NOVA IMS

Information Management School

Storing and Retrieving Data

Lecture 3

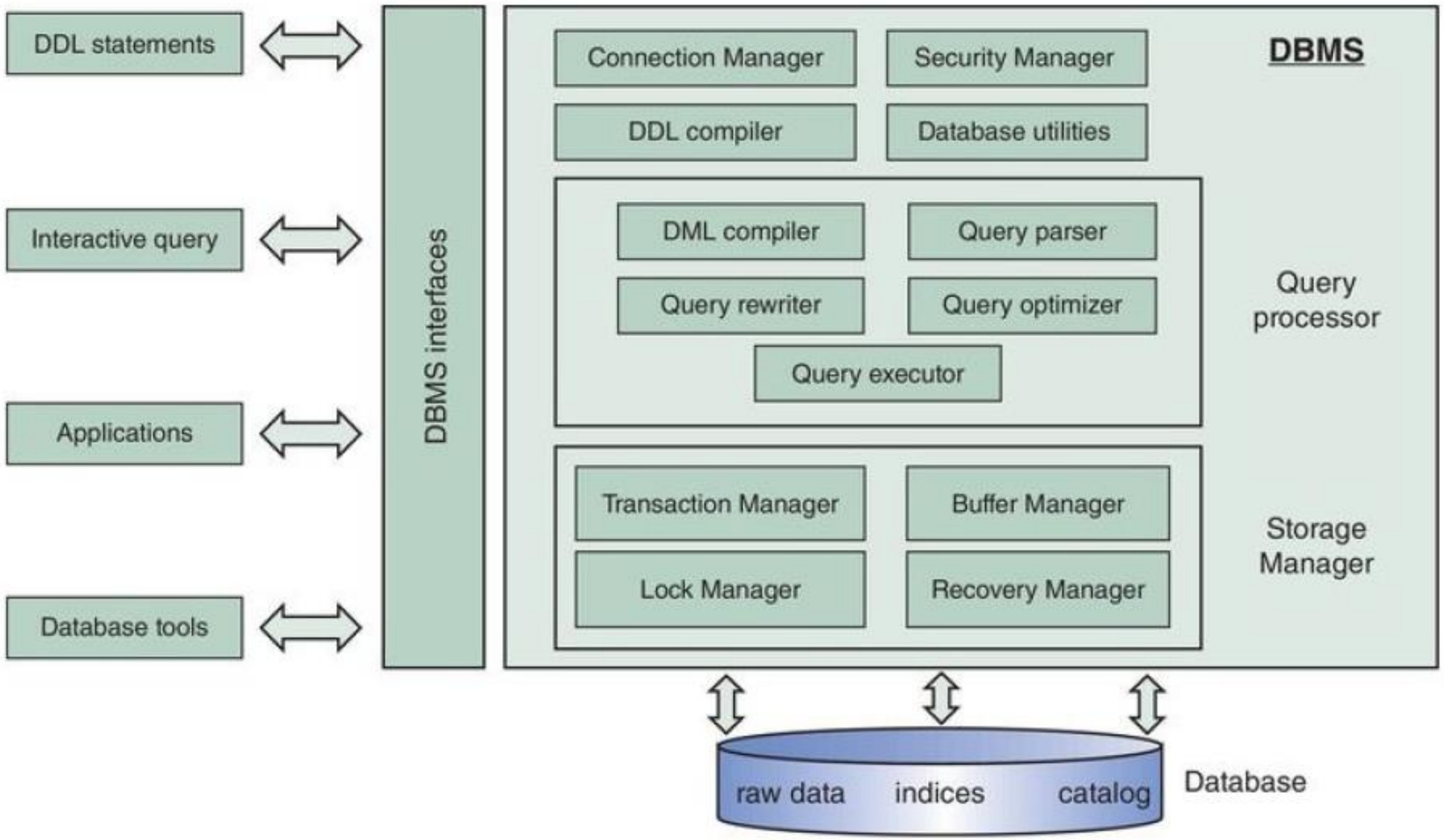
Structured Query Language (SQL) – Part 1

Lecturer: Mijail Naranjo-Zolotov
Email: mijail.naranjo@novaims.unl.pt

Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

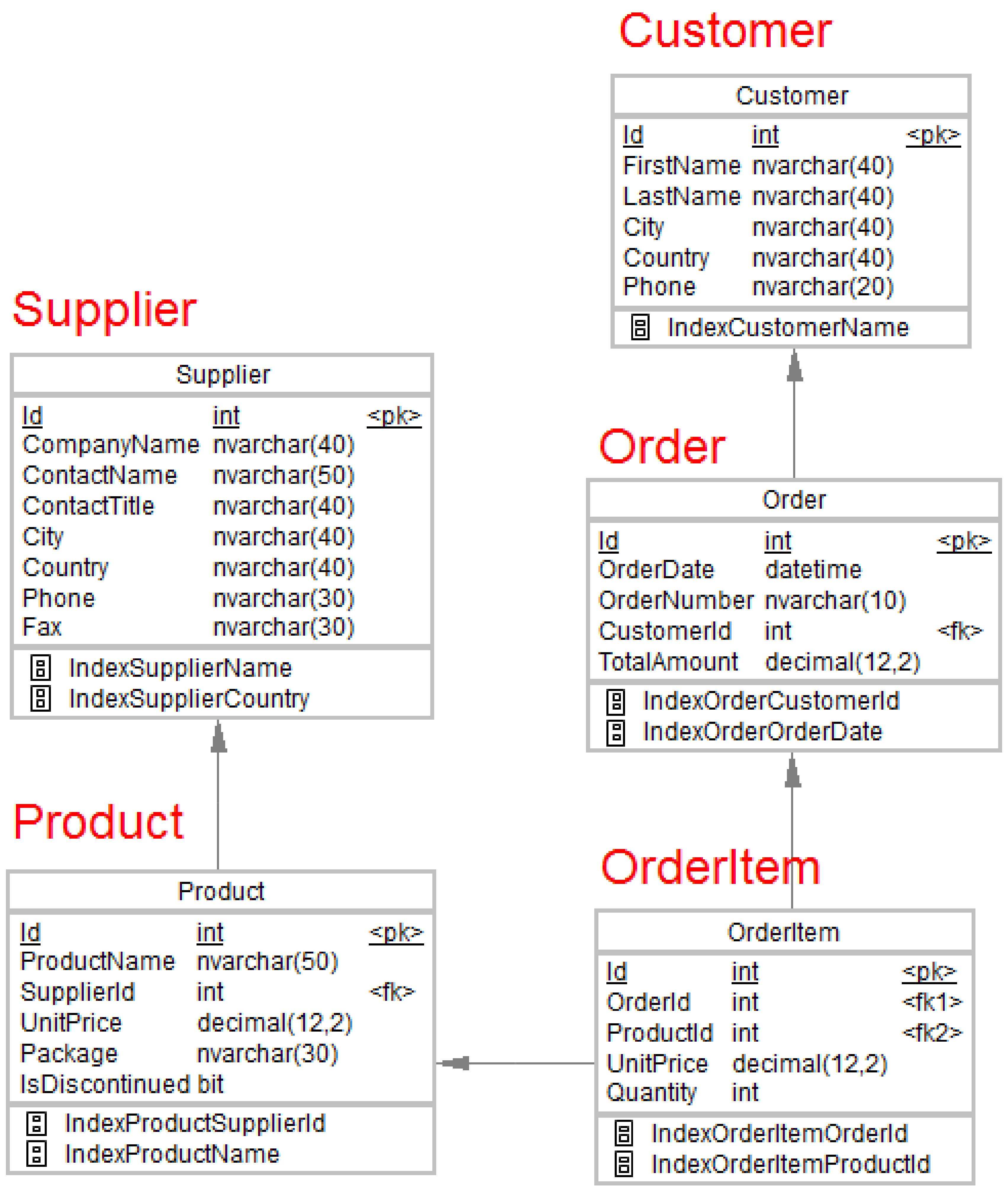
Previous class

Architecture of a DBMS



Relational Database Management System (RDBMS)

Information is stored in **tables**, and relations between data materialize in relations between tables



Customer

Customer		
Id	int	<pk>
FirstName	nvarchar(40)	
LastName	nvarchar(40)	
City	nvarchar(40)	
Country	nvarchar(40)	
Phone	nvarchar(20)	
IndexCustomerName		

Supplier

Supplier		
Id	int	<pk>
CompanyName	nvarchar(40)	
ContactName	nvarchar(50)	
ContactTitle	nvarchar(40)	
City	nvarchar(40)	
Country	nvarchar(40)	
Phone	nvarchar(30)	
Fax	nvarchar(30)	
IndexSupplierName		
IndexSupplierCountry		

Order

Order		
Id	int	<pk>
OrderDate	datetime	
OrderNumber	nvarchar(10)	
CustomerId	int	<fk>
TotalAmount	decimal(12,2)	
IndexOrderCustomerId		
IndexOrderOrderDate		

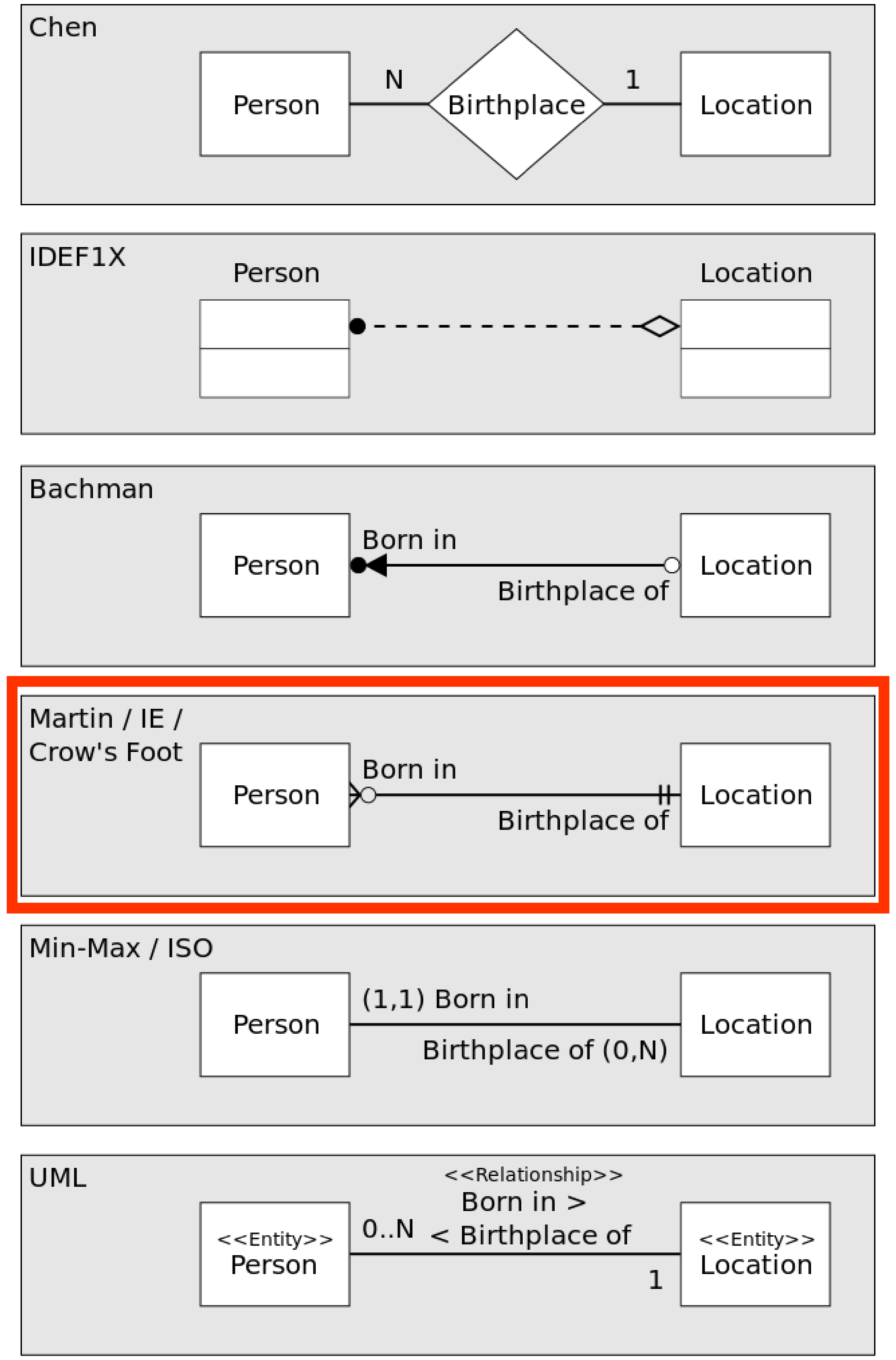
Product

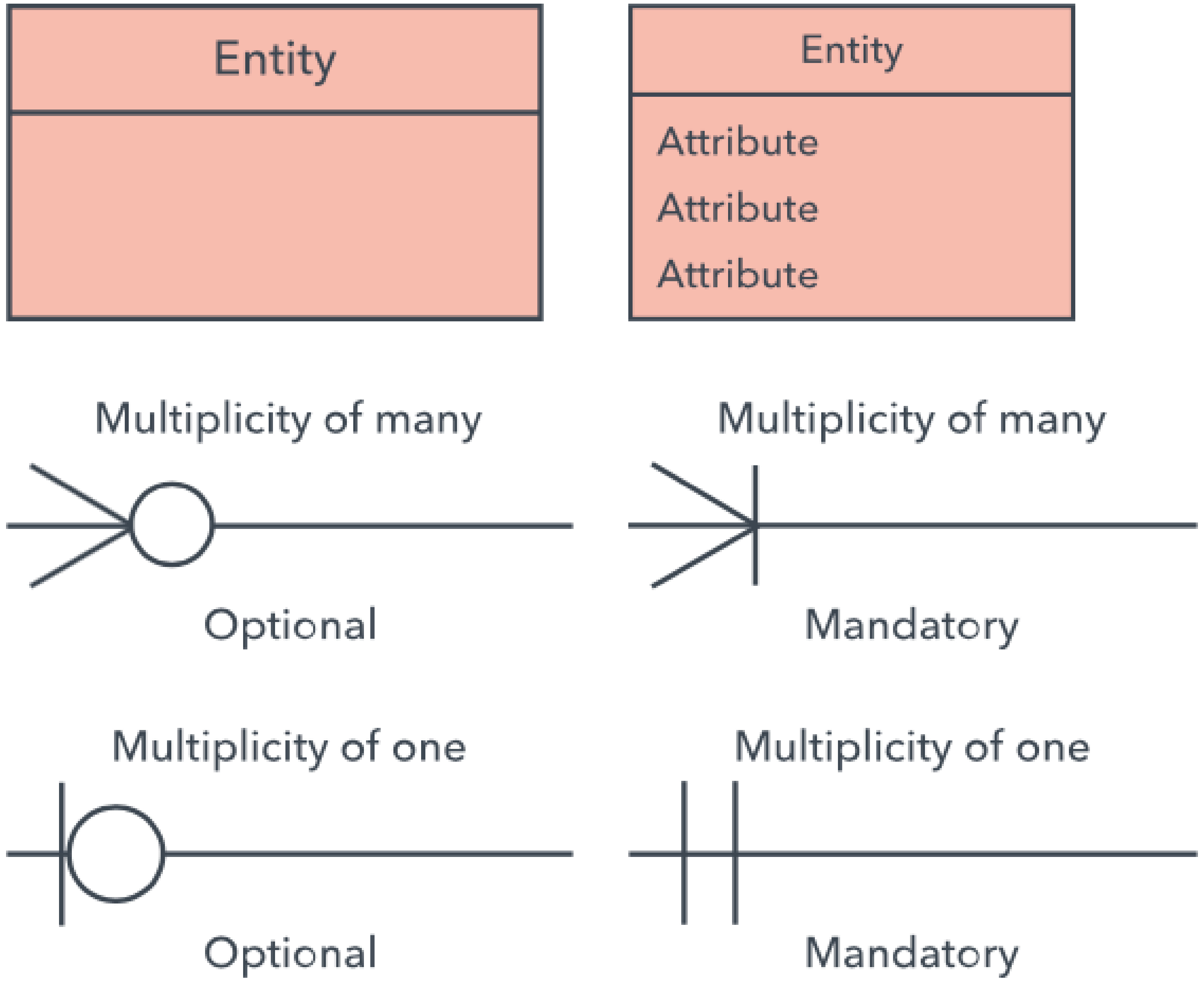
Product		
Id	int	<pk>
ProductName	nvarchar(50)	
SupplierId	int	<fk>
UnitPrice	decimal(12,2)	
Package	nvarchar(30)	
IsDiscontinued	bit	
IndexProductSupplierId		
IndexProductName		

OrderItem

OrderItem		
Id	int	<pk>
OrderId	int	<fk1>
ProductId	int	<fk2>
UnitPrice	decimal(12,2)	
Quantity	int	
IndexOrderItemOrderId		
IndexOrderItemProductId		

Crow’s foot Notation for the ERD





Today

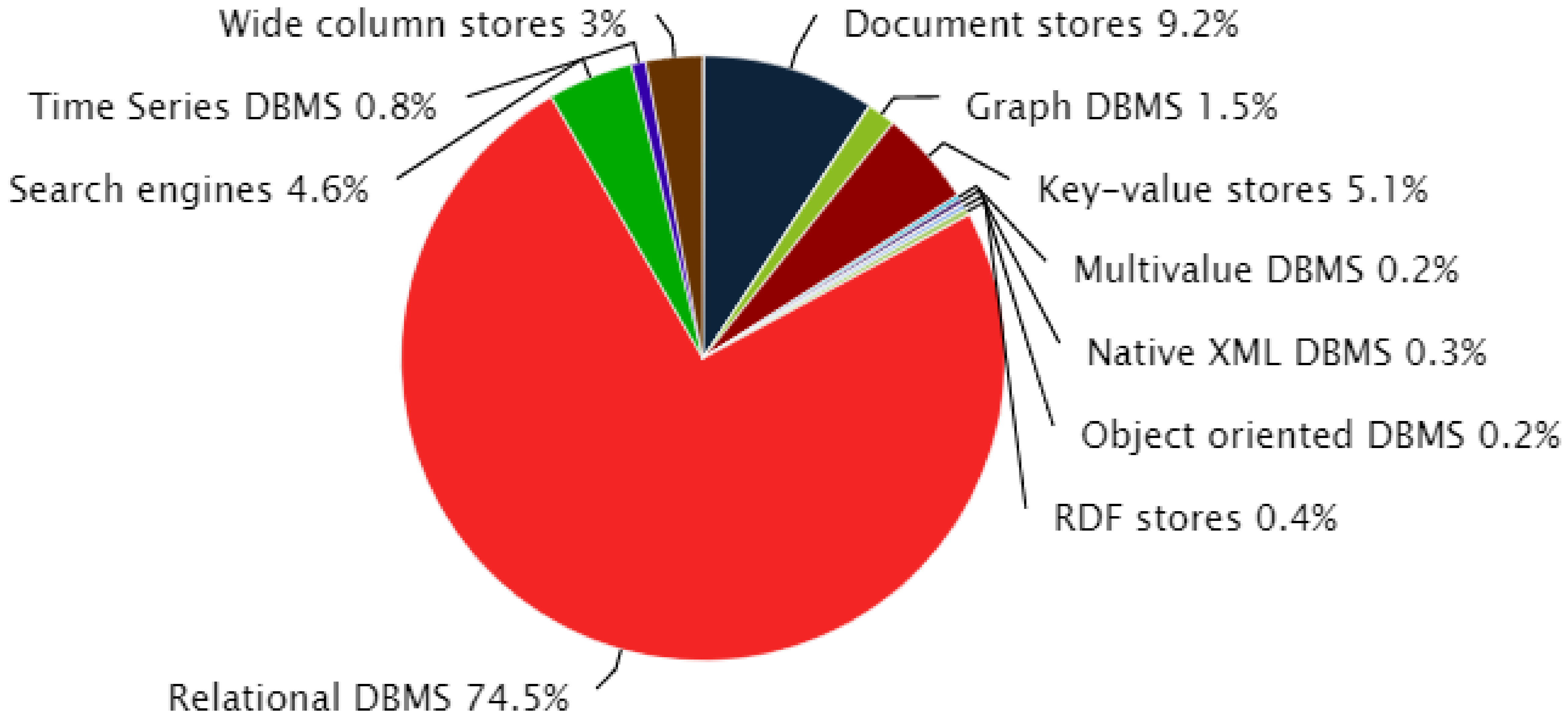
- Is SQL dead?
- Create a database
- Create tables

Is SQL dead?

Image source: https://www.rd.com/wp-content/uploads/2018/09/Dinosaur.jpg

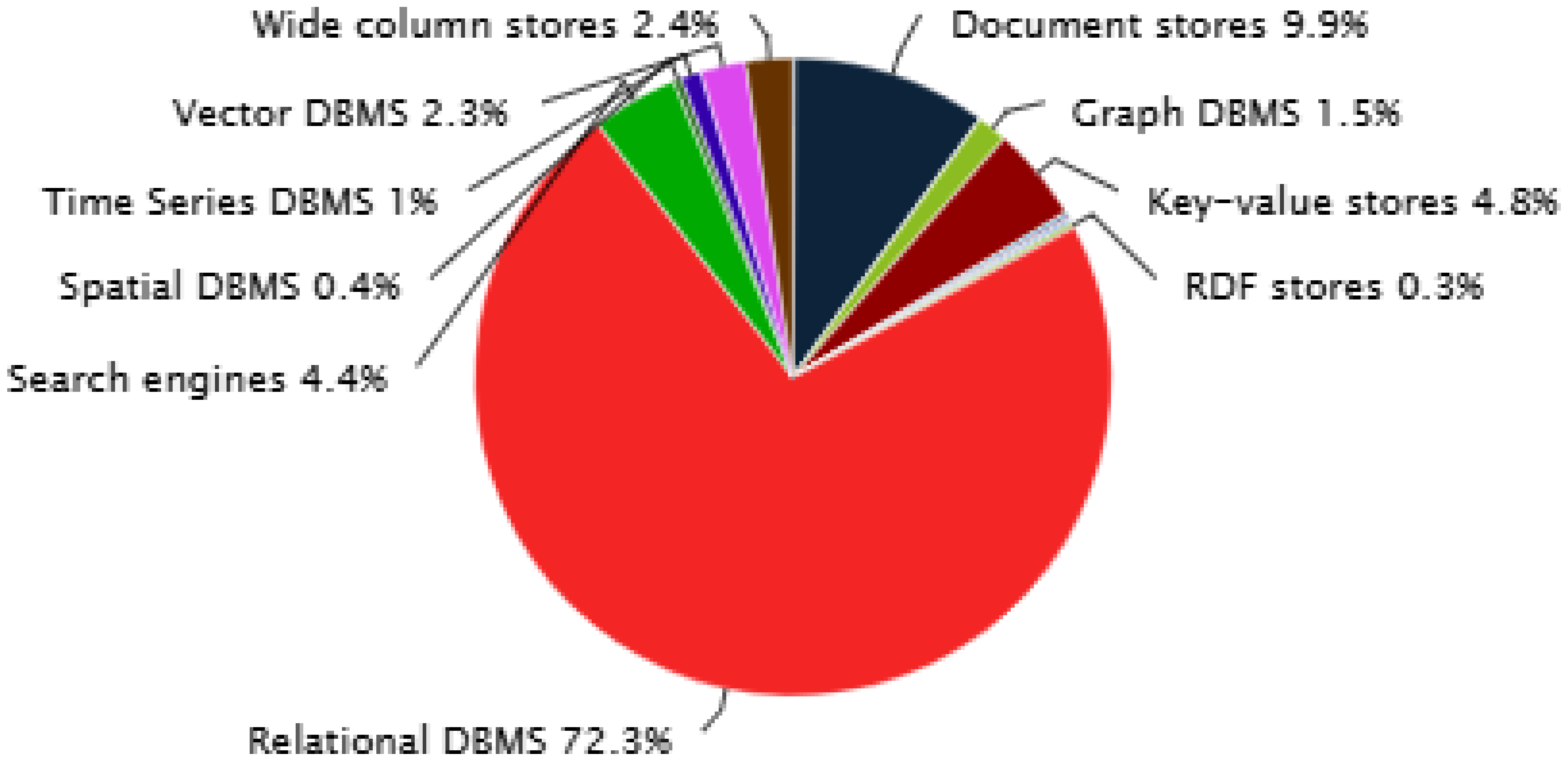
Let’s compare the two years (2020 - 2024)

2020



This chart shows the popularity of each category. More on how the scores are calculated: https://db-engines.com/en/ranking_definition

2024



Key Characteristics of SQL

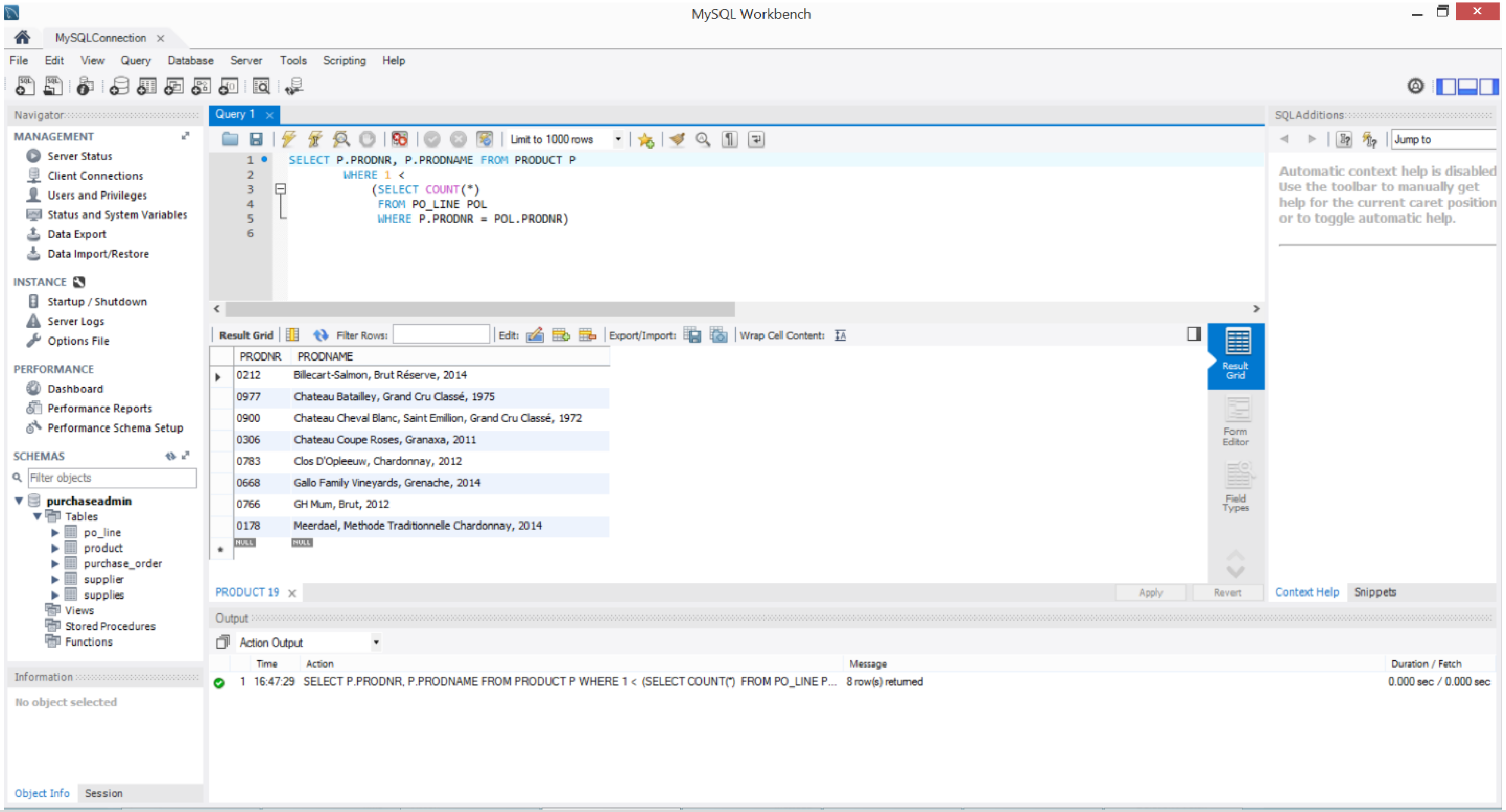
- First version, SQL-86 in 1986, most recent version in 2011 (SQL:2011)
- Accepted by the American National Standards Institute (ANSI) in 1986 and by the International Organization for Standardization (ISO) in 1987
- Each vendor provides its own implementation (also called SQL dialect) of SQL

Key Characteristics of SQL

- Set-oriented and declarative
- Free-form language
- Case insensitive
- Can be used both interactively from a command prompt or executed by a program

Key Characteristics of SQL

Example: Executed from MySQL workbench



```sql
SELECT P.PRODNR, P.PRODNAME FROM PRODUCT P
       WHERE 1 <
           (SELECT COUNT(*)
            FROM PO_LINE POL
            WHERE P.PRODNR = POL.PRODNR)
```

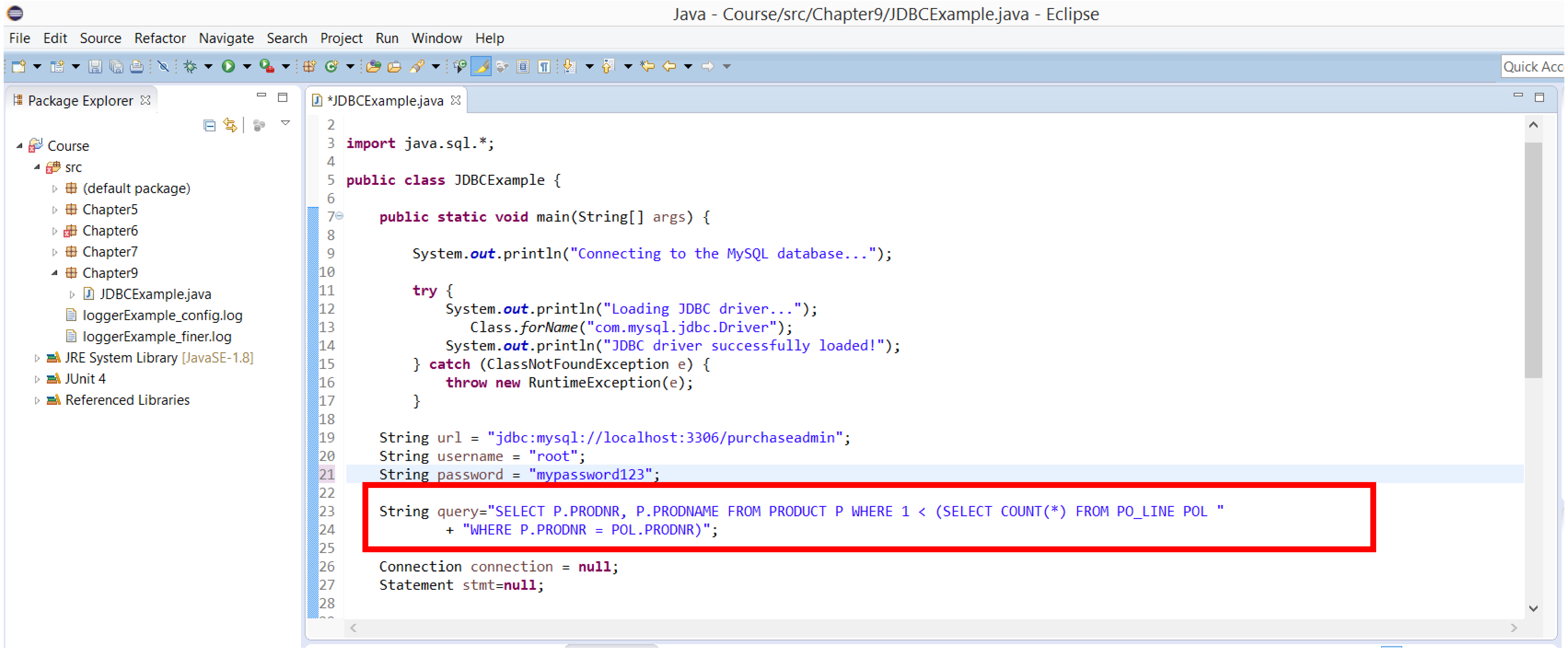
PRODNR	PRODNAME
0212	Billecart-Salmon, Brut Réserve, 2014
0977	Chateau Batailley, Grand Cru Classé, 1975
0900	Chateau Cheval Blanc, Saint Emilion, Grand Cru Classé, 1972
0306	Chateau Coupe Roses, Granaxa, 2011
0783	Clos D'Opleeuw, Chardonnay, 2012
0668	Gallo Family Vineyards, Grenache, 2014
0766	GH Mum, Brut, 2012
0178	Meerdael, Methode Traditionnelle Chardonnay, 2014
NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
   Time     Action                                                                     Message               Duration / Fetch
1  16:47:29 SELECT P.PRODNR, P.PRODNAME FROM PRODUCT P WHERE 1 < (SELECT COUNT(*) FROM PO_LINE P...  8 row(s) returned   0.000 sec / 0.000 sec
```

Key Characteristics of SQL

Example: Embedded in java code.



```java

import java.sql.*;

public class JDBCExample {

    public static void main(String[] args) {

        System.out.println("Connecting to the MySQL database...");

        try {
            System.out.println("Loading JDBC driver...");
               Class.forName("com.mysql.jdbc.Driver");
            System.out.println("JDBC driver successfully loaded!");
        } catch (ClassNotFoundException e) {
            throw new RuntimeException(e);
        }

    String url = "jdbc:mysql://localhost:3306/purchaseadmin";
    String username = "root";
    String password = "mypassword123";

    String query="SELECT P.PRODNR, P.PRODNAME FROM PRODUCT P WHERE 1 < (SELECT COUNT(*) FROM PO_LINE POL "
            + "WHERE P.PRODNR = POL.PRODNR)";

    Connection connection = null;
    Statement stmt=null;

```

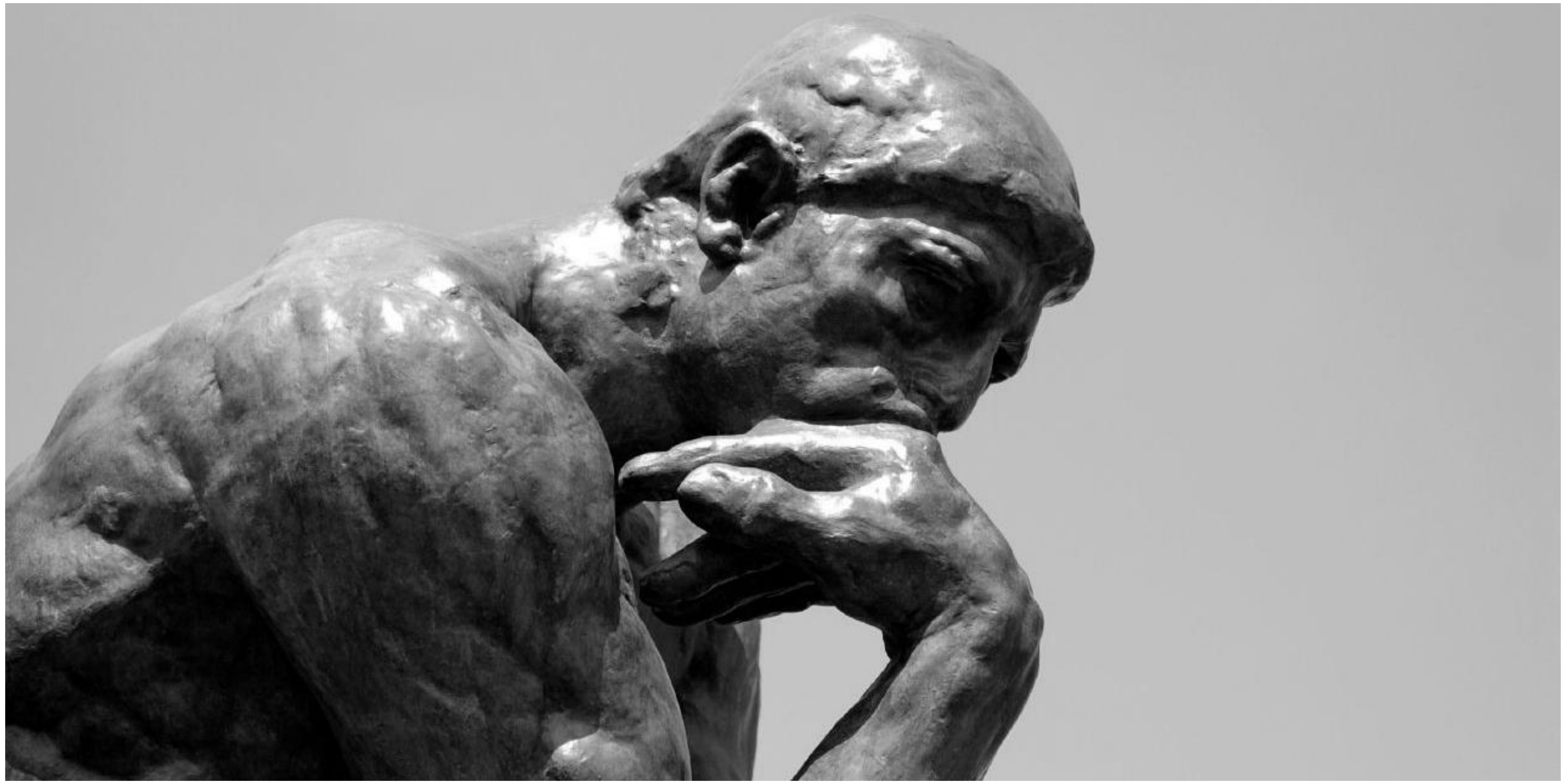
Recommended tutorials!!

Many of the examples are from:

https://www.mysqltutorial.org/

Philosophical moment

Will AI replace the SQL Analyst anytime soon?



Suggested readings

Will AI replace the SQL analyst anytime soon?

Scientific articles for home reading (shared in Moodle):

- Li, G., Zhou, X., & Cao, L. (2021, June). AI meets database: AI4DB and DB4AI. In Proceedings of the 2021 International Conference on Management of Data (pp. 2859-2866).
- Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2022). Do users write more insecure code with AI assistants?. arXiv preprint arXiv:2211.03622.

Create a Database

Check existing databases

```
SHOW DATABASES;
```

Create Database

```
CREATE DATABASE [IF NOT EXISTS] database_name
[CHARACTER SET charset_name]
[COLLATE collation_name]
```

A MySQL character set is a set of characters that are legal in a string

A MySQL collation is a set of rules used to compare characters in a particular character set.

use Database

```
USE database_name;
```

Drop Database

```
DROP DATABASE [IF EXISTS] database_name;
```

See tables present in the database

```
SHOW TABLES;
```

Create Tables

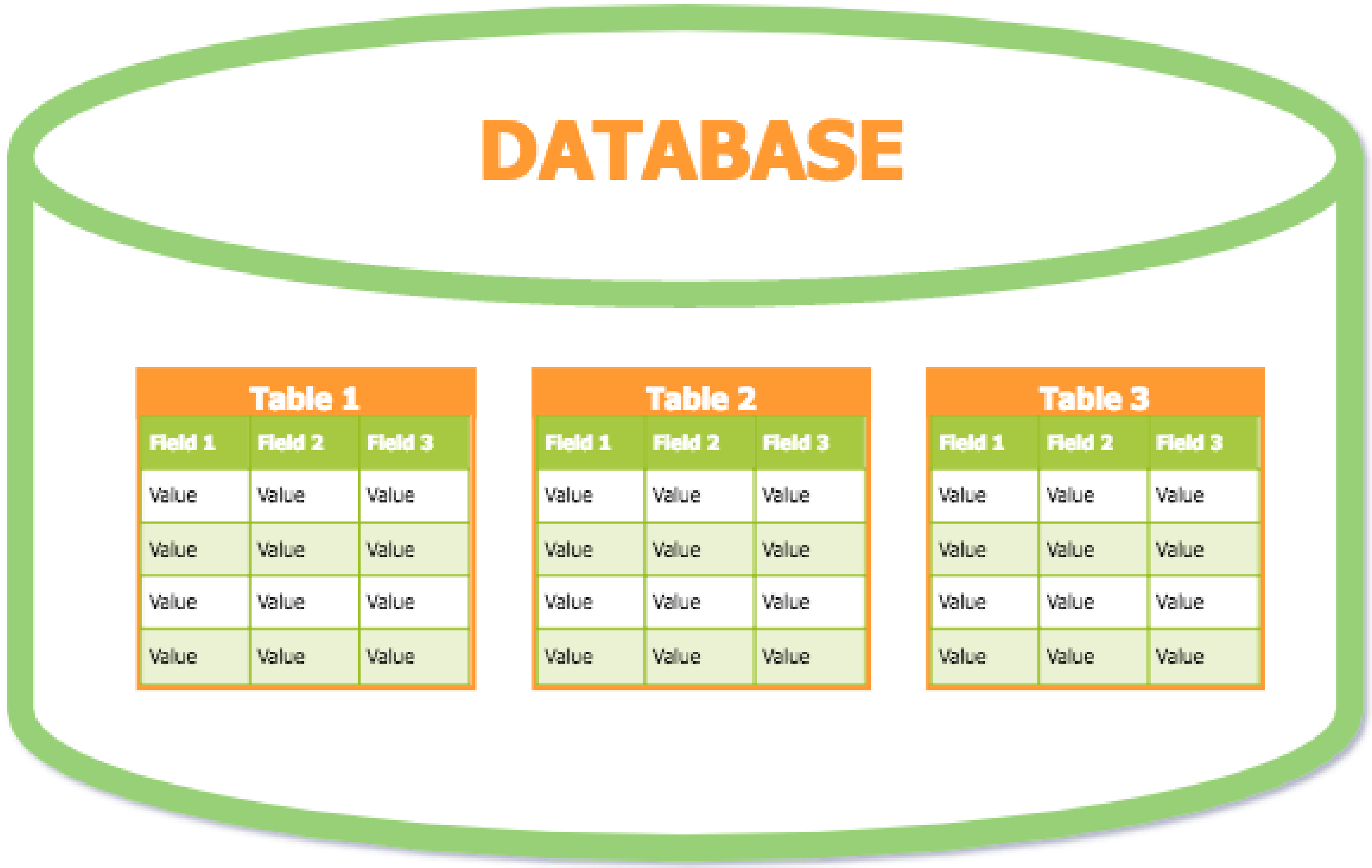


Table 1		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

Table 2		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

Table 3		
Field 1	Field 2	Field 3
Value	Value	Value
Value	Value	Value
Value	Value	Value
Value	Value	Value

Image source: https://database.guide/database-tutorial-part-1-about-databases-creating-databases-tables/

Table creation

Create a Table

```
CREATE TABLE [IF NOT EXISTS] table_name(
col1 data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT]
col2 data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT]
col3 data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT]
) ENGINE=storage_engine;
```

Table creation

Create a Table

```
CREATE TABLE [IF NOT EXISTS] table_name(
column_name data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT]
) ENGINE=storage_engine;
```

The column_name specifies the name of the column.
Each column has a specific data type and maximum length e.g.,VARCHAR(255)

Table creation – data types

The Data Type specifies what type of data the column can hold.

CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TEXT	Holds a string with a maximum length of 65,535 characters
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

Table creation

Create a Table

```
CREATE TABLE [IF NOT EXISTS] table_name(
column_name data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT]
) ENGINE=storage_engine;
```

The **NOT NULL** indicates that the column does not allow **NULL**

Table creation

Create a Table

```
CREATE TABLE [IF NOT EXISTS] table_name(
column_name data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT]
) ENGINE=storage_engine;
```

The **DEFAULT** value is used to specify the default value of the column

Table creation

Create a Table

```
CREATE TABLE [IF NOT EXISTS] table_name(
column_name data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT]
) ENGINE=storage_engine;
```

The **AUTO_INCREMENT** indicates that the value of the column is incremented by one automatically whenever a new row is inserted into the table. Each table has one and only one **AUTO_INCREMENT** column

Table creation

Create a Table

```
CREATE TABLE [IF NOT EXISTS] table_name(
column_name data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT]
) ENGINE=storage_engine;
```

Third, you can optionally specify the storage engine for the table in the **ENGINE** clause. You can use any storage engine such as InnoDB and MyISAM. If you don’t explicitly declare the storage engine, MySQL will use InnoDB by default.

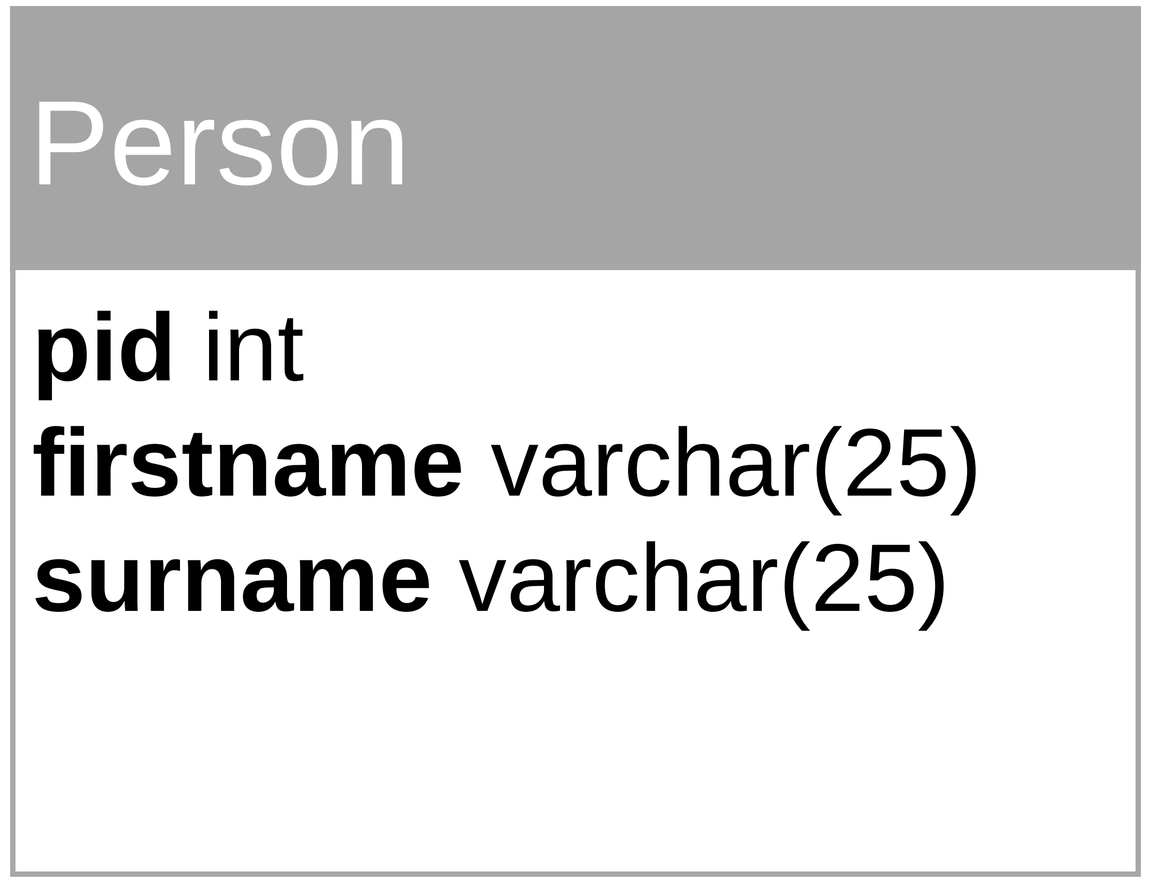
Table creation – primary key

Primary Keys

```
CREATE TABLE [IF NOT EXISTS] table_name(
col1 data_type(length) [PRIMARY KEY]
col2 data_type(length) [PRIMARY KEY]
col3 data_type(length) [PRIMARY KEY]
);
```

```
CREATE TABLE [IF NOT EXISTS] table_name(
col1 data_type(length)
col2 data_type(length)
col3 data_type(length)
PRIMARY KEY(col1, col2, …)
);
```

Table creation - example



Person

pid int
firstname varchar(25)
surname varchar(25)

```
CREATE TABLE Person (
pid int AUTO_INCREMENT NOT NULL,
firstname varchar(25) NOT NULL,
surname varchar(25) NOT NULL,
);
```

What are we missing?

Person
pid int **firstname** varchar(25) **surname** varchar(25)

```
CREATE TABLE Person (
pid int AUTO_INCREMENT NOT NULL,
firstname varchar(25) NOT NULL,
surname varchar(25) NOT NULL,
PRIMARY KEY (pid)
);
```

What are we missing? The Primary Key

Table creation - Foreign keys

```
CREATE TABLE [IF NOT EXISTS] table_name(
col1 data_type(length)
PRIMARY KEY(col1, col2, …)
CONSTRAINT constraint_name
FOREIGN KEY (columns)
REFERENCES parent_table (columns)
ON DELETE action
ON UPDATE action
);
```

Table creation - Foreign keys

```
CREATE TABLE [IF NOT EXISTS] table_name(
col1 data_type(length)
PRIMARY KEY(col1, col2, …)
CONSTRAINT constraint_name
FOREIGN KEY (columns)
REFERENCES parent_table(columns)
ON DELETE action
ON UPDATE action
);
```

The **CONSTRAINT** clause allows you to define constraint name for the foreign key constraint. If you omit it, MySQL will generate a name automatically.

Table creation - Foreign keys

```
CREATE TABLE [IF NOT EXISTS] table_name(
col1 data_type(length)
PRIMARY KEY(col1, col2, …)
CONSTRAINT constraint_name
FOREIGN KEY (columns)
REFERENCES parent_table(columns)
ON DELETE action
ON UPDATE action
);
```

The **FOREIGN KEY** clause specifies the columns in the child table that refers to primary key columns in the parent table.

Table creation - Foreign keys

```
CREATE TABLE [IF NOT EXISTS] table_name(
col1 data_type(length)
PRIMARY KEY(col1, col2, …)
CONSTRAINT constraint_name
FOREIGN KEY (columns)
REFERENCES parent_table(columns)
ON DELETE action
ON UPDATE action
);
```

The **REFERENCES** clause specifies the parent table and its columns to which the columns in the child table refer. The number of columns in the child table and parent table specified in the **FOREIGN KEY** and **REFERENCES** must be the same.

Table creation - Foreign keys

```
CREATE TABLE [IF NOT EXISTS] table_name(
col1 data_type(length)
PRIMARY KEY(col1, col2, …)
CONSTRAINT constraint_name
FOREIGN KEY (columns)
REFERENCES parent_table(columns)
ON DELETE action
ON UPDATE action
);
```

possible actions :
CASCADE, SET NULL, NO ACTION, RESTRICT

The **ON DELETE** clause allows you to define what happens to the records in the child table when the records in the parent table are deleted. If you omit the **ON DELETE** clause and delete a record in the parent table that has records in the child table refer to, MySQL will reject the deletion.

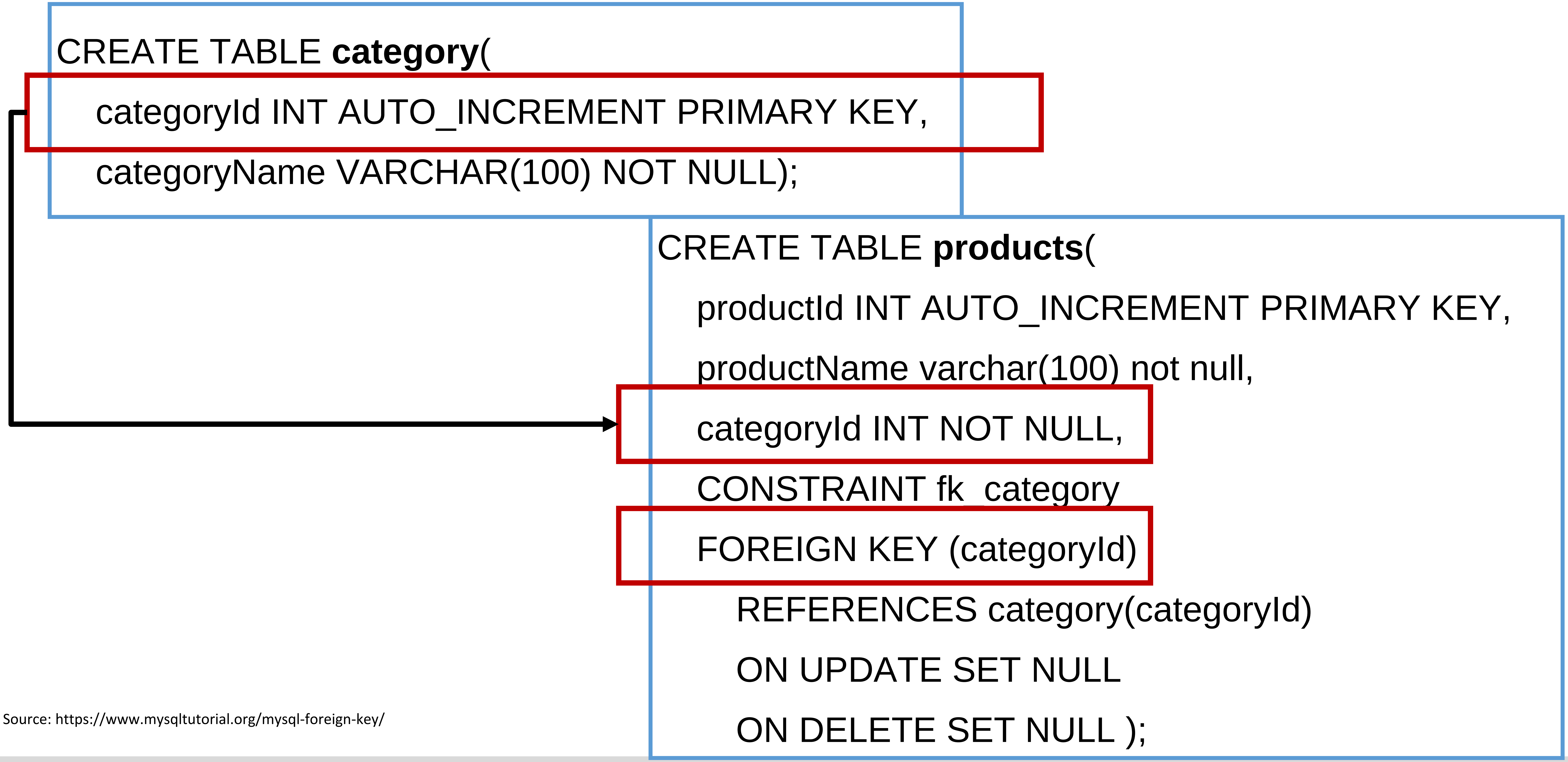
Table creation - Foreign keys

```
CREATE TABLE [IF NOT EXISTS] table_name(
col1 data_type(length)
PRIMARY KEY(col1, col2, …)
CONSTRAINT constraint_name
FOREIGN KEY foreign_key_name (columns)
REFERENCES parent_table(columns)
ON DELETE action
ON UPDATE action
);
```

possible actions :
CASCADE, SET NULL, NO ACTION, RESTRICT

The **ON UPDATE** clause enables you to specify what happens to the rows in the child table when rows in the parent table are updated. You can omit the **ON UPDATE** clause to let MySQL reject any updates to the rows in the child table when the rows in the parent table are updated.

Table creation - Foreign keys - Example



```
CREATE TABLE category(
    categoryId INT AUTO_INCREMENT PRIMARY KEY,
    categoryName VARCHAR(100) NOT NULL);
```

```
CREATE TABLE products(
    productId INT AUTO_INCREMENT PRIMARY KEY,
    productName varchar(100) not null,
    categoryId INT NOT NULL,
    CONSTRAINT fk_category
    FOREIGN KEY (categoryId)
        REFERENCES category(categoryId)
        ON UPDATE SET NULL
        ON DELETE SET NULL );
```

Source: https://www.mysqltutorial.org/mysql-foreign-key/

Drop a table

```
DROP TABLE [IF EXISTS] table_name [, table_name] …
```

The **DROP TABLE** statement removes a table and its data permanently from the database. In MySQL, you can also remove multiple tables using a single **DROP TABLE** statement, each table is separated by a comma (,).

Drop a table

```
DROP TABLE [IF EXISTS] table_name [, table_name] …
```

The **IF EXISTS** addition helps you prevent from attempt of removing non-existent tables. When you use **IF EXISTS** addition, MySQL generates a NOTE, which can be retrieved by using the SHOW WARNING statement. It is important to note that the **DROP TABLE** statement removes all existing tables and issues an error message or a NOTE when you have a non-existent table in the list.

Quick quiz

https://b.socrative.com/login/student/

Room: SRD24

END OF LECTURE 3