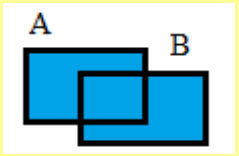
Введение в SQL

Экзистенциальные запросы

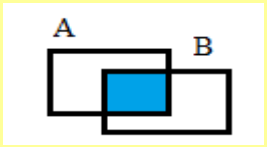
Представления

Трехзначная логика

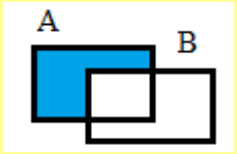
Объединение, пересечение, разность запросов в языке SQL

- объединение: A UNION B, A∪B; 

- *пересечение: A INTERSECT B, A∩B;*



- *разность:* A EXCEPT B ,A-B ;



Объединение, пересечение, разность запросов в языке SQL

- operator .[ALL] [CORRESPONDING [BY {columnl [, . . .]}]]
- При указании конструкции CORRES POND ING BY операция над множествами выполняется для указанных столбцов. Если задано только ключевое слово CORRESPONDING, а конструкция BY отсутствует, операция над множествами выполняется для столбцов, которые являются общими для обеих таблиц.

Объединение(логическое или)

- Пример выберем имена и фамилии преподавателей или студентов с непустым отчеством.

```
(SELECT surname, name
FROM student
WHERE patronym IS NOT NULL)
UNION
(SELECT surname, name
FROM teacher
WHERE patronym IS NOT NULL);
```

Или

```
(SELECT *
FROM student
WHERE patronym IS NOT NULL)
UNION CORRESPONDING BY surname, name
(SELECT *
FROM teacher
WHERE patronym IS NOT NULL);
```

Пересечение(логическое И). Пример

- Выберем фамилии и имена и отчества преподавателей , которые еще учатся (в магистратуре)(логическое И) (считаем, что полных тезок нет)

```
(SELECT surname, name, patronym
FROM student )
INTERSECT
(SELECT surname, name, patronym
FROM teacher );
```

Или

```
(SELECT *
FROM student)
INTERSECT CORRESPONDING BY surname, name, patronym
(SELECT *
FROM teacher);
```

Разность. Пример(логическое И НЕ)

- Выберем фамилии и имена и отчества студентов , которые не преподают (в магистратуре) (студенты минус преподаватели)(считаем, что полных тезок нет)

```
(SELECT surname, name, patronym
FROM student )
EXCEPT
(SELECT surname, name, patronym
FROM teacher );
```

Или

```
(SELECT *
FROM student)
EXCEPT CORRESPONDING BY surname, name, patronym
(SELECT *
FROM teacher);
```

Запросы с подзапросами в языке SQL

- использование законченных операторов SELECT, внедренных в тело другого оператора (select , update, delete)
- ГДЕ?
- В разделе FROM
- В разделах WHERE, HAVING
- В разделе Select

типы подзапросов

- *Скалярный подзапрос* возвращает значение, выбираемое из пересечения одного столбца с одной строкой, т.е. единственное значение. В принципе скалярный подзапрос может использоваться везде, где требуется указать единственное значение.
- *Строковый подзапрос* возвращает значения нескольких столбцов таблицы, но в виде единственной строки. Строковый подзапрос может использоваться везде, где применяется конструктор строковых значений, — обычно это предикаты.
- *Табличный подзапрос* возвращает значения одного или нескольких столбцов таблицы, размещенные в более чем одной строке. Табличный подзапрос может использоваться везде, где допускается указывать таблицу, например как операнд предиката IN.

пример

Пример скалярного подзапроса

Вывести группу, в которой учится столько же человек, как и в группе Z4431

```
Select st_group.* from st_group left join student on
    student.id_gr=st_group.id_gr
Group by student.id_gr
Having count (id_student)=
(select count(*) from st_group gr
  left join student st on st.id_gr=gr.id_gr where  number_gr=’ Z4431’)
```

Left join, т.к в группе может не быть студентов

Табличный подзапрос

Вывести всех студентов из групп ,где учится Иванов

```
Select student.* from student where id_gr in
(select st_group.id_gr from from st_group inner join student on
    student.id_gr=st_group.id_gr where student.surname=’Иванов’)
```

правила и ограничения, применяемые к подзапросам

1. В подзапросах не должна использоваться конструкция ORDER BY, хотя она может присутствовать во внешнем операторе SELECT.
2. Список выборки SELECT подзапроса должен состоять из имен отдельных столбцов или составленных из них выражений, за исключением случая, когда в подзапросе используется ключевое слово EXISTS.
3. По умолчанию имена столбцов в подзапросе относятся к таблице, имя которой указано в конструкции FROM подзапроса. Однако разрешается ссылаться и на столбцы таблицы, указанной в конструкции FROM внешнего запроса, для чего используются уточненные имена столбцов (как описано ниже).
4. Если подзапрос является одним из двух операндов, участвующих в операции сравнения, то подзапрос должен указываться в правой части этой операции. Например, приведенный ниже вариант записи запроса является некорректным, поскольку подзапрос размещен в левой части операции сравнения со значением столбца salary.

```
SELECT staffNo, fName, IName, position, salary
FROM Staff
WHERE (SELECT AVG(salary) FROM Staff) < salary;
```

Пример запроса с подзапросом

- Select st_group.number, count (id_st) as cnt from student join st_group
  on student.id_gr=st_group.id_gr
  group by student.id_gr
  having count (id_st) =
  (select max(cnt) from
  (select count (id_st) as cnt from student group by id_gr)q)

- Select st_group.number, from st_group join
  (select id_gr, count (id_st) as cnt from student group by id_gr)q1
  on q1.id_gr=st_group.id_gr join
  (select max(cnt) as mx from
  (select count (id_st) as cnt from student group by id_gr)q)q2
  on q2.mx=q1.cnt

коррелированный подзапрос

- *коррелированный* подзапрос – это особый вид подзапроса (табличного, однострочного или скалярного), а именно такой, в котором есть ссылка на некоторую «внешнюю» таблицу
- Получить названия поставщиков, которые поставляют деталь P1»).

```
SELECT DISTINCT S.SNAME
FROM S
WHERE ‘P1’ IN ( SELECT PNO
FROM SP
WHERE SP.SNO = S.SNO )
```

Пример update с подзапросом

student

id_st	surname	name	patronym	id_gr	rate
1	Иванов	Иван	Иванович	8	1
2	Петров	Петр	Петрович	2	6
3	Сидоров	Сидор	Сидорович	3	4
4	Кузнецов	Кузьма	Кузьмич	2	8
5	Иванова	Ирина	Игоревна	2	0
6	gr	gr	gr	7	0
7	gr	gr1	gr	7	0
8	тест	тест	тест	2	0
NULL	NULL	NULL	NULL	NULL	NULL

St_group

id_gr	num_gr	course	year_enter	is_studing	stud_cnt
1	4931	4	2017	1	NULL
2	4732	4	2016	0	NULL
3	M711	4	2016	0	NULL
4	4936	4	2017	1	NULL
6	3133	1	2021	1	NULL
7	Z9655	3	2019	1	NULL
8	6133K	3	2021	1	NULL
NULL	NULL	NULL	NULL	NULL	NULL

```
update st_group
left join
(select student.id_gr,count(*)  as cnt_st
from student where student.id_gr is not
null group by student.id_gr) as q1
 on st_group.id_gr=q1.id_gr
set stud_cnt=IFNULL(q1.cnt_st, 0)
where st_group.id_gr>0
```

```
update st_group
set stud_cnt=
(select count(*)  as cnt_st from student
where student.id_gr is not null
and student.id_gr=st_group.id_gr
group by student.id_gr)
where st_group.id_gr>0
```

id_gr	num_gr	course	year_enter	is_studing	stud_cnt
1	4931	4	2017	1	0
2	4732	4	2016	0	4
3	M711	4	2016	0	1
4	4936	4	2017	1	0
6	3133	1	2021	1	0
7	Z9655	3	2019	1	2
8	6133K	3	2021	1	1

id_gr	num_gr	course	year_enter	is_studing	stud_cnt
1	4931	4	2017	1	NULL
2	4732	4	2016	0	4
3	M711	4	2016	0	1
4	4936	4	2017	1	NULL
6	3133	1	2021	1	NULL
7	Z9655	3	2019	1	2
8	6133K	3	2021	1	1

Реляционное исчисление кортежей и реляционная алгебра

- В выражениях реляционной алгебры всегда явно задается некий порядок, а также подразумевается некая стратегия вычисления запроса.
- В реляционном исчислении не существует никакого описания процедуры вычисления запроса, поскольку в запросе реляционного исчисления указывается, *что, а не как следует извлечь.*

Реляционное исчисление кортежей

- Реляционное исчисление кортежей относится к исчислению предикатов
- В реляционном исчислении кортежей задача состоит в нахождении таких кортежей, для которых предикат является истинным. Это исчисление основано на *переменных кортежа.*
- запрос "найти множество всех кортежей S, для которых F(S) является истинным" можно записать следующим образом:
- (S | F (S) }
- {S | Student(S) S.avg_mark > 4.75}

Кванторы существования и общности

- *Квантор существования (∃),* используется в формуле, которая должна быть истинной хотя бы для одного экземпляра
- St_Group(G) (∃ S) (Student(S) (S.id_gr = G.id_gr) S.surname = ‘Иванов’)
- *Квантор общности (∀),* используется в выражениях, которые относятся ко веем экземплярам
- (∀S) (S.surname ≠ ‘Пупкин’)

Ключевые слова ANY и ALL

- Если подзапросу будет предшествовать ключевое слово ALL, условие сравнения считается выполненным только в том случае, если оно выполняется для всех значений в результирующем столбце подзапроса.
- Если тексту подзапроса предшествует ключевое слово ANY, то условие сравнения будет считаться выполненным, если оно удовлетворяется хотя бы для какого-либо (одного или нескольких) значения в результирующем столбце подзапроса.
- SOME — синоним ключевого слова ANY

Ключевые слова ANY и ALL

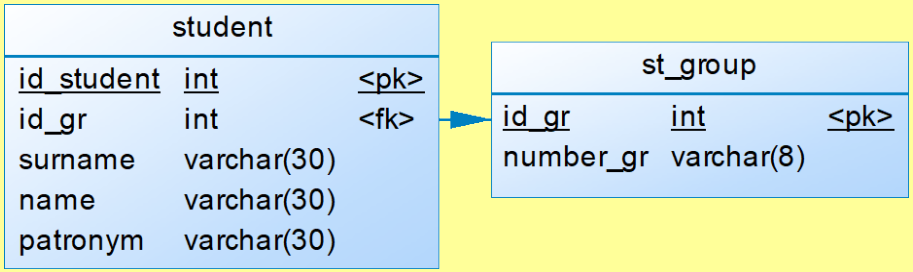
- *Найдите всех работников, чья зарплата превышает зарплату хотя бы* одного *работника отделения компании под номером 'вооз '.*

```
SELECT staffNo, fName, IName, position, salary
FROM Staff
WHERE salary > SOME(SELECT salary
FROM Staff
WHERE branchNo = 'B003');
```

Ключевые слова ANY и ALL *?*

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > ALL(SELECT salary
FROM Staff
WHERE branchNo = 'B003');
```

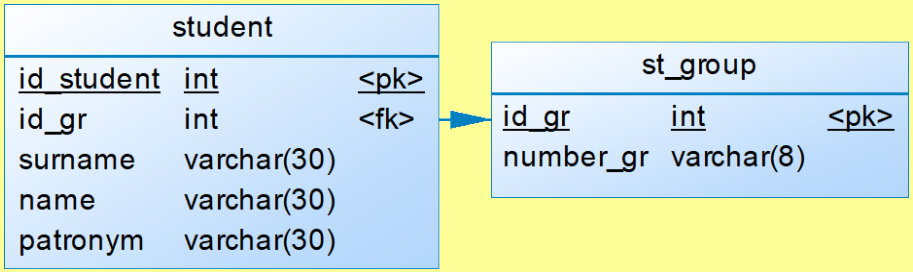
коррелированный подзапрос. Пример



- Получить имена студентов, которые учатся в группе «4831»).

```
SELECT DISTINCT Student.*
FROM Student
WHERE ‘4831’ IN ( SELECT number_gr
FROM st_group
WHERE  st_group.id_gr =Student.id_gr)
```

коррелированный подзапрос. Пример



- Получить имена студентов, которые учатся в группе «4831»).

```
SELECT DISTINCT Student.*
FROM Student
WHERE ‘4831’ IN ( SELECT number_gr
FROM st_group
WHERE  st_group.id_gr =Student.id_gr)
```

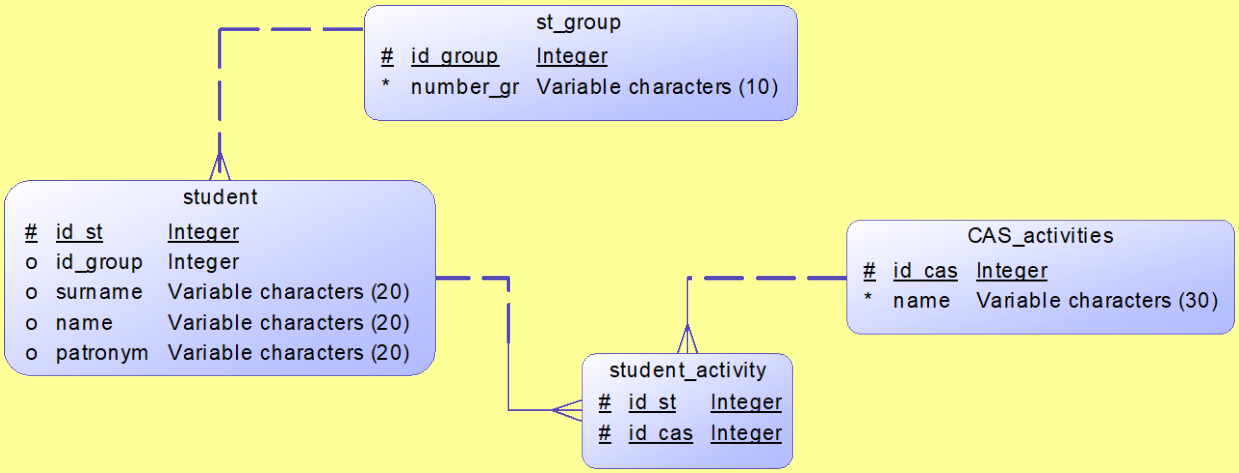
Ключевые слова EXISTS и NOT EXIST

- Для ключевого слова EXISTS результат равен TRUE в том и только в том случае, если в возвращаемой подзапросом результирующей таблице присутствует хотя бы одна строка. Если результирующая таблица подзапроса пуста, результатом обработки ключевого слова EXISTS будет значение FALSE. Для ключевого слова NOT EXISTS используются правила обработки, обратные по отношению к ключевому слову EXISTS

Ключевые слова EXISTS и NOT EXIST

- Например группа где есть студенты будет получена запросом
- Select st_group.* from st_group where exists (select id_student from student where student.id_gr=st_group.id_gr)
- Группа без студентов может быть найдена запросом
- Select st_group.* from st_group where not exists (select id_student from student where student.id_gr=st_group.id_gr)

Запрос на «все»



- «Определить **Студентов, которые посещают все кружки**»

Может быть переформулирован

- «Определить студентов, для которых Не должно существовать такого кружка, для которого (кружка) бы не было данного студента.
- Это деление студентов на кружки (все кружки)

Запрос на «все»

```
Select student.* from student
    inner join student_activity on student.id_st=student_activity.id_st
    where not exists
    (select * from CAS_activities
    where not exists
    (Select * from student as st
    inner join student_activity as st_a on st.id_st=st_a.id_st
    where st.id_st=student.id_st
    and st_a.id_cas= CAS_activities.id_cas))
```

На самом деле такой запрос можно разделить на 3 части- запроса:

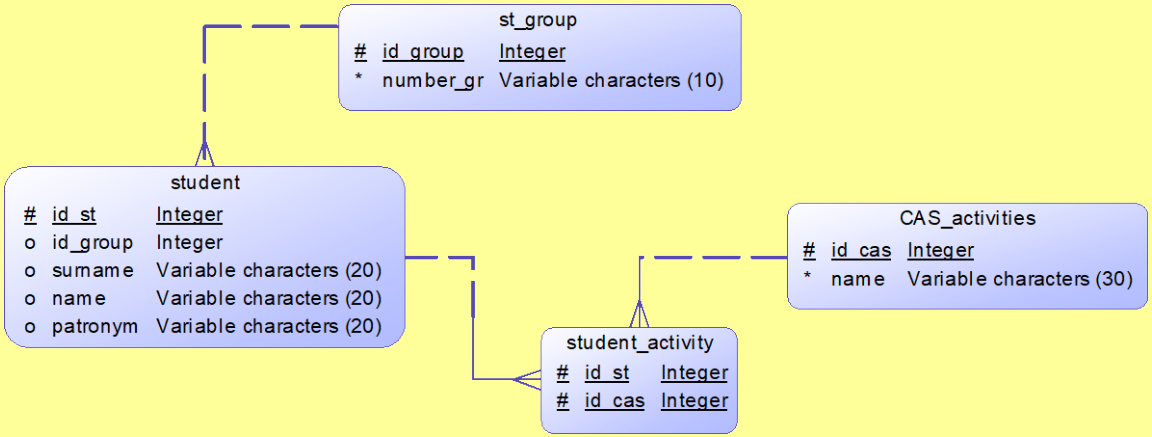
A

NOT EXISTS

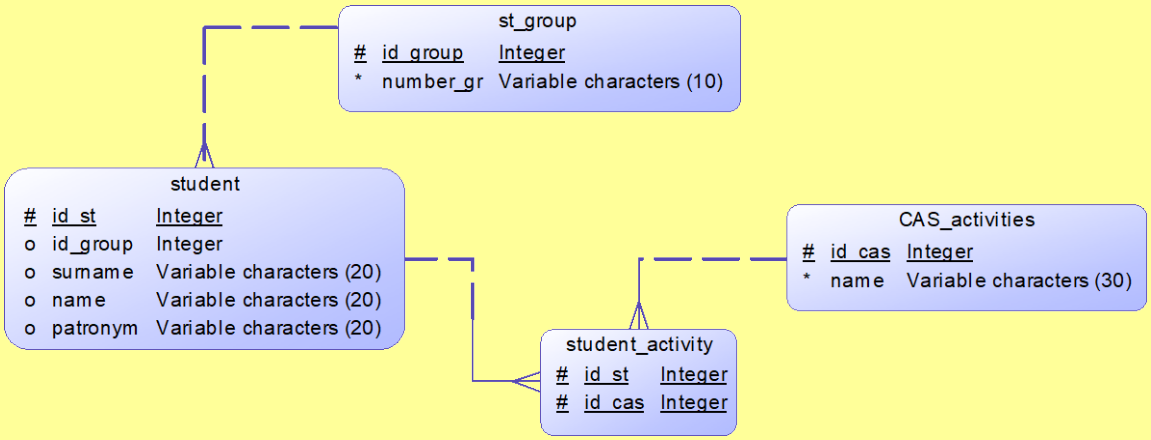
(B

NOT EXISTS (

C))

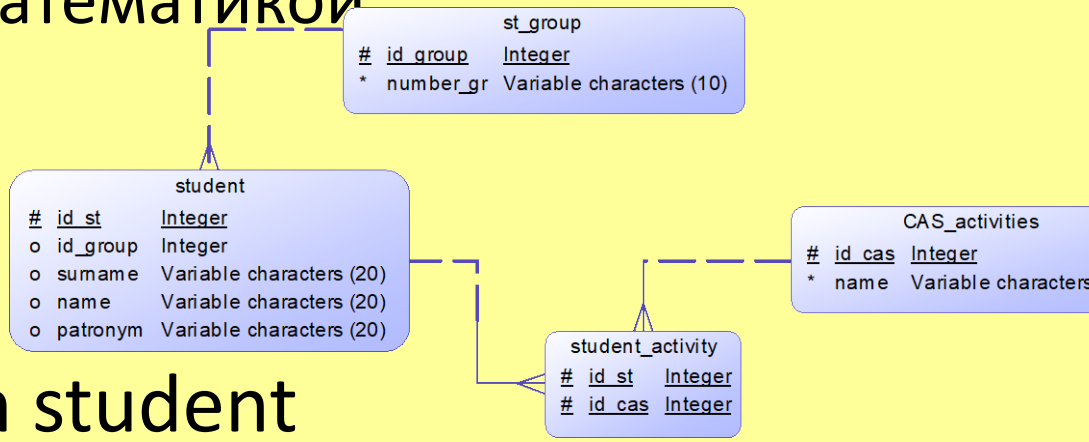


Упрощение запроса



- Select student.* from student
  where not exists
  (select * from CAS_activities
  where not exists
  (Select * from student student_activity as st_a
  where student.id_st=st_a.id_st
  and st_a.id_cas= CAS_activities.id_cas))

Студенты, которые посещают все кружки, связанные с математикой




```
Select student.* from student
  where not exists
  (select * from CAS_activities
  where name like '%математ%'
  and not exists
  (Select * from student student_activity as st_a
  where student.id_st=st_a.id_st
  and st_a.id_cas= CAS_activities.id_cas))
```

Представления (VIEW)

- **Представление.-** Динамически сформированный результат одной или нескольких реляционных операций, выполненных над отношениями базы данных с целью получения нового отношения. Представление является виртуальным отношением, которое не всегда реально существует в базе данных, но создается по запросу определенного пользователя в ходе выполнения этого запроса.

подходы для формирования представления

- *заменой представления* (под этим подразумевается замена представления оператором SQL, который обращается к базовым таблицам
- *материализацией представления,* готовое представление хранится в базе данных в виде временной таблицы, а его актуальность постоянно поддерживается по мере обновления всех таблиц, лежащих в его основе.

Типы представлений

- Представления могут быть горизонтальными и вертикальными.
- *Горизонтальное представление* позволяет ограничить доступ пользователей только определенными строками, выбранными из одной или нескольких таблиц.
- *Вертикальные представления* позволяют ограничить доступ пользователей только определенными столбцами, выбранными из одной или нескольких таблиц базы данных.

Создание удаление

-
- CREATE VIEW *ViewName* [(newColumnName [, . . .])] .
- AS *subselect* [WITH [CASCADED | LOCAL] CHECK OPTION]
- DROP VIEW *ViewName* [RESTRICT | CASCADE]

Пример

Приведем пример горизонтального представления:

Например, куратор группы 4732К может работать только с данными студентов этой группы

```
CREATE VIEW gr_4732К
AS
select student.* from student, st_group
    where student.id_gr =st_group.id_gr and number_gr='4732К'
```

А теперь допустим, что куратору не следует давать доступ к идентификаторам.

Таким образом, это будет вертикальное представление

```
CREATE VIEW gr_4732К_1
AS
select surname, name, patronym from student, st_group where student.id_gr
    =st_group.id_gr and number_gr='4732К'
```

Представление может быть обновляемым только если:

- в его определении не используется ключевое слово DISTINCT, т.е. из результатов определяющего запроса не исключаются повторяющиеся строки;
- каждый элемент в списке конструкции SELECT определяющего запроса представляет собой имя столбца (а не константу, выражение или агрегирующую функцию), причем имя каждого из столбцов упоминается в этом списке не более одного раза;
- в конструкции FROM указана только одна таблица, т.е. представление должно быть создано на базе единственной таблицы, по отношению к которой пользователь должен обладать необходимыми правами доступа. Если исходная таблица сама является представлением, то это представление также должно отвечать указанным условиям. Данное требование исключает возможность обновления любых представлений, построенных на базе соединения, объединения (UNION), пересечения (INTERSECT) или разности (EXCEPT) таблиц;
- в конструкцию WHERE не входят какие-либо вложенные запросы типа SELECT, которые ссылаются на таблицу, указанную в конструкции FROM;
- определяющий запрос не должен содержать конструкций GROUP BY и HAVING.
- Кроме того, любая строка данных, добавляемая с помощью представления, не должна нарушать требования поддержки целостности данных, установленные для исходной таблицы представления.

Основные преимущества и недостатки представлений языка SQL

- Преимущества
- Независимость от данных
- Актуальность
- Повышение защищенности данных
- Снижение сложности
- Дополнительные удобства
- Возможность настройки
- Обеспечение целостности данных
- Недостатки
- Ограниченные возможности обновления
- Структурные ограничения
- Снижение производительности

Трехзначная логика

- Проблема неопределенного значения (NULL)
- Если значение переменной A не известно, то не известен и результат любого ее сравнения, например вида A > B

Анекдот с баша

- ххх: в отпуск когда?
  yyy: хз. предлагаешь совместить?
  yyy: у тебя когда?
  ххх: у меня тоже хз, соответственно, *совместили*

Логические операторы

- Логические значения: *true (истина), false* (ложь) и *unknown* (не известно).

a	b	AND (И)
F	F	F
F	T	F
F	U	F
T	T	T
T	U	U
U	U	U

a	b	OR (ИЛИ)
F	F	F
F	T	T
F	U	U
T	T	T
T	U	T
U	U	U

a	NOT (НЕ)
F	T
T	F
U	U

Сказал: «не приду на встречу», - точно не пойду;
Сказал: «приду на встречу», - может и не пойду;
Ничего не сказал – может и приду

a	MAYBE
F	F
T	F
U	T

Алгебраические выражения

- Дано
- A = 3, B = 4
- C is UNK.

- A > B AND B > C : *false*
- A > B OR B > C : *unk*
- A < B OR B < C : true
- NOT (A = C) : *unk*

Преобразование выражений

- Сравнение x = x не обязательно в результате даст true.
- Логическое выражение p OR NOT (p) не обязательно в результате даст true.
- Вычисление логического выражения r JOIN r не обязательно в результате даст r.
- Операция INTERSECT больше не является частным случаем операции JOIN.
- Из равенства A = B AND B = C вовсе не обязательно следует равенство A = C.

Реализация в SQL

- Тип данных unknown? NULL?
- Проверка наличия неопределенного значения IS NULL и IS NOT NULL
- Проверка наличия значений true, false и unknown

p	*true*	*false*	*unknown*
P IS TRUE	true	false	false
P IS NOT TRUE	false	true	true
P IS FALSE	false	true	false
P IS NOT FALSE	true	false	true
P IS UNKNOWN	false	false	true
P IS NOT UNKNOWN	true	true	false

- Условия EXISTS. Оператор EXISTS языка SQL не является аналогичным квантору существования в трехзначной логике, поскольку он всегда возвращает значение true или false.
- Агрегирующие операторы SQL игнорируют unk кроме COUNT(*)

MAX(NULL)→NULL

- "Скалярные подзапросы". Если скалярное выражение фактически представляет собой табличное выражение, заключенное в круглые скобки, например, (SELECT S.CITY FROM S WHERE S .id= 1) если без строк unk
- NULL+1→NULL NULL+'ля ля ля'→NULL
- NULL*10 →NULL