly used string type. Uses always as much space as declered. Char(10) means to allocate always 10 characteres.	ex.: name char(10)
vachar	commonly used string type. Useses only as much space as you need, plus one byte to store the length of string.	ex.: name varchar(10)
text	used for storing large string data objects. Stores a variable amount of data (such as a document or other text file) up to 65,535 bytes in length.	
blob	used for storing large data (binary format). Stores a variable amount of data (such as an image, video, or other nontext file) up to 65,535 bytes in length.	
enum	a list, or enumeration of string values.	ex.: fruit_name ENUM('Apple', 'Orange', 'Pear')

Column Types

- There are more data types. We encourage you to browse the full list at least once:
  - https://dev.mysql.com/doc/refman/5.7/en/data-types.html
  - https://www.w3schools.com/sql/sql_datatypes.asp
- Once again. Chosing appropriate data type is really important for optimized storage and speed of our database.

SQL Naming conventions

- Most important think with naming conventions is to be consistent across all tables as much as possible.
- There is no such think as „the best” naming convention
- Remember you database structure can outlast any application that uses it and many developers and end users.
- Names of tables should be short but meaningful and precise.
- In naming convention most important attributes are:
  - Plural vs singluar
  - Capitalization
  - Spaces
  - Prefixes/suffixes
  - Primary key naming convention

SQL Naming conventions

- Plural vs singular:
  - Some guidelines advocate plural names for tables and singular names for columns. Table Customers can have a column customerId
  - On the other hand using plurals for table names can be sometimes confusing (plural forms are not always straightforward)
  - Plural forms are more natural when read as natural language. We are taking a customer from table full of customers. However sometimes when we address a database table as an object a notation customer.customerId is also natural.

SQL Naming conventions

- There are many options in terms of capitalizations:
  - alllowrcase
  - ALLUPPERCASE
  - camelCase
  - PascalCase
- Additionally we need to decide if we want to use underscore for separation:
  - All_lower_case
  - ALL_UPPER_CALE
  - damel_Case
  - Pascel_Case

SQL Naming conventions

- Prefixes/suffixes and primary key naming conventions.
  - We can meet many accepted prefixes:
    - IX – index
    - PK – Primary Key
    - FK – Foreign Key
    - CK – Check Constraint
    - DF – Default
    - UQ – Unique
- Primary Keys are for us of special importance. Few of popular conventions are listed below:
  - id
  - <table_name>Id
  - Id<table_name>

SQL Naming conventions

- Remember there is no one best answer. You can read more here:
  - https://stackoverflow.com/questions/522356/what-sql-coding-standard-do-you-follow
  - http://www.vertabelo.com/blog/technical-articles/naming-conventions-in-database-modeling
  - http://www.vertabelo.com/blog/technical-articles/an-unemotional-logical-look-at-sql-server-naming-conventions
  - https://www.codeproject.com/Articles/1065295/SQL-Server-Table-and-Column-Naming-Conventions
  - http://leshazlewood.com/software-engineering/sql-style-guide/

Student's Task (Database design)

- **basing on data below and using normalization rules, please design database in https://api.genmymodel.com/**

Name	Surname	DateOfBirth	HomeAddress	Employer	WorkedYears
Tom	Gold	1980-08-01	Silver Street 13, flat 12,Small City, 02-344	MircoTech	5
Tom	Gold	1980-08-01	Silver Street 13, flat 12,Small City, 02-344	BigTech	10
Ann	Smart	1960-06-07	Cambridge Street 11,Green City,04-233	Zara	3
Ann	Smart	1960-06-07	Cambridge Street 11,Green City,04-233	Nike	5
Alex	Brown	1964-06-07	Oxford Street 11,London City,00-231	EY	10

SQL Basic Syntax - CREATE/DROP database

- **Creating an empty database dbTest:**

```
create database dbTest;
```

- **Drop(remove) a database dbTest:**

```
drop database dbTest;
```

- **Show all tables:**

```
show tables; /*in sqlite use .tables*/
```

- **Show all columns in database dbTest:**

```
show columns from dbTest;  /* .schema dbTest*/
```

SQL Basic Syntax – CREATE TABLE

Create table table_name:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Create table customers with single Primary Key:

```
CREATE TABLE customers (
    idCustomer int NOT NULL AUTO_INCREMENT,
    lastName varchar(255) NOT NULL ,
    firstName varchar(255) NOT NULL ,
    address varchar(255) DEFAULT NULL,
    timeStamp time NULL,
    PRIMARY KEY (idCustomer)
);
```

AUTO_INCREMENT :
- column it is used on must be indexed
-column that is it used on cannot have a DEFAULT value
-can be only one AUTO_INCREMENT column per table

SQL Basic Syntax – CREATE TABLE (examples)

Create table, which contains one enumarated filed:

```
CREATE TABLE new_fruits_enum (
      fruit_name ENUM('Apple', 'Orange', 'Pear') DEFAULT 'Pear,
      );
```

Create table, which contains compound Primary Key {artist_id, album_id}:

```
CREATE TABLE album (artist_id INT(5) NOT NULL,
            album_id INT(4) NOT NULL AUTO_INCREMENT,
            album_name CHAR(128) DEFAULT NULL,
            PRIMARY KEY (artist_id, album_id)
      );
```

SQL Basic Syntax – ALTER TABLE

- **Change the name of column {played} to {last_played}**

```
ALTER TABLE Songs CHANGE played last_played TIMESTAMP;
```

- **Modify the data type and clauses of a column {name} in table artist :**

```
ALTER TABLE artist MODIFY name CHAR(64) DEFAULT "Unknown";
```

- **Add an extra column {bandSize} to an existing table artist:**

```
ALTER TABLE artist ADD bandSize int;
```

- **Add column {bandSize} to table artist in a specific position:**

```
ALTER TABLE artist ADD bandSize int AFTER artist_id;
```

- **Remove column {bandSize}**

```
ALTER TABLE artist DROP bandSize
```

- **Rename table artist to playlist**

```
ALTER TABLE artist RENAME TO playlist;
```

ALTER TABLE
- add new columns to a table
- remove existing columns
- change columns names, types, lengths

SQL Basic Syntax – INSERT data

We insert into table artist values of artist_id and surname with below query:

```
INSERT INTO artist VALUES (7, "Adamson");
```

Analogically, we can insert many records in single SQL query:

```
INSERT INTO artist VALUES (7, "Adamson"),
                          (8, ”Brick"),
                          (9,”Samuelson”);
```

Alternatively,we can use syntex as below:

```
INSERT INTO played (artist_id, album_id, track_id)
      VALUES (7,1,2),(7,1,3),(7,1,4);
```

Or

```
INSERT INTO played SET artist_id = 7, album_id = 1, track_id= 1;
```

SQL Basic Syntax – DELETE data

We delete all rows in table songs:

```
DELETE FROM songs;
```

We delete row with satisfies given condition artist_id=3, using following query :

```
DELETE FROM artist WHERE artist_id = 3;
```

SQL Basic Syntax – SELECT data

- **We can select all rows and all columns from a table:**

```
SELECT * FROM table_name;
```

- **Or restrict the list of columns that we are interested in:**

```
SELECT column1, column2, ...
FROM table_name;
```

- **Sometimes we are not interested in all the rows but only in unique values in column1 but also containing information from column2.**

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

Distinct :
- allows to selet unique values from given table

SQL Basic Syntax – SELECT data

- **Usually we want to retrieve just some rows (for example most recent or fulfilling certain criteria)**

```
SELECT * FROM customers WHERE lastName='Wilamowski';
SELECT * FROM orders WHERE orderValue >=10000;
```

- **We may not be sure what is the exact phrase we are looking for:**

```
SELECT * FROM customers WHERE lastName LIKE ,Wilamow%';
SELECT * FROM customers WHERE lastName=,_i%';
```

- **Or we know that there are many rows but we want only 5 most recent ones**

```
SELECT * FROM Orders ORDER BY orderDate DESC LIMIT 5;
```

- **Of course we can order records basing on many columns:**

```
SELECT time, track_name FROM track ORDER BY time, track_name;
```

- **We can also select the maximum/minimum value from column artist_id**

```
SELECT MAX(artist_id) FROM artist;
SELECT MIN(artist_id) FROM artist;
```

SQL Basic Syntax – UPDATE data

Change the values in {artist_name} column to uppercase:

```
UPDATE artist SET artist_name = UPPER(artist_name);
```

Change the values in {artist_name} column to NULL in table songs:

```
UPDATE songs SET artist_name = NULL;
```

We update values in rows,which satisfy given condition using WHERE clause:

```
UPDATE album SET album_name =”Dragon” WHERE artist_id = 1 AND
album_id = 2;
```

We can also update 10 lowest scores to 0 in table songs:

```
UPDATE songs SET score = 0 ORDER BY score ASC LIMIT 10;
```

OREDR BY :

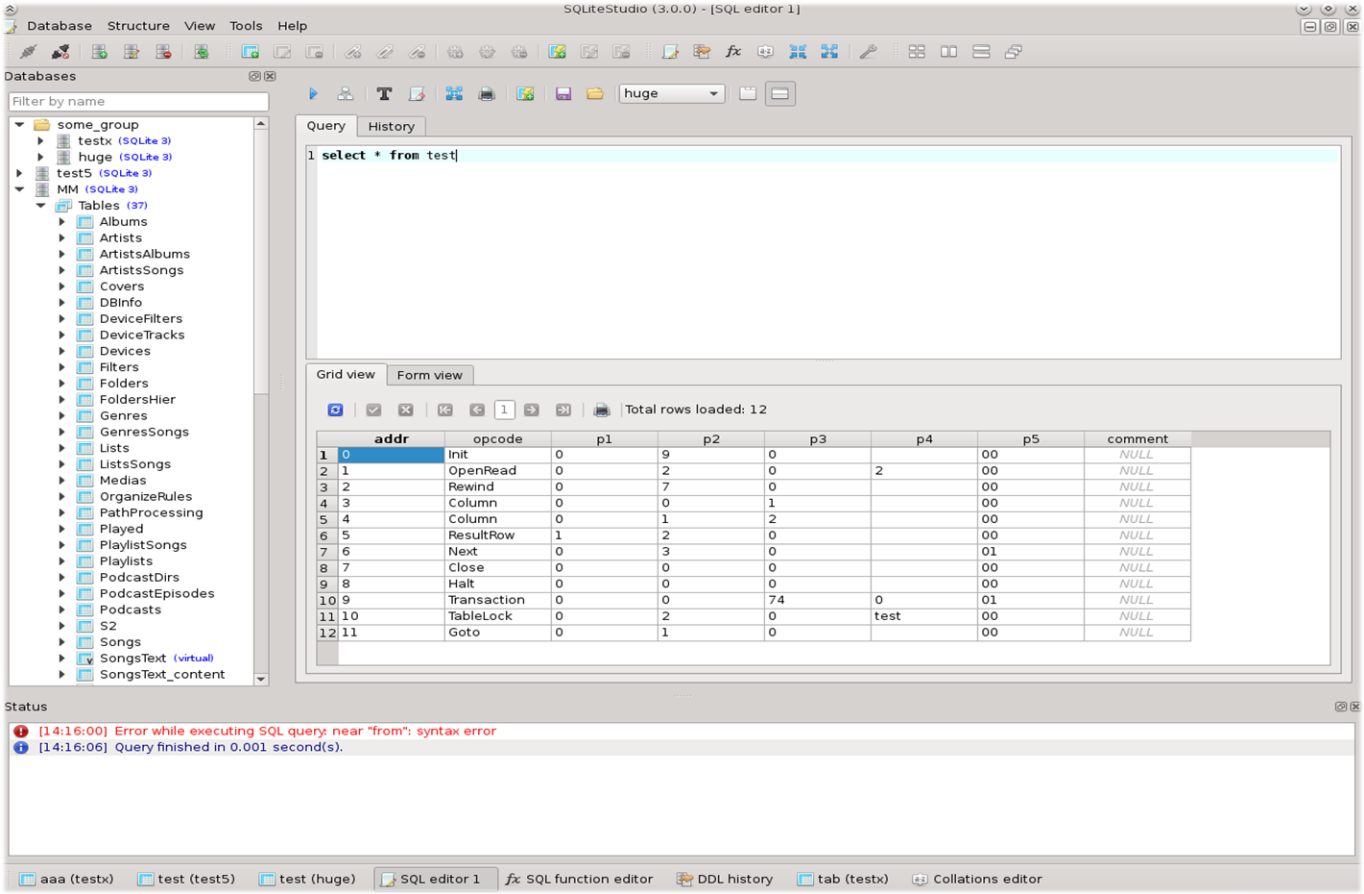
- ORDER BY *key_part1* DESC, *key_part2* DESC;
- -ORDER BY *key_part1* ASC, *key_part2* ASC;

SQLITE

- relational database management system contained in a C programming library
- designed in the spring of 2000 by Richard Hipp (part of the software for guided missile destroyers)
- open source
- light (takes only 0,3MB)
- one file (.sqlite or .db)
- implements most of the SQL-92 standard for SQL
- suitable for embedded devices and small-medium websites

SQLiteStudio

- freeware, portable (no installation needed)
- https://sqlitestudio.pl



Student's Task

1. Download SQLiteStudio from https://sqlitestudio.pl
2. Create database uni_{your_surname}
3. Create tables and choose suitable data types for their columns:

   student(student_id,name,surname,dateOfBirth,yearEnrolled),

   course(course_id,name,creditPoints,yearCommenced)

   staff(employee_id,name,surname,jobTitle)

   program(program_id,name,creditPoints, yearCommenced)

3. Write instert queries and fill tables with random data (min. 10 rows per table)
4. Present all students whose name starts with „W"
5. Present all students who are currenty at 4th year

Student's Task

7. Present all courses from table course starting with these which have the highest numer of credit points
8. Change the name of the student with lowest student_id into Adam
9. Change all values from column {name} in course table into uppercase
10. Delete the oldest student in table student
11. Remove column yearCommenced from course table
12. Rename table staff into employee
13. Send data base file via email to instructor

References

1. **Williams H., S. Tahaghoghi (2009). Learning MySQL., O'Reilly Media**
2. **Churcher C. (2007), Beginning Database Design.,Apress**

Links

- **http://www.studytonight.com/dbms/overview-of-dbms**
- **https://www.lucidchart.com/pages/er-diagrams/c?er=1**
- **https://www.python.org/**