Лекція 9-2020_2021

- На попередній лекції
- Розділ FROM, оператор JOIN
- Розділ WHERE команди SELECT
- Створення обчислювальних значінь
- Використання підсумкових функцій
- Розділ GROUP BY
- Сортування записів, ORDER BY

Что было в предыдущей лекции

SELECT применяется для извлечения строк, выбранных из одной или нескольких базовых таблиц, представлений или подчиненных запросов.

```
SELECT [TOP n |DISTINCT | DISTINCTROW|ALL]
        select_выражение,...
     [INTO {OUTFILE | DUMPFILE} 'file' options]
 [FROM источник_записей ]
 [WHERE условия_отбора_записей]
 [GROUP BY {атрибут | формула} [ASC | DESC],...]
 [HAVING фильтр_группы]
 [ORDER BY {атрибут | формула} [ASC | DESC],...]
 [LIMIT  колич_строк ]
```


Что было в предыдущей лекции

FROM – Указывает таблицы или представления, которые содержат поля, перечисленные в команде **SELECT**.

WHERE – Определяет, какие записи из таблиц, перечисленных в предложении **FROM**, следует включить в результат выполнения команды **SELECT**. Фактически это фильтр по всем записям.

ORDER BY – Сортирует записи, полученные в результате запроса, в порядке возрастания или убывания на основе значений указанного поля или полей.

Что было в предыдущей лекции

GROUP BY – Объединяет записи с одинаковыми значениями в указанном списке полей в одну группу.

HAVING – Определяет условия выбора из групп. Фактически это фильтр по записям внутри группы.

Порядок выполнения оператора **SELECT:**

Шаг 1. FROM. Определение источника записей для запроса. Вначале строится расширенное декартово произведение всех перечисленных источников (таблиц, представлений, подчиненных запросов), после чего выполняется условие объединения.

Что было в предыдущей лекции

Шаг 2. WHERE. Условное выражение этого раздела применяется к каждой строке источника записей, сформированного в FROM . Результат – набор записей, для которых значение условия = true.

Шаг 3. GROUP BY (если есть). Группировка записей по значениям столбцов из списка имен раздела GROUP BY.

Шаг 4. HAVING (если есть). Записи каждой группы фильтруются по условию в HAVING. Результат выполнения – строки, для которых значение условного выражения = true.

Шаг 5. Выполнение раздела SELECT. Результат – проекция.

Что было в предыдущей лекции

Раздел FROM. РБД могут работать со связанными источниками записей (базовые таблицы, представления, подзапросы). Возможно объединение источников записей за счет условия в разделе Where.

Возможно объединение таблицы сама с собой. В этом случае используются копии одной и той же таблицы, копиям присваиваются различные псевдонимы. Применяется в рекурсивных запросах.

Раздел FROM, объединение с использованием JOIN

JOIN — оператор языка SQL, реализация операции соединения реляционной алгебры. Входит в раздел **FROM** операторов **SELECT**, **UPDATE** или **DELETE**.

FROM
Table1
{**INNER** | {**LEFT** | **RIGHT** | **FULL**} **OUTER** | **CROSS** } **JOIN**
Table2
{**ON** <condition> (field_name [,... n])}

В большинстве СУБД при указании слов LEFT, RIGHT, FULL слово OUTER можно опустить. Слово INNER также в большинстве СУБД можно опустить.

В общем случае СУБД при выполнении соединения проверяет условие *condition*. Оператор сравнения: «=», «<», «>», «<=», «>=» или «<>»

INNER JOIN (внутреннее объединение)

Оператор *внутреннего соединения* INNER JOIN соединяет два источника записей. Порядок их следования в записи оператора не важен, т.к. оператор симметричный. Вначале строится расширенное декартово произведение. После проверяется условие соединения (вычисляется предикат соединения). Если условие истинно, в таблицу-результат добавляется соответствующая «соединённая» строка.

Это логический алгоритм действий. Физически каждая СУБД выполняет соединение по-разному, например, соединение вложенными циклами или хешированием. Но любая реализация дает такой же результат, как и логический алгоритм.

INNER JOIN (внутреннее объединение)

Поликлиника

Табл. «Пац»

КодП	Фам
1	Крутой
2	Семенов
3	Орлова
4	Онегин
6	Вий

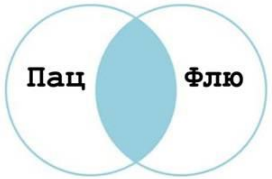
Табл. «Флю»

КодФ	ДатаФ	Рез	КодП
1	10.10.2008	ок	1
2	20.03.2008	бр	1
3	30.12.2009	ок	1
4	05.12.2009	ок	2
5	09.10.2009	пн	3
6	30.12.2009	ок	3
7	25.12.2009	пн	

PK- КодП — FK- КодП

INNER JOIN (внутреннее объединение)

Пример 11. Найти всех , кто делал хоть один раз Флю.




```
Select Фам, ДатаФ, Рез
From Пац INNER JOIN Флю
ON Пац.КодП =Флю.КодП;
```

Результат:

Крутой	10.10.2001	ок
Крутой	20.03.2002	бр
Крутой	30.12.2005	ок
Семенов	05.12.2005	ок
Орлова	09.10.2005	пн
Орлова	30.12.2005	ок

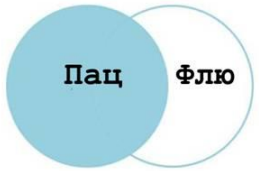
LEFT [OUTER] JOIN (левое внешнее объединение)

Выбираются все записи первой (левой) таблицы, и те записи из второй (правой), которые соответствуют условию объединения .

Пример 12. Для всех пациентов указать дату прохождения Флю

```
Select Фам, ДатаФ, Рез
From Пац LEFT JOIN Флю
ON Пац.КодП =Флю.КодП;
```

Крутой	10.10.2001	ок
Крутой	20.03.2002	бр
Крутой	30.12.2005	ок
Семенов	05.12.2005	ок
Орлова	09.10.2005	пн
Орлова	30.12.2005	ок
Онегин	null	null
Вий	null	null



Не симметричный оператор. Важен порядок следования таблиц в операторе **From.** OUTER можно опустить.

RIGHT [OUTER] JOIN (правое внешнее объединение)

Выбираются все записи второй (правой) таблицы, и те записи из первой (левой), которые соответствуют условию объединения .

Пример 13. Для всех пациентов указать дату прохождения Флю

```
Select Фам, ДатаФ, Рез
From Флю RIGHT JOIN Пац
ON Пац.КодП =Флю.КодП;
```

Крутой	10.10.2001	ок
Крутой	20.03.2002	бр
Крутой	30.12.2005	ок
Семенов	05.12.2005	ок
Орлова	09.10.2005	пн
Орлова	30.12.2005	ок
Онегин	null	null
Вий	null	null

Не симметричный оператор. Важен порядок следования таблиц в операторе **From.** OUTER можно опустить.

Здесь тот же самый запрос, тот же самый результат

FULL [OUTER] JOIN
(полное внешнее объединение)

1. В результат включается внутреннее соединение (INNER JOIN) первой и второй таблиц по предикату *p*.

2. В результат добавляются те записи первой таблицы, которые не вошли во внутреннее соединение на шаге 1. Для таких записей поля, соответствующие второй таблице, заполняются значениями NULL.

3. В результат добавляются те записи второй таблицы, которые не вошли во внутреннее соединение на шаге 1. Для таких записей поля, соответствующие первой таблице, заполняются значениями NULL.

Оператор симметричен.

FULL [OUTER] JOIN
(полное внешнее объединение)

Пример 14. Полное внешнее объединение.

```
Select Фам, ДатаФ, Рез
From Флю FULL JOIN Пац ON
Пац.КодП = Флю.КодП;
```

Крутой	10.10.2001	ок
Крутой	20.03.2002	бр
Крутой	30.12.2005	ок
Семенов	05.12.2005	ок
Орлова	09.10.2005	пн
Орлова	30.12.2005	ок
Онегин	null	null
Вий	null	null
null	25.12.2009	пн

CROSS JOIN

Оператор перекрёстного соединения, или декартова произведения CROSS JOIN соединяет две таблицы. Оператор является симметричным. Каждая строка одной таблицы соединяется с каждой строкой второй таблицы, давая тем самым в результате все возможные сочетания строк двух таблиц.

```
Select Фам, ДатаФ, Рез
  From Флю CROSS JOIN Пац;
```

или

```
Select Фам, ДатаФ, Рез
From Флю, Пац;
```


Раздел FROM. Значение NULL

NULL – специальное значение, указывающее на то, что мы не знаем, какое значение.

Оператор **IS NULL** используется для сравнения текущего значения со значением **NULL**

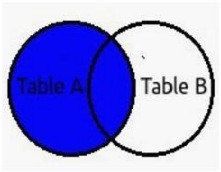
IS NOT NULL используется для проверки присутствия значения в поле.

Пример 15. Пациенты, которые никогда не делали Флю

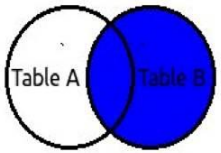
```
Select Фам, ДатаФ, Рез
From Пац LEFT JOIN Флю
ON Пац.КодП = Флю.КодП
Where (Флю.КодП is null);
```

Онегин	null	null
Вий	null	null

Выполнение оператора SELECT

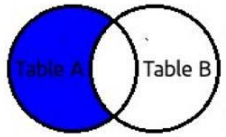



```
Select [list]
From A Left Join B
On A.Value = B.Value;
```

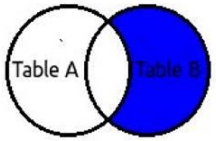



```
Select [list]
From A Right Join B
On A.Value = B.Value;
```


Выполнение оператора SELECT




```
Select [list]
From A Left Join B
On A.Value = B.Value
Where B.Value Is Null;
```




```
Select [list]
From A Right Join B
On A.Value = B.Value
Where A.Value Is Null;
```


Ограничение возвращаемых строк

Производитель/СУБД	Синтаксис ограничения
DB2	SELECT * FROM T FETCH FIRST 10 ROWS ONLY
Firebird	SELECT FIRST 10 * FROM T
Informix	SELECT FIRST 10 * FROM T
Interbase	SELECT * FROM T ROWS 10
Microsoft	SELECT TOP 10 * FROM T ORDER BY col
MySQL	SELECT * FROM T LIMIT 10
SQLite	SELECT * FROM T LIMIT 10
PostgreSQL	SELECT * FROM T LIMIT 10
Oracle	SELECT * FROM T WHERE ROWNUM <= 10

Выбрать все столбцы из таблицы T, выборку ограничить 10 строками.

Раздел WHERE

Задает условие отбора записей.

Для каждой записи из источника записей, определенного в разделе **FROM,** вычисляется логическое выражение из раздела **WHERE.** В результирующую выборку попадают те записи, для которых значение вычисленного логического выражения = true

Операции сравнения = | > | >= | < | <= | <>

Логические операторы: **Not, Or, And, Xor, Eqv**

Пример 1. Пациенты, проживающие на улице Сумской, в доме 8, в квартире 1.

```
Select Фам From Пациенты
  Where ((Ул="Сумская") and (Дом="8")
                        and (Кв=1));
```

Раздел WHERE. Логические выражения

Пример 2. Пациенты, кроме тех, кто проживает на ул. Сумской, в доме 8, в квартире 1.

```
Select Фам From Пациенты
  Where Not ((Улица="Сумская")  and
             (Дом="8")and (Кв=1));
```

Пример 3. Пациенты, проживающие на улице Сумской или Пушкинской:

```
Select Фам From Пациенты
  Where ((Улица="Сумская") Or
        (Улица="Пушкинская"));
```

Оператор IS NULL используется для сравнения текущего значения со значением NULL.

IS NOT NULL – для проверки присутствия значения в поле.

Раздел WHERE. Логические выражения

Пример 4. Пациенты, которые никогда не делали Флю

```
Select Фам, ДатаФ, Рез From Пац LEFT JOIN Флю
                    ON Пац.КодП = Флю.КодП
   Where (Флю.КодП is null);
```


Раздел WHERE, предикат IN

Определяет множество, которому данное значение может принадлежать (**IN**) или не принадлежать (**NOT IN**).

Пример 5. Сформировать список пациентов, проживающих на улице Сумской или Пушкинской.

```
Select Фам From Пациенты
     Where (Улица IN
  ("Сумская","Пушкинская"));
```

Пример 6. Сформировать список пациентов, кроме тех, кто проживает на улице Сумской или Пушкинской.

```
Select Фам From Пациенты
     Where (Улица NOT IN
  ("Сумская","Пушкинская"));
```


Раздел WHERE, предикат BETWEEN ... AND

Используется для сравнения значения с заданным диапазоном.

Границы включаются в диапазон (`between ... and`)

`a between b and c → a >= b and a <= c`

Границы не включаются (`NOT between ... and`)

`a not between b and c → a<b and a>c`

Пример 7. Вывести список врачей со стажем от 5 до 10 лет включительно.

```
Select Фам, Имя, Отч From Врачи
       Where (Стаж Between 5 and 10);
```

Раздел WHERE, предикат BETWEEN ... AND

Пример 8. Сформировать список врачей, кроме тех, у кого стаж от 5 до 10 лет

```
Select Фам, Имя, Отч From Врачи
       Where (Стаж NOT Between 5 and 10);
```

Предикат `between ... and` можно использовать для сравнения символьных значений (в этом случае сравнивается их ASCII код).

Пример 9. Сформировать список пациентов, у которых первые буквы фамилий находятся в диапазоне от A до B.

```
Select Фам From Пациенты  Where
 (Left(Фам,1) Between "A" and "B");
```


Раздел WHERE, предикат LIKE

Используется для сравнения строк с шаблоном.

Некоторые шаблоны, используемые с LIKE:

любой одиночный символ (**ACCESS - ?** , **MySQL - _**)

любое количество символов (**ACCESS - *** , **MySQL- %**)

Пример 10. Сформировать список пациентов, у которых фамилии начинаются с буквы Ф.

ACCESS
```
Select Фам From Пациенты  Where
                  (Фам Like "Ф*");
```

MySQL
```
Select Fam From Pacient Where
                  (Fam Like 'Ф%'); или
      Select Fam From Pacient Where
                  Left(Fam,1) Like 'Ф';
```


Способы формирования шаблона для использования в предикате LIKE

Пример 11. Сформировать список пациентов, у которых фамилии начинаются с буквы A.

1) Шаблон задается в виде константы
```
Select Fam From Pacient Where (Fam Like "A*");
```

2) Шаблон извлекается из поля List1 формы MyF
```
Select Fam From Pacient
  Where ( Left (Fam,1)= Forms!MyF! List1);
```

3) Шаблон задается в параметрическом запросе
```
Select Fam From Pacient
      Where Fam Like
        [Введите первую букву фамилии] & "*";
```


Построение вычисляемых значений

Для создания вычисляемого поля в списке SELECT следует указать некоторое выражение языка SQL. Применяются операции сложения, вычитания, умножения и деления, а также встроенные функции языка SQL.

Можно использовать имя столбца только той таблицы или запроса, которые указаны в списке предложения FROM.

Пример 12.
```
SELECT Товар.Цена * Продажа.Количество AS
НаСумму
FROM Товар INNER JOIN Продажа
ON Товар.КодТовара= Продажа.КодТовара
```

Использование итоговых функций

- Count (Выражение) - определяет количество записей в выходном наборе SQL-запроса;
- Min/Max (Выражение) - определяют наименьшее и наибольшее из множества значений в некотором поле запроса;
- Avg (Выражение) - эта функция позволяет рассчитать среднее значение множества значений, хранящихся в определенном поле отобранных запросом записей. Оно является арифметическим средним значением, т.е. суммой значений, деленной на их количество.
- Sum (Выражение) - вычисляет сумму множества значений, содержащихся в определенном поле отобранных запросом записей.

Особенности использования итоговых функций

1. Выражение – имена столбцов из одной или нескольких таблиц (представлений), указанных в FROM. Выражение может вычисляться и по значениям нескольких таблиц.
2. Функции возвращают единственное значение.
3. Аргумент COUNT, MIN, MAX – числовой или нечисловой.
4. Аргумент SUM и AVG – числовой. [исключение COUNT(*)].
5. При вычислении результатов функций сначала исключаются все значения NULL, после чего операция применяется к оставшимся конкретным значениям столбца.

Особенности использования итоговых функций

6. Вариант COUNT(*) – подсчет всех строк в результирующей таблице (значения могут быть любые).
7. Для исключения дублирующих значений до применения функции – перед именем столбца слово DISTINCT.
8. DISTINCT – в любом запросе, но не более одного раза.
9. Итоговые функции могут использоваться только в списке SELECT и в HAVING.
10. Если список в SELECT содержит итоговые функции, а в тексте запроса отсутствует GROUP BY, то список SELECT должен состоять **только** из итоговых функций с аргументами.

Примеры применения итоговых функций в SELECT

Пример 13. Продавцы производят продажу товара, о чем БД делается отметка с указанием продавца, проведшего сделку. Посчитать количество сделок за сегодня.

```
Select Count(Продавец) FROM Продажа
        Where ДатаПродажи=Date();
```

Пример 14. Сколько продавцов сегодня провели сделки.

```
Select Count(Distinct(Продавец)) FROM Продажа
Where ДатаПродажи=Date();
```

Пример 15. Сумма товара, проданногосегодня.

```
SELECT Sum(Товар.Цена* Продажа.Количество)
                                    AS Сумма
   FROM Товар INNER JOIN Продажа
       ON Товар.КодТовара= Продажа.КодТовара
   Where ДатаПродажи=Date();
```

Раздел GROUP BY

GROUP BY объединяет записи с одинаковыми значениями в указанном списке полей, используется для итоговых запросов.

Правила:

1. Если в оператор SELECT включено GROUP BY, список выбора должен состоять из выражений с итоговыми функциями или/и из столбцов, указанных в GROUP BY
2. Предложение GROUP BY не обязательное.
3. Значения NULL, которые находятся в полях, заданных в предложении GROUP BY группируются и не опускаются, но статистические функции не обрабатывают значение NULL.

Раздел GROUP BY

GROUP BY объединяет записи с одинаковыми значениями в указанном списке полей, используется для итоговых запросов.

Правила:

1. Если в оператор SELECT включено GROUP BY, список выбора должен состоять из выражений с итоговыми функциями или/и из столбцов, указанных в GROUP BY
2. Предложение GROUP BY не обязательное.
3. Значения NULL, которые находятся в полях, заданных в предложении GROUP BY группируются и не опускаются, но статистические функции не обрабатывают значение NULL.

Раздел GROUP BY

5. HAVING – условие отбора в группы (фильтр). Применяется только к столбцам, указанным в GROUP BY, к столбцам итоговых функций и к выражениям, содержащим итоговые функции.
6. Если в SELECT есть и WHERE и HAVING, то WHERE исключает записи из группировки, а HAVING – фильтр к записям после группировки.

Пример GROUP BY

Пример 16. Задача. Есть упрощенный вариант двух таблиц. Спец {КодС, Спец} Врач {КодВ, КодС, Фам, Стаж}

Запрос на объединение двух таблиц

```
SELECT Спец.Спец, Врачи.Фам, Врачи.Стаж
FROM Спец INNER JOIN Врачи
ON Спец.КодС = Врачи.КодС;
```

Результат сохраним, как **Врач_ст**

Спец	Фамилия	Стаж
Хирург	Попов	20
Терапевт	Котова	16
Хирург	Петров	18
Терапевт	Петрова	10
Невропатолог	Сидоров	10
Невропатолог	Иванова	15

GROUP BY. Почему одинаковые результаты?

По каждой специализации вывести фамилии врачей-ветеранов по стажу. Ветераны те, у кого стаж > 15 лет.

Спец	Фамилия	Стаж
Хирург	Попов	20
Терапевт	Котова	16
Хирург	Петров	18
Терапевт	Петрова	10
Невропатолог	Сидоров	10
Невропатолог	Иванова	15

Результат

Спец	Фамилия	Стаж
Невропатолог	Иванова	15
Терапевт	Котова	16
Хирург	Попов	20
Хирург	Петров	18

а)
```
SELECT Спец, Фам, Стаж
FROM Врач_ст
GROUP BY Спец, Фам, Стаж
HAVING (Max(Стаж) >=15);
```

б)
```
SELECT Спец, Фам, Стаж
FROM Врач_ст
GROUP BY Спец, Фам, Стаж
HAVING (Стаж >=15);
```

в)
```
SELECT Спец,Фам, Стаж
FROM Врач_ст
WHERE (Стаж >=15)
GROUP BY Спец, Фам,Стаж;
```

Почему одинаковые результаты?