Реляционная теория

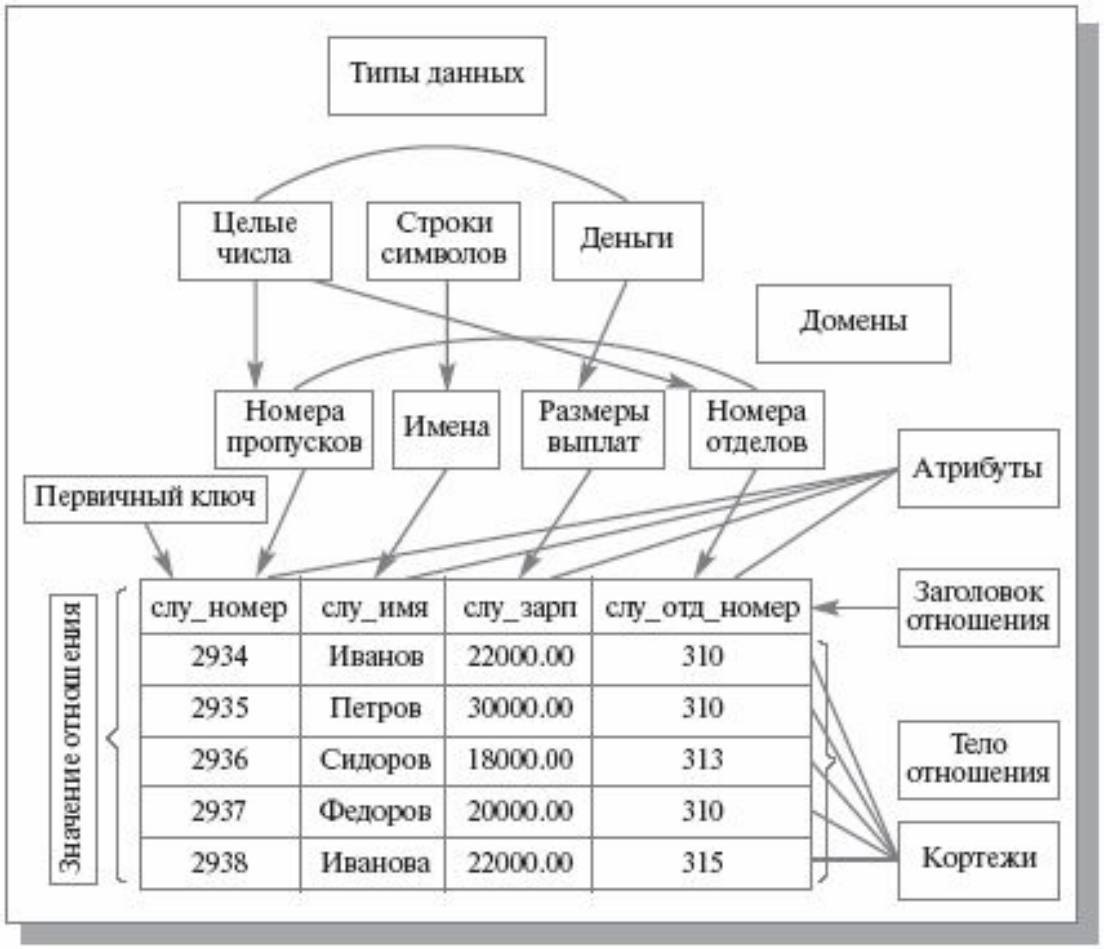
Реляционная таблица (отношение)

- ***Отношение*** представляет собой множество элементов, называемых кортежами.
- **Отношение**— Плоская таблица, состоящая из столбцов и строк
- **Кортеж** — Строка отношения.
- В математике корте́ж — последовательность конечного числа элементов.
- Многие математические объекты формально определяются как кортежи. Например, граф определяется как кортеж (V,E), где V — это набор вершин, а E — подмножество V × V, обозначающее рёбра.
- **Степень.** Степень отношения определяется количеством атрибутов, которое оно содержит.
- **Кардинальность** — Количество кортежей, которое содержится в отношении.

Альтернативные наименования

Официальные термины	Альтернативный вариант 1	Альтернативный вариант 2
Отношение (Relation**)**	**Таблица (table)**	Файл (file)
Кортеж (tuple)	Строка (row)	Запись (record)
Атрибут (attribute)	Столбец (column)	Поле (field)

Реляционная таблица: что есть что



Свойства реляционной таблицы

1. Таблица представляет собой двумерную структуру, состоящую из строк и столбцов
2. Каждая строка таблицы (кортеж, tuple) представляет собой отдельную сущность внутри набора сущностей
3. Каждый столбец таблицы представляет собой атрибут, и у каждого столбца есть свое имя
4. На каждом пересечении строки и столбца имеется единственное значение
5. Каждая таблица должна иметь атрибут или несколько атрибутов, уникально идентифицирующих каждую строку
6. Все значения в столбце должны отображаться в одинаковом формате. Например, если атрибуту присваивается формат целого, то все значения в столбце, представляющем данный атрибут должны быть целыми
7. Каждый столбец имеет определенный диапазон значений, называемый доменом атрибута
8. Порядок следования строк и столбцов не существенен.

Ключи реляционных баз данных

Тип ключа	Определение
Суперключ (superkey)	Атрибут(или комбинация атрибутов), уникально идентифицирующих каждую сущность в таблице
Потенциальный ключ (candidate key)	Минимальный суперключ. Суперключ, который не содержит подмножества атрибутов, которое само по себе является суперключом.
Первичный ключ (primary key)	Потенциальный ключ, выбранный для уникальной идентификации всех остальных значения атрибутов в любой строке. Не может содержать пустых значений
Вторичный ключ (secondary key)	Атрибут(или комбинация атрибутов), используемый исключительно в целях поиска данных
Внешний ключ (foreign key)	Атрибут(или комбинация атрибутов) в одной таблице, значения которого должны или совпадать со значениями первичного ключа другой таблицы, или быть пустыми

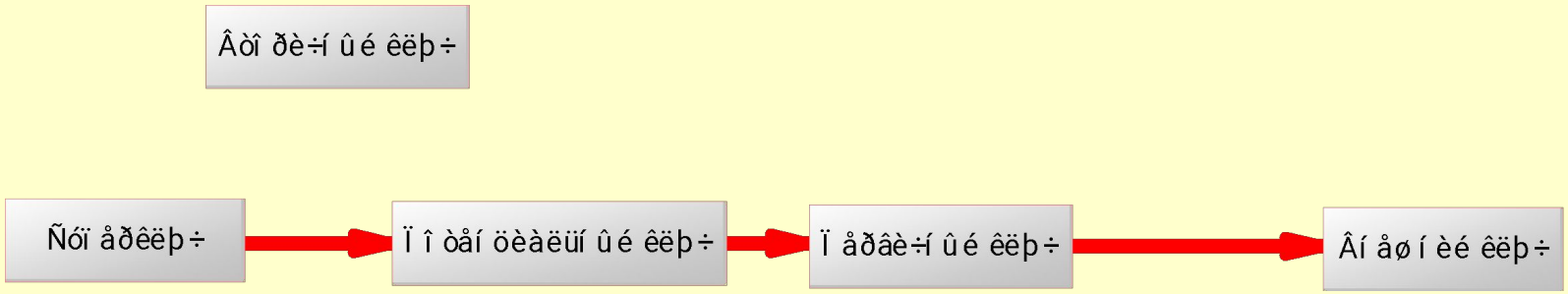
Ключи пример

Человек

id	инн	фамилия	имя	отчество	серия паспорта	номер паспорта	дата рождения

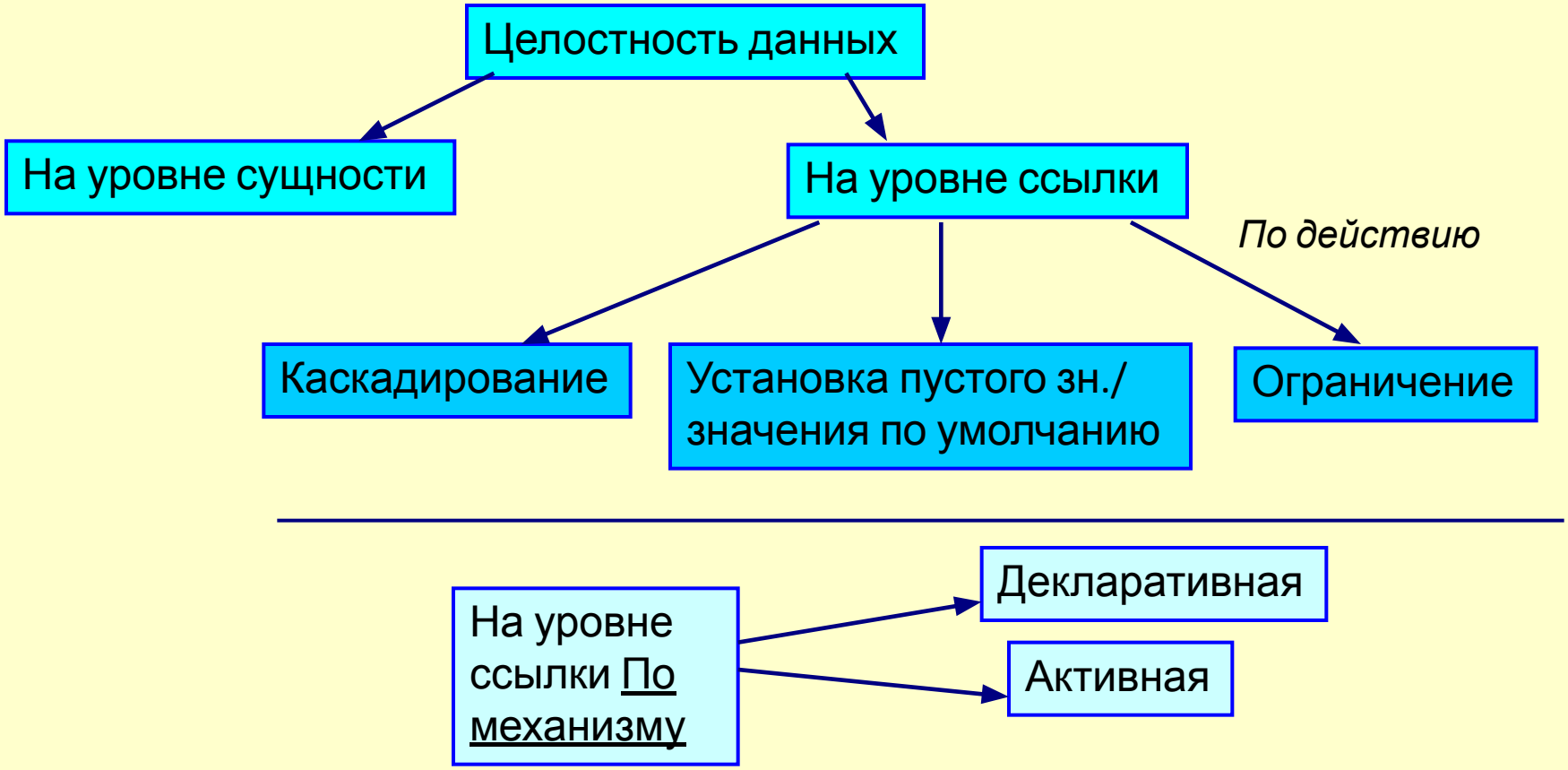
Суперключ	**Потенциальный ключ**	**Первичный ключ**	**внешний ключ**	**вторичный**
id	id	id	id	ф+и+о+ дата рождения
инн	инн			
ф+и+о+дата рождения	ф+и+о+дата рождения			
серия паспорта+ номер паспорта	серия паспорта+ номер паспорта			
ф+и+о+ИНН				
ф+инн				
…				
id+ИНН+ ф+и+о+серия+ номер+ дата рождения				

Ключи


Âòî ðè÷í ûé êëþ÷
Ñóï åðêëþ÷
Ï î òåí öèàëüí ûé êëþ÷
Ï ðâè÷í ûé êëþ÷
Âí åø í èé êëþ÷

Целостность данных

- **Пустое значение.** Указывает, что значение атрибута в настоящий момент неизвестно или неприемлемо для этого кортежа.



Введение в SQL

DML

Язык манипулирования данными

- INSERT
- UPDATE
- DELETE
- MERGE
- SELECT

Вставка данных

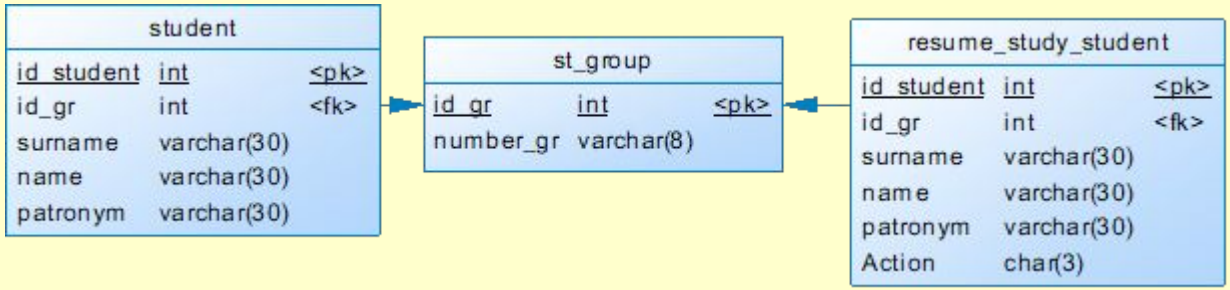
- INSERT INTO *TableName* [(*columnmList*)]
  VALUES (*dataValueList*)[, (*dataValueList2*)…]

- 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*

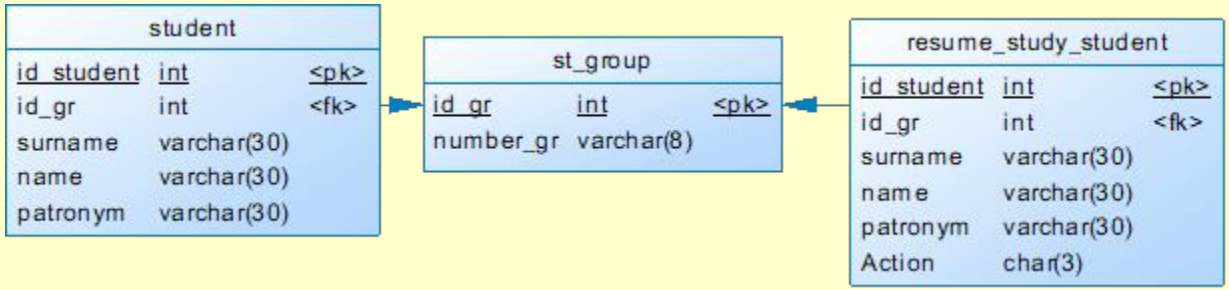
SET *columnName1 = dataValue1 [,*
columnName2= dataValue2]

[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;

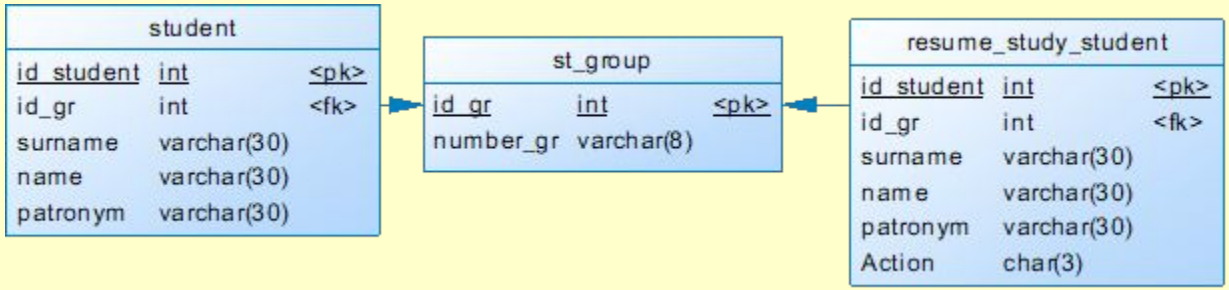
UPDATE Student
SET *patronym='Батькович' where patronym is Null*

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);
```

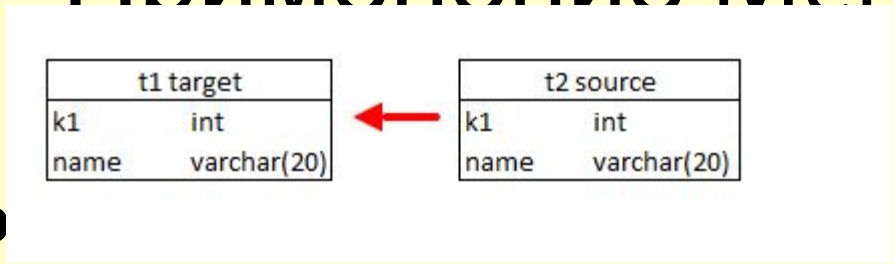
```
DELETE from Student
```

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

Применение Merge



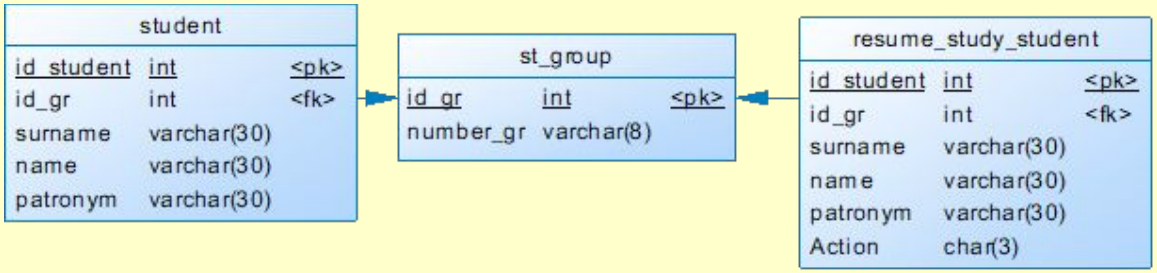
Новая вер
старой

- Если совпали ключи в старой (t1 target) и новой (t2 source) версиях, то обновляем старую версию
- Если не нашли в старой (t1 target), то вставляем в неё
- Если не нашли в новой версии (t2 source) , то удаляем из старой (t1 target)

Пакетное изменение

- Если совпали ключи в старой (t1 target) и новой (t2 source) версиях, то обновляем старую версию
- Если не нашли в старой (t1 target), то вставляем в неё
- Если не нашли в новой версии (t2 source) , то ничего делать не надо, т. к. строку не меняли

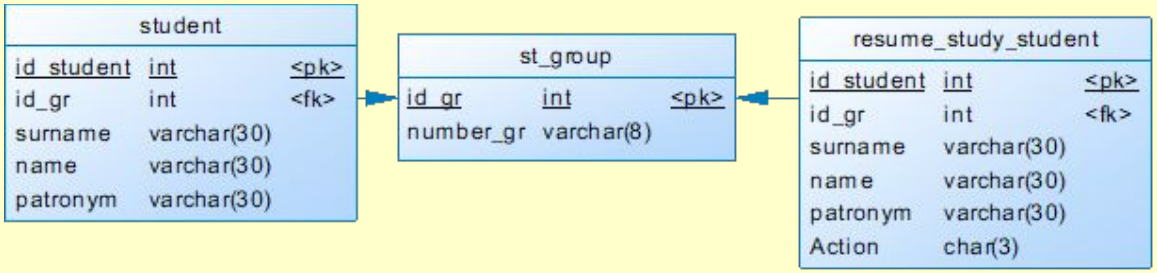
Слияние данных Merge (Синтаксис MS SQL Server)



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)
```