Campus Nyköping

- NET2 - 08
  - Hämta ut data från flera tabeller samtidigt

Joins

Förra gången

- "Redovisning" av egen databas
- Mer SQL
  - **Funktioner**
    - VanligaFunktioner.sql
  - **Datatyper och variabler**
    - Databasteknik 06 -SQL - Datatyper och variabler.pdf
  - **Aggregering**
    - Databasteknik 07 -SQL - Aggregering.pdf
  - Övningar
    - Övningsuppgifter – Med bilder - Aggregering av data.pdf

Idag

- Relationsdatabaser
  - Primary och Foreign Key
  - JOIN
  - Tre typer av relationer

Relationer

- Hittills har vi bara skrivit queries som hämtar data från en tabell åt gången. Det vanliga i relationsdatabaser är dock att man har flera tabeller där data i tabell A har en relation till data i tabell B.
- Att man delar upp data i flera tabeller beror på att man inte vill ha flera kopior av samma data.
  - Det tar onödig plats
  - För att man inte ska behöva uppdatera samma data på flera ställen (och därmed löpa en risk att det finns motsägelser i datat).

Böcker och författare i samma tabell

- En databas över böcker och dess författare skulle kunna se ut så här:

Id	Titel	Utgivningsår	Författare	Födelsedatum
1	Ondskan	1981	Jan Guillou	1944-01-17
2	Villospår	1995	Henning Mankell	1948-02-03
3	Brandvägg	1998	Henning Mankell	1948-02-03
4	Innan frosten	2002	Henning Mankell	*1947-04-13*

- Vi behöver då lagra Henning Mankells namn och födelsedatum i flera fält, och i detta fallet har det dessutom smugit sig in ett fel.
  - Vi har alltså upprepade och motsägelsefulla uppgifter i vår databas

Böcker och författare i olika tabeller

- Vi kan istället ha olika tabeller för böcker och författare:

Id	Titel	Utgivningsår	FörfattareID
1	Ondskan	1981	1
2	Villospår	1995	2
3	Brandvägg	1998	2
4	Innan frosten	2002	2

Id	Författare	Födelsedatum
1	Jan Guillou	1944-01-17
2	Henning Mankell	1948-02-03

- Nu behövs inte upprepade uppgifter om Mankells födelsedatum, och eftersom varje bok har ett **FörfattareID** som pekar ut vilken författare som skrivit boken så kan man ändå få ut den informationen

Primary keys, Foreign keys

- Med SQL-kommandot **join** kan vi länka ihop data från två eller fler tabeller så länge dessa har något gemensamt fält som talar om hur data från de olika tabellerna är relaterade till varandra.

- Eftersom alla författare i tabellen ovan har ett unikt Id, en så kallad primärnyckel, så kan vi spara det värdet i en kolumn i boktabellen för att koppla varje bok till en specifik författare. Kolumnen (**FörfattareID**) som pekar ut en rad i en annan tabell brukar kallas **foreign key**

- Vi har tidigare sett att Primary Key unikt identifierar en rad. Foreign Key är en annan tabells Primary Key

Länka ihop data via query

- Antag att vi har två tabeller enligt:

Id	Land
1	Sverige
2	Norge
3	Danmark

Id	Stad	LandId
1	Oslo	2
2	Köpenhamn	3
3	Helsingborg	4

- Relationen mellan de två tabellerna kan ses genom LandsID.
  - Således är Oslo kopplat till Norge och Köpenhamn till Danmark
  - Sverige har inte någon koppling till stad
  - Helsingborg har inte någon koppling till land.
- Låt oss se hur vi kan länka ihop informationen i SQL genom att använda olika typer av **joins**

Cross join

- Alla i första tabellen x alla i andra tabellen
- Det blir fort stort
- Används i princip aldrig

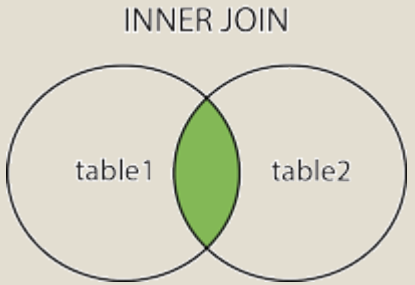
Sverige
Norge
Danmark

CROSS JOIN

Oslo
Köpenhamn
Helsingborg

Sverige	Oslo
Sverige	Köpenhamn
Sverige	Helsingborg
Norge	Oslo
Norge	Köpenhamn
Norge	Helsingborg
Danmark	Oslo
Danmark	Köpenhamn
Danmark	Helsingborg

Inner join (Join)



- Sverige har inga städer
- Helsingborg har inget land
- Ingen av dem kommer med
- The INNER JOIN keyword selects records that have matching values in **both** tables.

Sverige
Norge
Danmark

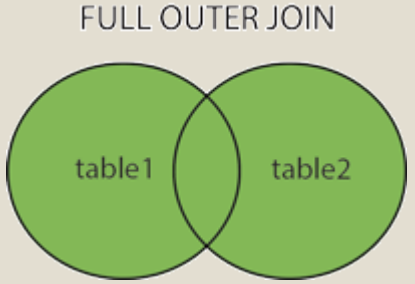
INNER JOIN

Oslo
Köpenhamn
Helsingborg

Norge	Oslo
Danmark	Köpenhamn

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

Full join (Outer Join)



- Sverige kommer med, med NULL som stad
- Helsingborg kommer med, med NULL som land
- The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

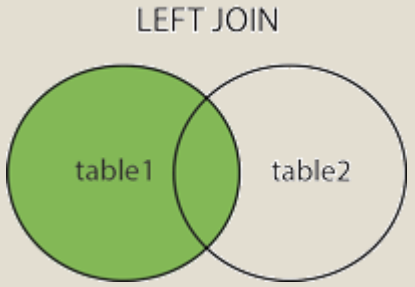
Sverige
Norge
Danmark

OUTER JOIN

Oslo
Köpenhamn
Helsingborg

Sverige	NULL
Norge	Oslo
Danmark	Köpenhamn
NULL	Helsingborg

Left join



- Sverige kommer med, med NULL som stad
- Helsingborg kommer inte med
- The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

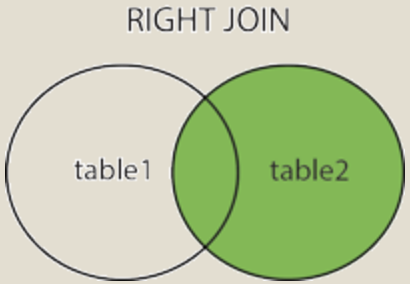
Sverige
Norge
Danmark

LEFT JOIN

Oslo
Köpenhamn
Helsingborg

Sverige	NULL
Norge	Oslo
Danmark	Köpenhamn

Right join



- Helsingborg kommer med, med NULL som land
- Sverige kommer inte med
- The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

Sverige
Norge
Danmark

RIGHT JOIN

Oslo
Köpenhamn
Helsingborg

Norge	Oslo
Danmark	Köpenhamn
NULL	Helsingborg

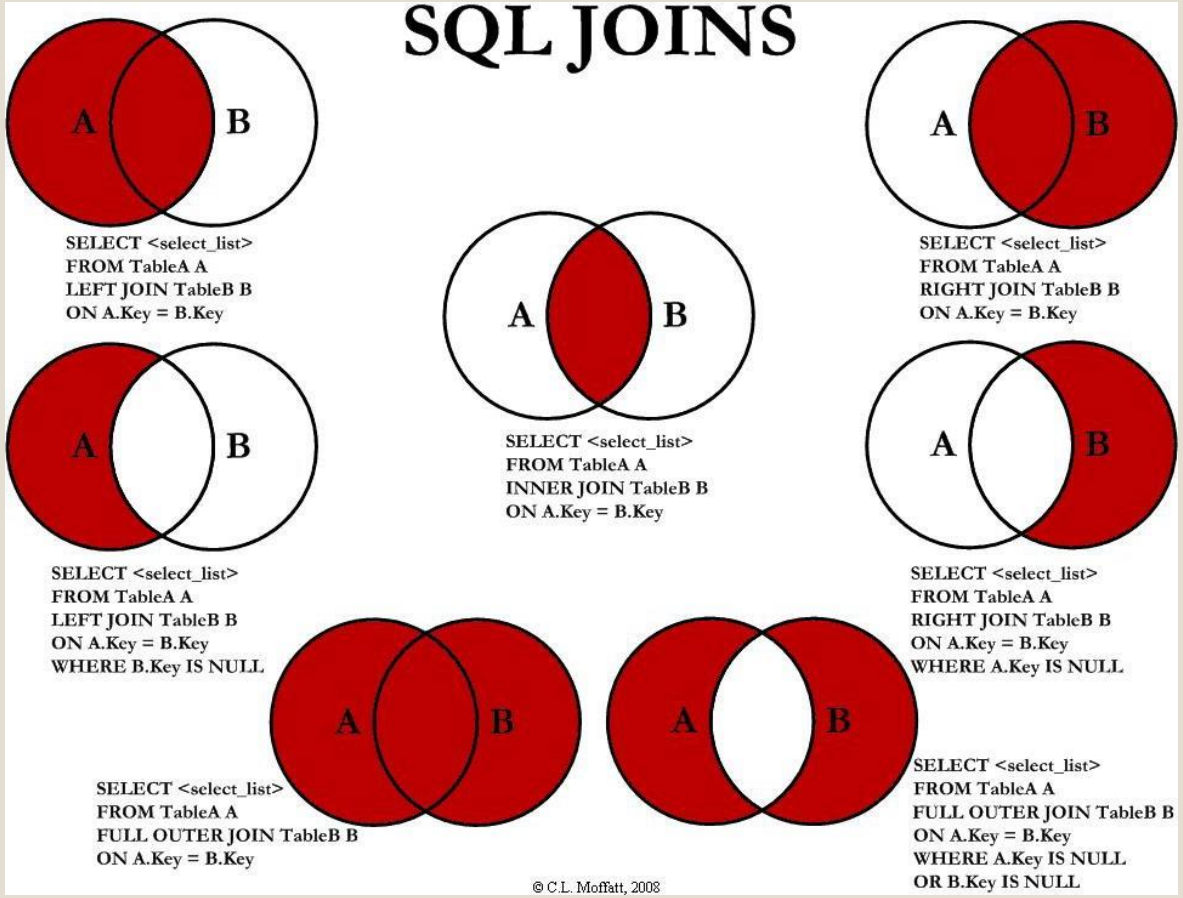
Exempel fråga

```
SELECT * FROM [länder] l
JOIN [städer] s ON s.landsId = l.Id;
```

- Använd **on** för att ange vilka kolumner man ska ”joina” på
- När vi bara skriver **join** så är det samma som **inner join**

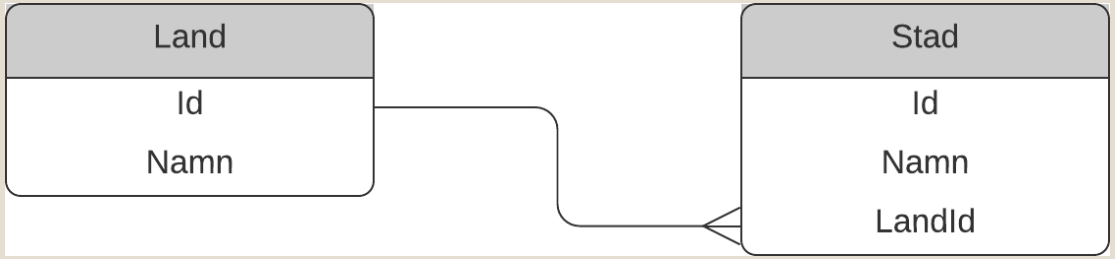
- Vi har behöver inte **as** när vi skapar alias för tabellerna

Alla JOINs med exempel



Typer av relationer: en-till-många

- Exemplet med länder och städer är en **en-till-många** relation:
  - Varje land kan ha flera städer
  - Varje stad ligger i ett land



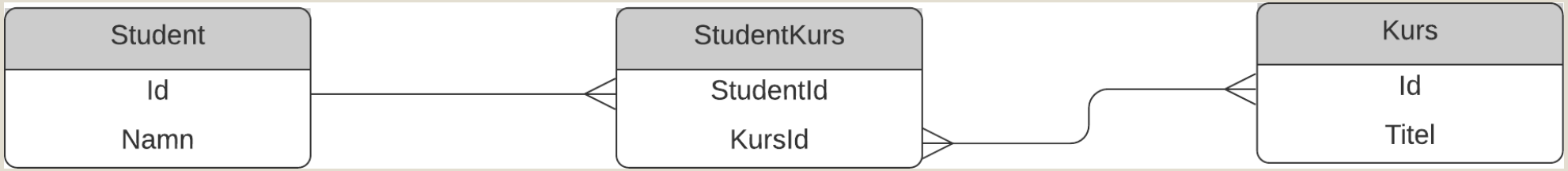
Typer av relationer: många-till-många

- Mellan Student och Kurs har vi ett **många-till-många** förhållande
  - En student läser flera kurser
  - En kurs har flera studenter

- Detta kan vi inte hantera i relationsmodellen
- Vi måste ta till en **junction table**

Typer av relationer: många-till-många

- Mellan Student och Kurs har vi ett **många-till-många** förhållande
  - En student läser flera kurser
  - En kurs har flera studenter



- Med tabellen StudentKurs knyter vi ihop Student med Kurs

JOIN-demo

- JoinLänderStäder.sql
- JoinKurserStudenter.sql

Övningsuppgifter

- Övningsuppgifter – Relationsdata – med bilder.pdf

Länkar

- SQL Joins (w3schools.com)
- SQL INNER JOIN Keyword (w3schools.com)
- SQL LEFT JOIN Keyword (w3schools.com)
- SQL RIGHT JOIN Keyword (w3schools.com)
- SQL FULL OUTER JOIN Keyword (w3schools.com)