Föreläsning 5

Dagens föreläsning går igenom

- SQL-sammanfattning
- Komplett exempel, från E/R till tabeller och SQL

SQL

Står för Structured Query Language och används för att hämta, uppdatera, lägga till och radera data och tabeller från en databas.

SQL är standard men existerar i olika variationer för olika databaser (Access, Oracle, MySQL, Sybase osv).

Följande exempel ger en SQL-överblick.

Man skiljer på kommandon för *Data Manipulation* och kommandon för *Data Definition*. I korthet kan man säga att data manipulation innebär att man arbetar med data i tabeller medan data definition innebär att man definierar databasens struktur, dvs vilka tabeller som ska ingå och hur tabellerna ska se ut. SQL-kommandon skrivs vanligtvis med stora bokstäver.

Data Manipulation

Följande fyra kommandon är används för datamanipulation.

SELECT (...FROM...WHERE)
För att hämta och visa information från tabellen.

UPDATE
För att ändra och uppdatera information i tabellen.

DELETE
För att radera information i tabellen.

INSERT INTO
För att lägga till ny information i tabellen.

SELECT

Hämtar rader och kolumner från en eller flera tabeller genom ett antal villkor.

Syntax
SELECT *kolumner (eller * för alla)* FROM *tabell/er [* WHERE *villkor]*

Till exempel:
SELECT kursnamn FROM Kurs WHERE kurskod='K100';
Hämtar fältet (kolumnen) kursnamn från alla poster (raderI i tabellen Kurs där kurskoden är K100

SELECT * FROM Kurs;
Hämtar alla fält från alla poster i tabellen Kurs (eftersom WHERE-villkor saknades)

I villkor används ' ' runt text, dock ej kring heltal (integers)

Jämförelser

Det går att kombinera villkor genom logiska operatorerna (AND OR NOT), samt jämförelse (<, >, <=, >=, <>)

Till exempel:

SELECT fornamn, efternamn FROM Studenter
WHERE postnummer <= 21155 AND ort = ’Malmö’ OR ort = ’Lund’;

Observera apostroferna runt Malmö och Lund (text) men inte runt 21155 (heltal).

ORDER BY

Används för att sortera informationen

SELECT * FROM Kurser ORDER BY kurskod;
Visar all information om alla kurser sorterat i bokstavsordning/nummerordning efter kurskod.

För att vända ordningen används DESC efter sorteringskolumnen;

SELECT * FROM Kurser ORDER BY kurskod DESC;

Mer om SELECT

Det finns ett antal funktioner som går att använda med SELECT;

COUNT (Räknar antalet rader)
MIN (Returnerar det lägsta värdet i kolumnen)
MAX (Returnerar det högsta värdet i kolumnen)
SUM (Returnerar en summa)
AVG (Returnerar ett medeltal)

Till exempel...

SELECT COUNT(*) FROM Studenter;
Returnerar antalet rader i tabellen Studenter.

SELECT MAX(skonummer) FROM Studenter;
Returnerar det största skonumret i tabellen Studenter.

SELECT MIN(skonummer) FROM Studenter;
Returnerar det minsta skonumret i tabellen Studenter.

SELECT SUM(skonummer) FROM Studenter;
Returnerar summan av alla skonummer i tabellen Studenter.

SELECT AVG(skonummer) FROM Studenter;
Returnerar medelvärdet av skonumren i tabellen Studenter.

BETWEEN ... AND ...

Används för att hitta värden inom ett visst spektrum. Returnerar värden mellan och INKLUSIVE de angivna kriterierna.

Syntax
... WHERE Kolumnnamn
BETWEEN Värde1 AND Värde2;

Exempel
SELECT * FROM Score WHERE score
BETWEEN 5000 AND 10000;

Hittar alla förekomster där poängen är mellan 5000 och 10000 inklusive 5000 och 10000.

DISTINCT
för att slippa dubbletter

SELECT Branchname FROM account; ger **Branchname** Crawley Crawley Crawley Bugstone Bugstone	SELECT DISTINCT Branchname FROM account; ger **Branchname** Bugstone Crawley

NOT

Används för att negera ett villkor.

Exempel

SELECT * FROM Account WHERE NOT Branchname=’Crawley’;

UPDATE

Används för att ändra information i tabellen.

UPDATE Tabell SET Kolumnnamn = ’Nytt värde’ WHERE (villkor);

Till exempel

UPDATE Studenter SET gatuadress = ’Amiralsgatan’ WHERE personnummer = ’630126-2351’;

DELETE

Används för att ta bort rader i en tabell efter vissa givna villkor.

DELETE FROM Tabell WHERE villkor;

Till exempel;

DELETE FROM Kurser WHERE Examinator = ’Elisabeth Nilsson’;

tar bort alla rader där Elisabeth Nilsson är examinator.

VARNING

DELETE FROM Tabell;

tar bort alla rader från tabellen!

INSERT INTO

Används för att lägga till nya rader i databasen.

INSERT INTO Tabell VALUES (värde1, värde2);

i detta fall finns det två kolumner i tabellen där det första är Number och det andra Text

Eller

INSERT INTO Tabell
(kolumnnamn1, kolumnnamn2) VALUES (värde1,värde2);

i detta fall kan det finnas fler än två kolumner i tabellen men värden läggs endast till i de två kolmner som angivits. För att lägga till mer information på denna rad måste sedan UPDATE användas – INSERT skapar alltid nya rader!

INSERT INTO (forts)

Till exempel:

```
INSERT INTO Kurser
(kurskod, kursnamn) VALUES
('KK569', 'Media design:1');
```

Nu skapas en ny rad i tabellen Kurser med kurskod KK569 och kursnamn Media design:1 men med ett tomt värde i kolumnen Examinator.

```
INSERT INTO Kurser
VALUES ('KK569', 'Media design:1', 'Simon Niedenthal');
```

Nu skapas en ny rad i tabellen Kurser med kurskod KK569 och kursnamn Media design:1 och examinator Simon Niedenthal.

SQL Data Definition

Används för att bygga databasens struktur

CREATE TABLE
ALTER TABLE
DROP TABLE

CREATE TABLE

Exempel

```
CREATE TABLE Students
(firstname TEXT,
lastname TEXT,
shoesize INT,
id INT PRIMARY KEY NOT NULL)
```

DROP

Tar bort en hel tabell och all data som finns i den.

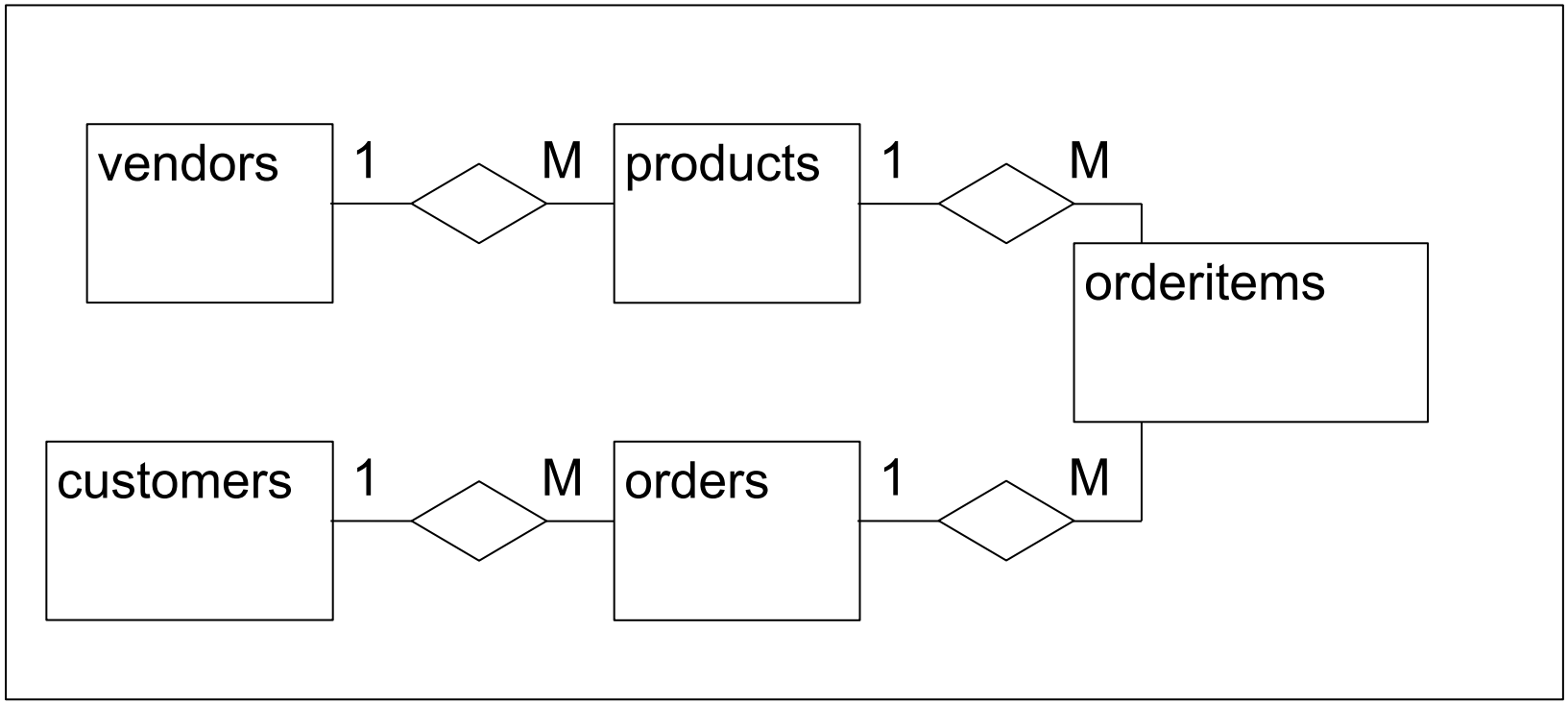
Syntax -
DROP TABLE Tabellnamn;

Till exempel –
DROP TABLE Game;

OBS Tar bort hela tabellen Game och allt som finns i den, så använd med försiktighet…

Ett komplett exempel

Vi kommer att använda en exempeldatabas hämtad från boken SAMS Teach Yourself SQL in 10 minutes med följande ER-diagram (attributen utelämnade).



vendors

vend_id	vend_name	vend_address	vend_city	vend_state	vend_zip	vend_country
BRE02	Bear Emporium	500 Park Street	Anytown	OH	44333	USA
BRS01	Bears R Us	123 Main Street	Bear Town	MI	44444	USA
DLL01	Doll House Inc.	555 High Street	Dollsville	CA	99999	USA
FNG01	Fun and Games	42 Galaxy Road	London		N16 6PS	England
FRB01	Furball Inc.	1000 5th Avenue	New York	NY	11111	USA
JTS01	Jouets et ours	1 Rue Amusement	Paris		45678	France

products

prod_id	vend_id	prod_name	prod_price	prod_desc
BNBG01	DLL01	Fish bean bag toy	3.49	Fish bean bag toy, complete with bean bag worms with which to feed it
BNBG02	DLL01	Bird bean bag toy	3.49	Bird bean bag toy, eggs are not included
BNBG03	DLL01	Rabbit bean bag toy	3.49	Rabbit bean bag toy, comes with bean bag carrots
BR01	BRS01	8 inch teddy bear	5.99	8 inch teddy bear, comes with cap and jacket
BR02	BRS01	12 inch teddy bear	8.99	12 inch teddy bear, comes with cap and jacket
BR03	BRS01	18 inch teddy bear	11.99	18 inch teddy bear, comes with cap and jacket
RGAN01	DLL01	Raggedy Ann	4.99	18 inch Raggedy Ann doll
RYL01	FNG01	King doll	9.49	12 inch king doll with royal garments and crown
RYL02	FNG01	Queen doll	9.49	12 inch queen doll with royal garments and crown

customers

cust_id	cust_name	cust_address	cust_city	cust_state	cust_zip	cust_country	cust_contact	cust_email
1000000001	Village Toys	200 Maple Lane	Detroit	MI	44444	USA	John Smith	sales@villagetoys.com
1000000002	Kids Place	333 South Lake Drive	Columbus	OH	43333	USA	Michelle Green	
1000000003	Fun4All	1 Sunny Place	Muncie	IN	42222	USA	Jim Jones	jjones@fun4all.com
1000000004	Fun4All	829 Riverside Drive	Phoenix	AZ	88888	USA	Denise L. Stephens	dstephens@fun4all.com
1000000005	The Toy Store	4545 53rd Street	Chicago	IL	54545	USA	Kim Howard	

orders

order_num	order_date	cust_id
20005	2001-05-01 00:00:00	1000000001
20006	2001-01-12 00:00:00	1000000003
20007	2001-01-30 00:00:00	1000000004
20008	2001-02-03 00:00:00	1000000005
20009	2001-02-08 00:00:00	1000000001

orderitems

order_num	order_item	prod_id	quantity	item_price
20005	1	BR01	100	5.49
20005	2	BR03	100	10.99
20006	1	BR01	20	5.99
20006	2	BR02	10	8.99
20006	3	BR03	10	11.99
20007	1	BR03	50	11.49
20007	2	BNBG01	100	2.99
20007	3	BNBG02	100	2.99
20007	4	BNBG03	100	2.99
20007	5	RGAN01	50	4.49
20008	1	RGAN01	5	4.99
20008	2	BR03	5	11.99
20008	3	BNBG01	10	3.49
20008	4	BNBG02	10	3.49
20008	5	BNBG03	10	3.49
20009	1	BNBG01	250	2.49
20009	2	BNBG02	250	2.49
20009	3	BNBG03	250	2.49

En massa exempel:

```
#sortering
#sortera dyraste varor först. om två varor har samma pris sorteras de i bokstavsordning

SELECT prod_name, prod_price FROM products ORDER BY prod_price DESC, prod_name;

#filtrering med WHERE

SELECT prod_name, prod_price
FROM products
WHERE prod_price<10;

#söka null-värden:

#följande funkar inte:
SELECT * FROM customers where cust_email='';

#följande funkar inte heller:
SELECT * FROM customers where cust_email=NULL;

#följande funkar däremot:
SELECT * FROM customers where cust_email IS NULL;

#BETWEEN kan användas istället fär >= och <=

SELECT prod_name,prod_price
FROM products
WHERE prod_price BETWEEN 5 AND 10;
```

```
#samma resultat får man med

SELECT prod_name,prod_price
FROM products
WHERE prod_price>=5 and prod_price<=10;

#NOT operatorn
#Observera parenteserna kring det som ska negeras. Det krävs av MySQL men inte av alla DBMS

SELECT prod_name FROM products WHERE NOT (prod_name='King Doll');

#samma sak med <>
SELECT prod_name FROM products WHERE prod_name<>'King Doll';

#LIKE operatorn för att hitta delar av texter
#sök poster som innehåller teddy
SELECT prod_name FROM products WHERE prod_name LIKE "%teddy%";

#sök poster som börjar med bokstaven R
SELECT prod_name FROM products WHERE prod_name LIKE "r%";

#sök poster som slutar med ordet bear
SELECT prod_name FROM products WHERE prod_name LIKE "%bear";

#beräknade fält. det finns ett stort antal funktioner och operator som kan användas,
#tex + - * /  trim() etc.
```

```
SELECT quantity,item_price,quantity*item_price
FROM orderitems
WHERE quantity*item_price>500;

#statistikfunktioner (aggregate functions)
#det finns ett antal funktioner för att beräkna statistik på sökta data, tex count() som räknar
hittade poster och avg() som beräknar medelvärdet av hittade värden och sum() som beräknar
summan.

SELECT AVG(prod_price) AS avg_price FROM products;

SELECT
COUNT(*) AS num_of_itemes,
MIN(prod_price) AS min_price,
MAX(prod_price) as max_price,
AVG(prod_price) as avg_price,
SUM(prod_price) as sum_price,
SUM(prod_price)/COUNT(*) as avg_price_v2
FROM products;


#det kanske viktigaste att förstå är hur man kopplar samman tabeller med join.
#vi har tittat på enkla exempel tidigare. vi börjar med några enkla exempel
#för att sedan gå vidare med mer avancerade.
```

```
#skapa en lista över alla försäljares produkter:

SELECT vend_name,prod_name,prod_price
FROM vendors,products
WHERE vendors.vend_id = products.vend_id;

#det vi gjort kallas ibland inner join, ibland, ibland natural join.
#det finns en alternativ syntax för join. det är en smaksak vilken man använder.

SELECT vend_name,prod_name,prod_price
FROM vendors INNER JOIN products
ON vendors.vend_id = products.vend_id;
```