Запрос Select языка SQL

Запрос Select

Общий синтаксис запроса **select** выглядит следующим образом:

```
select <col1>[,<col2>…] from <tab1> <alias1>
       [[left|right] join <tab2> <alias2>
              on <join cond>]
       [where <where cond>]
       [group by <col1>[,<col2>…][having <having cond>]]
       [order by <col1> [descending],
                                  [,<col2>[descending]…]]
```

Значения параметров следующие:

<col1>… <coln>	имена столбцов (полей таблицы) включая вызовы функций и преобразования типов *cast()*
<tab1>… <tabn>	имена таблиц и обзоров (view) из которых производится выборка данных
<alias1>… <aliasn>	алиасы (псевдонимы) задаваемые пользователем для удобства написания запроса
<join cond>	условие присоединения таблицы
<where cond>	условие отбора записей
<having cond>	условие группировки записей

Запрос Select

Ключевые конструкции запроса:

from	указывает главный источник запроса (таблица или обзор). Запрос будет выбирать столбцы из этой таблицы
join	позволяет присоединять к основному источнику запроса вторичные источники (таблицы или обзоры).
left join	используется для стандартного присоединения мастер-таблицы (связь 1:M)
where	задает условия отбора записей
group by	позволяет задать условие группировки записей, как правило при использовании функций *max()*, *avg()*, *min*, *count()* и т.п.
having	позволяет задать условия отбора записей после группировки
order by	устанавливает столбцы, по которым будет производится сортировка результата, descending задает обратный порядок сортировки

Условия **<where cond>** и **<having cond>** представляют из себя логические выражения. Если результат выполнения логического выражения истинен, то запись попадет в результат запроса, в противном случае – нет.

Запрос Select

Далее рассмотрим примеры запросов к базе данных **demo.gdb**, состоящей из двух связанных таблиц студентов и групп.

Выполните запросы и посмотрите их результат.

Получить список студентов с присоединенными именами групп

```
select G.GroupName,S.* from STUDENTS S
                left join GROUPS G on S.GroupID=G.GroupID
```

В данном примере используется псевдонимы (алиасы, alias) имен таблиц. Так, таблице **STUDENTS** назначается псевдоним **S**, а таблице **GROUPS** – псевдоним **G** (указаны после имени таблицы и отмечены красным цветом).

Обращение к полям в этом случае осуществляется указанием алиаса и имя поля через символ ‘.’. Выражение **S.*** выдает все поля таблицы, связанной с алиасом **S** (в данном случае – **STUDENTS**).

Условие соединения таблиц **S.GroupID=G.GroupID**, т.е. присоединяется строка из таблицы **GROUPS** для которой значение поля **GroupID** совпадает со значением аналогичного поля из **STUDENTS**.

Запрос Select

Модифицируем запрос добавив условие отбора.

Студенты, фильтрация по дате рождения

```
select G.GroupName,S.* from STUDENTS S
       left join GROUPS G on S.GroupID=G.GroupID
       where S.Birth>'01.01.1990'
```

Так же можно задать диапазон дат с помощью **between**

```
select G.GroupName,S.* from STUDENTS S
       left join GROUPS S on S.GroupID=G.GroupID
       where S.Birth between '01.01.1990' and
       '01.01.1995'
```

Студенты, фамилия которых начинается на заданную букву

```
select G.GroupName,S.* from STUDENTS S
       left join GROUPS G on S.GroupID=G.GroupID
       where S.Surn like 'Ш%'
```

Результатом данного запроса будут все студенты, фамилия которых начинается на букву **Ш**. Строковые поля фильтруются с помощью ключевого слова **like** и содержат в строке фильтра маску.

Запрос Select

Маска **like**

Маска может содержать произвольные символы и напоминает маску выбора файлов в ОС с тем отличием, что один символ обозначается ‘_’, а произвольная последовательность символов ‘%’.

Вывод всех пятибуквенных фамилий (5 ’_’ <пробел> ’%’)

```
select G.GroupName,S.* from STUDENTS S
       left join GROUPS G on S.GroupID=G.GroupID
       where S.Surn like '_____ %'
```

Условия можно комбинировать

```
select G.GroupName,S.* from STUDENTS S
       left join GROUPS G on S.GroupID=G.GroupID
       where S.Surn like '____%' and S.Birth>'01.01.90'
```

Вывод всех студентов из ‘ПР2’

```
select G.GroupName,S.* from STUDENTS S
       left join GROUPS G on S.GroupID=G.GroupID
       where G.GroupName='ПР2'
```

Запрос Select

Модифицируем запрос добавив сортировку с помощью **order by**

Сортировка по имени

```
select G.GroupName,S.* from STUDENTS S
       left join GROUPS G on S.GroupID=G.GroupID
       where S.GroupName='ПР2' order by S.Surn
```

Сортировка по дате рождения

```
select G.GroupName,S.* from STUDENTS S
       left join GROUPS G on S.GroupID=G.GroupID
       order by Birth
```

Сортировка по дате рождения в обратном порядке

```
select G.GroupName,S.* from STUDENTS S
       left join GROUPS G on S.GroupID=G.GroupID
       order by Birth descending
```

Если в запросе сортировка идет по нескольким полям, приоритет полей упорядочивания идет слева направо.

Запрос Select

Запрос – количество студентов в каждой группе

```
select G.GroupName,count(S.StudID) as sCount
       from STUDENTS S left join GROUPS G
       on S.GroupID=G.GroupID group by G.GroupName
```

Результат данного запроса – таблица, состоящая из двух столбцов: имени группы и количества студентов в ней. Подсчет количества студентов в группе осуществляется выражением **count**(S.StudID) **as** sCount. Фраза as **sCount** позволяет задать имя второго столбца, чтобы в дальнейшем к нему можно было обращаться по этому имени. Так как для каждой группы обрабатывается несколько студентов, в запросе присутствует группировка по имени группы **group by** G.GroupName. Таким образом, данный запрос обрабатывает все записи таблицы **STUDENTS** и подсчитывает их количество в рамках каждого значения **G.GroupName**.

Список групп с количеством студентов имя которых начинается с ‘**ПР**’

```
select G.GroupName,count(S.StudID) as sCount
       from STUDENTS S left join GROUPS G
       on S.GroupID=G.GroupID where G.GroupName
       like 'ПР%' group by G.GroupName
```

Запрос Select

Для добавления фильтрации результата по вычисляемому выражению, необходимо задать условие **having**.

Список групп, в которых более двух человек

```
select G.GroupName,count(S.StudID) as sCount
       from STUDENTS S left join GROUPS G
       on S.GroupID=G.GroupID group by G.GroupName
       having count(S.StudID)>2
```

Список групп, в которых более двух человек, с сортировкой по количеству человек и имени группы

```
select G.GroupName,count(s.studid) as sCount
       from STUDENTS S left join GROUPS G
       on S.GroupID=G.GroupID group by G.GroupName
       order by sCount,G.GroupName
```

Запрос Select

Список групп, в которых более двух человек, с сортировкой по имени группы и количеству человек

```
select G.GroupName,count(S.StudID) as sCount
       from STUDENTS S left join GROUPS G
       on S.GroupID=G.GroupID group by G.GroupName
       order by G.GroupName,sCount
```

Рассмотрим отличие **left join** от **right join**. Наглядно это будет видно при форматировании списков групп и количества человек. Сначала добавим в таблицу групп еще две группы.

```
insert into GROUPS (GROUPNAME) values ('ПР7');
insert into GROUPS (GROUPNAME) values ('ПР8');
```

Запрос Select

Рассмотрим два запроса:

Запрос – количество студентов в каждой группе

```
select G.GroupName,count(S.StudID) as sCount
       from STUDENTS S left join GROUPS G
       on S.GroupID=G.GroupID group by G.GroupName
```

Запрос - количество студентов в каждой группе

```
select G.GroupName,count(S.StudID) as sCount
       from STUDENTS S right join GROUPS G
       on S.GroupID=G.GroupID group by G.GroupName
```

Из результата видно, что **right join** (внешние слияние) выбирает и те группы, в которых нет студентов (или для которых нет соответствующих записей в таблице **STUDENTS**).

Обзоры view

Обзоры позволяют создавать “виртуальные” таблицы, позволяющие осуществлять выборку данных, например из нескольких таблиц.

Создадим обзор на основе запросов выше

```
create view StudList(FIO,gName) as
select S.FIO,G.GroupName as gName
       from STUDENTS S left join GROUPS G
       on S.GroupID=G.GroupID order by S.FIO
```

```
select * from StudList
```

Проанализируйте результат.

Обзоры view

Создадим обзор, вычисляющий возраст студентов, используя вспомогательные функции.

```
create view StudAge(FIO,AGE) as
  select FIO,
  Extract(YEAR from cast('Now' as date))-
       Extract(YEAR from Birthday) as AGE
  from students
```

Здесь

`cast('Now' as date)` возвращает текущую дату,

`Extract(YEAR from cast('Now' as date))` из текущей даты извлекает год

`Extract(YEAR from Birthday)` извлекает год из текущей даты

Рассмотрите результат двух запросов (по отдельности)

```
select * from StudAge
Select avg(AGE) from StudAge
```