Шелестян Анастасия IA2304

Вариант 1

1. Опишите, что такое реляционная база данных, как мы можем ее использовать, каковы основные преимущества и недостатки.

**Реляционные базы данных** это системы управления базами данных, которые хранят данные в структурированном виде, организованном в таблицы с четко определенными схемами и поддерживают запросы SQL.

Примеры: MySQL, PostgreSQL и база данных Oracle.

**Как мы можем использовать реляционную базу данных:**

Хранение, управление, обработка данных для последующего аналза.

**Основные преимущества:**

Целостность данных, SQL (Универсальный язык запросов для работы с данными)

Гибкость, то есть легко добавлять новые таблицы и связи по мере роста системы.

Безопасность, можно назначать права доступа разным пользователям.

Масштабируемость, система легко поддерживает сложные запросы и отчеты с большими объемами данных.

**Основные недостатки:**

Проблемы с масштабированием.

Производительность, сложные запросы могут замедлять работу системы.

Сложность, проектирование и поддержка базы требует значительных усилий.

Зависимость от схемы, изменение структуры базы данных (например, добавление новых полей) требует тщательного планирования и может быть затруднено.

Ресурсоемкость, для поддержания больших реляционных баз данных требуются мощные серверы и системы хранения.

1. Что такое нормализация, зачем нам нормализация. Опишите вторую форму нормализации, как мы можем ее решить, зачем нам нужна эта форма.

**Нормализация** — это процесс проектирования реляционной базы данных, который помогает организовать данные таким образом, чтобы уменьшить избыточность и зависимость, а также обеспечить целостность данных. Цель нормализации — эффективно структурировать схему базы данных, упрощая ее обслуживание, минимизировать аномалии данных и улучшить поиск данных.

2NF • Таблица находится во 2NF, если она уже находится в 1NF и если все неключевые столбцы полностью функционально зависят от всего первичного ключа (нет частичных зависимостей). • В 2NF каждый неключевой столбец должен зависеть от всего первичного ключа, а не только от его части. • 2NF особенно актуален в таблицах с составными (многостолбцовыми) первичными ключами.

1. Что такое команда ALTER, опишите ее и приведите три примера ее использования.

ALTER это команда для изменения структуры таблицы в SQL. Команда ALTER используется для внесения изменений в структуру существующей таблицы в базе данных. С ее помощью можно добавлять, изменять или удалять столбцы, а также накладывать ограничения на таблицу (например, первичные или внешние ключи).

ALTER TABLE, чтобы добавить новый столбец в существующую таблицу. Также можно изменить тип данных существующего столбца, переименовать существующую таблицу. Можете добавлять или удалять индексы для столбцов, изменить владельца таблицы.

1. Опишите 3 ограничения, а также что такое SetNull, когда мы можем его использовать и почему нам нужно это использовать.

Основных ограничения в реляционных БД: **PRIMARY KEY(**гарантирует уникальность каждой записи в таблице).**FOREIGN KEY** (Создает связь между таблицами).**UNIQUE (**Гарантирует, что все значения в столбце (или группе столбцов) уникальны, но допускает NULL).

SetNull это действие, которое выполняется при удалении или обновлении записи в **родительской таблице**. При использовании все связанные записи в **дочерней таблице** получают значение null в поле внешнего ключа.

Нужно использовать для сохранения данных, их чистоты и также гибкости.

1. Опишите команду «Insert», зачем она нам нужна, как мы можем ее использовать, в чем разница между «Insert» и «Create».

Команда INSERT используется для добавления новых записей (строк) в уже существующую таблицу базы данных. Она позволяет заполнить таблицу конкретными данными.

Разница между «Insert» и «Create». Первая добавляет данные строки в таблицу. Работает с существующими таблицами. Вторая с создает новую таблицу, без данных или объектов, создает структуру для хранения данных.

1. Из sql-скрипта опишите все связи из таблиц и перечислите все ограничения из них.

CREATE TABLE department (

department\_id SERIAL PRIMARY KEY,

department\_name VARCHAR(50) NOT null

);

CREATE TABLE employees (

employee\_id SERIAL PRIMARY KEY,

employee\_name VARCHAR(50) NOT null,

age int,

salary int

);

CREATE TABLE depemployees (

department\_id INT REFERENCES department(department\_id),

employee\_id INT REFERENCES employees(employee\_id),

PRIMARY KEY (department\_id, employee\_id)

);

Связи между таблицами

Связь между depemployees и department: Поле department\_id в таблице depemployees ссылается на поле department\_id в таблице department. Это создаёт связь **«один ко многим»:** один департамент может иметь много сотрудников.

Связь между depemployees и employees: Поле employee\_id в таблице depemployees ссылается на поле employee\_id в таблице employees. Это также создаёт связь **«один ко многим»:** один сотрудник может принадлежать к нескольким департаментам.

**Ограничения**

Ограничения первичного ключа (PRIMARY KEY) гарантирует уникальное значение для полей в таблицах.

Ограничения внешнего ключа (FOREIGN KEY) является внешним ключом, что обеспечивает ссылочную целостность между таблицами.

1. В таблице department поле department\_name имеет ограничение NOT NULL, что требует, чтобы это поле всегда содержало значение (не могло быть пустым).
2. Напишите скрипт select, чтобы вернуть всех сотрудников старше 30 лет.

SELECT employee\_id, employee\_name, age, salary

FROM employees

WHERE age > 30;

1. напишите скрипт select, чтобы вернуть средний возраст сотрудников

SELECT AVG(age)

FROM employees;

1. напишите скрипт select, чтобы заменить все имя\_сотрудника, у которого есть Смит, на Джексона.

UPDATE employees

SET employee\_name = 'Джексон'

WHERE employee\_name LIKE '%Смит%';

1. Напишите скрипт select с оператором Case для категоризации зарплаты: если зарплата>50000, то она высокая, если зарплата>30000, то средняя, иначе низкая.

SELECT salary,

CASE

WHEN salary > 50000 THEN 'Высокая'

WHEN salary > 30000 THEN 'Средняя'

ELSE 'Низкая'

END AS salary\_category

FROM employees;

1. Напишите скрипт для обновления возраста и зарплаты сотрудника, у которого в имени сотрудника есть Джонсон

UPDATE employees

SET age = 35,

salary = 60000

WHERE employee\_name LIKE '%Джонсон%';