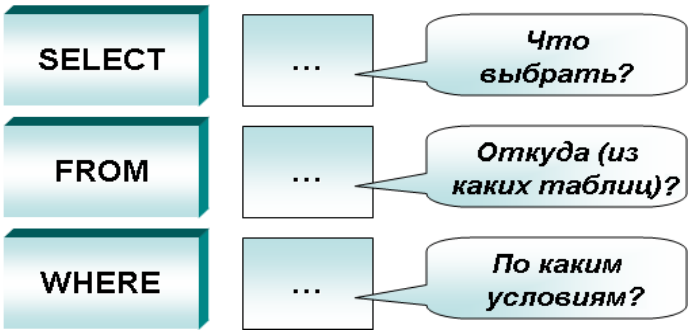
Лабораторная работа №1

Теория и примеры

1. **Структура запроса SELECT**



- **SELECT * FROM** Student;
- **SELECT** Name, City **FROM** Student **WHERE** age > 20;

  DISTINCT – отбирает уникальные значения

2. **Сортировка данных**

Предложение **ORDER BY** определяет порядок сортировки выходных данных. Если после имени поля записать служебное слово DESCENDING или сокращенный вариант DESC, то порядок сортировки по этому полю будет обратным.

- **SELECT** Name, City **FROM** Student **WHERE** age > 20 **ORDER BY** Name;
- **SELECT** Name, City **FROM** Student **WHERE** age > 20 **ORDER BY** Name DESC;

3. **Операторы в разделе WHERE**

3.1 Операторы фильтрации

Оператор	Описание
=	Равно
!= или <>	Не равно
>	Больше чем
<	Меньше чем
>=	Больше или равно
<=	Меньше или равно
BETWEEN	Между включенным диапазоном
LIKE	Поиск по шаблону
IN	Поиск данных по нескольким значениям, перечисленным через запятую

3.2 Расширенные фильтрации (AND, OR).

Язык **SQL** не ограничивается фильтрацией по одному условию, для собственных целей вы можете использовать достаточно сложные конструкции для выборки данных одновременно по многим критериям.

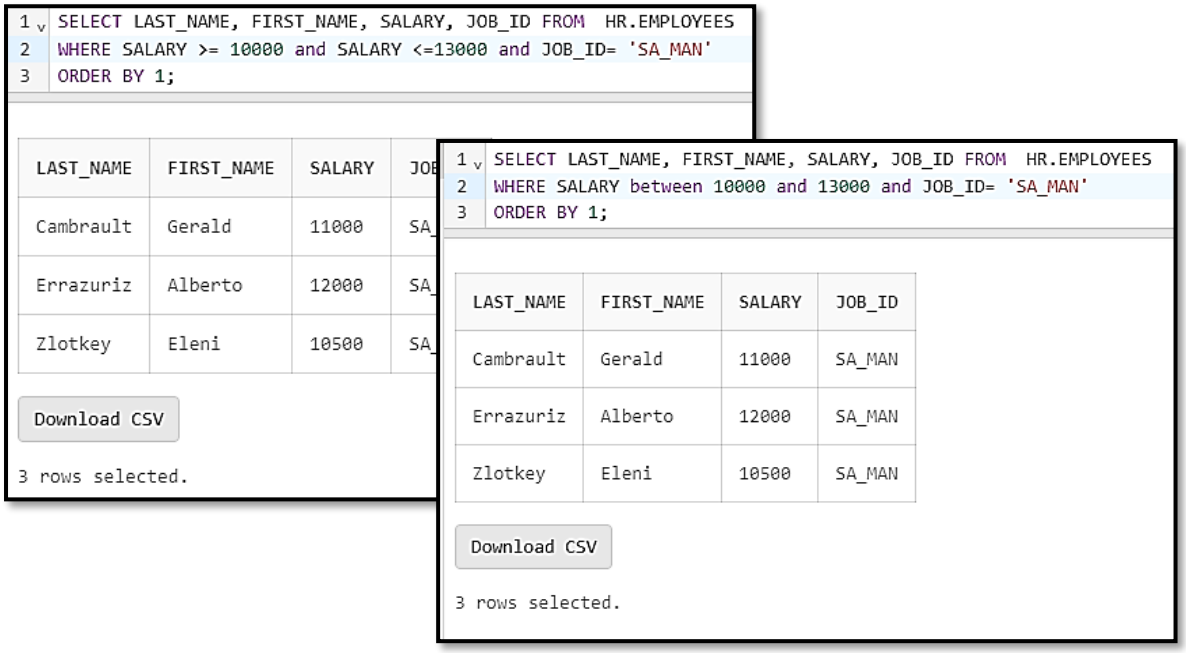
- **SELECT * FROM** product **WHERE** Amount > *40000* **AND** City = *'Toronto'*
- **SELECT** * **FROM** customers **WHERE** customer_name = 'IBM'
  OR customer_name = 'Hewlett Packard'
  OR customer_name = 'Microsoft';
- **SELECT * FROM product WHERE (**Product = *'Bikes'* **OR** Product = *'Skates'***) AND** Month= *'March'*

3.3 Фильтрация по диапазону значений (BETWEEN)

Для отбора данных, которые лежат в определенном диапазоне, используется оператор **BETWEEN (**включительно**).**

SELECT * FROM customers WHERE customer_id **BETWEEN** 4000 AND 4999;

SELECT * FROM customers WHERE customer_id >= 4000 **AND** customer_id <= 4999;



3.4 Оператор IN

- Оператор **IN** выполняет ту же функцию, что и **OR**, однако имеет ряд преимуществ:
- При работе с длинными списками, предложение с **IN** легче читать;
- Используется меньшее количество операторов, что ускоряет обработку запроса;

SELECT * **FROM** customers **WHERE** customer_name **IN** ('IBM', 'Hewlett Packard', 'Microsoft');

3.5 Символы подстановки и регулярные выражения (LIKE)

- Часто, для фильтрации данных, нам нужно будет осуществить выборку не по точному совпадению условия, а по приближенному значению. То есть когда, например, мы ищем товар, название которого соответствует определенному шаблону или содержит определенные символы или слова.
- Для таких целей в **SQL** существует оператор **LIKE**, который ищет приближенные значения. Для конструирования такого шаблона используются **метасимволы**
- **%** - Соответствует любой строке любой длины (в том числе нулевой длины)
- _ -Соответствует одному символу

- **SELECT** last_name **FROM** customers **WHERE** last_name **LIKE** 'Ap%';
- **SELECT** last_name **FROM** customers **WHERE** last_name **LIKE** '%e%';
- **SELECT** supplier_name **FROM** suppliers **WHERE** supplier_name **LIKE** 'Sm_th';
- **SELECT** * **FROM** suppliers **WHERE** account_number **LIKE** '92314_';

! LIKE не тоже самое «=»

- **SELECT salary FROM employees WHERE last_name LIKE 'R%';**

вернет зарплаты всех сотрудников с именем начинающимся на ‘R’

- **SELECT salary FROM employees WHERE last_name = 'R%';**

вернет зарплаты всех сотрудников с именем 'R%’

ESCAPE - экранирование символа

Используется для поиска значений с «%» или «_»

- **SELECT** * **FROM** suppliers **WHERE** supplier_name **LIKE** 'Water!%' ESCAPE '!';

  Этот запрос вернет всех suppliers, чье supplier_name имеют значение ‘Water%’.

- **SELECT** * **FROM** suppliers **WHERE** supplier_name **LIKE** 'H%!%' ESCAPE '!';
- **SELECT** last_name **FROM** employees **WHERE** last_name **LIKE** '%A_B%' ESCAPE '\';

3.6 SIMILAR TO с регулярными выражениями

- оператор **SIMILAR TO** возвращает истину или ложь в зависимости от того, соответствует ли его шаблон данной строке;
- похож на LIKE , за исключением того, что он интерпретирует шаблон, используя определение регулярного выражения в стандарте SQL;
- SIMILAR TO работает успешно, только если его шаблон соответствует всей строке;
- регулярные выражения SQL - это «смесь» нотации LIKE и общей (POSIX) нотации регулярных выражений.

  В SIMILAR TO используются _ и % в качестве подстановочных знаков,

  SIMILAR TO поддерживает эти метасимволы сопоставления с образцом, заимствованные из регулярных выражений POSIX:

  - | обозначает альтернативу (любую из двух альтернатив).
  - обозначает повторение предыдущего пункта ноль или более раз.
  - + обозначает повторение предыдущего элемента один или несколько раз.
  - ? обозначает повторение предыдущего элемента ноль или один раз.
  - { m } обозначает повторение предыдущего элемента ровно m раз.
  - { m ,} обозначает повторение предыдущего элемента m или более раз.
  - { m , n } означает повторение предыдущего пункта не менее m и не более n раз.
  - Круглые скобки () можно использовать для группировки элементов в один логический элемент.

- Выражение в квадратных скобках [...] определяет класс символов, как и в регулярных выражениях POSIX.

'abc'	**SIMILAR TO** 'abc'	true
'abc'	**SIMILAR TO** 'a'	false
'abc'	**SIMILAR TO** '%(b|d)%'	true
'abc'	**SIMILAR TO** '(b|c)%'	false
'-abc-'	**SIMILAR TO** '%\mabc\M%'	true
'xabcy'	**SIMILAR TO** '%\mabc\M%'	false

3.7 Выборка пустых записей (IS NULL).

В **SQL** существует специальный оператор для выборки пустых записей (называется **NULL**). Пустой записью считается любая ячейка в таблице, в которую не введены какие-либо символы. Если в ячейку введен **0** или **пробел**, то считается, что поле заполнено.

SELECT * **FROM** customers **WHERE** Last_Name **is NULL**;

3.8 Ключевое слово NOT

Ключевое слово **NOT** позволяет убрать ненужные значения из выборки. Также его особенностью является то, что оно проставляется перед названием столбца, участвующего в фильтровании, а не после.

- SELECT * FROM customers WHERE last_name IS **NOT NULL**;
- SELECT * FROM customers WHERE customer_name **NOT IN** ('IBM', 'Hewlett Packard', 'Microsoft');
- SELECT customer_name FROM customers WHERE customer_name **NOT LIKE 'S%';**
- SELECT * FROM customers WHERE customer_id **NOT BETWEEN** 4000 AND 4100;