Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МОСКОВСКИЙ ПОЛИТЕХ)

Факультет информационных технологий

Кафедра «Прикладная информатика» Форма обучения: очная

Лабораторная работа № 2

Тема: «Реализация структуры Data Vault»

по дисциплине

«Веб-аналитика»

|  |  |  |
| --- | --- | --- |
| Студент | (личная подпись) | А.Д. Филипенко |

Москва 2025

# ВВЕДЕНИЕ

В современном мире информационные технологии играют ключевую роль в управлении данными, особенно в таких областях, как система учета видеопроката. Эффективное хранение, обработка и анализ данных требуют использования структурированных баз данных, которые обеспечивают целостность, минимальную избыточность и высокую производительность. В данной лабораторной работе рассматривается процесс проектирования и создания базы данных для системы учета видеопроката, основанной на двух различных моделях: нормализованной модели (3NF) и модели Data Vault.

Нормализация до третьей нормальной формы (3NF) позволяет устранить избыточность данных и обеспечить их логическую целостность, что особенно важно для систем с высокой нагрузкой на чтение и запись. С другой стороны, модель Data Vault, основанная на принципах гибкости и масштабируемости, предоставляет возможность эффективного управления большими объемами данных, что актуально для систем, где требуется интеграция данных из различных источников и поддержка историчности.

Целью данной работы является изучение и применение принципов проектирования баз данных на основе модели 3NF и Data Vault, а также сравнение их преимуществ и недостатков в контексте системы учета видеопроката. В ходе работы будут разработаны структуры баз данных, соответствующие обеим моделям, и проведен анализ их применимости для решения задач учета и анализа данных в видеопрокате.

# Создание БД в PostgreSQL 3NF

В рамках первой части лабораторной работы нам необходимо спроектировать базу данных системы учета видеопроката.

Чтобы база данных соответствовала требованиям третьей нормальной формы (3NF), необходимо выполнить несколько шагов. Сначала нужно убедиться, что база данных соответствует первой нормальной форме (1NF). Это означает, что каждая таблица должна иметь уникальное имя, все атрибуты (столбцы) должны быть атомарными, то есть содержать только одно значение, и каждая строка в таблице должна быть уникальной, что обычно обеспечивается за счет использования первичного ключа.

Далее необходимо убедиться, что база данных соответствует второй нормальной форме (2NF). Для этого таблица должна уже соответствовать 1NF, а все неключевые атрибуты (столбцы, не входящие в первичный ключ) должны зависеть от всего первичного ключа, а не от его части. Если первичный ключ составной, то каждый неключевой атрибут должен зависеть от всех его компонентов.

После этого можно привести базу данных к третьей нормальной форме (3NF). Таблица должна соответствовать 2NF, и нужно убедиться, что все неключевые атрибуты не зависят друг от друга. Другими словами, не должно быть транзитивных зависимостей, где один неключевой атрибут зависит от другого неключевого атрибута. Если такие зависимости обнаружены, необходимо разделить таблицу на несколько, чтобы устранить транзитивные зависимости.

Для базы данных системы учета видеопроката были определены следующие необходимые таблицы:

* **Фильмы (Movies)**
* **Клиенты (Customers)**
* **Аренда (Rentals)**
* **Жанры (Genres)**
* **Актеры (Actors)**
* **Фильмы-Актеры (MovieActors)** (связь многие-ко-многим между фильмами и актерами)

С учётом соответствия базы данных третьей нормальной форме была получена следующая модель (Рисунок 1):

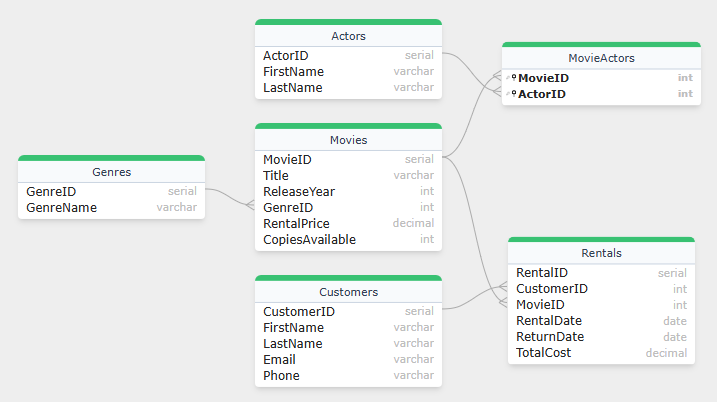


Рисунок 1 – Графическая модель базы данных

База данных, спроектированная для системы учета видеопроката, состоит из нескольких сущностей, каждая из которых играет важную роль в организации и управлении данными.

Фильмы (Movies) — это основная сущность, которая хранит информацию о фильмах, доступных для аренды. В ней содержатся такие данные, как название фильма, год выпуска, жанр, стоимость аренды и количество доступных копий. Эта таблица позволяет управлять каталогом фильмов и отслеживать их доступность для клиентов.

Клиенты (Customers) — сущность, предназначенная для хранения информации о клиентах видеопроката. В ней регистрируются данные о клиентах, такие как имя, фамилия, электронная почта и телефон. Это позволяет идентифицировать клиентов и связывать их с арендами, которые они совершают.

Аренда (Rentals) — сущность, которая фиксирует факты аренды фильмов клиентами. В ней хранятся данные о том, какой клиент арендовал какой фильм, дата аренды, дата возврата и общая стоимость аренды. Эта таблица является ключевой для учета операций видеопроката.

Жанры (Genres) — сущность, которая содержит информацию о жанрах фильмов. Она позволяет классифицировать фильмы по категориям, что упрощает поиск и фильтрацию фильмов для клиентов.

Актеры (Actors) — сущность, хранящая данные об актерах, снимавшихся в фильмах. В ней регистрируются имена и фамилии актеров, что позволяет связывать их с конкретными фильмами.

Фильмы-Актеры (MovieActors) — это вспомогательная сущность, которая реализует связь многие-ко-многим между фильмами и актерами. Она позволяет одному фильму быть связанным с несколькими актерами и наоборот, что отражает реальную структуру данных в кинематографе.

Каждая из этих сущностей важна для обеспечения полноценного функционирования системы учета видеопроката. Они позволяют эффективно управлять данными о фильмах, клиентах, арендах, жанрах и актерах, обеспечивая целостность и непротиворечивость информации..

Чтобы создать данную БД в PostgreSQL, необходимо перейти в PsqlShell и создать там базу данных «video\_store». После создания базы данных необходимо перейти в pgAdmin4 и создать необходимые таблицы следующим sql запросом:

CREATE TABLE Genres (

GenreID SERIAL PRIMARY KEY,

GenreName VARCHAR(255) NOT NULL

);

CREATE TABLE Actors (

ActorID SERIAL PRIMARY KEY,

FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL

);

CREATE TABLE Customers (

CustomerID SERIAL PRIMARY KEY,

FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL,

Email VARCHAR(255),

Phone VARCHAR(15)

);

CREATE TABLE Movies (

MovieID SERIAL PRIMARY KEY,

Title VARCHAR(255) NOT NULL,

ReleaseYear INT,

GenreID INT,

RentalPrice DECIMAL(10, 2),

CopiesAvailable INT,

FOREIGN KEY (GenreID) REFERENCES Genres(GenreID)

);

CREATE TABLE MovieActors (

MovieID INT,

ActorID INT,

PRIMARY KEY (MovieID, ActorID),

FOREIGN KEY (MovieID) REFERENCES Movies(MovieID),

FOREIGN KEY (ActorID) REFERENCES Actors(ActorID)

);

CREATE TABLE Rentals (

RentalID SERIAL PRIMARY KEY,

CustomerID INT,

MovieID INT,

RentalDate DATE NOT NULL,

ReturnDate DATE,

TotalCost DECIMAL(10, 2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (MovieID) REFERENCES Movies(MovieID)

);

На этом создание базы данных в 3 нормальной форме закончено, все необходимые таблицы были созданы и корректно связаны.

# Создание БД в PostgreSQL Data Vault

Методология Data Vault, разработанная Дэном Линстедом в начале 2000-х, предназначена для создания гибких, масштабируемых и устойчивых к изменениям хранилищ данных. Она особенно полезна в областях, где требуется адаптивность к изменениям бизнес-требований, таких как финансы, логистика и ритейл. Основные принципы Data Vault включают гибкость, поддержку историчности данных и четкое разделение данных и метаданных, что упрощает управление и анализ.

Data Vault состоит из трех типов таблиц: хабы (Hubs), связи (Links) и спутники (Satellites). Хабы хранят уникальные бизнес-ключи, связи описывают отношения между хабами, а спутники содержат описательные атрибуты и историю изменений. Например, хаб может хранить идентификаторы клиентов, связь — информацию о заказах, а спутник — историю изменения адресов клиентов.

Преимущества Data Vault включают адаптивность к изменениям, поддержку больших объемов данных, сохранение истории изменений и упрощение интеграции данных из различных источников. Однако методология требует глубокого понимания, может приводить к большому количеству таблиц и усложнению запросов, что требует дополнительных усилий для оптимизации.

Методология Data Vault предназначена для создания гибких, масштабируемых и устойчивых к изменениям хранилищ данных. Она основана на трех основных типах таблиц: Hub (центры), Link (связи) и Satellite (спутники). Реализация базы данных для системы учета видеопроката в методологии Data Vault выглядит следующим образом:

1. Хабы:

-- Хабы

CREATE TABLE Hub\_Movie (

MovieID SERIAL PRIMARY KEY,

MovieKey VARCHAR(255) NOT NULL UNIQUE,

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255)

);

CREATE TABLE Hub\_Customer (

CustomerID SERIAL PRIMARY KEY,

CustomerKey VARCHAR(255) NOT NULL UNIQUE,

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255)

);

CREATE TABLE Hub\_Actor (

ActorID SERIAL PRIMARY KEY,

ActorKey VARCHAR(255) NOT NULL UNIQUE,

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255)

);

CREATE TABLE Hub\_Genre (

GenreID SERIAL PRIMARY KEY,

GenreKey VARCHAR(255) NOT NULL UNIQUE,

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255)

);

1. Ссылки:

-- Связи

CREATE TABLE Link\_MovieActor (

MovieID INT,

ActorID INT,

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255),

PRIMARY KEY (MovieID, ActorID),

FOREIGN KEY (MovieID) REFERENCES Hub\_Movie(MovieID),

FOREIGN KEY (ActorID) REFERENCES Hub\_Actor(ActorID)

);

CREATE TABLE Link\_Rental (

RentalID SERIAL PRIMARY KEY,

CustomerID INT,

MovieID INT,

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255),

FOREIGN KEY (CustomerID) REFERENCES Hub\_Customer(CustomerID),

FOREIGN KEY (MovieID) REFERENCES Hub\_Movie(MovieID)

);

1. Спутники:

-- Спутники

CREATE TABLE Sat\_Movie (

MovieID INT,

Title VARCHAR(255) NOT NULL,

ReleaseYear INT,

RentalPrice DECIMAL(10, 2),

CopiesAvailable INT,

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255),

PRIMARY KEY (MovieID, LoadDate),

FOREIGN KEY (MovieID) REFERENCES Hub\_Movie(MovieID)

);

CREATE TABLE Sat\_Customer (

CustomerID INT,

FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL,

Email VARCHAR(255),

Phone VARCHAR(15),

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255),

PRIMARY KEY (CustomerID, LoadDate),

FOREIGN KEY (CustomerID) REFERENCES Hub\_Customer(CustomerID)

);

CREATE TABLE Sat\_Actor (

ActorID INT,

FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL,

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255),

PRIMARY KEY (ActorID, LoadDate),

FOREIGN KEY (ActorID) REFERENCES Hub\_Actor(ActorID)

);

CREATE TABLE Sat\_Genre (

GenreID INT,

GenreName VARCHAR(255) NOT NULL,

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255),

PRIMARY KEY (GenreID, LoadDate),

FOREIGN KEY (GenreID) REFERENCES Hub\_Genre(GenreID)

);

CREATE TABLE Sat\_Rental (

RentalID INT,

RentalDate DATE NOT NULL,

ReturnDate DATE,

TotalCost DECIMAL(10, 2),

LoadDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

RecordSource VARCHAR(255),

PRIMARY KEY (RentalID, LoadDate),

FOREIGN KEY (RentalID) REFERENCES Link\_Rental(RentalID)

);

# Заключение

В ходе выполнения лабораторной работы была спроектирована и реализована база данных для системы учета видеопроката на основе двух методологий: третьей нормальной формы (3NF) и Data Vault. Обе модели были успешно применены для организации структуры данных, что позволило сравнить их преимущества и особенности.

Использование 3NF обеспечило минимальную избыточность данных и высокую целостность, что особенно важно для систем с интенсивными операциями чтения и записи. Нормализация данных до третьей нормальной формы позволила четко разделить сущности, устранить аномалии и упростить поддержку базы данных. Однако эта модель может быть менее гибкой при необходимости интеграции данных из множества источников или при частых изменениях бизнес-правил.

С другой стороны, методология Data Vault продемонстрировала свою эффективность в создании гибкой и масштабируемой структуры данных. Благодаря разделению на хабы, связи и спутники, база данных стала устойчивой к изменениям и способной поддерживать историчность данных. Это особенно важно для систем, где требуется анализ данных в разрезе времени или интеграция данных из различных источников.

Обе модели имеют свои преимущества и могут быть использованы в зависимости от требований системы. 3NF подходит для систем с четко определенной структурой и небольшим количеством изменений, тогда как Data Vault является оптимальным выбором для сложных систем с большими объемами данных и частыми изменениями.

В результате работы были получены практические навыки проектирования баз данных в различных методологиях, что позволит в дальнейшем выбирать наиболее подходящий подход для решения конкретных задач. Лабораторная работа также продемонстрировала важность правильного проектирования базы данных для обеспечения эффективности, надежности и масштабируемости системы учета видеопроката.