Раздел 4. Основы языка SQL

Тема 4.1
Язык запросов DQL (Data Query Language)

Вопросы лекции:

1. Общая характеристика операторов SQL
2. Синтаксис команды SELECT
3. Запросы к одной таблице БД
4. Операция расширения
5. Операция группировки

SQL *(Structured Query Language)* — **язык структурированных запросов** — стандартный язык запросов по работе с реляционными БД. Он основан на реляционном исчислении.

Первый международный стандарт языка SQL был принят в 1989 г. (он был назван SQL/89 или SQL1).

Иногда стандарт SQL1 также называют **стандартом ANSI/ISO**, и подавляющее большинство доступных на рынке СУБД поддерживают этот стандарт полностью.

В конце 1992 г. был принят новый международный стандарт языка SQL, который в дальнейшем был назван SQL/92 или SQL2.

В 1999 году появился новый стандарт, названный SQL3 и далее он был доработан до стандарта **SQL-2003**

В **SQL-2003** **введены новые типы данных**, при этом предполагается **возможность задания сложных структурированных типов данных**, которые в большей степени соответствуют объектной ориентации.

Наконец, добавлен раздел, который **вводит стандарты на события и триггеры**, которые ранее не затрагивались в стандартах, хотя давно уже широко использовались в коммерческих в т.н. диалектах SQL СУБД.

В качестве действия могут выступать не только последовательность операторов SQL, но и **операторы управления ходом выполнения программы**.

Группы операторов языка SQL-2003.

- Язык (операторы) определения данных – **Data Definishion Language** (**DDL**).
- Язык (операторы) манипулирования данными – **Data Manipulation Language** (**DML**).
- Язык запросов - **Data Query Language** (**DQL**).
- Средства (операторы) управления транзакциями.
- Средства (операторы) администрирования данных.
- Процедурные средства и средства управления ходом выполнения программы.

Раздел 4. Основы языка SQL

Тема 4.1
Язык запросов (Data Query Language)

Вопросы лекции:

1. Общая характеристика операторов SQL
2. Синтаксис команды SELECT
3. Запросы к одной таблице БД
4. Операция расширения
5. Операция группировки

Язык запросов содержит всего одну команду – команду выборки данных **SELECT**.

Команда может содержать **большое разнообразие параметров** (**фраз**), делающих из неё целый полнофункциональный поисковый язык извлечения необходимых данных из реляционных БД.

Результатом выполнения команды SELECT всегда является **новая** таблица (возможно, пустая, из нескольких или из одной строки и т.п.).

Новая **таблица** может участвовать в других командах SQL для других объектов БД как результат вложенного запроса или передается клиенту для последующей обработки.

Всё множество запросов по команде SELECT можно группировать следующим образом:

- Запросы на выборку по одной таблице;
- Соединение таблиц в запросах – запросы к нескольким таблицам и сведение их в одну результирующую таблицу;
- Вложенные запросы – запросы в теле другого запроса;
- Комбинированные запросы – объединение нескольких запросов в один.

Синтаксис оператора SELECT может состоять из многих фраз и имеет следующий вид:

SELECT [ALL | DISTINCT **] (<Список столбцов> | *)**
[INTO список переменных **]**

FROM <Список таблиц или представлений> [IN <имя базы данных>]

- [**WHERE** <Предикат-условие выборки или соединения>]
- [**ORDER BY** <Список полей, по которым упорядочить вывод для пользователя>]
- [**GROUP BY** <Список выражений группировки результата>]
- [**HAVING** <Предикат-условие для отбора группы>]

Синтаксис оператора SELECT

- наличие ключевого слова ALL (по умолчанию) означает, что в результирующую таблицу включаются все строки, удовлетворяющие условиям запроса, что может привести к появлению в результирующей таблице одинаковых строк;
- ключевое слово DISTINCT предназначено для **приведения таблицы в соответствие с принципами теории отношений**, где предполагается отсутствие дубликатов строк;
- символ "*" определяет очень часто встречаемую ситуацию, когда в результирующий набор включаются все столбцы из исходной таблицы запроса.

Например, чтобы выбрать все данные всех столбцов из таблицы students, следует ввести команду

```
SELECT  *  FROM students ;
```

Однако это **практически неприменимый** формат.

Он может применяться для уточнения имеющихся в таблице столбцов или для небольших таблиц, так как обычно требуется выбрать не все столбцы и не все записи.

Простой отбор столбцов таблицы без условия поиска строк называется **операция проекции** (вертикальная селекция данных в реляционной алгебре) и выглядит так:

```
SELECT список_имен_столбцов FROM имя_таблицы ;
```

Например, SELECT st_fam, st_stip FROM students;

Но и **такие запросы достаточно редки**, ибо обычно требуется найти записи, удовлетворяющие какому-то заданному условию.

Раздел 4. Основы языка SQL

Тема 4.1
Язык запросов (Data Query Language)

Вопросы лекции:

1. Общая характеристика операторов SQL
2. Синтаксис команды SELECT
3. Запросы к одной таблице БД
4. Операция расширения
5. Операция группировки

Запросы к одной таблице БД

Результаты выборки по рассмотренным выше запросам могут не удовлетворять пользователей по двум причинам:

- результаты **не отсортированы** и могут выдаваться в различном порядке в разные моменты времени;
- в результатах могут быть строки-дубликаты (например, если имеются студенты-однофамильцы и стипендии у них одинаковы).

Применение параметра **DISTINCT** устраняет строки-дубликаты из результатов запроса,

```
SELECT DISTINCT st_fam, st_stip FROM students;
```

но при этом важно не потерять нужную информацию (если добавить в запрос столбец «**имя**» или «**номер студенческого билета**» – мы не потеряем некоторых студентов, не представленных в предыдущем запросе: `SELECT st_number, st_fam, st_stip FROM students;`

Запросы к одной таблице БД

Результаты выборки могут быть **отсортированы** по одному или сразу по нескольким столбцам таблиц.

Для этого в запросе используется параметр (фраза)

ORDER BY имя_столбца **[ASC** (по умолчанию) **| DESC]**,

например SELECT st_fam FROM students ORDER BY st_fam, st_date;

расставит студентов по алфавиту, а однофамильцев по дате рождения

Для текстовых полей сортировка реализуется по алфавиту, для числовых и «дата-время» - возрастанию или убыванию значений:

SELECT st_fam FROM stud ORDER BY st_fam; - по возрастанию букв алфавита;

SELECT st_fam FROM stud ORDER BY st_date DESC; - по убыванию даты рождения.

Запросы к одной таблице БД

Результаты выборки могут быть отсортированы по одному или сразу по нескольким столбцам таблиц. *(продолжение)*

SELECT st_fam, st_date FROM stud ORDER BY **st_fam, st_date** DESC;
- по одному столбцу - по возрастанию, по другому – по убыванию.

Сортировка может задаваться необязательно по столбцам, указанным для выборки, но и по любым, имеющимся в таблице запроса. При этом записи отсортируются, но видеть столбец второй сортировки в результате запроса мы не будем.

Важно, если сортировка **в обратном** порядке реализуется по нескольким столбцам – нужно после имени каждого столбца указывать ее направление:

SELECT st_fam FROM stud ORDER BY st_fam DESC, st_date DESC ;

Запросы к одной таблице БД

Если столбец (имя) указан вначале запроса для выборки, во фразе ORDER BY не обязательно снова повторять его имя, достаточно указать его порядковый номер в списке выборки:

```
SELECT st_fam, st_date FROM students ORDER BY 1 DESC;
```

Или

```
SELECT st_fam, st_date FROM students ORDER BY 1 DESC,
2 DESC;
```

Запросы к одной таблице БД

Результаты вышерассмотренных запросов обычно используются для отчетов, так как выбирают все записи таблицы **без предварительного отбора** по какому-либо условию.

Отбор строк (операция выборки – горизонтальной селекции в реляционной алгебре**)**

Критерии отбора строк таблиц указываются во фразе WHERE

```
SELECT * FROM students  WHERE  st_number=10;
```

Этот запрос возвратит полную запись одной таблицы по студенту 10.

Запросы к одной таблице БД

Отбор строк (операция выборки)

Условие отбора строк может включать следующие операции:

- Операции сравнения **<, >, <=, >=, =, != или <>** ;
- Операции проверки на отсутствие данных в столбце **IS NULL** или наоборот, их наличие **IS NOT NULL** ;
- Логические операции **AND, OR, NOT** ;
- Операция вхождения значения в заданный диапазон **BETWEEN** *начальное_знач* **AND** *конечное_знач* ;
- Операция принадлежности значения заданному множеству **IN (множество значений)**;
- Операция соответствия заданному шаблону (для текстовых столбцов) **LIKE** ‘шаблон‘ (***различается регистр букв!!!***);

Запросы к одной таблице БД

Отбор строк (операция выборки)

Например:

Вывести сведения о кафедрах университета, находящихся па первом этаже, учитывая тот факт, что номера аудиторий первого этажа лежат в диапазоне от 1 до 99.

Запрос будет выглядеть следующим образом:

```
SELECT *
FROM kafedra
WHERE Nom_Auditoria BETWEEN 1 AND 99;
```

Результат запроса:

Kod_kaf	Name_kaf	Nomtetef	Nom_Auditoria	Col_sotr	Zavjcaf
002	Общей математики	23-65-43	003	22	Махов
005	Прикладной	23-66-62	028	24	Ляхова

Запросы к одной таблице БД

Отбор строк (операция выборки)

В шаблоне для операции **LIKE** можно использовать два символа-заменителя:

% - заменяет последовательность из любых символов;

_ - заменяет один любой символ в заданной позиции.

Например, чтобы выбрать фамилии и телефоны всех студентов, у которых фамилия начинается на букву «С» и при этом отсортировать список по этим фамилиям, нужно ввести:

SELECT **st_fam**, **st_phone** FROM students WHERE st_fam LIKE '**C%**' ORDER BY **1**;

В свою очередь, операция **LIKE '%C%'** вернет список студентов, у которых в фамилии есть **хотя-бы одна** буква «С»,

а операция **LIKE 'C_____'** – (пять символов подчеркивания _) – выдаст фамилии **только из шести букв, начинающиеся на «С».**

У операции LIKE есть некоторые особенности (след слайд)

Отбор строк (операция выборки)

Использование символа пропуска

Если с помощью оператора LIKE необходимо найти строки, в которые входят сами символы-заменители % или _, следует использовать так называемый символ пропуска (*эскейп-символ*).

Поставив его в шаблоне перед символами % или _, указывается, что эти символы в данном запросе не являются заменителями любых других, а именно искомыми символами.

Для указания этой ситуации используется фраза ESCAPE .

Например: Вывести данные факультетов, в названиях которых имеется символ _ (подчеркивания) – типа «кафедра ин_языков».

SELECT * FROM Facultet WHERE kaf_name LIKE '%**?**_%' ESCAPE '**?**' ;

В некоторых СУБД можно это сделать и так: LIKE ['_']

Раздел 4. Основы языка SQL

Тема 4.1
Язык запросов (Data Query Language)

Вопросы лекции:

1. Общая характеристика операторов SQL
2. Синтаксис команды SELECT
3. Запросы к одной таблице БД
4. Операция расширения
5. Операция группировки

Создание вычисляемых столбцов (операция **расширения**)

Вычисляемый столбец – фиктивный столбец, данные которого не хранятся в БД, а вычисляются на основе данных других столбцов.

Вычисляемый столбец получает новое имя – псевдоним (alias).

SELECT …выражение_для _вычисления [AS] псевдоним …

В общем случае **псевдоним может быть присвоен и любому столбцу**, указанному для вывода в запросе.

В выражениях могут использоваться:

- Имена столбцов;
- Константы (Текстовые константы указываются в апострафах - ‘студент‘);
- Операции и операторы;
- Функции (*зависят от типа СУБД*);
- Круглые скобки.

Создание вычисляемых столбцов (операция **расширения**)

Оператор CASE в вычисляемых столбцах позволяет организовать ветвления и внести в запросы элементы процедурной логики.
Он аналогичен аналогичным операторам языков программирования.

```
SELECT cod_st, cod_sub, CASE mark
WHEN 5 THEN ‘ОТЛИЧНО’
WHEN 4 THEN ‘ХОРОШО’
WHEN 3 THEN ‘УДОВЛЕТВОРИТЕЛЬНО’
WHEN 2 THEN ‘ПЛОХО’
END mark FROM marks
```

вместо цифры выведет оценку в виде соответствующего слова.

Создание вычисляемых столбцов (операция **расширения**)

Использование агрегатных функций в вычисляемых столбцах позволяет организовать обработку статистических данных непосредственно на сервере, а не в клиентском приложении БД, что существенно снижает объем трафика.

Для этого в команду SELECT введены агрегатные (статистические, итоговые) функции.

Их особенность – каждая из них **вычисляет одно значение** по какому-либо столбцу (в том числе и вычисляемому) для множества строк таблицы.

Стандартом SQL определены 5 агрегатных функций:

- SUM(имя_столбца) – сумма значений заданного числового столбца;
- AVG(имя_столбца) – среднее значение числового столбца;
- MIN(имя_столбца), MAX(имя_столбца) – минимум и максимум
- COUNT([DISTINCT] * | имя_столбца) – подсчет количества строк.

DISTINCT – подсчет только уникальных значений столбца.

Создание вычисляемых столбцов (операция **расширения**)

Использование агрегатных функций в вычисляемых столбцах

Например:

Подсчитать средний балл по **всем** студентам и предметам:

```
SELECT AVG(mark) avg_mark FROM marks
```

Найти **минимальную и максимальную** даты рождения студентов:

```
SELECT MIN(born) min_date, MAX(born) max_date FROM students
```

Функция **COUNT** может использоваться как без аргументов, так и с ними.

В первом случае выводится **общее количество** строк таблицы,

во втором – количество строк, имеющих **ненулевые значения** по заданному столбцу таблицы.

Применение модификатора DISTINCT позволяет подсчитать количество **уникальных ненулевых** значений (например – количество оценок по категориям.

Запросы к одной таблице БД

Создание вычисляемых столбцов (операция **расширения**)

Использование агрегатных функций в вычисляемых столбцах

Важно!

Запросы, вычисляющие агрегатные функции по всей таблице, всегда возвращают одну строку, содержащую итоговые данные. Поэтому в списке выражений **за словом SELECT**, могут присутствовать **только эти функции** или выражения с их использованием.

Фразы типа WHERE и т.д. здесь недопустимы!

Для того, чтобы применять агрегатные функции не ко всей таблице, а к отдельным группам строк, над таблицей выполняется операция **группировки**.

Раздел 4. Основы языка SQL

Тема 4.1
Язык запросов (Data Query Language)

Вопросы лекции:

1. Общая характеристика операторов SQL
2. Синтаксис команды SELECT
3. Запросы к одной таблице БД
4. Операция расширения
5. Операция группировки

Запросы к одной таблице БД

Группировка и агрегатные функции (операция **группировки**)

Использование агрегатных функций по **группам строк**.

При группировке формируются группы с одинаковыми значениями в столбце группировки (или нескольких столбцах).

Запрос возвращает **столько строк**, **сколько** получилось **групп**. Фактически оно совпадает с количеством **уникальных** значений в столбце группировки, поэтому группировку принято выполнять по столбцам, содержащим большое количество повторяющихся значений, тогда количество групп будет значительно меньше, чем количество строк в таблице.

Например, в таблице с **кодами и оценками** студентов группировка по коду студента позволяет получить средний балл и количество оценок по каждому из студентов:

```
SELECT cod_st, avg(mark) avg_mark, count(mark) count_mark
FROM marks
GROUP BY cod_st
```

Запросы к одной таблице БД

Группировка и агрегатные функции (операция **группировки)**

Правило для запросов с группировкой – в списке выражений, который следует за словом SELECT, могут быть только выражения из фразы GROUP BY и агрегатные функции (или выражения на их основе).

Условия отбора групп:

После группировки можно отобрать группы, удовлетворяющие определенному условию. Для этого служит фраза HAVING.

Фраза HAVING может использоваться только после фразы GROUP BY.

Например, пусть требуется выбрать тех студентов, у которых средний бал >4. Текст запроса имеет вид:

```
SELECT cod_st, AVG(mark) avg_mark
FROM marks
GROUP BY cod_st
HAVING AVG(mark)>4
```