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

«Хранилища данных»

|  |  |  |
| --- | --- | --- |
| Студент | (личная подпись) | Е.С. Каленова |

Москва 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc186452763)

[1 3NF 4](#_Toc186452764)

[2 Data Vault 7](#_Toc186452765)

[ЗАКЛЮЧЕНИЕ 10](#_Toc186452766)

[ПРИЛОЖЕНИЕ 1 11](#_Toc186452767)

[ПРИЛОЖЕНИЕ 2 12](#_Toc186452768)

# ВВЕДЕНИЕ

В условиях стремительного развития информационных технологий и увеличения объемов данных, организации сталкиваются с необходимостью эффективного управления и анализа информации. Одним из подходов, который зарекомендовал себя как мощный инструмент для построения систем хранения данных, является методология Data Vault. Эта методология ориентирована на создание гибких, масштабируемых и устойчивых к изменениям хранилищ данных, что особенно актуально в эпоху больших данных и быстроменяющихся бизнес-требований.

Data Vault представляет собой архитектурный подход, который позволяет интегрировать данные из различных источников, обеспечивая их историзацию и возможность последующего анализа. Основные компоненты структуры Data Vault — это хабы, ссылки и факты, которые позволяют организовать данные таким образом, чтобы обеспечить их доступность и целостность. Такой подход не только упрощает процесс загрузки данных, но и значительно ускоряет их обработку и анализ.

Цель данной работы — изучить принципы реализации структуры Data Vault, рассмотреть ее ключевые элементы и преимущества, а также проанализировать примеры успешного применения этой методологии в различных отраслях. В ходе исследования будет проведен сравнительный анализ Data Vault с другими подходами к построению хранилищ данных, что позволит выявить сильные и слабые стороны данной методологии.

Таким образом, данная работа направлена на углубленное понимание структуры Data Vault и ее роли в современных системах управления данными, что позволит организациям более эффективно использовать свои ресурсы для достижения бизнес-целей.

# 3NF

Нормализация данных — это процесс организации данных в реляционной базе данных для уменьшения избыточности и повышения целостности данных. Существует несколько нормальных форм, каждая из которых имеет свои правила и требования. Третья нормальная форма (3NF) является одной из наиболее важных нормальных форм, обеспечивающих структурированность и эффективность базы данных.

Преимущества третьей нормальной формы:

1. Снижение избыточности данных: Устранение транзитивных зависимостей позволяет избежать дублирования данных.

2. Улучшение целостности данных: Поскольку данные организованы более логично, вероятность возникновения ошибок при обновлении данных снижается.

3. Облегчение управления изменениями: При изменении структуры данных или требований к базе данных легче вносить изменения, когда данные находятся в нормализованном виде.

Недостатки третьей нормальной формы

1. Сложность запросов: Нормализованные таблицы могут требовать более сложных SQL-запросов для получения данных из нескольких связанных таблиц.

2. Производительность: В некоторых случаях избыточная нормализация может привести к ухудшению производительности при выполнении запросов, особенно если требуется объединение большого числа таблиц.

Третья нормальная форма (3NF) является важным этапом в процессе нормализации баз данных. Она помогает организовать данные таким образом, чтобы минимизировать избыточность и повысить целостность данных. Несмотря на некоторые недостатки, преимущества, которые предоставляет 3NF, делают ее популярным выбором при проектировании реляционных баз данных.

Вариант 2 - Система должна описывать порядок выполнения практической работы по дисциплине «СУБД».

Система будет включать следующие сущности:

1. **Студенты (Students)**:
   * student\_id (PK) — уникальный идентификатор студента.
   * name — имя студента.
   * email — электронная почта студента.
2. **Практические работы (PracticalWorks)**:
   * work\_id (PK) — уникальный идентификатор работы.
   * title — название работы.
   * description — описание работы.
3. **Этапы выполнения (Stages)**:
   * stage\_id (PK) — уникальный идентификатор этапа.
   * work\_id (FK) — ссылка на практическую работу.
   * stage\_number — номер этапа.
   * stage\_description — описание этапа.
4. **Выполнение этапов (StageCompletion)**:
   * completion\_id (PK) — уникальный идентификатор выполнения этапа.
   * student\_id (FK) — ссылка на студента.
   * stage\_id (FK) — ссылка на этап.
   * completion\_date — дата выполнения этапа.
   * status — статус выполнения (например, "выполнено", "не выполнено").

Результат представлен на рисунке 1.

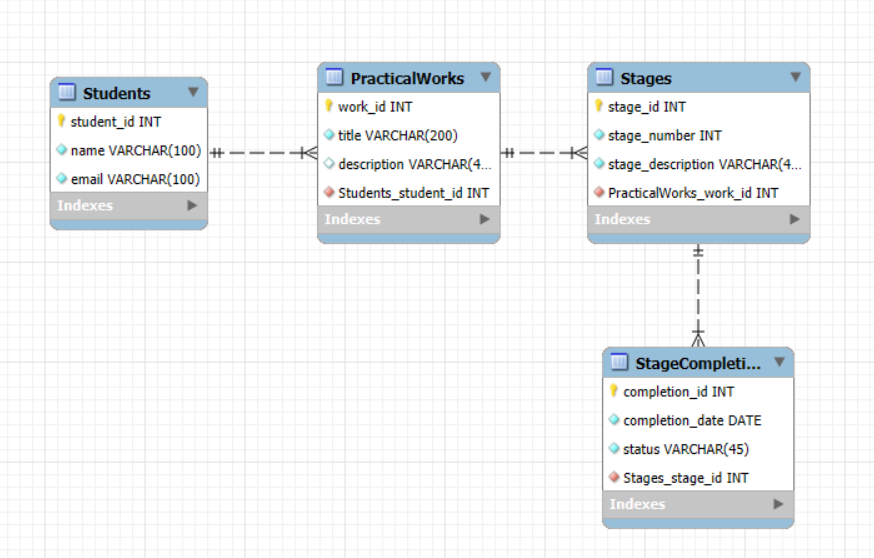


Рисунок 1 – БД

SQL код создания базы данных представлен в Приложении 1.

# Data Vault

Data Vault — это методология моделирования данных, разработанная для хранения и управления данными в хранилищах данных (data warehouses). Она была создана для решения проблем, связанных с традиционными подходами к моделированию данных, такими как звёздная и снежинка, особенно в условиях быстро меняющихся бизнес-требований и необходимости интеграции данных из различных источников.

Data Vault предлагает гибкую, масштабируемую и адаптивную архитектуру, которая позволяет эффективно управлять историей данных и обеспечивать высокую степень качества данных. Основная идея заключается в том, чтобы отделить бизнес-смысл от технической реализации данных.

Модель Data Vault состоит из трех основных типов объектов:

1. Хабы (Hubs):

• Хабы представляют собой уникальные бизнес-сущности, такие как клиенты, продукты или транзакции.

• Каждый хаб содержит уникальный идентификатор (например, ключ), атрибуты, которые описывают сущность, и метаданные о времени загрузки данных.

• Хабы не содержат исторических данных; они служат основой для других объектов.

2. Связи (Links):

• Связи используются для моделирования отношений между хабами.

• Они позволяют отслеживать связи между различными бизнес-сущностями и могут содержать атрибуты, описывающие эту связь.

• Связи также могут иметь временные метки для отслеживания изменений во времени.

3. Сателлиты (Satellites):

• Сателлиты хранят атрибуты, связанные с хабами и связями, включая исторические данные.

• Они позволяют сохранять изменения атрибутов во времени и могут содержать дополнительные метаданные, такие как информация о загрузке и источнике данных.

• Сателлиты обеспечивают гибкость в добавлении новых атрибутов без необходимости изменения структуры хабов и связей.

Data Vault представляет собой мощный подход к моделированию данных для хранилищ данных, обеспечивая гибкость, масштабируемость и возможность управления историческими данными. Его структура, состоящая из хабов, связей и сателлитов, позволяет организациям эффективно интегрировать данные из различных источников и адаптироваться к изменениям в бизнес-требованиях. Несмотря на некоторые недостатки, преимущества Data Vault делают его привлекательным выбором для многих организаций, стремящихся к более эффективному управлению данными.

Data Vault модель будет включать:

1. **Hubs**:
   * **Hub\_Students**:
     + student\_id (PK) — уникальный идентификатор студента.
   * **Hub\_PracticalWorks**:
     + work\_id (PK) — уникальный идентификатор работы.
   * **Hub\_Stages**:
     + stage\_id (PK) — уникальный идентификатор этапа.
2. **Links**:
   * **Link\_StageCompletion**:
     + completion\_id (PK) — уникальный идентификатор выполнения этапа.
     + student\_id (FK) — ссылка на студента.
     + stage\_id (FK) — ссылка на этап.
3. **Satellites**:

* **Sat\_Students**:
  + student\_id (FK) — ссылка на студента.
  + name — имя студента.
  + email — электронная почта студента.
* **Sat\_PracticalWorks**:
  + work\_id (FK) — ссылка на работу.
  + title — название работы.
  + description — описание работы
* **Sat\_Stages:**
  + stage\_id (FK) — ссылка на этап.
  + work\_id (FK) — ссылка на работу.
  + stage\_number — номер этапа.
  + stage\_description — описание этапа.
* **Sat\_StageCompletion**:
  + completion\_id (FK) — ссылка на выполнение этапа.
  + completion\_date — дата выполнения этапа.
  + status — статус выполнения.

SQL код создания базы данных представлен в Приложении 2.

# ЗАКЛЮЧЕНИЕ

3NF модель подходит для классической реляционной базы данных с нормализованной структурой.

Data Vault модель обеспечивает гибкость и масштабируемость, что полезно для систем с большим объемом данных и изменяющимися требованиями.

# ПРИЛОЖЕНИЕ 1

-- Создание таблицы Студенты

CREATE TABLE Students (

student\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL

);

-- Создание таблицы Практические работы

CREATE TABLE PracticalWorks (

work\_id SERIAL PRIMARY KEY,

title VARCHAR(200) NOT NULL,

description TEXT

);

-- Создание таблицы Этапы выполнения

CREATE TABLE Stages (

stage\_id SERIAL PRIMARY KEY,

work\_id INT REFERENCES PracticalWorks(work\_id) ON DELETE CASCADE,

stage\_number INT NOT NULL,

stage\_description TEXT NOT NULL,

UNIQUE(work\_id, stage\_number)

);

-- Создание таблицы Выполнение этапов

CREATE TABLE StageCompletion (

completion\_id SERIAL PRIMARY KEY,

student\_id INT REFERENCES Students(student\_id) ON DELETE CASCADE,

stage\_id INT REFERENCES Stages(stage\_id) ON DELETE CASCADE,

completion\_date DATE NOT NULL,

status VARCHAR(50) NOT NULL,

UNIQUE(student\_id, stage\_id)

);

# ПРИЛОЖЕНИЕ 2

-- Создание Hub таблиц

CREATE TABLE Hub\_Students (

student\_id SERIAL PRIMARY KEY

);

CREATE TABLE Hub\_PracticalWorks (

work\_id SERIAL PRIMARY KEY

);

CREATE TABLE Hub\_Stages (

stage\_id SERIAL PRIMARY KEY

);

-- Создание Link таблиц

CREATE TABLE Link\_StageCompletion (

completion\_id SERIAL PRIMARY KEY,

student\_id INT REFERENCES Hub\_Students(student\_id),

stage\_id INT REFERENCES Hub\_Stages(stage\_id),

UNIQUE(student\_id, stage\_id)

);

-- Создание Satellite таблиц

CREATE TABLE Sat\_Students (

student\_id INT REFERENCES Hub\_Students(student\_id),

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

PRIMARY KEY (student\_id)

);

CREATE TABLE Sat\_PracticalWorks (

work\_id INT REFERENCES Hub\_PracticalWorks(work\_id),

title VARCHAR(200) NOT NULL,

description TEXT,

PRIMARY KEY (work\_id)

);

CREATE TABLE Sat\_Stages (

stage\_id INT REFERENCES Hub\_Stages(stage\_id),

work\_id INT REFERENCES Hub\_PracticalWorks(work\_id),

stage\_number INT NOT NULL,

stage\_description TEXT NOT NULL,

PRIMARY KEY (stage\_id)

);

CREATE TABLE Sat\_StageCompletion (

completion\_id INT REFERENCES Link\_StageCompletion(completion\_id),

completion\_date DATE NOT NULL,

status VARCHAR(50) NOT NULL,

PRIMARY KEY (completion\_id)

);