1. Общие понятия

SQL — декларативный язык программирования (декларативные языки фокусируются на том, что должно быть сделано, а не как это делать. В таких языках мы описываем желаемый результат, а детали его достижения остаются скрытыми), применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Короче, SQL - язык запросов, с помощью которого можно работать с данными в базе данных.
СУБД (Система управления базами данных) — это программа, которая позволяет создавать, хранить, управлять и обрабатывать данные в базе данных. СУБД получает команды (чаще всего на языке SQL) и выполняет их. Каждая СУБД (PostgreSQL, MySQL, SQL Server, Oracle, SQLite) имеет свои особенности и расширения SQL, которые отличают её от других. Эти различия называют диалектами SQL.
PostgreSQL — объектно-реляционная СУБД с открытым исходным кодом. Она поддерживает не только стандартные реляционные таблицы, но и более сложные структуры данных (JSON, массивы, XML), а также продвинутые возможности, такие как репликация, расширения и полнотекстовый поиск.

Про регистрочувствительность:
В PostgreSQL имена, не оформленные двойными кавычками, интерпретируются в нижнем регистре. Между тем, если имя обрамлено кавычками, наподобие "Name", оно становится регистрозависимым. Сравним два запроса:
SELECT * FROM "Users" и SELECT * FROM users
В первом запросе существенна заглавная буква U, в то время как второй не различает регистр букв. Запросы работают независимо от регистра, если нет кавычек.
Строковые данные (например, в TEXT или VARCHAR) сравниваются с учетом регистра.
Ключевые слова (SELECT, WHERE, INSERT, UPDATE и т. д.) НЕ чувствительны к регистру.
Специальный тип citext позволяет хранить текст без учета регистра при сравнении: name CITEXT UNIQUE.

2. Архитектура ANSI-SPARC

Трехуровневая архитектура ANSI/SPARC имеет внешний, концептуальный и внутренний уровни. Цель трехуровневой архитектуры заключается в отделении пользовательского представления базы данных от её физического представления, что обусловлено рядом причин:
Каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя свое представление о них, и изменять свое представление о данных, не оказывая при этом влияния на других пользователей;
Пользователи не должны знать особенности физического хранения данных в базе;
АБД должен иметь возможность изменять структуру хранения данных в базе, не оказывая влияния на пользовательские представления;
АБД должен иметь возможность изменять концептуальную или глобальную структуру базы данных без влияния на всех пользователей.
Внешний уровень – это представление базы данных с точки зрения пользователей. Этот уровень описывает ту часть базы данных, которая относится к каждому пользователю.
Концептуальный уровень – это обобщающее представление базы данных. Этот уровень описывает то, какие данные хранятся в базе данных, а также связи, существующие между ними. Фактически, это полное представление требований к данным со стороны организации, которое не зависит от способа их хранения. На концептуальном уровне представлены следующие компоненты:
все сущности, атрибуты и связи;
накладываемые на данные ограничения;
информация о мерах обеспечения безопасности и поддержки целостности данных.
Описание сущности на концептуальном уровне должно содержать сведения о типах данных атрибутов и их длине, но не должно включать сведений об организации хранения данных, например, об объёме занятого пространства в байтах.
Внутренний уровень – это физическое представление базы данных в компьютере. Этот уровень описывает, как информация хранится в базе данных. Внутренний уровень описывает физическую реализацию базы данных и предназначен для достижения оптимальной производительности и обеспечения экономичности использования дискового пространства. Он содержит описание структур данных и организации отдельных файлов, используемых для хранения данных в запоминающих устройствах.

3. Про модельки и бла бла бла

Сущность — класс объектов, фактов, явлений, предметов, элементы которых будут храниться в базе данных.
Атрибут — важная характеристика (свойство) сущности, которой присваивается имя.
Связь — ассоциирование двух или более сущностей, выражающее форму взаимодействия между ними.
Про связи будто бы очев: 1:1, 1:M, M:M (нужна доп сущность), связи бывают необязательные (X:Y, где X может быть нулём).

1:M реализуется путем добавления в таблицу внешнего ключа: внешний ключ обычно добавляется в сущность-ассоциацию или в сущность-характеристику. Для моделирования характера связи на внешний ключ вводятся доп. ограничения (например, «один-к-одному» - UNIQUE).
Стержневая сущность — независимая, базовая сущность (Студент, Группа);
Ассоциативная сущность — связь вида "многие-ко-многим" ("*-ко-многим"и т. д.) между двумя или более сущностями;
Характеристическая сущность — связь вида "многие-к-одной" или "одна-к-одной" между двумя сущностями (частный случай ассоциации). Цель характеристики — описание или уточнение некоторой другой сущности.
Первичный ключ означает, что колонка используется как уникальный идентификатор строки в таблице. Основные структуры данных, которые могут использоваться для реализации PK это хэш-таблицы (хэш-значения будут использоваться для поиска в памяти места, где хранится запись, но они не упорядочены) и B-деревья (самобалансирующие деревья, поиск за логарифм). Ограничение первичного ключа эквивалентно комбинации ограничений уникальности и не-null. PostgreSQL позволяет не создавать первичный ключ для таблицы (требуется наличие первичного ключа для каждой таблицы).
Каждая колонка имеет свой тип данных: ограничивает список возможных значений, которые могут находиться в этой колонке и определяет «смысл» данных, хранящихся в колонке, чтобы данные могли использоваться при вычислениях.

PostgreSQL:
smallint — целые числа (2 байта);
integer — целые числа (4 байта);
bigint — целые числа (8 байт);
varchar(n) — переменной длины с ограничением;
char(n) — фиксированной длины, остаток заполняется пробелами;
text — переменной длины без задаваемого ограничения;
состояния: «true», «false» (третье состояние, «unknown» представляется через SQL-значение NULL);
«true» может задаваться следующими значениями: TRUE, 'true', 't', 'yes', 'y', 'on', '1';
«false» может задаваться следующими значениями: FALSE, 'false', 'f', 'no', 'n', 'off', '0';
timestamp — дата и время;
date — дата;
interval — временной интервал;
NULL — специальное значение (пустое, несуществующее значение), которое может быть записано в поле таблицы базы данных. Значения типа NULL не равны друг другу; столбец, содержащий значение NULL, игнорируется при вычислениях агрегатных значений (мин макс и тд) и в предложениях с DISTINCT, ORDER BY, GROUP BY значения NULL не отличаются друг от друга.
Колонка не должна содержать значение null: type NOT NULL (PK автоматически проверяется на NULL).
SERIAL — это специальный тип данных в PostgreSQL, который автоматически создает автоинкрементное поле (счётчик) для идентификаторов.

Оператор CHECK позволяет задать для определённой колонки выражение, которое будет осуществлять проверку помещаемого в эту колонку значения. Выражение должно возвращать логическое значение. Проверка CHECK проходит, если выражение, указанное в ограничении, возвращает значение истина или null. Если условие CHECK не выполняется, PostgreSQL выдаст ошибку, и операция будет отклонена.

```
1 CREATE TABLE STUDENTS (
2     ST_ID integer,
3     ST_NAME text,
4     FAILED_COURSES integer CONSTRAINT fcrs CHECK (FAILED_COURSES >= 0)
5 );
```

CONSTRAINT — это ограничение (правило), которое накладывается на столбец или таблицу, чтобы обеспечить целостность данных. Если дать ограничению имя, его можно потом легко изменить или удалить:
ALTER TABLE STUDENTS DROP CONSTRAINT fcrs;
Если бы имя не было задано, PostgreSQL сам бы сгенерировал случайное имя для ограничения (ALTER изменяет структуру существующей сущности).

UNIQUE — это ограничение (constraint), которое гарантирует, что значения в указанном столбце или группе столбцов будут уникальными:
UNIQUE не допускает повторяющихся значений в одном столбце или комбинации столбцов;
В PostgreSQL несколько NULL считаются разными значениями.
Если UNIQUE значения повторяются, то PostgreSQL выдаст ошибку, и операция будет отклонена.

```
1 CREATE TABLE enrollments (
2     student_id integer,
3     course_id integer UNIQUE,
4     UNIQUE (student_id, course_id)
5 );
```

В PostgreSQL по умолчанию столбец не имеет DEFAULT-значения, если оно явно не задано. Это означает, что при вставке (INSERT) без указанного значения столбец примет NULL.
Можно задать значение по умолчанию (DEFAULT) для столбца, чтобы оно автоматически использовалось, если при INSERT в этот столбец не передано значение. DEFAULT также можно изменять и удалять через ALTER TABLE.

```
1 CREATE TABLE students (
2     id serial PRIMARY KEY,
3     name text NOT NULL,
4     age integer DEFAULT 18
5 );
```

Если при вставке в students не указать age, будет использовано значение 18. Если столбец NOT NULL, но DEFAULT не указан, и при INSERT в него не передано значение, то будет ошибка.

4. DDL

DDL (Data Definition Language) — это набор команд SQL, предназначенных для создания, изменения и удаления структур базы данных (таблиц, индексов, схем, ограничений и т. д.). DDL-команды автоматически фиксируются (автоматический COMMIT). Это значит, что откатить их через ROLLBACK нельзя, если они уже выполнены.
Транзакция — это атомарная последовательность SQL-операций. Она гарантирует целостность данных в случае ошибок.
Транзакции управляются командами:
BEGIN; — начало транзакции
COMMIT; — подтверждение изменений
ROLLBACK; — откат изменений

Основные команды:

CREATE - создание таблицы, индексов (для поиска), схемы (пространство имён для таблицы).

ALTER – изменение объектов:
ALTER TABLE users ADD COLUMN email TEXT UNIQUE - добавляет столбец email, который должен быть уникальным;
ALTER TABLE users ALTER COLUMN age TYPE BIGINT - меняет тип age (атрибут уже был какого-то типа) на BIGINT;
ALTER TABLE users DROP COLUMN age - удаляет age из таблицы;
ALTER TABLE users RENAME TO customers - теперь таблица users называется customers (аналогично для переименования столбца);
ALTER TABLE users ALTER COLUMN email SET NOT NULL - теперь email обязателен для заполнения;
ALTER TABLE users ADD PRIMARY KEY (id).

DROP – удаление объектов:
DROP TABLE users - полностью удаляет таблицу users;
DROP SCHEMA sales CASCADE - ну логически подумать;
Удалить каскадно, то есть удалить с зависимостями (иначе рантайм).

TRUNCATE – очистка данных:
TRUNCATE TABLE users - структура останется.

5. DML

DML (Data Manipulation Language) — это подмножество SQL-команд, которое используется для манипулирования данными в базе данных. С помощью DML можно вставлять новые данные, обновлять или удалять существующие данные, а также извлекать их.
Основные команды:

SELECT — извлечение данных:
SELECT * FROM Employees - вернёт все строки и все столбцы из таблицы Employees;
SELECT first_name, last_name, position FROM Employees;
SELECT first_name, last_name, age FROM Employees WHERE age $>$ 30;
SELECT first_name, last_name, age FROM Employees ORDER BY age - вернёт список сотрудников, отсортированных

по возрасту в порядке возрастания (если в конце добавить DESC, то вернёт инвертированный набор).

INSERT — вставка данных:
INSERT INTO Employees (first_name, last_name, age, position)
VALUES
('John', 'Doe', 35, 'Manager'),
('Jane', 'Smith', 42, 'Engineer'),
('Mark', 'Taylor', 29, 'Developer');
INSERT INTO Employees (first_name, last_name, age, position)
SELECT first_name, last_name, age, position
FROM TemporaryEmployees
WHERE age > 30;
Этот запрос вставит в таблицу Employees данные, отобранные из таблицы TemporaryEmployees с условием, что возраст сотрудника больше 30 лет.

UPDATE — обновление данных:
UPDATE Employees
SET position = 'Senior' || position
WHERE age > 40;
Этот запрос добавит префикс "Senior" ко всем должностям сотрудников старше 40 лет. (SET используется для назначения нового значения столбцу в процессе обновления строки; || - конкатенация строк).
UPDATE Employees
SET position = 'Intern';
Этот запрос обновит должность всех сотрудников на "Intern".

DELETE — удаление данных:
DELETE FROM Employees
WHERE first_name = 'Alice' AND last_name = 'Johnson';
Запрос удалит строку, где имя сотрудника Alice Johnson.
DELETE FROM Employees;
Удаление всех данных таблицы, но структура останется (аналог TRUNCATE).