3.2: Анализ БД на избыточность

Введение

Избыточность в базах данных возникает, когда данные повторяются без необходимости, что может привести к проблемам с целостностью данных, увеличению занимаемого пространства и сложности в поддержке и обновлении данных. Основная цель анализа БД на избыточность - выявить логические недостатки проектирования и сформулировать рекомендации для их устранения.

Примеры неверного логического проектирования

1. Повторяющиеся данные:

Рассмотрим таблицу, где информация о клиентах и их заказах хранится в одной таблице:

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_name VARCHAR(50),

customer\_address VARCHAR(100),

product\_name VARCHAR(50)

);

В этом случае, если один и тот же клиент делает несколько заказов, его имя и адрес будут дублироваться, что приводит к избыточности.

2. Отсутствие нормализации:

Еще один пример — таблица, в которой хранятся сведения о студентах и курсах:

CREATE TABLE StudentCourses (

student\_id INT,

student\_name VARCHAR(50),

course\_name VARCHAR(50),

course\_id INT,

PRIMARY KEY (student\_id, course\_id)

);

Здесь имя студента повторяется для каждого курса, что является ненужной избыточностью.

Алгоритм устранения недостатков логической структуры

Чтобы устранить недостатки и снизить избыточность в БД, можно использовать следующий алгоритм:

1. Анализ текущей структуры:

- Провести ревизию текущих таблиц и их связей.

- Выявить повторяющиеся данные и структуры, которые можно оптимизировать.

2. Применение нормализации:

- Применить правила нормализации (1NF, 2NF, 3NF) для упорядочивания данных:

- 1NF: Убедитесь, что все атрибуты атомарные (то есть не содержат множественных значений).

- 2NF: Устраните частичную зависимость, отделив атрибуты, которые зависят только от части первичного ключа.

- 3NF: Уберите транзитивные зависимости, чтобы атрибуты не зависели друг от друга.

3. Создание новых таблиц:

- Выделить избыточные данные в отдельные таблицы. Например, в случае со студентами можно создать отдельные таблицы для студентов и курсов.

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(50)

);

CREATE TABLE Courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(50)

);

CREATE TABLE StudentCourses (

student\_id INT,

course\_id INT,

PRIMARY KEY (student\_id, course\_id),

FOREIGN KEY (student\_id) REFERENCES Students(student\_id),

FOREIGN KEY (course\_id) REFERENCES Courses(course\_id)

);

4. Назначение первичных и внешних ключей:

- Установить связи между таблицами с помощью первичных и внешних ключей, чтобы гарантировать целостность данных и минимизировать избыточность.

5. Тестирование новой модели:

- Провести тестирование новой структуры, чтобы убедиться, что все операции (вставка, обновление, удаление) функционируют корректно.

6. Мониторинг и доработка:

- После внедрения изменений необходимо мониторить производительность и целостность данных, вносить коррективы по мере необходимости.

Заключение

Анализ базы данных на избыточность и устранение недостатков в логической структуре является важным этапом проектирования и эксплуатации БД. Нормализация — это мощный инструмент, который помогает избавиться от дублирующихся данных, повышает эффективность работы с данными и уменьшает вероятность возникновения ошибок. Следуя представленному алгоритму, можно существенно улучшить качество и производительность базы данных.