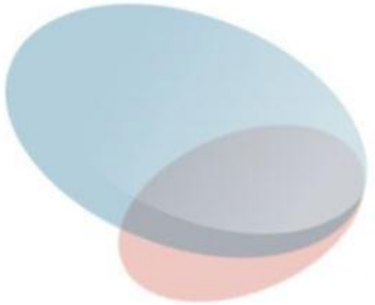

# СУБД
# Лекция 2
# Основы языка SQL

# Основы языка SQL

## II. Отбор данных из нескольких таблиц

**1) Естественное соединение** таблиц (**способ 1 - явное указание условий соединения**):

```
SELECT
    P.PNUM,
    P.PNAME,
    PD.DNUM,
    PD.VOLUME
  FROM P, PD
  WHERE P.PNUM = PD.PNUM;
```

| PNUM | PNAME | DNUM | VOLUME |
|---|---|---|---|
| 1 | Иванов | 1 | 100 |
| 1 | Иванов | 2 | 200 |
| 1 | Иванов | 3 | 300 |
| 2 | Петров | 1 | 150 |
| 2 | Петров | 2 | 250 |
| 3 | Сидоров | 1 | 1000 |

**2) Естественное соединение** таблиц (**способ 2** - ключевые слова **JOIN… USING**…):

```
SELECT P.PNUM, P.PNAME, PD.DNUM,PD.VOLUME
  FROM P JOIN PD USING  PNUM;
```

**Замечание**. Ключевое слово USING позволяет **явно указать, по каким из общих колонок** таблиц **будет производиться соединение**.

# Основы языка SQL

**3) Естественное соединение** таблиц (**способ 3** - ключевое слово **NATURAL JOIN**):

```
SELECT  P.PNUM, P.PNAME, PD.DNUM, PD.VOLUME
  FROM P NATURAL JOIN PD;
```

**Замечание.** В разделе FROM не указано, по каким полям производится соединение. NATURAL JOIN **автоматически соединяет по всем одинаковым полям в таблицах**.

**4) Естественное соединение трех таблиц**:

```
SELECT
    P.PNAME,
    D.DNAME,
    PD.VOLUME
  FROM
    P NATURAL JOIN PD NATURAL JOIN D;
```

| PNAME | DNAME | VOLUME |
|---|---|---|
| Иванов | Болт | 100 |
| Иванов | Гайка | 200 |
| Иванов | Винт | 300 |
| Петров | Болт | 150 |
| Петров | Гайка | 250 |
| Сидоров | Болт | 1000 |

# Основы языка SQL

**5) Прямое произведение** таблиц:

```
SELECT
    P.PNUM,
    P.PNAME,
    D.DNUM,
    D.DNAME
  FROM P, D;
```

| PNUM | PNAME | DNUM | DNAME |
|---|---|---|---|
| 1 | Иванов | 1 | Болт |
| 1 | Иванов | 2 | Гайка |
| 1 | Иванов | 3 | Винт |
| 2 | Петров | 1 | Болт |
| 2 | Петров | 2 | Гайка |
| 2 | Петров | 3 | Винт |
| 3 | Сидоров | 1 | Болт |
| 3 | Сидоров | 2 | Гайка |
| 3 | Сидоров | 3 | Винт |

**6) Соединение таблиц по произвольному условию**. Ответ на вопрос "какие поставщики имеют право поставлять какие детали?" дает запрос:

```
SELECT P.PNUM,P.PNAME,P.PSTATUS,
    D.DNUM,D.DNAME,D.DSTATUS
  FROM P, D
  WHERE P.PSTATUS >= D.DSTATUS;
```

# Основы языка SQL

## III. Использование имен корреляции (алиасов, псевдонимов)

Существуют запросы, в которых **таблица соединяется сама с собой**, или **одна таблица соединяется дважды с другой таблицей**. В этих случаях используются **имена корреляции (алиасы, псевдонимы)**, которые позволяют **различать соединяемые копии таблиц**.

Имена корреляции вводятся **в разделе FROM** и **идут через пробел после имени таблицы**.

Имена корреляции должны использоваться **в качестве префикса перед именем столбца и отделяются от имени столбца точкой**.

Если в запросе указываются **одни и те же поля из разных экземпляров одной таблицы**, они **должны быть переименованы** для устранения неоднозначности в именованиях колонок результирующей таблицы.

Определение имени корреляции действует только во время выполнения запроса.

# Основы языка SQL

**1)** Отобрать все пары поставщиков таким образом, чтобы **первый поставщик в паре имел статус, больший статуса второго поставщика**:

```
SELECT
   P1.PNAME AS PNAME1,
   P1.PSTATUS AS PSTATUS1,
   P2.PNAME AS PNAME2,
   P2.PSTATUS AS PSTATUS2
 FROM
   P P1, P P2
 WHERE P1.PSTATUS1 > P2.PSTATUS2;
```

| PNAME1 | PSTATUS1 | PNAME2 | PSTATUS2 |
|---|---|---|---|
| Иванов | 4 | Петров | 1 |
| Иванов | 4 | Сидоров | 2 |
| Сидоров | 2 | Петров | 1 |

# Основы языка SQL

**2)** Пусть **некоторые поставщики** (назовем их контрагенты) **могут выступать** как в качестве **поставщиков деталей**, так и в качестве **получателей**.

| Номер контрагента NUM | Наименование контрагента NAME |
|---|---|
| 1 | Иванов |
| 2 | Петров |
| 3 | Сидоров |

Отношение CONTRAGENTS

| Номер детали DNUM | Наименование детали DNAME |
|---|---|
| 1 | Болт |
| 2 | Гайка |
| 3 | Винт |

Отношение DETAILS (Детали)

| Номер поставщика PNUM | Номер получателя CNUM | Номер детали DNUM | Поставляемое количество VOLUME |
|---|---|---|---|
| 1 | 2 | 1 | 100 |
| 1 | 3 | 2 | 200 |
| 1 | 3 | 3 | 300 |
| 2 | 3 | 1 | 150 |
| 2 | 3 | 2 | 250 |
| 3 | 1 | 1 | 1000 |

Отношение CD (Поставки)

# Основы языка SQL

**Запрос** "кто кому какие детали в каком количестве поставляет".

```
SELECT
   P.NAME AS PNAME,
   C.NAME AS CNAME,
   DETAILS.DNAME,
   CD.VOLUME
 FROM
   CONTRAGENTS P,
   CONTRAGENTS C,
   DETAILS,
   CD
 WHERE
   P.NUM = CD.PNUM AND
   C.NUM = CD.CNUM AND
   D.DNUM = CD.DNUM;
```

| Наименование поставщика PNAME | Наименование получателя CNAME | Наименование детали DNAME | Поставляемое количество VOLUME |
|---|---|---|---|
| Иванов | Петров | Болт | 100 |
| Иванов | Сидоров | Гайка | 200 |
| Иванов | Сидоров | Винт | 300 |
| Петров | Сидоров | Болт | 150 |
| Петров | Сидоров | Гайка | 250 |
| Сидоров | Иванов | Болт | 1000 |

**Замечание**. Этот запрос может быть выражен большим количеством способов.

# Основы языка SQL

## IV. Использование в запросах агрегатных функций

**1)** Получить **общее количество** поставщиков (ключевое слово **COUNT**):

```
SELECT COUNT(*) AS N
  FROM P;
```

| N |
|---|
| 3 |

**2)** Получить **общее, максимальное, минимальное и среднее количества** поставляемых деталей (ключевые слова **SUM, MAX, MIN, AVG**):

```
SELECT
    SUM(PD.VOLUME) AS SM,
    MAX(PD.VOLUME) AS MX,
    MIN(PD.VOLUME) AS MN,
    AVG(PD.VOLUME) AS AV
  FROM PD;
```

| SM | MX | MN | AV |
|---|---|---|---|
| 2000 | 1000 | 100 | 333.33333333 |

# Основы языка SQL

## V. Использование агрегатных функций с группировками

**1) Для каждой детали** получить **суммарное поставляемое количество** (ключевые слова **GROUP BY**…):

```
SELECT
    PD.DNUM,
    SUM(PD.VOLUME) AS SM
    FROM PD
    GROUP BY PD.DNUM;
```

| DNUM | SM |
|---|---|
| 1 | 1250 |
| 2 | 450 |
| 3 | 300 |

**Замечание**. Этот запрос будет выполняться следующим образом.

Сначала строки исходной таблицы будут сгруппированы так, чтобы в **каждую группу попали строки с одинаковыми значениями DNUM**.

Потом **внутри каждой группы будет просуммировано поле VOLUME.**

**От каждой группы** в результирующую таблицу будет включена **одна строка.**

# Основы языка SQL

**Замечание**. **В списке отбираемых полей** оператора SELECT, содержащего раздел GROUP BY **можно включать** только агрегатные функции и **поля, которые входят в условие группировки**. Следующий запрос выдаст синтаксическую ошибку:

```
SELECT
   PD.PNUM,    PD.DNUM,
   SUM(PD.VOLUME) AS SM
   FROM PD
   GROUP BY PD.DNUM;
```

**Некоторые диалекты SQL не считают это за ошибку**. Запрос будет выполнен, **но предсказать, какие значения будут внесены в поле PNUM в результирующей таблице, невозможно** (в каждую полученную **группу** строк может входить несколько строк с **различными** значениями поля PNUM).

# Основы языка SQL

**2)** Получить номера деталей, **суммарное поставляемое количество которых превосходит 400** (ключевое слово **HAVING**…**- условие отбора групп**).

```
SELECT
    PD.DNUM, SUM(PD.VOLUME) AS SM
    FROM PD
    GROUP BY PD.DNUM
HAVING SUM(PD.VOLUME) > 400;
```

| DNUM | SM |
|---|---|
| 1 | 1250 |
| 2 | 450 |

**Замечание.** Условие, что суммарное поставляемое количество должно быть больше 400, не может быть сформулировано **в разделе WHERE**, т.к. в этом разделе **нельзя использовать агрегатные функции**.

**Условия, использующие агрегатные функции, должны быть размещены в специальном разделе HAVING**.

# Основы языка SQL

**Замечание.** В одном запросе могут встретиться как **условия отбора строк** в разделе **WHERE**, так и **условия отбора групп** в разделе **HAVING**. Условия отбора групп нельзя перенести из раздела HAVING в раздел WHERE. Аналогично и условия отбора строк нельзя перенести из раздела WHERE в раздел HAVING, за исключением условий, включающих поля из списка группировки GROUP BY.

**Замечание**. В MySQL допускается в HAVING использовать вместо агрегатной функции ее алиас.

```
SELECT
    PD.DNUM,
    SUM(PD.VOLUME) AS SM
    FROM PD
    GROUP BY PD.DNUM
    HAVING SM > 400;
```

# Основы языка SQL

## VI. Использование подзапросов

Удобным средством, позволяющим формулировать запросы более понятным образом, является **возможность использования подзапросов, вложенных в основной запрос**.

**1)** Получить список поставщиков, **статус** которых **меньше максимального** статуса в таблице поставщиков (**сравнение с подзапросом**).

```
SELECT *
  FROM P
  WHERE P.STATUS <
    (SELECT MAX(P.STATUS)
       FROM P);
```

# Основы языка SQL

**Замечание**. **Т.к. поле P.STATUS сравнивается** с результатом подзапроса, то **подзапрос должен** быть сформулирован так, чтобы **возвращать таблицу, состоящую ровно из одной строки и одной колонки**.

**Замечание**. Результат выполнения запроса будет эквивалентен результату следующей последовательности действий:

- **Выполнить один раз вложенный подзапрос** и получить максимальное значение статуса.
- **Просканировать таблицу поставщиков P,** каждый раз сравнивая значение статуса поставщика с результатом подзапроса, и **отобрать только те строки, в которых статус меньше максимального**.

# Основы языка SQL

**2) Использование предиката IN**. Получить список поставщиков, поставляющих деталь номер 2.

```
SELECT *
  FROM P
  WHERE P.PNUM IN
    (SELECT DISTINCT PD.PNUM
        FROM PD
        WHERE PD.DNUM = 2);
```

**Замечание**. В данном случае вложенный **подзапрос может** возвращать таблицу, **содержащую несколько строк (но один столбец)**.

# Основы языка SQL

**Замечание**. Результат выполнения запроса будет эквивалентен результату следующей последовательности действий:

- **Выполнить один раз вложенный подзапрос** и получить список номеров поставщиков, поставляющих деталь номер 2.
- **Просканировать таблицу поставщиков P**, каждый раз **проверяя, содержится ли номер поставщика в результате подзапроса**.

# Основы языка SQL

**3) Использование подзапросов в качестве вычисляемых полей**. Для каждого поставщика подсчитать количество поставляемых им видов деталей.

```
SELECT P.PNUM, P.PNAME,
    (SELECT COUNT(*)
        FROM PD
        WHERE PD.PNUM=P.PNUM) AS NUMBER0FDET
  FROM P
  ORDER BY P.PNUM;
```

**Замечание. Вложенный запрос выполняется многократно для каждого значения P.PNUM** из таблицы P. Для каждого значения P.PNUM из таблицы P в таблице PD выбираются строки с текущим значением PNUM из внешнего запроса и подсчитывается их количество.