INNEHÅLL

- ✓ Historik
- ✓ Vad är SQL
- ✓ Språkuppbyggnad
- ✓ SELECT
- ✓ Operatorer Logiska/Jämförelse
- ✓ SELECT med Alias och JOIN
- ✓ INSERT
- ✓ UPDATE
- ✓ DELETE

Chapter 8, 9, 11 – delar av.

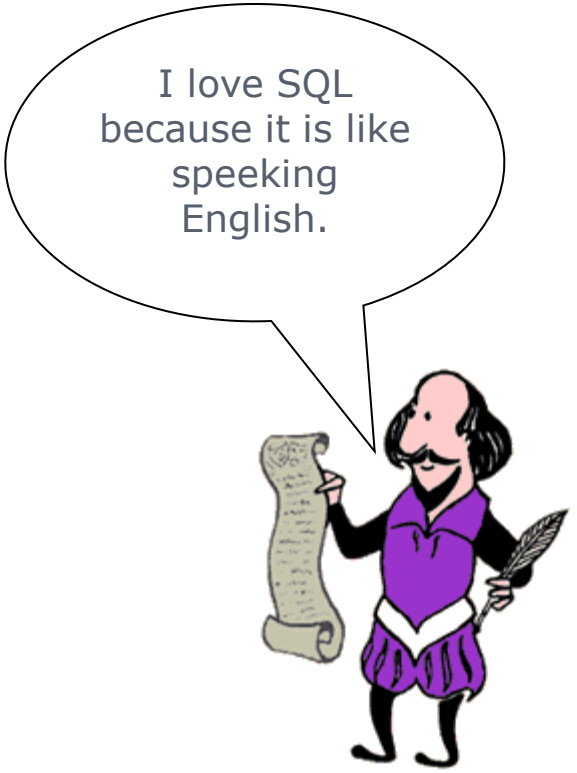
Beginning SQL Server 2008 for Developers

VAD ÄR SQL?

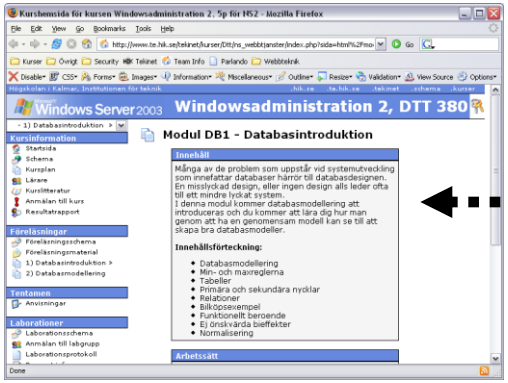
SQL kan i mångt och mycket liknas vid ett vanligt språk.

”Hämta alla studenter från tabellen ’Studenter’ som tillhör kursen ’webbtjänster’ och sortera studenterna i namn ordning”

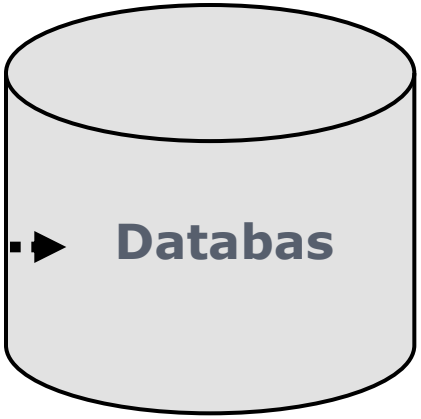
```
SELECT  namn, adress, telefon
FROM Studenter
WHERE Course = ’webbtjänster’
ORDER BY namn ;
```



Webbgränssnitt



Kommunikationen mellan en webbsida och databasen sker oftast med SQL



Vad är SQL? Wikipedia

SQL är ett språk som används för att ges åtkomst till och modifiera data i en databas.

SQL står för **S**tructured **Q**uery **L**anguage
(ös. *Strukturerat Frågespråk*)

SQL är en ANSI-standard för att modifiera och läsa från databaser.

ANSI = American National Standards Institute

Gavs ut år…	Benämning	Anna ben.	Omfattning
1986	SQL-86	SQL-87	Första publikationen
1989	SQL-89		Mindre revision
1992	SQL-92	SQL2	Major Update, vanligast använda
1999	SQL-99	SQL3	Uppgradering av SQL-92
2003	SQL-2003		Utökad med XML funktioner
2006	SQL-2006		Utveckling av XML funktionalitet
2008	SQL-2008		CURSOR, INSTEAD, TRUNCATE
2011	SQL-2011		

SQL Begrepp

- DML — Data Manipulating Language
  SELECT , DELETE, UPDATE, INSERT
- DDL — Data Definition Language
  CREATE, ALTER, DROP
- DCL — Data Control Language
  GRANT, REVOKE
- TCL — Transactional Control Language
  BEGIN, COMMIT, ROLLBACK

SQL Data Manipulation Language (DML)

DML är den syntaxdel av SQL-språket som innehåller de kommandon som gör att vi kan läsa ut och skriva data till tabellerna i databasen.

Här hittar vi följande kommandon:

Kommando	Används till
SELECT	Läser ut data från tabellen, urvalsfråga
UPDATE	Uppdaterar data i tabellen, uppdateringsfråga
DELETE	Tar bort data från tabellen, borttagningsfråga
INSERT	Lägger in ny data i tabellen, tilläggsfråga

SQL Data Definition Language (DDL)

DDL är den syntaxdel av SQL-språket som innehåller de kommandon som gör att vi kan skapa och radera tabeller samt sätta index på speciella kolumner i tabellerna.

Här hittar vi följande kommandon:

Kommando	Används till
CREATE	Skapar ny, databas, tabell, index mfl
ALTER	Modifierar en existerande
DROP	Raderar en databas, tabell, index

SQL Data Control Language (DCL)

DCL är den syntaxdel av SQL-språket som innehåller de kommandon som gör att vi kan arbeta med flera förändringar rättigheter för inloggning, användare etc.

Här hittar vi följande kommandon:

Kommando	Används till
GRANT	Tilldelning av rättigheter
REVOKE	Avsluta/återta rättighet
DENY	Neka rättighet

SQL Transaction Control Language (TCL)

TCL är den syntaxdel av SQL-språket som innehåller de kommandon som gör att vi kan arbeta med flera förändringar och att de blir genomförda och kontrollerar förloppet vid eventuella avbrott.

Här hittar vi följande kommandon:

Kommando	Används till
BEGIN TRANSACTION	Startar transaktionshantering
COMMIT TRANSACTION	Avslutar transaktionshantering
ROLLBACK TRANSACTION	Backar så transaktionen återställs

SELECT - SYNTAX

Syntaxen i en enklare form kan se ut på följande sätt där SELECT, FROM, WHERE och ORDER BY har sin givna inbördes ordning. :

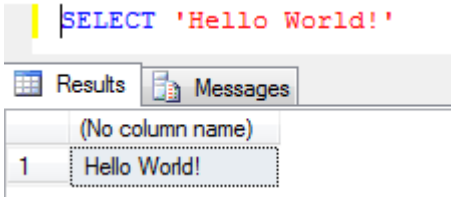
Syntax:

```
SELECT kolumn(er)
FROM tabell(er)
WHERE villkor operator villkor operator villkor
ORDER BY fält DESC, fält ASC;
```

SELECT	anger vilka fält, kolumner, som ska visas
FROM	anger vilken eller vilka tabeller som fälten ska visas ur
WHERE	anger villkoren för vilka poster som ska visas
ORDER BY	anger sorteringsordning på raderna

Du måste börja med SELECT – vilket också är det enda du behöver använda:

SELECT 'Hello World!';



Ordningen mellan nyckelorden är fast och kan inte ändras. Försöker du på annat sätt får du felmeddelande.

Exempel – tillåtet

```
SELECT Namn, Kundid
FROM Kund
WHERE Ort='KALMAR'
ORDER BY Namn DESC;
```

Vilket lämnar ett Resultatset

```
SELECT Namn, Kundid
 FROM Kund
 WHERE Ort='KALMAR'
 ORDER BY Namn DESC;
```

Results | Messages

	Namn	Kundid
1	Svensson Mekaniska	1
2	Karlssons El	3
3	IT Experten	4

Exempel – inte tillåtet

~~SELECT Namn, Kundid~~
~~WHERE Ort='KALMAR'~~
~~FROM Kund~~
~~ORDER BY Namn DESC~~

Vilket lämnar ett felmeddelande

```
SELECT Namn, Kundid
 WHERE Ort='KALMAR'
FROM Kund
 ORDER BY Namn DESC;
```

Messages

```
Msg 156, Level 15, State 1, Line 3
Incorrect syntax near the keyword 'FROM'.
```

SELECT

Vi använder normalt SELECT för att läsa ut data ur en eller flera tabeller.

Syntax:

```
SELECT kolumn(er)
FROM tabell;
```

Person:

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

Exempel:

```
SELECT PersonID, Efternamn
FROM Person;
```

Resultatet kallas för ett ”Resultset”

PersonID	Efternamn
1	Karlsson
2	Nilsson
3	Jansson
4	Karlsson

SELECT

Efter SELECT anger du vilka fält som ska finnas med i ditt resultat. Ordningen som du skriver fältnamn anger ordningen på kolumnerna i resultatet:

PID	Fnamn	Enamn	Ort
1	Kalle	Karlsson	Kalmar
2	Nisse	Nilsson	Nybro
3	Janne	Jansson	Nybro
4	Petter	Karlsson	Växjö

Exempel:

```
SELECT Ort, Enamn
FROM Person;
```

Ort	Enamn
Kalmar	Karlsson
Nybro	Nilsson
Nybro	Jansson
Växjö	Karlsson

Exempel:

```
SELECT Fnamn, Ort, Enamn
FROM Person;
```

Fnamn	Ort	Enamn
Kalle	Kalmar	Karlsson
Nisse	Nybro	Nilsson
Janne	Nybro	Jansson
Petter	Växjö	Karlsson

SELECT

Istället för att ange vilka kolumner som ska visa så använder vi * (asterisk) för att välja alla kolumner.

```
SELECT *
FROM Person;
```

Person:

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

Med * blir ordningen mellan kolumnerna som de är uppsatta i tabellen.

Syntax

SQL språket är inte case sensitive. Du kan blanda gemener och versaler som du önskar. Rekommendationen är att du alltid skriver nyckelorden med versal och har ny rad vid varje nyckelord.

Exempel:

```
SELECT PersonID, Efternamn
FROM Person
WHERE Personid=1;
```

Istället för:

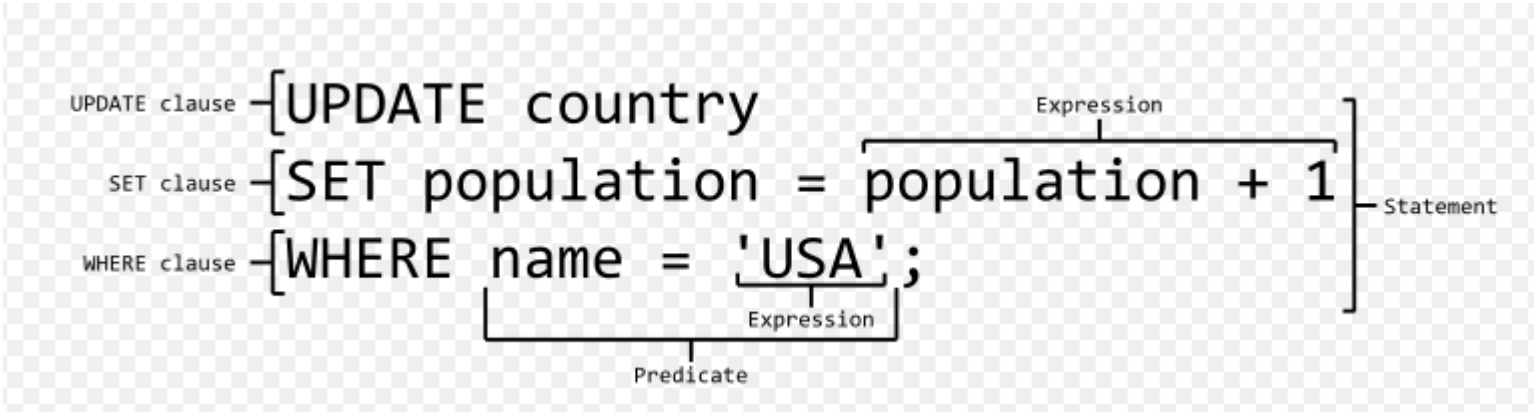
```
SELECT PersonID, Efternamn FROM Person WHERE Personid=1;
```

Ska man avsluta med ; (semikolon) eller inte?

Enligt ANSI avslutas en SQL sats med **;**
Microsofts produkter kräver inte detta.
Vill du vara portabel med din kod använder du semikolon ; I MySQL måste du använda ;

Radbrytningar, whitespaces, kan du använda dig av.

```
SELECT
PersonID,
Efternamn
FROM Person
WHERE
Personid=1;
```




```
SELECT PersonID, Efternamn
FROM Person
WHERE Personid=1;
```

Begrepp	***Översättning***	***Svenskt***
Clause	*sats /bisats/ klausul*	*nyckelord*
Expression	*uttryck*	*uttryck*
Predicate	*satsdel/förkunna*	*villkor*
Statement	*uttalande*	*sats (SQL sats)*

Operatorer Logiska SQL

Operator	Utför	Exempel
NOT eller !	INTE	Lon > 7000 AND NOT Ort='Kalmar'
AND eller &&	OCH	Ort='Kalmar' AND Namn Like '%son'
OR eller ||	ELLER	Lon <10000 OR Lon>14000
XOR	Exclusive OR	Select a XOR b
a XOR b är matematiskt lika med		a AND NOT b

Exempel:

```
SELECT Namn
FROM Produkt
WHERE Postnr=39351 OR Postnr=39364;
```

Operatorer Jämförelse SQL

Operator	Utför	Exempel
=	Lika med	A = B
<> !=	Inte lika med	A <> B, A!=B
<	Mindre än	10 < 20
<=	Mindre än eller lika med	10<=10
>	Större än	25>12
>=	Större än eller lika med	25>=24
BETWEEN	Mellan i området	BETWEEN 10 AND 20
IN	Finns i specifikation	7 IN (1,3,4,5,8,9)
IS NULL	Är Null	tel IS NULL
IS NOT NULL	Är inte Null	namn IS NOT NULL
LIKE	lika som	postnr like '393%'
LIKE	lika som	postnr like '39? 51'

Exempel:

```
SELECT Namn
FROM Produkt
WHERE Postnr BETWEEN 39351 AND 39364;
```

Operatorer Aritmetiska

Det finns ett antal operatorer som vi kan ta till hjälp för att göra beräkningar direkt i SQL-koden

Operator	Utför
+	addition
-	subtraction
*	multiplikation
/	division
%	division, modulus

Beräkna	Resultat
25+7	32
25-7	18
12*2	24
25/4	6,25
25%4	1

ProduktID	Antal	Pris
1	100	35
2	10	100
3	15	120

Exempel:

```
SELECT ProduktID, Antal*Pris
FROM Produkt;
```


ProduktID	(NO NAME)
1	3500
2	1000
3	1800

SELECT

För att läsa ut samtliga kolumner från tabellen kan du använda *:

Exempel:

```
SELECT *
FROM Person;
(* anger att alla fält i
den aktuella tabellen
ska hämtas)
```

→

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

DISTINCT används för att se till så att dubbletter inte utläses:

Exempel:

```
SELECT DISTINCT Efternamn
FROM Person;
(I tabellen ovan finns två Karlsson. När
DISTINCT används visas bara en.)
```

→

Efternamn
Karlsson
Nilsson
Jansson

TOP används för att läsa ut de första i sorteringsordning:

Exempel:

```
SELECT TOP 3 PersonID
FROM Person;
```

→

PersonID
1
2
3

SELECT … WHERE

För att kombinera utläsningen med villkor använder vi WHERE:

Syntax:

```
SELECT kolumn(er) FROM tabell
WHERE kolumn operator värde;
```

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

Exempel:

```
SELECT * FROM Person
WHERE Efternamn = 'Karlsson';
```

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
4	Petter	Karlsson

Exempel:

```
SELECT * FROM Person
WHERE Efternamn = 'Karlsson'
AND Fornamn='Petter';
```

PersonID	Fornamn	Efternamn
4	Petter	Karlsson

Exempel:

```
SELECT * FROM Person
WHERE Efternamn = 'Karlsson'
OR Efternamn='Nilsson';
```

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
2	Nisse	Nilsson
4	Petter	Karlsson

SELECT … WHERE

Exempel:

```
SELECT * FROM Person
WHERE (Efternamn = 'Karlsson'
OR Efternamn='Nilsson')
AND personID<4;
```

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
2	Nisse	Nilsson

Exempel:

```
SELECT * FROM Person
WHERE Efternamn Like '%son';
```

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

Exempel:

```
SELECT * FROM Person
WHERE Efternamn Like '%a%';
```

PersonID	Fnamn	Efternamn
1	Kalle	Karlsson
3	Janne	Jansson
4	Petter	Karlsson

SELECT … ORDER BY

För att sortera slutresultatet använder vi **ORDER BY**:

Syntax:

```
SELECT kolumn(er)
FROM tabell
WHERE kolumn operator värde
ORDER BY kolumn ordning;
```

Där ”ordning” är:

ASC Stigande ordning (A-Ö, 1-9 etc.) DEFAULT
DESC Fallande ordning (Ö-A, 9-1 etc.)

Exempel:

```
SELECT *
FROM Person
ORDER BY Efternamn ASC;
```

PersonID	Fornamn	Efternamn
3	Janne	Jansson
1	Kalle	Karlsson
4	Petter	Karlsson
2	Nisse	Nilsson

OBS! Ibland sätts ordningen automatiskt vilket beror på andra funktioner. Ex index, Group by etc.

ALIAS

Med Alias kan vi döpa om existerande kolumner samt döpa nyskapade.

Syntax 1:

```
SELECT kolumn AS nytt_namn
FROM …
```

ProduktID	Antal	Pris
1	100	35
2	10	100
3	15	120

Exempel:

```
SELECT ProduktID, Antal*Pris AS Totalt
FROM Produkt;
```

ProduktID	Totalt
1	3500
2	1000
3	1800

Syntax 2:

```
SELECT kolumn
FROM tabell AS nytt_tabellnamn
```

Exempel:

```
SELECT ProduktID, Antal*Pris AS Totalt
FROM Produkt AS Totaltabell;
```

Totaltabell:

ProduktID	Totalt
1	3500
2	1000
3	1800

JOIN (SLÅ IHOP TABELLER)

Tabell A

A	B	C	D

Tabell B

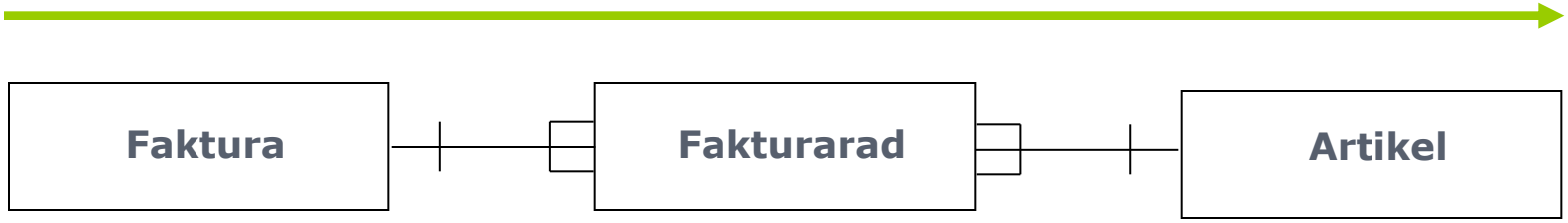
E	F	*A(fk)*

JOIN

”Adderas på bredden”
Resultset:

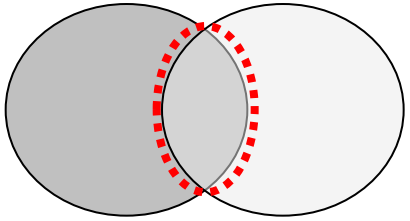
A	B	C	D	E	F

Nu kan vi äntligen börja dra slutsatser av våra databasmodeller!



INNER JOIN

INNER JOIN
Alla i tabellerna
matchande poster



Syntax:

```
SELECT kolumn(er)
FROM tabell_1
INNER JOIN tabell_2
ON primärnyckel = sekundärnyckel
WHERE kolumn operator värde
```

Exempel:

```
SELECT *
FROM Produkt AS P
INNER JOIN Produkttyp AS PT
ON P.typID = PT.typID
WHERE P.Produktnamn LIKE '%Gun%';
```

AS P och **AS PT** innebär att vi slipper skriva ut Produkt respektive Produkttyp senare i koden

ProduktID	Artnr	Produktnamn	typID	Benamning
1	3443	Stolen Gunnar	12	Sittmöbel
3	6534	Pallen Gun	12	Sittmöbel

SAMMANSLAGNING JOIN

Produkt:

ProduktID	Artnr	Produktnamn	typID
1	3443	Stolen Gunnar	12
2	5423	Bordet Jan	21
3	6534	Pallen Gun	12
4	6545	Kassen Jarl	2

Produkttyp:

ptID	Benamning
2	Förvaring
12	Sittmöbel
21	Bord

Exempel:

```
SELECT * FROM Produkt INNER JOIN Produkttyp;
```

Produkttyp*Produkt antal poster (4*3 = 12st):

Det är dessa vi är intresserade av.

ProduktID	Artnr	Produktnamn	typID	ptID	Benamning
1	3443	Stolen Gunnar	12	2	Förvaring
2	5423	Bordet Jan	21	2	Förvaring
3	6534	Pallen Gun	12	2	Förvaring
4	6545	Kassen Jarl	**2**	**2**	Förvaring
1	3443	Stolen Gunnar	**12**	**12**	Sittmöbel
2	5423	Bordet Jan	21	12	Sittmöbel
3	6534	Pallen Gun	**12**	**12**	Sittmöbel
...	...	...	...	...	...
4	6545	Kassen Jarl	2	21	Bord

SAMMANSLAGNING FORTS JOIN.

Produkt:

ProduktID	Artnr	Produktnamn	typID
1	3443	Stolen Gunnar	12
2	5423	Bordet Jan	21
3	6534	Pallen Gun	12
4	6545	Kassen Jarl	2

Produkttyp:

ptID	Benamning
2	Förvaring
12	Sittmöbel
21	Bord

Exempel 1:

```
SELECT *
FROM Produkt
INNER JOIN Produkttyp
  ON Produkt.typID = Produkttyp.ptID;
```

ProduktID	Artnr	Produktnamn	typID	ptID	Benamning
1	3443	Stolen Gunnar	12	12	Sittmöbel
2	5423	Bordet Jan	21	21	Bord
3	6534	Pallen Gun	12	12	Sittmöbel
4	6545	Kassen Jarl	2	2	Förvaring

Exempel 2:

```
SELECT Produkt.Produktnamn, Produkttyp.Benamning
FROM Produkt
INNER JOIN Produkttyp
 ON Produkt.typID = Produkttyp.ptID;
```

Produktnamn	Benamning
Stolen Gunnar	Sittmöbel
Bordet Jan	Bord
Pallen Gun	Sittmöbel
Kassen Jarl	Förvaring

SAMMANSLAGNING FORTS JOIN.

Produkt:

ProduktID	Artnr	Produktnamn	typID
1	3443	Stolen Gunnar	12
2	5423	Bordet Jan	21
3	6534	Pallen Gun	12
4	6545	Kassen Jarl	2

Produkttyp:

ptID	Benamning
2	Förvaring
12	Sittmöbel
21	Bord

Vi ställer oss frågan. Vilka produkter tillhör produkttypen ”Sittmöbel”?

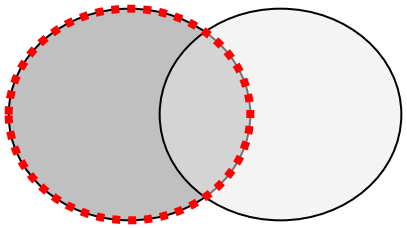
Lösning:

```
SELECT Produkt.Produktnamn
FROM Produkt
INNER JOIN Produkttyp
  ON Produkt.typID = Produkttyp.ptID
WHERE Produkttyp.Benamning = 'Sittmöbel';
```

Produktnamn
Stolen Gunnar
Pallen Gun

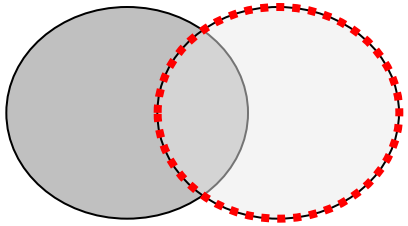
OUTER JOIN

LEFT OUTER JOIN



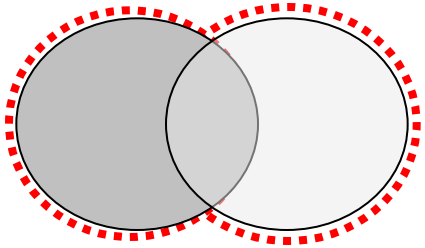
Tar med snittet (matchande poster) samt poster i vänster tabell som inte har någon länk till höger tabell

RIGHT OUTER JOIN



Tar med snittet samt poster i höger tabell som inte har någon länk till vänster tabell

FULL OUTER JOIN



Tar med snittet samt poster i höger och vänster tabell som inte har någon länk till motstående tabell. En kombination av både LEFT och RIGHT OUTER JOIN

LEFT OUTER JOIN

Produkt:

ProduktID	Artnr	Produktnamn	typID
1	3443	Stolen Gunnar	12
2	5423	Bordet Jan	21
3	6534	Pallen Gun	12
4	6545	Kassen Jarl	2
5	9875	Kvitto	

Produkttyp:

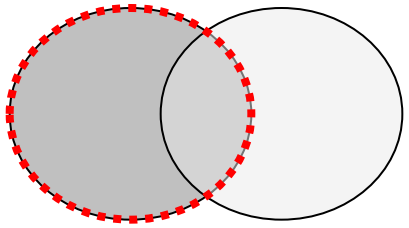
ptID	Benamning
2	Förvaring
12	Sittmöbel
21	Bord

Exempel:

```
SELECT *
FROM Produkt
LEFT OUTER JOIN Produkttyp
  ON Produkt.typID = Produkttyp.ptID;
```

LEFT syftar på denna tabell. (Syns tydligare om hela frågan skrivs på en rad)

ProduktID	Artnr	Produktnamn	typID	ptID	Benamning
1	3443	Stolen Gunnar	12	12	Sittmöbel
2	5423	Bordet Jan	21	21	Bord
3	6534	Pallen Gun	12	12	Sittmöbel
4	6545	Kassen Jarl	2	2	Förvaring
5	9875	Kvitto			



RIGHT OUTER JOIN

Produkt:

ProduktID	Artnr	Produktnamn	typID
1	3443	Stolen Gunnar	12
2	5423	Bordet Jan	21
3	6534	Pallen Gun	12
4	6545	Kassen Jarl	2
5	9875	Kvitto	

Produkttyp:

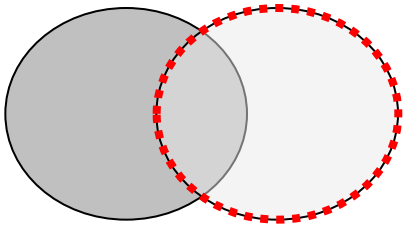
ptID	Benamning
2	Förvaring
12	Sittmöbel
21	Bord
55	Mattor

Exempel:

```
SELECT *
FROM Produkt
RIGHT OUTER JOIN Produkttyp
  ON Produkt.typID = Produkttyp.ptID
```

RIGHT syftar på denna tabell.

ProduktID	Artnr	Produktnamn	typID	ptID	Benamning
1	3443	Stolen Gunnar	12	12	Sittmöbel
2	5423	Bordet Jan	21	21	Bord
3	6534	Pallen Gun	12	12	Sittmöbel
4	6545	Kassen Jarl	2	2	Förvaring
				55	Mattor



SLÅ IHOP FLERA TABELLER

Exempel:

```
SELECT *
FROM Faktura AS F
INNER JOIN [Faktura rad] AS Fa
  ON F.FakturaID = Fa.FakturaradID
INNER JOIN Artikel AS A
  ON Fa.ArtikelID = A.ArtikelID
```

[tabellnamn] används i SQL Server för att undvika tvetydigheter i namn.
`tabellnamn` används i MySQL

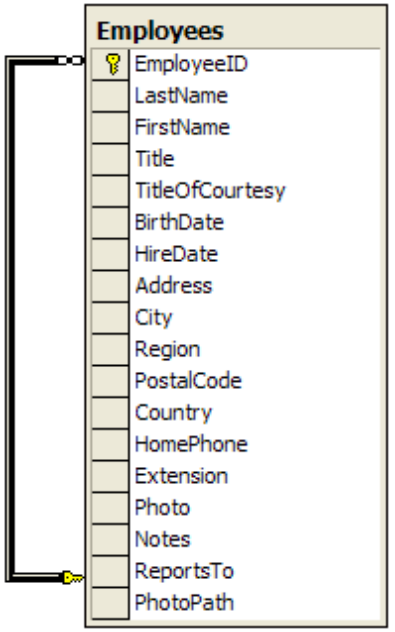
Används bland annat vid:

- Namn som innehåller mellanslag, blanksteg
- Tecken så som Å Ä Ö
- Då reserverade tabellnamn används (T.ex. From)

EGENRELATIONER

En egenrelation är en relation inom tabellen

- I ***Northwind***-databasen finns det en egenrelation
  - En chef har en eller flera anställda
- *"**Alias"*** krävs vid egenrelationer



Vi ställer oss frågan. Vilka Antällda har en chef och vem är i såfall det?

Exempel:

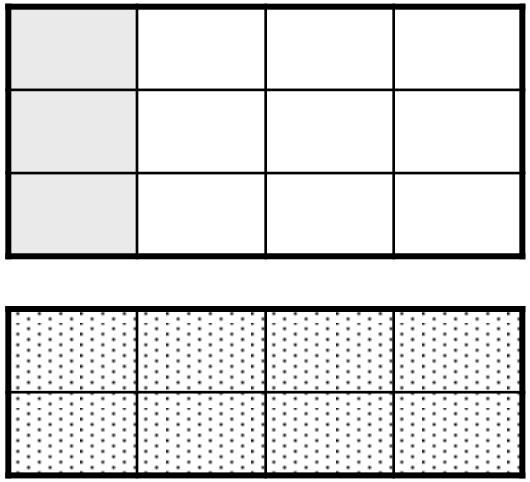
```
SELECT E.LastName AS Anställd, B.LastName AS Chef
FROM Employees AS E
INNER JOIN Employees AS B
  ON E.ReportsTo = B.EmployeeID
```

Anställd	Chef
Davolio	Fuller
Leverling	Fuller
Peacock	Fuller
...	...
Dodsworth	Buchanan

Hur blir det om vi istället frågar oss: Vilka anställda finns och vem har de eventuellt som chef?

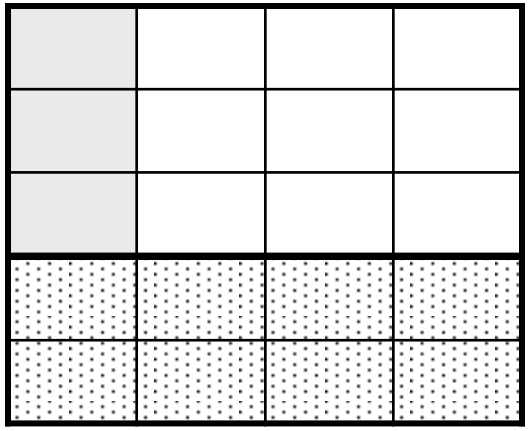
UNION OCH JOIN

Samma antal kolumner



UNION

”Adderas på längden”



Gemensam kolumn

A	B	C	D

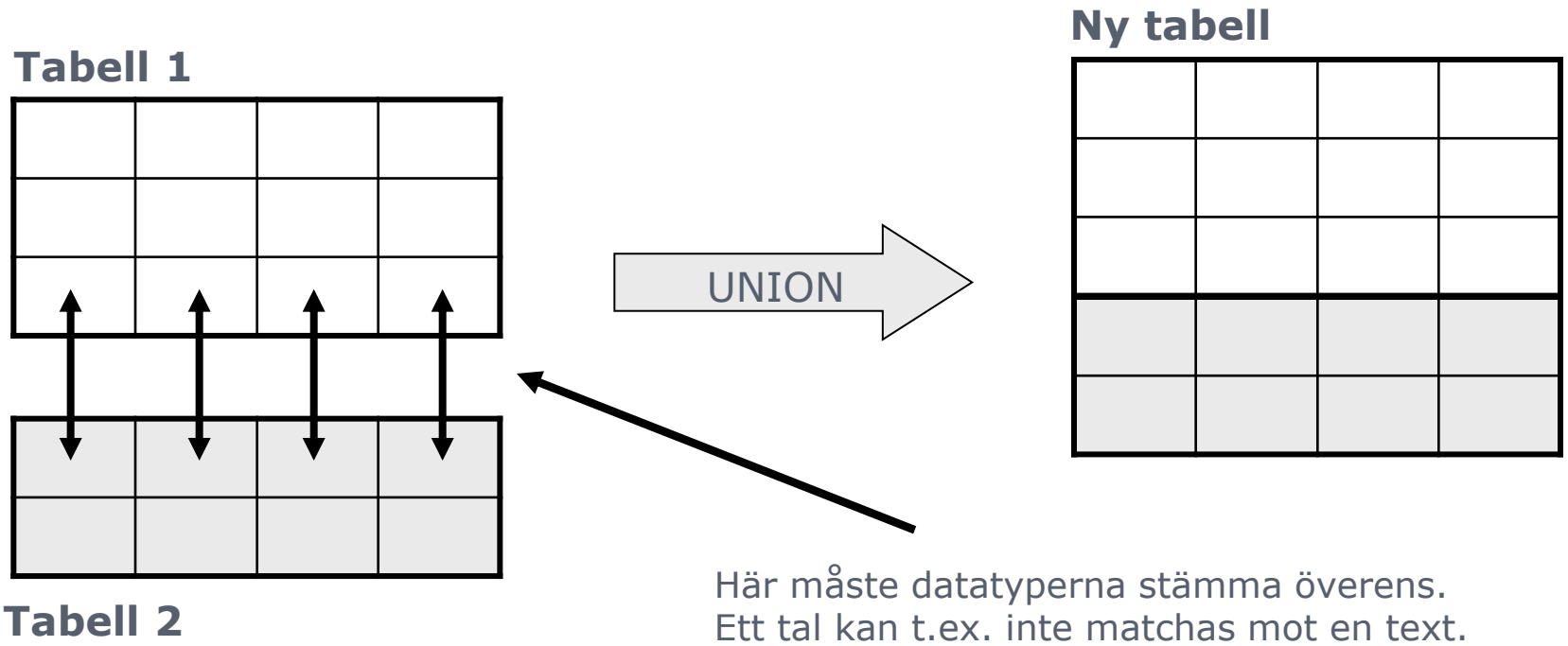
JOIN

”Adderas på bredden (och längden)”

A	B	C	D	E	F

E	F	A

UNION



Här måste datatyperna stämma överens.
Ett tal kan t.ex. inte matchas mot en text.

Syntax 1:

```
SELECT kolumn(er) FROM tabell_1 WHERE …
UNION
SELECT kolumn(er) FROM tabell_2 WHERE …
```

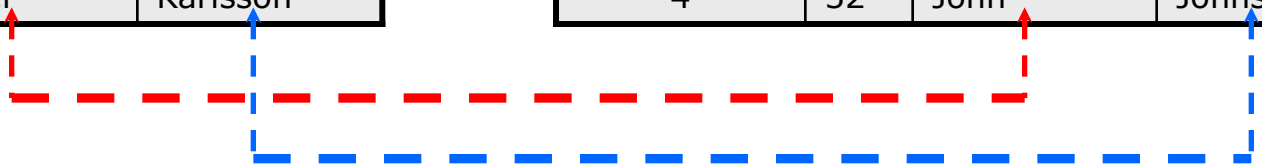
UNION

Emloyee:

EmpID	Firstname	Sirname
1	Kalle	Karlsson
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

Member:

MemeberID	Age	FName	LName
1	34	Peter	Johansson
2	54	Nisse	Nilsson
3	12	Karl	Svensson
4	32	John	Johnsson



Exempel:

```
SELECT Firstname, Sirname FROM Employee
UNION
SELECT Fname, LName FROM Member;
```

Firstname	Sirname
Kalle	Karlsson
Nisse	Nilsson
Janne	Jansson
Petter	Karlsson
Peter	Johansson
Karl	Svensson
John	Johnsson

Observera att kolumnrubrikerna blir samma som i första tabellen samt att Nisse Nilsson inte kommer att visas två gånger.
För att byta Kolumnrubrik använd AS i första selectsatsen:

```
SELECT Firstname AS First, Sirname AS Last FROM …
```

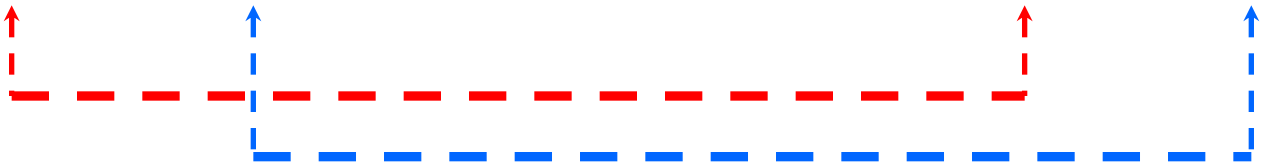
UNION ALL

Employee:

EmpID	Firstname	Sirname
1	Kalle	Karlsson
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

Member:

MemeberID	Age	FName	LName
1	34	Peter	Johansson
2	54	Nisse	Nilsson
3	12	Karl	Svensson
4	32	John	Johnsson



Exempel:

```
SELECT Firstname, Sirname FROM Employee
UNION ALL
SELECT Fname, LName FROM Member
```

Firstname	Sirname
Kalle	Karlsson
Nisse	**Nilsson**
Janne	Jansson
Petter	Karlsson
Peter	Johansson
Nisse	**Nilsson**
Karl	Svensson
John	Johnsson

För att sortera resultatet läggs satsen ORDER BY till sist:

```
...
UNION
SELECT FName, LName FROM Member
ORDER BY Sirname, Firstname
```

INSERT

INSERT INTO använder vi när vi vill lägga in data i tabeller:

Syntax när vi lägger in data i alla fält i och enligt fältordning i tabellen:

```
INSERT INTO tabell
VALUES (värde1, värde2, värde3, ...);
```

Exempel:

```
INSERT INTO Person
VALUES ('Lisa', 'Persson');
```


PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
…	…	…
5	**Lisa**	**Persson**

Fältet PersonID är räknare och det fältet uppdaterar inte vi.
Det gör databasen själv.

OBS!
Ordningen på fälten måste vara den som fälten har i tabellen.

INSERT FORTS…

Syntax när vi endast lägger in data i vissa fält:

```
INSERT INTO tabell (kolumn1, kolumn2, kolumn3, ...)
VALUES (värde1, värde2, värde3, ...);
```

Exempel:

```
INSERT INTO Person (Efternamn)
VALUES ('Bengtsson');
```

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
…	…	…
6		**Bengtsson**

Ordningen och vilka fält som data ska in i bestäms i parentesen på första raden.

OBS!
Om du inte har räknare på Pk måste Pk också uppdateras med ett giltligt värde.

INSERT – MÅNGA POSTER

Du kan med en INSERT INTO lägga till många poster.

Syntax när vi endast lägger in data i vissa fält:

```
INSERT INTO tabell (kolumn1, kolumn2, kolumn3, ...)
VALUES (värde1, värde2, värde3, ...),
       (värde1, värde2, värde3, ...),
       (värde1, värde2, värde3, ...),
       (värde1, värde2, värde3, ...);
```

Exempel:

```
INSERT INTO Person (namn, adress, postnr, Ort)
VALUES ('Bo Svensson', 'Storgatan 23','393 64, 'Kalmar'),
       ('Stina Malm', 'Granvägen 3','393 63, 'Kalmar'),
       ('Anna Svensson', 'Ekstigen 12','393 51, 'Kalmar'),
       ('Ola Ek', 'Ringvägen 93','393 62, 'Kalmar');
```

UPDATE

UPDATE används till att uppdatera/ändra innehållet i tabellerna:

Syntax:

```
UPDATE tabell
SET
kolumn1 = nytt_värde1,
kolumn2 = nytt_värde2,
WHERE kolumn = värde
```

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

Exempel 1 uppdatera ett fält:

```
UPDATE Person
SET Fornamn = 'Carl'
WHERE PersonID = 1;
```


PersonID	Fornamn	Efternamn
1	Carl	Karlsson

UPDATE

UPDATE används till att uppdatera/ändra innehållet i tabellerna:

Exempel 2 uppdatera flera fält:

```
UPDATE Person
SET
Fornamn = 'Carl',
Efternamn = 'Kolberg'
WHERE PersonID = 1;
```

PersonID	Fornamn	Efternamn
1	Carl	Kolberg

VARNING! VARNING!

Exempel 3 ödesdigra konsekvenser:

```
UPDATE Person
SET Fornamn = 'Carl';
```

PersonID	Fornamn	Efternamn
1	Carl	Karlsson
2	Carl	Nilsson
3	Carl	Jansson
4	Carl	Karlsson

Alla får namnet Carl!!!!
Glöm inte WHERE

DELETE

DELETE används när vi vill avlägsna poster från en tabell:

Syntax:

```
DELETE
FROM tabell
WHERE kolumn = värde
```

PersonID	Fornamn	Efternamn
1	Kalle	Karlsson
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

Exempel:

```
DELETE
FROM Person
WHERE PersonID = 1;
```

PersonID	Fornamn	Efternamn
2	Nisse	Nilsson
3	Janne	Jansson
4	Petter	Karlsson

DELETE används för att radera rader i tabeller och inte enstaka fältinnehåll.

När du använder DELETE så bör du tänka flera gånger.
Kontrollera gärna med SELECT och samma sökvillkor först.

DELETE

Det är alltför lätt att göra fel!

Nedanstående exempel tömmer hela tabellen

VARNING! VARNING!

```
DELETE
FROM Person;
```

Om du vill tömma en tabell och återställa räknaren

```
TRUNCATE TABLE Person
```