Лекция 5

Операции в реляционной алгебре

Уже на протяжении выполнения лабораторных работ можно было столкнуться с теоретико-множественными операциями из реляционный алгебры, например, `JOIN`. Причем, стоит заметить, что реляционная алгебра - замкнутая.

Операции

- **Проекция** - новое отношение, содержащее вертикальное подмножество исходного отношения, создаваемое посредством извлечения значений указанных атрибутов и исключения из результата строк-дубликатов. То есть с точки зрения Кодда не существует `SELECT` без `DISTINCT` ⇒ Проекция не является `SELECT` в строгом определении.

$$\pi_{a_1,a_2,\ldots,a_n}(R)$$

- **Выборка** - определяет новое отношение, которое сожержит все кортежи исходного, которые удволетворяют данному предикату. В SQL выборка реализована через инструкцию `WHERE Condition`.

$$\sigma_{predicate}(R)$$

- **Объединение** (`UNION`) - новое отношение, которое содержит все кортежи, сожержащиеся в R, все кортежи, содержащиеся только в S и кортежи, содержащелся в R и в S с исключением дубликатов. R и S должны быть совместимы для объединения: отношения совместимы, если они состоят из одинаковых атрибутов и каждая пара атрибутов имеет одинаковый домен. В SQL возможно сделать объединение так:

```
SELECT * FROM Table1
UNION
SELECT * FROM Table2;
```

$$R \cup S$$

- **Разность** - новое отношение, состоящее из кортежей, которые миеются в отношении R, но отсутсвуют в отношении S.

$$R - S$$

- **Пересечение** - определяет новое отношение, которое включает кортежи, входящие в обои отношения одновременно. Аналогично, отношения, к которым применяется пересечение, должны быть совместимы.

$$R \cap S$$

- **Декартово произведение** - определяет новое отношение, которое является результатом конкотенации каждого кортежа из отношения R с каждым кортежем из отношения S.

$$R \times S$$

- **Тета-соединение** (`JOIN`) - определяет новое отношение, которое содержит все возможные комбинации кортежей из двух исходных отношений , удовлетворяющие заданному предткату `F`.

$$R \bowtie_F S$$

$$F = R_a \theta S_{b_i}$$

$$\theta = \{<, >, =<\}$$

- **Экви-соединение** - определяет новое отношение, результатом которого содержит все возможные комбинации кортежей из двух исходных отношений, равные по какому либо атрибуту

$$R \bowtie_= S$$

- **Естественное соединение** - эквисоединение двух отношений, выполненное по всем общим атрибутам, из результатов которого исключается по одному экземпляру общего атрибута. *Пример: есть две таблицы с номером и серией паспорта. Объединяем по номеру и серии паспорта (все равные атрибуты) и выкидываем один столбец с серией и один столбец с номером.*

$$R \bowtie S$$

- **Левое внешнее соединение** - определяет новое отношение, которое содержит в себе все кортежи из отношения R, с конкатенацей к ним тех кортежей из отношения из S, имеющих совпадающие значения в общих атрибутах

$$R ⟕ S$$

```
В нормальных значках:
```

$$R \rhd\!\lhd S$$

- **Полусоединение** - определяет новое отношение, которое содержит в себе все кортежи из R, которые входят в экви-соединение R и S

$$R \triangleright S$$

Язык SQL

Существует 4 базовых оператора:

- **SELECT**
- **INSERT**
- **UPDATE**

Разберем `SELECT` подробнее

Что за что отвечает???

- **Выбор столбцов отношения**

```SQL
SELECT [ DISTINCT | ALL ] { * | ColumnExpression [AS NewName] [ , ... ] }
```

Ключевое слово `DISTINCT` определяет выбор только уникальных кортежей, `ALL` - явный выбор кортежей с дубликатами

Далее указываются имена столбцов (также возможны их алиасы) или `*`, которая определяет вывод всех столбцов отношения

- **Выбор исходного отношения**

```SQL
FROM TableName [ AS NewTableName]
[ { INNER | LEFT OUTER | FULL } JOIN OuterTable [ AS NewOuterTableName] ON
Condition ]
```

- **Фильтрация кортежей**

```SQL
[ WHERE Condition ]
```

- **Группировка**

```SQL
[ GROUP BY ColumnList [ , ... ] [HAVING Condition] ]
```

В инструкции `GROUP BY` производится группировка по указанному набору атрибутов и фильтрации через условие в `HAVING`

- **Сортироовка**

```SQL
[ORDER BY ColumnList [ , ... ] [ { ASC | DESC } ] ]
```

Итоговый синтаксис `SELECT`:

```SQL
SELECT [ DISTINCT | ALL ] { * | ColumnExpression [ AS NewName ] [ , ... ] }
FROM TableName [ AS NewTableName ]
[ { INNER | LEFT OUTER | FULL } JOIN OuterTable [ AS NewOuterTableName ] ON
Condition ]
[ WHERE Condition ]
[ GROUP BY ColumnList [ , ... ] [ HAVING Condition ] ]
[ ORDER BY ColumnList [ , ... ] [ { ASC | DESC } ] ]
```

Порядок выполнения операций

1. **FROM**
2. **ON**
3. **JOIN**
4. **WHERE**
5. **GROUP BY**
6. **HAVING**
7. **SELECT**
8. **DISTINCT**
9. **ORDER BY**