SQL Overview

SQL - це стандартна мова для доступу до баз даних та їхнього управління.

Що таке SQL?

SQL розшифровується як Structured Query Language (мова структурованих запитів).
SQL дозволяє вам отримувати доступ до баз даних та здійснювати над ними операції.
SQL став стандартом Інституту національних стандартів США (ANSI) у 1986 році і Міжнародної організації зі стандартизації (ISO) у 1987 році.

Що може робити SQL?

- SQL може виконувати запити до бази даних.
- SQL може отримувати дані з бази даних.
- SQL може додавати записи до бази даних.
- SQL може оновлювати записи у базі даних.
- SQL може видаляти записи з бази даних.
- SQL може створювати нові бази даних.
- SQL може створювати нові таблиці в базі даних.
- SQL може створювати збережені процедури в базі даних.
- SQL може створювати перегляди в базі даних.
- SQL може встановлювати права доступу до таблиць, процедур і переглядів.
- SQL є стандартом - АЛЕ Незважаючи на те, що SQL є стандартом ANSI/ISO, існують різні версії цієї мови.

Проте, щоб відповідати стандарту ANSI, всі вони підтримують принаймні основні команди (такі як SELECT, UPDATE, DELETE, INSERT, WHERE) подібним чином.

Примітка: Більшість програм для роботи з базами даних SQL також мають власні розширення, додатково до стандарту SQL!

Використання SQL на вашому веб-сайті
Для створення веб-сайту, який відображає дані з бази даних, вам знадобиться:

Програма для роботи з реляційною системою управління базами даних (наприклад, MS Access, SQL Server, MySQL).
Використання мови сценаріїв на стороні сервера, таких як PHP або ASP.
Використання SQL для отримання необхідних даних.
Використання HTML/CSS для стилюзації сторінки.

СКБД (СУБД)

СКБД означає Система керування базами даних з реляційною моделлю.

СКБД є основою для SQL та всіх сучасних систем управління базами даних, таких як MS SQL Server, IBM DB2, Oracle, MySQL і Microsoft Access.

Дані в СКБД зберігаються в об'єктах баз даних, які називаються таблицями. Таблиця - це колекція пов'язаних записів даних, яка складається з стовпців та рядків.

Придивіться до таблиці "Клієнти":

Приклад

```
SELECT * FROM Клієнти;
```

Кожна таблиця розбивається на менші сутності, які називаються полями. Поля в таблиці "Клієнти" включають CustomerID, CustomerName, ContactName, Address, City, PostalCode та Country. Поле - це стовпець у таблиці, призначений для збереження конкретної інформації про кожен запис у таблиці.

Запис, також називається рядком, це кожен окремий запис, що існує у таблиці. Наприклад, в таблиці "Клієнти" вище є 91 запис.

Стовпець - це вертикальна сутність у таблиці, яка містить всю інформацію, пов'язану з конкретним полем у таблиці.

SQL-вирази

Більшість дій, які вам потрібно виконати в базі даних, виконуються за допомогою SQL-виразів.

SQL-вирази складаються з ключових слів, які легко зрозуміти.

Наступний SQL-вираз повертає всі записи з таблиці з назвою "Клієнти":

Приклад
Виберіть всі записи з таблиці "Клієнти":

```
SELECT * FROM Клієнти;
```

У цьому навчальному посібнику ми навчимо вас різним SQL-виразам.

Таблиці бази даних
База даних найчастіше містить одну або декілька таблиць. Кожна таблиця ідентифікується іменем (наприклад, "Клієнти" або "Замовлення") і містить записи (рядки) з даними.

У цьому навчальному посібнику ми будемо використовувати відому вибірку навчальної бази даних "Північний вітер" (включено в MS Access та MS SQL Server).

Нижче наведено вибірку з таблиці "Клієнти", використаної в прикладах:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

У таблиці вище п'ять записів (по одному для кожного клієнта) та сім стовпців (CustomerID, CustomerName, ContactName, Address, City, PostalCode та Country).

Запам'ятайте, що...
Ключові слова SQL НЕ реагують на регістр: select те ж саме, що і SELECT.
У цьому навчальному посібнику ми писатимемо всі ключові слова SQL великими літерами.

Крапка з комою після SQL-виразів?
Деякі системи управління базами даних вимагають крапку з комою в кінці кожного SQL-виразу.

Крапка з комою є стандартним способом розділення кожного SQL-виразу в системах баз даних, які дозволяють виконати більше одного SQL-виразу в одному виклику до сервера.

У цьому навчальному посібнику ми будемо використовувати крапку з комою в кінці кожного SQL-виразу.

Деякі з найважливіших команд SQL
SELECT - витягує дані з бази даних
UPDATE - оновлює дані в базі даних
DELETE - видаляє дані з бази даних
INSERT INTO - додає нові дані в базу даних
CREATE DATABASE - створює нову базу даних
ALTER DATABASE - змінює базу даних
CREATE TABLE - створює нову таблицю
ALTER TABLE - змінює таблицю
DROP TABLE - видаляє таблицю
CREATE INDEX - створює індекс (ключ пошуку)
DROP INDEX - видаляє індекс

SQL-вираз SELECT

Вираз SELECT використовується для вибору даних з бази даних.

Приклад Отримайте свій власний SQL-сервер
Повернути дані з таблиці "Клієнти":

```
SELECT CustomerName, City FROM Клієнти;
```

Синтаксис

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

Тут column1, column2, ... - це назви полів таблиці, з якої ви хочете вибрати дані.

table_name представляє назву таблиці, з якої ви хочете вибрати дані.

Демонстраційна база даних
Нижче наведено вибірку з таблиці "Клієнти", використовуваної в прикладах:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Вибір усіх стовпців
Якщо ви хочете отримати всі стовпці без вказування кожного окремого імені стовпця, ви можете використовувати синтаксис SELECT *:

Приклад
Повернути всі стовпці з таблиці "Клієнти":

```
SELECT * FROM Клієнти;
```

SQL-вираз WHERE

У виразі WHERE використовується для фільтрації записів.

Він використовується для вибору лише тих записів, які задовольняють вказану умову.

Приклад Отримайте свій власний SQL-сервер
Вибрати всіх клієнтів з Мексики:

```
SELECT * FROM Клієнти
WHERE Країна='Mexico';
```

Синтаксис

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

Примітка: Вираз WHERE не використовується тільки в SELECT-операціях, його також можна використовувати в UPDATE, DELETE тощо!

Демонстраційна база даних
Нижче наведено вибірку з таблиці "Клієнти", використаної в прикладах:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country

1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Текстові поля проти числових полів
SQL вимагає використання одинарних лапок навколо текстових значень (більшість систем баз даних також дозволяють подвійні лапки).

Проте числові поля не повинні бути укладені в лапки:

Приклад

```
SELECT * FROM Клієнти
WHERE CustomerID=1;
```

Оператори в виразі WHERE
Ви можете використовувати інші оператори, ніж =, для фільтрації пошуку.

Приклад
Вибрати всіх клієнтів з CustomerID більше 80:

```
SELECT * FROM Клієнти
WHERE CustomerID > 80;
```

Ось деякі оператори, які можна використовувати в виразі WHERE:

Оператор	Опис
=	Рівно
>	Більше
<	Менше
>=	Більше або рівне
<=	Менше або рівне
<> або !=	Не рівно. Примітка: В деяких версіях SQL цей оператор може бути написаний як !=
BETWEEN	В межах певного діапазону
LIKE	Пошук за шаблоном
IN	Вказати кілька можливих значень для стовпця

SQL ORDER BY

Ключове слово ORDER BY використовується для сортування набору результатів в зростаючому або

спадаючому порядку.

Приклад Отримайте свій власний SQL-сервер
Відсортуйте продукти за ціною:

```
SELECT * FROM Продукти
ORDER BY Ціна;
```

Синтаксис

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Демонстраційна база даних
Нижче наведено вибірку з таблиці "Продукти", використаної в прикладах:

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35
Спадний порядок (DESC)					
Ключове слово ORDER BY за замовчуванням сортує записи в зростаючому порядку. Щоб відсортувати записи в спадаючому порядку, використовуйте ключове слово DESC.					

Приклад
Відсортуйте продукти від найдорожчого до найдешевшого:

```
SELECT * FROM Продукти
ORDER BY Ціна DESC;
```

Сортування за алфавітом
Для рядкових значень ключове слово ORDER BY сортує за алфавітом:

Приклад
Відсортуйте продукти за назвою ProductName:

```
SELECT * FROM Продукти
ORDER BY Назва;
```

Алфавітно у зворотньому порядку (DESC)
Щоб відсортувати таблицю у зворотньому алфавітному порядку, використовуйте ключове слово DESC:

Приклад
Відсортуйте продукти за назвою ProductName в зворотньому порядку:

```
SELECT * FROM Продукти
ORDER BY Назва DESC;
```

ORDER BY декілька стовпців
Наступний SQL-вираз вибирає всіх клієнтів з таблиці "Клієнти", відсортованих за стовпцем "Країна" та стовпцем "Ім'я клієнта". Це означає, що вони сортуються за Країною, але якщо деякі рядки мають однакову Країну, вони сортуються за Ім'ям клієнта:

Приклад

```
SELECT * FROM Клієнти
ORDER BY Країна, Ім_я_клієнта;
```

Використання ASC і DESC разом
Наступний SQL-вираз вибирає всіх клієнтів з таблиці "Клієнти", відсортованих за зростанням за стовпцем "Країна" та за спаданням за стовпцем "Ім'я клієнта":

Приклад

```
SELECT * FROM Клієнти
ORDER BY Країна ASC, Ім_я_клієнта DESC;
```

SQL-оператор AND

В умові WHERE може бути один або кілька операторів AND.

Оператор AND використовується для фільтрації записів на основі більше ніж одної умови, наприклад, якщо ви хочете повернути всіх клієнтів із Іспанії, імена яких починаються з літери 'G':

Приклад Отримайте свій власний SQL-сервер
Вибрати всіх клієнтів з Іспанії, імена яких починаються з літери 'G':

```
SELECT *
FROM Клієнти
WHERE Країна = 'Spain' AND CustomerName LIKE 'G%';
```

Синтаксис

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

```
WHERE condition1 AND condition2 AND condition3 ...;
```

AND проти OR
Оператор AND показує запис, якщо всі умови є TRUE.

Оператор OR показує запис, якщо хоча б одна з умов є TRUE.

Демонстраційна база даних
Нижче наведено вибірку з таблиці "Клієнти", використаної в прикладах:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Всі умови повинні бути TRUE
Наступний SQL-вираз вибирає всі поля з таблиці "Клієнти", де Країна - "Germany", Місто - "Berlin" і Поштовий код більше 12000:

```
SELECT * FROM Клієнти
WHERE Країна = 'Germany'
AND Місто = 'Berlin'
AND ПоштовийКод > 12000;
```

Комбінування AND і OR
Ви можете поєднувати оператори AND і OR.

Наступний SQL-вираз вибирає всіх клієнтів з Іспанії, імена яких починаються з "G" або "R".

Переконайтеся, що ви використовуєте дужки, щоб отримати правильний результат.

Приклад
Вибрати всіх іспанських клієнтів, імена яких починаються або на "G", або на "R":

```
SELECT * FROM Клієнти
WHERE Країна = 'Spain' AND (CustomerName LIKE 'G%' OR CustomerName LIKE 'R%');
```

Без дужок, оператор SELECT поверне всіх клієнтів з Іспанії, імена яких починаються з "G", плюс всіх клієнтів, імена яких починаються з "R", незалежно від значення країни:

```
SELECT * FROM Клієнти
WHERE Країна = 'Spain' AND CustomerName LIKE 'G%' OR CustomerName LIKE 'R%';
```

SQL-оператор OR

В умові WHERE може бути один або кілька операторів OR.

Оператор OR використовується для фільтрації записів на основі більше ніж одної умови, наприклад, якщо ви хочете повернути всіх клієнтів з Німеччини, а також тих, які з Іспанії:

Приклад Отримайте свій власний SQL-сервер
Вибрати всіх клієнтів з Німеччини або Іспанії:

```
SELECT *
FROM Клієнти
WHERE Країна = 'Germany' OR Країна = 'Spain';
```

Синтаксис

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

OR проти AND
Оператор OR показує запис, якщо хоча б одна з умов є TRUE.

Оператор AND показує запис, якщо всі умови є TRUE.

Демонстраційна база даних
Нижче наведено вибірку з таблиці "Клієнти", використаної в прикладах:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Принаймні одна умова повинна бути TRUE
Наступний SQL-вираз вибирає всі поля з таблиці "Клієнти", де або Місто - "Berlin", або Ім'я клієнта починається з літери "G", або Країна - "Norway":

```
SELECT * FROM Клієнти
WHERE Місто = 'Berlin' OR ИмяКлиента LIKE 'G%' OR Країна = 'Norway';
```

Комбінування AND і OR
Ви можете поєднувати оператори AND і OR.

Наступний SQL-вираз вибирає всіх клієнтів з Іспанії, імена яких починаються з "G" або "R".

Переконайтеся, що ви використовуєте дужки, щоб отримати правильний результат.

```
SELECT * FROM Клієнти
WHERE Країна = 'Spain' AND (ИмяКлиента LIKE 'G%' OR ИмяКлиента LIKE 'R%');
```

Без дужок, оператор SELECT поверне всіх клієнтів з Іспанії, імена яких починаються з "G", плюс всіх клієнтів, імена яких починаються з "R", незалежно від значення країни:

```
SELECT * FROM Клієнти
WHERE Країна = 'Spain' AND ИмяКлиента LIKE 'G%' OR ИмяКлиента LIKE 'R%';
```

SQL-оператор NOT

Оператор NOT використовується в поєднанні з іншими операторами для отримання протилежного, також називається негативного, результату.

У вибірковому операторі нижче ми хочемо повернути всіх клієнтів, які НЕ є з Іспанії:

Приклад Отримайте свій власний SQL-сервер
Вибрати тільки тих клієнтів, які НЕ є з Іспанії:

```
SELECT * FROM Клієнти
WHERE NOT Країна = 'Spain';
```

У вищезазначеному прикладі оператор NOT використовується в поєднанні з оператором =, але його можна використовувати в поєднанні з іншими операторами порівняння та/або логічними операторами. Див. приклади нижче.

Синтаксис

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Демонстраційна база даних
Нижче наведено вибірку з таблиці "Клієнти", використаної в прикладах:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

NOT LIKE
Приклад
Вибрати клієнтів, імена яких не починаються з літери 'A':

```
SELECT * FROM Клієнти
WHERE ИмяКлиента NOT LIKE 'A%';
```

NOT BETWEEN
Приклад
Вибрати клієнтів з CustomerID, які не знаходяться в проміжку від 10 до 60:

```
SELECT * FROM Клієнти
WHERE CustomerID NOT BETWEEN 10 AND 60;
```

NOT IN
Приклад
Вибрати клієнтів, які не з Парижу чи Лондона:

```
SELECT * FROM Клієнти
WHERE City NOT IN ('Paris', 'London');
```

NOT Greater Than
Приклад
Вибрати клієнтів з CustomerId, які не більше 50:

```
SELECT * FROM Клієнти
WHERE NOT CustomerID > 50;
```

Примітка: Існує оператор not-greater-then: !>, який дав би вам той же результат.

NOT Less Than
Приклад
Вибрати клієнтів з CustomerID, які не менше 50:

```
SELECT * FROM Клієнти
WHERE NOT CustomerId < 50;
```

Примітка: Існує оператор not-less-then: !<, який дав би вам той же результат.

SQL-оператор INSERT INTO

Оператор INSERT INTO використовується для вставки нових записів у таблицю.

Синтаксис INSERT INTO
Можливо написати оператор INSERT INTO двома способами:

1. Вказати як і назви колонок, так і значення для вставки:

```
INSERT INTO ім'я_таблиці (колонка1, колонка2, колонка3, ...)
VALUES (значення1, значення2, значення3, ...);
```

1. Якщо ви додаєте значення для всіх колонок таблиці, вам не потрібно вказувати назви колонок у SQL-запиті. Проте переконайтеся, що порядок значень співпадає з порядком колонок у таблиці. У цьому випадку синтаксис INSERT INTO буде наступним:

```
INSERT INTO ім'я_таблиці
VALUES (значення1, значення2, значення3, ...);
```

Демонстраційна база даних
Нижче наведено вибірку з таблиці "Клієнти", використованої в прикладах:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

Вставка прикладу
Наступний SQL-запит вставляє новий запис у таблицю "Клієнти":

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');
```

Після вставки вибірка з таблиці "Клієнти" виглядатиме наступним чином:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

Чи помітили ви, що ми не вставили жодного числа в поле CustomerID?
Колонка CustomerID є автоінкрементною, і вона буде генеруватися автоматично при вставці нового запису в таблицю.

Вставка даних лише у вказані колонки
Також можливо вставляти дані тільки в конкретні колонки.

Наступний SQL-запит вставляє новий запис, але вставляє дані лише в колонки "CustomerName", "City" і "Country" (CustomerID буде оновлено автоматично):

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

Після вставки вибірка з таблиці "Клієнти" виглядатиме наступним чином:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
...						

Вставка декількох рядків
Також можливо вставляти декілька рядків одночасно.

Щоб вставити декілька рядків даних, ми використовуємо той самий оператор INSERT INTO, але з декількома наборами значень:

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES
('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway'),
('Greasy Burger', 'Per Olsen', 'Gateveien 15', 'Sandnes', '4306', 'Norway'),
('Tasty Tee', 'Finn Egan', 'Streetroad 19B', 'Liverpool', 'L1 0AA', 'UK');
```

Переконайтеся, що ви розділяєте кожний набір значень комою ,.

Після вставки вибірка з таблиці "Клієнти" виглядатиме наступним чином:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway
93	Greasy Burger	Per Olsen	Gateveien 15	Sandnes	4306	Norway
94	Tasty Tee	Finn Egan	Streetroad 19B	Liverpool	L1 0AA	UK

SQL З'єднання (JOIN)

Команда JOIN використовується для комбінування рядків з двох чи більше таблиць на основі пов'язаного стовпця між ними.

Розгляньте вибірку з таблиці "Замовлення":

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Далі розгляньте вибірку з таблиці "Клієнти":

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Зверніть увагу, що стовпець "CustomerID" в таблиці "Замовлення" посилається на "CustomerID" в таблиці "Клієнти". Відношення між цими двома таблицями полягає в стовпці "CustomerID".

Тоді ми можемо створити такий SQL-запит (який містить INNER JOIN), що вибирає записи, які мають спільні значення в обох таблицях:

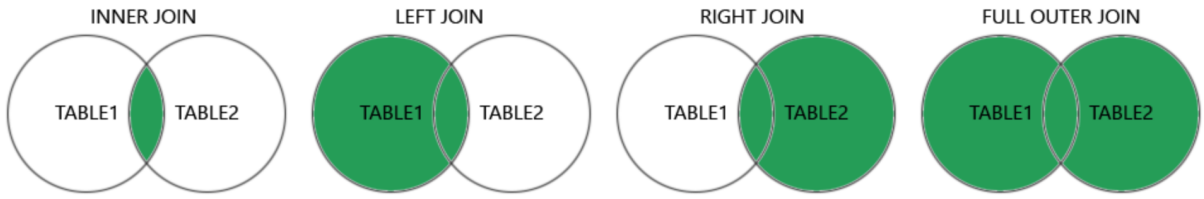
```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Цей запит поверне результат, подібний до наступного:

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

Різновиди SQL JOIN
Ось різновиди JOIN в SQL:

1. (INNER) JOIN: Повертає записи, що мають спільні значення в обох таблицях.
2. LEFT (OUTER) JOIN: Повертає всі записи з лівої таблиці і співпадаючі записи з правої таблиці.
3. RIGHT (OUTER) JOIN: Повертає всі записи з правої таблиці і співпадаючі записи з лівої таблиці.
4. FULL (OUTER) JOIN: Повертає всі записи, коли є відповідність в обох таблицях (спільні та невідповідні записи).



INNER JOIN

Ключове слово INNER JOIN вибирає записи, у яких є відповідні значення у обох таблицях.

Давайте розглянемо вибірку з таблиці "Продукти":

ProductID	ProductName	CategoryID	Price
1	Chais	1	18
2	Chang	1	19
3	Aniseed Syrup	2	10

Також вибірку з таблиці "Категорії":

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads

Ми з'єднаємо таблицю "Продукти" з таблицею "Категорії", використовуючи поле CategoryID в обох таблицях:

```
SELECT ProductID, ProductName, CategoryName
FROM Products
INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;
```

Синтаксис

```
SELECT назва_стовпця(стовпців)
FROM таблиця1
INNER JOIN таблиця2
ON таблиця1.назва_стовпця = таблиця2.назва_стовпця;
```

Примітка: Ключове слово INNER JOIN повертає лише рядки, які мають відповідність в обох таблицях. Це означає, що якщо у вас є продукт з відсутнім CategoryID або з CategoryID, який не присутній в таблиці "Категорії", цей запис не буде включено в результат.

Назви Стовпців
Добра практика - включати назву таблиці, коли вказуєте стовпці в SQL-запиті.

Приклад:

```
SELECT Products.ProductID, Products.ProductName, Categories.CategoryName
FROM Products
INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;
```

JOIN чи INNER JOIN
JOIN та INNER JOIN повертають однаковий результат.

INNER є типовим типом з'єднання для JOIN, тобто, коли ви пишете JOIN, синтаксичний аналізатор фактично записує INNER JOIN.

Приклад:

```
SELECT Products.ProductID, Products.ProductName, Categories.CategoryName
FROM Products
JOIN Categories ON Products.CategoryID = Categories.CategoryID;
```

JOIN для Трьох Таблиць
Наступний SQL-запит вибирає всі замовлення з інформацією про клієнта та перевізника:

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

SQL LEFT JOIN

Ключове слово LEFT JOIN повертає всі записи з лівої таблиці (table1) та відповідні записи з правої таблиці (table2). Результат - 0 записів з правого боку, якщо відповідності немає.

Синтаксис LEFT JOIN

```
SELECT назва_стовпця(стовпців)
FROM таблиця1
LEFT JOIN таблиця2
ON таблиця1.назва_стовпця = таблиця2.назва_стовпця;
```

Примітка: В деяких базах даних LEFT JOIN називається LEFT OUTER JOIN.

Приклад SQL LEFT JOIN
Наступний SQL-запит вибирає всіх клієнтів і будь-які замовлення, які можуть у них бути:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Цей запит поверне список всіх клієнтів та замовлень, які вони можуть мати. Якщо клієнт не має жодного замовлення, відповідний стовпець OrderID буде містити значення NULL.