# Работа с базами данных. Часть I

Croc Java School

# Реляционная модель данных

# Декартово произведение множеств

Декартовым (или прямым) произведением множеств $\mathbf{X}$ и $\mathbf{Y}$ называется множество, состоящее из всех возможных пар $(x, y)$ элементов этих множеств.

$$\mathbf{X} \times \mathbf{Y} = \{(x, y) \mid \forall x \in \mathbf{X}, \forall y \in \mathbf{Y}\}$$

# Отношение

Отношением $\mathbf{R}$ множеств $\mathbf{X}_1$, $\mathbf{X}_2$, ..., $\mathbf{X}_n$ называется подмножество декартова произведения этих множеств.

$$\mathbf{R} \subset \mathbf{X}_1 \times \mathbf{X}_2 \times \ldots \times \mathbf{X}_n$$

# Пример отношения

$\mathbf{S}$ = {круг, треугольник, квадрат}
$\mathbf{C}$ = {красный, зеленый}

$\mathbf{R} \subset \mathbf{S} \times \mathbf{C}$ = {
(треугольник, зеленый),
(треугольник, красный),
(круг, красный)
}

$\mathbf{R}$ — отношение на двух множествах или двумерное отношение.

# Структура отношения

В отношении принято выделять:

- схему (или заголовок) — множество пар (название атрибута, тип/домен);
- тело — множество кортежей: значений атрибутов, соответствующих схеме.

## В нашем примере

Схема = {
  (shape, string),
  (color, string)
}

Кортежи = {
  {(shape, треугольник), (color, зеленый)},
  {(shape, треугольник), (color, красный)},
  {(shape, круг), (color, красный)}
}

# Представление отношения в виде таблицы

| shape (string) | color (string) |
| --- | --- |
| треугольник | зеленый |
| треугольник | красный |
| круг | красный |

# Представление отношения в виде таблицы

В табличном представлении множество всех значений атрибута схемы называется столбцом/колонкой, а каждый кортеж — строкой таблицы.

Так как схема и тело отношения — множества, порядок колонок и строк в рамках отношения не определен.

# Реляционные базы данных

Набор данных, организованных в виде отношений (таблиц).

Каждая строка в таблице может быть помечена уникальным идентификатором или первичным ключом.

Строки из разных таблиц могут быть связаны внешними ключами.

Для организации доступа к данным обычно используется язык запросов SQL (Structured Query Language).

# Аспекты реляционных баз данных

Обеспечение целостности (ограничения)

- первичные ключи (primary keys)
- внешние ключи (foreign keys)
- ограничения типа `UNIQUE`, `NOT NULL`

Транзакционность

ACID

- атомарность
- согласованность
- изолированность
- долговечность

Язык запросов (SQL)

# Нормализация отношений

# Нормальные формы

Формальные критерии отношений, характеризующие их с точки зрения избыточности. Выделяется 6 нормальных форм (плюс 2 усиления), каждая из которых определяет более строгие критерии по отношению к предыдущей:
1NF, 2NF, 3NF, BCNF, 4NF, 5NF, DKNF, 6NF

Приведение системы отношений к более строгой нормальной форме называется нормализацией, обратно к более слабой — денормализацией.

На практике иногда приходится жертвовать нормализацией в целях оптимизации производительности.

# Реляционная алгебра

# Операции над отношениями

Выборка
Проекция
Пересечение
Объединение
Разность
Декартово произведение
Соединение

# Выборка

$\mathbf{S} = \mathbf{R}$ where $\varphi$

Выборка состоит из строк $\mathbf{R}$, удовлетворяющих условию $\varphi$. $\varphi$ — булева формула, которая применяется к кортежам из $\mathbf{R}$. Кортеж включается в результат выборки, если значение $\varphi$ на нем истинно.

Фильтрация таблицы по строкам.

# Выборка “красные фигуры”

$\mathbf{R}$ where color == “красный”

| shape (string) | color (string) |
| --- | --- |
| треугольник | красный |
| круг | красный |

# Проекция

$$\mathbf{P} = \mathbf{R}\{x, y, ..., z\}$$

Проекция состоит из перечисленных атрибутов $\mathbf{R}$.

Фильтрация таблицы по колонкам.

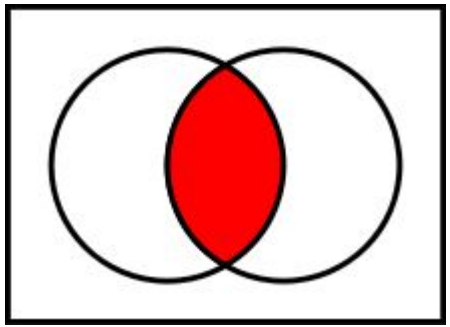
# Проекция “цвета”

**R** {color}

| color (string) |
| --- |
| зеленый |
| красный |

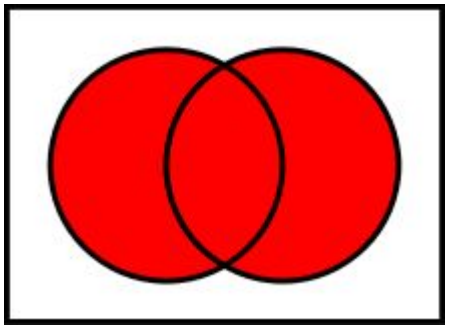
# Пересечение, объединение и разность

Для двух отношений $\mathbf{R}_1$ и $\mathbf{R}_2$ с одинаковыми схемами (заголовками).



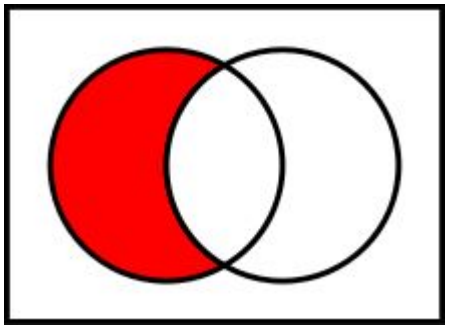
$\mathbf{R}_1 \cap \mathbf{R}_2 =$

$\{x \mid x \in \mathbf{R}_1 \wedge x \in \mathbf{R}_2\}$



$\mathbf{R}_1 \cup \mathbf{R}_2 =$

$\{x \mid x \in \mathbf{R}_1 \vee x \in \mathbf{R}_2\}$



$\mathbf{R}_1 \setminus \mathbf{R}_2 =$

$\{x \mid x \in \mathbf{R}_1 \wedge x \notin \mathbf{R}_2\}$

# Декартово произведение

$$\mathbf{J} = \mathbf{R}_1 \times \mathbf{R}_2$$

Схема $\mathbf{J}$ — объединение схем $\mathbf{R}_1$ и $\mathbf{R}_2$.

Тело $\mathbf{J}$ — декартово произведение множеств кортежей $\mathbf{R}_1$ и $\mathbf{R}_2$.

# Декартово произведение “фигуры и надписи”

$\mathbf{R}_1$ — таблица фигур, $\mathbf{R}_2$ — таблица знаков

| shape (string) | color (string) |
|---|---|
| треугольник | зеленый |
| треугольник | красный |
| круг | красный |

| label (string) |
|---|
| выход |
| парковка |

# Декартово произведение “фигуры и надписи”

$\mathbf{R}_1 \times \mathbf{R}_2$

| shape (string) | color (string) | label (string) |
|---|---|---|
| треугольник | зеленый | выход |
| треугольник | зеленый | парковка |
| треугольник | красный | выход |
| треугольник | красный | парковка |
| круг | красный | выход |
| круг | красный | парковка |

## Соединение

$\mathbf{J} = \mathbf{R}_1 \times \mathbf{R}_2$ where $\varphi$

Логически эквивалентна композиции (последовательному применению) операций декартова произведения и выборки по предикату $\varphi$.

# Соединение “красные парковки”

$\mathbf{R}_1 \times \mathbf{R}_2$ where $\mathbf{R}_1$.color = “красный” AND $\mathbf{R}_2$.label = “парковка”

| shape (string) | color (string) | label (string) |
|---|---|---|
| треугольник | красный | парковка |
| круг | красный | парковка |

# Эквивалентность формализмов (теорема Кодда)

## Теорема Кодда

Codd's theorem states that relational algebra and the domain-independent relational calculus queries, two well-known foundational query languages for the relational model, are precisely equivalent in expressive power.
— Wikipedia

# WAAAT?!

# Реляционная алгебра и SQL — это одно и то же

# SQL

# Схема

# CREATE TABLE

```
CREATE TABLE Figure(
  id INT PRIMARY KEY,
  shape VARCHAR(255) NOT NULL,
  color VARCHAR(255)
);

CREATE TABLE Sign(
  id INT PRIMARY KEY,
  figure INT NOT NULL,
  label VARCHAR(255),
  FOREIGN KEY (figure) REFERENCES Figure(id)
);
```

# DROP TABLE

```
DROP TABLE IF EXISTS Sign;
DROP TABLE IF EXISTS Figure;
```

# Данные

# SELECT

```
SELECT [DISTINCT] * | <атрибуты схемы>
FROM <таблица>
WHERE <предикат выборки>
ORDER BY <атрибуты схемы> ASC | DESC
```

## Все красные фигуры

```
SELECT *
FROM Figure f
WHERE f.color = 'красный'
```

Список атрибутов определяет порядок следования колонок в результате (проекция).
Звездочка означает “все атрибуты схемы”, но не фиксирует порядок.
f — алиас таблицы, который используется в списке атрибутов и предикатах для адресации атрибутов отношения.
Предикат WHERE задает фильтр по строкам.

## Какие фигуры зеленого цвета есть в таблице?

```
SELECT DISTINCT f.shape
FROM Figure f
WHERE f.color = 'зеленый'
```

Список атрибутов определяет порядок следования колонок в результате (проекция).
Звездочка означает “все атрибуты схемы”, но не фиксирует порядок.
Предикат WHERE задает фильтр по строкам.

# Операторы предикатов

```
AND, OR, NOT
=, <>, >, <, >=, <=
BETWEEN
LIKE
IS NULL
```

# JOIN

```
SELECT [DISTINCT] * | <атрибуты схем>
FROM <таблица 1>
    JOIN <таблица 2> ON <предикат соединения>
WHERE <предикат выборки>
ORDER BY <атрибуты схемы> ASC | DESC
```

# Соединим наши таблицы внешними ключами

# Внешние ключи

Figure

| id (int) | shape (string) | color (string) |
|---|---|---|
| 1 | треугольник | зеленый |
| 2 | треугольник | красный |
| 3 | круг | красный |

Sign

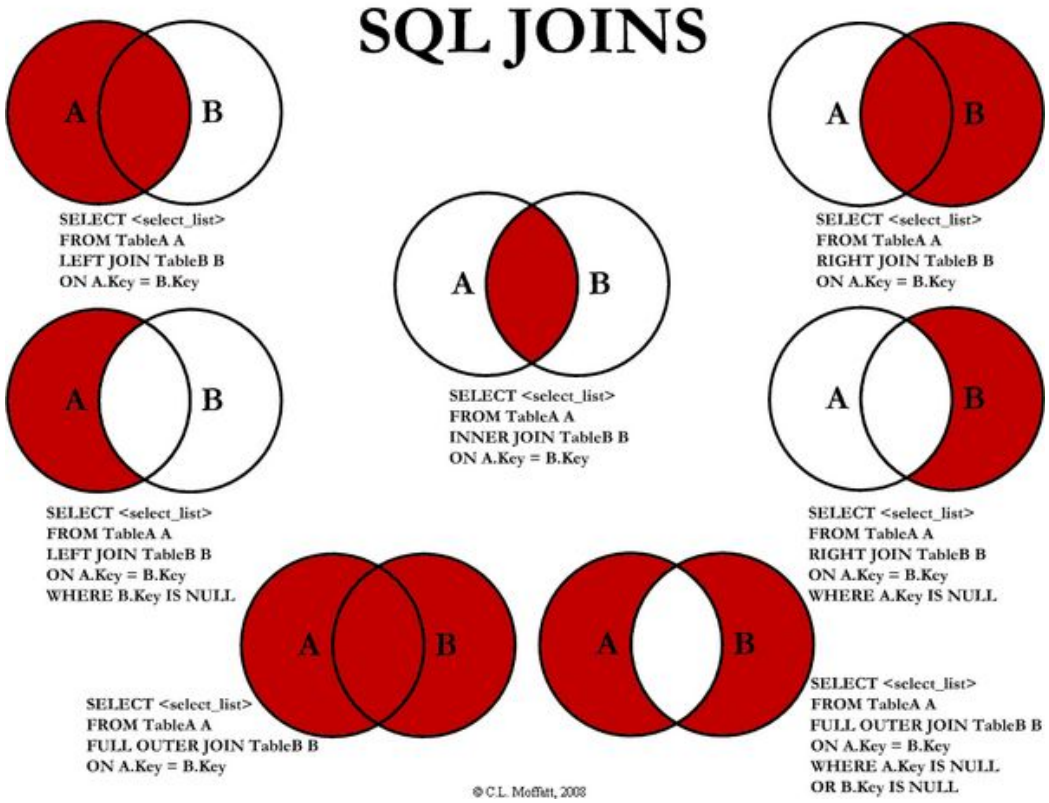
| figure (int) | label (string) |
|---|---|
| 1 | выход |
| 3 | парковка |
| 2 | только для персонала |

# Какие надписи размещаются на треугольниках?

```
SELECT s.label
FROM Figure f
    JOIN Sign s ON s.figure = f.id
WHERE f.shape = 'треугольник'
ORDER BY s.label
```

Предикат `s.figure = f.id` применяется к соединению, `f.shape = 'треугольник'` — к выборке.

# Джоины бывают разные



# INSERT

```
INSERT INTO <таблица> (<атрибуты>)
VALUES (<значения>)
```

# Вставка новой фигуры

```
INSERT INTO Figure (id, shape, color)
VALUES (4, 'прямоугольник', 'зеленый')
```

# UPDATE

```
UPDATE <таблица>
SET <атрибут> = <значение | выражение>
WHERE <предикат выборки>
```

## Сделать все треугольники розовыми

```
UPDATE Figure
SET color = 'розовый'
WHERE shape = 'треугольник'
```

# DELETE

```
DELETE FROM <таблица>
WHERE <предикат выборки>
```

# Ссылки

https://ru.wikipedia.org/wiki/Нормальная_форма#Нормальные_формы
https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins
http://webdam.inria.fr/Alice/