

SQL

NORMALISING

Reduce data redundancy and improve data integrity

Vi ska normalisera våra databaser för att undvika att lagra för mycket data

Samt att undvika såkallade **anomalies**

Titel	Författare	Födelsedatum
Pippi Långstrump	Astrid Lindgren	1907-11-14
Mio min Mio	Astrid Lindgren	1907-11-14
Emil i Lönneberga	Astrid Lindgren	1907-11-14
Bröderna Lejonhjärta	Astrid Lindgren	1907-11-14
Lord of the rings	J.R.R. Tolkien	1892-01-03
The two towers	J.R.R. Tolkien	1892-01-03
The return of the king	J.R.R. Tolkien	1892-01-03

ANOMALY == AVVIKELSE

Om vi duplicerar denna data så utsätts informationen för risker

- Update anomaly
- Insertion anomaly
- Deletion anomaly

Normalisering innebär att man delar upp tabeller så att riskerna elimineras.

Titel	Författare
Pippi Långstrump	Astrid Lindgren
Mio min Mio	Astrid Lindgren
Emil i Lönneberga	Astrid Lindgren
Bröderna Lejonhjärta	Astrid Lindgren
Lord of the rings	J.R.R. Tolkien
The two towers	J.R.R. Tolkien
The return of the king	J.R.R. Tolkien

Författare	Födelsedatum
Astrid Lindgren	1907-11-14
J.R.R. Tolkien	1892-01-03

Eventull risk

Titel	Författare	Födelsedatum
Pippi Långstrump	Astrid Lindgren	1907-11-14
Mio min Mio	Astrid Lindgren	1907-11-14
Emil i Lönneberga	Astrid Lindgren	1907-11-16
Bröderna Lejonhjärta	Astrid Lindgren	NULL
Lord of the rings	J.R.R. Tolkien	1892-01-03
The two towers	J.R.R. Tolkien	1892-01-03
The return of the king	J.R.R. Tolkien	1892-01-03

Födelsedagen behöver aldrig ändras men kan råka göra det ändå

NORMALISERING

Viktigt att dela upp data i olika tabeller för att undvika avvikelser

Det svåra är att sedan sätta ihop tabellerna igen för att få det resultat man söker

MED **SELECT** SÅ HÄMTAR VI ENBART DATA OCH
SKAPAR TILLFÄLLIGA NYA KOLUMNER. INGET I
DATABASEN MODIFIERAR.

DISKUSSION I PAR/GRUPP

Vilka kolumner och tabeller behövs för följande:

1. Instagram-post
2. Skolans register på personerna i FEND16: kontaktinfo, kurser etc.

Övergripande struktur, inte så detaljerat

Använd excel, sheets, paint, papper & penna etc.

När ni är klara, jämför med en annan grupp.

JOINS

Alla **JOINS** skapar en ny tabell som innehåller alla kolumner från både den första och den andra tabellen. Det som skiljer **JOINS** är vilka rader som tas med.

- **CROSS** - alla rader (behövs inga gemensamma kolumner)
- **INNER** - alla rader där värdet i en gemensam kolumn stämmer överens
- **LEFT** - alla rader från en INNER JOIN plus även de från den "vänstra" tabellen som inte matchar något i den högra
- **RIGHT** - som en LEFT JOIN fast tvärtom

Alla joins utgår ifrån CROSS JOIN

1
2
3

**CROSS
JOIN**

A
B
C

=

1
1
1
2
2
2
3
3
3

A
A
A
B
B
B
C
C
C

1

2

3

INNER
JOIN

A

B

C

=

2

A

3

B

1

2

3

LEFT
JOIN

A

B

C

=

1

2

3

A

B

1

2

3

RIGHT
JOIN

A

B

C

=

2

3

A

B

C

1

2

3

FULL
JOIN

A

B

C

=

1

2

3

A

B

C


```
SELECT *  
FROM books  
INNER JOIN authors  
ON books.author = authors.author
```

JOINS

JOINS hör till **FROM**-delen i en **SELECT**-query och är det första som händer när databasen hämtar data.

WHERE, SELECT och GROUP BY kommer senare.

BRA ATT HA GREJER

NAMNGIVNA TABELLER

```
SELECT * FROM authors AS a;
```

```
SELECT * FROM authors a;
```

Blir användbart när vi har flera olika tabeller! Mindre att skriva

AGGREGATE FUNCTIONS

CONCAT() - Läger ihop strängar

AVG() - Tar ut medelvärdet

MAX() - Tar ut maxvärdet

MIN() - Tar ut minvärdet

Det finns alltid flera olika sätt att lösa alla SQL-queries

```
SELECT * FROM city ORDER BY population ASC LIMIT 1;
```

GROUP BY

När man aggregerar (count, avg, max, min) måste man ibland gruppera efter olika kolumner

Vad händer när ni använder `MAX()` för att få ut flera maxvärden

```
SELECT author, COUNT(books) AS books FROM books;
```

```
SELECT author, COUNT(books) AS books FROM books GROUP BY author;
```