

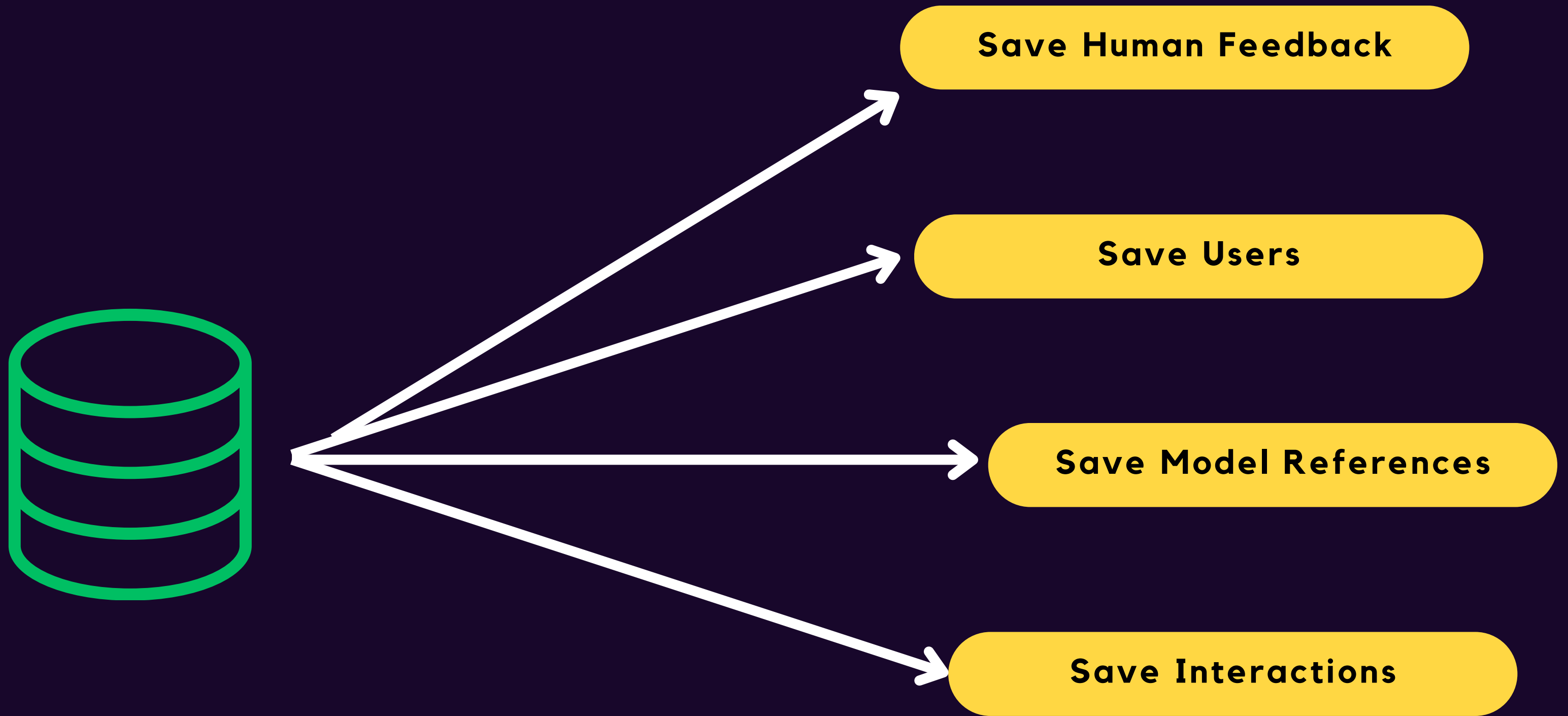
INTRODUCTION TO STORAGE

Apps without Storage are like humans without memory

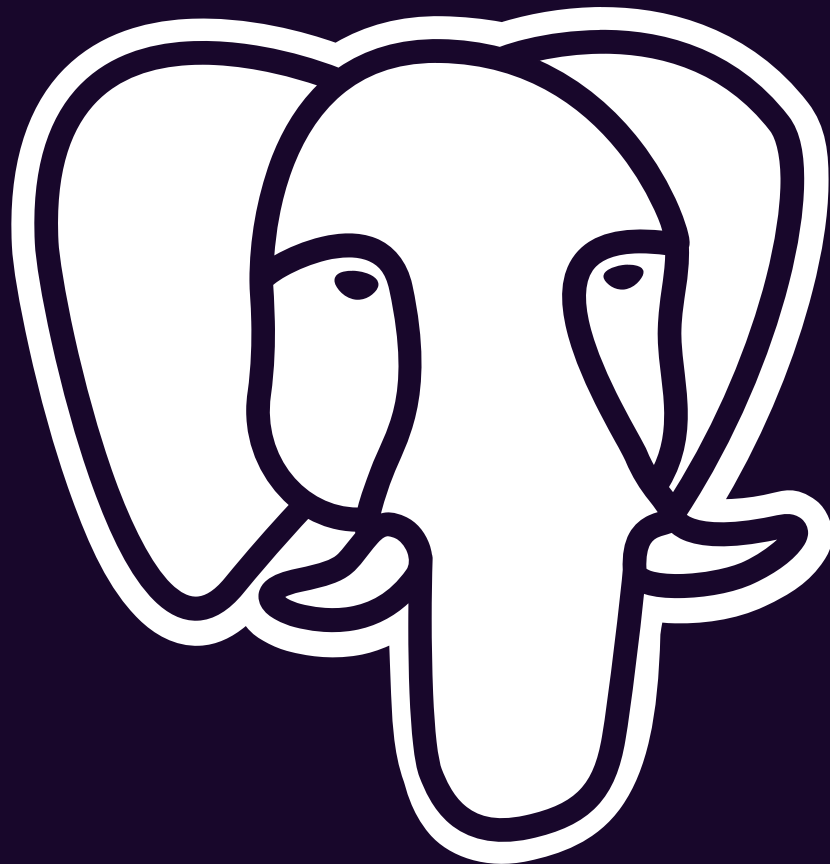
Imagine interacting with an application where you've to register every time you use it.

Or an application where you can't go through your previous interactions with the System.

Adding storage component to the App



Introduction to the Postgres Database



- **Open-Source Relational Database Management System (RDBMS)**
- **Advanced SQL Support (e.g. queries, transactions, user defined functions etc)**
- **Wide range of data types including e.g. JSON**
- **Supported by major cloud platforms**
- **High Performance, scalability and battle tested in Production environments**

Download Postgres for your OS

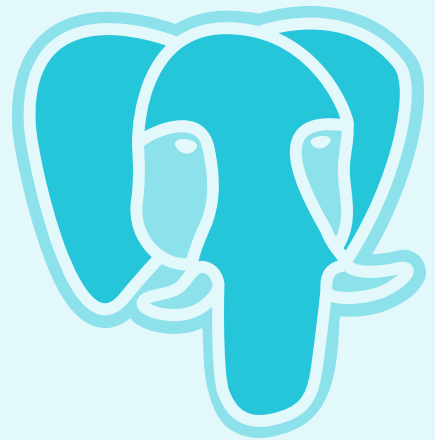


<https://www.postgresql.org/download/>

Postgres Docker Containers are available

- The benefit of using PostGres as a container is that you end up saving a lot on debugging time due to missing libraries on the OS

How to manage Postgres ?



<https://www.pgadmin.org/>

Important DB Concepts



- **Database is a collection of related data organized and stored in a structured format.**
- **Container for tables, indexes, functions, and other database objects.**
- **Each database in PostgreSQL operates independently**

Important DB Concepts: Tables



- Data in PostgreSQL is stored in tables, which consist of rows and columns.
- Each column has a specific data type (e.g., integer, text, timestamp).

Important DB Concepts: Primary Key



- A primary key uniquely identifies each row in a table.
- Primary keys are typically implemented using the **PRIMARY KEY** constraint.

Important DB Concepts: Foreign Key



- A foreign key establishes a relationship between two tables.
- Foreign keys are defined using the **FOREIGN KEY** constraint.

Important DB Concepts: Indexes



- **Indexes are data structures that improve the speed of data retrieval operations, such as querying and sorting.**
- **PostgreSQL supports various types of indexes, including B-tree.**

Important DB Concepts: Transactions



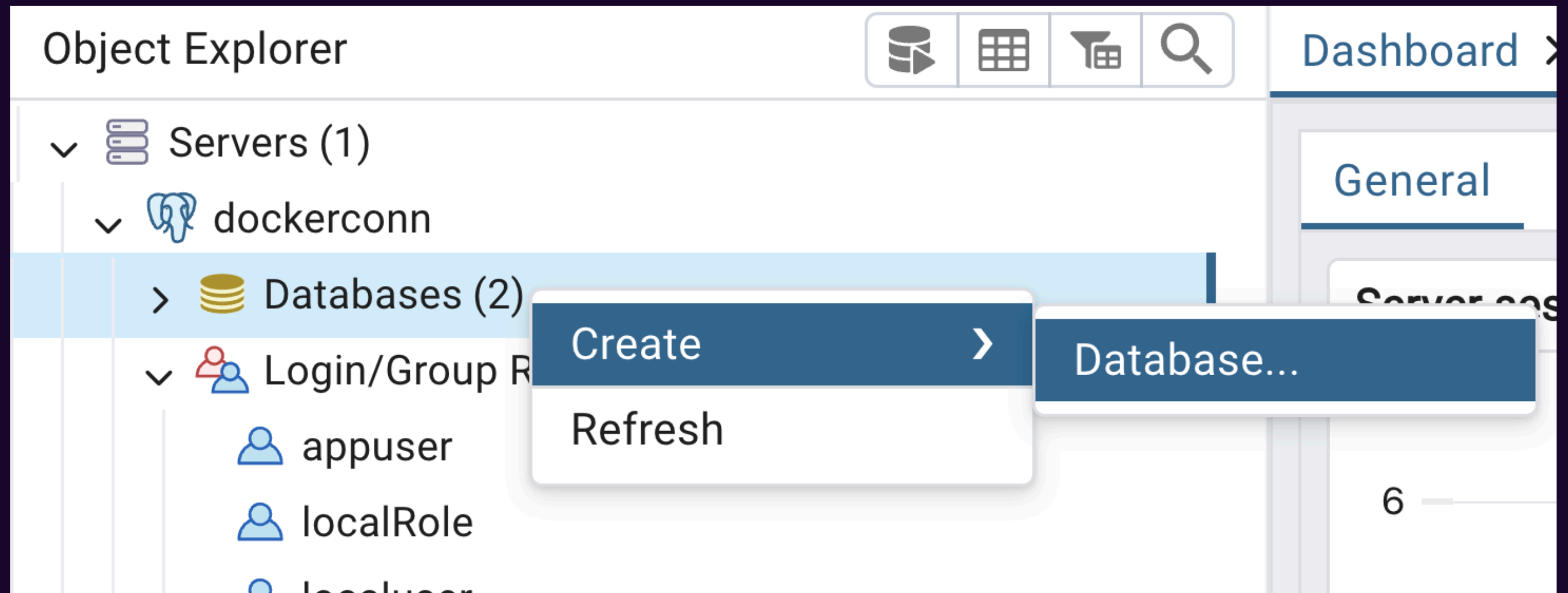
- Transactions ensure data integrity by grouping multiple database operations into a single unit of work.
- PostgreSQL supports ACID (Atomicity, Consistency, Isolation, Durability) properties for transactions.

Important DB Concepts: Schema




- **Schema: A schema within a database defines the structure of tables and other database objects.**
- **It provides a namespace for grouping related objects and organizing data logically.**

Storage in Action: Create Database



Provide a Name

 Create - Database ✕

General

Definition

Security

Parameters

Advanced


SQL


Database

OID

Owner

Comment

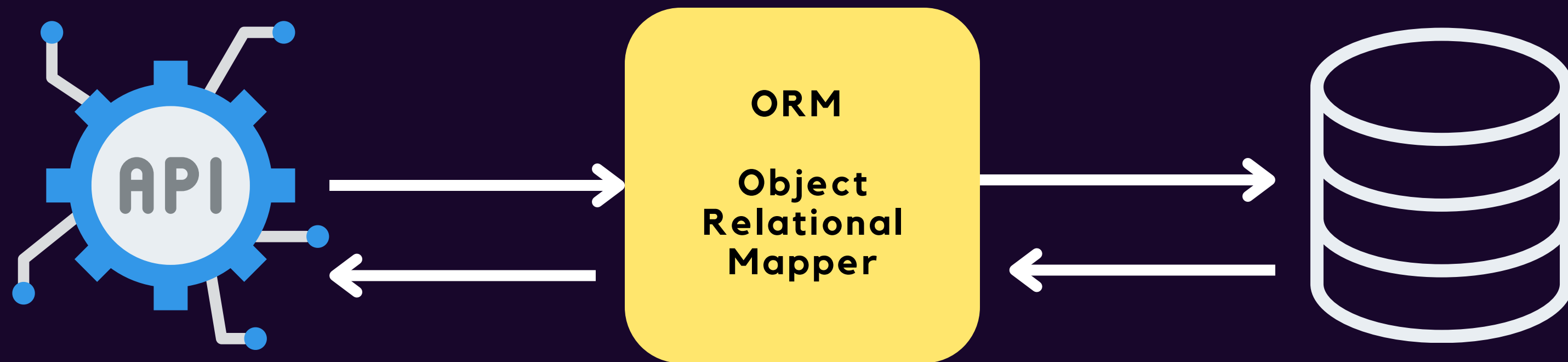


 localuser ▾

Task

- **Explore pg-admin by going through different (gui-based) options for:**
 - **Creating a Table**
 - **Create a Record**
 - **Updating a Record**
 - **Deleting a Record**
 - **Reading a Table**
 - **Reading a Table with conditions etc.**

Connecting Storage to FastAPI App



**ORM converts "Python" to "SQL"
and vice versa.**

Connecting Storage to FastAPI App

```
from fastapi import FastAPI, HTTPException, Depends
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, Session
from pydantic import BaseModel
```

Database Connection

```
# Database connection
DATABASE_URL = "postgresql://localuser:<pass>@localhost/readOne"
engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)

# FastAPI app
app = FastAPI()
```

SQLAlchemy Models

```
# Define SQLAlchemy models
Base = declarative_base()

class MLModel(Base):
    __tablename__ = "ml_models"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String)
    description = Column(String)
    version = Column(Integer)
```

Dependency to get DB connection

```
# Endpoint to create a new ML model
@app.post("/models/")
def create_model(data: dict):
    db_model = MLModel(**data)
    db = SessionLocal()
    db.add(db_model)
    db.commit()
    db.refresh(db_model)
    return db_model
```

Reading Data from the db

```
# FastAPI endpoint to get all ML models
@app.get("/models/")
def get_models():
    db = SessionLocal()
    models = db.query(MLModel).all()
    return models
```