Проектирование баз данных, ч.1

ФИО преподавателя: Смирнов Михаил Вячеславович

e-mail: smirnovmgupi@gmail.com

Лекция 3

Внешние и внутренние соединения.

Правила. Описание оператора SQL JOIN.

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

Выбор первого столбца соединения

Выбор первого операнда соединения

Выбор типа соединения

Приглашение к соединению

Выбор второго операнда соединения

Выбор условия соединения и второго столбца соединения при соблюдении условия эквивалентности

Типы используемых соединений

Небольшое предупреждение и корректировка. Приведенные типы соединений слегка субъективны. Связано это с тем, что данные соединения используются в СУБД MS SQL Server и могут немного отличаться от соединений в других СУБД.

Итак, мы рассмотрим 5 основных типов соединений операндов в БД:

- Внутреннее соединение - Inner join (IJ)
- Внешнее соединение - Outer join (OJ)
- Перекрестное соединение - Cross join (CJ)
- Полусоединение - Semi-join (SJ)
- Анти-полусоединение - Anti-semi-join (ASJ)

Практическое применение.

Разворачиваем следующие две таблицы в базе данных, используя скрипты cust и sales:

Customers (покупатели)

Результаты | Сообщения

	Cust_Id	Cust_Name
1	1	Craig
2	2	John Doe
3	3	Jane Doe

Sales (продажи)

Результаты | Сообщения

	Cust_Id	Item
1	2	Camera
2	3	Computer
3	3	Monitor
4	4	Printer

Условие эквивалентности для соединения

Внутреннее соединение (Inner JOIN)

Внутреннее соединение находит пары строк, которые соединяются и удовлетворяют предикату соединения.
Например, показанный ниже запрос (скрипт inner join) использует предикат соединения "S.Cust_Id = C.Cust_Id", позволяющий найти все продажи и сведения о клиенте с одинаковыми значениями Cust_Id:

SQLQuery4.sql - 18...test (mikko (52))*

```
select *
from Sales S inner join Customers C
on S.Cust_Id = C.Cust_Id
```

Результаты | Сообщения

	Cust_Id	Item	Cust_Id	Cust_Name
1	2	Camera	2	John Doe
2	3	Computer	3	Jane Doe
3	3	Monitor	3	Jane Doe

Важный момент!

Таблицы кодируются переменными (S-Sales, C-Customers) с целью упрощения процесса идентификации предиката

Cust_Id = 3 купил два наименования, поэтому он фигурирует в двух строках результирующего набора.
Cust_Id = 1 не купил ничего и потому не появляется в результате.
Для Cust_Id = 4 тоже был продан товар, но поскольку в таблице нет такого клиента, сведения о такой продаже не появились в результате.
Внутренние соединения полностью коммутативны. "A inner join B" и "B inner join A" эквивалентны.

Внешнее соединение(Outer JOIN)

Соединение двух таблиц, в результат которого в обязательном порядке входят строки либо одной, либо обеих таблиц. Бывает левое внешнее соединение, правое внешнее соединение или же полное внешнее соединение.
Предположим, что мы хотели бы увидеть список всех продаж; даже тех, которые не имеют соответствующих им записей о клиенте (левое внешнее соединение, ojleft_script).

```
select *
from Sales S left outer join Customers C
on S.Cust_Id = C.Cust_Id
```

Результаты | Сообщения

	Cust_Id	Item	Cust_Id	Cust_Name
1	2	Camera	2	John Doe
2	3	Computer	3	Jane Doe
3	3	Monitor	3	Jane Doe
4	4	Printer	NULL	NULL

Сервер возвращает вместо данных о клиенте значение NULL, поскольку для проданного товара 'Printer' нет соответствующей записи клиента.

Внешнее соединение(Outer JOIN)

Используя полное внешнее соединение (ojfull_script), можно найти всех клиентов (независимо от того, покупали ли они что-нибудь), и все продажи (независимо от того, сопоставлен ли им имеющийся клиент):

SQLQuery4.sql - 18...test (mikko (52))*

```
select *
from Sales S full outer join Customers C
on S.Cust_Id = C.Cust_Id
```

Результаты | Сообщения

	Cust_Id	Item	Cust_Id	Cust_Name
1	2	Camera	2	John Doe
2	3	Computer	3	Jane Doe
3	3	Monitor	3	Jane Doe
4	4	Printer	NULL	NULL
5	NULL	NULL	1	Craig

В случае правого внешнего соединения, в соответствии с таблицей будут выведены все покупатели, вне зависимости от того, покупали ли они что-нибудь.

Соединение	Выводятся …
A left outer join B	Все строки A
A right outer join B	Все строки B
A full outer join B	Все строки A и B

Перекрестное соединение (Cross JOIN)

Перекрестное соединение выполняет полное Декартово произведение двух таблиц. То есть это соответствие каждой строки одной таблицы - каждой строке другой таблицы. Для перекрестного соединения ***нельзя*** определить предикат соединения, используя для этого предложение ON.

```
select *
from Sales S cross join Customers C
```

	Cust_Id	Item	Cust_Id	Cust_Name
1	2	Camera	1	Craig
2	3	Computer	1	Craig
3	3	Monitor	1	Craig
4	4	Printer	1	Craig
5	2	Camera	2	John Doe
6	3	Computer	2	John Doe
7	3	Monitor	2	John Doe
8	4	Printer	2	John Doe
9	2	Camera	3	Jane Doe
10	3	Computer	3	Jane Doe
11	3	Monitor	3	Jane Doe
12	4	Printer	3	Jane Doe

cj_script

Перекрестные соединения используются довольно редко. Никогда не стоит пересекать две большие таблицы, поскольку это задействует очень дорогие операции и получится очень большой результирующий набор.

Полусоединение и анти-полусоединение (Semi-JOIN)

Полусоединение возвращает строки только одной из соединяемых таблиц, без выполнения соединения полностью. Анти-полусоединение возвращает те строки таблицы, которые не годятся для соединения с другой таблицей; т.е. они в обычном внешнем соединении выдавали бы NULL.
Чаще всего полусоединение используется в плане подзапроса с EXISTS (sj_script)

```
select *
from Customers C
where exists (
    select *
    from Sales S
    where S.Cust_Id = C.Cust_Id
)
```

	Cust_Id	Cust_Name
1	2	John Doe
2	3	Jane Doe

Существуют левые и правые полусоединения. Левое полусоединение возвращает строки левой (первой) таблицы, которые соответствуют строкам из правой (второй) таблицы, в то время как правое полусоединение возвращает строки из правой таблицы, которые соответствуют строкам из левой таблицы.
Подобным образом может использоваться анти-полусоединение для обработки подзапроса с NOT EXISTS.

Чтение на дом

- Книга «Системы баз данных», стр. 263-282 (если не читали по итогам прошлой лекции)
- Русский Кренке, стр. 287-230.
- Английский Кренке, 119-137.

Спасибо за внимание!