Introduktion till NoSQL

Not only SQL

www.everyloop.com

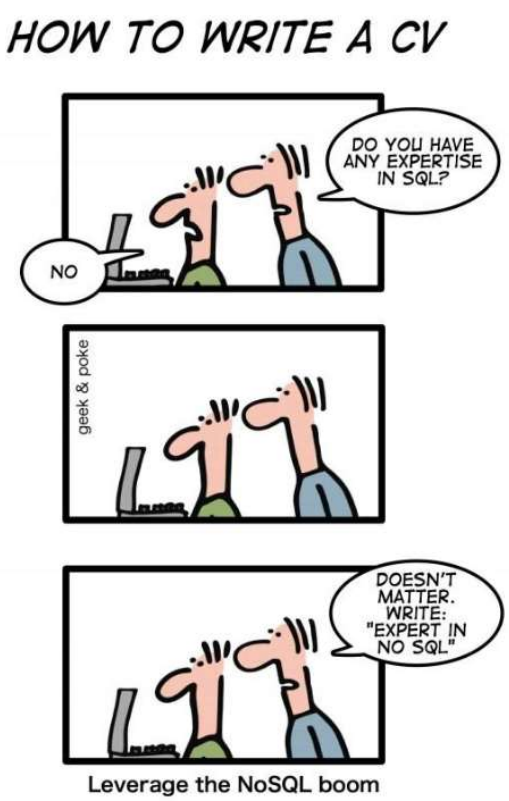
RDBMS

Relationsdatabaser lagrar data i tabeller och använder frågespråket SQL för att manipulera data.

Till höger ser vi några av de största och vanligaste RDBMS.

De olika systemen och SQL-dialekterna skiljer något, men är på det stora hela ganska lika.

NoSQL



Begreppet NoSQL uppkom i början av 2000-talet, och det som åsyftas är kort och gott databaser som är uppbyggda på andra sätt än de klassiska relationsdatabaserna.

Sådana databaser har funnits sedan 60-talet, men blev populära först då nya internettjänster som google och facebook växte fram, med behov av ”nya” arbetssätt för att lagra och analysera data.

Det finns olika typer av NoSQL-databaser, men gemensamt är att de inte lagrar data i tabellform, och inte använder standardspråket SQL.

Avvägningar

NoSQL-databaser har utvecklats för att lösa problem som varit svåra att lösa med traditionella RDBMS. De flesta NoSQL-system bygger på ”key-value”-konceptet, vilket gör de lättare att skala horisontellt.

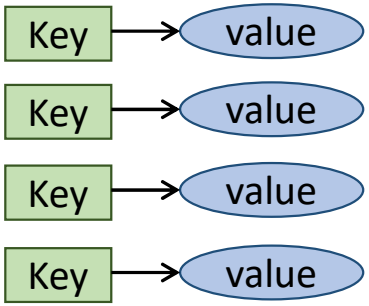
NoSQL är byggt för hög prestanda och skalbarhet, och är både snabbare och bättre på att hantera stora datamängder (Big data).

För att åstadkomma detta har man offrat dataintegritet och ACID-transaktioner. Det är även svårare att skriva komplexa queries, och använda normaliserad data med relationer.

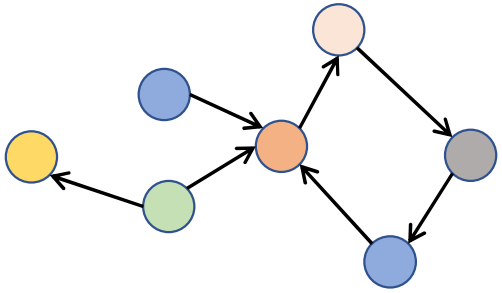
Detta är både en för- och nackdel. Oftast pratar man i NoSQL sammanhang om att denormalisera sin data, dvs. offra dataintegritet för att vinna prestanda.

Olika typer av NoSQL databaser

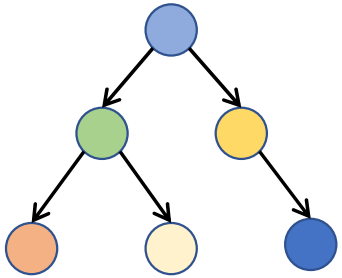
Key-Value



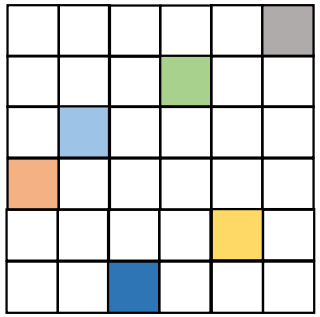
Graph



Document



Column-Family

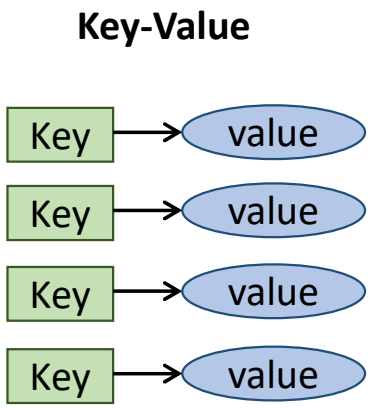


Key-Value stores

Key-Value stores är den enklaste formen av NoSQL-databas.

Varje post i databasen lagras som en nyckel (namn) tillsammans med sitt värde. All nycklar måste vara unika då de används för att komma åt ett visst värde.

I programmering brukar denna typ av datastruktur kallas dictionary eller map.



redis

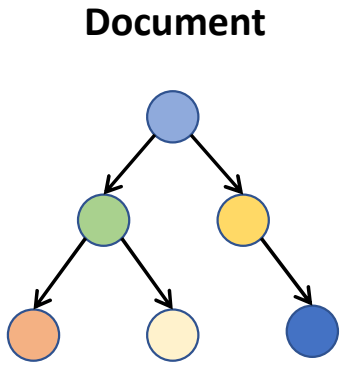
Azure Cosmos DB

Amazon DynamoDB

Document stores

Den mest använda typen av NoSQL-databas, framför allt genom MongoDB som blivit väldigt populär.

DBMS av detta slag samlar semi-strukturerad data i form av dokument som kan sägas motsvara rader i RDBMS, med skillnaden att strukturen på de olika dokumenten kan skilja inom en samling.



Amazon DynamoDB

Couchbase

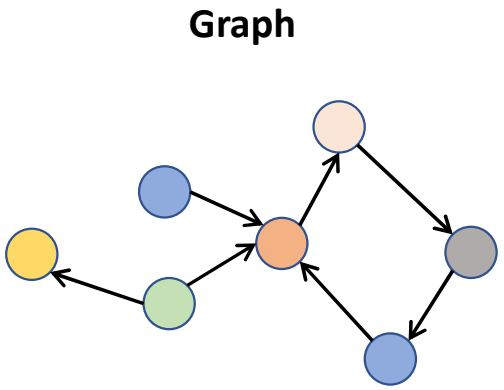
mongoDB

Graph DBMS

En graf i en grafdatabas är baserad på grafteori, och består av noder sammanbundna av bågar.

Varje nod kan ha ett flertal egenskaper (data-punkter) och bågarna representerar nodernas relation till varandra.

Grafdatabaser används till fördel när det är relationerna mellan de enskilda punkterna som är det viktiga (som i olika former av nätverk), och kan svara på frågor som kan vara svåra att verkställa i en relationsdatabas.



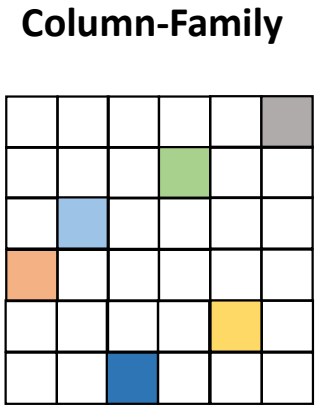
neo4j

The #1 Database for Connected Data

Wide column stores

Denna typ av NoSQL-databas påminner genom sin tabellstruktur en del om RDBMS. Skillnaden är dock att de inte lagrar data radvis, utan kolumnvis, vilket gör att man snabbt kan genomsöka enskilda kolumner.

Andra fördelar är att man columnerna kan skilja från rad till rad, samt att varje rad kan innehålla ett mycket stort antal kolumner.



SQL vs NoSQL

Generella aspekter

	SQL	**NoSQL**
Arkitektur	Centraliserad	Distribuerad
Datastruktur	Strukturerad data	Semi-strukturerad data
Normalisering	Normaliserad data	Denormaliserad data
Schema	Fast schema	Polymorfiskt schema
Dataintegritet	Hög	Låg
Transaktioner	ACID	Nej
Skalbarhet	Vertikal	Horisontell
Utvecklingstyp	Mindre flexibel, längre iterrationer	Agil, snabba iterrationer
Språk	SQL	Systemspecifik
Alternativ åtkomst	ORM-tools	REST-API

När använder man vad?

SQL	NOSQL
• När man jobbar med komplexa queries och avancerade data-analyser, eftersom komplexa queries mot flera tabeller är snabbare i SQL. • När man har en hög transaktionvolym eller komplexa transaktioner, eftersom SQL är mer stabilt och garanterar dataintegritet. • Om man är I behov av ACID, eller behöver definera exakt hur transaktioner påverkar database. • När man inte förväntar sig snabba förändringar i utvecklingen, eller inte jobbar med stora mängder data.	• När det är svårt att förutse hur din applikation växer över tid för att du ständigt lägger till nya funktioner. • När kontinuitet och 100% dataintegritet inte är din främsta prioritet. • När du har mycket data, många olika typer av data, och data kommer växa fort framöver. • När du behöver ett flexibelt system som är lätt att skala upp efterhand. • När din datamodell passar systemet.

Välja rätt!

I de allra flesta fall är RDBMS att föredra, framförallt i små och medelstora projekt, och där man inte vet i förväg hur data kommer efterfrågas.

Även i stora projekt med höga datavolymer, men där man har ett behov av komplexa frågor och relationer används RDBMS med fördel, dock eventuellt med så kallade read replicas för högre prestanda.

I stora projekt med höga krav på prestanda (läs: Big data) kan man behöva offra dataintegritet för snabbare behandling. De flesta utvecklare jobbar dock inte med Big data, så endast om datamodellen i sig är bättre lämpad för NoSQL bör det övervägas.