1.1.1. Команда выборки данных

Одной из главных функций SQL считается выполнение выборки. Поэтому рассмотрим этот процесс подробно.

Выборка – это обращение к БД с целью извлечь данные в виде, удобном для пользователя. Для выборки применяются запросы к БД. Иногда в SQL выделяют даже раздел, который называют языком запросов к данным DQL (Data Query Language). Фактически этот раздел языка ANSI SQL представлен только одной командой – SELECT. Но эта команда достаточно обширна. Она является ядром языка SQL. и используется для реализации операций проекции, ограничения, расширения.

Для пользователя РБД оператор SELECT является, пожалуй, одним из наиболее главных и полезных операторов языка SQL. Этот оператор позволяет производить:

– выбор данных (отбор записей и полей);
– вычисления и сравнения;
– упорядочение записей при выводе содержимого таблиц;
– группирование данных и применение к этим группам специальных групповых операций.

Источником данных (ИД) для запроса могут быть РТ или ранее созданные запросы.

После выполнения запроса на выборку создается набор записей в виде временной рабочей таблицы, содержащей данные, которые отбираются из ИД согласно заданным условиям.

В большинстве случаев с набором записей можно работать точно так же, как с таблицей: можно просматривать, выбирать и даже обновлять информацию. Однако в отличие от реальной таблицы, этот набор записей физически не существует в БД. Запрос с точки зрения пользователя можно рассматривать как шаблон, который создает набор записей из ИД только во время своего выполнения. При отсутствии данных результат представляет пустой набор записей.

Синтаксис инструкции SELECT определяется конструкциями, используемыми при реализации функций выборки. Инструкция в общем виде использует пять частей, которые делятся на две группы:

основная часть
SELECT [предикат] <список полей> выбрать
FROM <список ИД> из
дополнительные части
[WHERE <спецификация выбора записей>] где
[[GROUP BY <спецификация группировки>] группируя по
[HAVING <спецификация выбора групп>]] имея
[ORDER BY <спецификация сортировки>] упорядочить по.

Рассмотрим синтаксис инструкции по частям. При этом учтем следующее:

– инструкция языка SQL – это предложение (команда, оператор);

– отдельные составные части инструкции (список полей, спецификация) – это опции предложения;

– любая спецификация – это фраза, отвечающая требованиям синтаксиса предложения.

Из синтаксиса видно, что

– основная часть команды SELECT …FROM обязательна;

– опция предикат необязательна

– дополнительные части **WHERE, GROUP BY, ORDER BY н**еобязательны, они следуют за FROM;

– опция HAVING не может применяться без GROUP BY.

Рассмотрим опцию

SELECT [предикат] <список полей>.

Эта опция в основной части предложения позволяет выбрать данные из указанных столбцов и выполнить перед выводом их преобразование в соответствии с указанными выражениями или функциями.

Предикат предназначен для ограничения числа возвращаемых записей.

Предикат::=

[[ALL]	все (обычно по умолчанию)
| DISTINCT	позволяет отобрать различные записи, исключает записи, содержащие повторяющиеся данные в отдельных полях; в результат включаются только уникальные значения каждого из полей, указанных в списке
| TOP N]	отображение N первых записей.

Список полей предназначен для определения тех полей, которые отражаются в результате.

<Список полей> ::=

{ эл_SELECT элемент списка

[, эл2_SELECT] ...}

Отметим, что для разделения элементов списка используются запятые.

Рассмотрим синтаксис конструкции для элемента списка

эл_SELECT::=

[ИД.]

*| отбор всех полей

| значение | SQL_функция | системная_переменная

В свою очередь

Значение ::=

[ИД.] поле	имя поля
[AS псевдоним]	заголовок поля
[,[ИД.] полеК	имя К-ого поля
[AS псевдонимК] …]	заголовок К-ого поля

| (выражение)
| переменная
| константа

Текстовые константы должны заключаться в апострофы или двойные кавычки.

Выражение::= ({[[+] |–] {значение | функция_СУБД}[+|–|*|**]} ...)

Функция_СУБД – это любая существующая функция. Для преобразования или вычисления значений могут применяться общеизвестные функции или выражения, содержащие такие функции.

В качестве функции_СУБД могут также применяться специальные групповые (агрегирующие, статистические) SQL-функции, которые определяют одно значение по множеству значений поля-аргумента.

SQL_функция ::=

{SUM	сумма
| AVG	среднее значение
| MIN	минимальное значение
| MAX	максимальное значение
| COUNT}	количество
([[ALL]	
| DISTINCT]	
[ИД.] поле)	аргумент
| ([ALL] выражение).	

Из SQL-функций можно составлять любые выражения, но их вложенность не допускается. Если вычисляются SQL-функции или выражения, содержащие такие функции, в список полей могут только те поля, которые являются аргументами SQL-функций. Наличие других полей в списке не допустимо.

Ключевое слово DISTINCT используется для исключения полей-дубликатов перед применением функций. Для функций MAX и MIN это слово излишне.

Для подсчета всех без исключения записей в таблице, включая дубликаты, используется специальная функция

SQL_функция ::= COUNT(*).

С этой функцией слово DISTINCT не допускается.

Отметим специфику обработки неопределенных (пустых) значений (Null-значения). Если значение аргумента – пустое множество, то

– при наличии слова DISTINCT эти записи не учитываются;
– функция COUNT возвратит значение нуль;
– функция COUNT (*) обработает все записи так же, как обычные значения;
– другие функции обычно возвращают Null-значение.

Опция **FROM <список ИД>** определяет перечень тех ИД, из которых берутся поля для включения в результат запроса.

Рассмотрим примеры применения команды SELECT с опциями SELECT

и FROM. В качестве списка ИД будем рассматривать только один ИД, а именно тС. Таким образом,

Список ИД::= тС.

В процессе изучения команды выборки студент должен самостоятельно определить результаты выполнения команд, т.е. составить заголовок таблицы-результата и ее тело. Рекомендуется также указать, какую команду РА реализует соответствующая команда.

Итак, наша первая команда выборки на SQL

10) SELECT *
FROM тС → выборка всех сотрудников, выводятся все поля и все записи из тС, порядок вывода полей соответствует структуре тС, результат приведен в табл. 3.2.

Определение “выборка сотрудников” конечно означает не выборку самих сотрудников, а выборку информации о них.

Так как в качестве ИД взята тС, опция FROM тС будет присутствовать во всех командах данного подраздела

Приведем примеры реализации операции проекции.

11) SELECT Код_с, Фам, Имя, От FROM тС
12) SELECT Фам FROM тС
13) SELECT DISTINCT Фам FROM тС

Приведем примеры реализации операции расширения путем формирования вычисляемых полей.

Рассчитаем возраст сотрудников по формуле
год(дата_текущая) – год(д_рожд).

14) SELECT Код_с, Фам,
(год(дата_текущая) – год(д_рожд)) “ вычисление возраста
FROM тС

Третье поле фактически не имеет имени. Задание имени результирующего поля не обязательно, но рекомендуется.

В данном предложении год( ) – это встроенная функция конкректной СУБД. Обычно эта функция реализуется как year( ), а текущая дата – как date().

Таким образом обычно для реальных СУБД

15) SELECT Код_с, Фам,
year(date()) – year(д_рожд)
FROM тС

или с заданием заголовка

16) SELECT Код_с, Фам,
’возраст=’, year(date()) – year(д_рожд)
FROM тС

Наиболее удобный результат дает команда

17) SELECT Код_с, Фам,
year(date()) – year(д_рожд) As Возраст
FROM тС

18) SELECT Фам, Count(Код_с) FROM тС

Данная команда ошибочна, так как список полей наряду с SQL-функцией содержит поле Фам, которое не является аргументом SQL-функции. Правильный синтаксис имеет команда

19) SELECT Count(Код_с) FROM тС. 10

20) SELECT Count(Код_с) AS КвоС, 10
Max([Д_рожд]) AS МахДр,
Count([Д_ув]) AS КвоУвол, 3
Min([Д_ув]) AS MinДув,
FROM тС.

Рассмотрим синтаксис других опций.

Сначала рассмотрим опцию, которая используется для упорядочения записей

ORDER BY <спецификация сортировки>.

Спецификация сортировки задается фразой, определяющей список полей для упорядочения. Фраза имеет следующий синтаксис:

<спецификация сортировки>::=
{[ИД.] поле | ном_элем_SELECT}
[[ASC] | DESC]
[,{[ИД.] поле2 | ном_элем_SELECT2}
[[ASC] | DESC]] ...

Рассмотрим примеры применения опции ORDER.

21) SELECT Код_с, Фам, Имя, От FROM тС
ORDER BY Код_с

Обычно применяется более удобная для пользователя сортировка списков по трем полям

22) SELECT Код_с, Фам, Имя, От FROM тС
ORDER BY Фам, Имя, От.

Для реализации операции ограничения за счет задания условий отбора (выбора) записей используется опция

WHERE <спецификация выбора записей>.

Спецификация выбора записей задается фразой, которая включает набор условий для отбора записей

<спецификация выбора записей>::=
[NOT] WHERE_усл1 [[AND|OR][NOT] WHERE_усл2]...

Как видно из синтаксиса, критерий отбора строк может формироваться из одного условия или из нескольких условий, соединенных логическими операторами AND, OR, [NOT].

Для случая двух условий назначение логических операторов следующее:

AND – должны удовлетворяться оба условия ;

OR – должно удовлетворяться одно из условий;

AND NOT – должно удовлетворяться первое условие (усл1) и не должно

второе условие (усл2);

OR NOT – или должно удовлетворяться усл1 или не должно удовлетворяться усл2.

При отборе существует приоритет AND над OR: сначала выполняются все операции AND и только после этого выполняются операции OR.

Для условия отбора можно записать следубщий синтаксис

WHERE_усл ::=

знач { = | <> | < | <= | | = }

{ знач | (подзапрос) }

знач_1 [NOT] BETWEEN знач_2 AND знач_3 между

знач [NOT] IN { (конст [,конст]...) | (подзапрос) } принадлежит

знач IS [NOT] NULL не определено

[ИД.] поле [NOT] LIKE 'строка_символов' похоже на

[ESCAPE 'символ'] не включает

EXISTS (подзапрос) существует

При сравнении обычно действуют следующие правила обработки условий:

– числа сравниваются алгебраически; отрицательные числа считаются меньшими, чем положительные, независимо от их абсолютной величины;

– строки символов сравниваются в соответствии с их представлением в коде, используемом в конкретной СУБД, например, в коде ASCII;

– если сравниваются две строки символов, имеющих разные длины, то перед выполнением операции сравнения их длина уравнивается до большей за счет добавления пробелов справа в короткой строке.

Для получения желаемого результата условия отбора должны быть заданы в правильном порядке, который можно организовать введением скобок.

Рассмотрим примеры применения опции WHERE.

23) SELECT Фам, Д_рожд FROM тС

WHERE Д_рожд = '01-01-1980' символьная строка

24) SELECT Фам, Д_рожд FROM тС

WHERE Д_рожд BETWEEN '01-01-1980' AND '31-12-1980'

В операторе BETWEEN знач_2 должно быть меньше или равно знач_3.

25) SELECT Фам FROM тС

WHERE Д_рожд = '01-01-1980' AND Город = “Донецк”;

26) SELECT * FROM тС

WHERE Город= “Донецк” OR Город= “Макеевка”.

В последнем примере для краткой записи последовательности отдельных сравнений, соединенных операторами OR, можно применить форму IN

27) SELECT * FROM тС

WHERE Город IN (“Донецк”, “Макеевка”).

При использовании в условии формы LIKE 'строка_символов' интерпретация зависит от заданных символов так:

– символ _ (подчеркивание) заменяет любой одиночный символ;

– символ % (процент) заменяет любую последовательность из N символов, где N может быть нулем;

– все другие символы означают сами себя.

Например

28) SELECT * FROM тС
WHERE Город LIKE “М%”

Очень редко, но в поля вносятся знаки “_” и “%”. В этом случае для их поиска применяются дополнительные escape-символы, которые должны предшествовать знакам.

Рассмотрим образец поиска вида

LIKE ‘_/_a’ ESCAPE ‘/’.

В этом выражении символ '/’ объявлен escape-символом. Первый символ “_” в заданном шаблоне поиска будет соответствовать, как и ранее, любому символу в проверяемой строке. Второй символ “_”, следующий после escape-символа, будет интерпретироваться как обычное подчеркивание. Аналогично, символ 'a' будет интерпретироваться как буква а.

Для проверки содержимого поля на наличие в нем Null-значения предназначены специальные операторы IS NUll (является пустым) и IS NOT NULL (является не пустым). Другие операторы сравнения использовать нельзя.

29) SELECT * FROM тС
WHERE Д_ув Like ' %' результат Null

30) SELECT * FROM тС
WHERE Д_ув IS NULL

При использовании функций происходит расчет их значений по всему набору записей, определенных условием отбора

31) SELECT Фам, COUNT(Код_с) FROM тС
WHERE Город = “Донецк” ошибка

32) SELECT COUNT(Код_с) As Количество FROM тС
WHERE Город = “Донецк”

Рассмотрим опцию, которая применяется для группировки записей

GROUP BY <спецификация группировки>.

Спецификация группировки записей используется при создании группировочных запросов и задается фразой вида

<спецификация группировки>::=

[ИД.] поле имя поля

[,[ИД.] полеК] ... имя К-го поля

Группирование записей инициирует перекомпоновку записей по группам, каждая из которых имеет одинаковое значение в полях, включенных в спецификации группировки.

33) SELECT Фам FROM тС GROUP BY Фам

В результате этой команды происходит исключение записей-дубликатов.

34) SELECT Город FROM тС GROUP BY Город.

К группам данных можно применить агрегирующие SQL-функции. Для

этого их нужно указать в списке полей вывода. Применение SQL-функций приводит к замене всех значений группы на единственное значение, определенной SQL-функцией (сумма, количество и т.п.). Группирование записей позволяет реализовать реляционную операцию подведения итогов.

35) SELECT Фам, Count(Код_с) AS Кол FROM тС
GROUP BY Фам

В этой команде происходит группировка по полю Фам, которое включено в список полей вывода.

36) SELECT Город, COUNT(Код_с) FROM тС
GROUP BY Город

Надо учитывать, что опция GROUP BY не предполагает упорядочение. Поэтому рекомендуется одновременно с ней применять и опцию ORDER BY.

37) SELECT Город, COUNT(Код_с) FROM тС
GROUP BY Город
ORDER BY Город

Рассмотрим опцию отбора групп записей:

HAVING <спецификация выбора групп>.

В результат попадают только те группы, которые удовлетворяют заданной спецификации выбора групп. Ее синтаксис подобен синтаксису спецификации выбора записей:

<спецификация выбора групп>::=
[NOT] HAVING_усл [[AND|OR][NOT] HAVING_усл2]...

Синтаксис HAVING_усл почти не отличается от синтаксиса WHERE_усл. Только наряду со значениями в этих условиях могут использоваться SQL-функции.

HAVING_усл ::=
знач { = | < | < | <= | | = }
{ знач | (подзапрос) | SQL_функция }
{знач_1 | SQL_функция_1} [NOT] BETWEEN
{знач_2 | SQL_функция_2} AND {знач_3 | SQL_функция_3}
{знач | SQL_функция} [NOT] IN { (конст [,конст]...) | (подзапрос) }
{знач | SQL_функция} IS [NOT] NULL
[табл.] поле [NOT] LIKE 'строка_символов' [ESCAPE 'символ']
EXISTS (подзапрос)

Рассмотрим пример с опцией HAVING.

38) SELECT * FROM тС
GROUP BY Город
HAVING COUNT (*) < 5