**Що таке SQL?**

Отже, переходимо до **SQL**.

**SQL** - проста мова програмування, яка має небагато команд і якій може навчитися будь-який бажаючий. Розшифровується як **Structured Query Language** - мова структурованих запитів, яка була розроблена для роботи з **БД**, а саме, щоб отримувати/добавляти/змінювати дані, мати можливість опрацьовувати великі масиви інформації та швидко отримувати структуровану та згруповану інформацію. Є багато варіантів мови **SQL**, але у них всіх основні команди майже однакові. Також існує і багато **СУБД**, але основними з них являються: **Microsoft Access, Microsoft SQL Server, MySQL, Oracle SQL, IBM DB2 SQL, PostgreSQL та Sybase Adaptive Server SQL**. Щоб працювати з**SQL**кодом, нам потрібна буде одна з вище перелічених **СУБД**. Для навчання ми будемо використовувати **СУБД Microsoft Access**.

**SQL** як і інші мови програмування має свої команди (оператори), за допомогою яких віддаються вказівки для вибірки даних. Щоб розглянути як працюють оператори **SQL**, ми будемо використовувати уявну **БД** з інформацією про реалізовану продукцію:



Найпершим та найголовнішим оператором в **SQL** являється **SELECT**. З його допомогою ми можемо відбирати необхідні нам поля з даними в таблиці.

## 1. Вибірка окремих полів.

**SELECT Product FROM Sumproduct**



Бачимо, що наш **SQL запит** відібрав колонку **Product** з таблиці **Sumproduct**.

## 2. Вибірка кількох полів.

Припустимо, нам необхідно вибрати назву та кількість реалізованого товару. Для цього просто перераховуємо необхідні поля через кому:

**SELECT Product, Quantity FROM Sumproduct**



## 3. Вибірка всіх стовпців.

Якщо ж нам необхідно отримати всю таблицю зі всіма полями, тоді просто ставимо знак зірочка (**\***):

**SELECT \* FROM Sumproduct**



Вітаю, Ви зробили перші **SQL запити**.

**PS.**Всі оператори в **SQL** нечутливі до регістру, тому ви можете їх писати як великими буквами, так і маленькими (як правило, їх прийнято писати великими буквами, щоб розрізняти від назв полів та таблиць). Назви же таблиць та полів є навпаки чутливими до регістру та мають писатися точно як в **БД**.

В майбутньому нам може знадобитися посортувати нашу вибірку - в алфавітному порядку для тексту чи по зростанню/спаданню - для цифрових значень. Для таких цілей в **SQL** є спеціальний оператор **ORDER BY**.

## 1. Сортування вибраних даних.

Давайте усю нашу таблицю посортуємо по сумі реалізації продукції, а саме по стовпцю **Amount**.

**SELECT \* FROM Sumproduct ORDER BY Amount**



Бачимо, що запит посортував записи по зростанню в полі **Amount**. Обов'язково потрібно дотримуватись послідовності розташування операторів, тобто оператор **ORDER BY** має йти в самому кінці запиту. В іншому випадку буде отримане повідомлення про помилку.

Також, особливістю оператора **ORDER BY** є те, що він може сортувати дані по полю, якого ми не вибирали у запиті, тобто достатньо щоб воно взагалі було в **БД**.

## 2. Сортування за кількома полями.

Тепер посортуємо наш приклад додатково за ще одним полем. Нехай це буде поле **City**, яке відображає місце реалізації продукції.

**SELECT \* FROM Sumproduct ORDER BY Amount, City**



Черговість сортування буде залежати від порядку розташування полів в запиті. Тобто, в нашому випадку спочатку дані будуть посортовані по колонці **Amount**, а потім по **City**.

## 3. Напрямок сортування.

Незважаючи на те, що по замовчуванню оператор ***ORDER BY*** сортує по зростанню, ми можемо також прописати сортування значень по спаданню. Для цього в кінці кожного поля проставляємо оператор **DESC** (що є скороченням від слова **DESCENDING).**

**SELECT \* FROM Sumproduct ORDER BY Amount DESC, City**



В даному прикладі, значення в полі **Amount** були посортовані по спаданню, а в полі **City** - по зростанню. Оператор ***DESC*** застосовується лише для одного стовпця, тому при потребі його потрібно прописувати після кожного поля, яке приймає участь у сортуванні.

В більшості випадків необхідно отримувати не всі записи, а лише ті, які відповідають певним критеріям. Тому для здійснення фільтрації вибірки в **SQL** є спеціальний оператор **WHERE**.

## 1. Просте фільтрування оператором WHERE.

Давайте з нашої таблиці, наприклад, відберемо записи, які стосуються лише певного товару. Для цього ми зазначимо додатковий параметр відбору, який фільтруватиме значення по колонці**Product**.

#### Приклад запиту для відбору текстових значень:

**SELECT \* FROM Sumproduct WHERE Product** = 'Bikes'



Як бачимо, умова відбору взята в одинарні лапки, що є обов'язковим при фільтруванні текстових значень. При фільтруванні числових значень лапки не потрібні.

#### Приклад запиту для відбору числових значень:

**SELECT \* FROM Sumproduct WHERE Amount** > 40000 **ORDER BY Amount**



В цьому прикладі ми відібрали записи, в яких виручка від реалізації становила більше **40 тис. $**та, додатково, всі записи посортували по зростанню по полю **Amount.**

В таблиці нижче, зазначено перелік умовних операторів, які підтримуються **SQL**:

|  |  |
| --- | --- |
| **Знак операції** | **Значення** |
| = | Дорівнює |
| <> | Не дорівнює |
| < | Менше |
| <= | Менше або рівне |
| > | Більше |
| >= | Більше або рівне |
| BETWEEN | Між двома значеннями |
| IS NULL | Відсутній запис |

## 2. Фільтрування по діапазону значень (****BETWEEN****).

Для відбору даних, які лежать в певному діапазоні, використовується оператор **BETWEEN**. В наступному запиті будуть відібрані усі значення, які лежать в межах від **1000$** до **2000$** включно, в полі **Amount**.

**SELECT \* FROM Sumproduct WHERE Amount** **BETWEEN** 1000 **AND** 2000



Черговість сортування буде залежати від порядку розташування полів в запиті. Тобто, в нашому випадку спочатку дані будуть посортовані по колонці **Amount**, а потім по **City**.

## 3. Вибірка порожніх записів (****IS NULL****).

В **SQL** існує спеціальний оператор для вибірки порожніх записів (називаєьбся **NULL**). Порожнім записом вважається будь-яка комірка в таблиці, в яку не введено жодного символу. Якщо в комірку введено **0** або **пробіл**, то вважається, що поле заповнене.

**SELECT \* FROM Sumproduct WHERE Amount IS NULL**



В прикладі вище, ми навмисне видалили два значення в полі **Amount**, щоб продемонструвати роботу оператора **NULL**.

## 4. Розширене фільтрування (****AND, OR****).

Мова **SQL** не обмежується фільтруванням по одній умові, для власних цілей ви можете викоритсовувати досить складні конструкції для вибірки даних одночасно по багатьом критеріям. Для цього в **SQL** є додаткові оператори, які розширюють можливості оператора **WHERE**. Такими операторами являються: **AND, OR, IN, NOT**. Наведемо кілька прикладів роботи даних операторів.

**SELECT \* FROM Sumproduct WHERE Amount** > 40000 **AND** **City** = 'Toronto'



**SELECT \* FROM Sumproduct WHERE Month**= 'April' **OR Month**= 'March'



Давайте об'єднаємо оператори **AND** та **OR**. Для цього зробимо вибірку велосипедів (***Bikes***) та ковзанів (***Skates***), які були продані в березні (***March***).

**SELECT \* FROM Sumproduct WHERE Product**= 'Bikes' **OR Product**= 'Skates' **AND** **Month**= 'March'



Бачимо, що в нашу вибірку попало за багато значень (крім березня (**March**), також січень (**January**), лютий (**February**) та квітень (**April**)). В чому ж причина? А в тому, що **SQL** має пріоритети виконання команд. Тобто оператор **AND** має вищий пріоритет, ніж оператор **OR**, тому спочатку було відібрано записи з ковзанами, що продані в березні, а потім усі записи, що стосувалися велосипедів.

Отже, щоб отримати правильну вибірку, нам потрібно змінити пріоритети виконання команд. Для цього використаємо **дужки**, як в математиці. Тоді, спочатку будуть опрацьовані оператори в дужках, а потім - всі решта.

**SELECT \* FROM Sumproduct WHERE (Product**= 'Bikes' **OR Product**= 'Skates'**)** **AND** **Month**= 'March'



## 5. Розширене фільтрування (****оператор IN****).

**SELECT \* FROM Sumproduct WHERE ID IN** **(4, 12, 58, 67)**



Оператор **IN** виконує ту ж саму функцію, що і **OR**, проте має ряд переваг:

* При роботі з довгими списками, речення з **IN** легше читати;
* Використовується менша кількість операторів, що пришвидшує обробку запиту;
* Найважливіша перевага **IN** в тому, що в його конструкції можна використовувати додаткову конструкцію **SELECT**, що

відкриває великі можливості для створення складних підзапитів.

## 6. Розширене фільтрування (****оператор NOT****).

**SELECT \* FROM Sumproduct WHERE NOT City IN** **('Toronto', 'Montreal')**



Ключове слово **NOT** дозволяє забрати непотрібні значення із вибірки. Також його особливістю є те, що воно проставляється перед назвою стовпця, який бере участь у фільтруванні, а не після.