**Операторы сравнения**

Операторы сравнения позволяют сравнивать между собой значения столбцов таблиц,

значения выражений и константы, относящиеся к любым типам данных. Результатом срав-

нения является логическое значение:

• 1 (TRUE) – истинное значение, которое свидетельствует о том, что сравнение верно,

условие выполнено;

• 0 (FALSE) – ложное значение, которое свидетельствует о том, что сравнение неверно,

условие не выполнено;

• NULL – неопределенное значение, которое свидетельствует о том, что проверить

условие невозможно, поскольку один из операндов равен NULL.

**Оператор x = y**

Оператор «равно» возвращает следующие значения:

• 1 (TRUE) – если х и *y* совпадают;

• 0 (FALSE) – если *х* и *y* различны;

• NULL – если по крайней мере один из операндов равен NULL.

Например, выберите из таблицы Customers (Клиенты) строки, в которых значение в

столбце name равно «Крылов»:

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *name = 'КРЫЛОВ';*

**Оператор x < = > у**

В случае, когда оба операнда не равны NULL, данный оператор аналогичен опера-

тору «равно». Если один из операндов равен NULL, оператор < = > возвращает значение 0

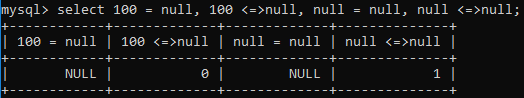
(FALSE), а если оба операнда равны NULL – значение 1 (TRUE).

Например, запрос

***SELECT*** *100 = NULL, 100< = >NULL, NULL = NULL, NULL< = >NULL;*

возвращает результат и наглядно иллюстрирует различие между операто-

рами = и < = >.



**Оператор x IS y, где y – TRUE, FALSE, UNKNOWN или NULL**

Выражением IS TRUE возвращает 1 (TRUE), если x – отличное от нулячисло или

отличная от нулевой («0000-00-00 00:00:00») дата и/или время, и 0 (FALSE) – в остальных

случаях.

Выражением IS FALSE возвращает 1 (TRUE), еслихравен нулю либо нулевой дате и/

или времени, и 0 (FALSE) – в остальных случаях.

**Примечание**

Если x является символьной строкой, то перед сравнением с TRUE

или FALSE эта строка преобразуется в число. Для этого отбрасываются

все символы, начиная с первого, недопустимого в числовом значении, а

начальная подстрока рассматривается как число. Если первый символ в

значении – буква или пустая строка («»), то x приравнивается к нулю.

Выражениях IS UNKNOWN их IS NULL возвращают 1 (TRUE), еслихравен NULL, и

0 (FALSE) – в остальных случаях.

Например, запрос

***SELECT*** *100* ***IS*** *TRUE, 0* ***IS*** *TRUE, '2007-12-12'* ***IS*** *TRUE,*

*'0000-00-00'* ***IS*** *TRUE, NULL* ***IS*** *TRUE;*

Короче просто проверят, как и в других языках. Если значение 1 и больше, то тру, если ноль, то фолс.

**Операторы x! = y, x <> y**

Оператор «не равно» возвращает следующие значения:

• 1 (TRUE) – если х и *y* различны;

• 0 (FALSE) – если *х* и *y* совпадают;

• NULL – если по крайней мере один из операндов равен NULL.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *name! = 'КРЫЛОВ';*

возвращает результат, обратный приведенному, то есть все строки, кроме

строк с фамилией «Крылов»

**Оператор x IS NOT y, где y – TRUE, FALSE, UNKNOWN или NULL**

Выражение х IS NOT TRUE возвращает 0 (FALSE), если х– отличное от нуля число или

отличная от нулевой («0000-00-00 00:00:00») дата и/или время, и 1 (TRUE) – в остальных

случаях.

Выражением IS NOT FALSE возвращает 0 (FALSE), еслиxравен нулю, нулевой дате и/

или времени, и 1 (TRUE) – в остальных случаях.

Выражениях IS NOT UNKNOWN их IS NOT NULL возвращают 0 (FALSE), если x

равен NULL, и 1 (TRUE) – в остальных случаях.

Например, запрос

***SELECT*** *100* ***IS NOT*** *TRUE, 0* ***IS NOT*** *TRUE,*

*'2007-12-12'* ***IS NOT*** *TRUE, '0000-00-00'* ***IS NOT*** *TRUE,*

*NULL* ***IS NOT*** *TRUE;*

**Оператор x < y**

Оператор «меньше» возвращает следующие значения:

• 1 (TRUE) – если х меньше у;

• 0 (FALSE) – если х равен *у* или х больше у;

• NULL – если по крайней мере один из операндов равен NULL.

Например, запрос

*SELECT \* FROM Customers WHERE name<'КРЫЛОВ';*

возвращает пустой результат, поскольку «Крылов» – наименьшее в алфавитном

порядке значение в столбце name (имя) таблицы Customers (Клиенты). Предшествующих

ему значений в столбце нет, и, следовательно, ни одна строка не удовлетворяет условию

отбора.

**Оператор x < = y**

Оператор «меньше либо равно» возвращает следующие значения:

• 1 (TRUE) – если х равно *y* или х меньше у;

• 0 (FALSE) – если х больше у;

• NULL – если по крайней мере один из операндов равен NULL.

Например, запрос

*SELECT \* FROM Customers WHERE name< = 'КРЫЛОВ';*

**Оператор x > y**

Оператор «больше» возвращает следующие значения:

• 1 (TRUE) – если х больше у;

• 0 (FALSE) – если х равно *у* или х меньше у;

• NULL – если по крайней мере один из операндов равен NULL.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *name>'КРЫЛОВ';*

**Оператор x > = y**

Оператор «больше либо равно» возвращает следующие значения:

• 1 (TRUE) – если *х* равно *у* или х больше у;

• 0 (FALSE) – если х меньше у;

• NULL – если по крайней мере один из операндов равен NULL.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *name> = 'КРЫЛОВ';*

возвращает все строки таблицы Customers (Клиенты)

**Оператор x BETWEEN a AND b**

Оператор «между» возвращает следующие значения:

• 1 (TRUE) – если *a < х <* b;

• 0 (FALSE) – если *х* меньше *a* или больше b;

• NULL – в остальных случаях.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers*

***WHERE*** *name* ***BETWEEN*** *'КРЫЛОВ'* ***AND*** *'OOO «Кускус»';*

(Почему-то, когда указывал имена, то отображала именно те имена что указал, а не те что между ними.)

**Оператор x NOT BETWEEN a AND b**

Оператор возвращает результат, противоположный результату оператора «между»:

• 1 (TRUE) – если х меньше *a* или больше b;

• 0 (FALSE) – если *a < х < b:*

• NULL – в остальных случаях.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers*

***WHERE*** *name* ***NOT******BETWEEN*** *'КРЫЛОВ'* ***AND*** *'ООО «Кускус»;*

**Оператор x IN (<Список значений>)**

Оператор «содержится в списке» возвращает следующие значения:

• 1 (TRUE) – если х совпадает с одним из элементов списка;

• 0 (FALSE) – если х не совпадает ни с одним из элементов списка;

• NULL – если х равен NULL, а также в тех случаях, когда в списке присутствует зна-

чение NULL и при этом *х* не совпадает ни с одним из элементов списка.

Оператор IN позволяет также сравнивать составные значения, то есть значение х и эле-

менты списка могут представлять собой наборы из нескольких величин (количество компо-

нентов во всех наборах должно быть одинаковым).

Например, запрос

***SELECT*** *\** ***FROM*** *Orders* ***WHERE*** *(date, product\_id)* ***IN***

*(('2007-12-12',1), ('2007-12-12',2),*

*('2007-12-13',1), ('2007-12-13',2));*

сравнивает каждую пару, состоящую из даты заказа (date) и номера товара

(customer\_id), со списком пар, и если оба компонента в паре совпадают с соответствующими

компонентами какой-либо пары из списка, то строка таблицы Orders (Заказы) будет вклю-

чена в результат запроса. Таким образом, запрос отбирает заказы товаров № 1 и № 2, сде-

ланные 12 и 13 декабря 2007 г.

**Оператор x NOT IN (<Список значений>)**

оператора IN:

• 1 (TRUE), если х не совпадает ни с одним из элементов списка;

• 0 (FALSE) – если х совпадает с одним из элементов списка;

• NULL, если х равен NULL, а также в тех случаях, когда в списке присутствует значе-

ние NULL и при этом *х* не совпадает ни с одним из элементов списка.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *rating* ***NOT******IN*** *(500,1500);*

возвращает результат, представленный в табл. 3.11.

Этот оператор, как и оператор IN, может работать с составными значениями, а также

со списком, полученным в результате подзапроса.

**Оператор x LIKE y**

Оператор сравнения с шаблоном возвращает следующие значения:

• 1 (TRUE) – если х соответствует шаблону у;

• 0 (FALSE) – если х не соответствует шаблону у;

• NULL – если х или *у* равен NULL.

В шаблоне можно использовать два специальных подстановочных символа:

• % – на месте знака процента может быть любое количество произвольных символов

операнда *х;*

• \_ – на месте знака подчеркивания может быть ровно один произвольный символ опе-

ранда *х.*

Например, следующий запрос выводит данные о тех клиентах, чьи имена содержат

кавычки:

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *name LIKE '%''%''%';*

Если требуется включить в шаблон знак процента или подчеркивания, которые должны

рассматриваться не как подстановочные, а как обычные символы, перед ними нужно поста-

вить обратную косую черту («\%», «\\_»). Если же шаблон должен содержать символ обрат-

ной косой черты, то ее нужно удвоить («\\»). Например, значение выражения \\_% LIKE \\\\_

\% истинное.

По умолчанию сравнение с помощью оператора LIKE выполняется без учета регистра

символов (то есть заглавная и строчная буквы рассматриваются как одинаковые). Для срав-

нения с учетом регистра (чтобы заглавная и строчная буквы рассматривались как разные)

необходимо указать ключевое слово BINARY или правило сравнения (COLLATE). Напри-

мер, выражение 'Крылов' LIKE 'крылов' истинно, а выражения 'Крылов' LIKE BINARY 'кры-

лов' и 'Крылов' LIKE 'крылов' COLLATE utf8\_bin ложны.

**Оператор x NOT LIKE y**

Оператор NOT LIKE возвращает результат, противоположный результату выполнения

оператора LIKE:

• значение 0 (FALSE) – если х соответствует шаблону у;

• значение 1 (TRUE) – если х не соответствует шаблону;

• значение NULL – если х или *у* равен NULL.

Например, следующий запрос выводит данные о тех клиентах, чьи имена не содержат

кавычек:

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *name* ***NOT******LIKE*** *'%»%';*

**Оператор STRCMP(x,y)**

Функция STRCMP() сравнивает строки х и *у* в соответствии с текущими правилами

сравнения и возвращает:

• – 1 – если х предшествует *у* в алфавитном порядке;

• 0 – если х и *у* совпадают;

• 1 – если х следует после *у* в алфавитном порядке;

• NULL – если по крайней мере один из аргументов равен NULL.

Например, зададим для таблицы Customers (Клиенты) правило сравнения, не учиты-

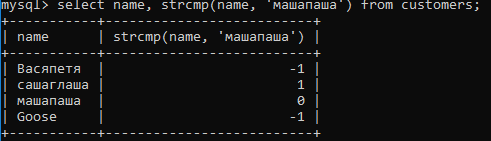
вающее регистр:

***ALTER******TABLE*** *Customers*

***CONVERT******TO******CHARACTER******SET*** *cp1251* ***COLLATE*** *cp1251\_general\_ci;*

В этом случае запрос

***SELECT*** *name,* ***STRCMP****(name,'крылов')* ***FROM*** *Customers;*



Зададим для таблицы Customers правило сравнения, учитывающее регистр:

***ALTER******TABLE*** *Customers*

***CONVERT******TO******CHARACTER******SET*** *cp1251* ***COLLATE*** *cp1251\_general\_cs;*

***SELECT*** *name,* ***STRCMP****(name,'крылов')* ***FROM*** *Customers;*

Различие результатов объясняется тем, что без учета регистра строки «Крылов» и

«крылов» эквивалентны, а с учетом регистра – различны.

При использовании сравнения по числовым кодам символов мы получим третий

результат, отличающийся от первых двух.

***ALTER******TABLE*** *Customers*

***CONVERT******TO******CHARACTER******SET*** *cp1251* ***COLLATE*** *cp1251\_bin;*

***SELECT*** *name,* ***STRCMP****(name,'крылов')* ***FROM*** *Customers;*

**Оператор MATCH (<Список столбцов>) AGAINST (<Критерий поиска>)**

(тупо поиск всех совпадений)

Оператор MATCH… AGAINST… выполняет поиск по заданным ключевым словам

в значениях указанных столбцов. При этом для столбцов должен быть создан полнотек-

стовый индекс (о полнотекстовых индексах вы узнали из главы 2). Для каждой строки

таблицы оператор MATCH. AGAINST. возвращает величину *релевантности,* которая харак-

теризует степень соответствия строки критерию поиска. Если оператор используется в пара-

метре WHERE команды SELECT, то результатом запроса будут строки с отличной от нуля

релевантностью, упорядоченные по убыванию релевантности (подобно результату поиска в

интернете с помощью поисковых систем).

Например, создадим полнотекстовый индекс для столбца description (наименование)

таблицы Products (Товары). ***Полнотекстовый индекс можно создать только для таблиц с***

***типом MyISAM, который не поддерживает связи между таблицами***. Поэтому вначале удалим

связь между таблицами Products и Orders (Заказы), удалив внешний ключ из таблицы Orders:

***ALTER******TABLE*** *Orders* ***DROP******FOREIGN******KEY*** *orders\_ibfk\_1;*

Затем изменим тип таблицы Products на MyISAM:

***ALTER******TABLE*** *Products* ***ENGINE*** *MyISAM;*

И наконец, создадим полнотекстовый индекс для столбца description:

***ALTER******TABLE*** *Products* ***ADD*** *FULLTEXT (description);*

После этого можно выполнять полнотекстовый поиск по столбцу description.

Например, запрос

***SELECT*** *\** ***FROM*** *Products*

***WHERE******MATCH*** *(description)* ***AGAINST*** *('Чайник Мосбытприбор');*

Если необходимо выполнить поиск по словам, которые могут встречаться более чем

в 50 % строк, необходимо использовать поиск в логическом режиме. Для этого необхо-

димо включить в выражение MATCH. AGAINST. параметр IN BOOLEAN MODE. Управлять

поиском в логическом режиме можно с помощью следующих спецсимволов:

• + – перед словом означает, что будут найдены только строки, содержащие это слово;

• – – перед словом означает, что будут найдены только строки, не содержащие это

слово;

• < – перед словом уменьшает «вес» этого слова при вычислении релевантности;

• > – перед словом увеличивает «вес» этого слова при вычислении релевантности;

• ~ – перед словом делает «вес» слова отрицательным (уменьшающим релевантность);

• \* — *после* слова означает произвольное окончание; например, запрос по слову +чай\*

выведет строки, содержащие слова «чайник», «чайница», «чайка» и т. п.;

• " – сочетание слов, заключенное в двойные кавычки, рассматривается как единое

слово;

• (' и ') – круглые скобки позволяют создавать вложенные выражения.

Например, запрос

***SELECT*** *\** ***FROM*** *Products*

***WHERE******MATCH*** *(description)*

***AGAINST*** *('-Чайник +Мосбытприбор'* ***IN******BOOLEAN******MODE****);*

возвращает строки, содержащие слово «Мосбытприбор», но не имеющие слова «Чай-

ник»

Еще один режим полнотекстового поиска – расширенный режим. Он отличается от

обычного тем, что в результат запроса, помимо строк, отвечающих заданному критерию

поиска, включаются строки, найденные по принципу схожести с несколькими первыми

строками, наиболее релевантными исходному критерию. Расширенный режим полезен при

поиске «наугад», когда заранее неясно, по какому критерию искать нужную строку. Для

поиска в логическом режиме необходимо включить в выражение MATCH… AGAINST…

параметр WITH QUERY EXPANSION:

***SELECT*** *\** ***FROM*** *Products*

***WHERE******MATCH*** *(description)*

***AGAINST*** *('Чайник Мосбытприбор'* ***WITH******QUERY******EXPANSION****);*

**Операторы сравнения с результатами вложенного запроса**

**EXISTS**

Оператор EXISTS возвращает значение 1 (TRUE), если результат подзапроса содержит

хотя бы одну строку, и значение 0 (FALSE), если подзапрос выдает пустой результат.

Например, получить список товаров, заказанных по крайней мере одним клиентом,

можно с помощью запроса

***SELECT*** *\** ***FROM*** *Products*

***WHERE******EXISTS***

*(****SELECT*** *\** ***FROM*** *Orders*

***WHERE*** *product\_id = Products.id*

***AND*** *customer\_id* ***IS******NOT*** *NULL);*

Обратите внимание, что в этом примере мы столкнулись с новой разновидностью вло-

женного запроса. В примерах, которые мы рассматривали ранее (см. подраздел «Вложен-

ные запросы»), вложенный запрос не использовал данные из внешнего запроса и поэтому

выполнялся только один раз, после чего найденные вложенным запросом данные обраба-

тывались внешним запросом. Однако в текущем примере вложенный запрос *связан* с внеш-

ним: в нем используется значение столбца id (идентификатор) таблицы Products (Заказы) –

таблицы, которая участвует во внешнем запросе. Следовательно, вложенный запрос *выпол-*

*няется отдельно для каждой строки* таблицы Products, каждый раз с новым значением

столбца id.

**NOT EXISTS**

Оператор возвращает результат, противоположный результату выполнения оператора

EXISTS: 1 (TRUE), если результат подзапроса не содержит ни одной строки, и 0 (FALSE),

если результат подзапроса непустой.

Например, получить список клиентов, заказавших все виды товаров, можно с помощью

следующего запроса:

***SELECT*** *\** ***FROM*** *Customers* ***WHERE******NOT******EXISTS***

*(****SELECT*** *\** ***FROM*** *Products* ***WHERE******NOT******EXISTS***

*(****SELECT*** *\** ***FROM*** *Orders*

***WHERE*** *product\_id = Products.id*

***AND*** *customer\_id = Customers.id));*

В этом запросе для каждого клиента и каждого товара самый «глубоко вложенный»

подзапрос отбирает заказы, в которых фигурируют этот клиент и этот товар. Если ни одного

такого заказа не найдено (то есть данный клиент не заказывал данный товар), то выпол-

нено условие отбора в «среднем» подзапросе. Следовательно, «средний» подзапрос выдает

непустой список товаров, которые не были заказаны данным клиентом, условие внешнего

запроса не выполняется и запись об этом клиенте не попадет в результат запроса. Если же

оказывается, что данный клиент заказывал данный товар, то, наоборот, условие отбора в

«среднем» подзапросе не выполняется, «средний» подзапрос возвращает пустой результат,

а значит, условие отбора во внешнем запросе выполнено и запись об этом клиенте будет

включена в результат запроса.

**IN и NOT IN**

Операторы IN и NOT IN, с которыми вы познакомились в подразделе «Операторы срав-

нения», позволяют проверить, содержится ли некоторое значение в результате подзапроса.

Рассмотрим еще один пример использования оператора IN.

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *'2007-12-12' IN*

*(****SELECT*** *date* ***FROM*** *Orders* ***WHERE*** *Customers.id = customer\_id);*

Для каждого клиента, то есть строки таблицы Customers, вложенный запрос выдает

даты заказов этого клиента. Если дата «2007-12-12» есть среди этих дат, то строка таблицы

Customers включается в результат запроса. Таким образом, запрос выводит информацию о

тех клиентах, которые сделали заказ 12 декабря 2007 г.

**ANY, SOME**

Ключевое слово ANY («какой-либо») используется совместно с операторами сравне-

ния, описанными в подразделе «Операторы сравнения». При использовании ANY результат

сравнения будет верным, если он верен хотя бы для одного из значений, выданных подза-

просом. Другими словами, результатом вычисления выражения

*x <Оператор сравнения> ANY (<Вложенный запрос>)*

может быть одно из следующих значений:

• 1 (TRUE) – если среди выданных подзапросом значений есть хотя бы одно значение

y, для которого выполнено условие *x* <Оператор сравнения> *y*;

• 0 (FALSE) – если среди выданных подзапросом значений нет ни одного такого зна-

чения y, для которого выражение *x* <Оператор сравнения> *y* истинно (TRUE) или не опреде-

лено (NULL), в том числе, если подзапрос возвращает пустой результат;

• NULL – если среди выданных подзапросом значений нет ни одного такого значения

y, для которого выражение *x* <Оператор сравнения> *y* истинно TRUE), но в то же время есть

одно или несколько значений *y,* для которых это выражение не определено (NULL).

Например, вывести информацию о клиентах, которые сделали хотя бы один заказ на

сумму более 5000, можно с помощью запроса

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *5000 <* ***ANY***

*(****SELECT*** *amount* ***FROM*** *Orders* ***WHERE*** *Customers.id = customer\_id);*

Для каждого клиента вложенный подзапрос получает из таблицы Orders (Заказы)

суммы заказов (столбец amount) этого клиента. Затем эти суммы сравниваются с величиной

5000, и запись о клиенте попадет в результат запроса, если хотя бы одна из этих сумм пре-

вышает 5000. Таким образом, запрос возвращает результат, представленный в табл. 3.11.

Отметим, что вложенный запрос может быть только *правым* операндом для оператора

сравнения: например, рассмотренный выше запрос нельзя переписать в виде

***SELECT*** *\** ***FROM*** *Customers* ***WHERE***

***ANY*** *(****SELECT*** *amount* ***FROM*** *Orders*

***WHERE*** *Customers.id = customer\_id)*

*> 5000;*

Ключевое слово SOME является синонимом ключевого слова ANY. Далее мы рассмо-

трим ключевое слово ALL.

**ALL**

Ключевое слово ALL («все»), как и ANY, используется совместно с операторами срав-

нения, описанными в подразделе «Операторы сравнения». При использовании ALL резуль-

тат сравнения будет верным, если он верен для *всех* значений, выданных подзапросом. Дру-

гими словами, результатом вычисления выражения

*x <Оператор сравнения> ALL <Вложенный запрос>*

может быть одно из следующих значений:

• 1 (TRUE) – если условие х <Оператор сравнения> *у* выполнено для всех *у,* выданных

подзапросом, а также в случае, если подзапрос возвращает пустой результат;

• 0 (FALSE) – если среди выданных подзапросом значений есть такое значение у, для

которого выражение х <Оператор сравнения> *у* ложно (FALSE);

• NULL – в остальных случаях.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers* ***WHERE*** *5000 < ALL*

*(****SELECT*** *amount* ***FROM*** *Orders* ***WHERE*** *Customers.id = customer\_id);*

выведет информацию не только о тех клиентах, у которых в каждом из заказов сумма

превышает 5000, но и о тех, кто не сделал ни одного заказа, ведь в последнем случае резуль-

тат запроса окажется пустым и условие отбора во внешнем запросе будет выполнено. Таким

образом, запрос возвращает результат, представленный в табл. 3.15.

Чтобы исключить клиентов, для которых нет зарегистрированных заказов, можно вве-

сти дополнительное условие отбора, например

***SELECT*** *\** ***FROM*** *Customers*

***WHERE*** *5000 < ALL*

*(****SELECT*** *amount* ***FROM*** *Orders*

***WHERE*** *Customers.id = customer\_id)*

***AND******EXISTS***

*(****SELECT*** *amount* ***FROM*** *Orders*

***WHERE*** *Customers.id = customer\_id);*

Как и при использовании ключевого слова ANY, в запросе с ключевым словом ANY

вложенный запрос может быть только правым операндом для оператора сравнения.

**Логические операторы**

**Оператор x AND y**

Оператор AND («и») возвращает следующие значения:

• 1 (TRUE) – если оба операнда – истинные значения;

• 0 (FALSE) – если один или оба операнда – ложные значения;

• NULL – в остальных случаях.

Иными словами, если вы соединили два условия отбора с помощью оператора AND, то

составное условие выполняется только тогда, когда выполняются одновременно оба соста-

вляющих условия.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers*

***WHERE*** *name* ***LIKE*** *'ООО%'* ***AND*** *rating>1000;*

не выводит ни одной строки. В таблице Customers (Клиенты) есть имена, начинающи-

еся с «ООО», и рейтинги, превышающие 1000, но ни одна из строк не удовлетворяет обоим

этим условиям одновременно.

Пара символов && является синонимом оператора AND.

**Оператор x OR y**

Оператор OR («или») возвращает следующие значения:

• 1 (TRUE) – если один или оба операнда – истинные значения;

• 0 (FALSE) – если оба операнда – ложные значения;

• NULL – в остальных случаях.

Иными словами, если вы соединили два условия отбора с помощью оператора OR, то

составное условие выполняется, если выполняется хотя бы одно из составляющих условий.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers*

***WHERE*** *name* ***LIKE*** *'ООО%'* ***OR*** *rating>1000;*

выводит строки таблицы Customers (Клиенты), в которых имя начинается с «ООО»,

а также строки, в которых рейтинг больше 1000 (см. табл. 3.6). Пара символов | |является

синонимом оператора OR.

**Оператор x XOR y**

Оператор XOR («исключающее или») возвращает следующие значения:

• 1 (TRUE) – если один из операндов – истинное значение, а другой – ложное;

• 0 (FALSE) – если оба операнда либо истинные значения, либо ложные;

• NULL – если хотя бы один из операндов равен NULL.

Иными словами, если вы соединили два условия отбора с помощью оператора XOR, то

составное условие выполняется, если выполняется *ровно одно* из составляющих условий.

Например, запрос

***SELECT*** *\** ***FROM*** *Customers*

***WHERE*** *name* ***LIKE*** *'ООО%'* ***XOR*** *rating>500;*

выводит строки таблицы Customers, в которых имя начинается с «ООО», а также те

строки, в которых рейтинг больше 500, за исключением тех строк, в которых эти условия

выполняются одновременно.

**Оператор NOT x**

Оператор NOT («не», то есть «отрицание») возвращает следующие значения:

• 1 (TRUE) – если операнд – ложное значение;

• 0 (FALSE) – если операнд – истинное значение;

• NULL – если операнд равен NULL.

Иными словами, условие отбора, созданное с помощью оператора NOT, выполняется,

если исходное условие не выполнено и не равно NULL.

Например, запрос

*SELECT \* FROM Customers*

*WHERE NOT (name LIKE 'ООО%' OR rating>1000);*

выводит те строки таблицы Customers, для которых условие name LIKE 'ООО% OR

rating>10 0 0 не выполнено и которые, следовательно, не были выведены запросом из пункта

«Оператор х OR у».

**Операторы и функции, основанные на сравнении**

• LEAST(*a*1,*a*2,…,*an*)

Данная функция возвращает наименьший из своих аргументов (либо NULL, если один

из аргументов равен NULL). Например, выражение LEAST(000 "Кускус",'Петров','Крылов')

возвращает значение «Крылов». Отметим, что в функции LEAST() можно указать только

фиксированное количество аргументов. Например, невозможно получить первое в алфавит-

ном порядке имя клиента с помощью запроса SELECT LEAST(name) FROM Customers; вме-

сто этого необходимо использовать групповую функцию MIN().

• GREATEST(*a*1,*a*2,…,*an*).

Данная функция возвращает наибольший из своих аргументов (либо NULL, если по

крайней мере один из аргументов равен NULL). Например, выражение GREATEST (ООО

«Кускус», Петров, Крылов) возвращает значение «Петров». Как и в функции LEAST(),

в функции GREATEST() можно указать только фиксированное количество аргументов.

Например, невозможно получить последнее в алфавитном порядке имя клиента с помощью

запроса SELECT GREATEST(name) FROM Customers; вместо этого необходимо использо-

вать групповую функцию MAX().

• INTERVAL(*a*,*b*1,*b*2,…,*bn*), где *b*1 < *b*2 < … < *bn.*

Функция INTERVAL возвращает порядковый номер наибольшего из чисел *b.,* не пре-

восходящих числа *a:*

• *bi* ≤ *a* < *bi*+1 – функция возвращает номер *i*;

• *a* < *b*1 – функция возвращает значение 0:

• *a* > *bn* – функция возвращает значение *n*;

• *a* равно NULL – функция возвращает значение -1.

Все аргументы этой функции являются целыми числами (если вы укажете аргумент

с другим типом данных, он будет преобразован в целочисленное значение). Чтобы функ-

ция возвращала корректный результат, необходимо, чтобы значения *bi* были упорядочены,

то есть выполнялось условие *b1 < b2 <… < bn.* Например, выражение INTERVAL(1500, 1000,

2000, 3000) возвращает значение 1.

• GOALESCE (*a*1,*a*2,…,*an*)

Данная функция возвращает первый из аргументов, который отличен от NULL (а

если все аргументы равны NULL, то возвращает значение NULL). Например, выражение

COALESCE(NULL,1/0,'TeKCT') возвращает значение «Текст», поскольку это первый аргу-

мент, отличный от NULL (при делении на 0 результатом является NULL).

• IF(a,b,c).

Данная функция проверяет, является ли истинным логическое значение или выражение

*a.* Если *a* истинно (то есть является числом, датой или временем, отличным от нулевых),

то функция возвращает значение *b,* а если *a* ложно или равно NULL, функция возвращает

значение *c.* Например, если требуется удвоить те рейтинги клиентов, которые превышают

1000, это можно сделать с помощью команды

*UPDATE Customers*

*SET rating = IF(rating>1000,rating\*2,rating);*

• IFNULL(a,b).

Данная функция возвращает значение *a,* если это значение отлично от NULL, и значе-

ние b, если *a* равно NULL. Например, если требуется всем клиентам, чей рейтинг не указан

(равен NULL), присвоить рейтинг 500, это можно сделать с помощью команды

*UPDATE Customers SET rating = IFNULL(rating,500);*

• NULLIF(a,b).

Данная функция возвращает значение NULL, если *a* = b, и значение *a* в противном

случае. Например, если требуется выполнить операцию, обратную операции из предыду-

щего пункта, то есть всем клиентам с рейтингом 500 присвоить неопределенный рейтинг,

это можно сделать с помощью команды

*UPDATE Customers SET rating = NULLIF(rating,500);*

• CASE x WHEN a1 THEN b1.

*[WHEN a2 THEN b2]*

*...*

*[WHEN an THEN bn]*

*[ELSE b0]*

*END*

или

*CASE WHEN x1 THEN b1*

*[WHEN x2 THEN b2]*

*...*

*[WHEN xn THEN bn]*

*[ELSE b0]*

*END*

Оператор CASE обеспечивает последовательную проверку списка условий и возвра-

щает значение в зависимости от того, какое из условий выполнено. В первом варианте зна-

чение выражения х сравнивается со значениями *a1, a2,…, an*:

• если *х = ai*, то оператор возвращает значение *bi;*

• если значение выражения *х* не совпало ни с одним из *a,* то оператор возвращает зна-

чение *b0*, заданное с помощью параметра ELSE;

• если значение выражения *х* не совпало ни с одним из *ai*, а параметр ELSE не задан,

то оператор возвращает значение NULL.

Во втором варианте последовательно проверяется истинность логических выражений

*хi:*

• если *хi* истинно, то оператор возвращает значение *bi;*

• если ни одно из выражений *хi* не является истинным, то оператор возвращает значение

*b0*, заданное с помощью параметра ELSE;

• если ни одно из выражений *хi* не является истинным, а параметр ELSE не задан, то

оператор возвращает значение NULL.

Например, запрос

***SELECT*** *date, customer\_id, amount,* **(в данном случае важная запятая!!!)**

***CASE******WHEN*** *amount< = 5000* ***THEN*** *'Малый'*

***WHEN*** *amount* ***BETWEEN*** *5000* ***AND*** *15000* ***THEN*** *'Средний'*

***WHEN*** *amount>15000* ***THEN*** *'Крупный'*

***END***

***FROM*** *Orders*

***ORDER******BY*** *customer\_id, amount* ***DESC****;*