(1) Select and study examples of real-world databases

1. Select the two scenarios "Online Video Streaming - Entertainment" and "eCommerce - Shopping online" for your research.

(2) Recommend a database solution for the selected scenario

1. 在线视频流场景（**Online Video Streaming - Entertainment**）
    - **Data model analysis**: The data stored in this scenario includes user viewing records (such as viewing content, viewing time, pause location, and rewind operation) and massive video content information. Among them, user viewing records are structured data and can be stored through relational databases. Video content information, such as the metadata of the video (title, genre, actor, director, etc.) and the video itself, is more suitable for storage in the form of documents and is semi-structured data.
    - **Database type selection**: For the part of the user's viewing history, a relational database such as MySQL can be used to ensure the consistency and integrity of the data, and facilitate

complex queries, such as querying the user's viewing preferences in a specific time period. For video content information, a document-based database such as MongoDB can flexibly process semi-structured data, facilitate the storage and query of video metadata, and can easily cope with the continuous update and expansion of video content.

- **Database software tools**: MySQL is a mature relational database management system with strong transaction processing capabilities and data integrity assurance, suitable for processing structured data storage and query. MongoDB, on the other hand, provides high performance, scalability, and a flexible data model for storing and processing semi-structured data, such as video metadata.
- **Other considerations**
    - **Data consistency**: MySQL's ACID feature ensures the accuracy and completeness of the user's viewing records and prevents data inconsistencies. MongoDB also guarantees data consistency to some extent, but in a distributed environment additional configuration may be required to ensure strong consistency.
    - **Performance**: MySQL can provide efficient read and write performance through indexing, query optimization, and

other technologies to meet the needs of fast query of users' viewing records. MongoDB's document-based data model has high performance when handling read and write operations on video metadata, especially when the data volume is large and the data structure is not fixed.

- **Security**: Both provide certain security mechanisms, such as user authentication and authorization. MySQL has a mature security system for enterprise-level applications, which can meet the security requirements of online video streaming platforms for user data and business data. MongoDB is also constantly strengthening its security features to ensure the security of data during storage and transmission.
- **Cost**: MySQL is available in open-source versions and is relatively inexpensive, but commercial license fees may need to be considered in large-scale applications. The open-source version of MongoDB also has some cost advantages, while its Enterprise Edition offers more advanced features and technical support at a cost.
- **Community support**: MySQL has a large community with a wealth of documentation, tutorials, and experience sharing, making it easy to find solutions when you

encounter problems. MongoDB's community is also very active, constantly pushing the boundaries of its technology and applications.

2. 电子商务场景（**eCommerce - Shopping online**）
    - **Data model analysis**: The data of e-commerce platforms includes product information (such as name, description, price, inventory, etc.), customer information (such as name, address, contact information, purchase history, etc.) and order information (such as order number, order time, product details, payment status, etc.), most of which are structured data.
    - **Database type selection**: Relational databases are ideal for e-commerce scenarios, such as PostgreSQL. It is able to handle the complex relationships between structured data, ensure the consistency and integrity of data, and meet the needs of efficient storage and query of product information, customer information and order information.
    - **The database software tool** :P ostgreSQL is a powerful, open-source relational database that supports a wide range of data types, complex queries, and transaction processing. It provides high reliability and data integrity assurance, and is suitable for scenarios such as e-commerce platforms that require high data accuracy.

- **Other considerations**
    - **Data consistency**:P ostgreSQL strictly follows the ACID principle to ensure the consistency and correctness of data in multi-user concurrent operations and distributed environments, and to ensure the accuracy of order processing, inventory management and other services.
    - **Performance**: By optimizing query execution plans, indexing, and other technologies, PostgreSQL can provide fast read and write performance to meet the needs of e-commerce platforms for large number of order processing and product queries during peak periods.
    - **Security**:P ostgreSQL provides powerful security features, including user authentication, authorization, encryption, etc., to ensure the security of customer information and transaction data, and prevent data leaks and malicious attacks.
    - **Cost**: Open-source PostgreSQL reduces the initial cost, while its community support and rich plug-in ecosystem can help enterprises reduce development and maintenance costs to a certain extent. For large-scale e-commerce platforms, you may need to consider hardware

    expansion and technical support costs based on actual needs.

  - **Community support**:P ostgreSQL has an active community that provides a large number of documentation, tutorials, and technical support, where enterprises can get solutions to their problems, and at the same time participate in community contributions to promote the development of database technology.

二、MEDIUM TASKS

(1) Comparison between relational databases and non-relational databases

1. **definition**
   - **Relational database**: Built based on the relational model, it uses tables to store data and establishes associations between tables through foreign keys. Data is organized in rows and columns, following ACID properties to ensure data consistency, integrity, and reliability.

- **Non-relational databases**: Do not follow the traditional relational model and use a variety of data models, such as key-value, document, column-family, and graph. It is designed to meet the needs of large-scale, high-concurrency, and flexible data storage, with relatively low data consistency requirements and better scalability.

2. **advantage**
    - **Relational databases**
        - **Strong data consistency and integrity**: ACID transactions and strict schema definitions ensure the accuracy and reliability of data under multi-user concurrent operations.
        - **Suitable for complex queries**: It supports the powerful SQL language, which can easily perform complex operations such as multi-table association queries and aggregate queries, and is suitable for processing data with complex relationships.
        - **Good data standardization**: Standardized design of data is carried out through paradigm theory, which reduces data redundancy and improves data storage efficiency and maintainability.

- **Non-relational databases**
    - **High scalability**: It can easily cope with massive data and high concurrent access, and achieve horizontal expansion through a distributed architecture, such as adding more nodes to a distributed key-value storage system to increase system capacity.
    - **Flexible data model**: You can choose an appropriate data model based on different application scenarios, such as a document database for storing semi-structured data, and a graph database for handling complex relational networks.
    - **Performance advantages**: In specific scenarios, such as frequent reads and writes and simple data structures, non-relational databases can provide higher read/write performance and reduce data processing latency.

3. **limitations**
    - **Relational databases**
        - **Limited scalability**: In the face of large-scale data and high concurrent reads and writes, vertical scaling (adding hardware resources) is expensive and performance improvement is limited, making it difficult to achieve horizontal scaling.

    - **Data model is not flexible enough**: Fixed table structures and schema definitions are not flexible enough when working with semi-structured or unstructured data, and may require complex design and transformation.
    - **Performance impact on complex transactions**: When processing a large number of concurrent transactions, due to transaction consistency requirements, performance degradation may occur, affecting the response speed of the system.
- **Non-relational databases**
    - **Weak data consistency**: Some non-relational databases may not be able to provide strong consistency guarantees in a distributed environment, which may lead to data inconsistencies in the short term, requiring additional processing at the application layer to ensure the eventual consistency of data.
    - **Relatively weak query capabilities**: Unlike relational databases, which support a unified and powerful query language, different types of non-relational databases require their own specific query methods, which are expensive to learn and use.

        - **Lack of mature transaction support**: While some non-relational databases are gradually improving transaction support, the overall ACID transaction support relative to relational databases is not perfect, and it may be difficult to handle complex transaction scenarios.

4. **Software examples**
    - 关系型数据库:MySQL、Oracle、PostgreSQL、SQLite 等。
    - **Non-relational databases**
        - **Key-value pair storage**: Redis, DynamoDB, etc.
        - 文档型存储：MongoDB、CouchDB 等。
        - 列族存储：HBase 等。
        - 图形存储：Neo4j 等。
5. **Usage scenarios**
    - **Relational databases**
        - **Enterprise-level applications**, such as financial systems, customer relationship management systems (CRMs), and enterprise resource planning (ERP) systems, need to deal with complex business logic and strict data consistency requirements.
        - **Scenarios with high data integrity requirements**, such as bank transaction systems, order management on e-

    commerce platforms, etc., ensure the accuracy and reliability of data.

- **Non-relational databases**
    - **Big data processing and analysis**: Scenarios such as log storage, data mining, and real-time data analysis require large-scale, semi-structured, or unstructured data to be processed, requiring high read/write performance and scalability.
    - **Social network and graph data processing**: It is used to store complex relational network data such as user relationships and social graphs, such as Facebook, Twitter and other social platforms use graph databases to manage the relationship between users.
    - **Internet of Things (IoT) data storage**: Processing real-time data generated by large amounts of sensors, such as device status information, environmental monitoring data, etc., requires fast write and query performance, as well as good scalability.

(2) Create a comparison table

Compare items	Relational databases	Non-relational databases
Database structure - the type of data and how it is stored	Based on a table, data is stored in rows and columns, and tables are associated with tables through foreign keys. Ideal for structured data stores.	Multiple data models, such as key-value pairs, documents, column families, graphs, and more. Structured, semi-structured, and unstructured data can be stored, depending on the data model.
Data storage - The amount of data	Large-scale data can be processed effectively within a certain range, but performance bottlenecks can be encountered when scaling at scale.	Designed for large-scale data, it can be easily scaled to store massive amounts of data with a distributed architecture.
Support for ACID transactions (atomicity, consistency, isolation, durability)	Yes, providing strong transaction consistency assurance.	Partial, different types of NoSQL databases are supported to varying degrees, and some may only provide eventual consistency.
Whether normalization is supported	Yes, the data normalization design is carried out through the paradigm theory to reduce redundancy.	Generally, normalization is not emphasized, and the data model is more flexible, but there may be some data redundancy.
integrity constraints; Data accuracy	Provides a variety of integrity constraint	The integrity constraints are relatively weak, and

Compare items	Relational databases	Non-relational databases
	mechanisms, such as primary keys, foreign keys, and unique constraints, to ensure data accuracy.	some databases provide a certain verification mechanism, but mainly rely on the application layer to ensure data accuracy.
Scalability - Horizontal and vertical scaling	Scaling vertically is relatively easy, but scaling horizontally is more complex and requires complex distributed technologies.	It has strong horizontal scalability and can be linearly expanded by adding nodes to adapt to large-scale data and high-concurrency access.
Simplicity: Ease of use, support usability	The SQL language is easy to use, with extensive documentation and community support.	Different types of NoSQL databases have different ease of use, and some have steep learning curves, but community support is evolving.
complexity	The data model is relatively simple, but it can become complex when dealing with complex relationships and distributed transactions.	There are many different data models, each with its own unique complexity, and the overall system architecture can be complex.
cost	Open-source versions such as MySQL and PostgreSQL are	Open-source NoSQL databases, such as MongoDB and Redis, are less expensive, and

Compare items	Relational databases	Non-relational databases
	less expensive, but commercial versions may be more expensive. Hardware, maintenance, and other costs need to be considered.	some commercial products are charged based on functionality. Distributed architectures may require more hardware resources, and cost is impacted by scale.
reliability	High reliability is provided through transaction and log mechanisms, and data is not easy to lose.	Most NoSQL databases provide some reliability through data replication and distributed architectures, but in some cases data inconsistencies can occur.
Mode flexibility	The pattern is fixed, and modifying the mode may require complex operations.	Schema flexibility allows the data structure to be dynamically adjusted based on demand, eliminating the need to pre-define strict schemas.
Performance - Read and Write	The read performance is good, but the write performance may be affected in complex transactions and high concurrency.	The read and write performance varies depending on the data model and application scenarios, and generally has high performance in specific scenarios (such as simple read

Compare items	Relational databases	Non-relational databases
		and write and large-scale write).
Storage requirements	You need to define the table structure and data type in advance, which is highly efficient but may have some redundancy.	