

# Relationsdatabas

Ett kort intro

# Relationsdatabasmodellen

- Vad är en databas? Vad är en relationsdatabas?
- *A shared collection of logically related data (and description of this data), designed to meet the information needs of an organization.*  
[Connolly]
- *A database is a collection of related data.*  
[Elmasri]
- *A database is nothing more than a computerized record-keeping system.*  
[Date]
-

# Varför?

- Varför ska vi ha en databas?
  - Lagra (samma) data över tid.
  - Arbeta med (samma) data mellan (och i) användare, system, applikationer.
  - ...
- Vad är det för data?
  - ?

# Relationsdatabasmodellen

- Data lagras och organiseras på ett strukturerat och logiskt sätt – jmfr med OOP.
- Allting utgår ifrån tabeller med rader och kolumner. I varje cell finns data. Varje rad kan ses som ett 'objekt' och varje kolumn har en datatyp och ett värde.
- Tabellerna har relationer mellan varandra, jmfr klasser.
- För att skapa en robust och stabil databas, som ska ingå i ett system, följs vissa principer och en designprocess följs.

# Vilka databaser finns?

- Relationsdatabaser
  - MySql, PostgreSQL, MariaDb, MS SQL Server...
- NoSql
  - Document store
    - MongoDB, CouchDb, ElasticSearch...
  - Key-value stored
    - Memcached, redis, Informix...
  - Object
    - JADE, ODABA, ObjectStore...

# Skillnader mellan relationsdb och NoSql

- En av de största skillnaderna är att i relationsdatabasen byggs relationer mellan data – i vissa typer av NoSql-db finns inga relationer.
- Generellt\* sätt är NoSql-db snabbare, prestandamässigt, men de är mer specialiserade för 'typer av data'. En relationsdatabas kan användas för ett helt system. Att uppnå det med en NoSql-db kan bli svårt... mkt svårt.

\*) gäller inte alla...



# Databasens grundstenar - tabellen

- Med tabeller bygger vi alltså upp en relationsdatabas.
- En tabell består av rader och kolumner.  
En tabell ges ett namn.  
Varje kolumn ges ett namn och en datatyp.

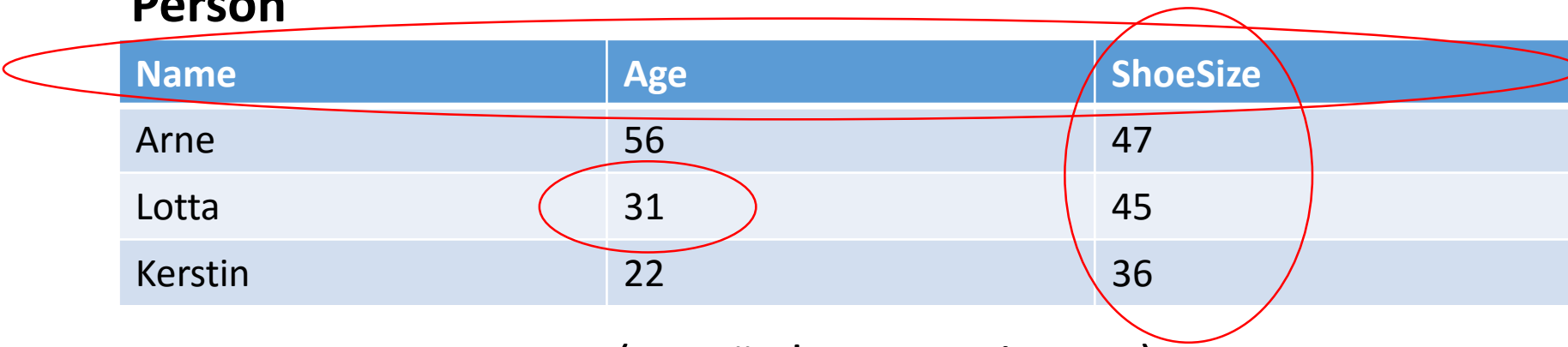
Rad

Person

Kolumn

Name	Age	ShoeSize
Arne	56	47
Lotta	31	45
Kerstin	22	36

Data (ett värde av en viss typ)





# Kärt barn har många namn

- Rad heter också row, tuple. Flera rader kallas ofta recordset.
- Kolumn heter också column, attribut, attribute, fält, field
- Datat heter också värde, value

Name	Age	ShoeSize
Arne	56	47
Lotta	31	45
Kerstin	22	36



# Tabeller, tabeller, tabeller

- Gör ett "Adressregister" med personer och deras adresser;
- Hur ska vi göra? Jmfr med att göra klasser i valfritt OOP-språk...

## **Hitta substantiv!**

person, namn, gata, telefonnummer, email, stad, postkod, etc

- Rita tabell(er)

Surname	Lastname	Street	Zip	City	Email
Lars	Larsson	Broadway 1	12345	New York	l.@ooo.ru
Lisa	Svensson	Broadway 45	12345	New York	ll@gmail.com
Kajsa	Karlsson	Gatan 2	12332	Kalmar	klklk@daf.se
Lars	Larsson	Broadway 1	12345	New York	l.l@ooo.ru
Lars	Larsson	Broadway 2	12345	New York	l.@ooo.ru
Lars	Larsson	Broadway 3	12345	New York	l.@ooo.ru
Göte	Larsson	Broadway 3	12345	New York	g@hej.ru
Helmut	Karlsson	Broadway 3	12345	New York	hk@lol.ug

- Kanske kom vi fram till något liknande detta ovan.
- Jaha ja.
- - Tänk om det finns flera Lars Larsson - hur vet vi vem som är just den Lars?
- - Tänk om Lars har fler än en adress?
- - Tänk om flera bor på samma adress?

# Redundans

- Föregående exempel visar redundans (om än lite).
- Redundans – samma data på flera ställen samtidigt.
- Redundans != bra.
- Vi tycker inte om redundans.
- Redundans är dåligt.
- Fy för redundans.
- Bort med redundans.
- ~~Slakta redundans.~~
- ~~Bränn ner hela skiten.~~

# Hur kan vi undvika redundans?

- Redundans/redundancy - samma data finns på flera ställen... Det vill vi absolut undvika!

Surname	Lastname	Street	Zip	City	Email
Lars	Larsson	Broadway 1	12345	New York	l.@ooo.ru
Lisa	Svensson	Broadway 45	12345	New York	ll@gmail.com
Kajsa	Karlsson	Gatan 2	12332	Kalmar	klklk@daf.se
Lars	Larsson	Långgatan 2	24345	Kalmar	l.@ooo.ru
Lars	Larsson	Knasvägen 1	72345	Svängsta	l.@ooo.ru

# Ofta kan det lösas genom att bryta ut data till flera tabeller

Surname	Lastname	Street	Zip	City	Email
Lars	Larsson	Broadway 1	12345	New York	l.@ooo.ru
Lisa	Svensson	Broadway 45	12345	New York	ll@gmail.com
Kajsa	Karlsson	Gatan 2	12332	Kalmar	klklk@daf.se

Person

Surname	Lastname
Lars	Larsson
Lisa	Svensson
Kajsa	Karlsson

Address

Street	Zip	City	Email
Broadway 1	12345	New York	l.@ooo.ru
Broadway 45	12345	New York	ll@gmail.com
Gatan 2	12332	Kalmar	klklk@daf.se

# Men...

- Från föregående tabeller – hur kan vi nu veta vem som bor på vilken/vilka adresser?
- Eller omvänt, vilka adresser tillhör vilka personer?



# Nycklar - PK

- Lösningen är att använda s.k nycklar/keys.
- Primär nyckel – Primary Key (PK)
  - Lägg till eller använd ett befintligt attribut som unikt identifierar en rad - Primärnyckel (PK)
- M.h.a PK kan vi dela upp data och låta:
  - logiskt sammanhängande data samlas i en tabell.
  - exempelvis, från föregående exempel, det som bygger upp en person och det som bygger upp en adress. Vi får således 2st tabeller.
- En PK är alltså ett attribut som unikt identifierar en rad (tuple).  
Vi kan veta vem av alla Lars som är just vår Lars!!!

# Exempel PK

Person

<u>PersonId</u>	Surname	Lastname
1	Lars	Larsson
2	Lisa	Svensson
3	Kajsa	Karlsson

Address

<u>AddressId</u>	Street	Zip	City	Email
1	Broadway 1	12345	New York	l. @ooo.ru
2	Broadway 45	12345	New York	ll@gmail.com
3	Gatan 2	12332	Kalmar	klklk@daf.se

- Vi har skapat ett nytt attribut i varje tabell eftersom vi ur de existerande inte har hittat något attribut som är unikt.
- Med en PK, Primärnyckel (Primary Key), kan vi unikt identifiera en specifik rad, dvs vem som har vilken adress.

# Men hur kopplar vi ihop tabellerna?

- Dvs, vi vill skapa en relation mellan tabellerna för att tala om vem som har vilken adress.
- Till vår räddning har vi ytterligare en nyckel som kallas Främmande Nyckel (Foreign Key), FK.
- FK är ett attribut vars värde härleds från en annan tabells PK.

# FK -> PK

Person

<u>PersonId</u>	Surname	Lastname
1	Lars	Larsson
2	Lisa	Svensson
3	Kajsa	Karlsson

Address

<u>AddressId</u>	PersonId	Street	Zip	City	Email
1	1	Broadway 1	12345	New York	l. @ooo.ru
2	2	Broadway 45	12345	New York	ll@gmail.com
3	1	Gatan 2	12332	Kalmar	klklk@daf.se

# Övning

- Skivbolag med artister  
*eller*  
Fotbollslag med spelare  
*eller*  
???

Gör tabell(er) på papper e.l likn.

- Rita upp/presentera kort på whiteboarden.

# Hur kan vi då arbeta med en databas

- Vi måste ha tillgång till en (relations)databas -> installera en själv eller använd någon annans.
- 2 typer av gränssnitt mot en relationsdatabas;
  - Det råa terminal-gränssnittet med SQL.
  - ett RDBMS (MySQL: Workbench, Navicat, phpMyAdmin etc etc).
- Dessutom bör vi ha en webbserver och ett språk (ramverk) för att kunna använda vår databas i en webbapplikation.... Vi behöver en hel stack.
- En vanlig stack är LAMP - > Linux, Apache, MySQL, Php

# Installation av en webbstack

- För att kunna jobba med webbutveckling behöver vi en miljö med de tekniker, mjukvaror och annat vi ska använda och behöver. Dessa behöver alltså installeras på ett OS. Vi behöver minst;
  - OS (Windows, Linux, OSX)
  - Webserver (Apache, IIS, nginx, lighttpd, ...)
  - Databas (relationsdb, nosql)
  - Språk/ramverk/toolkit (php, rails, django, node, ...)
  - Eventuellt annat beroende på projekt (git, grunt, yeoman, gvm, ...)

# 2 alternativ till vart vi ska ha miljön

- Lokalt
  - Vi installerar allt på vår fysiska maskin (helt själv eller med t ex Wamp, Xamp, Bitstack, ...).
  - Vi installerar allt i en virtuell maskin på vår fysiska maskin (Vagrant).
- I 'molnet'
  - Webbhotell.
  - En egen (eller hyrd) fysisk server kopplad till nätet.
  - En s.k VPS (Linode, Digital Ocean, City Network, Azure ...).
  - PaaS (Google App Engine, Heroku, AWS, OpenShift, Cloudera ...).
  - Cloud Development Environment (Cloud9, Kodingen, Nitrous ...).
  - ?



Plats	Fördelar	Nackdelar
Lokalt standalone eller via t ex Wamp, Xamp	Vi har allt med oss där vi har vår dator. Säkerhet. Billigt.	Bökigt att konfigurera. Kan vara stökigt att ändra/lägga till saker. Går i längden inte att sätta en app i produktion. Svårt att samarbeta med team.
Lokalt virtuellt med t ex Vagrant	Se ovan. Ett helt OS. Alla i teamet kan ha samma miljö. Säkert*, billigt.	Linux-kunskaper krävs för att åstadkomma något alls. Kan kräva ytterligare kunskaper för konfiguration i t ex chef, puppet etc.
Webbhotell	Någon annan sköter underhållet.	Kostar pengar. Läs under 'Lokalt via te x Wamp, Xamp' förutom samarbete. Oftast väldigt begränsad miljö.
Egen fysisk server	Helt OS.	Mycket goda Linux-kunskaper krävs samt om hårdvara. Säkerhet, underhåll. Kostar pengar.
VPS	Slipper underhåll av hårdvara . Helt OS. Upptid. Team-samarbete.	Goda Linux-kunskaper krävs. Kan bli dyrt.
PaaS	Se ovan. Smidigt för en applikation – betalar bara för vad man använder.	Begränsade möjligheter i konfiguration. Ofta Linux-kunskaper samt specifika beroende på lösning. Kostar ofta pengar.
Cloud Based Development Environment	Se ovan. Ofta shell-, och IDE-access i browsern. Tillgänglighet.	Begränsade möjligheter i konfiguration (ingen root). Linux-kunskaper. Kostar pengar om det ska vara vettigt.

# Övning – installera en stack/databas

- Först och främst behöver du installera en databas. I denna övning används MySQL, vilket gör att det krävs antingen mysql eller mariadb.

Framöver behöver du även en webbserver samt ett ”webbprogrammeringsspråk”, dvs en hel stack. I kursen kommer exempel med php att användas.

- En övningsdatabas (world):
  - <http://dev.mysql.com/doc/index-other.html>  
(välj InnoDB-versionen)
  - Setup: <http://dev.mysql.com/doc/world-setup/en/index.html>