Раздел 4. Основы языка SQL

Тема 4.1 (часть 2)
Язык запросов (Data Query Language)

Вопросы лекции:

1. **Соединение таблиц в запросах.**
2. Вложенные запросы и комбинированные запросы.
3. Представления.

Соединение таблиц в запросах

Запросы к одной таблице составляют малую долю всех запросов к БД. Например, в таблице оценок содержатся только внешние ключи к таблице студентов и запрос только к ней не позволит представить ее наглядно.

Так как в нормализованной БД обычно много таблиц, поэтому операция соединения таблиц в запросах одна из самых распространенных операций в запросах на выборку (языке DQL).

Соединять таблицы в запросах можно различными способами, при этом могут быть получены разные результаты.

Способы соединения таблиц в запросе SELECT:

1. Декартово произведение.
2. Внутреннее (естественное) соединение таблиц.
3. Внешнее соединение таблиц (операция OUTER JOIN).
4. Самосоединения таблиц.

Соединение таблиц в запросах

Декартово произведение

Это операция, при выполнении которой каждая строка одной таблицы соединяется (склеивается) с каждой строкой другой таблицы.

Выполняется в том случае, если во фразе FROM присутствует список из нескольких таблиц, но **не задается никакой специальной операции соединения** (нет фразы JOIN).

Например, запрос типа SELECT …….**FROM таблица1, таблица2** возвратит количество строк, равное произведению числа строк в обеих таблицах и если не применять операций отбора столбцов, их число суммируется.

Применяется крайне редко, например, если в БД есть таблица с одной строкой-константами (наименование учреждения, реквизиты, справочная информация), то ее всегда можно объединить с другими любыми таблицами.

Наиболее частыми являются запросы с операцией JOIN

Соединение таблиц в запросах

Внутреннее (естественное) соединение таблиц

Можно соединять **только таблицы**, имеющие общие столбцы.

При выполнении данной операции соединяются (склеиваются) только строки, **имеющие общие значения в столбце связи**.

Обычно **применяется для соединения таблиц, связанных отношением «один-ко-многим»**, а в качестве столбцов связи используется первичный ключ главной таблицы и соответственно – внешний ключ подчиненной.

Возвращаются только строки, имеющие связанные записи в подчиненной таблице.

Имеется 2 равноценных по результату но разных по записи способа реализации:

А) **выборка** из декартова произведения;

Б) операция соединения **[INNER] JOIN**

Внутреннее (естественное) соединение таблиц

А) выборка из декартова произведения;

Например, если требуется из таблиц «Студенты» и «Оценки» выбрать фамилии всех студентов, имеющих оценки (есть связанные записи), это можно сделать так:

```
SELECT students.name_st, marks.mark
FROM students, marks
WHERE students.cod_st=marks.cod_st
```

name_st	mark
Иванов	5
Иванов	3
…	
Петров	4
Петров	5
Петров…	5

Б) операция соединения [INNER] JOIN

Тот же запрос:

```
SELECT students.name_st, marks.mark
FROM students JOIN marks
ON students.cod_st=marks.cod_st
```

В текстах запросов можно использовать синонимы имен таблиц типа – st.name_st, m.mark. Можно связывать любые столбцы, совпадающие по типу, но аккуратно.

Соединение таблиц в запросах

Внутреннее (естественное) соединение таблиц

Если требуется выбрать информацию из трех и более таблиц, например «Студенты», «Предметы» и «Оценки» - это можно сделать так:

А) выборка из декартова произведения;

```
SELECT st.cod_st, st.name_st, pr.name_pr, m.mark
FROM students st, marks m, predmet pr
WHERE st.cod_st=m.cod_st AND pr.cod_pr=m.cod_pr
```

cod_st	name_st	name_sub	mark
1	Иванов	Математика	5
1	Иванов	Физика	4
…	…	…	
2	Петров	Физика	4
2	Петров	Информатика	5
2	Петров	История	4
…	…	…	…
3	Иванов	Математика	4
3	Иванов	Информатика	3
…	…	…	…

Б) операция соединения [INNER] JOIN

Тот же запрос:

```
SELECT st.cod_st, st.name_st, pr.name_pr, m.mark
FROM students st
JOIN marks m ON st.cod_st=m.cod_st
JOIN predmet pr ON pr.cod_pr=m.cod_pr
```

Для каждой добавляемой таблицы необходимо условие ее соединения, иначе – декартово произв.

Внешнее соединение таблиц (операция OUTER JOIN)

При внешнем соединении, в отличие от внутреннего, в результат выборки попадают **не только все связанные строки** обеих таблиц, **но и строки** (из одной или обеих), для которых нет связанных в другой таблице.

При этом недостающим значениям столбцов присваивается NULL.

Возможны **три варианта** внешнего соединения двух таблиц:

- **LEFT** [OUTER] JOIN – левое внешнее соединение – выборка дополняется строками таблицы, указанной слева от слова JOIN;
- **RIHGT** [OUTER] JOIN – правое внешнее соединение – дополняется строками таблицы, указанной справа от слова JOIN;
- **FULL** [OUTER] JOIN – полное внешнее соединение – из обеих таблиц.

Например, запрос по среднему баллу студентов - даже если нет оценки будет таким:

```
SELECT st.cod_st, st.name_st, AVG (m.mark) avg_mark
FROM students st LEFT JOIN marks m ON st.cod_st=m.cod_st
GROUP BY st.cod_st, st.name_st
```

(студенты без оценок попадут с NULL)

Самосоединения таблиц

При самосоединении производится соединение таблицы **со своей копией**.

Используются достаточно редко, для решения некоторых задач, **например для поиска в таблице повторяющихся значений некоторых столбцов** – однофамильцы, из одного города и т.д.

Например:

```
SELECT DISTINCT s1.name_st
FROM students s1, students s2
WHERE s1.name_st=s2.name_st AND s1.cod_st<>s2.cod_st
```

Слово DISTINCT добавлено на случай, если имеются фамилии, повторяющиеся более двух раз.

В общем – эту задачу можно решить и обычным запросом к одной таблице с использованием HAVING COUNT – это лишь пример самосоединения.

Раздел 4. Основы языка SQL

Тема 4.1 (часть 2)

Язык запросов (Data Query Language)

Вопросы лекции:

1. Соединение таблиц в запросах.
2. **Вложенные запросы и комбинированные запросы.**
3. Представления.

Вложенные запросы

Вложенными являются запросы, которые содержатся **в теле другого запроса**.

Могут использоваться

- во фразах FROM, WHERE и HAVING, а также
- в списке после слова SELECT, создавая вычисляемый столбец.

Вложенные запросы являются средством для алгоритмического подхода к решению сложной задачи выборки – представления ее в виде последовательности более простых выборок в виде вложенных запросов.

Основные правила оформления вложенных запросов:

- Тело вложенного запроса всегда заключается в скобки;
- Вложенные запросы могут содержать другие вложенные запросы, при этом выполнение запроса всегда начинается с самого «глубокого» вложенного запроса и заканчивается внешним запросом;
- Во вложенном запросе не следует использовать фразу ORDER BY, поскольку сортировка результатов должна быть выполнена один раз после полного выполнения всего запроса

Вложенные запросы

Вложенные запросы могут быть:

- вложенные запросы во фразе FROM;
- вложенные запросы как альтернатива соединению таблиц в запросах;
- вложенные запросы в условиях отбора (WHERE и HAVING);
- вложенные запросы в качестве вычисляемых столбцов;
- коррелированные и некоррелированные вложенные запросы.

Вложенные запросы

- вложенные запросы во фразе FROM;

```
SELECT MIN (avg_mark) min_avg_mark, MAX(avg_mark) max_avg_mark
FROM (SELECT AVG (mark) avg_mark FROM marks GROUP BY cod_st)
```

Данный запрос выводит минимальный и максимальный средний балл студентов

- вложенные запросы как альтернатива соединению таблиц в запросах;

Запрос выводит оценку студента Иванова по математике

```
SELECT mark FROM marks
WHERE cod_st IN
(SELECT cod_st FROM students WHERE name_st='Иванов')
AND cod_sub IN
(SELECT cod_sub FROM subjects WHERE name_sub='Математика')
```

Вложенные запросы

- вложенные запросы в условиях отбора (WHERE и HAVING);

```
SELECT name_st FROM students WHERE born=
(SELECT MAX(born) FROM students)
```

Данный запрос выводит самых молодых студентов

- вложенные запросы в качестве вычисляемых столбцов;

Запрос выводит средний балл студентов в вычисляемый столбец.

```
SELECT …(SELECT …) FROM …
```

```
SELECT cod_st, name_st,
(SELECT AVG(mark) FROM marks WHERE cod_st=students.cod_st)
avg_mark
FROM students
```

Комбинированные запросы

Над результатами запросов можно выполнять обычные операции над множествами – комбинировать.

Они выполняются только над запросами, которые возвращают одинаковое количество столбцов одинакового типа

Согласно стандартам SQL установлены следующие операции комбинирования запросов:

- UNION [ALL] – объединение запросов;
- INTERSECT [ALL] – пересечение запросов;
- ECXEPT (в СУБД ORACLE - MINUS) [ALL] – разность запросов.

Ключевое слово [ALL] запрещает удаление дубликатов из результатов операций над запросами

```
SELECT name_st FROM students1
UNION ALL //всех однофамильцев, не удаляет дубликаты
SELECT name_st FROM students2
//если нет all удалит всех однофамильцев!!

SELECT name_st FROM students1
INTERSECT //выводит только однофамильцев
SELECT name_st FROM students2

SELECT name_st FROM students1
MINUS //все students1 если ни один не входит в students2
SELECT name_st FROM students2
```

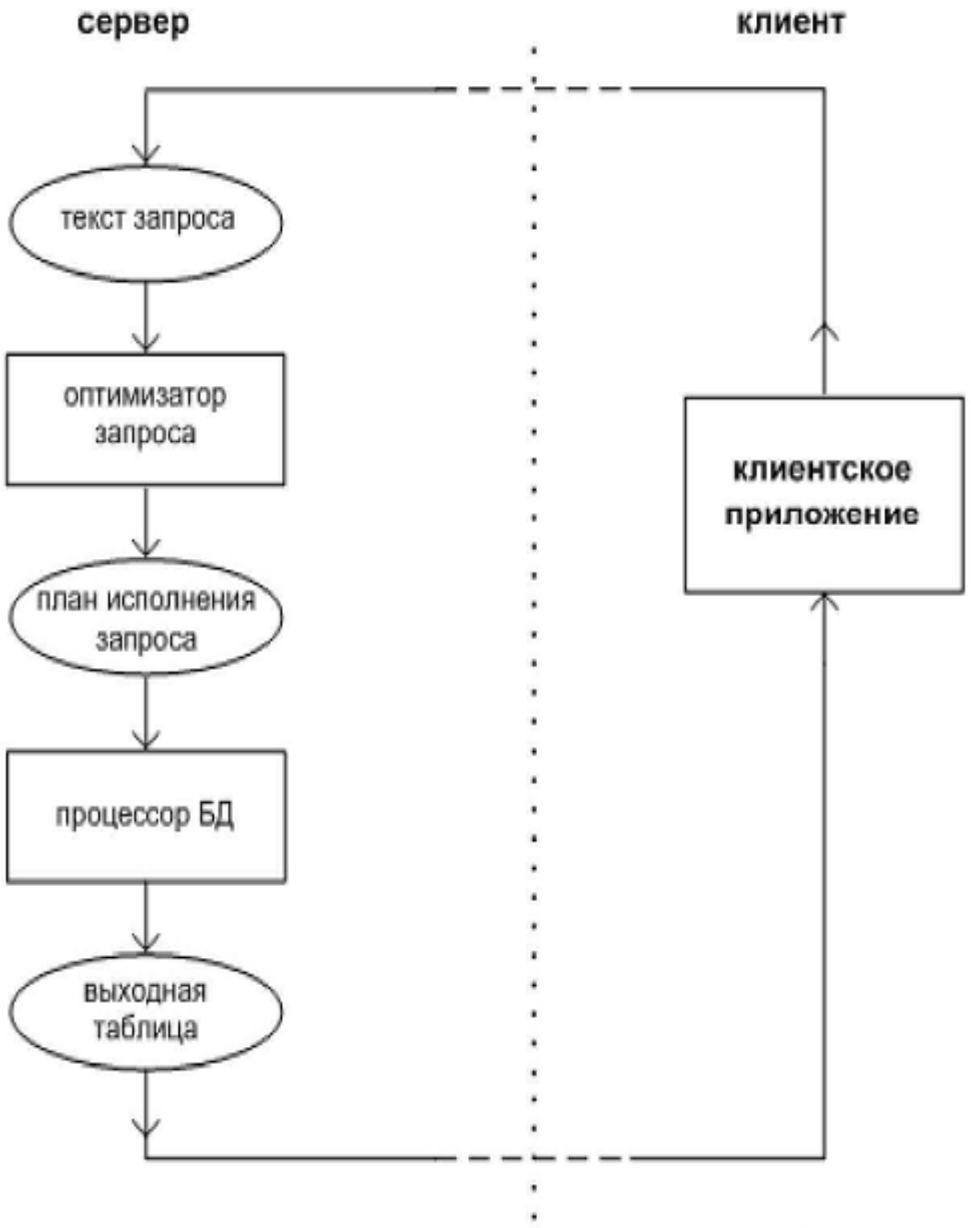
Раздел 4. Основы языка SQL

Тема 4.1 (часть 2)

Язык запросов (Data Query Language)

Вопросы лекции:

1. Соединение таблиц в запросах.
2. Вложенные запросы и комбинированные запросы.
3. **Представления.**



Порядок обработки SQL-запроса SELECT ….

Часто выполняемые запросы на выборку нет смысла пересылать по сети и компилировать каждый раз заново.

Разумнее их постоянно хранить в БД вместе с планами выполнения, что ускорит для СУБД обработку таких запросов, особенно если они являются комбинированными или вложенными.

Представление – это **именованные запросы** на выборку, сохраненные в БД, которые при любом обращении к ним по имени создают виртуальную таблицу, наполняя ее актуальными данными из БД.

Это позволяет заранее оптимизировать их планы выполнения и тем самым ускорить выполнение многократно выполняемых одинаковых запросов к часто изменяющимся данным.

Представление позволяет пользователю возможность видеть только часть базы данных и скрывать некоторые таблицы или их части.

С представлениями можно работать также как и с реальными таблицами, включать их в другие запросы и т.д.

Представление – это **именованные запросы** на выборку, сохраненные в БД, которые при любом обращении к ним по имени создают виртуальную таблицу, наполняя ее актуальными данными из БД.

Это позволяет заранее оптимизировать их планы выполнения и тем самым ускорить выполнение многократно выполняемых одинаковых запросов к часто изменяющимся данным.

```
CREATE VIEW имя [(список_столбцов)]
AS
SELECT ...// любой запрос на выборку
```

```
CREATE VIEW stud_mark (cod_st, name_st, avg_mark)
AS
SELECT  st.cod_st, st.name_st, AVG(m.mark)
FROM students st LEFT JOIN marks m
ON st.cod_st=m.cod_st
GROUP BY st.cod_st, st.name_st
```