Введение в SQL

DML

Язык манипулирования данными

- INSERT
- UPDATE
- DELETE
- MERGE
- SELECT

Вставка данных

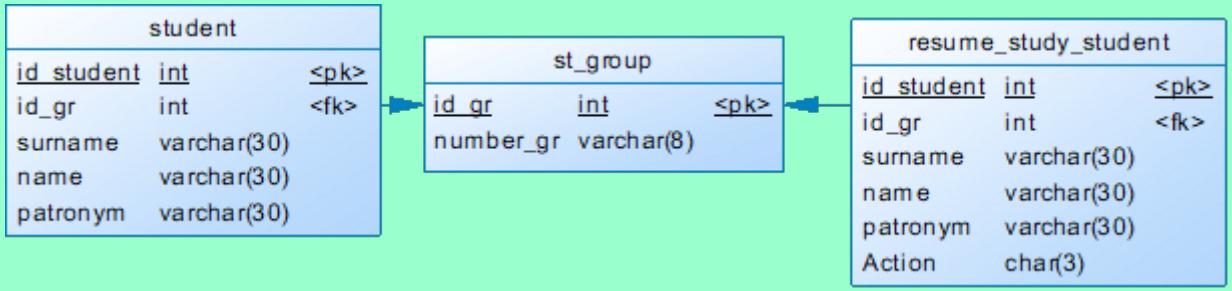
- INSERT INTO *TableName* [(*columnmList*)]
  VALUES (*dataValueList*)[, (*dataValueList2*)…]

- INSERT INTO *TableName* [(*columnmList*)]
  SELECT …

Вставка данных. Условия

- количество элементов в обоих списках должно быть одинаковым;
- должно существовать прямое соответствие между позицией одного и того же элемента в обоих списках, поэтому первый элемент списка *dataValueList* считается относящимся к первому элементу списка *columnList*, второй элемент списка *dataValueList* — ко второму элементу списка *columnList* и т.д.;
- типы данных элементов списка *dataValueList* должны быть совместимы с типом данных соответствующих столбцов таблицы.

Вставка данных. Пример



- INSERT INTO student (id_student, id_gr, surname, name, patronym)
  values (10, 2, 'Иванов', 'Иван', 'Иванович'),
  (11, 2, ‘Петров', ‘Петр', ‘Петрович');
- INSERT INTO student values (10, 2, 'Иванов','Иван','Иванович');
- INSERT INTO student (id_student, id_gr, surname, name, patronym)
SELECT id_student, id_gr, surname, name, patronym from resume_study_student;
- INSERT INTO student
SELECT id_student, id_gr, surname, name, patronym from resume_study_student

Когда можно/ нужно перечислять столбцы

- Когда задаем значения не всем столбцам
- Свойства неуказанного столбца должны допускать пустые значения или иметь значение по умолчанию
- Когда необходимо указывать много внешних ключей

Правила описания констант

- Числа идут без кавычек(1 2)
- Если число дробное, разделитель –точка(1.5)
- Все строковые и текстовые данные заключаются в одинарные или двойные кавычки('a string' "another string")
- Даты 'YYYY-MM-DD‘, 'YY-MM-DD‘ ('2012-12-31', '2012/12/31', '2012^12^31', and '2012@12@31‘)
- 'YYYYMMDD' 'YYMMDD' ('20070523' '070523')
- Время 'D HH:MM:SS' (D- дни 0-34) 'HH:MM:SS', 'HH:MM', 'D HH:MM', 'D HH', or 'SS‘ (‘5 10:5:2’)
- 'D HH:MM:SS.fraction‘(‘5 10:5:2.5’)
- '8:3:2' ='08:03:02'.
- Дата время 'YYYY-MM-DD HH:MM:SS‘, 'YY-MM-DD HH:MM:SS'
- ('2012-12-31 11:30:45', '2012^12^31 11+30+45', '2012/12/31 11*30*45', and '2012@12@31 11^30^45‘)
- 'YYYYMMDDHHMMSS' 'YYMMDDHHMMSS‘
- (20070523091528)

Модификация данных

- UPDATE *TableName*

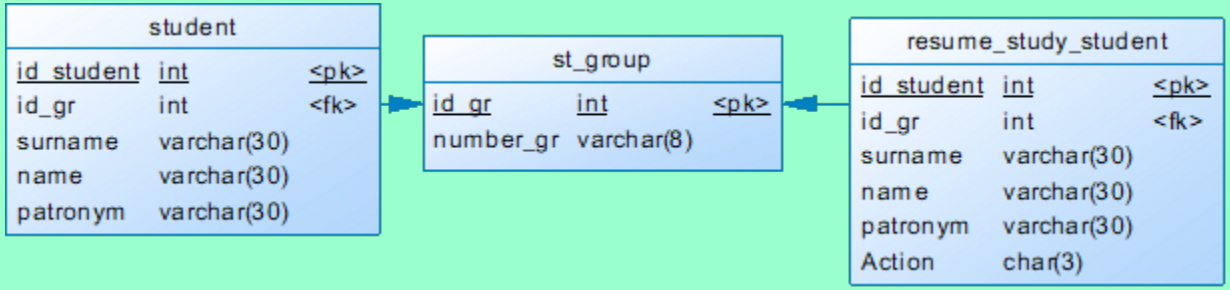
SET *columnName1 = dataValue1 [,*
columnName2= dataValue2]

[WHERE *searchCondition]*

Условия (WHERE)

- Сравнение. Сравниваются результаты вычисления одного выражения с результатами вычисления другого выражения. (<,>,=,<>)
- Диапазон. Проверяется, попадает ли результат вычисления выражения в заданный диапазон значений. (имя_поля BETWEEN знач_1 AND знач_2)
- Принадлежность к множеству. Проверяется, принадлежит ли результат вычисления выражения к заданному множеству значений. (имя_поля in (знач_1,знач_2…, знач_n))
- Значение NULL. Проверяется, содержит ли данный столбец NULL (неопределенное значение) (имя_поля IS NULL).
- Соответствие шаблону. Проверяется, отвечает ли некоторое строковое значение заданному шаблону. (имя_поля LIKE шаблон)

Модификация данных Пример



UPDATE Student
SET *id_gr = 2*
WHERE id_student=1;

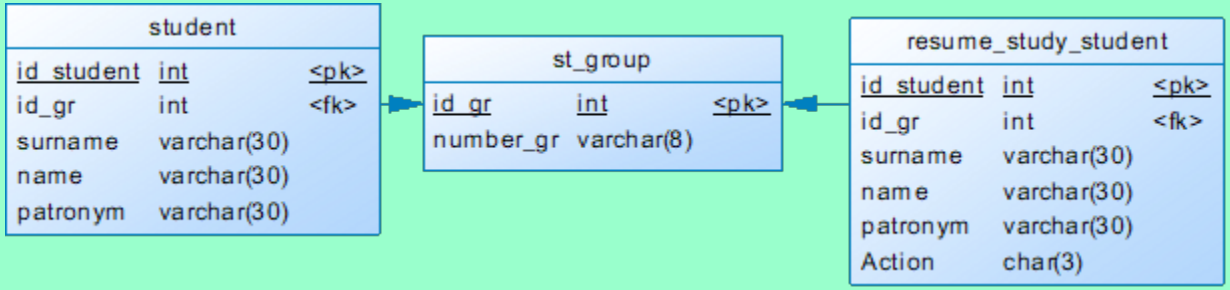
UPDATE Student
SET *patronym='Батькович' where patronym is Null*

UPDATE Student
SET id_student=id_student+1;

Удаление данных

- DELETE FROM *TableName*
- [WHERE searchCondition]

Удаление данных Пример




```
DELETE from Student
WHERE  id_student in (1,2);
```

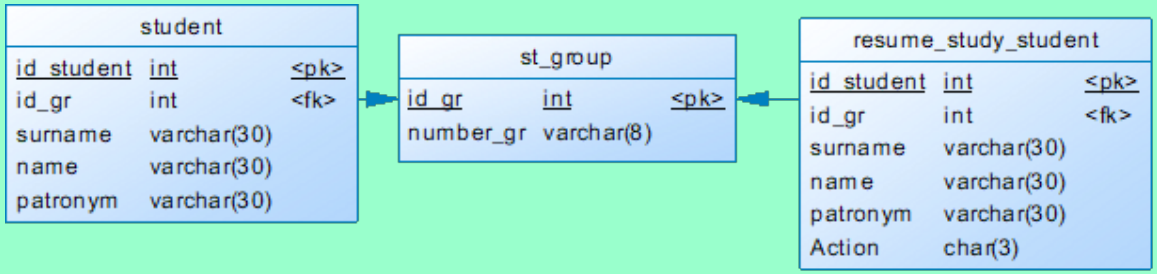
```
DELETE from Student
```

Слияние данных Merge

```
MERGE
   [ TOP ( expression ) [ PERCENT ] ]
   [ INTO ] <target_table> [ WITH ( <merge_hint> ) ] [ [ AS ] table_alias ]
   USING <table_source>
   ON <merge_search_condition>
   [ WHEN MATCHED [ AND <clause_search_condition> ]
      THEN <merge_matched> ] [ ...n ]
   [ WHEN NOT MATCHED [ BY TARGET ] [ AND <clause_search_condition> ]
      THEN <merge_not_matched> ]
   [ WHEN NOT MATCHED BY SOURCE [ AND <clause_search_condition> ]
      THEN <merge_matched> ] [ ...n ]
   [ <output_clause> ]
   [ OPTION ( <query_hint> [ ,...n ] ) ]
```

- Поддерживается MS SQL, Oracle

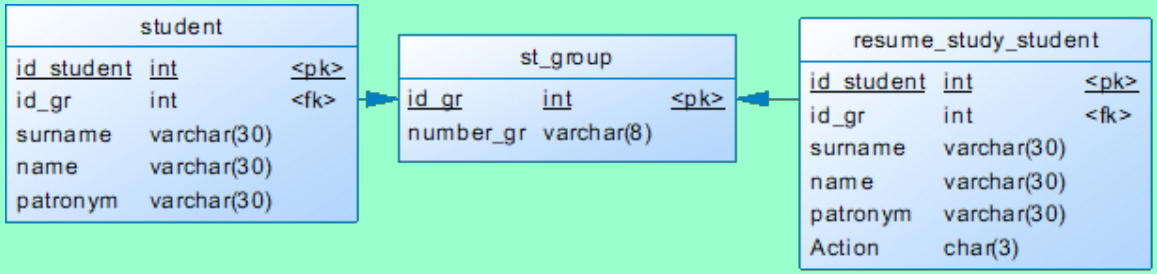
Слияние данных Merge (Синтаксис MS SQL Server)



Поле Action :
'New' – добавить строку со студентом в таблицу Student
'Mod' – изменить строку со студентом в таблице Student
'Del' – удалить студента из таблицы Student

```
MERGE INTO Student AS S
USING resume_study_student AS R
ON S.id_student = R.id_student
WHEN MATCHED AND
R.Action = 'Mod'
THEN UPDATE
SET  id_gr=R.id_gr, surname= R.surname, name=R.name, patronym =R.patronym
WHEN MATCHED AND
R.Action = 'Del'
THEN DELETE
WHEN NOT MATCHED AND
I.Action = 'New'
THEN INSERT
VALUES (R.id_student,R.id_gr, R.surname, R.name, R.patronym)
```

Слияние данных Merge (Синтаксис MS SQL Server)




```
MERGE student trg -- таблица приемник
    USING new_study_student src -- таблица источник
    ON trg.id_student = src.id_student -- условие слияния
-- 1. Строка есть в trg (student), но нет сопоставления со строкой из
    src(new)
    WHEN NOT MATCHED BY SOURCE THEN DELETE
-- 2. Есть сопоставление строки trg со строкой из источника src
    WHEN MATCHED THEN
    UPDATE SET trg.id_gr=src.id_gr, trg.surname= src.surname,
    trg.name=src.name, trg.patronym =src.patronym
    -- 3. Строка не найдена в trg (student), но есть в src (new)
     WHEN NOT MATCHED BY TARGET THEN
    INSERT(id_student,id_gr, surname, name, patronym) VALUES
    (src.id_student, src.id_gr, src.surname, src.name, src.patronym)
```

Выборка данных Select

- SELECT [DISTINCT | ALL]
- { *| *[columnExpression [*AS *newName]] [, ...] }*
- FROM TableName [alias] [,..]
- [WHERE condition]
- [GROUP BY *columnList*] [HAVING condition]
- [ORDER BY *columnList]*

columnExpression (что может идти после select)

- имена столбцов;
- **агрегирующие (агрегатные) функции;**
- константы;
- выражения, включающие комбинации перечисленных выше элементов.

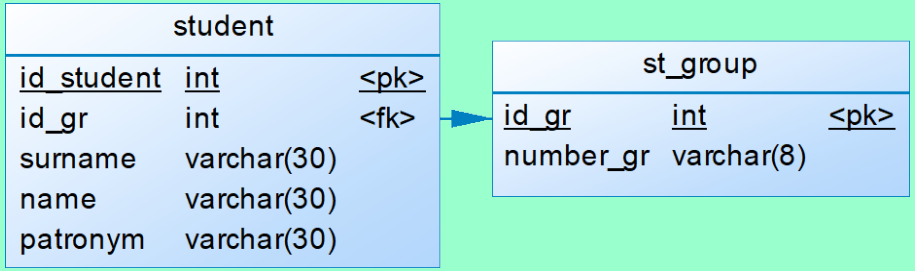
Раздел FROM

- FROM table1, table2
  WHERE table1.key1=table2.key1
- FROM table1 JOIN table2
  ON table1.key1=table2.key1

Условия (WHERE)

- Сравнение. Сравниваются результаты вычисления одного выражения с результатами вычисления другого выражения. (<,>,=,<>)
- Диапазон. Проверяется, попадает ли результат вычисления выражения в заданный диапазон значений. (имя_поля BETWEEN знач_1 AND знач_2)
- Принадлежность к множеству. Проверяется, принадлежит ли результат вычисления выражения к заданному множеству значений. (имя_поля in (знач_1,знач_2…, знач_n))
- Значение NULL. Проверяется, содержит ли данный столбец NULL (неопределенное значение) (имя_поля IS NULL).
- Соответствие шаблону. Проверяется, отвечает ли некоторое строковое значение заданному шаблону. (имя_поля LIKE шаблон)

Select пример



- select id_student, student.id_gr, surname, name, patronym
  from student, st_group
  where
  student.id_gr =st_group.id_gr
  and number_gr='4631'
- select id_student, student.id_gr, surname, name, patronym
  from student join st_group on student.id_gr =st_group.id_gr
  where number_gr='4631'

Соответствие шаблону

- атрибут LIKE pattern [ESCAPE 'escape_char']
- %. Символ процента представляет любую последовательность из нуля или более символов (поэтому часто именуется также *подстановочным символом).*
- _ Символ подчеркивания представляет любой отдельный символ.
- ESCAPE character по умолчанию \

Соответствие шаблону пример

Пример	Описание
address LIKE ‘4%'	первый символ значения обязательно должен быть символом 4, а все остальные символы не представляют интереса и не проверяются.
address LIKE ‘4_ _ _'	Этот шаблон означает, что значение должно иметь длину, равную строго 4 символам, причем первым символом обязательно должен быть символ ' 4 '
address LIKE ' %ич'	любую последовательность символов длиной не менее двух символов, причем последними символами обязательно должен быть символы ич.
address LIKE '%слово%'	любая последовательность символов, включающая подстроку «слово»;
address NOT LIKE ' Ш%'	любые строки, которые не начинаются с символа Ш
Field LIKE ‘%50\%%'	Строки, содержащие подстроку «50%»
Field LIKE ‘%40|%%'ESCAPE '|'	Строки, содержащие подстроку «40%»

Сортировка ORDER BY

- Имя поля ACS (По умолчанию) ↑
- Имя поля DESC ↓

Select MySQL

- SELECT
- [ALL | DISTINCT | DISTINCTROW]
- [HIGH_PRIORITY]
- [STRAIGHT_JOIN]
- [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
- [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
- *select_expr* [, *select_expr* ...]
- [FROM *table_references*
- [PARTITION *partition_list*]
- [WHERE *where_condition*]
- [GROUP BY {*col_name* | *expr* | *position*}, ... [WITH ROLLUP]]
- [HAVING *where_condition*]
- [WINDOW *window_name* AS (*window_spec*)
- [, *window_name* AS (*window_spec*)] ...]
- [ORDER BY {*col_name* | *expr* | *position*}
- [ASC | DESC], ... [WITH ROLLUP]]
- [LIMIT {[*offset*,] *row_count* | *row_count* OFFSET *offset*}]
- [INTO OUTFILE '*file_name*'
- [CHARACTER SET *charset_name*]
- *export_options*
- | INTO DUMPFILE '*file_name*'
- | INTO *var_name* [, *var_name*]]
- [FOR {UPDATE | SHARE} [OF *tbl_name* [, *tbl_name*] ...] [NOWAIT | SKIP LOCKED]
- | LOCK IN SHARE MODE]]

ОПЦИИ

SELECT ... INTO OUTFILE 'file_name'.	запись выбранных строк в файл, указанный в file_name. Данный файл создается на сервере и до этого не должен существовать
SELECT ... INTO DUMPFILE 'file_name'.	Запись в файл только одной строки без символов завершения столбцов или строк и без какого бы то ни было экранирования. Это полезно для хранения данных типа BLOB в файле.
SELECT college, region, seed FROM tournament ORDER BY region, seed;	Сортировка по региону и рейтингу посеву
SELECT college, region AS r, seed AS s FROM tournament ORDER BY r, s;	Сортировка по региону и посеву
SELECT college, region, seed FROM tournament ORDER BY 2, 3;	Сортировка по региону и рейтингу
SELECT * FROM tbl LIMIT 5,10;	Строки 6-15
SELECT * FROM tbl LIMIT 5; SELECT * FROM tbl LIMIT 0, *5;*	Первые 5 строк

Виды соединений в языке SQL

- Декартово произведение CROSS JOIN
- Внутреннее соединение INNER JOIN (часто просто JOIN)
- Внешние соединения:
- Левое соединение LEFT JOIN
- Правое соединение RIGHT JOIN
- Полное (внешнее) соединение OUTER JOIN

Декартово произведение CROSS JOIN

student

id_student	id_gr	surname	name	patronym
1	1	Петров	Петр	Петрович
2	2	Иванов	Иван	Иванович
3	*NULL*	Пупкин	Василий	Федорович

st_group

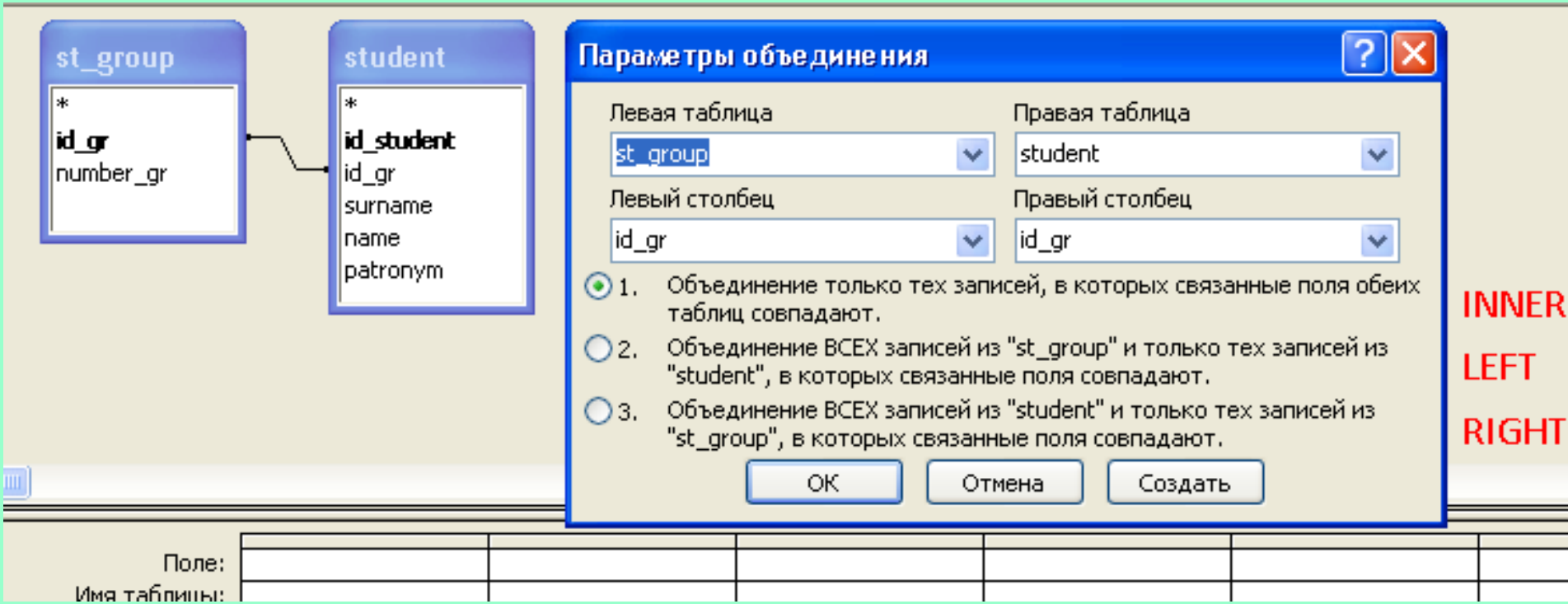
id_gr	number_gr
1	Z4431K
2	Z5432K
3	B5433

- Select * from st_group cross join student

st_group CROSS JOIN student

st_group.id_gr	number_gr	id_student	student.id_gr	surname	name	patronym
1	Z4431K	1	1	Петров	Петр	Петрович
2	Z5432K	1	1	Петров	Петр	Петрович
3	B5433	1	1	Петров	Петр	Петрович
1	Z4431K	2	2	Иванов	Иван	Иванович
2	Z5432K	2	2	Иванов	Иван	Иванович
3	B5433	2	2	Иванов	Иван	Иванович
1	Z4431K	3	*NULL*	Пупкин	Василий	Федорович
2	Z5432K	3	*NULL*	Пупкин	Василий	Федорович
3	B5433	3	*NULL*	Пупкин	Василий	Федорович

Виды соединений наглядно



Внутреннее соединение INNER JOIN

- Внутреннее соединение симметрично

SELECT *tableName1.*, tableName2.**
FROM *tableName1* INNER JOIN *tableName2*
on tableName1.field1= tableName2.field1

Эквивалентно

SELECT *tableName1.*, tableName2.**
FROM *tableName2* INNER JOIN *tableName1*
on tableName2.field1= tableName1.field1

Эквивалентно

SELECT *tableName1.*, tableName2.**
FROM *tableName1, tableName2*
WHERE *tableName1.field1= tableName2.field1*

Внутреннее соединение INNER JOIN Пример

st_group

id_gr	number_gr
1	Z4431K
2	Z5432K
3	B5433

student

id_student	id_gr	surname	name	patronym
1	1	Петров	Петр	Петрович
2	2	Иванов	Иван	Иванович
3	*NULL*	Пупкин	Василий	Федорович

```
Select number_gr, id_student, surname, name, patronym
    from st_group inner join student on
    student.id_gr=st_group.id_gr
```

st_group INNER JOIN student

number_gr	id_student	surname	name	patronym
Z4431K	1	Петров	Петр	Петрович
Z5432K	2	Иванов	Иван	Иванович

Левое соединение LEFT JOIN
Правое соединение RIGHT JOIN

- Левое и правое соединения симметричны друг по отношению к другу

SELECT *tableName1.*, tableName2.**
FROM *tableName1* LEFT JOIN *tableName2*
on tableName1.field1= tableName2.field1

Эквивалентно

SELECT *tableName1.*, tableName2.**
FROM *tableName2* RIGHT JOIN *tableName1*
on tableName2.field1= tableName1.field1

Эквивалентно

SELECT *tableName1.*, tableName2.**
FROM *tableName1, tableName2*
WHERE *tableName1.field1= tableName2.field1 or*
tableName2.field1 is Null

Левое соединение LEFT JOIN Пример

st_group

id_gr	number_gr
1	Z4431K
2	Z5432K
3	B5433

student

id_student	id_gr	surname	name	patronym
1	1	Петров	Петр	Петрович
2	2	Иванов	Иван	Иванович
3	*NULL*	Пупкин	Василий	Федорович

Select number_gr, id_student, surname, name, patronym from st_group left join student on student.id_gr=st_group.id_gr

Select number_gr, id_student, surname, name, patronym from student right join st_group on st_group.id_gr=student.id_gr

st_group LEFT JOIN student

number_gr	id_student	surname	name	patronym
Z4431K	1	Петров	Петр	Петрович
Z5432K	2	Иванов	Иван	Иванович
B5433	*NULL*	*NULL*	*NULL*	*NULL*

Правое соединение RIGHT JOIN Пример

st_group

id_gr	number_gr
1	Z4431K
2	Z5432K
3	B5433

student

id_student	id_gr	surname	name	patronym
1	1	Петров	Петр	Петрович
2	2	Иванов	Иван	Иванович
3	*NULL*	Пупкин	Василий	Федорович

Select number_gr, id_student, surname, name, patronym
from st_group right join student on student.id_gr=st_group.id_gr

Select number_gr, id_student, surname, name, patronym
from student left join st_group on
st_group.id_gr=student.id_gr

st_group RIGHT JOIN student

number_gr	id_student	surname	name	patronym
Z4431K	1	Петров	Петр	Петрович
Z5432K	2	Иванов	Иван	Иванович
NULL	3	Пупкин	Василий	Федорович

Внешнее соединение OUTER JOIN

- Внешнее соединение симметрично

SELECT *tableName1.*, tableName2.**
FROM *tableName1* OUTER JOIN *tableName2*
on tableName1.field1= tableName2.field1

Эквивалентно

SELECT *tableName1.*, tableName2.**
FROM *tableName2* OUTER JOIN *tableName1*
on tableName2.field1= tableName1.field1

Эквивалентно

SELECT *tableName1.*, tableName2.**
FROM *tableName1, tableName2*
WHERE *tableName1.field1= tableName2.field1*
or tableName1.field1 is NULL
or tableName2.field1 is NULL

Внешнее соединение OUTER JOIN Пример

st_group

id_gr	number_gr
1	Z4431K
2	Z5432K
3	B5433

student

id_student	id_gr	surname	name	patronym
1	1	Петров	Петр	Петрович
2	2	Иванов	Иван	Иванович
3	*NULL*	Пупкин	Василий	Федорович

Select number_gr, id_student, surname, name, patronym
from st_group outer join student on
student.id_gr=st_group.id_gr

st_group OUTER JOIN student

number_gr	id_student	surname	name	patronym
Z4431K	1	Петров	Петр	Петрович
Z5432K	2	Иванов	Иван	Иванович
NULL	3	Пупкин	Василий	Федорович
B5433	*NULL*	*NULL*	*NULL*	*NULL*

Все соединения Пример

st_group

id_gr	number_gr
1	Z4431K
2	Z5432K
3	B5433

student

id_student	id_gr	surname	name	patronym
1	1	Петров	Петр	Петрович
2	2	Иванов	Иван	Иванович
3	*NULL*	Пупкин	Василий	Федорович

st_group INNER JOIN student

number_gr	id_student	surname	name	patronym
Z4431K	1	Петров	Петр	Петрович
Z5432K	2	Иванов	Иван	Иванович

st_group LEFT JOIN student

number_gr	id_student	surname	name	patronym
Z4431K	1	Петров	Петр	Петрович
Z5432K	2	Иванов	Иван	Иванович
B5433	*NULL*	*NULL*	*NULL*	*NUL*

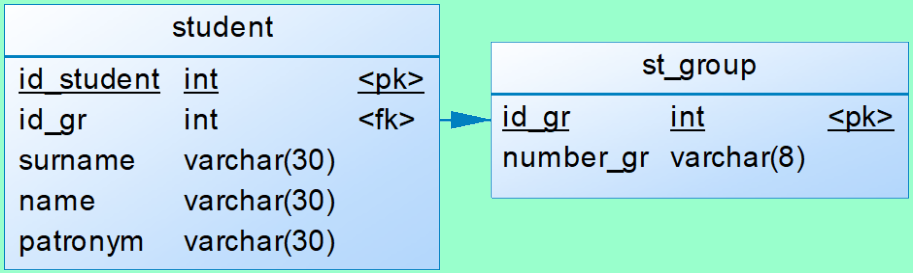
st_group RIGHT JOIN student

number_gr	id_student	surname	name	patronym
Z4431K	1	Петров	Петр	Петрович
Z5432K	2	Иванов	Иван	Иванович
NULL	3	Пупкин	Василий	Федорович

st_group OUTER JOIN student

number_gr	id_student	surname	name	patronym
Z4431K	1	Петров	Петр	Петрович
Z5432K	2	Иванов	Иван	Иванович
NULL	3	Пупкин	Василий	Федорович
B5433	*NULL*	*NULL*	*NULL*	*NULL*

Пример использования различных видов соединений



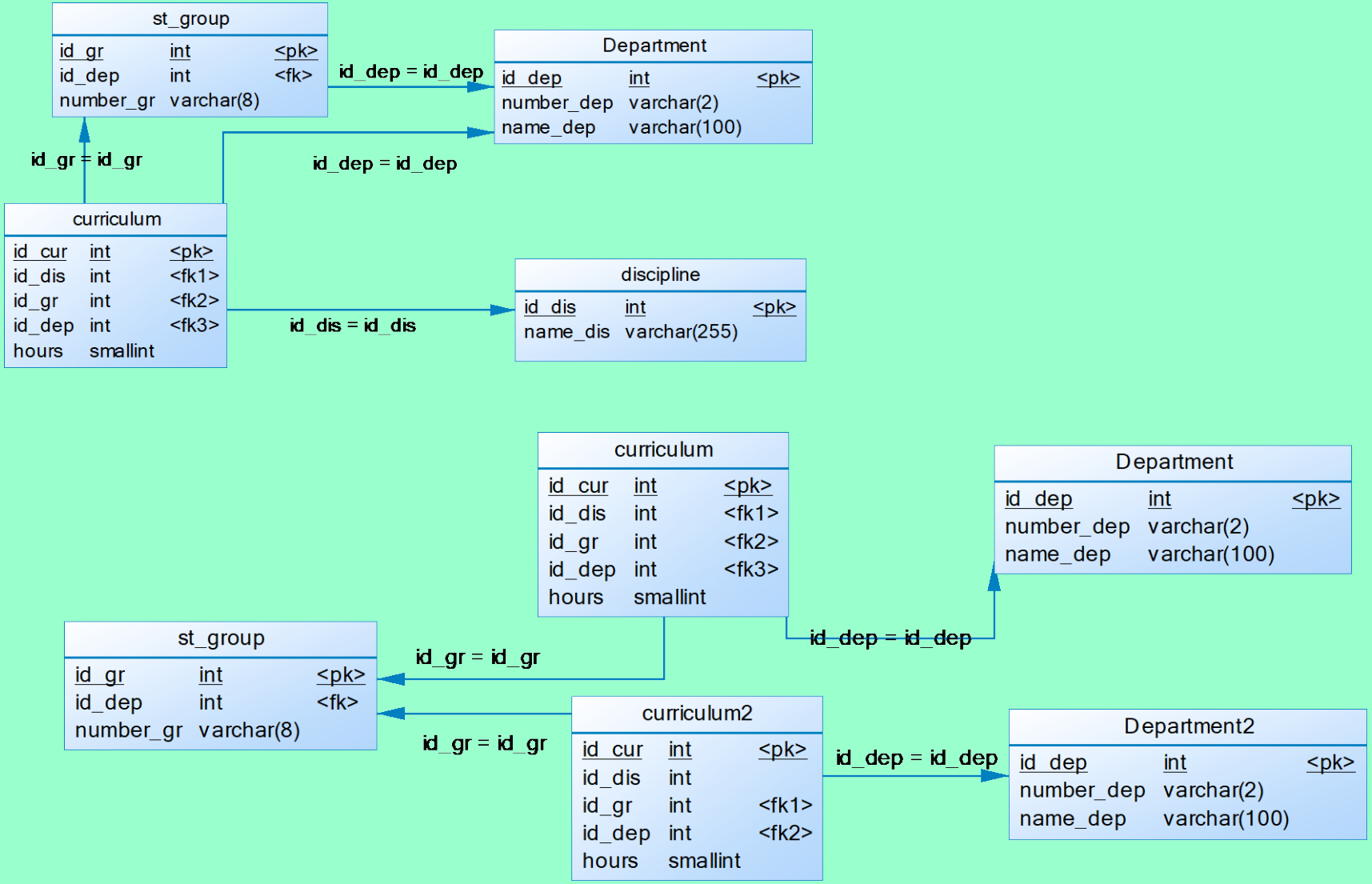
Студенты без групп

- select id_student, student.id_gr, surname, name, patronym
  from student left join st_group on student.id_gr
  =st_group.id_gr
  where st_group.id_gr is null

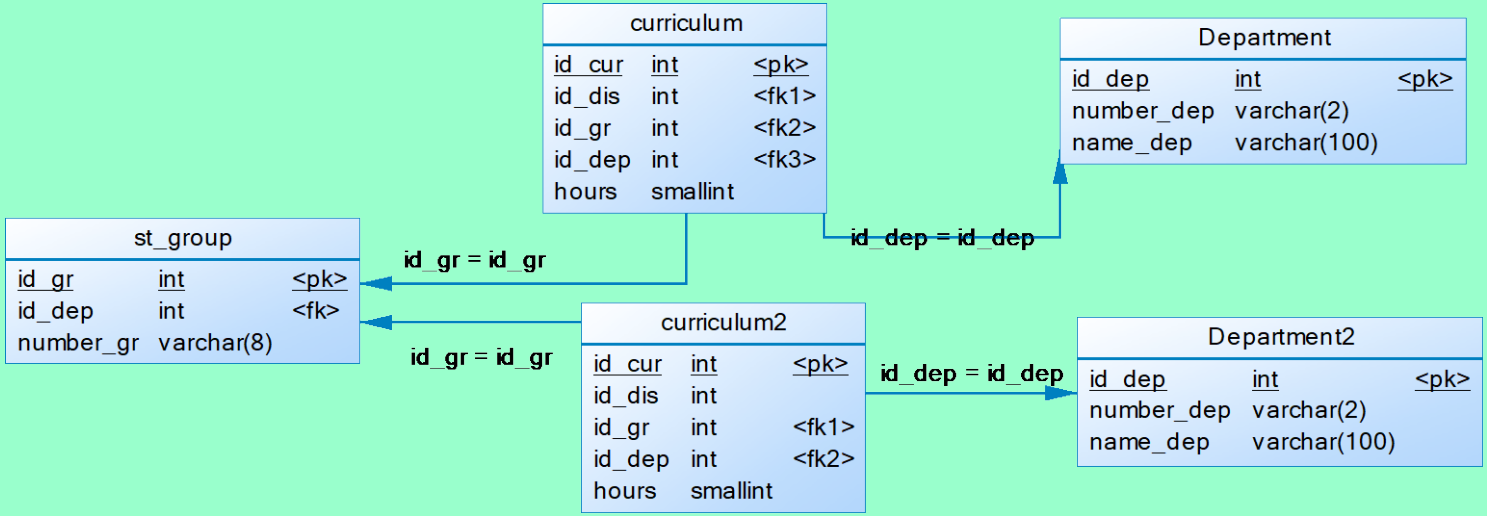
Группы без студентов

- select id_student, student.id_gr, surname, name, patronym
  from student right join st_group on student.id_gr
  =st_group.id_gr
  where student. id_student is null

Запросы с использованием псевдонимов (Группы, у которых ведет дисциплины и 43 и 41 кафедра)



Запросы с использованием псевдонимов (Группы, у которых ведет дисциплины и 43 и 41 кафедра)



Select distinct st_group.id_gr, number_gr from st_group

inner join curriculum on curriculum.id_gr= st_group.id_gr

inner join Department on Department.id_dep=curriculum.id_dep

inner join curriculum **as curriculum2**
on **curriculum2**.id_gr= st_group.id_gr

inner join Department as **Department2** on
Department2.id_dep=**curriculum2**.id_dep

where Department.number_dep=’43’ and
Department2.number_dep=’41’

Агрегатные функции

- Count
- Sum
- Min
- Max
- Avg

Агрегатные функции

- В списке выборки SELECT
- В конструкции HAVING

Агрегатные функции

- Функции COUNT, MIN и MAX применимы как к числовым, так и к нечисловым полям, тогда как
- функции SUM и AVG могут использоваться только в случае числовых полей
- За исключением COUNT (*) , при вычислении результатов любых функций сначала исключаются все пустые значения, после чего требуемая операция применяется только к оставшимся непустым значениям

COUNT (*)

student

id_student	id_gr	surname	name	patronym
1	1	Петров	Петр	Петрович
2	2	Иванов	Иван	Иванович
3	*NULL*	Пупкин	Василий	Федорович

- COUNT (*)— его назначение состоит в подсчете всех строк в таблице, независимо от того, содержатся там пустые, повторяющиеся или любые другие значения.
- Select count(id_gr) from student 2
- select count(*) from student 3

DISTINCT

student

id_student	id_gr	surname	name	patronym
1	1	Петров	Петр	Петрович
2	2	Иванов	Иван	Иванович
3	*NULL*	Пупкин	Василий	Федорович
3	*2*	Сидоров	Сидор	Сидорович

Select count(id_gr) from student 3
Select count(distinct id_gr) from student 2

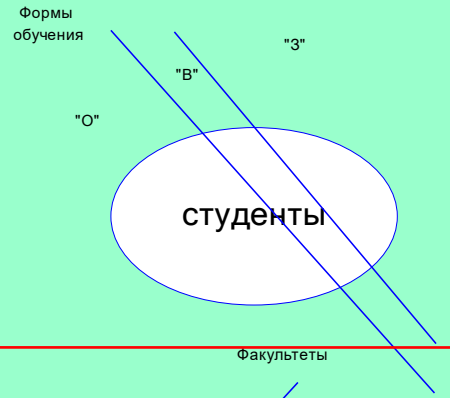
- DISTINCT не имеет смысла для функций MIN и MAX.
- DISTINCT может оказывать влияние на результаты выполнения функций SUM и AVG,

Группирование результатов (GROUP BY)

Student_uni		
id_student	int	<pk>
surname	varchar(30)	
name	varchar(30)	
patronym	varchar(30)	
Form_study	varchar(1)	
Faculty	varchar(1)	
department	varchar(2)	

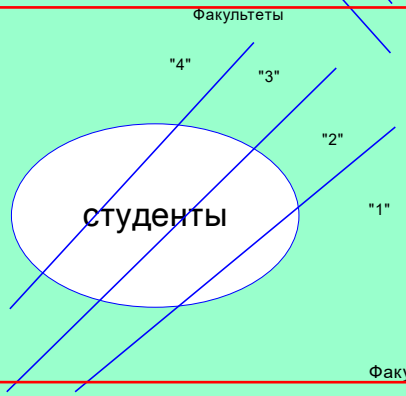
- **SELECT count (id_student) as all_st from Student_Uni**
- SELECT count (id_student) as all_st, Form_study from Student_Uni GROUP BY Form_study

all_st	Form_study
400	O
20	B
200	3



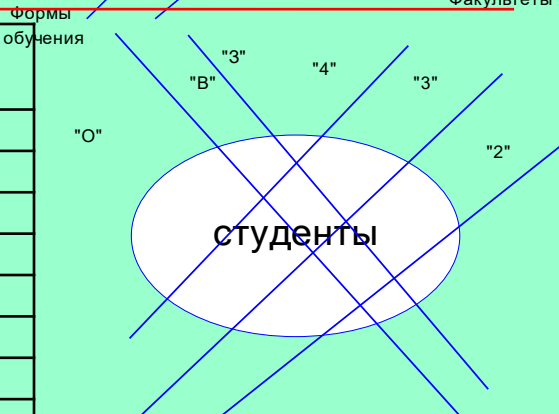
- SELECT count (id_student) as all_st, Faculty from Student_Uni GROUP BY Faculty

all_st	Faculty
100	1
100	2
200	3
220	4



- SELECT count (id_student) as all_st, Faculty, Form_study from Student_Uni GROUP BY Faculty, Form_study

All_st	Faculty	Form_study
100	1	O
60	2	O
40	2	3
100	3	O
40	3	B
60	3	3
	4	O



?

- SELECT count (id_student) as all_st,
  Form_study from Student_Uni
  GROUP BY id_student

- SELECT count (*) as all_st, Form_study from
  Student_Uni
  GROUP BY id_student