Практический курс тестирования программного обеспечения

Урок 6

Test Club 2014

http://www.testclub.com.ua

План занятия:

1. Основы реляционных баз данных
2. Работа с SQL

1. Основы реляционных баз данных

Базой данных (БД) называется организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность сведений об объектах, процессах, событиях или явлениях, относящихся к некоторой предметной области, теме или задаче. Она организована таким образом, чтобы обеспечить информационные потребности пользователей, а также удобное хранение этой совокупности данных, как в целом, так и любой ее части.

Другими словами это хранилище данных.

1. Основы реляционных баз данных

Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного вида. Каждая строка таблицы содержит данные об одном объекте (например, автомобиле, компьютере, пользователе), а столбцы таблицы содержат различные характеристики этих объектов - атрибуты (например, номер двигателя, марка процессора, телефоны фирм или клиентов).

1. Основы реляционных баз данных

Таким образом все данные представлены в виде таблиц, разбитых на строки и столбцы. На их пересечении расположены данные.

- Столбцы называют **полями** или **атрибутами**
- Строки называют **записями** или **кортежами**

1. Основы реляционных баз данных

Основные свойства таблиц в реляционных БД:

1. В таблице не может быть двух одинаковых строк
2. Столбцы располагаются в определенном порядке, который определяется при создании таблицы. В таблице может быть ни одной строки, но должен быть хотя бы один столбец
3. У каждого столбца есть уникальное имя(в пределах таблицы) и все значения в столбце имеют один тип данных(числа, текст, дата…)
4. На пересечении каждого столбца и строки может находиться атомарное значение(одно значение, не состоящее из группы значений)

1. Основы реляционных баз данных

Первичный ключ(Primary key) – столбец, значения которого во всех строках различны. Первичные ключи менять нельзя.

Вторичный ключ(Foreign key) – столбец, каждое значение которого соответствует какому либо первичному ключу из другой таблицы.

1. Основы реляционных баз данных

СУБД (системы управления базами данных) – совокупность языковых и программных средств, которые осуществляют доступ к данным; позволяют их создавать, менять, удалять; обеспечивают безопасность и т.д.

Другими словами СУБД – система, позволяющая создавать базы данных и манипулировать сведениями из них.

Несмотря на наличие международного стандарта, многие компании, занимающиеся разработкой СУБД (например, Oracle, Microsoft, MySQL), вносят изменения в язык SQL, применяемый в разрабатываемой СУБД. Таким образом, появляются специфичные для каждой конкретной СУБД диалекты языка SQL.

1. Основы реляционных баз данных

Любая СУБД позволяет выполнять следующие операции с данными:

- добавление записей в таблицы;
- удаление записей из таблицы;
- обновление значений некоторых полей в одной или нескольких записях в таблицах БД;
- поиск одной или нескольких записей, удовлетворяющих заданному условию.

1. Основы реляционных баз данных

Для выполнения этих операций применяется механизм запросов. Результатом выполнения запросов является либо отобранное по определенным критериям множество записей, либо изменения в таблицах. Запросы к базе формируются на специально созданном для этого языке, который называется **SQL**

2. Работа с SQL

Structured Query Language(SQL) – стандартный язык управления реляционными базами данных с архитектурой клиент-сервер.

1. Download client: http://bit.ly/Uwfz43
2. Connection name: any name
3. Host: *h04.hvosting.ua*
4. Username: *user*(1..10)*
5. Password: *12*
6. Schema: *runtest*

2. Работа с SQL

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

2. Работа с SQL

SQL can be divided into two parts:

1. Data Definition Language (DDL)
2. The Data Manipulation Language (DML)

2. Работа с SQL

DDL statements are used to define the database structure:

- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

2. Работа с SQL

DML statements are used for managing data:

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database

2. Работа с SQL

2.1. DDL

Create new table:

1. Right click on DB(i.e., “runtest”)
2. Choose “Create new” -> “Table” option
3. Specify table name (“User*_test_table”)
4. Add and configure columns
5. Check the code for table creation (“CREATE code” tab)
6. Click on “Save” button

2. Работа с SQL

2.1. DDL

DB data types:

- **INT** – any number from “-2,147,483,648” to “2,147,483,648”
- **VARCHAR(x)** – Where x is the number of characters to store.
- **DECIMAL(p,s)** – where p is a precision value; s is a scale value. For example, numeric(6,2).
- **FLOAT** – store fractional (non-integer) values
- **TEXT** - maximum string length is 2 147 483 647 characters*
- **BIT** – stands for Boolean type and can be “0” or “1”

2. Работа с SQL
2.1. DDL

Rules for fields in DB:

- **Auto increment** – RDBMS will automatically increment value of this field for every insert operation
- **Primary key** – data for this filed will be used for building related queries between tables. This field should be unique
- **Not null** – the value of this field is mandatory
- **Unique** – value of this field should be unique
- **Zero fill** – if value is not indicated, RDBMS will set “0” in this field. Value “” and “0” is not equal for DB

2. Работа с SQL

2.1. DDL

Populate the newly created table:

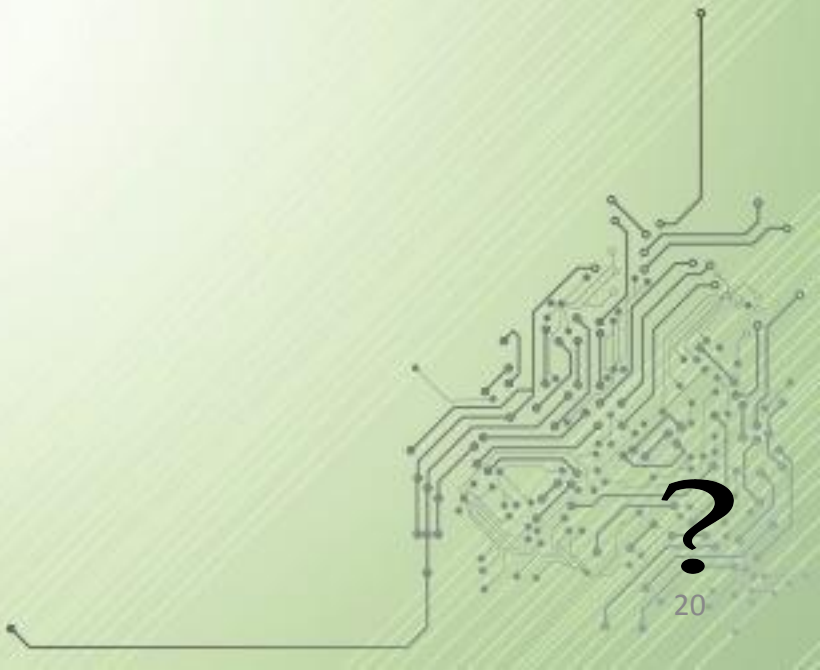
1. Select appropriate table
2. Select “Data” tab
3. Click on “+” button
4. Populate all columns in the newly added row
5. Click somewhere outside the row to apply changes

2. Работа с SQL

2.1. DDL

Delete the table:

1. Right click on appropriate table
2. Choose “Drop …” option
3. Confirm drop action



2. Работа с SQL
2.2. DML

Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. ”Persons" or "Orders"). Tables contain records (rows) with data.

Below is an example of a table called "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The table above contains three records (one for each person) and five columns (P_Id, LastName, FirstName, Address, and City).

2. Работа с SQL
2.2. DML

The SQL SELECT Statement:

Most of the actions you need to perform on a database are done with SQL statements.

The following SQL statement will select **all** the records in the "Persons" table:

SELECT * FROM Persons; *

The result-set will look like this

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

SQL is not case sensitive, but field names and values ARE case sensitive

2. Работа с SQL
2.2. DML

The SQL SELECT Statement:

Now we want to select the content of the columns named "LastName" and "FirstName" from the table above.

We use the following SELECT statement:

SELECT LastName, FirstName FROM Persons;

The result-set will look like this:

LastName	FirstName
Hansen	Ola
Svendson	Tove
Pettersen	Kari

?

2. Работа с SQL
2.2. DML

The SQL SELECT DISTINCT Statement:

In a table, some of the columns may contain duplicate values. This is not a problem, however, sometimes you will want to list only the different (distinct) values in a table.

The **DISTINCT** keyword can be used to return only distinct (different) values.

SELECT DISTINCT City FROM Persons;

The result-set will look like this:

City
Sandnes
Stavanger

?

2. Работа с SQL
2.2. DML

SQL WHERE Clause:

The WHERE clause is used to extract only those records that fulfill a specified criterion.

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select only the persons living in the city "Sandnes" from the table above:

SELECT * FROM Persons WHERE City = 'Sandnes';

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

2. Работа с SQL
2.2. DML

Quotes Around Text Fields

SQL uses single quotes around text values.

Although, numeric values should not be enclosed in quotes.

For text values:

This is correct: **SELECT * FROM Persons WHERE FirstName = 'Tove';**

This is wrong: **SELECT * FROM Persons WHERE FirstName = Tove;**

For numeric values:

This is correct: **SELECT * FROM Persons WHERE Year=1965;**

This is wrong: **SELECT * FROM Persons WHERE Year='1965';**

2. Работа с SQL
2.2. DML

Operators Allowed in the WHERE Clause

With the WHERE clause, the following operators can be used:

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	If you know the exact value you want to return for at least one of the columns

Note: In some versions of SQL the **<>** operator may be written as **!=**

?

2. Работа с SQL
2.2. DML

The SQL AND & OR Operators:

- AND operator displays a record if both the first condition and the second condition are true.
- OR operator displays a record if either the first condition OR the second condition is true.

2. Работа с SQL
2.2. DML

AND operator example:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select only the persons with the first name equal to "Tove" AND the city equal to "Svendson":

SELECT * FROM Persons WHERE City='Sandnes' AND FirstName='Tove';

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes

2. Работа с SQL
2.2. DML

OR operator example:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select only the persons with the first name equal to "Tove" OR the first name equal to "Ola":

SELECT * FROM Persons WHERE FirstName='Tove' OR FirstName='OLA' ;

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

2. Работа с SQL
2.2. DML

Combining AND & OR:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the persons with the city equal to "Sandnes" AND the first name equal to "Tove" OR to "Kari":

SELECT * FROM Persons WHERE City='Sandness' AND (FirstName='Tove' OR FirstName='Kari');

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes

?

2. Работа с SQL
2.2. DML

The SQL LIKE Operator:

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern;

2. Работа с SQL
2.2. DML

LIKE operator example #1:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the persons from city that starts with "s":

SELECT * FROM Persons WHERE City LIKE 'S%';

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

2. Работа с SQL
2.2. DML

LIKE operator example #2:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the persons from city that ends with "s":

SELECT * FROM Persons WHERE City LIKE '%s';

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

2. Работа с SQL
2.2. DML

LIKE operator example #3:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the persons living in a city that contains the pattern "tav":

SELECT * FROM Persons WHERE City LIKE '%tav%';

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
3	Pettersen	Kari	Storgt 20	Stavanger

2. Работа с SQL
2.2. DML

LIKE operator example #4:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the persons living in a city that does NOT contain the pattern "tav", by using the NOT keyword:

SELECT * FROM Persons WHERE City NOT LIKE '%tav%';

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

?

2. Работа с SQL
2.2. DML

The SQL ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set by one or more columns.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword.

2. Работа с SQL
2.2. DML

ORDER BY keyword example #1:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select all the persons from the table above, however, we want to sort the persons by their last name:

SELECT * FROM Persons ORDER BY LastName;

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
4	Nilsen	Tom	Vingvn 23	Stavanger
3	Pettersen	Kari	Storgt 20	Stavanger
2	Svendson	Tove	Borgvn 23	Sandnes

2. Работа с SQL
2.2. DML

ORDER BY keyword example #2:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select all the persons from the table above, however, we want to sort the persons descending by their last name:

SELECT * FROM Persons ORDER BY LastName DESC;

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Tom	Vingvn 23	Stavanger
1	Hansen	Ola	Timoteivn 10	Sandnes

?

2. Работа с SQL
2.2. DML

The SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

It is possible to write the INSERT INTO statement in two forms:

1. The first form does not specify the column names where the data will be inserted, only their values:

INSERT INTO table_name VALUES (value1,value2,value3,...);

2. The second form specifies both the column names and the values to be inserted:

INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);

2. Работа с SQL
2.2. DML

INSERT INTO example #1:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to insert a new row in the "Persons" table:

INSERT INTO Persons
VALUES (4,'Nilsen', 'Johan', 'Bakken 2', 'Stavanger');

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

2. Работа с SQL
2.2. DML

INSERT INTO example #2:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

The following SQL statement will add a new row, but only add data in the "P_Id", "LastName" , "FirstName" columns:

INSERT INTO Persons (P_Id, LastName, FirstName) VALUES (5, 'Tjessem', 'Jakob');

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

?

2. Работа с SQL
2.2. DML

The SQL UPDATE Statement:

The UPDATE statement is used to update existing records in a table.

UPDATE table_name SET
column1=value1,column2=value2,...
WHERE some_column=some_value;

Note: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

2. Работа с SQL
2.2. DML

UPDATE example #1:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

Now we want to update the person "Tjessem, Jakob" in the "Persons" table:

UPDATE Persons SET Address='Nissestien 67', City='Sandnes'
WHERE LastName='Tjessem' AND FirstName='Jakob';

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

2. Работа с SQL
2.2. DML

UPDATE example #2:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

Be careful when updating records. If we had omitted the WHERE clause in the example above, like this:

UPDATE Persons SET Address='Nissestien 67', City='Sandnes';

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Nissestien 67	Sandnes
2	Svendson	Tove	Nissestien 67	Sandnes
3	Pettersen	Kari	Nissestien 67	Sandnes
4	Nilsen	Johan	Nissestien 67	Sandnes
5	Tjessem	Jakob	Nissestien 67	Sandnes

?

2. Работа с SQL
2.2. DML

The SQL DELETE Statement:

The DELETE statement is used to delete rows in a table.

DELETE FROM table_name WHERE some_column=some_value;

Note: The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

2. Работа с SQL
2.2. DML

DELETE example #1:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

Now we want to delete the person "Tjessem, Jakob" :

DELETE FROM Persons WHERE LastName='Tjessem' AND FirstName='Jakob';

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

2. Работа с SQL
2.2. DML

DELETE example #2:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

Now we want to delete all rows from "Persons" table:

DELETE FROM Persons;

As result-set all the data from Persons table will be deleted.

Note: Be very careful when deleting records. You cannot undo this statement!

?

2. Работа с SQL
2.2. DML

SQL JOINS:

The JOIN keyword is used in an SQL statement to query data from two or more tables, based on a relationship between certain columns in these tables.

Tables in a database are often related to each other with keys.

The purpose is to bind data together, across tables, without repeating all of the data in every table.

2. Работа с SQL
2.2. DML

SQL JOINS:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

Note: "P_Id" is the primary key in the "Persons" table.

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Note: "O_Id" is the primary key in the "Orders" table, "P_Id" column refers to the persons in the "Persons" table without using their names. Relationship between the two tables above is the "P_Id" column.

2. Работа с SQL
2.2. DML

SQL JOINS:

Before we continue with examples, we will list the types of JOIN you can use, and the differences between them.

- **JOIN:** Return rows when there is at least one match in both tables
- **LEFT JOIN:** Return all rows from the left table, even if there are no matches in the right table
- **RIGHT JOIN:** Return all rows from the right table, even if there are no matches in the left table

2. Работа с SQL
2.2. DML

SQL JOINS: INNER JOIN

The **INNER JOIN** keyword selects all rows from both tables as long as there is a match between the columns in both tables:

```
SELECT column_name(s) FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name;
```

PS: INNER JOIN is the same as JOIN

2. Работа с SQL
2.2. DML

SQL JOINS: INNER JOIN

Now we want to list all the persons with any orders:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id ORDER BY Persons.LastName;
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678

If there are rows in "Persons" that do not have matches in "Orders", those rows will NOT be listed.

?

2. Работа с SQL
2.2. DML

SQL JOINS: LEFT JOIN

The **LEFT JOIN** keyword returns all rows from the left table (table_name1), even if there are no matches in the right table (table_name2):

```
SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name;
```

2. Работа с SQL
2.2. DML

SQL JOINS: LEFT JOIN

Now we want to list all the persons and their orders - if any:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons LEFT JOIN Orders
ON Persons.P_Id=Orders.P_Id ORDER BY Persons.LastName;
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
Svendson	Tove	

The LEFT JOIN keyword returns all the rows from the left table (Persons), even if there are no matches in the right table (Orders).

?

2. Работа с SQL
2.2. DML

SQL JOINS: RIGHT JOIN

The RIGHT JOIN keyword returns all the rows from the right table (table_name2), even if there are no matches in the left table (table_name1):

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name;
```

2. Работа с SQL
2.2. DML

SQL JOINS: RIGHT JOIN

Now we want to list orders with containing persons - if any:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons RIGHT JOIN Orders
ON Persons.P_Id=Orders.P_Id ORDER BY Persons.LastName;
```

The result-set will look like this:

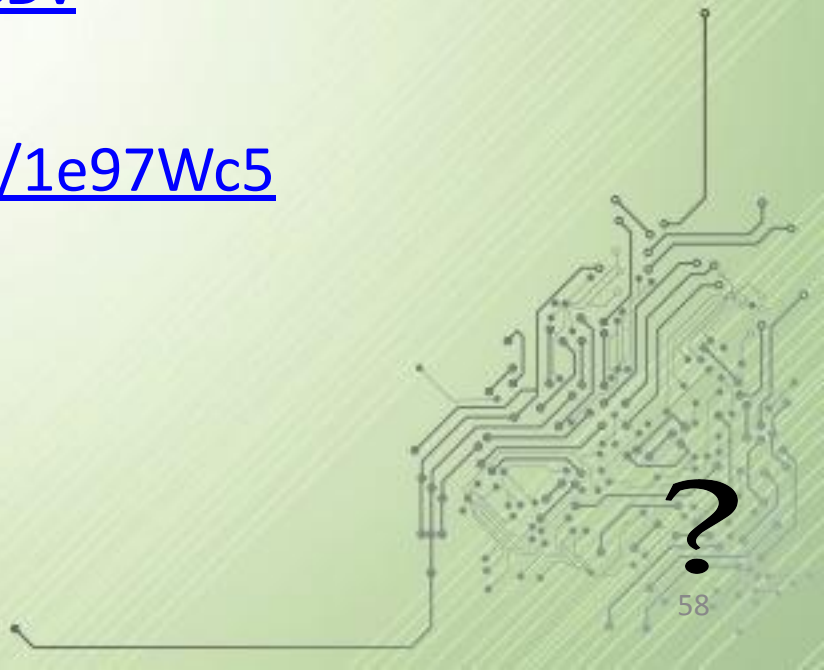
LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
		34764

The RIGHT JOIN keyword returns all the rows from the right table (Orders), even if there are no matches in the left table (Persons).

2. Работа с SQL

What to read and what should you know about definitions related to SQL:

1. Data types http://bit.ly/1l20Lj9
2. Primary Key http://bit.ly/1ixPFaP
3. Foreign Key http://bit.ly/Ux5CDv
4. Index http://bit.ly/19FI00c
5. Stored procedure http://bit.ly/1e97Wc5
6. Nested queries
7. Work with strings in SQL



Домашнее задание:

Создать таблицы:

CountryInfo_user*	PeopleInfo_user*
city_id (PK, int)	id (PK, int)
CountryName varchar(50);	Name varchar(50);
CityName varchar(50);	Surname varchar(50);
Population int;	city_id int;
isCapital (BIT)	isOccupied (BIT)

Заполнить таблицы данными (50 записей в PeopleInfo_user*, 20 записей в CountryInfo_user*)

http://bit.ly/TlwLIA - для быстрой генерации данных

Домашнее задание:

Написать запросы:

- Вывести имена и фамилии всех безработных из столиц Великобритании и Испании
- Посчитать количество работающих в Украине и Польше, проживающих в городах миллионниках и не являющихся столицами

Запросы сохранить в отдельных файлах my_queries_item_[description].sql.

Также необходимо зайти в свою папку (“User*”) и скопировать туда сохраненные запросы.

Нажать на вашей папке (“User*”) правой кнопкой и выполнить операцию SVN commit.