***КОМАНДА SELECT***

Команда **SELECT** извлекает данные из столбцов одной или нескольких таблиц. Команда SELECT сам по себе является запросом. Если он используется как предложение внутри другого оператора, то он называется подзапросом. В операторе SELECT обязательно должно присутствовать предложение FROM. Остальные предложения не являются необходимыми.

Предложение SELECT может использоваться как:

* самостоятельная команда на получение и вывод строк таблицы, сформированной из столбцов и строк одной или нескольких таблиц (представлений);
* фраза выбора в командах CREAT VIEW, DECLARE CURSOR или INSERT;
* средство присвоения глобальным переменным значений из строк сформированной таблицы (INTO-фраза).

**Синтаксис команды:**

**SELECT** alias.столбец\_1, alias.столбец\_2 ….,

функция(столбец\_3), (столбец\_1+ столбец\_2) as выражение

**FROM** таблица\_1 alias, таблица\_2 alias, (запрос\_1) alias

**WHERE** условие\_1 [OR [NOT]| AND [NOT]] условие\_2

**GROUP BY** Групповой\_столбец\_1, групповой\_столбец2

**HAVING** условие\_по\_групповому\_столбцу

**ORDER BY** столбец\_1 **ASC**|**DESC**

После указания перечня выбираемых столбцов и операций над ними, указывается служебное слово FROM – оно определяет ресурс данным, т.е. откуда будут выбираться данные, в качестве ресурса данных могут выступать таблицы, представления, запросы.

Таблице или столбцу можно присвоить альтернативное имя (алиас - **alias**), действие которого будет действительно только в пределах оператора, в котором оно определено. Если альтернативное имя стоит после имени столбца в списке оператора SELECT, то оно будет использоваться вместо настоящего имени как заголовок для данного столбца. Использование имен столбцов и алиаса приведен в примере ниже.

|  |
| --- |
| select emp.first\_name "Имя", emp.last\_name "Фамилия"  from HR.employees emp |

Альтернативное имя таблицы можно использовать при соединении таблицы с самой собой в соотносящемся запросе. При использовании таблиц с одинаковыми именами полей требуется указание имени таблицы перед именем колонки таблицы.

Для выбора данных по условию применяют предложение **WHERE**. После его указания определяется перечень условий, разделяемые между собой логикой – элементы AND, NOT AND, OR, NOT OR. Условием в языке SQL называется сочетание одного или нескольких выражений и логических операторов, вырабатывающих значение TRUE (истина) или FALSE (ложь). Каждое условие можно представить в виде операторов сравнения (см. таблица).

|  |  |
| --- | --- |
| **Оператор** | **Значение/действие в SQL** |
| = | равно |
| != или <> | не равно |
| >= | больше или равно |
| <= | меньше или равно |
| IN | равен любому элементу в |
| NOT IN | не равен любому элементу в |
| ANY | Сравнивает с любым из значений в списке. Употребляется после =, !=, >, <, <=, >=. |
| ALL | Сравнивает с каждым значением в списке. Употребляется после =, !=, >, <, <=, >=. |
| BETWEEN | больше или равно значения\_1 и меньше или равно значение\_2 |
| NOT | не больше и не равно |
| EXISTS | Истина, если подзапрос извлекает хотя бы одну строку |
| IS NULL | Истина, значение есть NULL. Проверки типа x=NULL – являются неправильными. |
| IS NOT NULL | Истина, если значение не пустое |

Рассмотрим примеры с использованием операторов сравнения и предиката WHERE. В первом примере мы выбираем все столбцы таблицы «employees» и определяем тех работников, у которых оклад равен 2500. Для вывода всех столбцов в запросе используется символ «\*».

|  |
| --- |
| select \*  from HR.employees emp  where emp.salary=2500 |

Найдем всех работников, у которых значение поле «salary» больше 2000 и не равно 2500.

|  |
| --- |
| select \*  from HR.employees emp  where emp.salary != 2500 and emp.salary > 2000 |

Найдем всех работников, у которых оклад от 2000 и до 3000 включительно.

|  |
| --- |
| select \*  from HR.employees emp  where emp.salary between 2000 and 3000 |

Найдем всех работников, у которых нет начальника.

|  |
| --- |
| select \*  from HR.employees emp  where emp.manager\_id is null |

Найдем все работников, работающих в департаментах с кодами 20, 40, 50

|  |
| --- |
| select \*  from HR.employees emp  where emp.department\_id in (20,40,50) |

Для группировки результирующего запроса по столбцам используется выражение **GROUP BY,** после которого указывается список столбцов, по которым будет группироваться таблица, для вычисляемых столбцов применяются групповые функции SUM, AVG, MIN, MAX, COUNT при определении списка выбираемых столбцов.

Определим количество работников в каждом департаменте.

|  |
| --- |
| select emp.department\_id,count(emp.employee\_id)  from HR.employees emp  group by emp.department\_id |

Для получения данных по условию для сгруппированной таблицы применяется элемент **HAVING,** после чего указывается условие для сгруппированной таблицы. Условие определяется аналогично условиям **WHERE**. В нижеприведенном примере, мы оставили те департаменты, у которых количество работников больше пяти.

|  |
| --- |
| select emp.department\_id,count(emp.employee\_id)  from HR.employees emp  group by emp.department\_id  having count(emp.employee\_id) > 5 |

Для сортировки полученных данных в запросе применяется **ORDER BY.**  После указания выражения перечисляются столбцы, покоторым будет сортировка, и тип сортировки по возрастанию (**ASC**) или убыванию (**DESC**). По умолчанию сортировка по возрастанию.

Отсортируем работников по номерам департаментов по убыванию:

|  |
| --- |
| select \*  from HR.employees emp  order by emp.department\_id desc |

Отсортируем работников по фамилии и имени департаментов по убыванию:

|  |
| --- |
| select \*  from HR.employees emp  order by emp.last\_name, emp.first\_name asc |

Для выбора всех или уникальных записей применяют выражения [**DISTINCT** | **UNIQUE** | **ALL**] после слово SELECT, затем перечисляются столбцы, по которым будет отслеживаться уникальность данных.

**Синтаксис команды:**

**SELECT** [**DISTINCT | UNIQUE | ALL** ] alias.столбец\_1, alias.столбец\_2

**FROM** таблица\_1 alias

Выражения DISTINCT или UNIQUE позволяют осуществить выбор только уникальных записей. Выражение ALL – выбор всех записей из таблицы.

Для получения объединения, пересечения и разности запросов используется служебные слова **UNION**, **UNION ALL**, **MINUS**, **INTERSECT**:

**Синтаксис команды:**

SELECT запрос1

[UNION | UNION ALL | MINUS | INTERSECT]

SELECT запрос2

UNION – объединяет не дублирующиеся данные нескольких запросов

UNION ALL – объединяет все данные

MINUS – производит вычитание данных из запроса1 запрос2

INTERSECT – показывает одинаковые данные имеющиеся в запросах

Оператор **UNION** объединяет выходные строки каждого из запросов в один результирующий набор. Если определен параметр **ALL**, то сохраняются все дубликаты выходных строк, в противном случае в результирующем наборе остаются только уникальные строки. Заметим, что можно связывать вместе любое число запросов. Кроме того, с помощью скобок можно менять порядок объединения.

При этом должны выполняться следующие условия:

* Количество выходных столбцов каждого из запросов должно быть одинаковым.
* Выходные столбцы каждого из запросов должны быть сравнимыми между собой (в порядке их следования) по типам данных.
* В результирующем наборе используются имена столбцов, заданные в первом запросе.
* Предложение ORDER BY применяется к результату соединения, поэтому оно может быть указано только в конце составного запроса.

Для выполнения операций **пересечения** и **разности** запросов. Этими предложениями являются **INTERSECT** (пересечение) и **MINUS** (разность), которые работают аналогично предложению **UNION**. В результирующий набор попадают только те строки, которые присутствуют в обоих запросах (INTERSECT) или только те строки первого запроса, которые отсутствуют во втором (MINUS).

Рассмотрим примеры. В первом примере получаем объединение запросов с дубликатами строк.

|  |
| --- |
| SELECT kod, fio FROM s\_fiz\_lic3  UNION ALL  SELECT kod, fio FROM s\_fiz\_lic3; |

Объединяем запросы с устранением дубликатов.

|  |
| --- |
| SELECT kod, fio FROM s\_fiz\_lic3  UNION  SELECT kod, fio FROM s\_fiz\_lic3; |

***Немного о соединения таблиц***

Соединения двух и более таблиц могут выполняться в одном запросе с указанием условий соединения.

**Пример:** Выбрать фамилии сотрудников, номера и названия отделов,

в которых они работают.

**Решение 1.**

|  |
| --- |
| select emp.last\_name, emp.department\_id, dept.department\_name  from HR.employees emp,  HR.departments dept  where dept.department\_id = emp.department\_id; |

**Решение 2.**

|  |
| --- |
| select emp.last\_name, emp.department\_id, dept.department\_name  from HR.employees emp  join departments dept on emp.department\_id = dept.department\_id; |

Соединяем те строки таблиц emp и dept, которые имеют одинаковые значения столбца deptno. Поскольку deptno имеется в обеих таблицах, в условии соединения следует уточнить название столбца названием его таблицы, например, emp.deptno. В списке фразы SELECT только для одного столбца необходимо указание таблицы emp.deptno или dept.deptno. Если этого не сделать, появится сообщение об ошибке.

Остальные столбцы ename и dname имеются только в одной таблице. При желании префиксы можно поставить и перед их именами.

Замечание: Различайте связи и соединения таблиц.

Связи работают во время манипулирования данными, обеспечивая выполнение ограничений ссылочной целостности. Соединения создаются в запросах. Их смысл целиком на совести программиста создающего запрос.

***Схема таблиц HR***

******

***Задание для аудиторной и самостоятельной работы:***

1. Получить данные по департаментам: для каждого департамента определить минимальную и максимальную зарплату, определить дату устройства на работу первого и последнего сотрудника, а также общее количество сотрудников в департаменте. Сортировать по количеству сотрудников (по убыванию)
2. Для каждого года вывести количество сотрудников нанятых в этом году. Сортировать по количеству сотрудников.
3. Найти управляющих (а именно их manager\_id), у которых количество подчиненных больше 5 и сумма всех зарплат подчиненных больше 50000.
4. Получить максимальную зарплату среди всех средних зарплат по департаменту.