Databasteknik

Agenda

- Översikt
  - Syfte/mål, kursplanering
  - Delmoment
- Idag
  - Vad är en databas?
  - Olika typer av databaser
  - SQL
  - Installera Management Studio

Delmoment

- SQL/relationsdatabas (Del 1)
  - Ställa frågor/queries till en databas
  - Manipulera en databas
  - TSQL (Microsofts variant av SQL)
- Databasdesign (Del 2)
  - Normalisering till tredje normalformen, 3NF
  - Constraints, nycklar
  - Skapa UML-diagram över databas

Moment

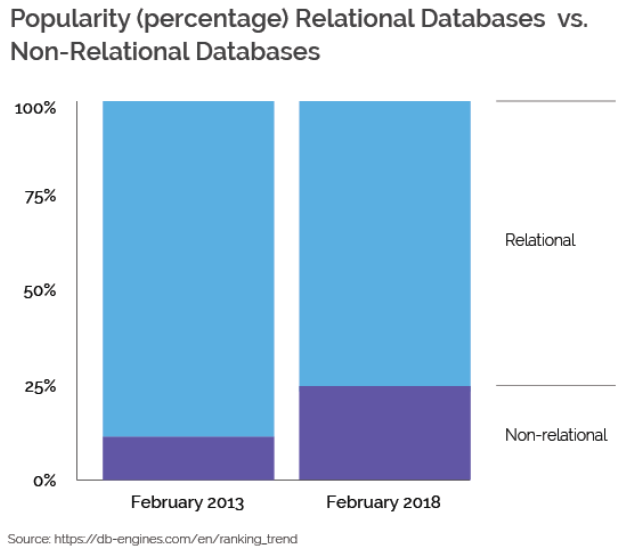
- Entity Framework (Del 3)
  - Designa en relationsdatabas med C# (code first)
  - Utföra queries mot database med LINQ
  - Data access layer, DBContext
- MongoDB, NoSQL (Del 4)
  - Dokumentdatabas

Vad är en databas?

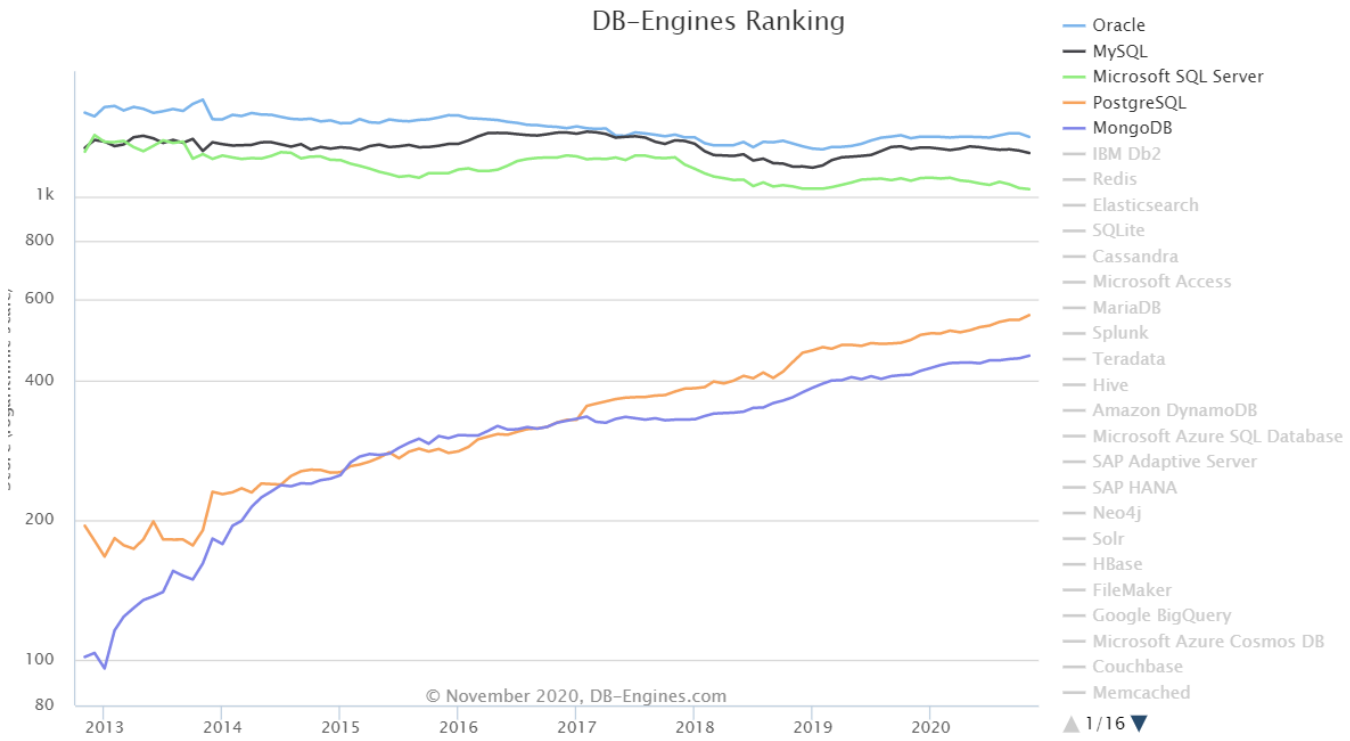
- En samling organiserad data
- Söka i den
  - Kallas *query*
  - “Ge mig alla filmer som släpptes 2014 och börjar på ‘i’”
- Uppdatera data
  - “Ändra mitt lösenord”
- Lägga till eller ta bort data
  - “Skapa en ny användare”

Olika typer av databaser

- Kan vara av olika storlekar och komplexitet
  - En idrottsförening har en enkel databas som lagrar medlemmarna
  - Twitter har en enorm databas med miljarder av tweets
- Finns olika typer av databaser som ofta delas in i två katergorier
  - Relationsdatabas (SQL-databas)
  - Icke-relationsdatabas (NoSQL-databas)
- NoSQL delas in i ytterligare typer
  - T.ex. dokumentdatabas, graph, search engine
- Alla databastyper har olika för- och nackdelar



Topp 5!


DB-Engines Ranking
1k
800
600
400
200
100
80
2013
2014
2015
2016
2017
2018
2019
2020
© November 2020, DB-Engines.com
Oracle
MySQL
Microsoft SQL Server
PostgreSQL
MongoDB
IBM Db2
Redis
Elasticsearch
SQLite
Cassandra
Microsoft Access
MariaDB
Splunk
Teradata
Hive
Amazon DynamoDB
Microsoft Azure SQL Database
SAP Adaptive Server
SAP HANA
Neo4j
Solr
HBase
FileMaker
Google BigQuery
Microsoft Azure Cosmos DB
Couchbase
Memcached
1/16

Skillnader

Relationsdatabas (SQL)

- Uppbyggd av tabeller med rader och kolumner (tänk Excel) där tabellerna har kopplingar (relationer)
- Använder schemas som ställer höga krav på datan
- Normaliserad (3NF), låg redundans

NoSQL

- Inte krav på specifik datastruktur
- Bra för system där enormt mycket data lagras
- Kan parallellisera

Relationsdatabas

- Fördelar
  - Låg eller ingen upprepad data (normaliserad)
  - Eftersom schemas används så är kvaliteten på datan hög, dvs. den lagrar sällan ”fel” data
  - Lätt att läsa ut data från
- Nackdelar
  - Går inte att skala horisontellt (ha datan på olika servrar)
  - Svår att ändra när ett system växer
  - Inte optimal i system med väldigt mycket data

NoSQL

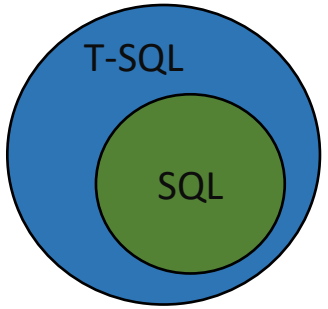
- Fördelar
  - Databaserna kan lagras på flera servrar och skalar horisontellt
  - Bra för enorma datamängder
  - Lättare att ändra struktur i framtiden
- Nackdelar
  - Lägre kvalitet på att datan är korrekt
  - Svårare att ändra data

I denna kurs

- Relationsdatabas
  - Microsoft SQL Server
  - SQL Server Management Studio som gränssnitt
- NoSQL
  - MongoDB som är en dokumentdatabas
  - Compass som gränssnitt

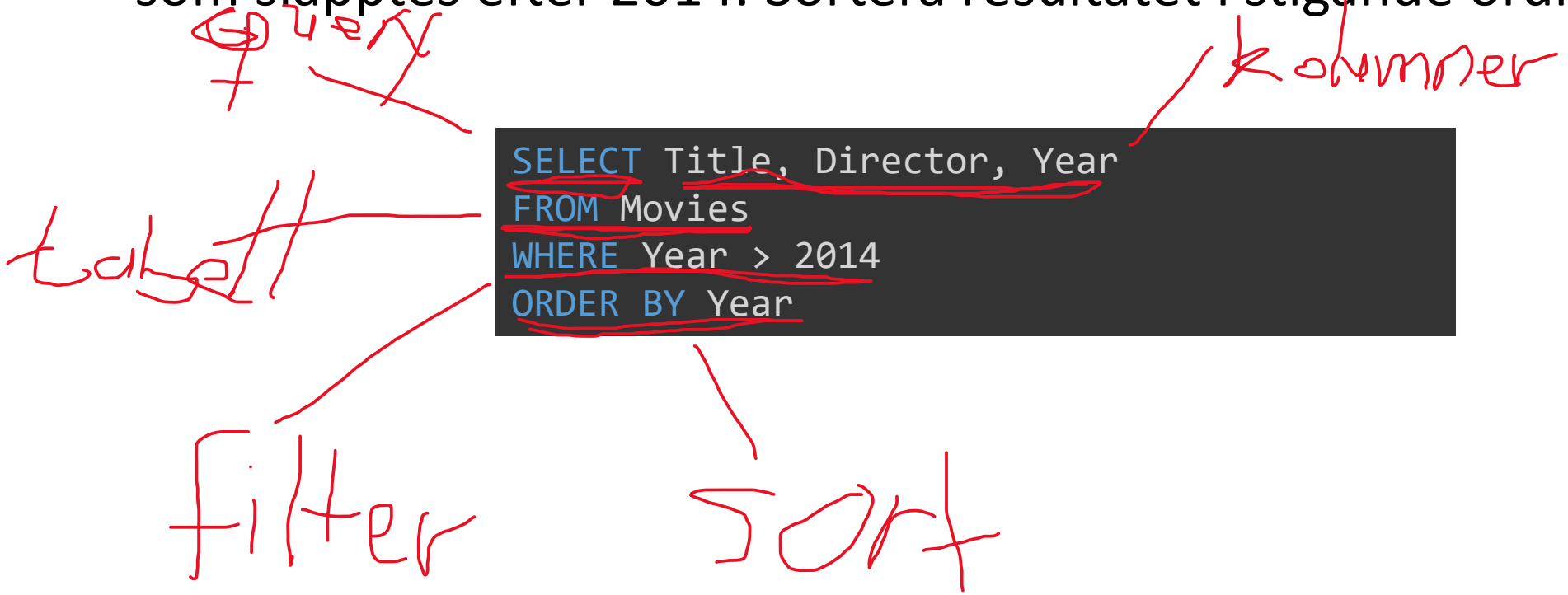
SQL

- Structured **Query** Language
- Används för att kommunicera med en relationsdatabas
  - Query/fråga en databas efter data
  - Manipulera data (skapa, ta bort och ändra)
- Finns olika ”dialekter” av SQL med utökad funktionalitet
  - Vi kommer använda T-SQL (Microsofts version av SQL)



SQL statement

- ”Hämta alla filmtitlar, regissörer och releasedatum i tabellen Movies som släpptes efter 2014. Sortera resultatet i stigande ordning.”




```
SELECT Title, Director, Year
FROM Movies
WHERE Year > 2014
ORDER BY Year
```

SQL statement

- Ett *statement* börjar med ett av dessa nyckelord
  - SELECT, INSERT, UPDATE, DELETE
- SELECT – läsa/hämta data
- INSERT – skapa data
- UPDATE – ändra befintlig data
- DELETE – ta bort data

```
SELECT Title, Director, Year
FROM Movies
WHERE Year > 2014
ORDER BY Year
```

- När jag säger *query* syftar jag på ett *statement* och inte bara SELECT

Query

- Efter SELECT väljs kolumner
  - I detta fall Title, Director och Year
  - Separeras med komma (,)
  - Används * utförs *queryn* på alla kolumner
- Med FROM väljs tabellen i databasen *queryn* ska utföras på
- WHERE filtrerar datan
  - ”Hämta de filmer som släpptes **efter** 2014”

```
SELECT Title, Director, Year
FROM Movies
WHERE Year > 2014
ORDER BY Year
```

Query

- Med ORDER BY sorteras queryn på vald kolumn
- Sortering görs stigande (ascending) eller fallande (descending)
  - ASC för stigande och är standard om inget anges
  - DESC för fallande
- ”Sortera filmerna på år i stigande ordning”

```
SELECT Title, Director, Year
FROM Movies
WHERE Year > 2014
ORDER BY Year
```

WHERE med flera villkor

- Filtrering görs med WHERE
- Med flera villkor använder vi OR och AND
  - OR är samma som ||
  - AND är samma som &&
- ”Hämta filmerna som släpptes efter 2014 **OCH** regisserades av Christopher Nolan”

```
SELECT Title, Director, Year
FROM Movies
WHERE Year > 2014
AND Director = 'Christopher Nolan'
ORDER BY Year
```

Övning - SELECT

- https://sqlbolt.com/lesson/select_queries_introduction

Övning - WHERE

- https://sqlbolt.com/lesson/select_queries_with_constraints

Tills nästa gång

- SQLBolt – lär dig SQL i browsern
  - https://sqlbolt.com/lesson/introduction
  - Gör fram t.o.m lesson 6 tills nästa gang
  - Lesson 6 är ett bra intro till JOINS som vi kommer prata om nästa gång
- Installera SQL Server Management Studio
  - https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15
  - Fyll på med data från script
- Nästa gång
  - GROUP BY
  - JOINS
  - CRUD