PostgreSQL Database Design Project

In this project, you will design and implement a PostgreSQL database system from scratch. You'll go through the entire database design process, from conceptual modeling to physical implementation.

Topic Suggestions

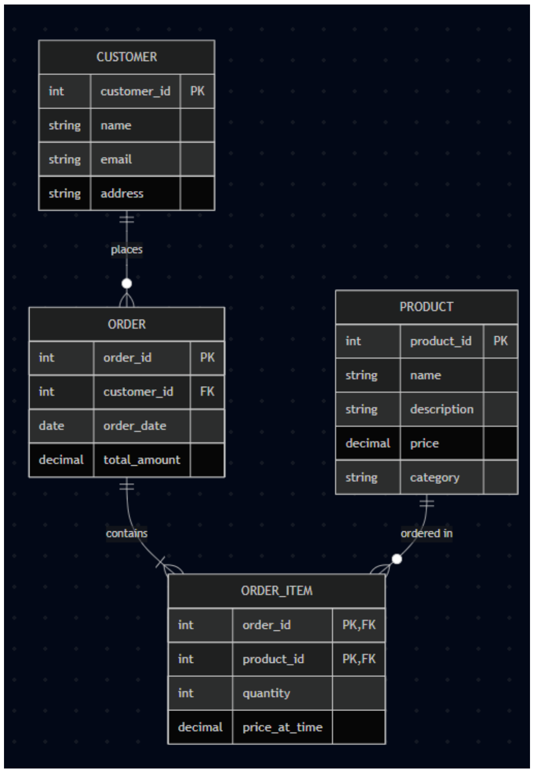
- Library Management System
- E-commerce Platform
- Hospital Information System
- University Registration System
- Inventory Management
- Social Media Platform
- Hotel Reservation System
- Movie Database
- Recipe/Cooking Database
- Sports League Management

Step 1: Conceptual Design

Begin by identifying the real-world entities your database will track. Think about:

- What are the main objects/entities in your system?
- What attributes describe each entity?
- How do these entities relate to each other?
- What business rules or constraints apply?

Create an ERD to visualize these concepts.



Step 2: Logical Design

Transform your conceptual model into a logical database schema:

- Map entities to tables
- Map attributes to columns
- Define primary keys
- Establish foreign key relationships
- Apply normalization principles (aim for 3NF)

Step 3: Physical Design

Consider implementation details:

- Choose appropriate data types for each column
- Define constraints (NOT NULL, UNIQUE, CHECK)
- Plan indexes for performance
- Design views for common queries

Step 4: Implementation

Write SQL scripts to CREATE your database objects (tables, views, indexes).

- CREATE table
- CREATE index
- CREATE view AS

Step 5: User Access Control

Define roles (CREATE ROLE) and permissions (GRANT):

PostgreSQL Cheat Sheet for Database Design

Data Types

Category	Data Type	Description	Example
Numeric	INTEGER	Whole number	age INTEGER

	BIGINT	Large whole number	population BIGINT
	SERIAL	Auto-incrementing integer	id SERIAL PRIMARY KEY
	DECIMAL(p,s)	Exact numeric with precision	price DECIMAL(10,2)
	REAL	Single precision floating-point	temperature REAL
	DOUBLE PRECISION	Double precision floating-point	scientific_measurement DOUBLE PRECISION
String	CHAR(n)	Fixed-length character string	state_code CHAR(2)
	VARCHAR(n)	Variable-length character string	username VARCHAR(50)
	TEXT	Unlimited length text	description TEXT
Binary	BYTEA	Binary data ("byte array")	file_data BYTEA
Date/Time	DATE	Calendar date (year, month, day)	birth_date DATE
	TIME	Time of day	appointment_time TIME
	TIMESTAMP	Date and time	created_at TIMESTAMP
	INTERVAL	Time period	duration INTERVAL
Boolean	BOOLEAN	True/false value	is_active BOOLEAN
Special	UUID	Universally unique identifier	id UUID DEFAULT gen_random_uuid()
	JSON	JSON data	preferences JSON
	JSONB	Binary JSON data (faster)	document JSONB
	ARRAY	Array of values	tags TEXT[]
	INET	IPv4 or IPv6 network address	ip_address INET

SQL Syntax Tables

Operation	Syntax	Example
Create Table	CREATE TABLE table_name (column_definitions);	CREATE TABLE users (user_id SERIAL PRIMARY KEY, username VARCHAR(50));
Drop Table	DROP TABLE table_name;	DROP TABLE temp_data;
Rename Table	ALTER TABLE table_name RENAME TO new_name;	ALTER TABLE users RENAME TO customers;

Constraints

Constraint	**Syntax**	**Example**
Primary Key	CONSTRAINT pk_name PRIMARY KEY (column)	ALTER TABLE users ADD CONSTRAINT pk_users PRIMARY KEY (user_id);
Foreign Key	CONSTRAINT fk_name FOREIGN KEY (column) REFERENCES table(column)	ALTER TABLE orders ADD CONSTRAINT fk_user FOREIGN KEY (user_id) REFERENCES users(user_id);
Unique	CONSTRAINT constraint_name UNIQUE (column)	ALTER TABLE users ADD CONSTRAINT unique_email UNIQUE (email);
Check	CONSTRAINT constraint_name CHECK (condition)	ALTER TABLE products ADD CONSTRAINT positive_price CHECK (price > 0);
Not Null	ALTER TABLE table_name ALTER COLUMN column_name SET NOT NULL;	ALTER TABLE users ALTER COLUMN username SET NOT NULL;
Default Value	ALTER TABLE table_name ALTER COLUMN column_name SET DEFAULT value;	ALTER TABLE users ALTER COLUMN created_at SET DEFAULT CURRENT_TIMESTAMP;

Indexes

Index Type	**Syntax**	**Example**
Basic Index	CREATE INDEX index_name ON table_name(column);	CREATE INDEX idx_users_email ON users(email);
Unique Index	CREATE UNIQUE INDEX index_name ON table_name(column);	CREATE UNIQUE INDEX idx_unique_email ON users(email);
Multi-Column Index	CREATE INDEX index_name ON table_name(column1, column2);	CREATE INDEX idx_orders_user_date ON orders(user_id, order_date);

Queries

Query Type	**Syntax**	**Example**
Basic Select	SELECT columns FROM table WHERE condition;	SELECT * FROM users WHERE is_active = true;
Join	SELECT columns FROM table1 JOIN table2 ON condition;	SELECT u.username, o.order_date FROM users u JOIN orders o ON u.user_id = o.user_id;
Aggregation	SELECT agg_function(column) FROM table GROUP BY column;	SELECT COUNT(*), SUM(amount) FROM orders GROUP BY user_id;
Subquery	SELECT columns FROM table WHERE column IN (SELECT...);	SELECT username FROM users WHERE user_id IN (SELECT user_id FROM orders);

Views

Operation	**Syntax**	**Example**
Create View	CREATE VIEW view_name AS SELECT...;	CREATE VIEW active_users AS SELECT * FROM users WHERE is_active = true;
Create Materialized View	CREATE MATERIALIZED VIEW view_name AS SELECT...;	CREATE MATERIALIZED VIEW order_summary AS SELECT COUNT(*) FROM orders;
Refresh Materialized View	REFRESH MATERIALIZED VIEW view_name;	REFRESH MATERIALIZED VIEW order_summary;

User Permissions

Operation	**Syntax**	**Example**
Create Role	CREATE ROLE role_name;	CREATE ROLE app_read;
Create User	CREATE USER username WITH PASSWORD 'password';	CREATE USER john WITH PASSWORD 'secure_password';
Grant Role	GRANT role TO user;	GRANT app_read TO john;
Grant Table Permission	GRANT privileges ON table TO role;	GRANT SELECT ON users TO app_read;
Grant Schema Permission	GRANT privileges ON SCHEMA schema TO role;	GRANT USAGE ON SCHEMA public TO app_read;
Grant All Tables	GRANT privileges ON ALL TABLES IN SCHEMA schema TO role;	GRANT SELECT ON ALL TABLES IN SCHEMA public TO app_read;