

Бази даних



Реляційні (SQL)

Дані зберігаються в таблицях (рядки та стовпці) зі строгими зв'язками між ними.

Використовують SQL (Structured Query Language) для запитів.

Приклади:

PostgreSQL

MySQL

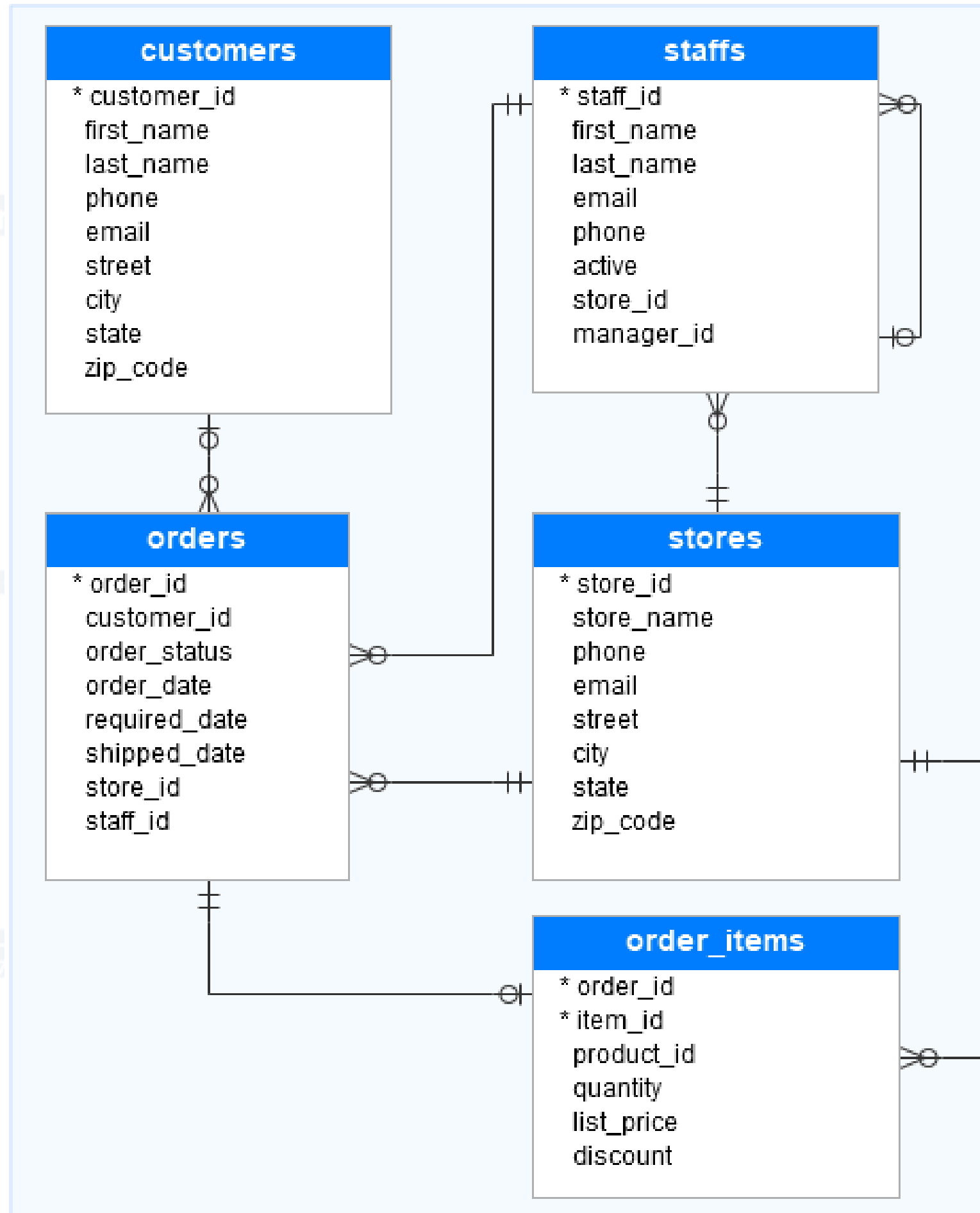
Microsoft SQL Server

Oracle Database

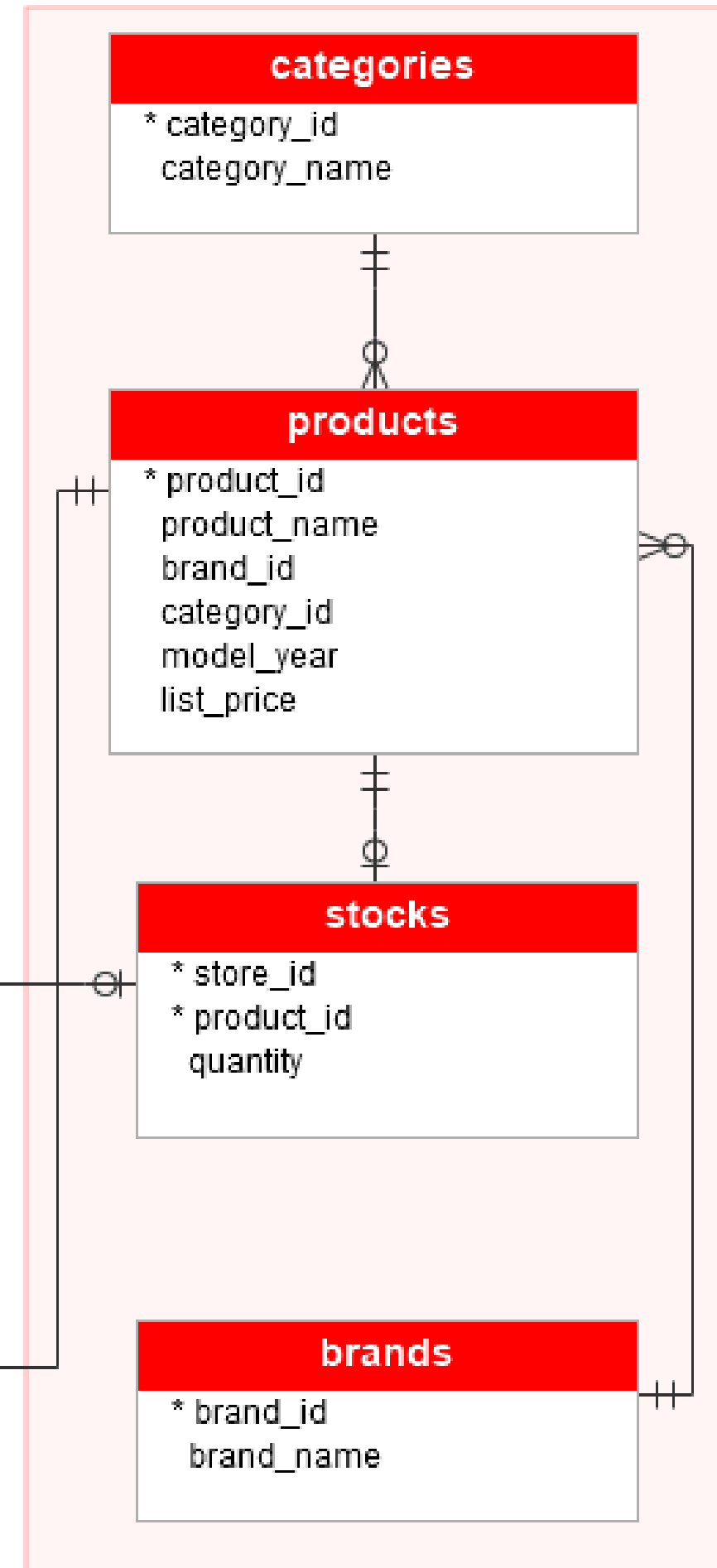
SQLite



Sales



Production





Документо-орієнтовані (NoSQL)

Дані зберігаються у вигляді документів (зазвичай у JSON або BSON форматі).

Гнучка структура (не потрібні суворі схеми).

Приклади:

MongoDB

CouchDB

RethinkDB

```
{
  "_id": "4f5b5c85-d8d3-4f58-8acf-3f5e5e4e59ea",
  "Items": [
    {
      "ProductId": 1,
      "ProductName": "Elden Ring",
      "Price": "49.97",
      "Quantity": 1
    },
    {
      "ProductId": 2,
      "ProductName": "FIFA 23",
      "Price": "69.97",
      "Quantity": 1
    }
  ]
}
```





```
from pymongo import MongoClient
```

```
client = MongoClient("mongodb://localhost:27017/")
```

```
db = client["mydatabase"]
```

```
collection = db["users"]
```

```
user = {"name": "Іван", "age": 25, "city": "Київ"}
```

```
insert_result = collection.insert_one(user)
```

```
user = collection.find_one({"name": "Іван"})
```

```
collection.update_one({"name": "Іван"}, {"$set": {"city": "Одеса"}})
```

```
collection.delete_one({"name": "Олег"})
```

Ключ-значення (Key-Value)

Дані представлені як пари ключ-значення.

Висока швидкість доступу.

Приклади:

Redis

Amazon DynamoDB

Riak



Простий ключ (Simple Primary Key)

Це єдине унікальне поле, яке ідентифікує запис у таблиці.
Також називається Partition Key (PK) або Hash Key.

UserId (PK) | Name

1001 | Олександр

1002 | Марія

1003 | Іван

Складений ключ (Composite Primary Key)

Складається з **Partition Key (PK)** та **Sort Key (SK)**.

Partition Key визначає, у якій **партиції** буде збережено дані.

Sort Key дозволяє групувати дані та швидко шукати серед них. <->

CustomerId (PK)	OrderId (SK)	OrderDate
-----------------	--------------	-----------

12345	1001	2024-02-10
12345	1002	2024-02-11
67890	2001	2024-02-12

Глобальний вторинний індекс (GSI – Global Secondary Index)

Дозволяє запитувати дані за будь-яким іншим атрибутом, окрім Primary Key.

GSI може мати свій власний Partition Key і Sort Key.

Дозволяє гнучкіші запити, але займає більше пам'яті.

UserId (PK) | Name | Email (GSI)

1001	Олександр	alex@gmail.com
1002	Марія	maria@yahoo.com
1003	Іван	ivan@ukr.net

Локальний вторинний індекс (LSI – Local Secondary Index)

Використовується тільки у таблицях зі складеним ключем (Partition Key + Sort Key).

Має той самий Partition Key, але інший Sort Key.

Дозволяє альтернативне сортування записів.

Основний ключ: CustomerId (PK), OrderId (SK)

Локальний індекс: CustomerId (PK), OrderDate (SK)

CustomerId (PK) | OrderId (SK) | OrderDate (LSI)

12345	1001	2024-02-10
12345	1002	2024-02-11
67890	2001	2024-02-12



```
import boto3
```

```
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
```

```
table = dynamodb.Table(table_name)
```

```
table.put_item( Item={ 'UserId': '1', 'Name': 'Іван', 'Age': 25, 'City': 'Київ' } )
```

```
response = table.get_item(Key={'UserId': '1'})
```

```
table.update_item( Key={'UserId': '1'}, UpdateExpression="SET City = :new_city", ExpressionAttributeValues={':new_city': 'Одеса'} )
```

```
table.delete_item(Key={'UserId': '3'})
```

Графові бази даних: короткий огляд

Графові бази даних (Graph Databases) — це NoSQL-бази, спеціально створені для роботи з вузлами (nodes) та зв'язками (edges). Вони ефективні для зберігання і запитів до сильно пов'язаних даних, таких як соціальні мережі, рекомендаційні системи або управління взаємозв'язками між сутностями.

Основні поняття

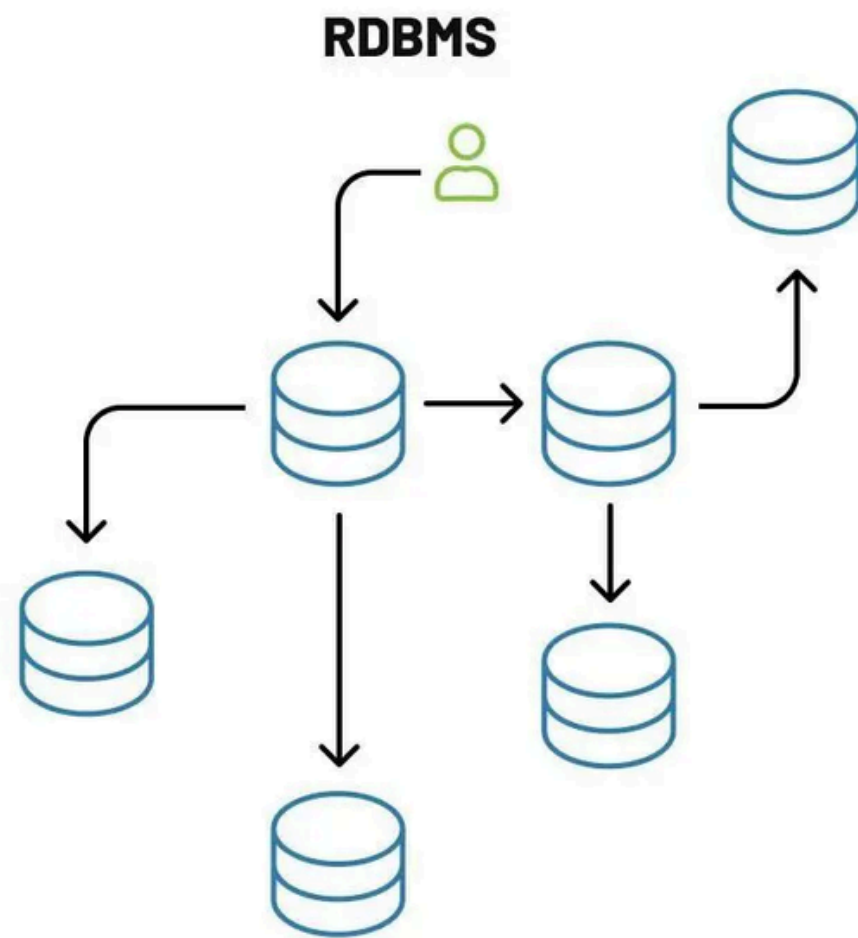
Вузли (Nodes) – об'єкти (аналог рядків у SQL).

Зв'язки (Edges) – зв'язки між об'єктами, які можуть містити атрибути.

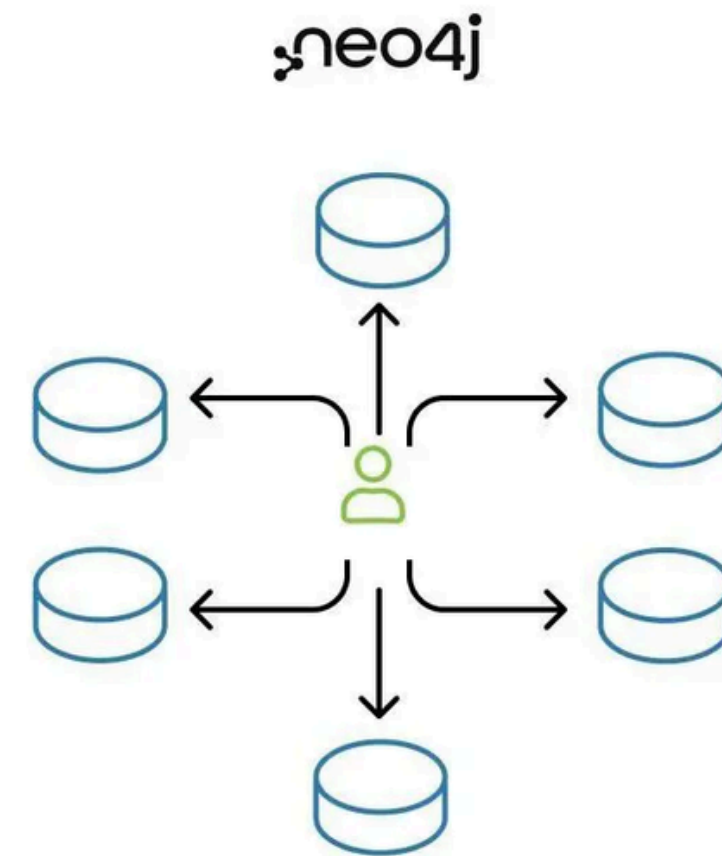
Властивості (Properties) – додаткові характеристики вузлів і зв'язків.

Мітки (Labels) – категорії для групування вузлів.

Task: Assemble holistic 'Person' entity as stored across the enterprise

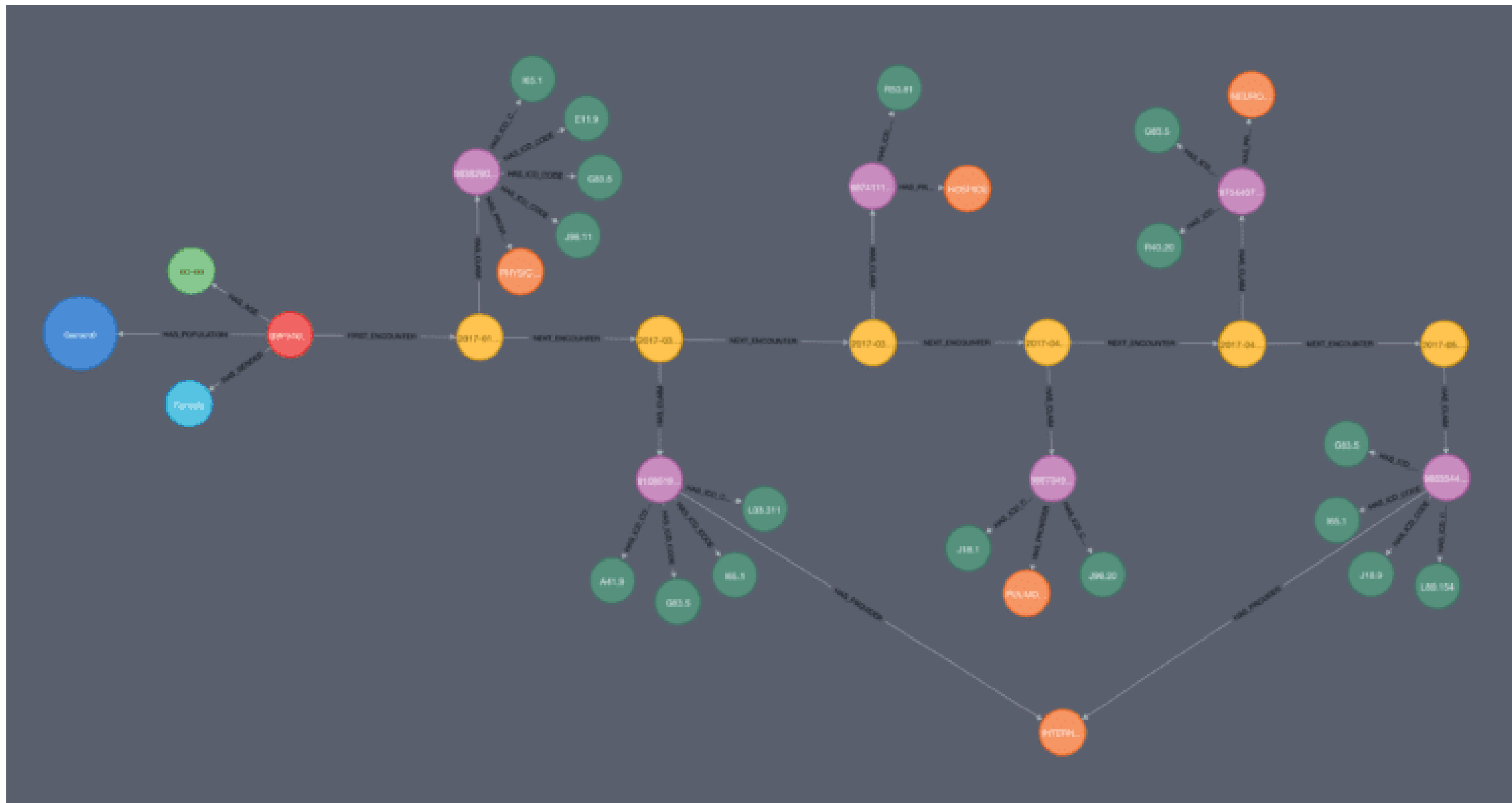


Query code



Query code







Neo4j (найпоширеніша, використовує Cypher)

Amazon Neptune (AWS-орієнтована)

ArangoDB (гібрид графової та документної БД)

JanusGraph (працює з Cassandra і HBase)



```
from neo4j import GraphDatabase
```

```
# Підключення до локального або віддаленого Neo4j-сервера
```

```
URI = "bolt://localhost:7687" # Bolt-протокол для швидкого підключення
```

```
USERNAME = "neo4j"
```

```
PASSWORD = "password"
```

```
# Ініціалізація драйвера
```

```
driver = GraphDatabase.driver(URI, auth=(USERNAME, PASSWORD))
```

```
def create_person(tx, name, age):
```

```
    query = """ CREATE (p:Person {name: $name, age: $age})
```

```
    RETURN p
```

```
    """
```

```
    result = tx.run(query, name=name, age=age)
```

```
    return result.single()[0]
```

```
# Виконання запиту
```

```
with driver.session() as session:
```


```
    person = session.write_transaction(create_person, "Іван", 25)
```

```
    print("Created:", person)
```

Колонкові (Columnar) NoSQL бази даних

Колонкові бази даних (Column-Family Stores) — це тип NoSQL баз даних, який зберігає дані у вигляді колонок замість рядків, що забезпечує високу швидкість обробки великих обсягів даних. Вони оптимізовані для аналітики, швидкого читання та масштабованості.

- ◆ Apache Cassandra – висока масштабованість, популярна в Big Data
- ◆ Apache HBase – працює поверх Hadoop, підходить для аналітики
- ◆ Google Bigtable – використовується в Google Analytics, YouTube
- ◆ ScyllaDB – швидша альтернатива Cassandra



```
from cassandra.cluster import Cluster

# Підключення до кластера Cassandra
cluster = Cluster(["127.0.0.1"])
session = cluster.connect()
# Створення Keyspace (аналог бази даних)

session.execute("""
CREATE KEYSPACE IF NOT EXISTS test_keyspace
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}
""")
# Використання Keyspace session.set_keyspace("test_keyspace")
```

PostgreSQL

Тип: Об'єктно-реляційна база даних (ORDBMS).

Основні особливості:

Підтримка складних типів даних, JSONB.

Надійність та ACID-транзакції.

Розширюваність (можна додавати свої функції, типи, оператори).

Підтримка PostGIS для геопросторових даних.

Використання: Веб-додатки, аналітика, геодані, складні реляційні структури.

MySQL

Тип: Реляційна база даних (RDBMS).

Основні особливості:

Висока швидкість на читаннях.

Широке використання в веб-розробці (WordPress, Shopify).

Підтримка реплікації та кластеризації.

Недостатня підтримка складних запитів у порівнянні з PostgreSQL.

Використання: Веб-додатки, CMS, електронна комерція.



MariaDB

Тип: Форк MySQL (RDBMS).

Основні особливості:

Повністю сумісна з MySQL.

Швидша робота з певними запитами.

Додаткові можливості (JSON, розширені механізми реплікації).

Використання: Веб-сайти, корпоративні додатки.



Microsoft SQL Server

Тип: Реляційна база даних (RDBMS).

Основні особливості:

Глибока інтеграція з продуктами Microsoft (Azure, Power BI).

Висока продуктивність для enterprise-рішень.

Вбудована аналітика та підтримка JSON.

Дорога комерційна ліцензія.

Використання: Великі корпорації, фінанси, BI (бізнес-аналітика).



Oracle Database

Тип: Реляційна база даних (RDBMS).

Основні особливості:

Масштабованість та безпека.

Вбудовані механізми реплікації та резервного копіювання.

Висока продуктивність для великих транзакційних систем.

Висока вартість.

Використання: Банки, великі корпорації, державні установи.



SQLite

Тип: Вбудована реляційна база даних (RDBMS).

Основні особливості:

Не потребує серверної частини.

Легка та швидка.

Підтримує SQL, але з обмеженнями.

Використання: Мобільні додатки (Android, iOS), невеликі вбудовані системи.

Amazon Aurora

Тип: Хмарна реляційна база даних (RDBMS).

Основні особливості:

Сумісна з MySQL та PostgreSQL.

Автоматичне масштабування та реплікація.

Висока відмовостійкість.

Використання: Хмарні рішення, AWS-проекти.

