Лекция 4

Формальное описание оператора SELECT

```
SELECT [DISTINCT]
{ {function_agregate | expr [AS new_field_name] } .,:
      | specification.*
      | *
      [INTO list_variable]
      FROM {{ имя_таблицы [AS] [table_alias] [(field .,:)]}
                  | {subquery [AS] subquery_alas [(field .,:)]}
                  | union_table
                  | constructor_of_table_value
                  | {TABLE имя_таблицы [AS] alias [(field .,:)]}
} .,:
      [WHERE condition]
      [GROUP BY {{ имя_таблицы | alias }.field} .,: {COLLATE name}]      [HAVING condition]
      [{ UNION | INTERSECT | EXCEPT } [ALL]
                  [CORRESPONDING [BY (field.,:)]]
                  SELECT_operator | {TABLE имя_таблицы} | constructor_of_table_value

                              [ORDER BY] {{field_result [ASC|DESC]}.,:}
                              |{{ integer [ASC|DESC]}.,:} ;
```

Агрегатные функции

Функции агрегирования

- **COUNT(*имя_поля*)** - количество значений указанного поля, неявляющихся **NULL-значениями**.
- **COUNT** (*) - подсчет количества всех значений поля в группе, не игнорирует значения **NULL.**
- **COUNT (distinct *имя_поля*)** – количество разных **не-NULL** значений указанного поля.
- **AVG(*имя_поля*)** - определение среднего значения.
- **SUM (*имя_поля*)** - подсчет суммы всех значений группы. Если при этом получаемое значение выходит за пределы суммируемого типа данных, то инициируется ошибка выполнения SQL-оператора.
- **MAX(*имя_поля*)** - определение максимального значения из группы.
- **MIN(*имя_поля*)** - определение минимального значения из группы.

HAVING оператора **SELECT** определяет предикат аналогично фразе **WHERE**, но применяемый к строкам, полученным в результате выполнения функций агрегирования

Особенности использования функций агрегирования

- В качестве выражения в агрегатных функциях может использоваться любая константа, функция, комбинация из названий столбцов, констант и функций, соединенных арифметическими или другими допустимыми в контексте данной СУБД операциями.
- Обработка значений NULL в агрегатных функциях по разному реализована в различных СУБД. Какие-то СУБД обрабатывают значение как 0, другие игнорируют его.
- По умолчанию в агрегатных функциях используется опция **ALL**.

Простейший синтаксис для использования функций агрегирования:

SELECT aggr_function([ALL|DISTINCT] <выражение>)
{,aggr_function([ALL|DISTINCT] <выражение>)}*
FROM <имя таблицы>

В большинстве СУБД при использовании хотя бы одной агрегатной функции недопустимо использование после **SELECT** выражений, отличных от агрегатных.

Простейшее агрегирование.

Определение общего количества студентов:

SELECT COUNT(*) AS "Количество студентов" FROM student;

Средний балл по итогам результатов экзамено:.

SELECT AVG(Mark) AS AvgMark FROM exam_result;

Агрегирование с использованием опции факторизации. Количество групп, в которых учатся студенты:

SELECT COUNT(DISTINCT GroupNumber) AS GroupCount FROM student;

Агрегирование двух значений:

Даты рождения самого младшего и самого старшего студентов.

SELECT MIN(BirthDate) AS MinBirthDate, MAX(BirthDate) AS MaxBirthDate FROM student;

Примеры

SELECT COUNT(*) FROM students;
(Подсчитать общее количество студентов)

SELECT AVG (grade) FROM grades WHERE student_id IN (SELECT id FROM students WHERE course_id = 1);
(Подсчитать среднюю оценку студентов проходящих курс с id 1)

SELECT MIN (age) FROM students;
(Найти возраст самого младшего студента)

SELECT SUM (duration) FROM courses;
(Подсчитать общее количество часов на все курсы)

Дубликаты в агрегатных функциях

По умолчанию все вышеперечисленных пять функций учитывают все строки выборки для вычисления результата. Но выборка может содержать повторяющиеся значения. Если необходимо выполнить вычисления только над уникальными значениями, исключив из набора значений повторяющиеся данные, то для этого применяется оператор **DISTINCT**.
По умолчанию вместо **DISTINCT** применяется оператор **ALL**, который выбирает все строки:

Пример:
SELECT COUNT(DISTINCT manufacturer) FROM products
(Подсчитать число уникальных производителей)

Сортировка данных

Упорядочивание результирующего набора

Фраза **ORDER BY** применяется для упорядочивания результирующего набора, которое выполняется в соответствии со значениями столбцов, указанных в списке после фразы **ORDER BY**.

Сначала производится упорядочивание по первому указанному столбцу, потом по второму и т.д.

При упорядочивании можно указать опцию **ASC** (по возрастанию) или **DESC** (по убыванию).

Например:

SELECT f1,f2 FROM tbl1 ORDER BY f2;

Сортировка. ORDER BY

Оператор **ORDER BY** сортируют значения по одному или нескольких столбцам.

По умолчания **ORDER BY** использует сортировку **ASC** - по возрастанию. Можно сделать сортировку по убыванию добавив **DESC** после указания колонки для сортировки.

```
SELECT * FROM students ORDER BY age;
```
(Выбрать всех студентов начиная от самого младшего и к самому старшему студенту)

```
SELECT * FROM students ORDER BY age DESC;
```
(Выбрать всех студентов начиная от самого старшего к младшему студенту)

Сортировку можно производить по нескольким колонкам одновременно. Для этого нужно после **ORDER BY** указать нужные колонки через запятую.
Строки будут отсортированы сначала по первой колонке, и там где значения повторяются для этой колонки, строки будут отсортированы по второй.

Группировка GROUP BY

Группировка данных в таблицах

Для анализа данных с одинаковыми значениями атрибутов записи таблицы разбиваются на группы. Интересующие значения указываются после ключевого слова **GROUP BY**.

В результате для полученных групп записей можно определить некоторые характерные значения с помощью ***агрегатных функций***, вычисляющих одно значение для каждой группы:

COUNT	Количество записей
SUM	Сумма значений
AVG	Среднее значение
MIN	Минимальное значение
MAX	Максимальное значение

Запрос. Сколько участков обслуживает поликлиника (ск.разных номеров участков).

```
SELECT COUNT (DISTINCT НомУчастка)
FROM Участки
GROUP BY НомерУчастка
```

Участки

КодАдреса
КодУлицы
НомДома
НомКорп
НомУчастка

Результат

НомУчастка	COUNT (*)
2	3
1	2
3	

Результат

COUNT (DI
3

Упрощенный синтаксис для агрегирования с условием для группировки

SELECT <список из агрегатных функций и выражений для группировки>

FROM …

WHERE условия_на_ограничения_строк

GROUP BY <выражения для группировки>

HAVING <условие для группировки> -- Команда HAVING позволяет фильтровать результат группировки

<условие для группировки> – это логическое выражение, построенное на основе агрегатных функций и <выражений для группировки>

Пример: агрегирование с условием для группировки

Номера зачеток студентов, получавших на экзаменах только 5.

SELECT StudentId FROM exam_result

GROUP BY StudentId

HAVING MIN(Mark) = 5;

Группировка. GROUP BY

Команда **HAVING** позволяет фильтровать результат группировки, сделанной с помощью команды **GROUP BY**.

```
SELECT *
FROM имя_таблицы
        WHERE условие
GROUP BY поле_для_группировки
        HAVING условие_группировки
```

Упрощенный синтаксис для использования функций агрегирования с группировкой

SELECT <список из агрегатных функций и выражений для группировки> FROM
…<таблица или подзапрос (SELECT …FROM …)
GROUP BY <выражения для группировки>

Пример: агрегирование с группировкой по одному полю

Количество студентов в каждой группе:

```
SELECT GroupNumber AS "Номер группы", COUNT(*) AS "Кол-во студентов в группе"
FROM student GROUP BY GroupNumber;
```

Пример: агрегирование с группировкой по двум полям

Сколько экзаменов назначено на каждую дату и аудиторию?

```
SELECT ClassRoom, ExamDate, COUNT(*) AS ExamCount
FROM exam_sheet GROUP BY ClassRoom, ExamDate;
```

Пример: агрегирование с группировкой по значению функции

Как даты рождения студентов распределены по годам?

```
SELECT  YEAR(BirthDate) AS "Year", COUNT(*) AS "Count of students” FROM
student GROUP BY YEAR(BirthDate);
```

Группировка. GROUP BY

Пример

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Группировка. GROUP BY

Пример

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

Expr1000	Country
3	Argentina
2	Austria
2	Belgium
9	Brazil
3	Canada
2	Denmark
2	Finland
11	France

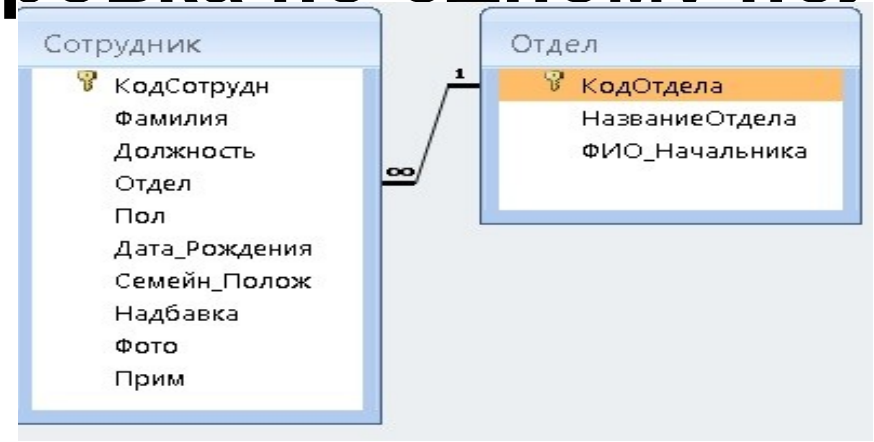
Группировка. GROUP BY

Пример 2

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
        HAVING COUNT(CustomerID) > 10
```

Expr1000	Country
11	France
11	Germany
13	USA

Группировка по одному полю

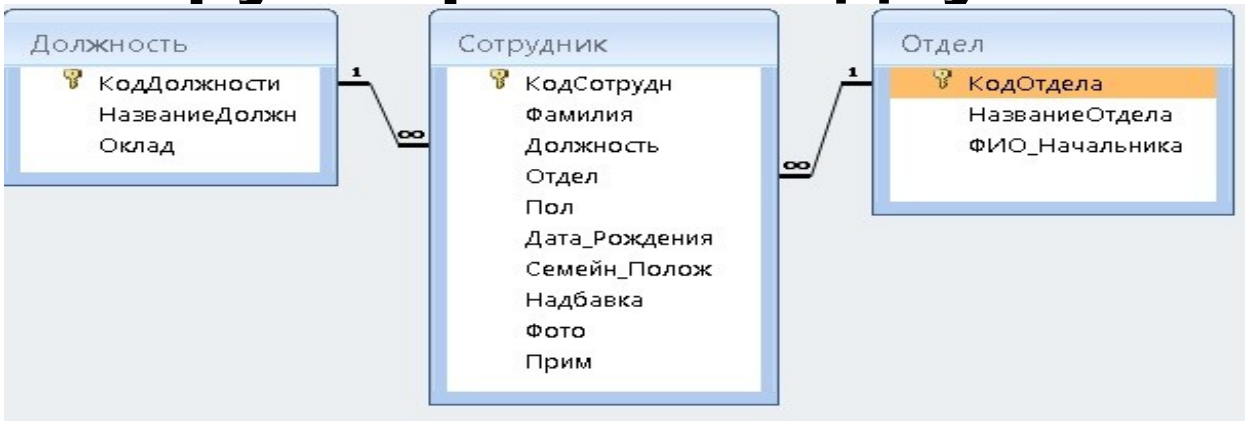



```
SELECT Отдел.НазваниеОтдела, Count(Сотрудник.КодСотрудн) AS [Count-КодСотрудн]
FROM Отдел INNER JOIN Сотрудник ON Отдел.КодОтдела = Сотрудник.Отдел
GROUP BY Отдел.НазваниеОтдела;
```

Копия "Группир1поля"

Отдел	Count-КодСотрудн
Бухгалтерия	1
Маркетинг	2
Плановый	2
Реклама	2

Группировка по двум полям




```
SELECT Отдел.НазваниеОтдела, Должность.НазваниеДолжн,
    Count(Сотрудник.КодСотрудн) AS [Count-КодСотрудн]
FROM Отдел INNER JOIN (Должность INNER JOIN Сотрудник ON
    Должность.КодДолжности = Сотрудник.Должность) ON
    Отдел.КодОтдела = Сотрудник.Отдел
GROUP BY (                                              Должн;
```

Отдел	НазваниеДолжн	Count-КодСотрудн
Бухгалтерия	Бухгалтер	1
Маркетинг	Маркетолог	1
Маркетинг	Менеджер	1
Плановый	Экономист	2
Реклама	Менеджер	2

Приведение типов

Неявное приведение типов

В большинстве случаев преобразование типов осуществляется автоматически
Что будет, если к числу прибавить строку?
```
SELECT '5' + 3;
```
(Выведет: 8)

```
SELECT '5st' + 3;
```
(Выведет: 8)

В этом случае строка преобразуется в число, а затем выполняется операция сложения. Но что будет, если строку невозможно преобразовать в число?
```
SELECT 'str' + 3;
```
(Выведет: 3)

```
SELECT 3 + 'str';
```
(Выведет: 3)
Если строку невозможно преобразовать в число, то она приравнивается к нулю.

Явное приведение типов

Для явного преобразования типов используются функции:

- **CAST**(<Выражение> AS <Тип>);
- **CONVERT**(<Выражение>, <Тип>).

Функция **CONVERT** также позволяет преобразовать кодировку строки.
Параметр <Тип> может принимать следующие значения:

- BINARY;
- CHAR;
- DATE;
- DATETIME;
- SIGNED [INTEGER];
- TIME;
- UNSIGNED [INTEGER].

Явное приведение типов

Примеры:
SELECT CAST(20091201 AS DATE);
(Приводит цифровое значение даты к типу date)

SELECT CAST(PI() AS UNSIGNED) AS PI;
(Приводит число pi к целому)

SELECT CONVERT('Какой нибудь текст' USING koi8r);
(Преобразует текст в кодировку koi8r)