Базы данных

Определения

Атрибут** — свойство некоторой сущности. Часто называется полем таблицы.

Домен атрибута — множество допустимых значений, которые может принимать атрибут.

Кортеж — конечное множество взаимосвязанных допустимых значений атрибутов, которые вместе описывают некоторую сущность (строка таблицы).

Отношение — конечное множество кортежей (таблица).

Схема отношения — конечное множество атрибутов, определяющих некоторую сущность. Иными словами, это структура таблицы, состоящей из конкретного набора полей.

Проекция — отношение, полученное из заданного путём удаления и (или) перестановки некоторых атрибутов.

Функциональная зависимость между атрибутами (множествами атрибутов) X и Y означает, что для любого допустимого набора кортежей в данном отношении: если два кортежа совпадают по значению X, то они совпадают по значению Y. Например, если значение атрибута «Название компании» — Canonical Ltd, то значением атрибута «Штаб-квартира» в таком кортеже всегда будет Millbank Tower, London, United Kingdom. Обозначение: {X} -> {Y}.

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц).

Метод нормальных форм (НФ) состоит в сборе информации о объектах решения задачи в рамках одного отношения и последующей декомпозиции этого отношения на несколько взаимосвязанных отношений на основе процедур нормализации отношений.

**Цель нормализации: исключить избыточное дублирование данных, которое является причиной аномалий, возникших при добавлении, редактировании и удалении кортежей(строк таблицы).

Степень отношения - это число его атрибутов

Кардинальное число (мощность отношения) — это число его кортежей

Проекция — в результате операции формируется новое отношение, содержащее только те атрибуты из R, которые были указаны в проекции:

Нормализация

Первая нормальная форма

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Фирма	Модели
BMW	M5, X5M, M1
Nissan	GT-R

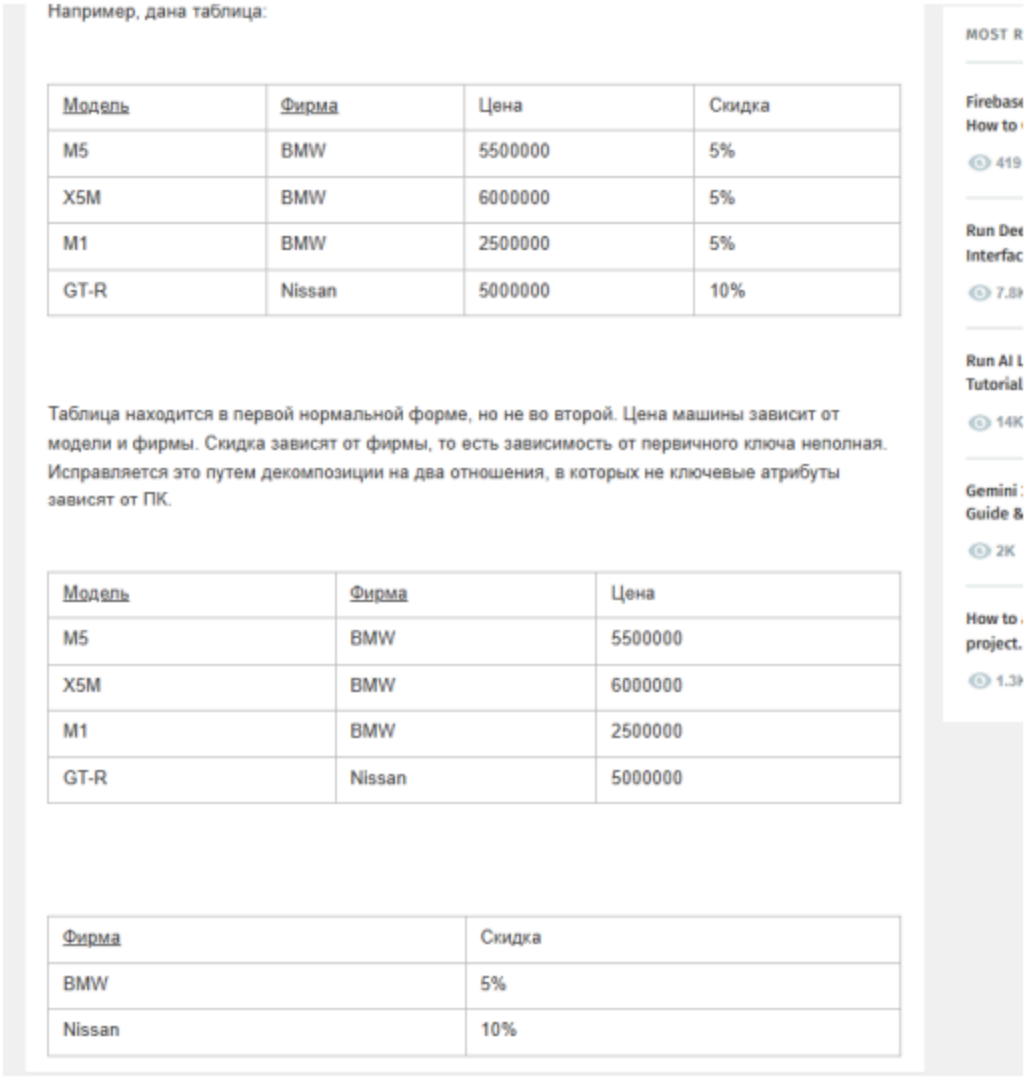
Нарушение нормализации 1НФ происходит в моделях BMW, т.к. в одной ячейке содержится список из 3 элементов: M5, X5M, M1, т.е. он не является атомарным. Преобразуем таблицу к 1НФ:

Фирма	Модели
BMW	M5
BMW	X5M
BMW	M1
Nissan	GT-R

Вторая НФ

Чтобы база данных находилась во второй нормальной форме (2NF), необходимо чтобы ее таблицы удовлетворяли следующим требованиям:

- Таблица должна находиться в первой нормальной форме
- Таблица должна иметь ключ
- Все неключевые столбцы таблицы должны зависеть от полного ключа (***в случае если он составной***)



Например, дана таблица:

Модель	Фирма	Цена	Скидка
M5	BMW	5500000	5%
X5M	BMW	6000000	5%
M1	BMW	2500000	5%
GT-R	Nissan	5000000	10%

Таблица находится в первой нормальной форме, но не во второй. Цена машины зависит от модели и фирмы. Скидка зависят от фирмы, то есть зависимость от первичного ключа неполная. Исправляется это путем декомпозиции на два отношения, в которых не ключевые атрибуты зависят от ПК.

Модель	Фирма	Цена
M5	BMW	5500000
X5M	BMW	6000000
M1	BMW	2500000
GT-R	Nissan	5000000

Фирма	Скидка
BMW	5%
Nissan	10%

MOST R

Firebas
How to

419

Run De
Interfac

7.8

Run AI L
Tutorial

14K

Gemini
Guide &

2K

How to
project.

1.3

3нф форма

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Нормальная форма Бойса-Кодда (НФБК) (частная форма третьей нормальной формы)

Представим, что у нас есть организация, которая реализует множество различных проектов. При этом в каждом проекте работа ведётся по нескольким функциональным направлениям, в каждом из которых есть свой куратор. Сотрудник может быть куратором только того направления, на котором он специализируется, т.е. если сотрудник программист, он не может курировать в проекте направление, связанное с бухгалтерией.

Допустим, что нам нужно хранить информацию о кураторах всех проектов по каждому направлению.

В итоге мы реализуем следующую таблицу, в которой первичный ключ составной *«Проект + Направление»*, так как в каждом проекте есть несколько направлений работы и поэтому, зная только проект, мы не можем определить куратора направления, так же как зная только направление, мы не сможем определить куратора, нам нужно знать и проект и направление, чтобы определить куратора этого направления в этом проекте.

Таблица проектов и кураторов.

Проект	**Направление**	**Куратор**
1	Разработка	Иванов И.И.
1	Бухгалтерия	Сергеев С.С.
2	Разработка	Иванов И.И.
2	Бухгалтерия	Петров П.П.
2	Реализация	John Smith
3	Разработка	Андреев А.А.

Наша таблица находится в третьей нормальной форме, так как у нас есть первичный ключ, а неключевой столбец зависит от всего ключа, а не от какой-то его части.

Но в данном случае таблица не находится в нормальной форме Бойса-Кодда, дело в том, что зная куратора, мы можем четко определить, какое направление он курирует, иными словами, часть составного ключа, т.е. *«Направление»*, зависит от неключевого атрибута, т.е. *«Куратора»*.

Чтобы привести данную таблицу к нормальной форме Бойса-Кодда, необходимо, как всегда сделать декомпозицию данного отношения, т.е. разбить эту таблицу на несколько таблиц.

Таблица кураторов.

Идентификатор куратора	**ФИО**	**Направление**
1	Иванов И.И.	Разработка
2	Сергеев С.С.	Бухгалтерия
3	Петров П.П.	Бухгалтерия

4	John Smith	Реализация
5	Андреев А.А.	Разработка

Таблица связи кураторов и проектов.

Проект	Идентификатор куратора
1	1
1	2
2	1
2	3
2	4
3	5

Таким образом, в таблице кураторов у нас хранится список кураторов и их специализация, т.е. направление, которое они могут курировать, а в таблице связи кураторов и проектов отражается связь проектов и кураторов.

4нф форма

https://info-comp.ru/fourth-normal-form

5нф форма

https://info-comp.ru/fifth-normal-form

6нф форма

https://info-comp.ru/sixth-normal-form

Денормализация

● Бывает, что для повышения производительности запросов производится денормализация: несколько отношений объединяют в одно;
● В результате: ➢ можно повысить эффективность выполнения некоторых запросов (уменьшается число соединений таблиц); ➢ увеличивается избыточность данных; ➢ требуется больше усилий на поддержание целостности БД;

Функциональные зависимости

Транзитивные

Частичные

A1 -> B, A -> B, B не зависит от A2

Полная

Полная функциональная зависимость: A2 в полной функциональной зависимости от A1 , если A1 → A2 , но нет зависимостей вида A3 → A2 , где A3 — подмножество A1 .

Аксиомы Армстронга

1. Рефлексивность: если A2 — подмножество A1 , то A1 → A2
2. Дополнение: если A1 → A2 , то A1 ,A3 → A2 ,A3
3. Транзитивность: если (A1 → A2) Λ (A2 → A3), то A1 → A3

Аномалии

Аномалии вставки

```
INSERT INTO STUDENTS
        VALUES(57, 'Nina Simonova', 'P3100', 'E. Kirov');
INSERT INTO STUDENTS
        VALUES(58, 'Petr Uvarov', 'P3100', 'Egor Lomov');
```

STUDENTS

StudID	StudName	Group	GrMentor
1	Ivan Petrov	P3100	Egor Kirov
3	Vasily Ivanov	P3101	Roman Ivov
34	Gleb Anisimov	P3100	Egor Kirov
57	Nina Simonova	**P3100**	**E.Kirov**
58	Petr Uvarov	**P3100**	**Egor Lomov**

Аномалии модификации

```
UPDATE STUDENTS
    SET GrMentor = 'Eugene Lomov'
    WHERE StudName = 'Ivan Petrov';
```

STUDENTS

StudID	StudName	Group	GrMentor
1	Ivan Petrov	P3100	**Eugene Lomov**
3	Vasily Ivanov	P3101	Roman Ivov
34	Gleb Anisimov	P3100	Egor Kirov

Аномалии удаления

```
DELETE FROM STUDENTS
    WHERE StudName = 'Vasily Ivanov';
```

STUDENTS

StudID	StudName	Group	GrMentor
1	Ivan Petrov	P3100	Eugene Lomov
~~3~~	~~Vasily Ivanov~~	~~P3101~~	~~Egor Kirov~~
34	Gleb Anisimov	P3100	Egor Kirov

- Данных о группе P3101 больше нет.

Различие триггерной функции и обычной

Обычные функции

1. **Вызов**: Вызываются явно из кода приложения или запроса
2. **Выполнение**: Выполняются только когда их явно вызывают
3. **Возврат значения**: Всегда возвращают результат (если не void)

Триггерные функции

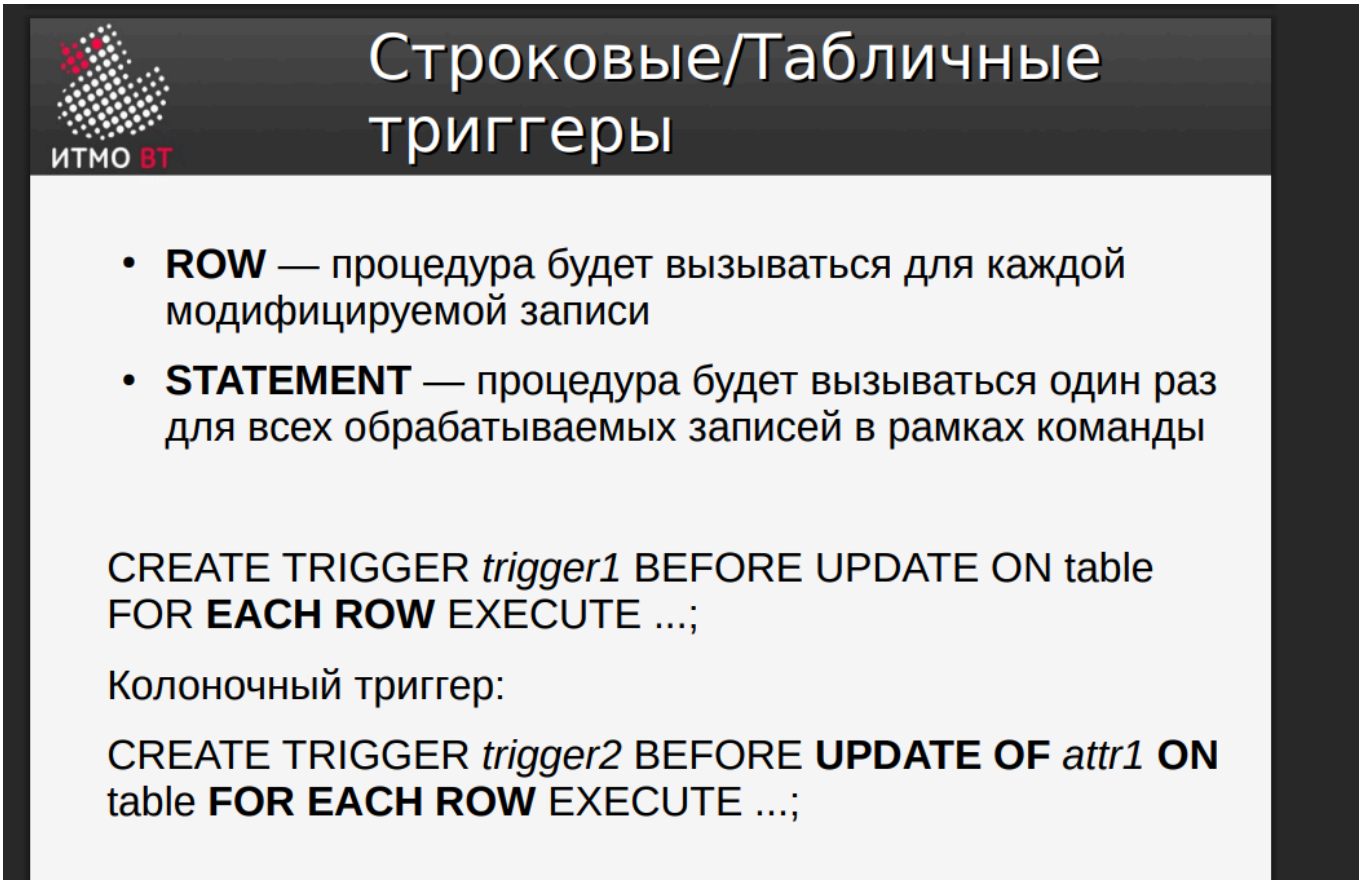
1. **Вызов**: Автоматически вызываются при наступлении определенного события (INSERT, UPDATE, DELETE)
2. **Выполнение**: Срабатывают неявно в ответ на изменения данных
3. **Контекст**: Имеют доступ к специальным переменным (NEW, OLD в PostgreSQL)
4. **Возврат значения**: Часто возвращают модифицированные данные или ничего (зависит от СУБД)

PL/pg sql

PL/pgSQL это процедурный блочный язык для СУБД PostgreSQL.

```
Нужен для процедур(функций), триггеров, транзакций
```

Строковые/табличные триггеры



что такое TG_ ?

Триггер изменения данных объявляется как функция без аргументов и возвращаемым типом `trigger`. Обратите внимание, что функция должна быть объявлена без аргументов, даже если она ожидает получить некоторые аргументы, указанные в `CREATE TRIGGER` — такие аргументы передаются через `TG_ARGV`, как описано ниже.

Когда функция PL/pgSQL вызывается как триггер, в блоке верхнего уровня автоматически создаются несколько специальных переменных. Это:

`NEW record`

новая строка базы данных для `INSERT` / `UPDATE` операций в триггерах уровня строк. Эта переменная имеет значение null в триггерах уровня операторов и для `DELETE` операций.

`OLD record`

старая строка базы данных для `UPDATE` / `DELETE` операций в триггерах уровня строк. Эта переменная имеет значение null в триггерах уровня операторов и для `INSERT` операций.

`TG_NAME name`

название сработавшего триггера.

`TG_WHEN` `text`

`BEFORE`, `AFTER`, или `INSTEAD OF`, в зависимости от определения триггера.

`TG_LEVEL` `text`

`ROW` или `STATEMENT`, в зависимости от определения триггера.

`TG_OP` `text`

операция, для которой был срабатывает триггер: `INSERT`, `UPDATE`, `DELETE`, или `TRUNCATE`.

`TG_RELID` `oid` (ссылки `pg_class`. `oid`)

Идентификатор объекта таблицы, вызвавшей вызов триггера.

`TG_RELNAME` `name`

таблица, вызвавшая вызов триггера. Теперь это устарело и может исчезнуть в будущем выпуске. Используйте `TG_TABLE_NAME` вместо этого.

`TG_TABLE_NAME` `name`

таблица, вызвавшая вызов триггера.

`TG_TABLE_SCHEMA` `name`

схема таблицы, вызвавшей вызов триггера.

`TG_NARGS` `integer`

количество аргументов, переданных триггерной функции в `CREATE TRIGGER` операторе.

`TG_ARGV` `text[]`

аргументы из `CREATE TRIGGER` оператора. Индекс отсчитывается от 0. Недопустимые индексы (меньше 0 или больше или равны `tg_nargs`) приводят к нулевому значению.