Lecture 7. Database Security and NoSQL Databases (Chapter 24 and 30)

alisa.lincke@lnu.se

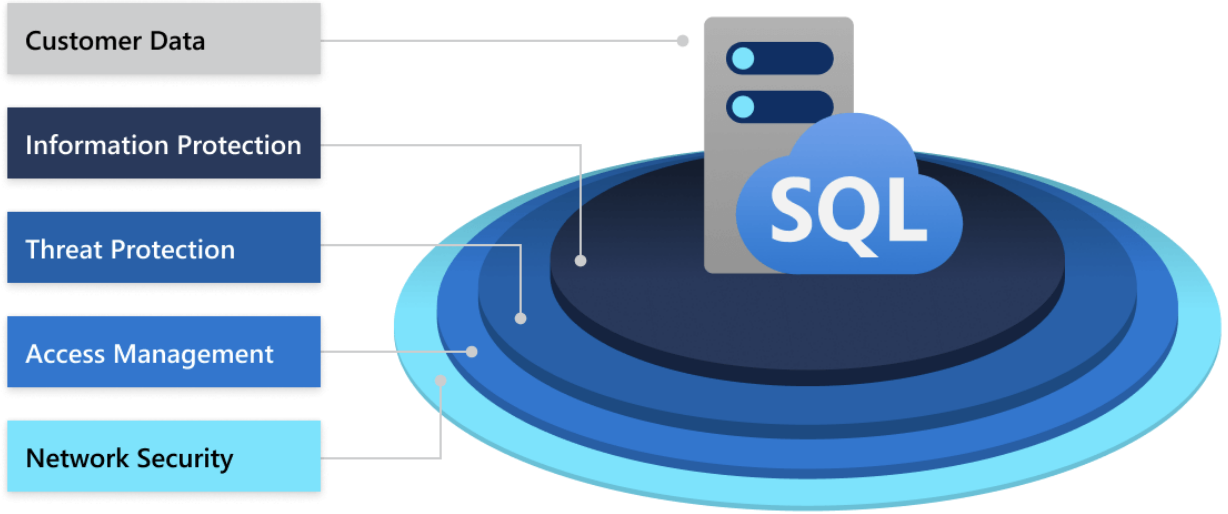
Linnæus University

Outline

- Database Security
  - Sensitive Data
  - Privileges and Role-based Access Control
- Break 10 min
- NoSQL Databases
  - Document-Based (MongoDB)
  - Key-Value Stores
  - Graph-based Stores (Neo4j, OrientDB)

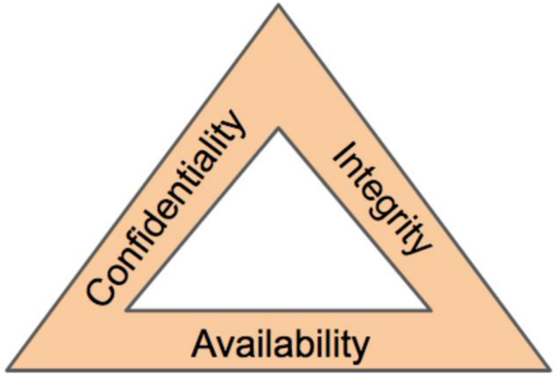
Database Security

- The objective of database security is to secure sensitive data and maintain the confidentiality, integrity, and availability, (CIA) of the database
- Database security protects the database management system and associated applications, systems, physical and virtual servers, and network infrastructure



Source: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-database-security

What is CIA Triad?



- Confidentiality
  - Protect information from unauthorized access and misuse
  - Protect sensitive information according to GDPR regulations
- Integrity
  - Improper modification of information or unauthorized alternation
  - Provide assurance in the accuracy and completeness of data
  - Secure access control on the system level (e.g., system users are only able to alter information that they are legitimately authorized to alter)
- Availability
  - Must be available to authorized users

Outsiders and Insiders

- Outsiders include anyone from lone hackers to cybercriminals seeking business disruption
- Insiders may comprise current or former employees, curiosity seekers, and customers or partners who take advantage of their position of trust to steal data, or who make a mistake resulting in an unintended security event
- Both outsiders and insiders create risk for the security of personal data, financial data, trade secrets, and regulated data

Threats of Security

- Insider Threats:
  - A malicious insider with bed intent
  - A negligent person within the organization who exposes the database to attack through careless actions
  - Human error: weak passwords, password sharing, configuration mistakes, and other irresponsible user behaviors which cause nearly 90% of security breaches
- Outsider Threats:
  - SQL/NoSQL injection Attacks
  - Compromising or stealing the credentials of a privileged administrator or application.
  - Stealing data from nonproductive environments such as DevTest which are usually not encrypted
- Database Management System Vulnerabilities

Database Security Layers

Security Level	Description	Database Security Solutions
Physical level	The organization has own data center, servers, own cloud services. This level is vulnerable for infrastructure damage due to physical/natural disaster, human accidents, and malicious attacks from internal or external personnel.	Security of premise (locks, camera, security personnel, accessed by authorized individuals, access is recorded, logged). Security of data centers
Network Level	The data communication happens via network.	HTTPs protocols, VPN or SSH connection, block all public network access to database servers, firewalls
Operation System Level		Regular security updates, patches updates
Database level	Sensitive information stored separately, GDPR	Privileges and access control, Data encryption, backup encryption,
DBMS level		Control Access, Strong passwords, regular security updates (patches)
Application level	Decides authorized access to the backend. This level of security should ensure attacker should not get control on hardware and other applications	Authentication, web application firewall (WAF), Secure coding practices to prevent SQL injection attacks, session management

Database Security Level Threats

Threat	Description	Suggestions
Data loss and leakage	Unauthorized updating, deletion, removal or extraction of data	• Data encryption at rest • Authentication and authorization • backup and retention policies • Secure APIs and Data integrity checks should be implemented
Access data and control	Due to lack of access control mechanism, confidential information can be seen or used by authorized users	Access control mechanism should be implemented Key based access, various encryption techniques
SQL Injections	Attackers inject malicious SQL code into input fields or query parameters to manipulate the database or extract sensitive information.	Secure coding practices, additional security check before sending the request to the server.

Database Administrator (DBA)

- Classifies users and data in accordance with the policy of the organization
- DBA has superuser account which provides powerful capabilities that are not available to regular database accounts and users.
- DBA is responsible for overall security of the database system.
- Has privileged commands to perform the following actions:
  - Account creation
  - Privilege granting
  - Privilege revocation
  - Security level assignment

Access Control, User Accounts, and Database Audits

- A person or group request an database account, and which data will be accessed
- DBA creates new account number and password with a certain privileges rights (remove ,create, alternate, read, write, etc.)
- The DBA usually creates table with all users which have access to the database, this table is encrypted with two columns: account number and password.
- The database system must also keep track of all operations on the database that are applied by a certain user (using System Log files).
- When the user is login, the DBMS can record the account number with associated device/computer. Is important to keep track of the database alternation operations (update, delete).
- Database audit is performed to find **illegal or unauthorized operations**
- **Access control** is done by granting and revoking of privileges

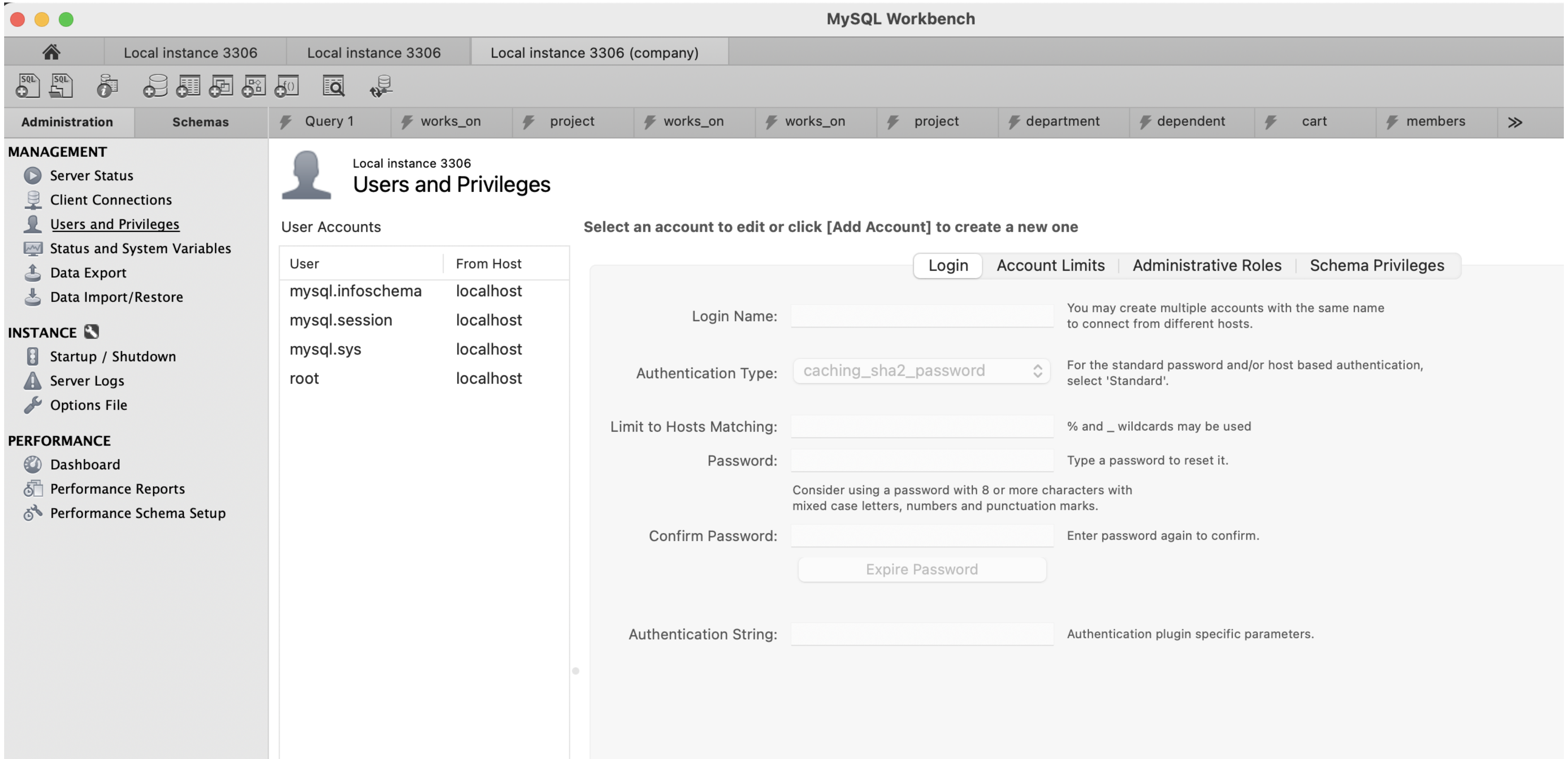
Privileges

- A privilege in a database management system is the permission to execute certain actions on the database.
- Two levels for assigning privileges:
  - Account level:
    - Example: CREATE, DROP, ALTER, SELECT privileges
  - Relation (or Table) level:
    - Example: DELETE, SELECT, UPDATE, INSERT privileges
- MySQL Privileges: https://dev.mysql.com/doc/mysql-security-excerpt/5.7/en/privileges-provided.html

Access Matrix Model

- The granting and revoking privileges organized so called the **access matrix model**, where**:**
  - Rows represents *subjects* (users, accounts, programs)
  - Columns represents *objects* (relations, records, columns, views, operations)
  - Each position in a matrix *M(i,j)* represent the type of privileges (*read, write, update*)
  - *read* (SELECT), *write* (INSERT), and *update* (DELETE, UPDATE, INSERT) privileges
- The one who created the SCHEMA is the owner account and has right to grand or revoke privileges.
- Granting and revoking can be done in two ways:
  - Using Views (Recommended)
  - Using Grant Option

MySQL Workbench Users and Privileges



Local instance 3306

Users and Privileges

User Accounts

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

Select an account to edit or click [Add Account] to create a new one

Login Name: You may create multiple accounts with the same name to connect from different hosts.

Authentication Type: caching_sha2_password For the standard password and/or host based authentication, select 'Standard'.

Limit to Hosts Matching: % and _ wildcards may be used

Password: Type a password to reset it.

Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

Confirm Password: Enter password again to confirm.

Authentication String: Authentication plugin specific parameters.

Web Application Security Level Threats

Threat	Description	Suggestion
SQL injection attack	Attacker inserts a malicious code into SQL standard queries that gives him access to the database.	A strong user input detection and sanitization systems should be developed and implemented in the application
Cross-site scripting	Intruder adds a code/script into the web page which may be stored permanently or reflected just for the time on the web page	Various technologies like Web Application Vulnerability Detection Technology, Content Filtering, Content Based Data Leakage Prevention Technology etc. are available to detect and mitigate the attack
Cookie poisoning	Intruder can change the content of the cookie	Cookie saving should be disabled. Cookie cleanup is necessary. Session Management. HTTPS. Use session tokens
Backdoor and debug options	website debugging options if left by the developer then attacker can enter into the website easily and modify the content	At the time of website publishing, debug option should be disabled
Hidden field manipulation	Hidden fields are used by the developers to maintain the state. If it gets noticed then attacker can use to enter in the service	Use as less as possible of hidden fields and also query strings

SQL injection Attack (1)

- Attacker injects a string input through the application:
  - Changes or manipulates SQL statement to attacker’s advantage
- Types of attacks:
  - SQL Manipulation: changes the SQL command in the application, for example by adding conditions to the WHERE clause of a query
  - Or expanding a query components using set operations as UNION, INTERSECT etc.
  - Typical attack occurs during database login:
    - SELECT * FROM users WHERE username=“jake” and password=“jakespasswd”;
    - With SQL injection:
    - SELECT * FROM users WHERE username=“jake” and password=“jakespasswd” OR ‘x’=‘x’;

true

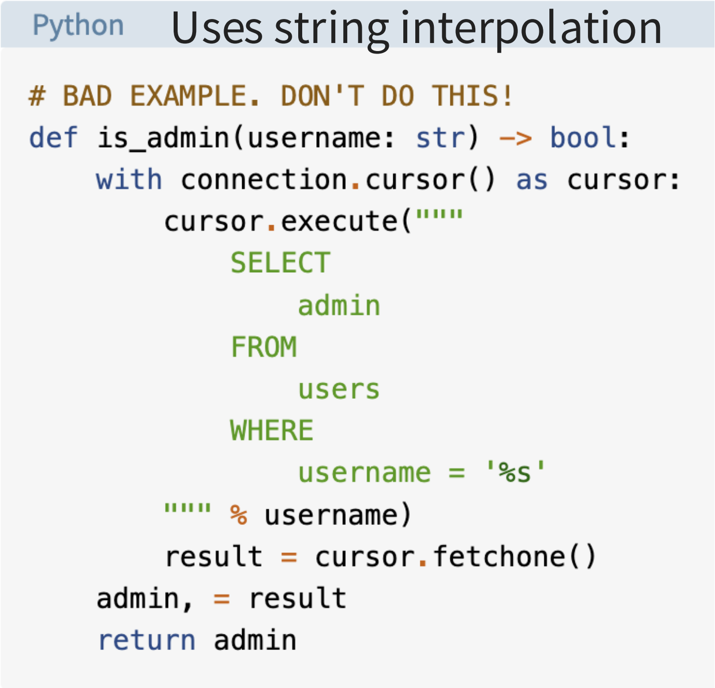
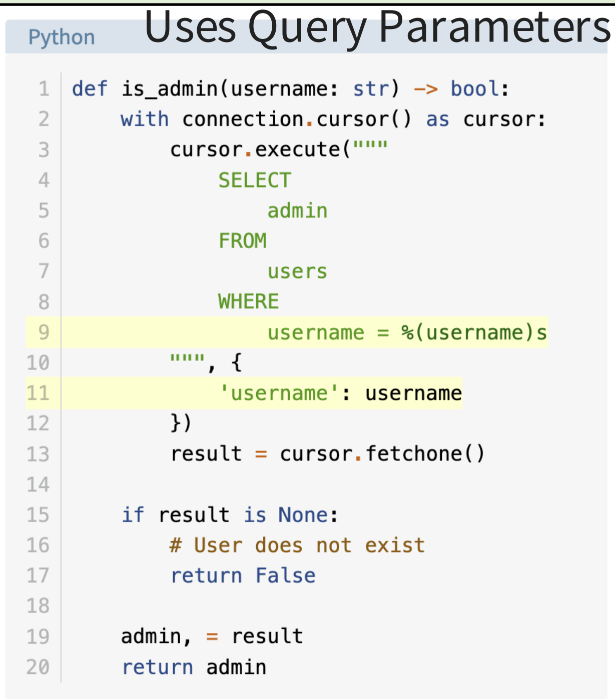
SQL injection Attack (2)

- Code Injection
  - The attacker can inject code into a program to change the course of execution
- Function Call Injection
  - a database function inserted into a vulnerable SQL statement to manipulate the data

Protection Techniques against SQL Injection

- **Bind Variables (Using Parameterized Statements in Python)**

Example for Python Applications taken from https://realpython.com/prevent-python-sql-injection/

Not Secure Approach (good for SQL injection attacks)	Secure Approach to prevent SQL Injection attacks
 Python Uses string interpolation	 Python Uses Query Parameters

Uses string interpolation:

```python
# BAD EXAMPLE. DON'T DO THIS!
def is_admin(username: str) -> bool:
    with connection.cursor() as cursor:
        cursor.execute("""
            SELECT
                admin
            FROM
                users
            WHERE
                username = '%s'
        """ % username)
        result = cursor.fetchone()
    admin, = result
    return admin
```

Uses Query Parameters:

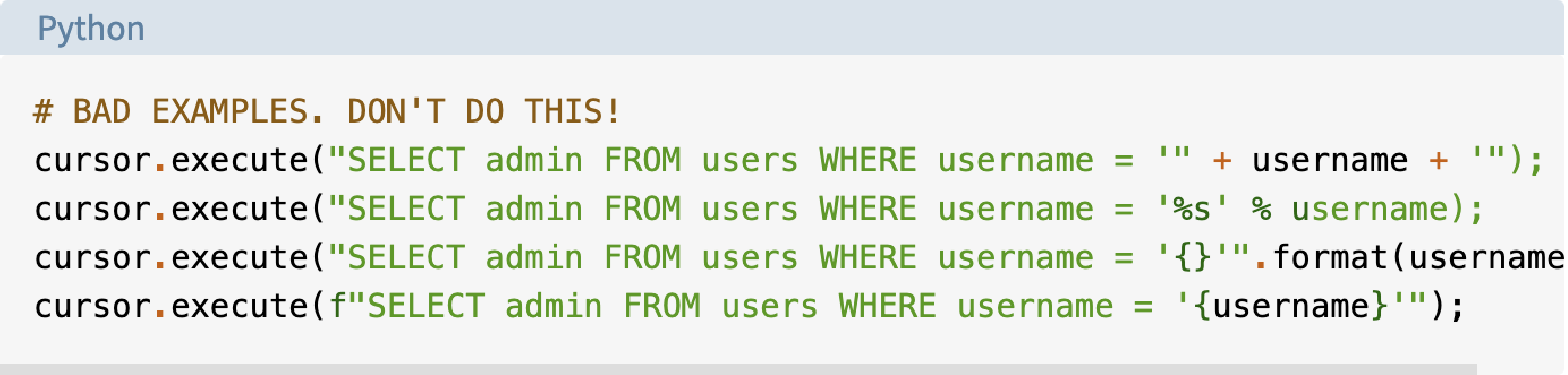
```python
def is_admin(username: str) -> bool:
    with connection.cursor() as cursor:
        cursor.execute("""
            SELECT
                admin
            FROM
                users
            WHERE
                username = %(username)s
        """, {
            'username': username
        })
        result = cursor.fetchone()

    if result is None:
        # User does not exist
        return False

    admin, = result
    return admin
```

Bad SQL Query Examples

Not Secure



Python

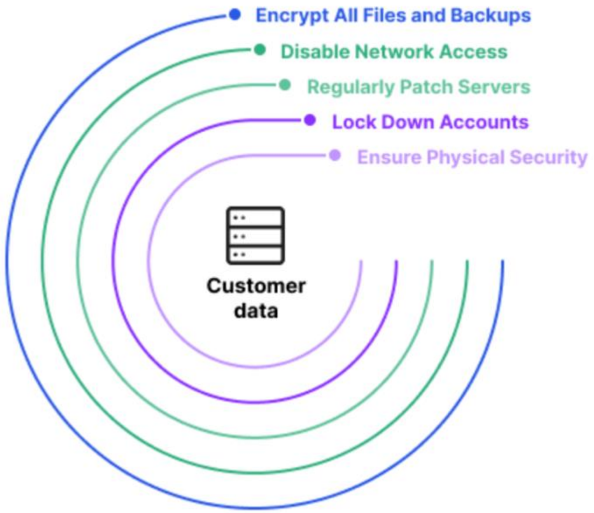
```
# BAD EXAMPLES. DON'T DO THIS!
cursor.execute("SELECT admin FROM users WHERE username = '" + username + '");
cursor.execute("SELECT admin FROM users WHERE username = '%s' % username);
cursor.execute("SELECT admin FROM users WHERE username = '{}'".format(username
cursor.execute(f"SELECT admin FROM users WHERE username = '{username}'");
```

Secure SQL Query Examples

```
# SAFE EXAMPLES. DO THIS!
cursor.execute("SELECT admin FROM users WHERE username = %s'", (username, ));
cursor.execute("SELECT admin FROM users WHERE username = %(username)s", {'username':
username});
```

Database Security Best Practices

- Separate database servers from application server
- Isolate sensitive data from non-sensitive data
- Set up an HTTPS proxy server
- Avoid using default network ports
- Use real-time database monitoring
- Use database and web application firewalls
- Deploy data encryption protocols
- Create regular encrypted backups of your database
- Use strong user authentication
- Use security patches regularly in database management system
- Deploy regular vulnerability testing



Sensitive Data and GDPR

- Sensitivity of data is a measure of the importance assigned to the data by its owner for the purpose of protection.
- Some databases contain no sensitive data, while other only sensitive data, or both sensitive and not sensitive data.
- According to GDPR, sensitive data is a personal data, and "'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;" [3]
- DBA must ensure additional security for columns containing the sensitive (or personal) information (use private/public key encryption on both sides database and application)

How to store sensitive data?

- Use Secure Sockets Layer (SSL) is a standard security technology for establishing an encrypted link between a server and a client
- Use a secure encryption key
- The encrypted sensitive data can be stored as a BLOB type in MySQL and there are build in MySQL encryption functions
- Use encryption/decryption in the application code
- Transfer encrypted data over Internet
- Delete sensitive data which you no longer need
- Encrypt backups
- MySQL Enterprise TDE enables data-at-rest encryption by encrypting the physical files of the database. Data is encrypted automatically, in real time, prior to writing to storage and decrypted when read from storage. As a result, hackers and malicious users are unable to read sensitive data directly from database files.

Break 10 min

Data

- The amount of data worldwide has been growing since 1994, as the result there is an explosive growth in the amount of data generated and communicated over networks worldwide.
- The applications collecting/generating every day information:
    - The social media websites (LinkedIn with more than 250 million users, Facebook with 1.3 billion and 800 million active users everyday, Twitter has ca 980 million with ca. 1 billion tweets per day)
    - Satellite imagery
    - Communication Networks (Telenor, Telia, etc.)
    - Banking
    - Other

Data Examples

- **Network data:**
    - Facebook: 500 million users
    - Twitter: 300 million users
    - Tele Communication data
    - Transport data
- **Document data:**
    - Web as a document repository ca 50 billions of web pages
    - Wikipedia: 4 million articles
    - Archives
- **Financial data:**
    - Banking, Accounting
- **Transaction data:**
    - Credit card companies: billions of transactions per day.
    - Queries in search engines (e.g., Google)
    - Membership cards allows to collect information about customer preferences/needs

- **Sensors data:**
    - Mobile sensors
    - Internet of Things (IoT) sensors network
    - Climate data: thousands of station
- **Linked data:**
    - Subtype of network data with semantics
- **Geographical data:**
    - Maps, geodata
- **Event-data:**
    - App log data if user interaction with App
- **Video data:**
    - Human movements in Sport,
- **Image data:**
    - Satellite imagery
    - Medical Images

Characteristics of Data

- Dependencies:
  - nondependencies (e.g., text), and
  - dependency-oriented data having relationship in time (time series, sequential data, spatial data)
- Data structure:
  - Structured: table (column, rows) or CSV file, network/graph (nodes, edges), objects with nested objects (JSON files)
  - Unstructured: image (pixels in rows,columns), voice data, text data

Characteristics of Big Data

The Gartner Group introduced five V's characteristics for Big Data:

- **Volume**: refers to the size of the data stored and managed by the system. Examples: sensors, social media, environmental recording devices, credit card readers (transactional data), and more.
- **Velocity**: refers to frequency or speed of data to be generated, stored, processed. For example, *streaming data* (sensors, telecommunication data, health vital signals data, stock exchanges,)
- **Variety**: refers to structure/type of data, event data (clickstream, social media), location data (e.g., geospatial data, maps), images (surveillance, satellites, medical scanning), supply chain data, sensors data, video data (movies, YouTube streaming, etc.)
- **Veracity**: refers to the credibility of the source, and the suitability of data for its target audience (trust, and availability)
- **Value**: refers to what can we do with this data (to solve some problem, need, statistics, quality)

SQL-based Data and NoSQL based Data

- Data for SQL-based databases are:
  - University database
  - Hospital database
  - Traveling Agency database
  - Accounting database
  - Banking databases
  - Other…
- Data for NoSQL based databases are:
  - Social media data (network structure + document-based structure)
  - Archives (text data), images, videos (stored as files + document-based database)
  - Event-data or user interaction data with App, usually stored in JSON format, thus document-based database)
  - Sensors data (stored in files or in time-series databases TSBD (e.g.,InfluxData))

NoSQL Databases

- NoSQL (Not Only SQL) are other databases to suit the particular data and its characteristics (5 Vs), and application domain.
- NoSQL characteristics:
  - **Scalability:** where usually horizontal scalability is used by adding more nodes for data storage and processing as the volume of data grows
  - **Availability :** guaranties high availability due to using the distributed approach. In addition, using two access techniques: *hashing* and *range partitioning*
  - **Replication:** support master-slave, and master-master replication.
  - **Consistency:** Horizontal partitioning of the files records in NoSQL is usually used to access concurrently the records. In addition, many NoSQL applications does not require serializability.
  - **Not Required Schema:** allows semi-structured, self-described data (JSON objects). All constrains should be programmed in the application program
  - **Less Powerful Query Language:** only a subset of SQL based language is used (no JOIN operations)
  - **Versioning**: some NoSQL databases allows to store multiple versions of the data items, with the timestamps of when the data version was created.

NoSQL Databases Categories

- 1. Document-based NoSQL database
- 2. NoSQL Key-Value Stores
- 3. Column based or wide column NoSQL:
- 4. Graph-based NoSQL

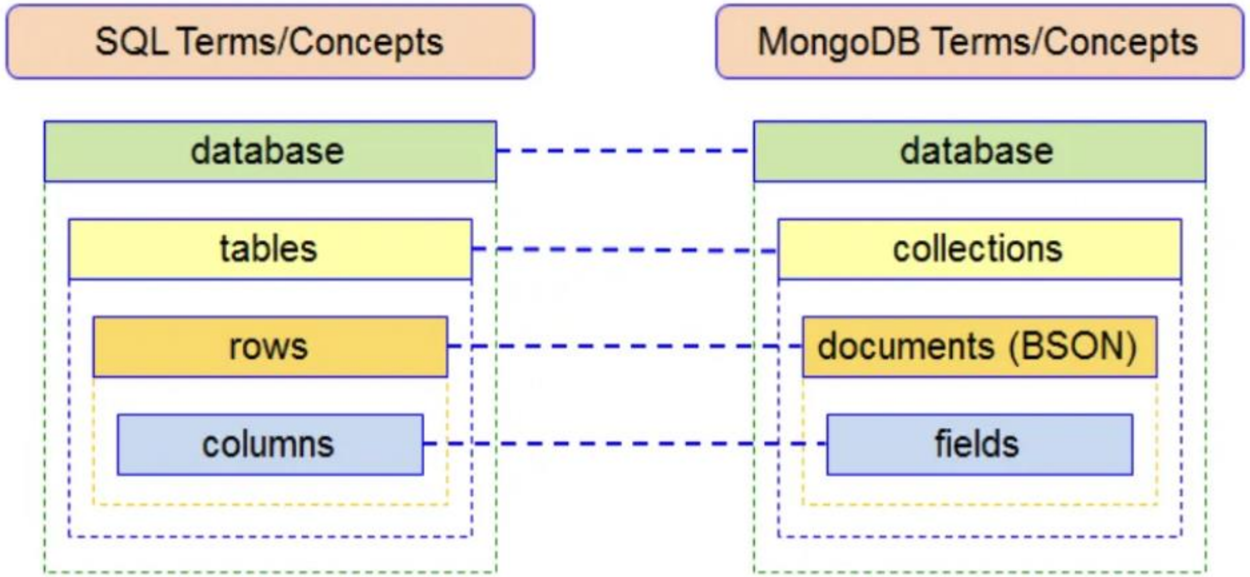
1. Document-based NoSQL database

- Stores data in the form of collections of similar documents/objects
- Document is self-described data usually in BJSON format (Binary JavaScript Object Notation)
- Documents are accessible via their document id, or also indexes.
- Example JSON document/object:

```
{
‘id’:  this.gameID,
‘type’: "playmode",
‘event"’: "point_selection",
‘state’: {‘game_progress’: {‘fields’: {Money: 10, Joy: 50, Health: 30}, ‘score’: 10} ‘value’:{name: ‘Banana’, times: 5}, event_count: 4},
‘timestamp’ : 1667736467
}
```

- Examples of well-known databases: MongoDB, CouchDB, DocumentDB, other

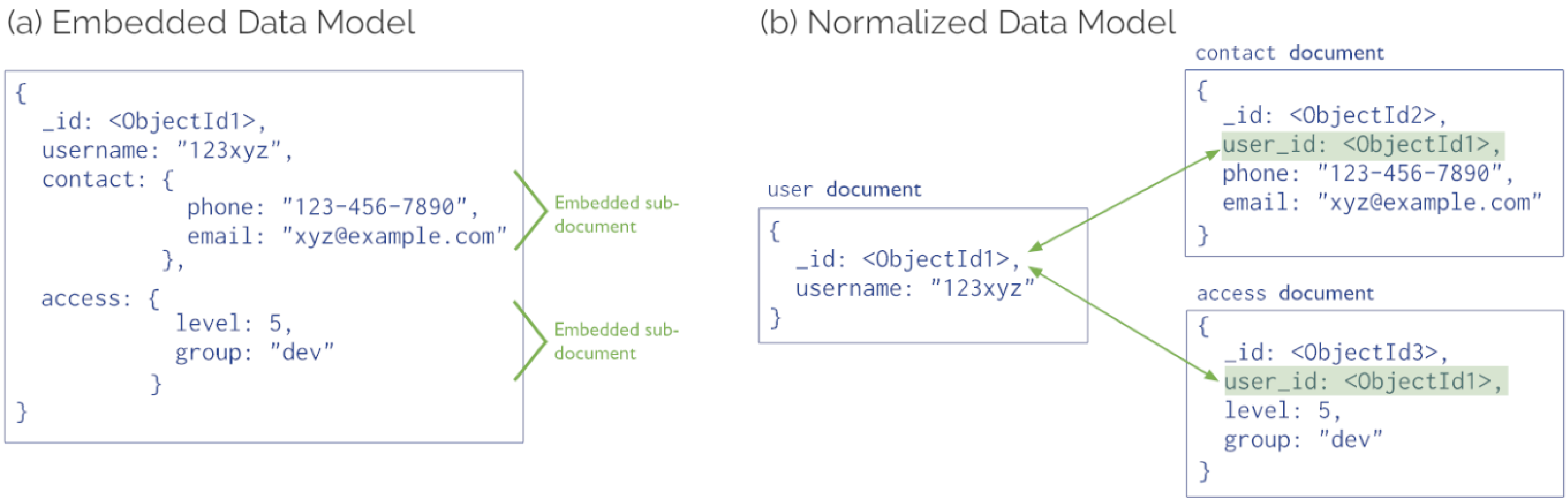
MongoDB



SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
tables	collections
rows	documents (BSON)
columns	fields

Image taken from: https://medium.com/zenofai/scaling-dynamodb-for-big-data-using-parallel-scan-1b3baa3df0d8

MongoDB Data Model (Flexible Schema)



(a) Embedded Data Model

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
              phone: "123-456-7890",
              email: "xyz@example.com"
           },
  access: {
             level: 5,
             group: "dev"
          }
}
```

(b) Normalized Data Model

user document

```
{
  _id: <ObjectId1>,
  username: "123xyz"
}
```

contact document

```
{
  _id: <ObjectId2>,
  user_id: <ObjectId1>,
  phone: "123-456-7890",
  email: "xyz@example.com"
}
```

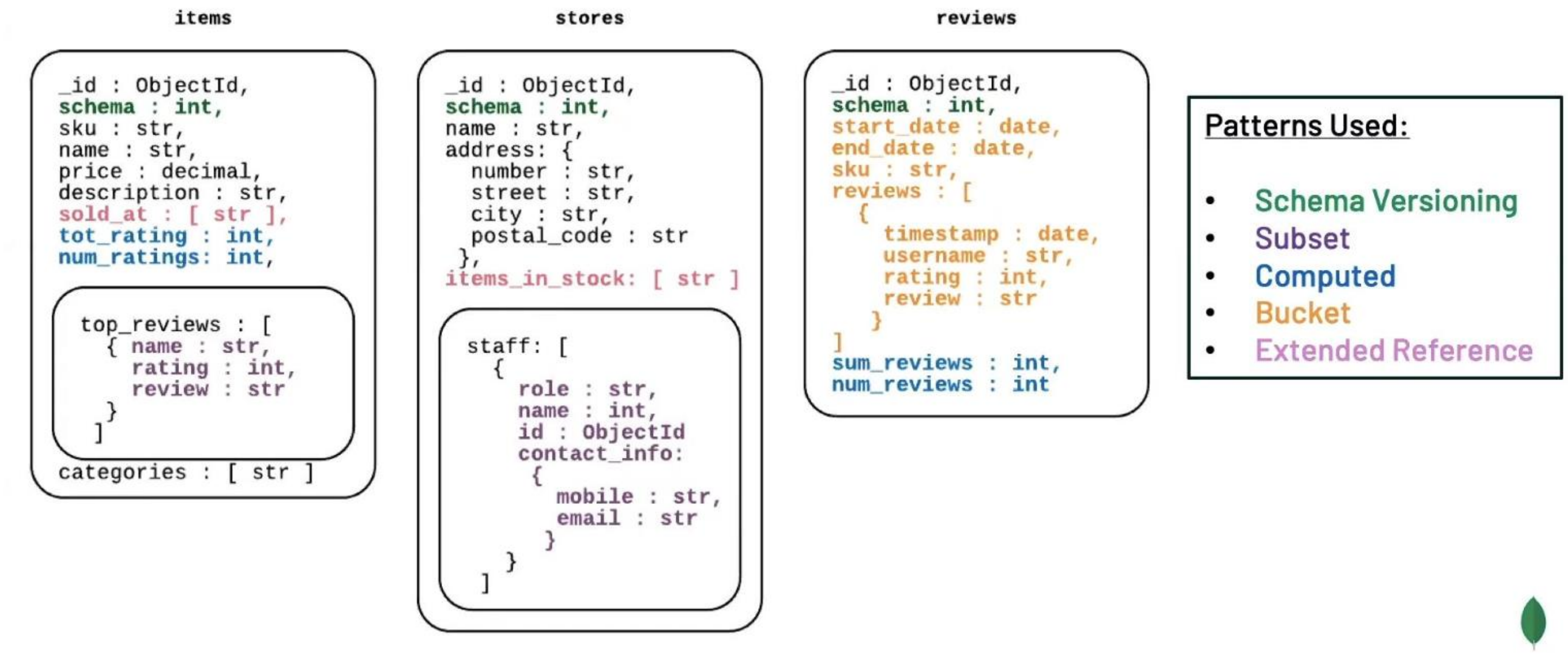
access document

```
{
  _id: <ObjectId3>,
  user_id: <ObjectId1>,
  level: 5,
  group: "dev"
}
```

Model relationships in MongoDB: https://devopedia.org/data-modelling-with-mongodb

Example MongoDB Schema (Model)

Web application, JavaScript, Mongoose library (DB-API)



items

```
_id : ObjectId,
schema : int,
sku : str,
name : str,
price : decimal,
description : str,
sold_at : [ str ],
tot_rating : int,
num_ratings: int,

  top_reviews : [
    { name : str,
      rating : int,
      review : str
    }
   ]
categories : [ str ]
```

stores

```
_id : ObjectId,
schema : int,
name : str,
address: {
  number : str,
  street : str,
  city : str,
  postal_code : str
 },
items_in_stock: [ str ]

  staff: [
    {
      role : str,
      name : int,
      id : ObjectId
      contact_info:
       {
         mobile : str,
         email : str
       }
     }
   ]
```

reviews

```
_id : ObjectId,
schema : int,
start_date : date,
end_date : date,
sku : str,
reviews : [
  {
    timestamp : date,
    username : str,
    rating : int,
    review : str
   }
]
sum_reviews : int,
num_reviews : int
```

Patterns Used:

- Schema Versioning
- Subset
- Computed
- Bucket
- Extended Reference

An example MongoDB data model using various design patterns.
Source: Genkina 2020, 31:38

MongoDB Operations

Using MongoDB CLI (Command-line interface)

- Create database:
  - use “db_name”
- Create collection:
  - db.createCollection(name,structure)
  - For example: db.createCollection(“project”,{capped:boolean, size:int,max:int})
- CRUD operations:
  - db.collection_name.insert(<document(s)>)
  - db.collection_name.remove(<condition>)
  - db.collection_name.find(<condition>)

MongoDB

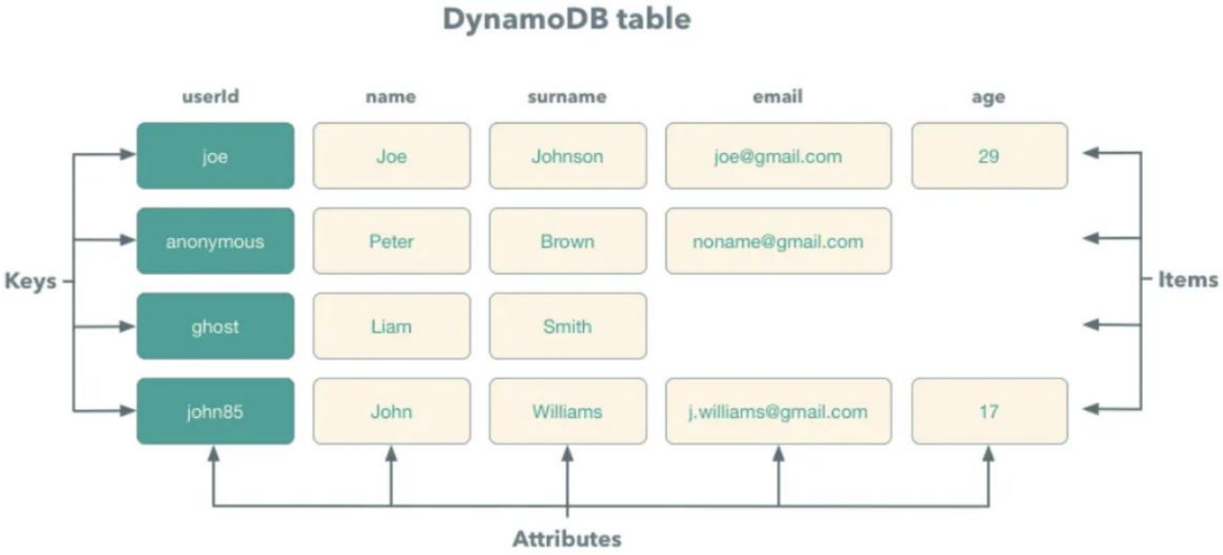
- MongoDB Server should be installed OR
- Using MongoDB Atlas cloud (no need to install Mongodb server)
- The DB-API is used to access mongodb from application
- For example, for web applications the JavaScript based Mongoose library is used

2. Key-Value Stores

- These systems have a simple data model based on fast access by the key to the value associated with they key
- The key is a unique identifier associated with a data item (value)
- The value can be a record, an object or a document, or even more complex data structure. Support different data types (strings of bytes, arrays of bytes, tuples, JSON objects)
- No query language
- Set of operations that can be used by the application programmers (GET,PUT,DELETE).
- Main characteristic: is that every value (data item) must associate with unique key and that retrieving the value by using key must be very fast.
- Usability/Applicability Examples: for streaming data, for real-time data processing and analyzes.
- Databases: Redis, Apache Kafka, Apache Cassandra, DynamoDB, other

DynamoDB (1)

- Provided by Amazon Web Services (AWS)
- Uses concepts of table, items, and attributes
- Item is a value

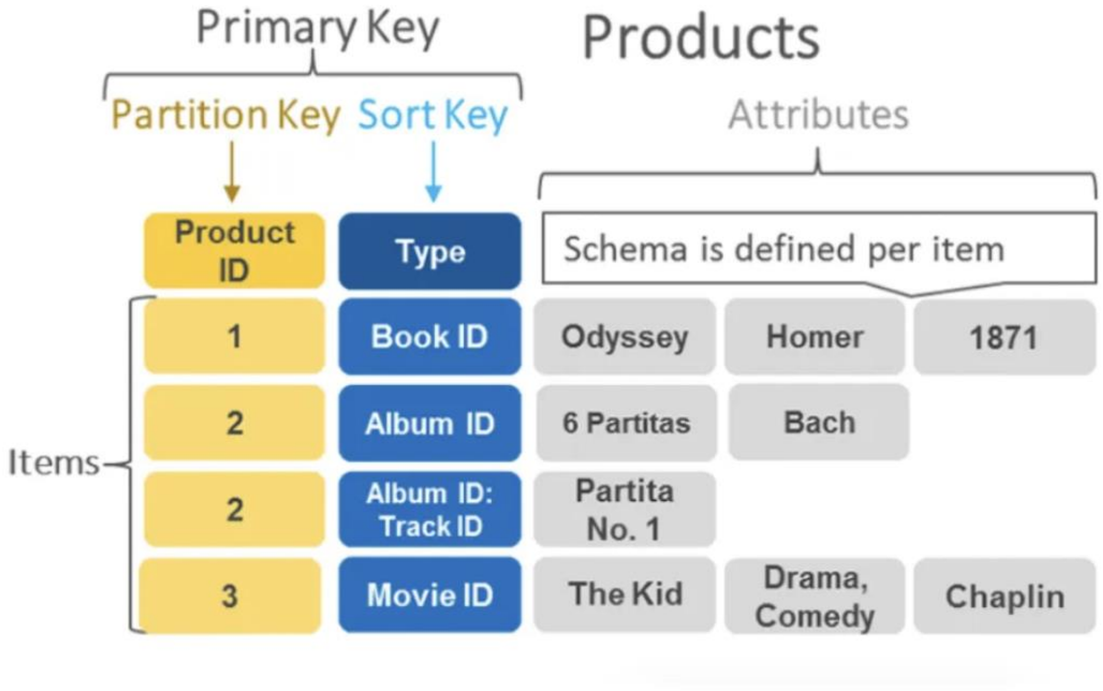


DynamoDB table

userId	name	surname	email	age
joe	Joe	Johnson	joe@gmail.com	29
anonymous	Peter	Brown	noname@gmail.com	
ghost	Liam	Smith		
john85	John	Williams	j.williams@gmail.com	17

DynamoDB (2)

- Table has a ***name*** and ***primary key***
- A primary key consists from two attributes (partition key, sort key).
- Partition key is used for hashing, and because there are will be same partition key, additional sorting key is used for ordering records in the same partition.



Product ID	Type	Schema is defined per item		
1	Book ID	Odyssey	Homer	1871
2	Album ID	6 Partitas	Bach	
2	Album ID: Track ID	Partita No. 1		
3	Movie ID	The Kid	Drama, Comedy	Chaplin

Graph-based Databases

- Graph databases is represented as a graph, which is a collection of vertices (nodes) and edges.
- Nodes and edges can be labeled to indicate the types of entities and relationships they represent
- Uses graph theory and algorithms for optimizing the data search
- Own query language (e.g., Cypher)
- Applications: analyzing social networks data, recommendations, geospatial data, postal delivery network
- Databases: Neo4j, OrientDb,

Neo4j

- Uses concepts of ***nodes*** and ***relationships (edges)***
- Separate structure for *data structure* and *graph structure*.
- Every node has a label (name) and properties (attributes)
- Relationships are edges
- Paths used for traversal in a graph (has start and end node)
- Indexing and node identifier. Each node has unique identifier, in addition user can create indexes for collection of nodes that have a particular label.
- https://console.neo4j.org/

NoSQL Playgrounds

- Mongodb:
  - https://mongoplayground.net/
- Monogodb, Neo4j, Cassandra:
  - https://bitbucket.prodyna.com/projects/NOS/repos/nosql-playground/browse
- Kafka:
  - https://kafka-docker-playground.io/#/
  - https://www.conduktor.io/blog/kafka-playground-two-free-kafka-clusters-without-operational-hassles/
- Redis:
  - https://try.redis.io/
- Neo4j:
  - https://neo4j.com/sandbox/
  - https://console.neo4j.org/