Relational and NoSQL Databases

HAYDEN VASS | DVP 2
E: HAVASS@STUDENT.FULLSAIL.EDU

MOBILE DEVELOPMENT
FULL SAIL UNIVERSITY

Difference between relational and noSQL.

RELATIONAL VS NOSQL

SECTION 1

DIFFRENCES

A relational database is a collection of information stored in tables and specific columns within those tables that scales vertically. With similar information formatted in a uniform manner, data can be stored, optimized, compared and shared quickly and more efficiently. Meanwhile, the ability to store data vertically means that you can expand the databases capabilities by improving its hardware. With relational databases, a database administrator can formulate unique information by gathering details from a particular column and use any number of functions to generate, merge, or compare data from other tables. The ability to store, track and manipulate very specific data makes relational databases a powerful and dynamic tool. Some pros to relational databases include its powerful and easy to understand query language.

SQL language has been around for a while and is well known. Not to mention it's easy to learn. Relational databases are also capable of handling large amounts of data in a single query. The SQL language allows the user to tailor their queries to encompass almost all the information they need. Though the pros can be enticing, relational databases do have their down sides. They tend to predefine their data, which makes the extremely rigid. SQL also does not use a hierarchical method to store data. This means a DBA might have to use extra steps when adding additional data. Another down side to relational databases its inability to convert and store custom data types from programming languages.

As a relational database implements a series of tables to store information, a non-relational database does not. Non-relational databases have grown more popular in recent years as the limitations of relational databases become more exposed. While relational databases are great for storing and tracking constant data, they are rigid in nature. While SQL databases can be scaled by strengthening its hardware, NoSQL databases can be enhanced by adding additional servers to the collection. This allows information to be stored in a more efficient manner. This also makes it easier for NoSQL databases to store vast and ever changing data.

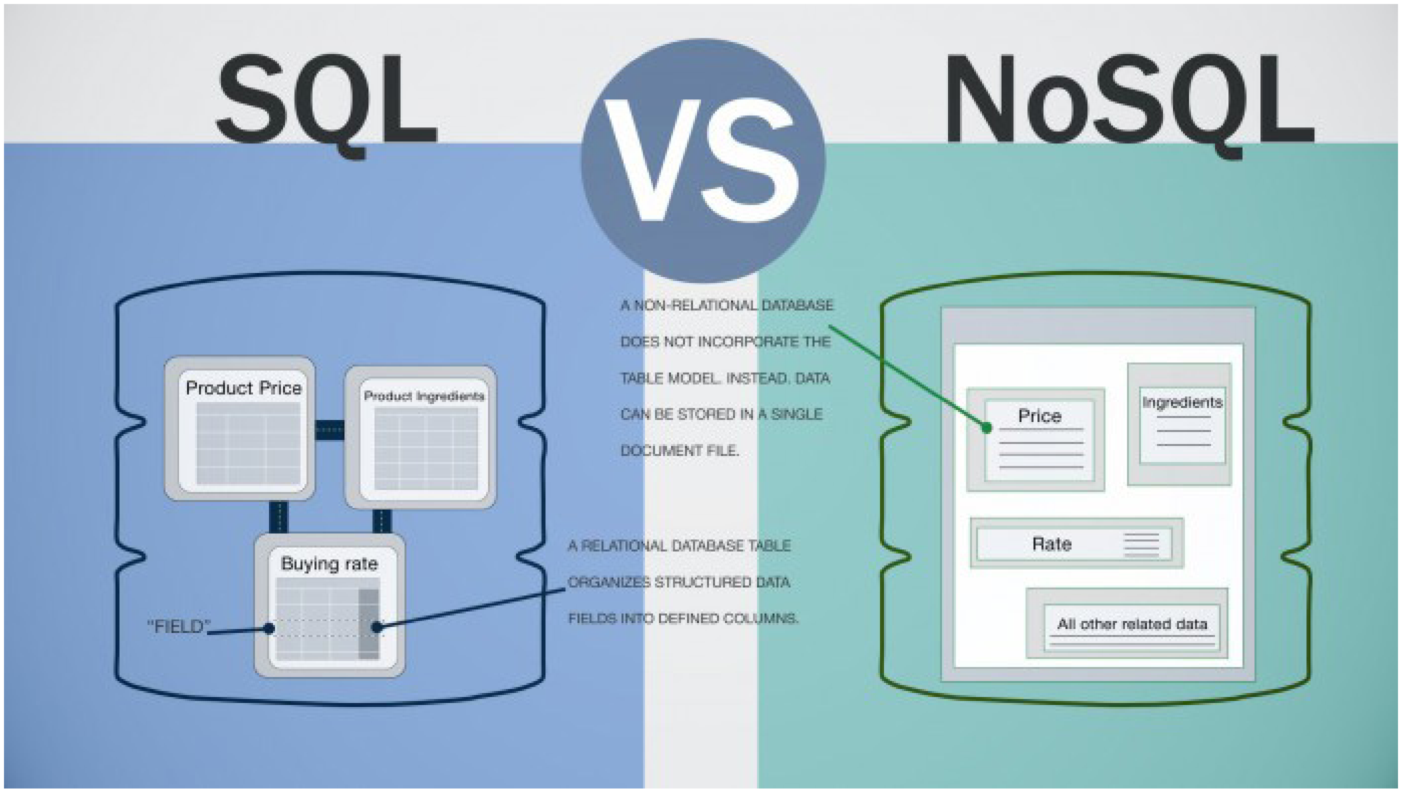
Reversely, non-relational databases are typically built for specific data models and can be extremely flexible and scale horizontally. Non-relational databases can use a variety of data models including graph, document and key-value.

RELATIONAL VS NOSQL

SECTION 1

DIFFRENCES

The pros to using a non- relational data base include its flexible schemas, which allow for faster and more hands on development. They are also have high performance. Since they are often built for a specific data model they can access and distinguish that data at an increased speed. Some disadvantages of non-relational databases usually stem from its relative age. Many non-relative databases have less support and are less mature than some of the older relational databases put out by major corporations.


SQL
VS
NoSQL
Product Price
Product Ingredients
Buying rate
"FIELD"
A NON-RELATIONAL DATABASE DOES NOT INCORPORATE THE TABLE MODEL. INSTEAD, DATA CAN BE STORED IN A SINGLE DOCUMENT FILE.
A RELATIONAL DATABASE TABLE ORGANIZES STRUCTURED DATA FIELDS INTO DEFINED COLUMNS.
Price
Ingredients
Rate
All other related data

NoSQL Twitter Features

TWITTER FEATURES

THREE FEATURES

One features of Twitter that could be utilized in a NoSQL database is the actual ability to tweet. NoSQL databases have tremendous writing performance. Tweets could easily be taken from the user, formatted to fit a specific data model, and written to the database. With Twitter users posting nearly 500 million tweets a day, a traditional SQL database would struggle to keep up with that kind of data demand. On the other hand, a NoSQL database could have dedicated clusters to store information based on whatever Twitter wanted to define. Storing data in this method would also allow Twitter to mine more meaningful data from peoples tweets. Whereas in a relational database, schemas would have to be predefined. This doesn’t necessarily limit the amount data that it can store, but it does limit how specific the data can be.

Another Twitter feature that would thrive in a NoSQL database is the ability to save users tweets. Seeing as NoSQL databases do not have predefined schemas, Twitter would have an easier time storing and tracking specific data being saved. This data could then be used to craft a better user experience by suggesting related material similar to that the user saved. Another benefit of using a NoSQL server in this way is an increased capability to track trends. As trends change, the content of what people are saving would change. Utilizing a NoSQL database, a database administrator can easily add new fields to the database. To do this in a relational database would require significant amount of resources to alter the database, most likely being taken offline at the same time.

The last feature twitter would most likely handle with a NoSQL database is the search function. With a traditional relational database, everything is stored in that one database. With a NoSQL database, various data can be stored in different clusters. This would allow the users search parameters to processed and returned in a fraction of the time. For example if a user wanted to search a specific hashtag, a relational database would have to first sort through all of twitters data to find tables that contain hashtags, then sort through all of that data, then eventually waterfall down into what the user is looking for. In a NoSQL database, the request could simply be directed towards a server cluster specifically designed to house the type of information the user is looking for.

TWITTER FEATURES

PROS / CONS

Pros of tweeting and NoSQL:
- Large amounts of tweets occur every day. NoSQL is suited for that kind of workload.
- dedicated server clusters for different data
- flexible and easily scaled

Cons of tweeting and NoSQL:
- Can have questionable consistency at times
- Being dynamic is great, but could possibly lead to redundant data. For example, creating a new field for data that might already exist in another field.

Pros of saving and NoSQL:
- Can easily define trends.
- Ability provide a more tailored user experience since more specific data can be stored.
- Cheaper to scale.

Cons of saving and NoSQL:
- Could possibly store vast amounts of useless data

Pros of searching and NoSQL:
- Faster searches because of sharding databases
- Cheaper to scale.

Cons of searching and NoSQL:
- More assets to manage.
- The search might have to filter through more data due to the fact more specific information can be stored in NoSQL.

SQL Facebook Feature

FACEBOOK FEATURE

ONE FEATURE

One feature of Facebook that could be found in a relational database is the “groups” function. Groups could be contained in their own sub table holding information on group members, hierarchy, age, address, etc. A relational database would work in this instance because regardless of who joins or leaves the group they will all share the same information. This information is required when joining Facebook, enforcing the fact each user has this data attached to their profile. Facebook groups also generally focus around particular topics. This means that post made to the group can be saved with a general idea of what kind of data it contains.

PROS / CONS

Pros:

- SQL scales vertically, which fits how Facebook groups scale.
- Easy to track growth and activity.

Cons:

- Rigid data tracking, especially if group topics change constantly.

NoSQL Databases

NOSQL DATABASES

POPULAR DATABASES

1. MongoDB
Mongo is probably the most well known and most popular non-relational database on the market at the moment. It utilizes documents as its datatype and stores JSON files, much as relational database stores tables. The strength of Mongo comes from its ability to convert the JSON files and use them inside of an IDE. Mongo is fast and scales well, and is consistent. One of the downsides to Mongo is its limitation on document size, which is currently 16mb. It supports a variety of different programming languages such as Java, Python and C++.

2. Neo4j
Neo4j is a graph based non-relational database written in Java. Though it represents its data in graph format, it stores its data as key value pairs. These pairs are then combined together in a graph. This way of representing and storing data gives Neo4J an edge when working with complex data models containing a lot of unique data between objects. Though Neo4J is great at navigating through data, it does clock in slower than other databases when dealing with singular or simple data request.

3. Cassandra
Cassandra is a column based database developed by Apache. Compared to most other non-relational data bases, Cassandra's query language and data is most similar to traditional SQL databases. Instead of storing information in columns, Cassandra stores its data in rows. This allows Cassandra to manipulate data in various rows without having to join them together or requiring other rows to share the same column. Cassandra does exceptionally well storing large amounts of data as well as writing data to the database. It does fall short when it comes to scalability and when dealing with complex data that spans over multiple tables.

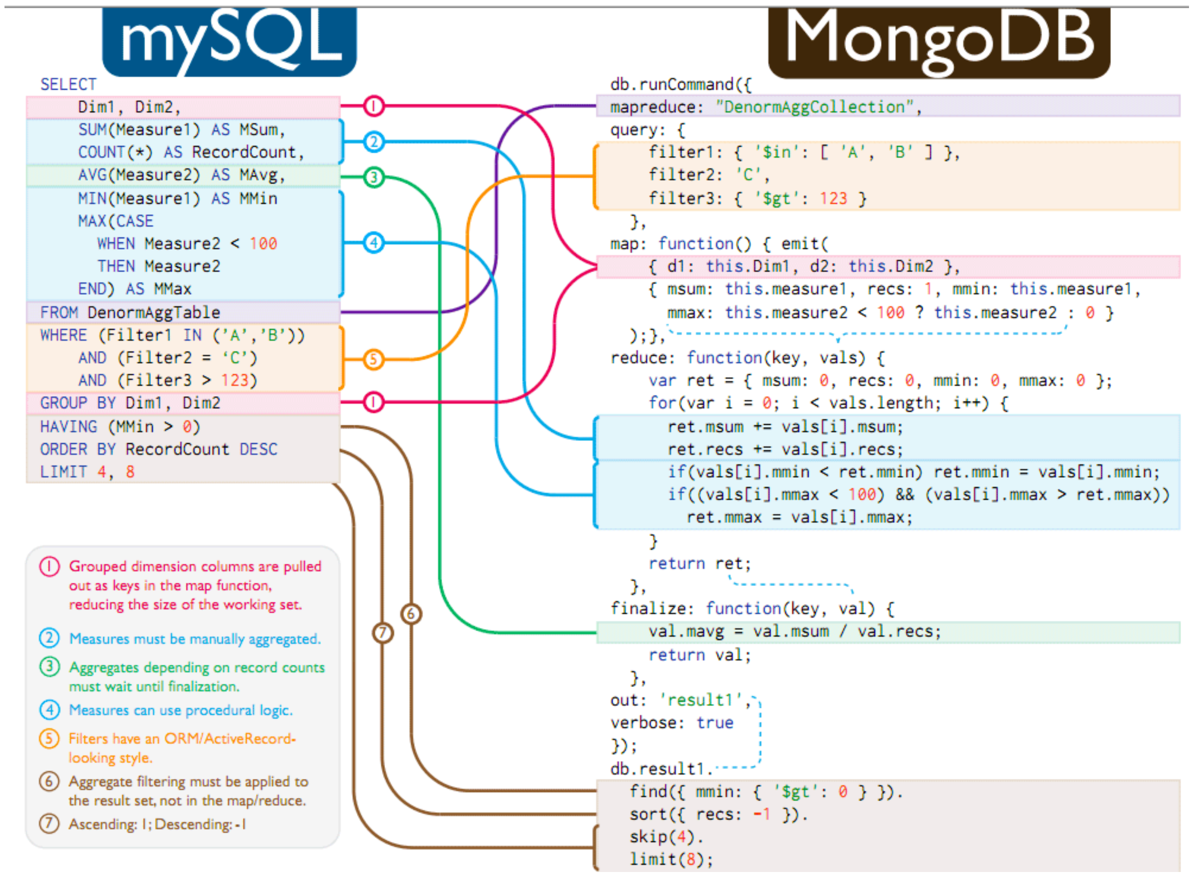
4. Oracle NoSQL
Much like Neo4J, Oracle NoSQL stores its data in key value pairs, but does not represent its data in graph format. Oracle NoSQL utilizes the idea of a "master" key. This master key links all the children keys together. Much like key value pairs in C# does. As one would expect from a product put out by Oracle, the support provided to ONoSQL is provided at an enterprise level. It stores and processes large amounts of data with ease, and can process singular pieces of simple data lighting fast. The downside to Oracle NoSQL is its inability to read

NOSQL DATABASES

POPULAR DATABASES

write to the data base in an efficient manner. When it comes to this aspect of ONoSQL, its performance is much less of that of Oracles standard SQL database.

5. Redis
Redis stands for remote dictionary server is a blazing fast NoSQL database currently being used by big names such as Snapchat, GitHub StackOverflow and Pinterest. It supports a variety of data structures like strings, hashes and list. Redis also supports a variety of coding languages like C, C++, C#, Phython, Objective – C and PHP. What sets Redis apart is its ability to work with an in-memory dataset and its ability to manipulate complex data types.



```
SELECT
    Dim1, Dim2,
    SUM(Measure1) AS MSum,
    COUNT(*) AS RecordCount,
    AVG(Measure2) AS MAvg,
    MIN(Measure1) AS MMin
    MAX(CASE
      WHEN Measure2 < 100
      THEN Measure2
    END) AS MMax
FROM DenormAggTable
WHERE (Filter1 IN ('A','B'))
    AND (Filter2 = 'C')
    AND (Filter3 > 123)
GROUP BY Dim1, Dim2
HAVING (MMin > 0)
ORDER BY RecordCount DESC
LIMIT 4, 8
```

```
db.runCommand({
mapreduce: "DenormAggCollection",
query: {
    filter1: { '$in': [ 'A', 'B' ] },
    filter2: 'C',
    filter3: { '$gt': 123 }
  },
map: function() { emit(
    { d1: this.Dim1, d2: this.Dim2 },
    { msum: this.measure1, recs: 1, mmin: this.measure1,
      mmax: this.measure2 < 100 ? this.measure2 : 0 }
  );},
reduce: function(key, vals) {
    var ret = { msum: 0, recs: 0, mmin: 0, mmax: 0 };
    for(var i = 0; i < vals.length; i++) {
      ret.msum += vals[i].msum;
      ret.recs += vals[i].recs;
      if(vals[i].mmin < ret.mmin) ret.mmin = vals[i].mmin;
      if((vals[i].mmax < 100) && (vals[i].mmax > ret.mmax))
        ret.mmax = vals[i].mmax;
    }
    return ret;
  },
finalize: function(key, val) {
    val.mavg = val.msum / val.recs;
    return val;
  },
out: 'result1',
verbose: true
});
db.result1.
  find({ mmin: { '$gt': 0 } }).
  sort({ recs: -1 }).
  skip(4).
  limit(8);
```

① Grouped dimension columns are pulled out as keys in the map function, reducing the size of the working set.
② Measures must be manually aggregated.
③ Aggregates depending on record counts must wait until finalization.
④ Measures can use procedural logic.
⑤ Filters have an ORM/ActiveRecord-looking style.
⑥ Aggregate filtering must be applied to the result set, not in the map/reduce.
⑦ Ascending: 1; Descending: -1

NOSQL DATABASES

PROS / CONS

MongoDB:
Pros:
- Document Validation
- Intergraded storage engines

Cons:
- Doesn’t work will with applications that use complex transactions
- Does not work well with data from older databases.

Neo4J:
Pros:
- Fast queries when defining relationships between nodes.
- Easy to modify and change.

Cons:
- Not intended for task that involve simple relationships.
- Probably requires the user to learn a new language.
- Not a lot of support available.

Cassandra:
Pros:
- Highly Scalable.
- Fast write output/ decent read input.
- Flexible.

Cons:
- Limited or no use of aggregate functions
- Unreliable.

Oracle NoSQL
Pros:
- Scales well.
- Handles big data efficiently.
- Flexible data models.

Cons:
- Small community base
- Not many experts exist, making trouble shooting difficult.

NOSQL DATABASES

PROS / CONS

Redis
Pros:
- Fast.
- Supports a variety of data types.

Cons:
- Requires knowledge of lua to learn every aspect of Redis.
- Heavy resource consumption.

NoSQL Weather App

WEATHER APP

FUNCTIONALITY

The most functional NoSQL solutions for a weather application would be Redis and Neo4j. Redis is a key-value based NoSQL server, which would be optimal for storing information such as city, state, and weather patterns most common in those areas. The fact that it can also handle various data types would mean it would function well with Neo4j, a graph based solution that also uses key value pairs. Neo4J would be used to handle more complex data relationships associated with weather. Such as what are the specific effects on a region pending different atmospheric and other weather conditions. This data could then be analyzed and passed to the user to give the most accurate forecast.

Refrences

REFERENCES

The Key Differences Between SQL and NoSQL database. (2018). Agira Technologies. Retrieved 31 August 2018, from http://www.agiratech.com/the-key-differences-between-sql-and-nosql-database/

Twitter Usage Statistics - Internet Live Stats. (2011). Internetlivestats.com. Retrieved 1 September 2018, from http://www.internetlivestats.com/twitter-statistics/

People Search Import user. (2018). Slideshare.net. Retrieved 2 September 2018, from https://www.slideshare.net/kevinweil/nosql-at-twitter-nosql-eu-2010/110-People_Search_Import_user_data

The 5 Features to Look for in a NoSQL Database. (2017). DataStax: always-on data platform | NoSQL | Apache Cassandra. Retrieved 2 September 2018, from https://www.datastax.com/2017/09/the-5-features-to-look-for-in-a-nosql-database

Why NoSQL Database | Couchbase. (2018). Couchbase.com. Retrieved 2 September 2018, from https://www.couchbase.com/resources/why-nosql

MySQL vs. MongoDB: The Pros and Cons When Building a Social Network. (2015). Morpheus. Retrieved 2 September 2018, from https://www.morpheusdata.com/blog/2015-04-01-mysql-vs-mongodb-the-pros-and-cons-when-building-a-social-network

Redis Explained in 5 Minutes or Less - Credera. (2014). Credera. Retrieved 2 September 2018, from https://www.credera.com/blog/technology-insights/java/redis-explained-5-minutes-less/

Redis. (2018). Redis.io. Retrieved 2 September 2018, from https://redis.io/

Redis: What and Why? – codeburst. (2018). codeburst. Retrieved 2 September 2018, from https://codeburst.io/redis-what-and-why-d52b6829813

Garden, H., & Software, C. (2001). What are relational databases?. HowStuffWorks.

REFERENCES

Retrieved 28
August 2018, from https://computer.howstuffworks.com/question599.htm
Advantages of Relational Databases. (2018). Techspirited. Retrieved 28 August 2018, from https://techspirited.com/advantages-of-relational-databases

Top 5 SQL Databases [Infographic] - DZone Database. (2018). dzone.com. Retrieved 28 August 2018, from https://dzone.com/articles/top-5-sql-databases
To SQL or not to SQL. (2017). Capgemini Engineering. Retrieved 28 August 2018, from https://capgemini.github.io/design/sql-vs-nosql/

Farrar, G. (2016). The Advantages and Challenges of using Microsoft SQL Server (MSSQL). Turbonomic. Retrieved 28 August 2018, from https://turbonomic.com/blog/microsoft-sql- server-advantages-challenges-virtualization-admins- view/?utm_referrer=https%3A%2F%2Fwww.google.com%2F

The Pros and Cons of 8 Popular Databases. (2017). KeyCDN Blog. Retrieved 28 August 2018, from https://www.keycdn.com/blog/popular-databases/

What is DB2? - Definition from Techopedia. (2018). Techopedia.com. Retrieved 28 August 2018, from https://www.techopedia.com/definition/24360/db2

The Advantages of DB2 | Techwalla.com. (2018). Techwalla. Retrieved 28 August 2018, from https://www.techwalla.com/articles/the-advantages-of-db2

What Is A Non Relational Database. (2018). MongoDB. Retrieved 28 August 2018, from https://www.mongodb.com/scale/what-is-a-non-relational-database

What is NoSQL? | Nonrelational Databases, Flexible Schema Data Models | AWS. (2018). Amazon Web Services, Inc.. Retrieved 28 August 2018, from https://aws.amazon.com/nosql/
MongoDB vs MySQL Comparison: Which Database is Better?. (2017). Hacker Noon. Retrieved 28 August 2018, from https://hackernoon.com/mongodb-vs-mysql-comparison-which-database-is- better-e714b699c38b

REFERENCES

source, O. (2018). Consider the Apache Cassandra database. Ibm.com. Retrieved 28 August 2018, from https://www.ibm.com/developerworks/library/os-apache-cassandra/index.html
10 things you should know about NoSQL databases. (2018). TechRepublic. Retrieved 28 August 2018, from https://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql- databases/

Images

NOSQL · Build web application with Golang. (2018). Astaxie.gitbooks.io. Retrieved 4 September 2018, from https://astaxie.gitbooks.io/build-web-application-with-golang/en/05.6.html

Relational (SQL) vs NoSQL Database Models. (2018). ao. Retrieved 4 September 2018, from https://ao.gl/relational-sql-vs-nosql-database-models/

HAYDEN VASS | DVP 2
E: HAVASS@STUDENT.FULLSAIL.EDU