#### Министерство науки и высшего образования Российской Федерации

#### Федеральное государственное бюджетное образовательное учреждение

#### высшего образования

#### «Владимирский государственный университет

#### имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа

по дисциплине

"Бизнес анализ данных"

Тема доклада:

Нормализация баз данных. Виды нормальных форм (1-6). Схемы типа «звезда» и «снежинка».

Сравнительный анализ схем организации БД

Выполнил:

ст. гр. ПРИм-124

Парахин К.В.

Приняла:

Преподаватель

кафедры ИСПИ

Тимофеев А.А.

Владимир, 2025 г.

Начнем с понятия нормализации баз данных

Нормализация базы данных — это процесс организации данных для минимизации избыточности и обеспечения целостности. Она предполагает разбиение таблиц и установление между ними связей. В процессе нормализации используются различные нормальные формы (НФ), каждая из которых накладывает определённые ограничения на структуру данных.

Теперь проведем обзор и сравнение различных нормальных форм

1. Первая нормальная форма (1НФ)

Является базовой формой представления данных  
Все атрибуты содержат только атомарные (неделимые) значения.

Отсутствуют повторяющиеся группы данных.

1. Вторая нормальная форма (2НФ)

Данные соответствуют 1НФ.

Каждый неключевой атрибут зависит от всего первичного ключа, а не от его части.

1. Третья нормальная форма (3НФ)

Данные соответствуют 2НФ.

Все неключевые атрибуты зависят только от первичного ключа, а не от других неключевых атрибутов – то есть исключаются все транзитивные зависимости.

1. Нормальная форма Бойса-Кодда (НФБК)

Данные соответствуют 3НФ.

Каждый детерминант является потенциальным ключом. Детерминант – это определяющее множество атрибутов для каждой выборки данных

1. Четвертая нормальная форма (4НФ)

Данные соответствуют НФБК.

В таблице отсутствуют многозначные зависимости.

1. Пятая нормальная форма (5НФ)

Данные соответствуют 4НФ.

Исключены избыточные соединения, возникающие из-за избыточной декомпозиции – устраняются зависимости соединения

1. Шестая нормальная форма (6НФ)

Данные соответствуют 5НФ.

Учитываются временные зависимости и поддерживаются гибкие структуры хранения данных – данные разбиваются на максимально независимые отношения

Далее поговорим о схемах организации баз данных, таких как «звезда» и «снежинка»

Эти схемы используются в хранилищах данных (Data Warehouse) и системах OLAP для организации многомерных данных.

Схема «Звезда»

Состоит из одной центральной фактовой таблицы, содержащей ключевые показатели (например, продажи, доход), и связанных с ней измерений.

Плюсы:

Простая и понятная структура.

Высокая скорость выполнения запросов.

Упрощенное администрирование.

Минусы:

Избыточность данных в таблицах измерений (повторяющаяся информация).

Схема «Снежинка»

Расширяет схему «Звезда», нормализуя таблицы измерений (разделяя их на подкатегории).

Плюсы:

Уменьшает объем хранимых данных.

Более структурированное хранение информации.

Минусы:

Сложнее запросы и их выполнение.

Усложненное администрирование.

Схема «снежинка» получается более сложная (имеет больше подкатегорий), чуть менее производительная (из-за использования дополнительных объединяющих операторов), но более оптимизированная по размеру хранимых данных  
Она подходит для экономии памяти при накоплении больших данных и решении более сложных аналитических задач – схема «звезда» при этом предпочтительна для быстрого анализа данных, для несложных предметный аналитических запросов

Далее рассмотрим представленные выше понятия в контексте конкретной тестовой предметной области – предметной области ставок на спортивные события

Данные данной предметной области связаны с набором произведенных ставок на различные спортивные события в рамках букмекерской организации

Связанный dataset можно посмотреть по ссылке: <https://www.kaggle.com/datasets/zygmunt/betfair-sports>

Он представлен в формате .csv, содержит чуть более 1 млн записей  
В нем представлены следующие колонки: sport\_id, event\_id, settled\_date, full\_description, scheduled\_off, event, dt actual\_off, selection\_id, selection, odds.

Он включает данные о различных видах спорта (sport\_id), событиях (event\_id), времени начала матчей (scheduled\_off, dt actual\_off), описании событий (full\_description, event), исходах (selection) и коэффициентах (odds). Также указаны даты завершения ставок (settled\_date) и идентификаторы исходов (selection\_id).

Решаемая задача в области анализа данных - анализ поведения пользователей и эффективности ставок на основе исторических данных

Основные сценарии использования аналитической подсистемы:

За указанный период для каждого спорта представить отчет по данным, в котором выявлены аномалии в коэффициентах исходов различных видов ставок. Данные представить в разрезе спорта и вида ставки-исхода.

За указанный период для каждого спорта представить данные с анализом популярности и эффективности различных видов ставок. Данные представить в разрезе спорта и вида ставки – отсортировав по относительной популярности – в процентах от общего кол-ва ставок.

Для данной системы могут быть выполнены следующие аналитические запросы.

Сначала представлю пример двух достаточно простых запросов:

* Анализ популярности видов спорта (запрос на топ-5, топ-10 самых популярных и, соответственно, прибыльных видов спорта среди игроков)
* Анализ времени отклонения начала событий (можно найти среднее отклонение времени запланированного и реального старта спортивного события – что можем потенциально использоваться системой принятия ставок для внутренних оптимизаций)

Более сложные аналитические запросы:

* Выявление аномалий в коэффициентах ставок (можно проанализировать медианные значения коэффициентов исходов событий конкретных видов спорта – и найти те подозрительные события, в которых отклонения достаточно большие)
* Анализ эффективности конкретных видов ставок (для определенного спорта вывести самые популярные виды ставок – и, соответственно, исходов, на которые игроки ставили чаще всего)

Для данных можно выделить следующие сущности: Sport, Event, Bet, Selection

В нотации UML для сущностей данных этой предметной области была построена UML Class Diagram:

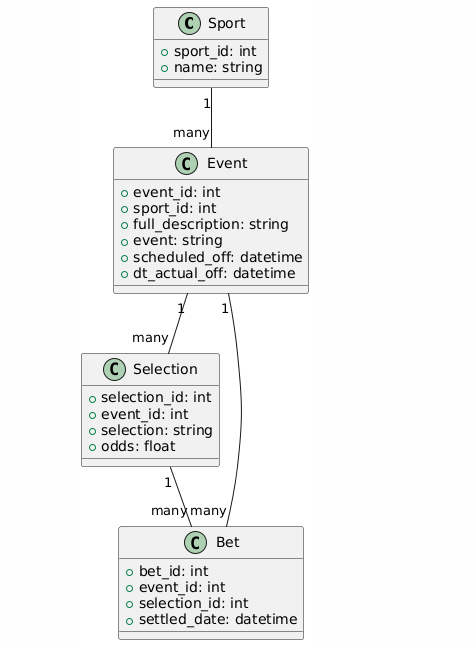


Рисунок 1. Диаграмма классов

Далее пойдем в сторону реализации физической схемы базы данных. Для этого можно использовать ER-диаграмму:

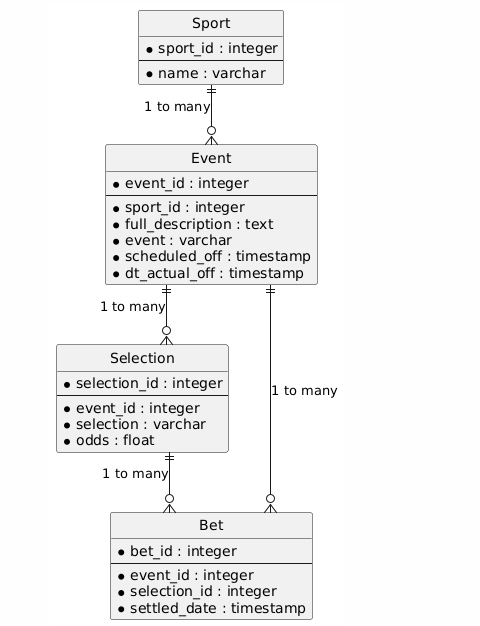


Рисунок 2. ER-диаграмма

И на примере реляционной СУБД Postgre SQL представим скрипт создания таблиц реляционной БД:

CREATE TABLE Sport (

sport\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL

);

CREATE TABLE Event (

event\_id SERIAL PRIMARY KEY,

sport\_id INT REFERENCES Sport(sport\_id),

full\_description TEXT,

event VARCHAR(200),

scheduled\_off TIMESTAMP,

dt\_actual\_off TIMESTAMP

);

CREATE TABLE Selection (

selection\_id SERIAL PRIMARY KEY,

event\_id INT REFERENCES Event(event\_id),

selection VARCHAR(200),

odds FLOAT

);

CREATE TABLE Bet (

bet\_id SERIAL PRIMARY KEY,

event\_id INT REFERENCES Event(event\_id),

selection\_id INT REFERENCES Selection(selection\_id),

settled\_date TIMESTAMP

);

Далее я пробую заполнить данную БД данными и провести на основе этого OLTP простые аналитические запросы:

Запрос получения самых популярных видов спорта:

SELECT

S.name AS sport\_name,

COUNT(B.bet\_id) AS total\_bets

FROM

Bet B

JOIN Event E ON B.event\_id = E.event\_id

JOIN Sport S ON E.sport\_id = S.sport\_id

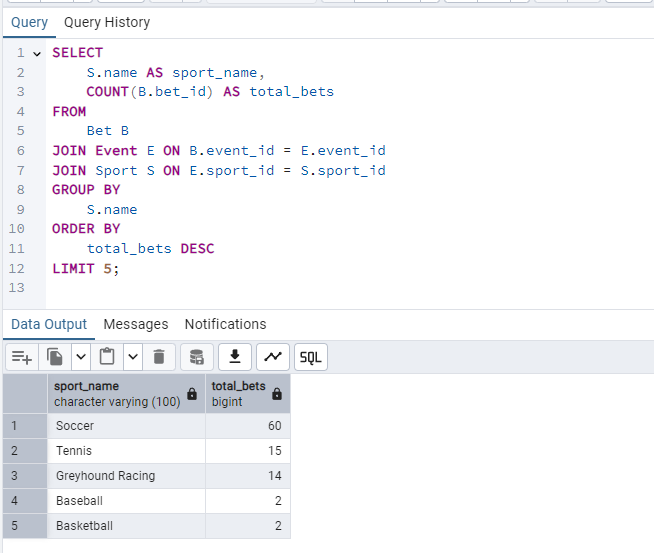
GROUP BY

S.name

ORDER BY

total\_bets DESC

LIMIT 5;



Запрос получения среднего отклонения начала спортивного события от запланированного – в зависимости от вида спорта:

SELECT

s.sport\_id,

s.name,

AVG(ABS(EXTRACT(EPOCH FROM (e.dt\_actual\_off - e.scheduled\_off)))) AS avg\_delay\_seconds

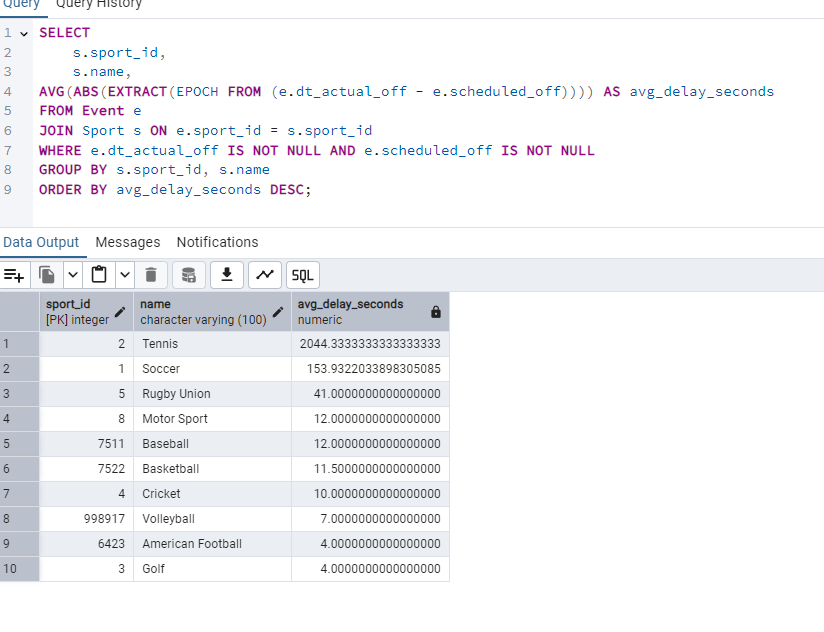
FROM Event e

JOIN Sport s ON e.sport\_id = s.sport\_id

WHERE e.dt\_actual\_off IS NOT NULL AND e.scheduled\_off IS NOT NULL

GROUP BY s.sport\_id, s.name

ORDER BY avg\_delay\_seconds DESC;



Далее проведем анализ нормализации данной БД на основе описанных в начале доклада совокупности 6 нормальных форм

Данная БД находится в 1 нормальной форме – так как все поля содержат неделимые значения, каждая строка уникальная и содержит первичный ключ, массивы отстутствуют

Данная БД находится во 2 нормальной форме – так как в каждой таблице все не ключевые поля зависят только от своих первичных ключей

Данная БД находится в 3 нормальной форме – в каждой таблице все неключевые атрибуты не зависят транзитивно от первичного ключа:

*В Event все поля связаны с event\_id,*

*В Selection все поля зависят от selection\_id, без транзитивных зависимостей.*

*В Bet все данные привязаны к bet\_id.*

БД также находится в нормальной форме Бойса-Кодда – так как в ней нет на каждую таблицу нескольких детерминант – то есть нету многозначных зависимостей – и все зависимости относятся исключительно к первичному ключу

БД также соответствует 4 нормальной форме – так как в ней отсутствуют многозначные зависимости между соответствующими внешними ключами на первичные ключи других таблиц

*В Selection каждый selection\_id может относиться только к одному event\_id.*

*В Bet каждое событие (event\_id) связано с только одной ставкой selection\_id.*

БД также соответствует 5 нормальной форме – так как в ней отсутствуют какие-либо зависимости соединения связанных отношений (через оператор JOIN)

Но БД не соответствует 6 нормальной форме – так как в ней нету отдельных таблиц для детерминации временных наборов данных

Для преобразования БД в 6 нормальную форму – необходимо разбить таблицы с bets и selections на временные сущности, как, например:

*Event(event\_id, sport\_id, full\_description, event\_type\_id)*

*EventSchedule(event\_id, scheduled\_off)*

*EventActual(event\_id, dt\_actual\_off)*

Далее рассмотрим возможные преобразования данного транзакционного хранилища данных (OLTP) – в аналитические хранилища данных OLAP – в виде схем организации БД, таких как: схема «звезда» и схема «снежинка»

Для схемы «звезда» создадим центральную таблицу фактов (которая будет хранить измеряемые показатели, ставки и их коэффициенты) и таблицу измерений (доп. характеристики: вид спорта, событие, ставка, дата ставки)

Таблица фактов:

CREATE TABLE Fact\_Bets (

bet\_id SERIAL PRIMARY KEY,

event\_id INT,

selection\_id INT,

sport\_id INT,

odds FLOAT,

settled\_date TIMESTAMP,

bet\_amount DECIMAL(10,2),

payout DECIMAL(10,2));

Таблицы измерений:

CREATE TABLE Dim\_Sport (

sport\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL);

CREATE TABLE Dim\_Event (

event\_id SERIAL PRIMARY KEY,

sport\_id INT REFERENCES Dim\_Sport(sport\_id),

event VARCHAR(200),

scheduled\_off TIMESTAMP,

dt\_actual\_off TIMESTAMP);

CREATE TABLE Dim\_Selection (

selection\_id SERIAL PRIMARY KEY,

event\_id INT REFERENCES Dim\_Event(event\_id),

selection VARCHAR(200));

CREATE TABLE Dim\_Date (

date\_id SERIAL PRIMARY KEY,

full\_date DATE,

year INT,

month INT,

day INT,

weekday VARCHAR(10));

Благодаря такой своеобразной структуре данная БД может помочь аналитике в более сложном аналитическом запросе, например, получение самых популярных видов ставок по видам спорта:

SELECT

s.name AS sport\_name,

sel.selection,

COUNT(\*) AS bet\_count

FROM Fact\_Bets fb

JOIN Dim\_Sport s ON fb.sport\_id = s.sport\_id

JOIN Dim\_Selection sel ON fb.selection\_id = sel.selection\_id

GROUP BY s.name, sel.selection

ORDER BY bet\_count DESC

LIMIT 10;

Для схемы «снежинка» оставим ту же таблицу фактов, но дополнительно проведем нормализацию таблиц измерений – а именно выделим отдельные таблицы: Dim\_Event\_Type, Dim\_Selection\_Type, Dim\_Sport\_Category, а также уберем избыточные данные из таблицы Events – для оптимизации хранения данных

Измененные таблицы измерений:

CREATE TABLE Dim\_Sport (

sport\_id SERIAL PRIMARY KEY,

category\_id INT,

name VARCHAR(100) NOT NULL);

CREATE TABLE Dim\_Sport\_Category (

category\_id SERIAL PRIMARY KEY,

category\_name VARCHAR(100));

CREATE TABLE Dim\_Event (

event\_id SERIAL PRIMARY KEY,

sport\_id INT REFERENCES Dim\_Sport(sport\_id),

event\_type\_id INT,

scheduled\_off TIMESTAMP,

dt\_actual\_off TIMESTAMP);

CREATE TABLE Dim\_Event\_Type (

event\_type\_id SERIAL PRIMARY KEY,

type\_name VARCHAR(100));

CREATE TABLE Dim\_Selection (

selection\_id SERIAL PRIMARY KEY,

event\_id INT REFERENCES Dim\_Event(event\_id),

selection\_type\_id INT);

CREATE TABLE Dim\_Selection\_Type (

selection\_type\_id SERIAL PRIMARY KEY,

selection VARCHAR(200));

Благодаря такой своеобразной структуре данная БД может помочь аналитике в еще более сложном аналитическом запросе, например, получение выборки аномальных коэффициентов на исходы спортивных событий:

SELECT

sc.category\_name AS sport\_category,

s.name AS sport\_name,

et.type\_name AS event\_type,

sel\_type.selection AS selection\_type,

AVG(fb.odds) AS avg\_odds,

PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY fb.odds) AS median\_odds,

(AVG(fb.odds) - PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY fb.odds)) AS deviation

FROM Fact\_Bets fb

JOIN Dim\_Sport s ON fb.sport\_id = s.sport\_id

JOIN Dim\_Sport\_Category sc ON s.category\_id = sc.category\_id

JOIN Dim\_Event e ON fb.event\_id = e.event\_id

JOIN Dim\_Event\_Type et ON e.event\_type\_id = et.event\_type\_id

JOIN Dim\_Selection sel ON fb.selection\_id = sel.selection\_id

JOIN Dim\_Selection\_Type sel\_type ON sel.selection\_type\_id = sel\_type.selection\_type\_id

GROUP BY sc.category\_name, s.name, et.type\_name, sel\_type.selection

HAVING ABS(AVG(fb.odds) - PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY fb.odds)) > 0.5

ORDER BY deviation DESC;