Базы данных

Язык запросов SQL.
Команда SELECT.
Дополнительные возможности

Самосоединение

В команде SELECT можно обратиться к одной и той же таблице несколько раз. При этом для каждой таблицы необходимо задать свой алиас, чтобы можно было обращаться к полям этих таблиц. Система будет выполнять такой запрос на основе декартова произведения таблиц, поэтому необходимо указывать условие соединения. А для того чтобы исключить соединение записи таблицы с самой собой в запросе на самосоединение необходимо также указывать условие типа "не равно" (<>, >, <).

Пример использования самосоединения:

Вывести список детей сотрудников, у которых есть младшие братья или сёстры:

```
SELECT e.name, c1.name AS child1, c1.born AS born1,
    c2.name AS child2, c2.born AS born2
FROM children c1, children c2, emp e
    WHERE c1.tabno=e.tabno          -- первое условие соединения
    AND c1.tabno=c2.tabno  -- второе условие соединения
    AND c1.born<c2.born    -- условие исключения
ORDER BY 1, 3;
```

Результат самосоединения

TabNo	**Name**	Born	Sex
988	Вадим	03.05.1995	м
110	Ольга	18.07.2001	ж
023	Илья	19.02.1987	м
023	Анна	26.12.1989	ж
909	Инна	25.01.2008	ж
909	Роман	21.11.2006	м
909	Антон	06.03.2009	м

NAME	CHILD1	BORN1	CHILD2	BORN2
Малова Л.А.	Илья	19.02.1987	Анна	26.12.1989
Серова Т.В.	Роман	21.11.2006	Инна	25.01.2008
Серова Т.В.	Роман	21.11.2006	Антон	06.03.2009
Серова Т.В.	Инна	25.01.2008	Антон	06.03.2009

Подзапросы

Подзапрос – это запрос SELECT, расположенный внутри другой команды.

Подзапросы можно разделить на следующие группы в зависимости от возвращаемых результатов:

✓**скалярные** – запросы, возвращающие единственное значение (начинаются с немодифицированного оператора сравнения);

✓**векторные** – запросы, возвращающие от 0 до нескольких элементов (начинаются с оператора *IN* или модифицированного оператора сравнения);

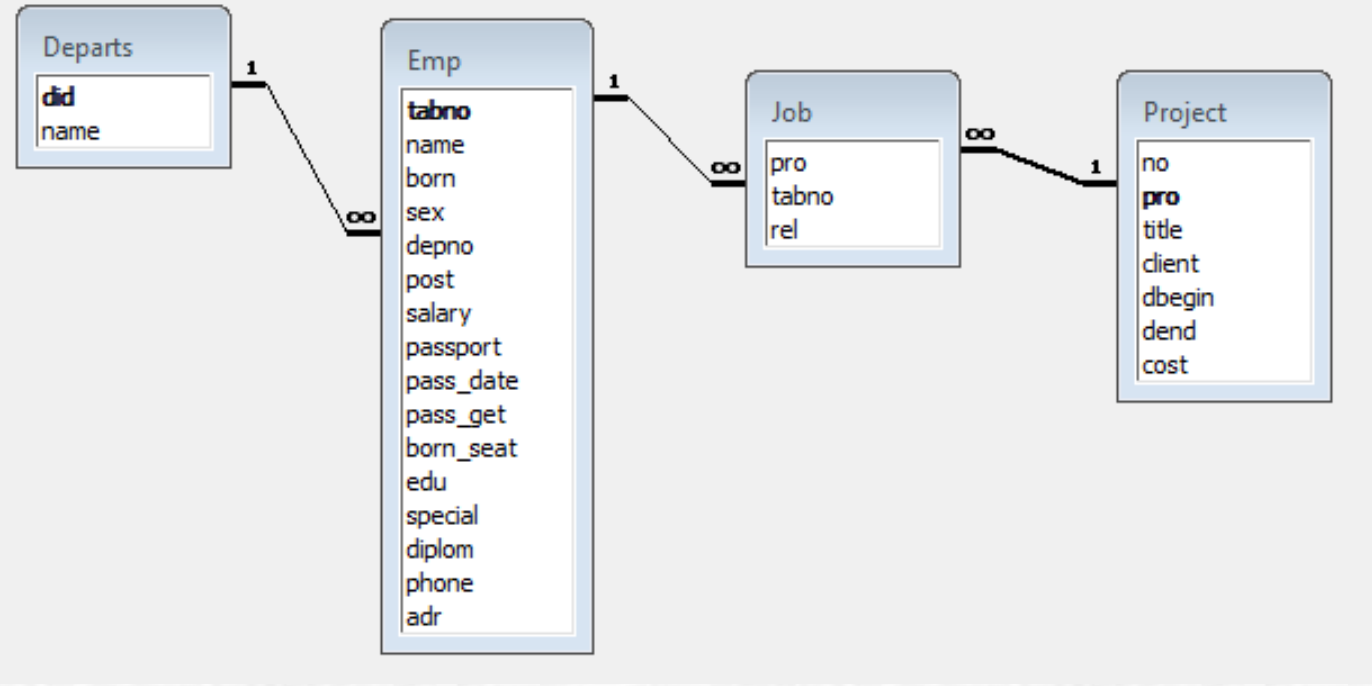
✓**табличные** – запросы, возвращающие таблицу (обычно, запросы на существование, начинаются с оператора *EXISTS*).

Подзапросы бывают:

✓**некоррелированные** – не содержат ссылки на запрос верхнего уровня; вычисляются один раз для запроса верхнего уровня;

✓**коррелированные –** содержат условия, зависящие от значений полей в основном запросе; вычисляются для каждой строки запроса верхнего уровня.

Пример БД: проектная организация



Departs – отделы,

Emp – сотрудники,

Project – проекты,

Job – участие в проектах.

Данные таблицы Emp (сотрудники)

TabNo	*DepNo*	Name	Post	Salary	Born	Phone
988	*1*	Рюмин В.П.	начальник отдела	48500.0	01.02.1970	115-26-12
909	*1*	Серова Т.В.	вед. программист	48500.0	20.10.1981	115-91-19
829	*1*	Дурова А.В.	экономист	43500.0	03.10.1978	115-26-12
819	*1*	Тамм Л.В.	экономист	43500.0	13.11.1985	115-91-19
100	*2*	Волков Л.Д.	программист	46500.0	16.10.1982	null
110	*2*	Буров Г.О.	бухгалтер	42880.0	22.05.1975	115-46-32
023	*2*	Малова Л.А.	гл. бухгалтер	59240.0	24.11.1954	114-24-55
130	*2*	Лукина Н.Н.	бухгалтер	42880.0	12.07.1979	115-46-32
034	*3*	Перова К.В.	делопроизводитель	32000.0	24.04.1988	null
002	*3*	Сухова К.А.	начальник отдела	48500.0	08.06.1948	115-12-69
056	*5*	Павлов А.А.	директор	80000.0	05.05.1968	115-33-44
087	*5*	Котова И.М.	секретарь	35000.0	16.09.1990	115-33-65
088	*5*	Кроль А.П.	зам.директора	70000.0	18.04.1974	115-33-01

Расположение подзапросов в командах DML

В команде **INSERT**:

- Вместо **VALUES**, например, добавление данных из одной таблицы в другую:

```
insert into emp select * from new_emp;
```

В команде **UPDATE**:

- в части **WHERE** для вычисления условий, например, повышение зарплаты на 10% всем участникам проектов:

```
update emp set salary = salary*1.1
        where tabNo IN (select tabNo from job);
```

- в части **SET** для вычисления значений полей, например, повышение зарплаты на 10% за каждое участие сотрудника в проекте:

```
update emp e set salary = salary*(1+(select count(*)/10 from job j
                                    where j.tabNo = e.tabNo) );
```

В команде **DELETE**:

- в части **WHERE** для вычисления условий, например, удаление сведений об участии в закончившихся проектах:

```
delete from job
        where pro IN (select pro from project where dend < sysdate);
```

Расположение подзапросов в команде select

- Чаще всего подзапрос располагается в части **WHERE.**

Пример 1. Вывести список сотрудников, у которых зарплата выше, чем средняя по предприятию:

```
select * from emp
    where salary > (select avg(salary) from emp);
```

DEPNO	NAME	POST	SALARY
2	Малова Л.А.	гл. бухгалтер	59240
5	Павлов А.А.	директор	80000
5	Кроль А.П.	зам. директора	70000

Пример 2. Вывести список сотрудников, у которых зарплата выше, чем средняя по каждому отделу предприятия:

```
select * from emp
    where salary > ALL (select avg(salary) from emp group by depno);
```

Примеры использования подзапросов в части WHERE

Выдать список сотрудников, имеющих детей:

а) с помощью операции соединения таблиц:

```
SELECT  distinct  e.*
    FROM  emp e, children c
    WHERE e.tabno=c.tabno;
```

б) с помощью некоррелированного векторного подзапроса:

```
SELECT  *
    FROM  emp
    WHERE  tabno  IN  (SELECT  tabno FROM children);
```

в) с помощью коррелированного табличного подзапроса:

```
SELECT  *
    FROM  emp  e
    WHERE  EXISTS (SELECT * FROM children c
                   WHERE e.tabno=c.tabno);
```

Расположение подзапросов в команде select

- Подзапрос в части **FROM.**

Например, выведем список сотрудников, у которых зарплата выше, чем средняя в отделе, в котором работает данный сотрудник, через коррелированный подзапрос:

```
select * from emp e
    where salary > (select avg(salary) from emp m
                    where m.depno = e.depno);
```

Это работает долго, т.к. коррелированный подзапрос вычисляется для каждой строки основного запроса. Можно ускорить выполнение данного запроса:

```
select *
    from emp e,
    (select depno, avg(salary) sal
        from emp
        group by depno) m          -- подзапрос вычисляется 1 раз
    where m.depno = e.depno
        and salary > sal;
```

Расположение подзапросов в команде select

- Подзапрос в части **HAVING.**

  Например, выведем список отделов, в которых средняя зарплата ниже, чем средняя по предприятию:

```
select depno, avg(salary) sal
    from emp
    group by depno
    having avg(salary) < (select avg(salary) from emp);
```

- Подзапрос в части **SELECT.**

  Например, выведем список сотрудников с указанием количества проектов, в которых они участвуют:

```
select depno, name,
    (select count(*) from job j where j.tabno = e.tabno) cnt
from emp e;
```

Этот запрос выведет даже тех сотрудников, которые не участвуют в проектах (для них **cnt** будет равен 0).

Представления

Представление (view, обзор) – это хранимый запрос, создаваемый на основе команды *SELECT*. Представление реально не содержит данных. Запрос, определяющий представление, выполняется тогда, когда к представлению происходит обращение с другим запросом, например, SELECT, UPDATE и т.д.

Назначение представлений:

- Хранение сложных запросов.
- Представление данных в виде, удобном пользователю.
- Сокрытие конфиденциальной информации.
- Предоставление дифференцированного доступа к данным.

Создание представления выполняется командой **CREATE VIEW**:

CREATE [OR REPLACE] VIEW <имя представления>
[(<список имён столбцов>)]
AS <запрос> [WITH CHECK OPTION];

Запрос (команда SELECT), на основании которого создаётся представление, называется **определяющим запросом**, а таблицы, к которым происходит обращение в определяющем запросе – **базовыми таблицами.**

Определяющий запрос по стандарту SQL не может включать предложение ORDER BY.

Представления: пример

Создать представление "Сотрудники с детьми" (для удобного представления данных о детях сотрудников):

```
CREATE VIEW  emp_child(depno, name, child, sex, born)
      AS  SELECT  e.depno,  e.name,  c.name,  c.sex,  c.born
      FROM  emp  e,  children  c
      WHERE  e.tabno = c.tabno;
SELECT * FROM emp_child;
```

DEPNO	NAME	CHILD	SEX	BORN
2	Буров Г.О.	Ольга	ж	18.07.2001
2	Малова Л.А.	Илья	м	19.02.1987
2	Малова Л.А.	Анна	ж	26.12.1989
…	…	…	…	…
1	Серова Т.В.	Антон	м	06.03.2009

Представления: пример

Создать представление "Сотрудники 2-го отдела" (для предоставления полного доступа к данным о сотрудниках 2-го отдела начальнику этого отдела):

```
CREATE  VIEW   emp2
      AS  SELECT  *
      FROM emp
      WHERE  depno = 2 WITH CHECK OPTION;
SELECT  *  FROM  emp2;
```

TABNO	DEPNO	NAME	POST	SALARY	BORN	PHONE
110	2	Буров Г.О.	бухгалтер	42880	22.05.75	115-46-32
100	2	Волков Л.Д.	программист	46500	16.10.82	
130	2	Лукина Н.Н.	бухгалтер	42880	12.07.79	115-46-32
023	2	Малова Л.А.	гл. бухгалтер	59240	24.11.54	114-24-55

Представления: примеры

Создать представление "Сотрудники" (без данных о зарплате, для сокрытия конфиденциальной информации):

```
CREATE  VIEW   employees
      AS  SELECT  tabno, depno, name, post, born, phone
      FROM emp;
```

Создать представление "Статистика по проектам" (для хранения сложных запросов): название проекта, ФИО руководителя, количество исполнителей, количество консультантов.

```
CREATE  VIEW   pro_stat
      AS  SELECT  title, e.name,
          (select count(*) from job j where j.pro=p.pro and rel='исполнитель') jobs,
          (select count(*) from job j where j.pro=p.pro and rel='консультант') consult
      FROM emp e, project p, job j
          where e.tabno=j.tabno and j.pro=p.pro
          and j.rel='руководитель';
```

Обновляемые представления

Представление может быть обновляемым и не обновляемым. Обновляемым является представление, при обращении к которому можно обновить базовую таблицу.

Пример обновления базовой таблицы ***emp*** через представление ***emp2***:

```
UPDATE  emp2
SET  salary = 48000
WHERE  tabno = '100';
```

Изменения будут произведены в базовой таблице и отразятся в представлении.

```
SELECT  *  FROM  emp2;
```

TABNO	DEPNO	NAME	POST	SALARY	BORN	PHONE
110	2	Буров Г.О.	бухгалтер	42880	22.05.75	115-46-32
100	2	Волков Л.Д.	программист	**48000**	16.10.82	
130	2	Лукина Н.Н.	бухгалтер	42880	12.07.79	115-46-32
023	2	Малова Л.А.	гл. бухгалтер	59240	24.11.54	114-24-55

Обновляемые представления

Вносимые изменения могут выйти за рамки определяющего запроса и поэтому не будут видны через представление. Если необходимо защитить данные от такого вмешательства, то нужно в команде создания представления указать ключевые слова **WITH CHECK OPTION**: тогда система отвергнет изменения, выходящие за рамки определяющего запроса.

По стандарту SQL-2 представление не является обновляемым, если определяющий запрос:

- содержит ключевое слово *DISTINCT*;
- содержит множественные операции (*UNION* и др.);
- содержит предложение *GROUP BY*;
- ссылается на другое необновляемое представление;
- содержит вычисляемые выражения в списке выбора;
- выбирает данные более чем из одной таблицы.

Оператор CASE

Оператор **CASE** может быть использован в одной из двух синтаксических форм записи:

1-я форма:

```
CASE <проверяемое выражение>
    WHEN <сравниваемое выражение 1> THEN <возвращаемое значение 1>
    …
    WHEN <сравниваемое выражение N> THEN <возвращаемое значение N>
    [ELSE <возвращаемое значение>]
END
```

2-я форма:

```
CASE
        WHEN <предикат 1> THEN <возвращаемое значение 1>
        …
        WHEN <предикат N> THEN <возвращаемое значение N>
        [ELSE <возвращаемое значение>]
END
```

Особенности использования CASE

- Все предложения **WHEN** должны иметь одинаковую синтаксическую форму, то есть нельзя смешивать первую и вторую формы.
- При использовании первой синтаксической формы условие **WHEN** удовлетворяется, как только значение проверяемого выражения станет равным значению выражения, указанного в предложении **WHEN**.
- При использовании второй синтаксической формы условие **WHEN** удовлетворяется, как только предикат принимает значение **TRUE**.
- При удовлетворении условия оператор **CASE** возвращает значение, указанное в соответствующем предложении **THEN**.
- Если ни одно из условий **WHEN** не выполнилось, то будет использовано значение, указанное в предложении **ELSE**.
- При отсутствии **ELSE**, будет возвращено **NULL**-значение.
- Если удовлетворены несколько условий, то будет возвращено значение предложения **THEN** первого из них, так как остальные просто не будут проверяться.

Примеры использования оператора CASE

1) Посчитать количество студентов дневной и вечерней формы обучения:

```
create view students_number
(DEPARTMENT,YEAR,DAY_FORM,EVENING_FORM) as
select gr.department, gr.year,
    count(case when gr.study='ДНЕВНАЯ'   then 1 else null end)  form1,
    count(case when gr.study='ВЕЧЕРНЯЯ' then 1 else null end)  form2
  from groups gr,   students st
    where gr.group_code = st.group_code
    group by gr.department, gr.year, gr.study
    order by gr.department, gr.year asc;
```

Примеры использования оператора CASE

2) Вывести все имеющиеся модели ПК с указанием цены. Отметить самые дорогие и самые дешевые модели.

```
SELECT DISTINCT model, price,
  CASE price
    WHEN (SELECT MAX(price) FROM
PC)
       THEN 'Самый дорогой'
    WHEN (SELECT MIN(price) FROM PC)
       THEN 'Самый дешевый'
    ELSE 'Средняя цена'
  END comment
FROM PC
 ORDER BY price;
```

model	price	comment
1232	350.0	Самый дешевый
1260	350.0	Самый дешевый
1232	400.0	Средняя цена
1232	600.0	Средняя цена
1233	600.0	Средняя цена
1121	850.0	Средняя цена
1233	950.0	Средняя цена
1233	980.0	Самый дорогой