# БАЗЫ ДАННЫХ

Язык запросов SQL. Команда SELECT

# Пример БД: проектная организация

**Emp** – сотрудники:

**tabno** – табельный номер сотрудника, первичный ключ;
**name** – ФИО сотрудника, обязательное поле;
**born** – дата рождения сотрудника, обязательное поле;
**sex** – пол сотрудника, обязательное поле;
**depno** – номер отдела, обязательное поле, внешний ключ;
**post** – должность сотрудника;
**salary** – оклад, больше МРОТ;
**passport** – серия и номер паспорта, уникальный обязательный атрибут;
**pass_date** – дата выдачи паспорта, обязательное поле;
**pass_get** – кем выдан паспорт, обязательное поле;
**born_seat** – место рождения сотрудника;
**edu** – образование сотрудника;
**special** – специальность по образованию;
**diplom** – номер диплома;
**phone** – телефоны сотрудника;
**adr** – адрес сотрудника;
**edate** – дата вступления в должность, обязательное поле.

# Пример БД: проектная организация

**Departs** – отделы:

**did** – номер отдела, первичный ключ;

**name** – название отдела, обязательное поле.

**Project** – проекты:

**No** – номер проекта, первичный ключ;

**title** – название проекта, обязательное поле;

**pro** – краткое название проекта, обязательное уникальное поле;

**client** – заказчик, обязательное поле;

**dbegin** – дата начала выполнения проекта, обязательное поле;

**dend** – дата завершения проекта, обязательное поле;

**cost** – стоимость проекта, обязательное поле.

**Job** – участие в проектах:

**pro** – краткое название проекта, внешний ключ;

**tabNo** – номер сотрудника, участвующего в проекте, внешний ключ;

**rel** – роль сотрудника в проекте; может принимать одно из трех значений:
'исполнитель', 'руководитель', 'консультант'.

Первичный ключ – комбинация полей **pro** и **tabNo**.

# Команда SELECT – выборка данных

Общий синтаксис:

SELECT [{ ALL | DISTINCT }] { *список_вывода* | * }
  FROM *имя_таблицы1* [ *алиас1* ] [, *имя_таблицы2* [ *алиас2* ].,..]
  [ WHERE *условие_отбора_записей* ]
  [ GROUP BY { *имя_поля* | *выражение* }.,.. ]
  [ HAVING *условие_отбора_групп* ]
  [ UNION [ALL] SELECT …]
  [ ORDER BY *имя_поля1* | *целое* [ ASC | DESC ]
    [, *имя_поля2* | *целое* [ ASC | DESC ].,..]];

Примеры:

```
select * from departs;
select   name, post   from emp;
```

# Формирование списка вывода (проекция)

Общий синтаксис списка вывода:

[{all | distinct}] { * | *выражение1* [*алиас1*] [, *выражение2* [*алиас2*] .,..]}

Список ввода находится между ключевыми словами **SELECT** и **FROM**.

- Вывести все поля всех записей из таблицы Проекты (Project):

  **select * from project;**

2. Вывести список сотрудников с указанием их должности и № отдела:

   **select depno, name, post**
   **from emp;**

3. Вывести список сотрудников с указанием их должности и зарплаты:

   **select name 'ФИО', post 'Должность', salary*0.87 'Зарплата'**
   **from emp;**

Установить другой формат вывода даты:

**alter session set nls_date_format = 'dd/mm/yyyy';**

# Формирование списка вывода (проекция)

1. Вывести должности и оклады сотрудников:

```
select  post,  salary
        from  emp;
```

2. Вывести должности и оклады сотрудников **без повторов**:

```
select  DISTINCT  post,  salary
        from  emp;
```

3. Вывести отделы и должности сотрудников **без повторов**:

```
select  DISTINCT  depno,  post
        from  emp;
```

4. Задание: вывести список сотрудников с указанием ФИО, даты рождения и адреса.

```
select name  'ФИО',  born  'Дата рождения',  adr  'Адрес'
        from  emp;
```

# Упорядочение результата

1. Вывести данные из таблицы Проекты в порядке даты начала проекта:

```
select *
    from Project
    order by  dbegin;
```

2. Упорядочить список сотрудников по отделам и по ФИО:

```
select depno, name, post
    from emp
    order by  depno,  name;      -- order by 1,2;
```

3. Упорядочить сотрудников по зарплате (от большей к меньшей):

```
select name  'ФИО',  post  'Должность',  salary  'Зарплата'
    from emp
    order by  3  DESC;
```

4. Упорядочить данные об отделах, должностях и зарплатах:

```
select depno  'Номер отдела',  post  'Должность',  salary  'Зарплата'
    from emp
    order by  1,  3  DESC,  2;
```

# Выбор данных из таблицы (селекция)

**WHERE –** содержит условия выбора отдельных записей. Условие является логическим выражением и может принимать одно из 3-х значений:

- TRUE – истина,
- FALSE – ложь,
- NULL – неизвестное, неопределённое значение (интерпретируется как ложь).

Условие формируется путём применения различных операторов и предикатов.

**Операторы сравнения:**

| | | |
|---|---|---|
| = равно, | <>, != не равно, | > больше, |
| >= больше или равно, | <= меньше или равно, | < меньше. |

1. Вывести список сотрудников 2-го отдела:

```
select *  from emp
      where depno = 2;
```

2. Вывести список текущих проектов:

```
select *  from  project
      where  dend > sysdate;
```

**-- sysdate – функция, возвращающая текущую дату**

# Логические операторы

**Для формирования условий используются следующие логические операторы:**

AND – логическое произведение (И),
OR – логическая сумма (ИЛИ),
NOT – отрицание (НЕ).

Операция И:

| a | b | a AND b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Операция ИЛИ:

| a | b | a OR b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Операция НЕ:

| a | NOT a |
|---|---|
| 0 | 1 |
| 1 | 0 |

# Выбор данных из таблицы по условию

1. Вывести список сотрудников 2-го отдела с зарплатой больше 30000 рублей:

```
select * from emp
    where depno = 2 AND salary > 30000 ;
```

2. Вывести список сотрудников-мужчин, родившихся после 1979 года:

```
select * from emp
    where born > '31/12/1979' AND sex = 'м';
```

3. Вывести список сотрудников 2-го и 5-го отделов:

```
select * from emp
    where depno=2 OR depno = 5;
```

4. Вывести список сотрудников 2-го и 5-го отделов в зарплатой не менее 30000:

```
select * from emp
    where (depno=2 OR depno = 5) AND salary >= 30000 ;
```

5. Вывести список всех сотрудников, кроме сотрудников 2-го и 5-го:

```
select * from emp
    where NOT (depno=2 OR depno = 5);
```

# Выбор данных из таблицы по условию

Задание 1: вывести список текущих проектов стоимостью более 2 млн. рублей.

```
select  *
        from  project
        where  dend > sysdate  AND  cost > 2000000;
```

Задание 2: вывести список сотрудников, работающих в должностях 'инженер' и 'ведущий инженер'.

```
select  *
        from  emp
        where  post = 'инженер'  OR  post = 'ведущий инженер' ;
```

Задание 3: вывести список сотрудников, работающих в должности 'охранник', с зарплатой более 20000 рублей.

```
select  *
        from  emp
        where  post = 'охранник'  AND  salary > 20000;
```

# Предикаты формирования условия

**Предикат вхождения в список значений:**

***имя_поля*** **IN ( *значение1* [, *значение2*,... ] )**

***выражение*** **IN ( *значение1* [, *значение2*,... ] )**

Примеры:

- Список сотрудников отделов 5, 8 и 9:

```
select *
    from emp
    where depno  IN ( 5, 8, 9 ) ;
```

- Список сотрудников, работающих в должностях 'инженер' и 'ведущий инженер' :

```
select  *
    from  emp
    where  post  IN ( 'инженер', 'ведущий инженер' );
```

# Предикаты формирования условия

**Предикат вхождения в диапазон:**

***имя_поля* BETWEEN *минимальное_значение* AND *максимальное_значение***

***выражение* BETWEEN *минимальное_значение* AND *максимальное_значение***

Минимальное значение должно быть меньше либо равно максимальному.

Примеры:

- Список всех сотрудников со 2-го по 5-й отделы:

```
select *
    from emp
    where depno BETWEEN 2 AND 5 ;
```

- Список сотрудников с чистой зарплатой от 20 до 30 тысяч рублей:

```
select *
    from emp
    where salary*0.87 BETWEEN 20000 AND 30000;
```

# Предикаты формирования условия

**Предикат поиска подстроки: *имя_поля* LIKE '*шаблон*'**

Этот предикат применяется только к полям типа CHAR и VARCHAR.
Возможно использование шаблонов:
'_' – один любой символ,
'%' – произвольное количество любых символов (в т.ч., ни одного).

Примеры:

- Список всех сотрудников-экономистов:

```
select *  from emp
      where post  LIKE  '%экономист%' ;
```

- Список всех инженеров-специалистов (кроме просто инженеров):

```
select  *  from  emp
      where  post  LIKE  'инженер_%' ;
```

# Предикаты формирования условия

**Предикат поиска неопределенного значения:**

***значение*** **IS [NOT] NULL**

Если значения является неопределенным (NULL), то предикат IS NULL выдаст истину, а предикат IS NOT NULL – ложь.

Примеры:

- Список всех сотрудников, у которых нет телефона (номер телефона неопределен):

```
select *
    from emp
    where phone IS NULL;
```

- Список все проекты, у которых определена стоимость:

```
select *
    from project
    where cost IS NOT NULL;
```

# Использование предикатов

Задание 1: вывести список сотрудников, которых зовут 'ЮРИЙ'.

```
select *
    from emp
    where name LIKE '%ЮРИЙ%';
```

Задание 2: вывести список проектов стоимостью от 1 до 2 млн. рублей.

```
select *
    from project
    where cost BETWEEN 1000000 AND 2000000;
```

Задание 3: вывести список сотрудников, которые являются начальниками отделов.

```
select *
    from emp
    where post LIKE 'нач%отдел%';
```

# Агрегирующие функции

**COUNT** – подсчёт количества строк (значений). Применяется к записям и полям любого типа. Имеет 3 формата вызова:

- **count (*)** – количество строк результата;
- **count (*имя_поля*)** – количество значений указанного поля, не являющихся *NULL*-значениями.
- **count (distinct *имя_поля*)** – количество разных не-NULL значений указанного поля.

**MAX, MIN** – определяет максимальное (минимальное) значение указанного поля в результирующем множестве. Применяется к полям любого типа.

**SUM** – определяет арифметическую сумму значений указанного числового поля в результирующем множестве записей.

**AVG** – определяет среднее арифметическое значений указанного числового поля в результирующем множестве записей. Не учитывает *NULL*-значения, и сумма значений поля делится на количество определённых значений.

# Примеры использования функции COUNT

1. Вывести количество сотрудников:

```
select  count(*)
        from  emp;
```

2. Вывести количество сотрудников с телефонами:

```
select  count( phone )
        from  emp;
```

3. Вывести количество разных должностей сотрудников:

```
select   count (DISTINCT  post)
        from  emp;
```

4. Задание: вывести количество сотрудников 6-го отдела.

```
select   count(*)
        from  emp
        where  depno = 6;
```

# Примеры использования агрегирующих функций

1. Вывести максимальную и минимальную стоимость проектов:

```
select  max(cost) "Максимальная цена",  min(cost)  "Минимальная цена"
        from  project;
```

2. Вывести сумму зарплаты сотрудников 8-го отдела:

```
select  sum(salary)
        from  emp
        where  depno = 8;
```

3. Вывести среднюю зарплату сотрудниц предприятия:

```
select  avg(salary)
        from emp
        where  sex = 'Ж';
```

4. Вывести даты начала работы над первым проектом и завершения работы над последним проектом:

```
select   min(dbegin),  max(dend)
        from  project;
```

# Группировка данных: предложение GROUP BY

Агрегирующие функции обычно используются совместно с предложением ***GROUP BY***.

Например, следующая команда считает количество сотрудников по отделам:

```
select  depno,  count(*)
        from emp
        group by depno;
```

| depno | name | … |
|---|---|---|
| 1 | Белов С.В. | |
| 1 | Иванова К.Е. | |
| 1 | Седов О.Л. | |
| 2 | Волков Н.Е. | |
| 2 | Рогов И.Л. | |
| 3 | Санина В.П. | |
| 3 | Дымова С.Т. | |
| 3 | Павлов К.Д. | |
| 3 | Орлов Т.Ф. | |

| depno | count(*) |
|---|---|
| 1 | 3 |
| 2 | 2 |
| 3 | 4 |

# Примеры использования GROUP BY

1. Вывести минимальную и максимальную зарплату в каждом отделе:

```
select depno, MIN(salary) minsal, MAX(salary) maxsal
    from emp
    group by depno;
```

2. Вывести количество разных должностей в каждом отделе:

```
select depno, COUNT(distinct post) cnt
    from emp
    group by depno;
```

3. Посчитать сумму зарплат в каждом отделе:

```
select depno, SUM(salary) allsal
    from emp
    group by depno;
```

4. Посчитать среднюю зарплату по каждой должности:

```
select post, AVG(salary) avgsal
    from emp
    group by post;
```

# Использование GROUP BY

Правило использования *GROUP BY* :

В списке вывода при использовании *GROUP BY* могут быть указаны только **функции агрегирования, константы и поля, перечисленные в *GROUP BY*.**

Если включить в список выбора поля, не указанные в *GROUP BY*, то СУБД не будет выполнять такой запрос и выдаст ошибку "нарушение условия группирования" (not a GROUP BY expression).

Например, нельзя получить сведения о том, у каких сотрудников самая высокая зарплата в своём отделе с помощью такого запроса:

```
select  depno,  name,  max(salary) as max_sal
        from  emp
        group by  depno;
```

**Этот запрос синтаксически неверен!**

| depno | name | salary |
|---|---|---|
| 1 | Белов С.В. | 58000 |
| 1 | Иванова К.Е. | 28000 |
| 1 | Седов О.Л. | 41000 |
| 2 | Волков Н.Е. | 40000 |
| 2 | Рогов И.Л. | 32000 |
| **3** | Санина В.П. | **47000** |
| **3** | Дымова С.Т. | 29000 |
| **3** | Павлов К.Д. | **47000** |
| **3** | Орлов Т.Ф. | 30000 |

# Группировка по нескольким полям

1. Сумма зарплаты по отделам и по должностям:

```
select  depno,  post,  count(*),  sum(salary)
        from  emp
        group by  depno,  post;
```

2. Количество мужчин и женщин по отделам:

```
select  depno,  sex,  count(*)
        from  emp
        group by  depno,  sex;
```

**Задание**: вывести информацию о зарплате и количестве сотрудников, которые получают такую зарплату.

```
select  salary,  count(*)
        from  emp
        group by  salary;
```

# Использование фразы HAVING

Если необходимо вывести не все записи, полученные в результате группировки (GROUP BY), то условие на группы можно указать во фразе HAVING (но не во фразе WHERE).

Пример. Список отделов, в которых работает больше пяти человек:

```
select  depno,  count(*), 'человек(а)'
        from  emp
        group by  depno
        having  count(*)>5;
```

**Правило: нельзя указывать агрегирующие функции в части WHERE – это синтаксическая ошибка!**

**Задание**: вывести список отделов, в которых средняя зарплата больше 30000 рублей.

```
select  depno,  avg(salary)
        from  emp
        group by  depno
        having  avg(salary) > 30000;
```

# Операции реляционной алгебры

**Унарные операции:**

- **селекция** – выбор из таблицы подмножества строк по условию.
  Например, список сотрудников 5-го отдела:

```
select *
    from emp
    where depno = 5;
```

- **проекция** – выбор из таблицы подмножества столбцов.
  Например, сведения о должности и зарплате сотрудников:

```
select  distinct name,  post,  salary
    from  emp;
```

# Бинарные операции реляционной алгебры

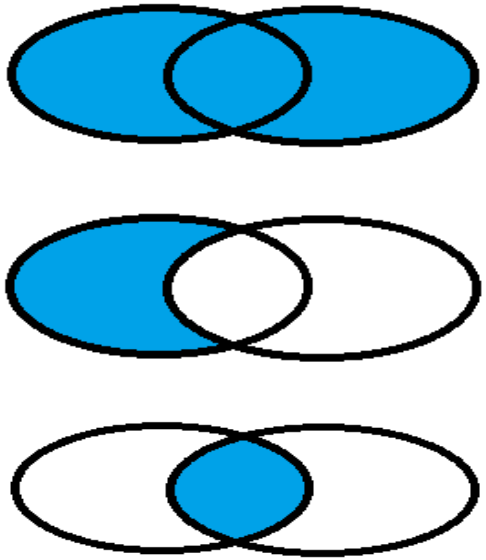
**Бинарные операции РА:**

- **разносхемные** – применяются к любым двум отношениям.

- **односхемные** – применяются к односхемным отношениям. Исходные отношения должны иметь одинаковое количество столбцов одинаковых (или сравнимых) типов. ***Сравнимыми*** считаются типы, относящиеся к одному и тому же семейству данных (в таблице полужирным шрифтом выделены базовые типы).

# Бинарные односхемные операции РА

- ✓ **Объединение** двух односхемных отношений содержит все строки исходных отношений без повторов.
- ✓ **Разность** двух односхемных отношений содержит все строки первого отношения, не входящие во второе отношение (без повторов).
- ✓ **Пересечение** двух односхемных отношений содержит все строки, входящие и в первое, и во второе отношения (без повторов).



Добавим в нашу БД проектной организации таблицу "Архив должностей":

```
create table  archive (
        tabno    number(6) REFERENCES  emp,        -- ссылка на сотрудника
        name     varchar2(100)  not null,          -- ФИО сотрудника
        dbegin   date  not null,                   -- начало работы в должности
        post     varchar(50)  not null             -- должность
);
```

# Операция объединения

**Объединение** реализуется с помощью специального ключевого слова **UNION** (или **UNION ALL**, если не нужно удалять повторы).

Примеры:

- Список сотрудников с телефонами или адресами (если нет телефона):

```
select depno, name, PHONE
    from emp where phone is not null
UNION ALL
select depno, name, ADR
    from emp where phone is null;
```

- Список сотрудников со всеми переводами с одной должности на другую:

```
select tabno, name, edate, post
    from emp
UNION ALL
select tabno, name, dbegin, post
    from archive
   order by 1, 3;
```

# Разность отношений

**Разность** реализуется с помощью специального ключевого слова **MINUS**.

Примеры:

- Список сотрудников 5-го и 8-го отделов, которые не являются инженерами:

```
select * from emp
      where depno IN (5, 8)
MINUS
select * from emp
      where post LIKE '%инженер%'
order by depno;
```

- Список сотрудников, которые не переводились на другие должности:

```
select tabno, name
      from emp
MINUS
select tabno, name
      from archive;
```

# Пересечение отношений

**Пересечение** реализуется с помощью специального ключевого слова **INTERSECT**.

Примеры:

- Список сотрудников 5-го и 8-го отделов, которые являются инженерами:

```
select * from emp
      where depno IN (5, 8)
INTERSECT
select * from emp
      where post LIKE '%инженер%'
order by depno;
```

- Список сотрудников, которые переводились на другие должности:

```
select tabno, name
      from emp
INTERSECT
select tabno, name
      from archive;
```

# Применение односхемных операций РА

**Задание 1**: вывести список должностей, которые занимают (или занимали) сотрудники.

```
select post from emp
UNION
select post from archive;
```

**Задание 2**: вывести список должностей, на которые переназначены другие сотрудники.

```
select post from emp
INTERSECT
select post from archive;
```

**Задание 3**: вывести список должностей, которые в настоящее время не занимает ни один сотрудник.

```
select post from archive
MINUS
select post from emp;
```

# Разносхемные операции РА

Декартово произведение (ДП): операция над двумя произвольными (возможно, разносхемными) отношениями. Результат ДП – все комбинации строк исходных отношений. Пример:

**"Студенты"**

| *Группа* | *ФИО* |
|---|---|
| СТ-4/09 | Рогов В.П. |
| СТ-4/09 | Белова О.Г. |

**"Предметы"**

| *Предмет* |
|---|
| Базы данных |
| Сетевые технологии |

**"Оценки"**

| *Оценка* |
|---|
| удовл. |
| хор. |
| отл. |

**Декартово произведение: "Оценки студентов"**

| *Группа* | *ФИО* | *Предмет* | *Оценка* |
|---|---|---|---|
| СТ-4/09 | Рогов В.П. | Базы данных | удовл. |
| СТ-4/09 | Рогов В.П. | Базы данных | хор. |
| СТ-4/09 | Рогов В.П. | Базы данных | отл. |
| СТ-4/09 | Рогов В.П. | Сетевые технологии | удовл. |
| СТ-4/09 | Рогов В.П. | Сетевые технологии | хор. |
| СТ-4/09 | Рогов В.П. | Сетевые технологии | отл. |
| СТ-4/09 | Белова О.Г. | Базы данных | удовл. |
| СТ-4/09 | Белова О.Г. | Базы данных | хор. |
| СТ-4/09 | Белова О.Г. | Базы данных | отл. |
| СТ-4/09 | Белова О.Г. | Сетевые технологии | удовл. |
| СТ-4/09 | Белова О.Г. | Сетевые технологии | хор. |
| СТ-4/09 | Белова О.Г. | Сетевые технологии | отл. |

# Разносхемные операции РА

Пример декартова произведения реальных таблиц:

```
select  *
        from  depart,  emp;
```

Если в части FROM указываются 2 и более таблицы, то СУБД по умолчанию строит их декартово произведение.

Другая разносхемная операция – соединение: селекция от декартова произведения.

Примеры.

1. Список отделов и их сотрудников:

```
select  *
        from  depart, emp
        where  emp.depno = depart.did;
```

2. Список проектов и их участников:

```
select  *
        from  project,  emp,  job
        where  emp.tabno = job.tabno
           and  job.pro = project.pro;
```

# Применение операции соединения

Задание 1: вывести сотрудников с указанием ролей, которые они исполняют в проектах.

```
select  e.name,  j.rel
    from  emp  e,  job  j
    where  e.tabNo = j.tabNo;
```

Задание 2:  вывести список проектов с указанием их руководителей.

```
select  p.title,  e.name
     from  emp  e,   job  j,   project  p
     where e.tabno = j.tabno
              and  j.pro = p.pro
              and  j.rel = 'руководитель';
```

# Применение операции соединения

Задание 3: вывести список сотрудников с указанием количества проектов, в которых они участвуют.

```
select name, count(*)
    from emp, job
    where emp.tabno=job.tabno
    group by emp.tabno, emp.name;
```

Задание 4: вывести список проектов, в которых участвует более 5 сотрудников.

```
select p.title, count(*)
    from job j, project p
    where p.pro = j.pro
    group by p.pro, p.title
   having count(*) > 5;
```

# Общий алгоритм выполнения операции *SELECT*

1. Выбор записей из указанной таблицы (***from***).
2. Проверка для каждой записи условия отбора (***where***).
3. Группировка полученных в результате отбора записей (***group by***) и вычисление для этих групп значений агрегирующих функций.
4. Выбор тех групп, которые удовлетворяют условию отбора групп (***having***).
5. Сортировка полученных записей в указанном порядке (***order by***).
6. Извлечение из полученных записей тех полей, которые заданы в списке вывода, и формирование результирующего отношения.

   Если в части FROM указывается 2 и более таблицы, то приведенный алгоритм выполняется для декартова произведения этих таблиц.