Paper Introduction

This paper introduces the difference between SQL and NoSQL in detail, explains the detailed characteristics of the two, and recommends specific examples in the manual, analysis, design, index, query and other operations are also reflected in the article, and the final database is not only realized The mission requirements. Also added a distributed database to make the results better

SQL And NoSQL

What is a relational database

Relational database (Relational database) is a database created on the basis of a relational model. It uses mathematical concepts and methods such as set algebra to process data in the database. Various entities in the real world and various connections between entities are represented by relational models.

Database: including one or more tables
Table (Relation): a collection of data organized in the form of columns and rows
Column (Attribute): often referred to as a field in the database
Row (value group Tuple): often referred to as records in the database
We can understand it as: Departmental database refers to a database that uses a relational model to organize data. For example, there may be a table named authors (Relation) with information about authors. Each column (value group Tuple) contains a specific type of information, such as the author's last name. Each line (Attribute) contains all the information about a specific author: name, address, etc.

In a relational database, a table is a relationship, and a relational database can contain multiple tables. The main representatives of relational databases: SQL Server, Oracle, MySQL, PostgreSQL.

Advantages of relational databases

Transaction consistency: Maintain data consistency through transaction processing
Complex query: support SQL, can perform complex queries such as JOIN
Easy to understand: The two-dimensional table structure is a concept very close to the logical world, and the relational model is easier to understand than other models such as mesh and hierarchy.
Easy to use: the general SQL language makes it very convenient to operate relational databases
Easy to maintain: Rich integrity (entity integrity, referential integrity and user-defined integrity) greatly reduces the probability of data redundancy and data inconsistency

Disadvantages of relational databases

Read and write performance: When the amount of data reaches a certain scale, because the system logic of the relational database is very complicated, in order to maintain consistency, it is very prone to concurrency problems such as deadlocks, which causes its read and write speed to drop very seriously
Table structure update: The table structure can be updated after it is defined, but it will become more complicated if there are major structural changes
High concurrency: The user concurrency of the website is very high, often reaching tens of thousands of read and write requests per second. For traditional relational databases, hard disk I/O is a big bottleneck
Mass data: For relational databases, querying in a table containing massive data is very inefficient

What is a non-relational database

Non-relational database (NoSQL) is a general term for database management systems that are different from traditional relational databases.

Contemporary typical relational databases exhibit poor performance in some data-sensitive applications, such as indexing a large number of documents, web services for high-traffic websites, and sending streaming media. The typical implementation of a relational database is mainly adjusted to perform small-scale transactions with frequent reads and writes, or large batches with very few write accesses.

The structure of NoSQL usually provides weak consistency guarantees, such as eventual consistency, or transactions are limited to a single data item.

NoSQL proposes another concept. For example, it is stored in key-value pairs and the structure is not fixed. Each tuple can have different fields. Each tuple can add some key-value pairs as needed, so that it will not Limited to a fixed structure, you can reduce some time and space overhead.

There are many significant differences between NoSQL and SQL, the most important of which is that NoSQL does not use SQL as a query language. Its data storage does not require a fixed table mode, and often avoids the use of SQL JOIN operations, which generally have the characteristics of horizontal scalability.

Non-relational databases include:

Temporary key-value storage: memcached, Redis=

Permanent key-value storage: ROMA, Redis

Document-oriented databases: MongoDB, CouchDB

Column-oriented database: Cassandra, HBase

Main representatives: MongoDB, Redis, CouchDB

Advantages of non-relational databases

The emergence of non-relational databases is mostly due to the insufficient performance of relational databases, so the advantages of non-relational databases are also obvious:
Read and write performance: No need to go through the SQL layer analysis, high read and write performance. The main example is Redis. Because of its simple logic and pure memory operation, its performance is very good. A single node can handle more than 100,000 read and write operations per second;
Simple expansion: Based on key-value pairs, data is not coupled and easy to expand. A typical example is Cassandra. Since its architecture is similar to the classic P2P, the cluster can be expanded by easily adding new nodes;
Multiple storage formats: support key-value format, document format, and image format, while relational databases only support basic types;
Low cost: This is a common feature of most distributed databases, because it is mainly open source software, and there is no expensive license cost;

Disadvantages of non-relational databases

Does not provide support for SQL: If an industry standard such as SQL is not supported, a certain amount of learning and application migration costs will be incurred for users
The supported features are not rich enough: the functions provided by existing products are relatively limited. Most NoSQL databases do not support transactions, nor do they provide various additional functions such as BI and reports like MS SQL Server and Oracle.
Existing products are not mature enough: most products are still in their infancy stage, which is different from the decades of perfection of relational databases.

Common NoSQL usage scenarios are as follows:

The database table schema changes frequently, such as online shopping malls. Maintaining product attributes often requires adding fields. This means that the code and configuration of the ORMapping layer need to be changed. If the data volume of the table exceeds one million, adding fields will bring additional overhead. (Rebuild indexes, etc.).

Database table fields are complex data types
For complex data types, such as SQL Sever provides scalability support, like xml type fields. It is difficult for the DB layer to build an efficient index on the xml field, and the application layer needs to analyze and convert from character stream to dom. NoSQL is stored in json format, which provides original ecological support, and is much more efficient and convenient than traditional relational databases.

Highly concurrent database requests
Such applications are common in web2.0 websites. Many applications have very low requirements for data consistency, while the transactions of relational databases and large table JOINs have become "performance killers".

Distributed storage of massive data
If large-scale commercial data such as Oracle is used for the storage of massive data, the cost of the entire solution is very high, and a lot of money is spent on software and hardware. NoSQL distributed storage can be deployed on cheap hardware and is a very cost-effective solution.

The main difference between SQL and NoSQL

1. Storage method:
SQL data is stored in tables with a specific structure; NoSQL is more flexible and extensible, and the storage method can be saved in JSON documents, hash tables or other methods.
SQL usually stores data in the form of database tables. NoSQL storage methods are more flexible, such as using JSON-like file storage.

2. The relationship between the data of the table/data set
In SQL, you must define the table and field structure before you can add data, such as defining the primary key, index, trigger, and stored procedure of the table. The table structure can be updated after it is defined, but it will become more complicated if there are major structural changes.
In NoSQL, data can be added at any time and anywhere without the need to define tables first. NoSQL can also create indexes in the data set. Take MongoDB as an example, it will automatically create a unique value _id field after the data set is created, so that the index can be added after the data set is created.
From this point of view, NoSQL may be more suitable for projects with unclear or undecided initialization data.

3. External data storage
If you need to add external related data in SQL, the standardized approach is to add a foreign key to the original table to associate the external data table. For example, it is necessary to add reviewer information to the borrowing table, and first establish an reviewer table, so that if we need to update the personal information of the reviewer, we only need to update the reviewer table without updating the borrower table.

In NoSQL, in addition to this standardized external data table approach, we can also use the following non-standardized methods to put external data directly into the original data set to improve query efficiency. The disadvantages are also obvious, and it will be more troublesome to update the reviewer's data.

4. JOIN query in SQL
In SQL, you can use the JOIN table link method to query the data in multiple relational data tables with a simple query statement. NoSQL currently does not provide a JOIN-like query method to query data in multiple data sets. Therefore, most NoSQL uses non-standardized data storage methods to store data.

5. Data coupling
It is not allowed to delete external data that has been used in SQL. For example, "Xiong San" in the reviewer table has been assigned to the borrower Xiongda, then the data of Xiong San will not be allowed to be deleted in the reviewer table to ensure Data integrity. In NoSQL, there is no such strong coupling concept, and any data can be deleted at any time.

6. Affairs
In SQL, if the data of multiple tables need to be updated in the same batch, that is, if one of the tables fails to update, the other tables cannot be updated successfully. This scenario can be controlled by transactions, and the transaction can be submitted uniformly after all commands are completed. There is no concept of transaction in NoSQL, and the operation of each data set is atomic.
7. Add, delete, modify and check grammar
8. Query performance
Under the premise of the same level of system design,because the consumption of JOIN queries is omitted in NoSQL, the theoretical performance is better than SQL.

Indexes

What is an index

The database index is like a table of contents in front of a book, which can speed up the query speed of the database. An index is a structure that sorts the values of one or more columns in a database table. If you want to find a specific employee by his or her last name, the index helps to obtain information faster than searching all rows in the table

Common query algorithms: sequential search, binary search, binary sort tree search, hash hash method, block search, balanced multi-way search tree B-tree (B-tree)

Indexes are implemented at the storage engine layer, not at the server layer, so different storage engines have different index types and implementations.

Advantages and disadvantages of indexing

advantage:
1. Greatly speed up data retrieval;

2. Create a unique index to ensure the uniqueness of each row of data in the database table;
3. The connection between the accelerometer and the table;
4. When using grouping and sorting clauses for data retrieval, the time for grouping and sorting in the query can be significantly reduced.

Disadvantages:

1. The index needs to occupy physical storage space outside the data table
2. It takes a certain amount of time to create and maintain indexes
3. When updating the table, the index needs to be rebuilt, which reduces the speed of data maintenance.

Index type:

According to the function of the database, you can create indexes in the database designer: unique index, primary key index and clustered index. Although unique indexes help locate information, for best performance results, it is recommended to use primary keys or unique constraints instead.

Unique index: UNIQUE For example: create unique index stusno on student (sno);
Indicates that each index value of this index only corresponds to a unique data record. For a single-column unique index, this ensures that a single column does not contain duplicate values. For a multi-column unique index, ensure that the combination of multiple values is not repeated.

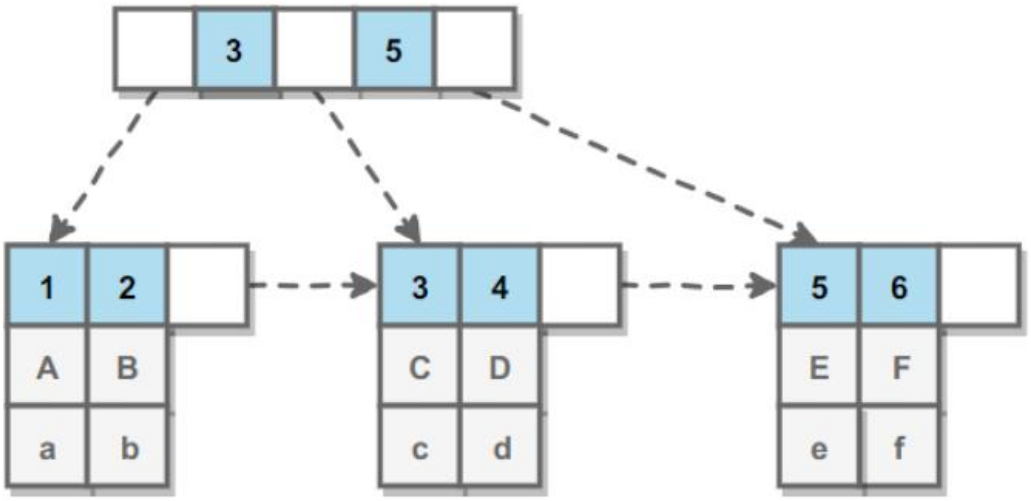
Primary key index:
Database tables often have a column or combination of columns whose value uniquely identifies each row in the table. This column is called the primary key of the table. Defining a primary key for a table in the database diagram will automatically create a primary key index, which is a specific type of unique index. The index requires that each value in the primary key is unique. When the primary key index is used in the query, it also allows quick access to the data.

Clustered index (also called clustered index):
In a clustered index, the physical order of rows in the table is the same as the logical (index) order of the key values. A table can only contain one clustered index. If an index is not a clustered index, the physical order of the rows in the table does not match the logical order of the key values. Compared with non-clustered indexes, clustered indexes usually provide faster data access speed.

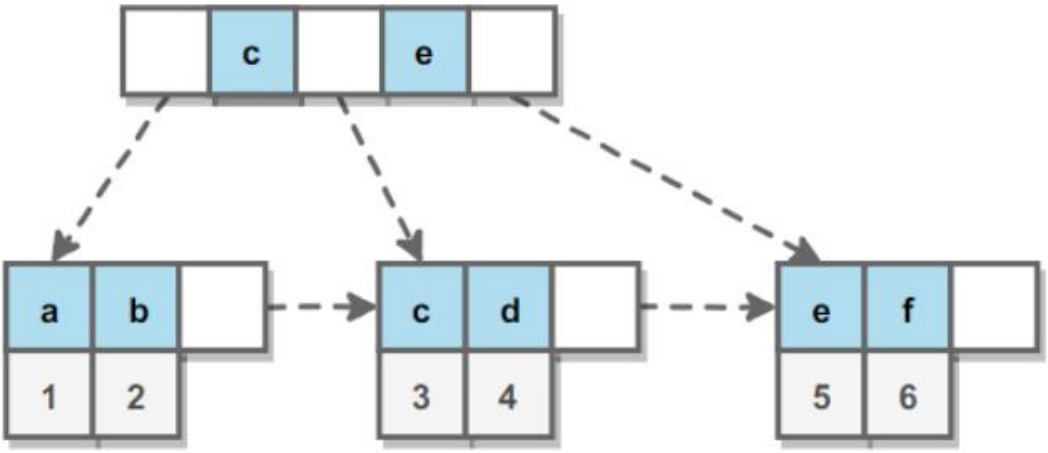
How the index is implemented

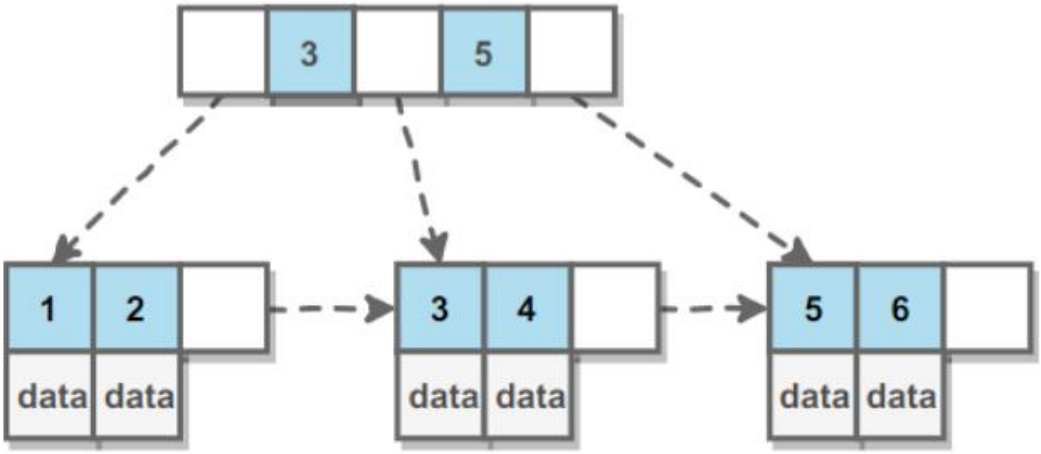
1. B+Tree index
The default creation of MyISAM and InnoDB storage engines is B+tree index, Memory storage engine can also be B+tree index (or Hash index)
InnoDB's B+Tree index is divided into main index and auxiliary index.
The data field of the leaf node of the main index records complete data records. This index

method is called a clustered index. Because data rows cannot be stored in two different places, a table can only have one clustered index



The data field of the leaf node of the auxiliary index records the value of the primary key, so when using the auxiliary index to search, you need to find the primary key value first, and then search in the main index





2. Hash Index

The hash index can be searched in O(1) time, but it loses the orderliness and cannot be used for

sorting and grouping. It only supports precise search and cannot be used for partial search and range search.

InnoDB storage engine has a special function called "adaptive hash index". When an index value is used very frequently, a hash index will be created on top of the B+Tree index, so that B+Tree Index has some advantages of hash index, such as fast hash lookup

3. Full-text index
The MyISAM storage engine supports full-text indexing to find keywords in the text, rather than directly comparing whether they are equal.

The search condition uses MATCH AGAINST instead of ordinary WHERE. Full-text indexing is implemented using an inverted index, which records the mapping of keywords to the documents where they are located. InnoDB storage engine also began to support full-text indexing in MySQL 5.6.4

4. Spatial Data Index
The MyISAM storage engine supports spatial data index (R-Tree), which can be used for geographic data storage. The spatial data index will index data from all dimensions, and any dimension can be effectively used for combined queries. Must use GIS related functions to maintain data.

Transaction

Database creates a ticket_ID Theatre_ Repertaire_ id, Theatre_ id
Customer_ID_card, ticket_seat_number, ticket_cost and other data tables have been set up to use different primary and foreign keys for each table, and a relationship has been established. Between each table. Set the ticket ID of the ticket table to this primary key table. In this table, there are other information, such as the contrast ID of the theater, theater ID, theater seat number, consumer ID card, ticket price, etc. In the data query, you can enter key information such as seats, ticket prices and theater names. By discovering the ticket table, you can then query the information created by each table through detailed information.

Database transactions usually include a series of reads or read operations
Write to the database. It serves two purposes
1. It provides a way to recover from database failure to normal state.
Operation, also provides a way to maintain database consistency
In an abnormal state. 2. When multiple applications access the database at the same time,
Can provide isolation between these applications to prevent them
Operations that interfere with each other. Database transactions can solve the following two problems: 1. Sometimes
The reservation of the venue is recorded, but the details of the specific date

The requirements of the production company are not registered in the system. 2.
Customers can book tickets that are not registered as events.
If the details of a specific date are not recorded in the system, then
Submitting is not allowed for a long time.
If there is no relevant ticket, the consumer cannot book a ticket.

Distributed Database Solution

Establishment of distributed database

1.M_ Space table is the basic table, which is used to store theater space information, and can be distributed horizontally. Conditions: distinguish different specifications of the theater
SELECT * FROM M_ SPACE WHERE Type = 'B' SELECT * FROM M_ SPACE WHERE Type = 'S'

2.T_ Book is a company booking table, which can be vertically distributed. Column 1:
Store company information and company booking time
BOOK_ NO|BOOK_ NAME|BOOK_ TIME_ FROM|BOOK_ TIME_ TO
Column 2: store company booking information and fees
BOOK_ NO |THEATRE_ NO|THEATRE_ Type|COST
Column 3: store the ticket information corresponding to the theater reserved by the company
BOOK_ NO |TICKET

3.T_ BUY_ Ticket is a ticket table, which can be distributed horizontally. Conditions: distinguish the number of tickets purchased
SELECT * FROM T_ BUY_ TICKET WHERE NUM >5
SELECT * FROM T_ BUY_ TICKET WHERE NUM <=5

4.T_ Ticket is the activity ticket price table, which can be distributed horizontally. Conditions: distinguish the professional level of the performance
select * from T_ TICKET where PRO_ LEVEL ='A' select * from T_ TICKET where PRO_ LEVEL ='P'

Reconstruction of distributed database

1.SELECT * FROM M_ SPACE WHERE Type = 'B' union SELECT * FROM M_ SPACE
WHERE Type = 'S' select
a.BOOK_ NO
,a.BOOK_ NAME
,a.BOOK_ TIME_ FROM
,a.BOOK_ TIME_ TO
,b.THEATRE_ NO
,b.THEATRE_ Type
,b.COST

```
,c.TICKET
from T_ BOOK a
full outer join T_ BOOK b on a.BOOK_ NO=b.BOOK_ NO
full outer join T_ BOOK c on a.BOOK_ NO=c.BOOK_ NO
```

2. SELECT * FROM T_ BUY_ TICKET WHERE NUM >5 union SELECT * FROM T_ BUY_ TICKET WHERE NUM <=5

3.select * from T_ TICKET where PRO_ LEVEL ='A' union select * from T_ TICKET where PRO_ LEVEL ='P'