Torsdag 24 november – Olika typer av databaser

Även denna dag undervisning på distans.

Obs! Idag endast kl 09.00 – 13.00.

Vid kl 12.00 samling för summering av dagen och möjlighet att ställa frågor.

En återblick på kursens övergripande innehåll

- Relationsdatabaser och hur olika servrar och databashanterare fungerar.
  => Tänk kvalitativt, kunna förstå på ett principiellt plan. Även bra att skaffa sig kännedom om skillnader mellan olika implementationer.
- Lagra data, i såväl fysisk som virtualiserad miljö.
  => Tänk även här på ett principiellt plan, att vara orienterad om begreppen. Inga krav på att förstå allt ner till filnivå.
- Administrera databaser, hantera användare, säkerhetskopiera mm.
  => Mer av "hands on" inom detta område. Dock inte ner på detaljnivå för att fintrimma.
- Modellera data och utforma databaser
  => Förhållandevis mycket ändå inom detta område, för att det är ett sätt att lära sig tolka mellan verksamhet och teknisk lösning.
- Normalisera och optimera databaser.
  => Även här kanske mer på ett principiellt plan, att ha nosat på de första stegen och vara orienterad om begreppen.
- Manipulera databaser och data, genom såväl direkta anrop som lagrade procedurer.
  => Ganska omfattande, men inte för att alla ska bli experter på SQL, utan snarare för att det är ett bra sätt att "lära sig kartan".

Skillnader i SQL mellan olika lösningar

Alla dessa olika databashanterare som går att välja mellan ...

Vi har redan jämfört mellan Sqlite och MS SQL Server. Lite som David vs Goliat.

För- och nackdelar med snabba flexibla lösningar jämfört med mer “stabila” implementationer.

Delvis olika syntax. Stora skillnader i funktionalitet.

ACID = Atomicity, Consistency, Isolation, Durability.

BASE = Basically Available, Soft state, Eventually consistent.

En pessimistisk approach vs en optimistisk?

Jämförelser mellan MySQL, PostgreSQL mm

Punkterna nedan redovisar endast några exempel på databashanterare och deras egenskaper. Fler går att räkna upp.

- MySQL / MariaDB
  - Går att välja mellan två olika “motorer”, InnoDB eller MyISAM (ACID respektive icke ACID).
  - Stöd för JSON.
  - Funktionsbaserade index.
  - Många användare av kombinationen Linux, Apache, MySQL, PHP.
  - MySQL Workbench, ett populärt verktyg för både modellering och administration.
- PostgreSQL[1]
  - Stöd för objektdatabaser (ORDBMS).
  - JSON-data.
  - Geografiska datatyper.
  - Rikt på funktioner.
  - Avancerad transaktionshantering.
- MS Access
  - Filbaserad databas (format .mdb).
  - Många användare på grund av att den följer med MS Office.
  - Bra för enkla lokala ändamål. Usel för applikationer med flera olika användare.
- Paradox
  - Filbaserad databas, med en fil per tabell, en fil per index, osv. (Udda lösning, som dock har sina fördelar.)
  - Starkt kopplat till applikationer i Delphi / Pascal.
  - Förekommer sällan i nya lösningar, men kan leva kvar i gamla system.

1 Lite nostalgi, en artikel om att köra PostgreSQL på Windows för 20 år sedan: https://www.sitepoint.com/use-postgresql-php-windows/

NoSQL

Betyder det No som i “None”, eller som i “Not only”? En bättre benämning kunde vara NoRel, Not Relational.

MongoDB är ett exempel, som inte har tabeller och rader på det sätt vi vant oss vid, utan är en databas bestående av dokument.
BSON = Binary encoded Javascript Object Notation (JSON).

Måste vi nu glömma allt vi lärt oss om att definiera schema?

Annorlunda sätt att göra urval.

- .find(…) istället för kommandot select
- “field: value” istället för villkoret “field=value”
- , (komma) istället för operatorn “and”
- $or: [… , …] istället för operatorn “or”

Fördelar:

- Flexibla strukturer.
- Inga join-operationer, eftersom hierarkiska data redan är ordnade i dokument (ibland inbäddade i andra dokument).
- Snabbare, och effektivare minneshantering.
- Skalbart. Bra för att processa och analysera stora datamängder. (Därav namnet “humongous”?)

Nackdelar:

- Kan vara svårt (för den som är van vid relationsdatabaser) att sätta sig in i annorlunda tänk kring “schema”.
- Inte samma strikthet kring datatyper.
- Inte lika utvecklad felhantering.
- Begränsat frågespråk.

Kort övning med MongoDB

Installationsanvisningar på https://www.mongodb.com/docs/manual/tutorial/ .

Skapa en ny databas “Nackademin”: use Nackademin

(Finns databasen nu? Testa med: show dbs Nej, den visas inte förrän vi faktiskt har lagt till data.)

Skapa en ny collection: db.createCollection("Students")

Lägg till data:

```
db.Students.insertMany( [
{ name: "Knatte", birth_year: 2010, courses: ["Databashantering", "Programmering"] },
{ name: "Fnatte", birth_year: 2010, courses: ["Databashantering"] },
{ name: "Tjatte", birth_year: 2010, courses: ["Databashantering"] }
] )
```

Att vi får lägga till en lista med inte bara en utan två eller flera kurser på en student verkar bryta mot alla regler om normalisering?

Gör urval, motsvarigheten till “select * from Students”: db.Students.find()

Lägg till sortering efter namn: db.Students.find().sort("name")

För att hitta en specifik student: db.Students.find({name: "Knatte"})

För att lägga till en kurs till på en student: db.Students.updateOne({ name: "Tjatte" }, { $push: { courses: "Järnvägsbyggande" } })

För att lägga till en ny egenskap på en student: db.Students.updateOne({ name: "Fnatte" }, { $set: { email: "fnatte@ankeborg.se" } })

(Får man verkligen göra så? Lägga till en “kolumn” som inte fanns tidigare, på bara en “rad”? Ja, faktiskt. För det är NoSQL.)

Läs mer på till exempel https://www.mongodb.com/docs/mongodb-shell/run-commands/ eller https://www.w3schools.com/mongodb/index.php .