1.1.1. Объединение данных из нескольких таблиц

Мы рассмотрели команды с одним ИД – тС.

В общем виде в качестве источника запроса могут использоваться несколько ИД. Причем, в качестве ИД могут использоваться не только базовые таблицы РБД, хранящиеся в физической памяти машины, но и результаты выполнения ранее созданных запросов, которые в основном представляются в виде виртуальных (временных, рабочих) таблиц. Некоторые СУБД дают возможность создавать представления БД или курсоры, которые фактически являются хранимыми в РБД запросами с именованными полями. С их помощью создаются виртуальные таблицы, позволяющие пользователям иметь свой взгляд на данные без увеличения их объема в БД.

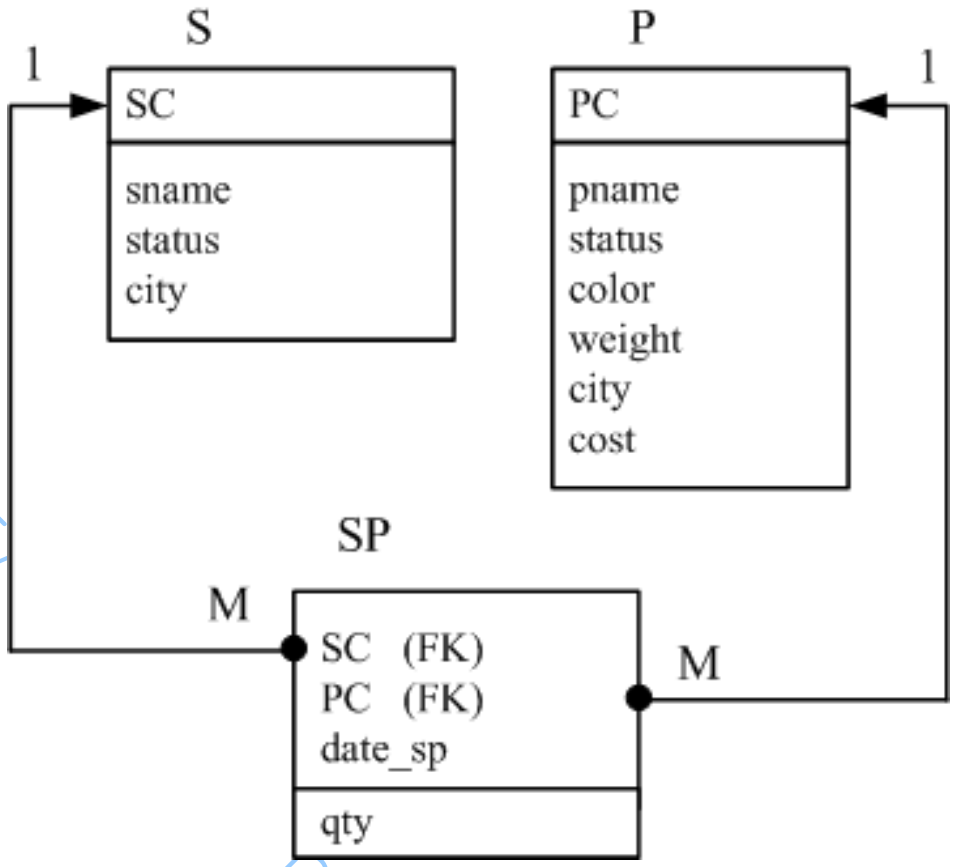
Способы объединения данных из нескольких таблиц:

1) с помощью условия объединения WHERE;

2) с помощью (горизонтальных) объединений JOIN;

3) с помощью использования подзапросов в запросе.

Все примеры будут рассмотрены на примере **базы данных поставщиков и деталей.**



2.8.5.1 Объединение данных из нескольких таблиц с помощью условия объединения WHERE

Одна из трудностей при выборке сразу из нескольких таблиц - это совпадение имён полей.

```
SELECT *
FROM S, SP
WHERE S.SC > SP.SC
```

Фактически, такой запрос позволяет избежать противоречий между одинаковыми именами полей. Однако, сложность не в этом, а в алгоритме работы подобного SQL-запроса.

Пример: Выдать детали и поставщиков, находящихся в одном городе (не обязательно, что поставщик поставляет именно эту деталь)

```
SELECT   SNAME, PNAME, S.CITY
FROM   S, P
WHERE   S.CITY=P.CITY;
```

Одинаковые имена полей предваряются именами соответствующих таблиц

Пример: Выдать поставщиков, которые поставляют детали не из своего города (с данными о деталях) (*Объединение более, чем двух таблиц!*)

```
SELECT   SC, SNAME, S.CITY,
                 PC, PNAME, P.CITY
FROM   S, P, SP
WHERE  (S.SC=SP.SC) AND
                 (P.PC=SP.PC) AND
                 (S.CITY<>P.CITY);
```

Скобки лишними не бывают!

2.8.5.2 Объединение данных из нескольких таблиц с помощью SQL объединений JOIN

- **INNER JOIN**
- **OUTER JOIN**
- **CROSS JOIN**

- внутреннее объединение **INNER JOIN** (синоним JOIN, ключевое слово INNER можно опустить)

Пример: Выдать детали и поставщиков, находящихся в одном городе (не обязательно, что поставщик поставляет именно эту деталь)

```
SELECT   SNAME, PNAME, S.CITY
FROM   S   JOIN   P
                 ON   S.CITY=P.CITY;
```

Выбираются только совпадающие данные из объединяемых таблиц

Пример: Выдать поставщиков, которые поставляют детали не из своего города (с данными о деталях) (*Объединение более, чем двух таблиц!*)

```
SELECT   SC, SNAME, S.CITY,
                 PC, PNAME, P.CITY
```

```
FROM   (S   JOIN  SP  ON   S.SC=SP.SC)
              JOIN   P   ON   P.PC=SP.PC
WHERE   S.CITY<>P.CITY;
```

- внешнее объединение - **OUTER JOIN**

Такое объединение вернет данные из обеих таблиц (совпадающие по условию объединения) ПЛЮС дополнит выборку оставшимися данными из внешней таблицы, которые по условию не подходят, заполнив недостающие данные значением NULL

LEFT OUTER JOIN или LEFT JOIN

Возвращаются все строки левой_таблицы.

Данными правой_таблицы дополняются только те строки левой_таблицы, для которых выполняются условия_соединения.

Для недостающих данных вместо строк правой_таблицы вставляются NULL-значения.

RIGHT OUTER JOIN или RIGHT JOIN

Работает одинаково, разница заключается в том что RIGHT - указывает что "внешней" таблицей будет находящаяся справа.

FULL OUTER JOIN или FULL JOIN

Возвращаются все строки левой_таблицы и правой_таблицы. Если для строк левой_таблицы и правой_таблицы выполняются условия_соединения, то они объединяются в одну строку. Для строк, для которых не выполняются условия_соединения, NULL-значения вставляются на место левой_таблицы, либо на место правой_таблицы, в зависимости от того данных какой таблицы в строке не имеется.

Пример: Выдать поставщиков и коды поставленных ими деталей с указанием даты поставки (причем указывать поставщиков, не поставивших ничего)

```
SELECT   SNAME, PC, DATE_SP
FROM   S   LEFT OUTER JOIN   SP
                ON   S.SC=SP.SC;
```

Результирующая таблица:

	PC	DATE_SP
Smith	P1	21.01.10
NAME	P2	21.01.10
Smith	P1	30.03.10
Smith	P3	25.03.10
…	…	…
Adams	NULL	NULL

Пример:

```
SELECT   SNAME, PC, DATE_SP
FROM   S   LEFT OUTER JOIN   SP
                ON   S.SC=SP.SC
WHERE S.SC IS NULL;
```

Результирующая таблица:

NAME	PC	DATE_SP
Adams	NULL	NULL

- **CROSS JOIN** - возвращает перекрестное (декартово) объединение двух таблиц.

Результатом будет выборка всех записей первой таблицы объединенная с каждой строкой второй таблицы.

Важным моментом является то, что для кросса не нужно указывать условие объединения.

2.8. Объединение данных из нескольких таблиц Объединение таблиц самих с собой

ПСЕВДОНИМЫ - временные имена таблиц и полей (переменные диапазона, переменные корреляции).

Чтобы объединить таблицу саму с собой, все повторяемые имена столбца, заполняются префиксами имени таблицы. Чтобы ссылаться к этим столбцам внутри запроса, нужно иметь два различных имени для этой таблицы - псевдонима. Они определяются в предложении **FROM** запроса (указывается имя таблицы, оставляется пробел, и затем набирается псевдоним для нее).

Пример: Определить пары поставщиков с одинаковым статусом (запрос с WHERE).

```
SELECT  ONE.SC, TWO.SC, ONE.STATUS
FROM   S  ONE,   S   TWO
WHERE   (ONE.STATUS=TWO.STATUS)
            AND    (ONE.SC<TWO.SC);
```

Второе условие позволяет исключить повторения пар (S1,S2 и S2,S1) и вывод пар типа (S1, S1).

Пример: Определить пары поставщиков с одинаковым статусом (запрос с JOIN).

```
SELECT  ONE.SC, TWO.SC,
            ONE.STATUS
FROM   S  ONE JOIN  S   TWO
          ON   ONE.STATUS=TWO.STATUS
WHERE   ONE.SC<TWO.SC;
```

Замечания:

- можно использовать любое (без излишеств) число псевдонимов для одной таблицы в запросе;
- можно использовать псевдонимы для создания альтернативных имен и для нескольких таблиц в команде (например, если таблицы имеют очень длинные и сложные имена);
- не всегда обязательно использовать каждый псевдоним или таблицу которые упомянуты в предложении FROM запроса, в предложении SELECT, т.к. иногда предложение или таблица становятся запрашиваемыми исключительно потому что они могут вызываться в предикате запроса.

2.8. Объединение данных из нескольких таблиц

с помощью использования подзапросов в запросе

Часто невозможно решить поставленную задачу путем одного запроса.

Это особенно актуально, когда при использовании условия поиска в предложении **WHERE** значение, с которым надо сравнивать, заранее не определено и должно быть вычислено в момент выполнения оператора **SELECT**.

В таком случае приходят на помощь законченные операторы **SELECT**, **ВНЕДРЕННЫЕ** в тело другого оператора **SELECT**.

Внутренний подзапрос представляет собой также оператор SELECT, а кодирование его предложений подчиняется тем же правилам, что и основного оператора SELECT.

Внешний оператор SELECT использует результат выполнения внутреннего оператора для определения содержания окончательного результата всей операции.

Внутренние запросы могут быть помещены непосредственно после оператора сравнения (=, <, >, <=, >=, <>) в предложения WHERE и HAVING внешнего оператора SELECT – они получают название **подзапросов** или **вложенных запросов**.

ПОДЗАПРОС – это инструмент создания временной таблицы, содержимое которой извлекается и обрабатывается внешним оператором. Текст подзапроса должен быть заключен в скобки.

Внутренние операторы **SELECT** могут применяться также в операторах **INSERT**, **UPDATE** и **DELETE**.

Синтаксис подзапроса:

WHERE

< имя / константа > < оператор>

< подзапрос >

К подзапросам применяются следующие
правила и ограничения:

- фраза **ORDER BY** не используется, хотя и может присутствовать во внешнем подзапросе;
- список в предложении **SELECT** состоит из имен отдельных столбцов или составленных из них выражений – за исключением случая, когда в подзапросе присутствует ключевое слово **EXISTS**;
- по умолчанию имена столбцов в подзапросе относятся к таблице, имя которой указано в предложении FROM. Однако допускается ссылка и на столбцы таблицы, указанной во фразе FROM внешнего запроса, для чего применяются квалифицированные имена столбцов (т.е. с указанием таблицы);
- если подзапрос является одним из двух операндов, участвующих в операции сравнения, то запрос должен указываться в правой части этой операции.

Существует **два типа подзапросов**:

- **скалярный подзапрос** возвращает единственное значение (в принципе, он может использоваться везде, где требуется указать единственное значение);
- **табличный подзапрос** возвращает множество значений, т.е. значения одного или нескольких столбцов таблицы, размещенные в более чем одной строке (он возможен везде, где допускается наличие таблицы).

Пример: Вывести данные о всех поставках поставщика Smith.

```
SELECT  *
FROM   SP
WHERE   SC = SC поставщика SMITH;
```

Возникает вопрос: "Какой SC у поставщика Smith?

Для того, чтобы это узнать нужно воспользоваться вложенным запросом.

```
SELECT  *
FROM   SP
WHERE   SC = (SELECT   SC
                FROM    S
```

WHERE SNAME='SMITH');

Не нужно помнить коды поставщиков!

Сначала SQL выполнит внутренний запрос, который расположен в скобках и результат подставит во внешний запрос.

Но должно выполнятся **два условия**:

- у внутреннего запроса, в качестве результата должен быть **только один столбец** (это значит, что нельзя написать во внутреннем запросе **SELECT *** , а можно только **SELECT ИмяОдногоПоля**, причем имя должно быть только одно и тип его должен совпадать с типом сравниваемого значения).
- результатом внутреннего запроса, должна быть **только одна строка**. Если внутренний запрос вернёт несколько строк или вообще ничего не вернёт, то могут возникнуть проблемы. Для большей надёжности с подзапросом используется оператор **DISTINCT**.

Пример: Вывести данные о всех поставках, количество деталей в которых больше среднего количества деталей в поставках, сделанных 10.10.2010 г.

Агрегатные функции во вложенных подзапросах: подзапрос должен возвращать одно значение!

```
SELECT  *
FROM    SP
WHERE  QTY > (SELECT  AVG (QTY)
                  FROM    SP
                  WHERE  DATE_SP ='2010-10-10');
```

Пример: Извлечь данные о деталях, у которых статус в 2 раза больше, чем статус поставщика с именем Smith.

Использование выражения в подзапросе.

Не нужно помнить статус поставщика!

```
SELECT  *
```

```
FROM   P
WHERE   STATUS =
                (SELECT   STATUS*2
                 FROM   S
                 WHERE   SNAME='SMITH');
```

Пример: Вывести для каждого значения статуса количество поставщиков с этим статусом. Оставить только те группы, у которых статус больше среднего статуса поставщиков из Парижа.

Подзапросы в условии HAVING

```
SELECT   STATUS,  COUNT (SC) FROM    S
GROUP  BY   STATUS
HAVING    STATUS  > (SELECT  AVG (STATUS)
                        FROM   S WHERE  CITY = 'PARIS');
```

Во всех предыдущих запросах подзапрос выполнялся только один раз!

ЗАПРОСЫ С КОРРЕЛИРОВАННЫМИ ВЛОЖЕННЫМИ ПОДЗАПРОСАМИ

В операторе SELECT из внутреннего подзапроса можно ссылаться на столбцы внешнего запроса, указанного во фразе SELECT. Такой подзапрос выполняется для каждой строки таблицы, определяя условие ее вхождения в формируемый результирующий набор.

Пример: Найти все детали. Поставленные 10.10.2010 г.

```
SELECT  *
FROM  P
WHERE   '2010-10-10'  IN
                (SELECT  DATE_SP
                 FROM   SP
                 WHERE  P.PC = SP.PC );
```

Пример: Найти все детали. Поставленные 10.10.2010 г.

(*другая формулировка того же запроса*)

```
SELECT  DISTINCT  P.PC, PNAME,
                  DATE_SP
FROM  P, SP
WHERE   (P.PC=SP.PC)  AND
                  (DATE_SP ='2010-10-10');
```

Если не использовать DISTINCT, деталь может быть выведена несколько раз (для каждого поставщика, поставившего ее указанного числа)

Пример: Вывести названия и номера всех деталей, которые имеют более одного поставщика.

```
SELECT  PC, PNAME
FROM   P
WHERE   1 <
    (SELECT  COUNT (DISTINCT  SC)
     FROM   SP
     WHERE   PC=P.PC);
```

Если перед именем поля не указывается ссылка на таблицу, то данное поле относится к таблице текущего подзапроса.

Чтобы обращаться из внутреннего запроса к внешнему, необходимо произвести **соотнесение таблицы со своей копией** для чего используются слова outer и inner - псевдонимы, которые назначаются таблицам

Пример: Найти все поставки со значением количества деталей в поставке выше среднего значения для этой детали.

```
SELECT  *
FROM   SP   OUTER
WHERE  QTY >=
          (SELECT  AVG (QTY)
           FROM   SP   INNER
```

WHERE INNER.PC = OUTER.PC);

Аналитическая обработка данных в запросе

Такой запрос будет выполнятся по следующему **алгоритму:**

- Выбрать строку из таблицы SP во внешнем запросе.
  Это будет текущая строка-кандидат.
- Сохранить значения из этой строки-кандидата в псевдониме с именем outer.
- Выполнить подзапрос. Везде, где псевдоним данный для внешнего запроса найден (в этом случае "outer"), использовать значение для текущей строки-кандидата INNER.PC=OUTER.PC. Использование значения из строки-кандидата внешнего запроса в подзапросе называется - внешней ссылкой.
- Оценить предикат внешнего запроса на основе результатов подзапроса выполняемого в предыдущем шаге. Он определяет - выбирается ли строка-кандидат для вывода.

Подзапросы, возвращающие множество значений

Во многих случаях значение, подлежащее сравнению в предложениях **WHERE** или **HAVING**, представляет собой не одно, а несколько значений.

Вложенные подзапросы генерируют непоименованное промежуточное отношение, временную таблицу.

Оно может использоваться только в том месте, где появляется в подзапросе.

К такому отношению невозможно обратиться по имени из какого-либо другого места запроса.

<u>Специальные операторы:</u>

•{ WHERE | HAVING } выражение [NOT] IN (подзапрос);

•{WHERE | HAVING } [NOT] EXISTS (подзапрос);

•{ WHERE | HAVING } выражение оператор_сравнения ANY (подзапрос);

•{ WHERE | HAVING } выражение оператор_сравнения ALL (подзапрос);

Пример: Вывести данные о всех поставках поставщиков из Лондона.

```
SELECT  *
FROM    SP
WHERE    SC   IN
              (SELECT  SC  FROM   S
               WHERE   CITY='LONDON');
```

Оператор IN в данном случае принимает подзапрос в качестве аргумента.

Пример: Вывести данные о всех поставках поставщиков из Лондона (другая формулировка того же запроса):

```
SELECT  DATE_SP, SP.SC, PC, QTY
FROM   S, SP
WHERE   (S.SC = SP.SC) AND
              (CITY = 'LONDON')
```

Пример: Найти всех поставщиков, совершивших хотя бы одну поставку с количеством деталей больше 200.

```
SELECT    SC
FROM    S
WHERE   EXISTS  (SELECT *
                          FROM    SP
                 WHERE   S.SC = SP.SC
                    AND   QTY > 200);
```

Пример: Найти всех поставщиков, поставивших более одной детали.

```
SELECT   DISTINCT   SC
FROM    SP   OUTER
```

```
WHERE   EXISTS
    (SELECT  *
     FROM    SP   INNER
     WHERE   INNER.SC = OUTER.SC
       AND  INNER.PC <> OUTER.PC);
```

Без DISINCT каждый поставщик будет выбран один раз для каждой своей детали

Пример: Вывести список поставщиков, за которыми числится только одна деталь.

```
SELECT   DISTINCT   SC
FROM    SP   OUTER
WHERE    NOT   EXISTS
    (SELECT  *
     FROM    SP   INNER
     WHERE   INNER.SC = OUTER.SC
       AND  INNER.PC <> OUTER.PC);
```

Пример: Найти всех поставщиков, поставляющих детали из своего города.

```
SELECT   *
FROM    S
WHERE  CITY = ANY (SELECT  CITY
                   FROM    P, SP
                   WHERE  P.PC = SP.PC
                     AND  S.SC = SP.SC);
```

Пример: Найти всех поставщиков, у которых статус выше, чем у любого поставщика из Лондона.

```
SELECT  *
```

```
FROM   S
WHERE   STATUS >
    ALL (SELECT    SATUS
            FROM    S
            WHERE   CITY= ‘LONDON’);
```