**СИНТАКСИС И КОМАНДА ССЫЛКИ**

Это приложение содержит более краткое описание различных команд SQL. Цель состоит в том, чтобы дать вам быструю и точную ссылку и определение SQL. Первый раздел этого приложения определяет элементы, используемые для создания команд SQL; второй, подробности синтаксиса и предложения с кратким описанием самих команд. Далее показаны стандартные условные обозначения ( они называются BNF условиями):

 Ключевые cлова набираются в верхнем регистре.

 SQL и другие специальные условия заключаются в угловые скобки и набираются курсивом.(< *and* >)

 Необязательные части команд находятся в квадратных скобках ( [and] ).

 Многоточие ( .... ) указывает на то что предшествующая часть команды может повторяться любое число раз.

 Вертикальна полоса ( | ) означает - то, что ей предшествует может быть заменено на то что следует за ней.

 Фигурные Скобки ( {and} ) указывают - все что внутри них, должно быть расценено как целое, для оценки других символов ( например, вертикальных полос или эллипсов ).

 Двойное двоеточие и равняется (:: = ) означают - то что следует за ними является определением того что им предшествует.

Кроме того, мы будем использовать следующую последовательность (.,.. ) чтобы указывать, что предшествующее этому может повторяться любое число раз с индивидуальными событиями отделяемыми запятыми. Атрибуты которые не являются частью официального стандарта будут отмечены как (\*нестандартные\*) в описании.

**ОБРАТИТЕ ВНИМАНИЕ: Терминология которую мы используем здесь, не официальна терминология ANSI. Официальная терминология может вас сильно запутать, поэтому мы несколько ее упростили.  
  
По этой причине, мы иногда используем условия отличающиеся от ANSI, или используем те же самые условия но несколько по-другому. Например, наше определение - < predicate > отличается от используемой в ANSI комбинации стандартного определения < predicate > с < search condition >.**

**SQL ЭЛЕМЕНТЫ**

Этот раздел определяет элементы команд SQL. Они разделены на две категории: **Основные элементы языка**, и **Функциональные элементы языка**.

**Основные элементы** - это создаваемые блоки языка; когда SQL исследует команду, то он сначала оценивает каждый символ в тексте команды в тер- минах этих элементов. Разделители< separator > отделяют одну часть команды от другой; все что находится между разделителями < separator > обрабатывается как модуль. Основываясь на этом разделении, SQL и интерпретирует команду.

**Функциональные элементы** - это разнообразные вещи отличающиеся от ключевых слов, которые могут интерпретироваться как модули. Это - части команды, отделяемые с помощью разделителей < separator >, имеющих специальное значение в SQL. Некоторые из них являются специальными для определенных команд и будут описаны вместе с этими командами по- зже, в этом приложении. Перечисленное здесь, является общими элементы для всех описываемых команд. Функциональные элементы могут определяться в терминах друг друга или даже в собственных терминах. Например, предикат < predicate >, наш последний и наиболее сложный случай, содержит предикат внутри собственного определения. Это потому, что предикат < predicate > использующий AND или OR может содержать любое число предикатов < predicate > которые могут работать автономно. Мы представляли вам предикат < predicate > в отдельной секции в этом приложении, из-за разнообразия и сложности этого функционального элемента языка. Он будет постоянно присутствовать при обсуждении других функциональных частей команд.

**ЭЛЕМЕНТЫ ЯЗЫКА БЕЙСИКА**

ЭЛЕМЕНТ ОПРЕДЕЛЕНИЕ

< separator > < comment > | < space > | < newline >

< comment > --< string > < newline >

< space > пробел

< newline > реализационно-определяемый конец символьной строки

< identifier > < letter >[{< letter or digit > | < underscore}... ]

< ИМЕЙТЕ ВВИДУ: Следу строгому стандарту ANSI, символы

должны быть набраны в верхнем регистра, а идентификатор

< identifier > не должен быть длиннее 18-ти символов.

ЭЛЕМЕНТ ОПРЕДЕЛЕНИЕ

< underscore > -

< percent sign > %

< delimiter > любое из следующих: , ( ) < > . : = + " - | <>

> = < = или < string >

< string > [любой печатаемый текст в одиночных кавычках]

Примечание: В < string >, две последовательных одиночных

кавычки ( ' ' ) интерпретируются как одна ( ' ).

< SQL term > окончание, зависящее от главного языка.

(\*только вложенный\*)

**ФУНКЦИОНАЛЬНЫЕ ЭЛЕМЕНТЫ**

Следующая таблица показывает функциональные элементы команд SQL

и их определения:

ЭЛЕМЕНТ ОПРЕДЕЛЕНИЕ

< query > Предложение SELECT

< subquery > Заключенное в круглых скобках предложение

SELECT внутри другого условия, которое, фактически,

оценивается отдельно для каждой строки-кандидата

другого предложения.

< value expression > < primary >

| < primary > < operator > < primary >

| < primary > < operator > < value expression >

< operator > любое из следующих: + - / \*

< primary > < column name >

| < literal >

| < aggregate function >

| < built-in constant >

| < nonstandard function >

< literal > < string > | < mathematical expressio

ЭЛЕМЕНТ ОПРЕДЕЛЕНИЕ

< built-in constant > USER | < implementation-dehned constant >

< table name > < identifier >

< column spec > [< table name > | < alias >.]< column name >

< grouping column > < column spec > | < integer >

< ordering column > < column spec > | < integer >

< colconstraint > NOT NULL | UNIQUE

| CHECK (< predicate >)

| PRIMARY KEY

| REFERENCES < table name >[(< column name >)]

< tabconstraint > UNIQUE (< column list >)

| CHECK (< predicate >)

| PRIMARY KEY (< column list >)

| FOREIGN KEY (< column list >)

REFERENCES < table name >[(< column list >)]

< defvalue > ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ = < value expression >

< data type > Допустимый тип данных ( См. [Приложение B](http://www.sql.ru/docs/sql/u_sql/b.shtml)

для описания типов обеспечиваемых ANSI или

[Приложение C](http://www.sql.ru/docs/sql/u_sql/c.shtml) для других общих типов. )

< size > Значение зависит от < data type >( См. [Приложение B](http://www.sql.ru/docs/sql/u_sql/b.shtml). )

< cursor name > < identifier >

< index name > < identifier >

< synonym > < identifier >(\*nonstandard\*)

< owner > < Authorization ID >

< column list > < column spec > .,..

< value list > < value expression > .,..

< table reference > { < table name > [< alias >] } .,..

**ПРЕДИКАТЫ**

Следующее определяет список различных типов предиката < predicate > описанных на следующих страницах:

**< predicate > ::= [NOT]**

{ < comparison predicate >

| < in predicate >

| < null predicate >

| < between predicate >

| < like predicate >

| < quantified predicate >

| < exists predicate > }

[ANDI OR < predicate > ]

< predicate > - это выражение, которое может быть верным, неверным, или неизвестным, за исключением < exists predicate > и < null predicate >, которые могут быть только верными или неверными.

Будет получено неизвестно если NULL значения предотвращают вывод полученного ответа. Это будет случаться всякий раз, когда NULL значение сравнивается с любым значением. Стандартные операторы Буля - AND, OR, и NOT - могут использоваться с предикатом. NOT верно = неверно, NOT неверно = верно, а NOT неизвестно = неизвестно. Результаты AND и OR в комбинации с предикатами, показаны в следующих таблицах:

AND

AND Верно Неверно Неизвестно

Верно верно неверно неизвестно

Неверно неверно неверно неверно

Неизвестно неизвестно неверно неизвестно

OR

OR Верно Неверно Неизвестно

Верно верно верно верно

Неверно верно неверно неизвестно

Неизвестно верно неизвестно неизвестно

Эти таблицы читаются способом наподобие таблицы умножения: вы объединяете верные, неверные, или неизвестные значения из строк с их столбцами чтобы на перекрестье получить результат. В таблице AND, например, третий столбец (Неизвестно) и первая строка (Верно) на пересечении в верхнем правом углу дают результат - неизвестно, другими словами: Верно AND Неизвестно = неизвестно. Порядок вычислений определяется круглыми скобками. Они не представляются каждый раз. NOT оценивается первым, далее AND и OR. Различные типы предикатов < predicate > рассматриваются отдельно в следующем разделе.

**< comparison predicate > (предикат сравнения)**

***Синтаксис***

< value expresslon > < relational op > < value expresslon > |  
< subquery >   
< relatlonal op > :: =  
=  
| <   
| >   
| <   
| >=   
| < >

Если либо < value expression > = NULL, либо < comparison predicate > = неизвестно; другими словами, это верно если сравнение верно или неверно если сравнение неверно.  
< relational op > имеет стандартные математические значения для числовых значений; для других типов значений, эти значения определяются конкретной реализацией.  
Оба < value expression > должны иметь сравнимые типы данных. Если подзапрос < subquery > используется, он должен содержать одно выражение < value expression > в предложении SELECT, чье значение будет заменять второе выражение < value expression > в предикате сравнения < comparision predicate >, каждый раз когда < subquery > действительно выполняется.

**< between predicate >**

***Синтаксис***

< value expression > [NOT] BETWEEN < value expression >  
AND < value expression >

< between predicate > - A BETWEEN B AND C , имеет такое же значение что и < predicate > - ( A >= B AND < = C). < between predicate > для которого A NOT BETWEEN B AND C, имеет такое же значение что и NOT ( BETWEEN B AND C). < value expression > может быть выведено с помощью нестандартного запроса < subquery > ( \*nonstandard\* ).

**< in prediicate >**

***Синтаксис***

< value expression > [NOT] IN < value list > | < subquery >

Список значений < value list > будет состоять из одного или более перечисленных значений в круглых скобках и отделяемых запятыми, которые имеют сравнимый с < value expression > тип данных. Если используется подзапрос < subquery >, он должен содержать только одно выражение < value expression > в предложении SELECT ( возможно и больше, но это уже будет вне стандарта ANSI ). Подзапрос < subquery > фактически, выполняется отдельно для каждой строки-кандидата основного запроса, и значения которые он выведет, будут составлять список значений < value list > для этой строки. В любом случае, предикат < in predicate > будет верен если выражение < value expression > представленное в списке значений < value list >, если не указан NOT. Фраза A NOT IN (B, C) является эквивалентом фразы NOT (A IN (B, C)).

**< like predicate >**

***Синтаксис***

< charvalue > [NOT] LIKE < pattern > [ESCAPE  
< escapechar >]

< charvalue > - это любое \*нестандартное\* выражение < value expression > алфавитно-цифрового типа. < charvalue > может быть, в соответствии со стандартом, только определенным столбцом < column spec >. Образец < pattern > состоит из строки которая будет проверена на совпадение с < charvalue >. Символ окончания < escapechar > - это одиночный алфавитно-цифровой символ. Совпадение произойдет, если верны следующие условия :

 Для каждого символа подчеркивания < underscore > в образце < pattern > которая не предшествует символу окончания < escapechar >, имеется один соответствующий ему символ < charvalue >.

 Для каждого < percent sign > в образце < pattern >, который не предшествует < escapechar >, имеются нули или более соответствующие символы в < charvalue >.

 Для каждого < escapechar > в < pattern > который не предшествует другому < escapechar >, нет никакого соответствующего символа в < charvalue >.

 Для каждого иного символа в < pattern >, один и тот же символ устанавливается у соответствующей отметке в < charvalue >.  
  
Если совпадение произошло, < like predicate > - верен, если не был указан NOT. Фраза NOT LIKE 'текст' - эквивалентна NOT ( A LIKE 'текст').

**< null predicate >**

***Синтаксис***

< column spec > IS [NOT] NULL

< column spec > = IS NULL, если NULL значение представлено в этом столбце. Это сделает < null predicate > верным если не указан NULL. Фраза < column spec > IS NOT NULL, имеет тот же результат что и NOT ( < column spec > IS NULL ).

**< quantified predicate >**

***Синтаксис***

< value expression > < relational op >  
< quantifier > < subquery >   
< quantifier > :: = ANY | ALL | SOME

Предложение SELECT подзапроса < subquery > должно содержать одно и только одно выражение значения < value expression >. Все значения выведенные подзапросом < subquery > составляют набор результатов < result set >. < value expression > сравнивается, используя оператор связи < relational operator >, с каждым членом набора результатов < result set >. Это сравнение оценивается следующим образом:

 Если < quantifier > = ALL, и каждый член набора результатов < result set > делает это сравнение верным, < quantified predicate > - верен.

 Если < quantifier > = ANY, и имеется по крайней мере один член из набора результатов < result set > , который делает верным это сравнение, то < quantified predicate > является верным.

 Если набор результатов < result set > пуст, то < quantified predicate > верен, если < quantifier > = ALL , и неверен если иначе.

 Если < quantifier > = SOME, эффект - тот же что и для ANY.

 Если < quantified predicate > не верен и не неверен, он - неизвестен.

**< exists predicate >**

***Синтаксис:***

EXISTS (< subquery >)

Если подзапрос < subquery > выводит одну или более строк вывода, < exists predicate > - верен; и неверен если иначе.

**SQL КОМАНДЫ**

Этот раздел подробно описывает синтаксис различных команд SQL. Это даст вам возможность быстро отыскивать команду, находить ее синтаксис и краткое описание ее работы.   
  
ИМЕЙТЕ ВВИДУ Команды которые начинаются словами - EXEC SQL, а также команды или предложения заканчивающиеся словом - могут использоваться только во вложенном SQL.

**BEGIN DECLARE SECTION (НАЧАЛО РАЗДЕЛА ОБЪЯВЛЕНИЙ)**

***Синтаксис***

EXEC SQL BEGIN DECLARE SECTION < SQL term > < host-language variable declarations > EXEC SQL END DECLARE SECTION < SQL term >

Эта команда создает раздел программы главного языка для объявления в ней главных переменных, которые будут использоваться во вкладываемых операторах SQL. Переменна SQLCODE должна быть включена как одна из объявляемых переменных главного языка.

**CLOSE CURSOR (ЗАКРЫТЬ КУРСОР)**

***Синтаксис***

EXEC SQL CLOSE CURSOR < cursor name > < SQL term >;

Эта команда указывает курсору закрыться, после чего ни одно значение не сможет быть выбрано из него до тех пор пока он не будет снова открыт.

**COMMIT (WORK) (ФИКСАЦИЯ (ТРАНЗАКЦИИ))**

***Синтаксис***

COMMIT WORK;

Эта команда оставляет неизменными все изменения сделанных в базе данных, до тех пор пока начавшаяся транзакция не закончится, и не начнется новая транзакция.

**CREATE INDEX (СОЗДАТЬ ИНДЕКС)**

(\*NONSTANDARD\*) (НЕСТАНДАРТНО)

***Синтаксис***

CREATE [UNIQUE] INDEX < Index name >  
ON < table name > (< column list >);

Эта команда создает эффективный маршрут с быстрым доступом для поиска строк содержащих обозначенные столбцы. Если UNIQUE - указана, таблица не сможет содержать дубликатов(двойников) значений в этих столбцах.

**CREATE SYNONYM (\*NONSTANDARD\*)  
(СОЗДАТЬ СИНОНИМ) ( \*НЕСТАНДАРТНО\* )**

***Синтаксис***

CREATE IPUBLICl SYNONYM < synonym > FOR  
< owner >.< table name >;

Эта команда создает альтернативное(синоним) им для таблицы. Синоним принадлежит его создателю, а сама таблица, обычно другому пользователю. Используя синоним, его владелец может не ссылаться к таблице ее полным ( включая им владельца ) именем. Если PUBLIC - указан, синоним принадлежит каталогу SYSTEM и следовательно доступен всем пользователям.

**CREATE TABLE (СОЗДАТЬ ТАБЛИЦУ)**

***Синтаксис***

CREATE TABLE < table name >  
({< column name > < data type >[< size >]  
[< colconstralnt > . . .]   
[< defvalue >]} . , . . < tabconstraint > . , . .);

Команда создает таблицу в базе данных. Эта таблица будет принадлежать ее создателю. Столбцы будут рассматриваться в поименном порядке. < data type > - определяет тип данных который будет содержать столбец. Стандарт < data type > описывается в [Приложении B](http://www.sql.ru/docs/sql/u_sql/b.shtml); все прочие используемые типы данных < data type >, обсуждались в [Приложении C](http://www.sql.ru/docs/sql/u_sql/c.shtml). Значение размера < size > зависит от типа данных < data type >.  
< colconstraint > и < tabconstraint > налагают ограничения на значения ко торые могут быть введены в столбцу.   
< defvalue > определяет значение(по умолчанию) которое будет вставлено автоматически, если никакого другого значения не указано для этой строки. ( См. [Главу 17](http://www.sql.ru/docs/sql/u_sql/ch17.shtml) для подробностей о самой команде CREATE TABLE и[Главы 18](http://www.sql.ru/docs/sql/u_sql/ch18.shtml) И [19](http://www.sql.ru/docs/sql/u_sql/ch19.shtml) для подробностей об ограничениях и о < defvalue > ).

**CREATE VIEW (СОЗДАТЬ ПРОСМОТР)**

***Синтаксис***

CREATE VIEW < table name >  
AS < query >   
[WITH CHECK OPTION];

Просмотр обрабатывается как люба таблица в командах SQL. Когда команда ссылается на имя таблицы < table name >, запрос < query > выполняется, и его вывод соответствует содержанию таблицы указанной в этой команде.   
Некоторые просмотры могут модифицироваться, что означает, что команды модификации могут выполняться в этих просмотрах и передаваться в таблицу, на которую была ссылка в запросе < query >. Если указано предложение WITH CHECK OPTION, эта модификация должны также удовлетворять условию предиката < predicate > в запросе < query >.

**DECLARE CURSOR (ОБЪЯВИТЬ КУРСОР)**

***Синтаксис***

EXEC SQL DECLARE < cursor name > CURSOR FOR   
< query >< SQL term >

Эта команда связывает им курсора < cursor name >, с запросом < query >. Когда курсор открыт ( см. OPEN CURSOR ), запрос < query > выполняет ся, и его результат может быть выбран(командой FETCH) для вывода. Если курсор модифицируемый, таблица на которую ссылается запрос < query >, может получить изменение содержания с помощью операции модификации в курсоре ( См. [Главу 25](http://www.sql.ru/docs/sql/u_sql/ch25.shtml) о модифицируемых курсорах ).

**DELETE (УДАЛИТЬ)**

***Синтаксис***

DELETE FROM < table name >  
{ [WHERE < predicate >]; }   
| WHERE CURRENT OF < cursorname >< SQL term >

Если предложение WHERE отсутствует, ВСЕ строки таблицы удаляются. Если предложение WHERE использует предикат < predicate >, строки, ко торые удовлетворяют условию этого предиката < predicate > удаляются. Если предложение WHERE имеет аргумент CURRENT OF(ТЕКУЩИЙ) в имени курсора < cursor name >, строка из таблицы < table name > на ко торую в данный момент имеется ссылка с помощью имени курсора < cursor name > будет удалена. Форма WHERE CURRENT может использоваться только во вложенном SQL, и только с модифицируемыми курсорами.

**EXEC SQL (ВЫПОЛНИТЬ SQL)**

***Синтаксис***

EXEC SQL < embedded SQL command > < SQL term >

EXEC SQL используется чтобы указывать начало всех команд SQL, вложенных в другой язык.

**FETCH (ВЫБОРКА)**

***Синтаксис***

EXEC SQL FETCH < cursorname >  
INTO < host-varlable llst >< SQL term >

FETCH принимает вывод из текущей строки запроса < query >, вставляет ее в список главных переменных < host-variable list >, и перемещает кур сор на следующую строку. Список < host-variable list > может включать переменную indicator в качестве целевой переменной ( См. [Главу 25](http://www.sql.ru/docs/sql/u_sql/ch25.shtml). )

**GRANT (ПЕРЕДАТЬ ПРАВА)**

***Синтаксис (стандартный)***

GRANT ALL [PRIVILEGES]  
| {SELECT   
| INSERT   
| DELETE   
| UPDATE [(< column llst >)]   
| REFERENCES [(< column llst >)l } . , . .   
ON < table name > . , . .  
TO PUBLIC | < Authorization ID > . , . .  
[WITH GRANT OPTION];

Аргумент ALL(ВСЕ), с или без PRIVILEGES(ПРИВИЛЕГИИ), включает каждую привилегию в список привилегий. PUBLIC(ОБЩИЙ) включает всех существующих пользователей и всех созданных в будущем. Эта команда дает возможность передать права для выполнения действий в таблице с указанным именем. REFERENCES позволяет дать права чтобы использовать столбцы в списке столбцов < column list > как родительский ключ для внешнего ключа. Другие привилегии состоят из права выполнять команды для которых привилегии указаны их именами в таблице. UPDATE, подобен REFERENCES, и может накладывать ограничения на определенные столбцы. GRANT OPTION дает возможность передавать эти привилегии другим пользователям.

***Синтаксис ( нестандартный )***

GRANT DBA   
| RESOURCE   
| CONNECT ... .   
TO < Authorization ID > . , . .   
[IDENTIFIED BY > password >

CONNECT дает возможность передавать право на регистрации и некоторые другие ограниченные права. RESOURCE дает пользователю право создавать таблицы. DBA дает воз- можность передавать почти все права. IDENTIFIED BY используется вместе с CONNECT, для создания или изменения пароля пользователя.

**INSERT (ВСТАВКА)**

***Синтаксис***

INSERT INTO < table name > (< column llst >)  
VALUES (< value llst >) I < query >;

INSERT создает одну или больше новых строк в таблице с именем < table name >. Если используется предложение VALUES , их значения вставляются в таблицу с именем < table name >. Если запрос < query > указан, каждая строка вывода будет вставлена в таблицу с именем < table name >. Если список столбцов < column list > отсутствует, все столбцы таблицы < table name >, принимаются в упорядоченном виде.

**OPEN CURSOR (ОТКРЫТЬ КУРСОР)**

***Синтаксис***

EXEC SQL OPEN CURSOR < cursorname >< SQL term >

OPEN CURSOR выполняет запрос связанный с курсором < cursor name >. Вывод может теперь извлекать по одной строке для каждой команды FETCH.

**REVOKE (\*NONSTANDARD\*)  
(ОТМЕНИТЬ ПОЛНОМОЧИЯ) (НЕСТАНДАРТНО)**

***Синтаксис***

REVOKE { ALL [PRIVILEGES]   
| < privilege > . , . . } [ON < table name >]   
FROM { PUBLIC   
| < Authorization ID > . , . . };

Привилегия < privelege > может быть любой из указанных в команде GRANT. Пользователь дающий REVOKE должен иметь те же привилегии, что и пользователь который давал GRANT. Предложение ON может быть использовано, если используется привилегия специального типа для особого объекта.

**ROLLBACK (WORK)  
(ОТКАТ) (ТРАНЗАКЦИИ)**

***Синтаксис***

ROLLBACK WORK;

Команда отменяет все изменения в базе данных, сделанные в течение те- кущей транзакции. Она кроме того заканчивается текущую, и начинает новую транзакцию.

**SELECT (ВЫБОР)**

***Синтаксис***

SELECT { IDISTINCT | ALL] < value expression > . , . . } / \*   
[INTO < host variable list > (\*embedded only\*)]   
FROM < table reference > . , . .   
[WHERE < predicate >]   
[GROUP BY < grouping column > . , . .]  
[HAVING < predicate >]   
[ORDER BY < ordering column > [ASC | DESC] . , . . ];

Это предложение организует запрос и выводит значения из базы данных (см. Глава 3 - Глава 14 ). Применяются следующие правила:

 Если ни ALL, ни DISTINCT - не указаны, принимается - ALL.

 Выражение < value expression > состоит из < column spec >, агрегатной функции < aggregate funct >, нестандартной функции < nonstandard fu nction >, постоянной < constant >, или любой их комбинации с операторами в допустимых выражениях.

 Ссылаемая таблица < table reference >, состоит из имени таблицы, включая префикс владельца если текущий пользователь не владелец, или синоним(нестандартно) для таблицы. Таблица может быть или базовой таблицей или просмотром. В принципе, псевдоним может указать, как- ой синонимом используется для таблицы только на врем текущей ко- манды. Имя таблицы или синоним должны отделяться от псевдонима одним или более разделительными знаками < separator >.

 Если используется GROUP BY, все столбцы < column spec > используемые в предложении SELECT, должны будут использоваться как группа столбцов < grouping column >, если они не содержатся в агрегатной функции < aggregate funct >. Вся группа столбцов < grouping column > должна быть представлена среди выражений < value expressions > указанных в предложении SELECT. Для каждой отдельной комбинации значений группы столбцов < grouping column >, будет иметься одна и только одна строка вывода.

 Если HAVING используется, предикат < predicate > применяется к каждой строке произведенной предложением GROUP BY, и те строки которые сделают этот предикат верным, будут выведены.

 Если ORDER BY используется, вывод имеет определенную последовательность. Каждый идентификатор столбца < column identifer > ссылается к указанному < value expression > в предложении SELECT. Если это < value expression > является указанным столбцом < column spec >, < co lumn identifier > может быть таким же как < column spec >. Иначе < co lumn identifier > может быть положительным целым числом, указываю щим место где находится < value expression > в последовательности предложения SELECT. Вывод будет сформирован так чтобы помещать значения содержащиеся в < column identifier > в порядке возрастания, если DESC не указан. Имя идентификатора столбца < column identifier > стоящее первым в предложении ORDER BY будет предшествовать позже стоящим именам в определении последовательности вывода.

Предложение SELECT оценивает каждую строку-кандидат таблицы в которой строки показаны независимо. Строка-кандидат определяется следующим образом:

 Если только одна ссылаемая таблица < table reference > включена, каждая строка этой таблица в свою очередь является строкой-кандидатом.

 Если более одной ссылаемой таблицы < table reference > включено, каждая строка каждой таблицы должна быть скомбинирована в свою очередь с каждой комбинацией строк из всех других таблиц. Каждая такая комбинация будет в свою очередь строкой-кандидатом.

Каждая строка-кандидат производит значения, которые делают предикат < predicate > в предложении WHERE верным, неверным, или неизвестным. Если GROUP BY не используется, каждое < value expression > применяется в свою очередь для каждой строки-кандидата чье значение делает предикат верным, и результатом этой операции является вывод.  
Если GROUP BY используется, строки-кандидаты комбинируются, используя агрегатные функции. Если никакого предиката < predicate > не установлено, каждое выражение< value expression > применяется к каждой строке-кандидату или к каждой группе. Если указан DISTINCT, дубликаты( двойники) строк будут удалены из вывода.

**UNION (ОБЪЕДИНЕНИЕ)**

***Синтаксис***

< query > {UNION [ALL] < query > } . . . ;

Вывод двух или более запросов < query > будет объединен. Каждый запрос < query > должен содержать один и тот же номер < value expression > в предложение SELECT и в таком порядке что 1.. n каждого, совместим по типу данных < data type > и размеру < size > с 1.. n всех других.

**UPDATE (МОДИФИКАЦИЯ)**

***Синтаксис***

UPDATE < table name >  
SET { < column name > = < value expression > } . , . .  
{[ WHERE < predlcate >]; }   
| {[WHERE CURRENT OF < cursorname >]   
< SQL term >]}

UPDATE изменяет значения в каждом столбце с именем < column name > на соответствующее значение < value expression >. Если предложение WHERE использует предикат < predicate >, то только строки таблиц чьи текущие значения делают тот предикат < predicate > верным, могут быть изменены. Если WHERE использует предложение CURRENT OF, то значения в строке таблицы с именем < table name > находящиеся в курсоре с именем < cursor name > меняются. WHERE CURRENT OF пригодно для использования только во вложенном SQL, и только с модифицируемыми курсорами. При отсутствии предложения WHERE - все строки меняются.

**WHENEVER (ВСЯКИЙ РАЗ КАК)**

***Синтаксис***

EXEC SQL WHENEVER < SQLcond > < actlon > < SQL term >  
< SQLcond > :: = SQLERROR | NOT FOUND | SQLWARNING   
(последнее - нестандартное)   
< action > :: = CONTINUE | GOTO < target > | GOTO < target >  
< target > :: = зависит от главного языка