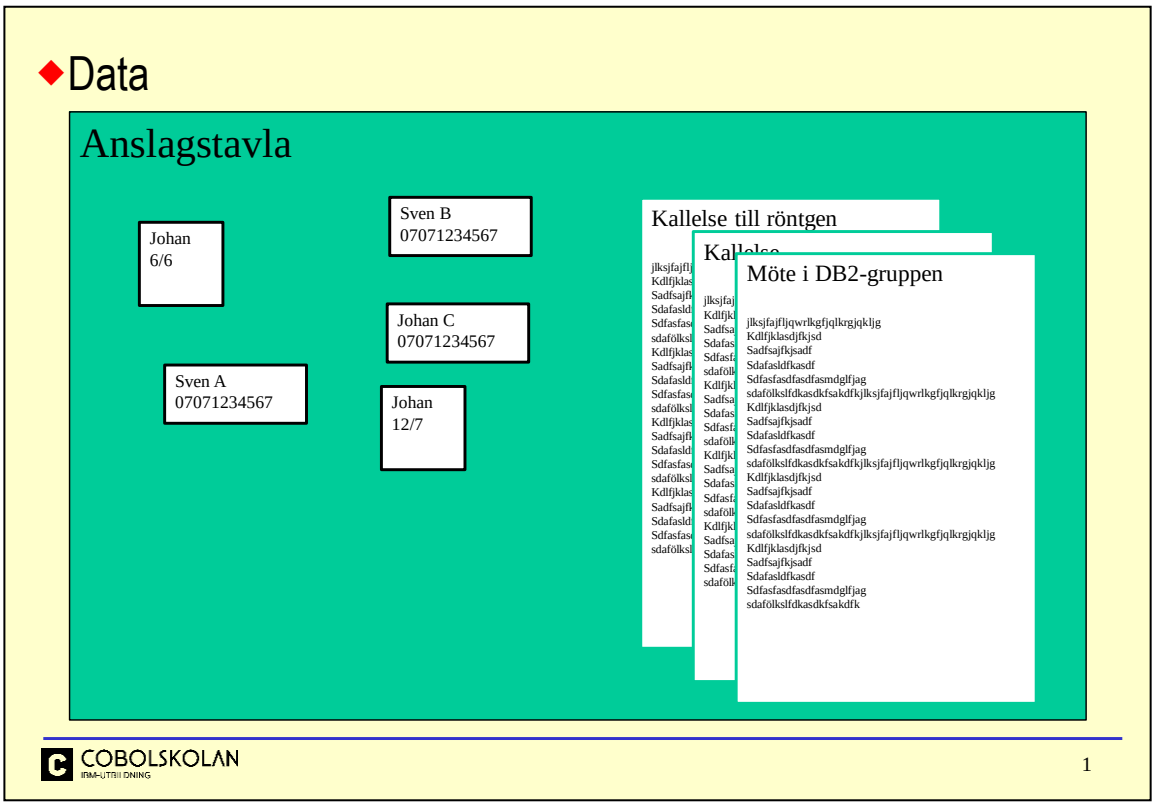
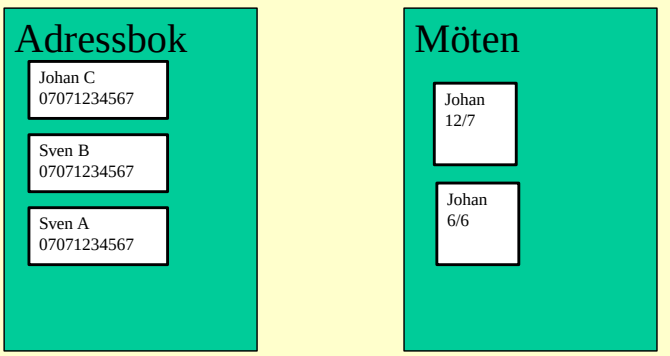


Vi omges överallt av data, allt från små anteckningar på lösa lappar till dokument och hela internet. Detta är ostrukturerad data som vi inte kan använda i våra vanliga applikationer men det är ett stort område för forskning hur man söker i sådant data.

Data

- För att vi ska kunna hantera data på ett strukturerat sätt måste vi ordna det
- Första steget är att samla samma typ av data på ett ställe



COBOLSKOLAN IBM-UTBILDNING 2

Ordna data har vi gjort länge, förr i tiden i böcker i papper eller för större mängder i kartotek; lådor med pappkort med information.

Här ser vi att alla telefonnummer har lagts i adressboken och mötesanteckningar i mötesboken. Det är dock inte sorterat så för att hitta något måste vi läsa/scanna igenom allt.

Databas

- En databas är en ordnad och sökbar samling (strukturerad) data
- Även de gamla pappersadressboken och kalendern är databaser
- En databas är alltså inget som kommit med datorerna utan är ett sätt att hantera data, ju större mängd data desto viktigare

COBOLSKOLAN IBM-UTBILDNING 3

På bilderna ser vi större databaser på kartotekskort. I ett bibliotek var alla böcker förtecknade på kort som var ordnade efter typ av bok samt författare.

Stora företag kunde ha jättelika maskiner som matade fram kartotekslådor.

Databas

- Även om data ligger som poster i en flat fil kan vi uppfatta datat som en tabell vi läser igenom sekvensiellt

Förnamn	Efternamn	Adress	Postnr	Postort	Telefon1	Telefon2	mejladress
Sven	Andersson	Andersg 1	12344	Andersstad	07071234567		
Sven	Brorsson	Brorsg 2	23455	Svenstad	07071234567		
Johan	Classon	Klassegatan 1	34567	Classtad	07071234567		

COBOLSKOLAN IBM-UTBILDNING 4

Den här strukturen kan definieras på många sätt:

En fillayout för ”vanliga” filer, man måste veta hur filen ser ut, var varje fält börjar och hur det är definierat.

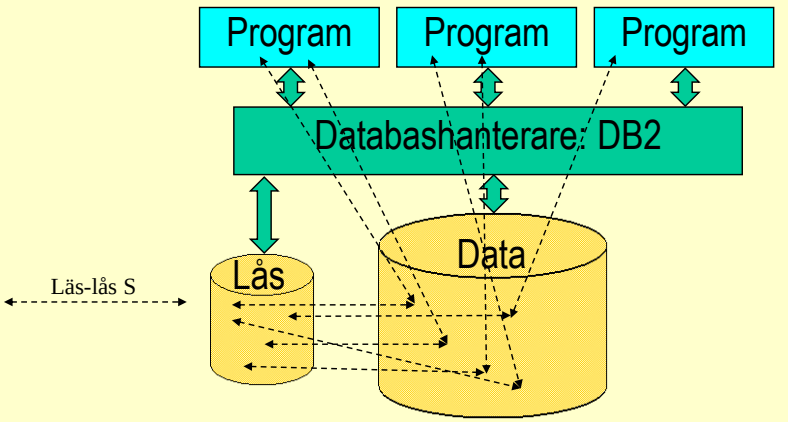
En databashanterare vet hur filen ser ut.

I XML har varje fält en etikett.

Men ändå måste man som användare känna till strukturen!

Vad gör databashanteraren - tillgänglighet

- Samtidig användning – tillgänglighet
- Många program kan läsa samtidigt



COBOLSKOLAN IBM-UTBILDNING 5

Samtidig användning är viktigt annars kan ju bara en i taget använda datat.

Detta hanteras med lås som DB2 sköter om, i det här fallet med läsande program som tas S-lås (Share). Det finns mängder av skilda lås på skilda nivåer i DB2 men vi ska hålla det enkelt.

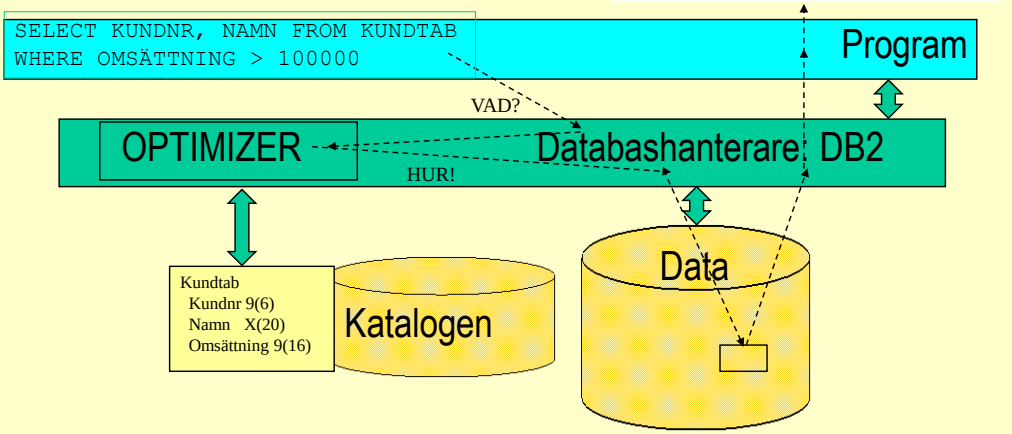
När ett program läser ett visst data tas ett S-lås som hålls tills programmet går vidare eller avslutas. Det finns även varianter där alla lås hålls till programslut, brukar undvikas.

S-låset indikerar att detta data inte får förändras eftersom mitt program är beroende av att denna information är konsistent. Om mitt program t.ex. kontrollerar ett saldo på ett konto eller ett lagersaldo får ju inte detta ändras innan jag är klar.

Hur gör databashanteraren – vet hur data ser ut

Vi frågar med SQL och DB2 tar reda på hur den ska läsa data

KUNDNR	NAMN	OMSÄTTNING
012589	Stora Banken	2374823747
529013	Nisses konsult	5600
854231	Cobolskolan	456256
852136	Monitor IT-utbildning	0



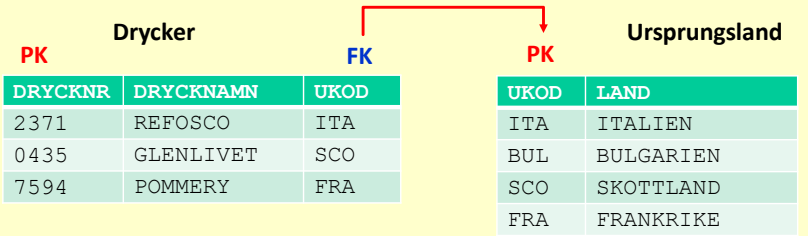
COBOLSKOLAN IBM-UTBILDNING 6

Vår SQL går in i DB2 som kontrollerar syntax, att tabell och kolumner finns och om du/programmet har behörighet att läsa. Därefter gör optimizern en sökvägsanlys för att hitta ett bra sätt att läsa data, man kan säga att den kompilerar din SQL, data läses och formateras så att det kan skickas över till programmet.

Foreign Key

- Foreign Key
  - A *Foreign Key* is an attribute in one relation, whose values are required to match those of the *Primary Key* of some relation.

En främmande nyckel är alltså ett värde som vi kan finna som primärnyckel i en annan tabell. I exemplet nedan har vi en landkod som kan hittas som unik identitet för ett land i tabellen med information om länder.



DRYCKNR	DRYCKNAMN	UKOD
2371	REFOSCO	ITA
0435	GLENLIVET	SCO
7594	POMMERY	FRA

UKOD	LAND
ITA	ITALIEN
BUL	BULGARIEN
SCO	SKOTTLAND
FRA	FRANKRIKE

COBOLSKOLAN IBM-UTBILDNING 7

FOREIGN KEY är enklast uttryckt en kolumn i en tabell som refererar till PRIMARY KEY i en annan (eller samma) tabell.

FOREIGN KEY måste naturligtvis tillhöra samma DOMAIN som den PRIMARY KEY som man refererar till.

Referens från FOREIGN KEY till PRIMARY KEY är den vanligast använda vid JOIN mellan tabeller, även om det inte måste vara så.

Det går i princip att jämföra vilka kolumner som helst, under förutsättning att de tillhör samma DOMAIN.

◆SELECT *

Dryck

DRYCKNR	DRYCKNAMN	UKOD	DSKOD
2202	GIGONDAS	FRA	01
2757	PARADOR	SPA	01
2274	MUSCADET	FRA	11
5200	MORGON	FRA	03

```
SELECT *
FROM UTB00.DRYCK
```

DRYCKNR	DRYCKNAMN	UKOD	DSKOD
2202	GIGONDAS	FRA	01
2757	PARADOR	SPA	01
2274	MUSCADET	FRA	11
5200	MORGON	FRA	03

COBOLSKOLAN IBM-UTBILDNING 8

Detta är den enklaste typen av SELECT, utan några urval. Den ger helt enkelt en kopia av tabellen. Asterisken innebär att *alla kolumner i den ordning de fysiskt ligger i tabellen* fås i resultat-tabellen.

FROM talar om vilken eller vilka tabeller som skall vara med i resultatet.

Tabellnamn består av flera delar: *location.ägare.namn*. När man jobbar interaktivt och med sina egna tabeller behöver man bara ange tabellnamnet, ska man använda någon annans tabell måste man ange bade ägare och tabellnamn. Lokation är ovanligt eftersom det betyder att tabellen finns på en remote site. När man skriver SQL i ett program behöver man inte ange mer än tabellnamnet eftersom ägare (eller prefix) läggs till vid kompilering.

I fortsättningen utesluter vi ägardelen i exemplen.

AS för att namnge kolumner

Pris

DRYCKNR	VOLYM	PRIS
2202	750	100
2274	750	84
2274	500	48
5200	750	109

MINPRIS	MAXPRIS	SUMPRIS	ANTAL	NYKOL
48	109	341	4	A

- MIN (kolumn) ger lägsta värdet
- MAX, SUM, AVG, COUNT

```
SELECT MIN(PRIS) AS MINPRIS, MAX(PRIS) AS MAXPRIS
      ,SUM(PRIS) AS SUMPRIS, COUNT(*)  AS ANTAL
      ,'A' AS NYKOL
FROM PRIS
```

COBOLSKOLAN IBM-UTBILDNING 9

Med AS kan vi döpa (om) kolumner. Ordet AS behöver inte anges utan om man bara skriver SELECT MIN(PRIS) MINPRIS så blir det rätt men lite mindre tydligt.

Man kan även skapa en ny kolumnn med att ange en konstant, text eller numerisk, och helst namnge den.

Subselect

Dryck

DRYCKNR	DRYCKNAMN	UKOD	DSKOD
2202	GIGONDAS	FRA	01
2757	PARADOR	SPA	01
2274	MUSCADET	FRA	11
5200	MORGON	FRA	03

Pris

DRYCKNR	VOLYM	PRIS
2202	750	100
2274	750	84
2274	500	48
5200	750	109

DRYCKNR
2202
2274
2274
5200

DRYCKNR
2202
2274
5200

```
SELECT D.DRYCKNAMN
FROM DRYCK AS D
WHERE D.DRYCKNR IN
   (SELECT P.DRYCKNR
    FROM PRIS AS P)
ORDER BY D.DRYCKNAMN
```

DRYCKNAMN
GIGONDAS
MUSCADET
MORGON

COBOLSKOLAN IBM-UTBILDNING 10

Genom att köra två queries efter varandra, där den första skapar en tabell med alla drycknummer från pristabellen som sedan kan användas som restrict-villkor i den andra queryn, så löser vi problemet med den statiska IN-satsen från föregående sida.

En SUBSELECT är en SELECT som skapar ett mellanresultat som utgör selektionsvillkor för en yttre SELECT.

SELECT-satsema utförs inifrån och utåt. I vårt exempel ger den inre SELECTen en svarstabell med fyra rader med en kolumn.

SUBSELECTer kan *nästas* i godtyckligt antal nivåer och kan skrivas överallt i villkoren där ett värde eller, som här, en tabell kan anges. Svarstabellen från SUBSELECT:en får bara ha en kolumn när vi har en kolumn före IN och om den har noll rader har den "värdet" NULL och vi får därmed inget svar, noll rader, från hela SQL:en.

Vi får inte ha med ORDER BY i en SUBSELECTen. Det är ju den slutliga svarstabellen som vi vill ha presenterad i en viss ordning och ORDER BY skall alltså stå sist i den yttersta SELECT-satsen.

EXISTS, logisk test: finns – finns inte

Vilka butiker har dryck nr 0001 i lager?

Lager

BUTIKSNR	DRYCKNR	VOLYM	ANTAL
01	0001	750	98
01	0020	750	112
02	2274	750	64
02	0001	500	67
03	2268	750	142

Butik

BUTIKSNR	ADRESS
01	BUTIK1
02	BUTIK2
03	BUTIK3

```
SELECT B1.ADRESS
FROM BUTIK AS B1
WHERE EXISTS (SELECT 'X'
              FROM LAGER AS L2
              WHERE L2.BUTIKSNR = B1.BUTIKSNR
              AND   L2.DRYCKNR = 0001)
```

ADRESS
BUTIK1
BUTIK2

COBOLSKOLAN IBM-UTBILDNING 11

Ett av de viktigaste uttrycken i SQL är EXISTS. Den lämnar värdet TRUE eller FALSE beroende på SUBSELECTens resultat. Om villkoren i SUBSELECTen uppfylls fås TRUE annars FALSE.

När man använder EXISTS måste det också finnas en bakåtreferens från den inre till den yttre SELECTen, annars skulle alla rader i den yttre SELECTen vara beroende av samma förutsättningar i SUBSELECTen. Detta skulle innebära att antingen alla eller ingen rad i den yttre SELECTEN fås i slutresultatet.

Eftersom man bara är intresserad av om villkoren i SUBSELECTen är uppfyllda eller inte, så behöver man inte ha en kolumnlista i SELECT-satsen utan kan använda SELECT 'X'.

I detta exempel vill vi veta i vilka butiker drycken med nummer 0001 finns. Vår fråga kan formuleras som:

För varje butik: Är det sant att butiken har dryck med nummer 0001 i lager?

Lägg även märke till namnsättningen av alias: B1 och L2. På detta sätt är det lättare att se på vilken nivå tabellen finns.

Funktioner – exempel stränghantering

Dryck

DRYCKNR	DRYCKNAMN	UKOD	DSKOD
2202	GIGONDAS	FRA	01
2757	PARADOR	SPA	01
2274	MUSCADET	FRA	11
5200	MORGON	FRA	03

```
SELECT CONCAT ( CONCAT (DIGITS(D.DRYCKNR), ' '),
                SUBSTR(D.DRYCKNAMN, 1, 5)) AS TEXT
FROM DRYCK AS D;
```

TEXT
02202 GIGON
02757 PARAD
02274 MUSCA
05200 MORGO

```
SELECT CONCAT ( CONCAT (
       STRIP( DIGITS (DRYCKNR), LEADING, '0')
                        ), ' '),
       SUBSTR(D.DRYCKNAMN, 1, 5)) AS TEXT
FROM DRYCK AS D;
```

TEXT
2202 GIGON
2757 PARAD
2274 MUSCA
5200 MORGO

COBOLSKOLAN IBM-UTBILDNING 12

CONCAT (a, b) lägger ihop två textsträngar.

DIGITS (c) gör om det numeriska fältet till text.

> Om vi använder CHAR (drycknr) får vi resultatet ’02202.’ Decimalpunkten kommer alltså med!

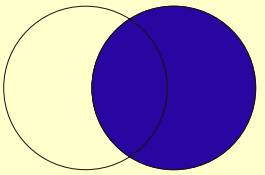
SUBSTR (d, startposition, antal-tecken) tar ut en del av en textsträng.

STRIP (e, LEADING/TRAILING/BOTH, *constant*)

Right Outer JOIN

Dryck

DRYCKNR	DRYCKNAMN	UKOD	DSKOD
2202	GIGONDAS	FRA	01
2757	PARADOR	SPA	01
2274	MUSCADET	FRA	11
5200	MORGON	FRA	03



Pris

DRYCKNR	VOLYM	PRIS
2202	750	100
2274	750	84
2274	500	48
5200	750	109
7030	750	90

DRYCKNAMN	VOLYM	PRIS
GIGONDAS	750	100
MUSCADET	750	84
MUSCADET	500	48
MORGON	750	109
-	750	90

```
SELECT D.DRYCKNAMN, P.VOLYM, P.PRIS
FROM DRYCK AS D RIGHT OUTER JOIN PRIS AS P
  ON D.DRYCKNR = P.DRYCKNR
```

COBOLSKOLAN IBM-UTBILDNING 13

RIGHT join tar med rader (kolumner) från den andra tabellen (högra) och kombinerar med data från den första (vänstra) där det finns någon match. Om detta låter som en omvänd LEFT är det helt rätt. Om det inte finns någon match kommer det ändå med en ej komplett rad där saknat data är NULL (som vi ska gå igenom i detalj senare).

UPDATE

Dryck

DRYCKNR	DRYCKNAMN	UKOD	DSKOD
2202	GIGONDAS	FRA	01
2757	PARADOR	SPA	01
2274	MUSCADET	FRA	11
5200	MORGON	FRA	03

Pris

DRYCKNR	VOLYM	PRIS
2202	750	100
2274	750	84
2274	500	48
5200	750	109

Pris

DRYCKNR	VOLYM	PRIS
2202	750	110
2274	750	84
2274	500	48
5200	750	109

```
UPDATE PRIS
   SET PRIS = PRIS * 1.1
WHERE VOLYM = 750
  AND DRYCKNR IN (SELECT DRYCKNR FROM DRYCK WHERE DSKOD = '01')
```

COBOLSKOLAN IBM-UTBILDNING 14

En UPDATE kan, som all annan SQL, vara hur komplex som helst, här har vi med en enkel IN-sökning mor en subselect.

Eftersom man bara kan uppdatera raderna i en tabell i taget måste vi vid UPDATE konstruera alla villkor på detta vis, vi kan inte använda några konstruktioner med join, union etc. utom i subselecter.

INSERT

Pris

DRYCKNR	VOLYM	PRIS
2202	750	100
2274	750	84
2274	500	48
5200	750	109

Pris

DRYCKNR	VOLYM	PRIS
2202	750	100
2274	750	84
2274	500	48
5200	750	109
2757	750	65
2757	500	45

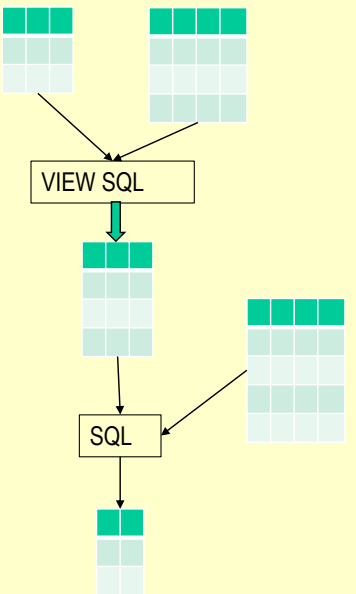
```
INSERT INTO PRIS
   (DRYCKNR, VOLYM, PRIS)
VALUES (2757, 750, 65);

INSERT INTO PRIS
   (VOLYM, DRYCKNR, PRIS)
VALUES (500, 2757, 45);
```

COBOLSKOLAN IBM-UTBILDNING 15

I förra exemplet antog vi att kolumnerna i tabellen låg i en viss ordning men det är alltid säkrare att tala om vilka kolumner man avser.

VIEW

- Vyer och tabeller
- Vyer är fördefinierade SQL-satser
- Definitionerna lagras i DB2
- Körs när de används
- DB2 skriver om din SQL tillsammans med SQL:en i vydefinitionen



COBOLSKOLAN IBM-UTBILDNING 16

En vy definieras som en sparad SQL-sats, denna lagras internt i DB2:s katalog och refereras till som vilken tabell som helst i din SQL. DB2 kommer att försöka skriva om din SQL tillsammans med vydefinitionens SQL. Konceptuellt kan man anse att DB2 först kör vyn och sedan använder resultatet som en ny tabell. I många fall stämmer detta, framför allt för mer komplicerade vyer men DB2 försöker göra exekveringen så effektiv som möjligt.

Nested Table

Dryck

DRYCKNR	DRYCKNAMN	UKOD	DSKOD
2202	GIGONDAS	FRA	01
2757	PARADOR	SPA	01
2274	MUSCADET	FRA	11
5200	MORGON	FRA	03

Pris

DRYCKNR	VOLYM	PRIS
2202	750	100
2274	750	84
2274	500	48
5200	750	109

```
SELECT DRYCKNR, DRYCKNAMN
FROM DRYCK
HERE UKOD = 'FRA'
```

DRYCKNR	DRYCKNAMN
2202	GIGONDAS
2274	MUSCADET
5200	MORGON

```
SELECT FRANSKA.DRYCKNAMN, P.VOLYM, P.PRIS
FROM (SELECT DRYCKNR, DRYCKNAMN FROM DRYCK
      WHERE UKOD = 'FRA') AS FRANSKA
   JOIN PRIS AS P
   ON FRANSKA.DRYCKNR = P.DRYCKNR
```

DRYCKNAMN	VOLYM	PRIS
GIGONDAS	750	100
MUSCADET	750	84
MUSCADET	500	48
MORGON	750	109

COBOLSKOLAN IBM-UTBILDNING 17

Tidigare hade vi detta exempel där vi kunde skapa en tabell ”i flykten” och selektera från den.

WITH – Common Table Exression

```
WITH PRISDAT (DRYCKNR, VOLYM, MAXDAT) AS
  (SELECT P2.DRYCKNR ,P2.VOLYM ,MAX(P2.PRDATUM)
   FROM UTB00.PRIS P2
   WHERE P2.PRDATUM <= '2014-02-01'
   GROUP BY P2.DRYCKNR ,P2.VOLYM)
SELECT L.DRYCKNR ,L.VOLYM ,L.ANTAL ,P.PRIS
      ,L.ANTAL * P.PRIS AS VARDE
FROM   UTB00.LAGER L
  JOIN UTB00.PRIS P
    ON    L.DRYCKNR = P.DRYCKNR
      AND L.VOLYM   = P.VOLYM
WHERE (P.DRYCKNR, P.VOLYM, P.PRDATUM) IN
   (SELECT DRYCKNR ,VOLYM ,MAXDAT FROM PRISDAT)
  AND L.BUTIKSNR = '01'
UNION ALL
SELECT 0 AS DRYCKNR, 0 AS VOLYM, 0 AS ANTAL, 0 AS PRIS
      ,SUM(VARDE) AS VARDE
FROM
(SELECT (L.ANTAL * P.PRIS) AS VARDE
 FROM UTB00.LAGER L
 JOIN UTB00.PRIS P ON L.DRYCKNR = P.DRYCKNR
                  AND L.VOLYM = P.VOLYM
 WHERE (P.DRYCKNR, P.VOLYM, P.PRDATUM) IN
   (SELECT DRYCKNR ,VOLYM ,MAXDAT FROM PRISDAT)
   AND L.BUTIKSNR = '01'
  ) AS XTAB
ORDER BY DRYCKNR, VOLYM
```

COBOLSKOLAN IBM-UTBILDNING 18

Här ser vi hur man kan förenkla SQL genom att bryta ut gemensamma delar och lägga dessa som WITH.

SQLCODE

- SQLCODE:

  DB2 returns the following codes in SQLCODE:

  If SQLCODE = 0, execution was successful.

  If SQLCODE > 0, execution was successful with a warning.

  If SQLCODE < 0, execution was not successful.

- SQLCODE +100 anger att inget data fanns att hämta.
  - SQLSTATE '02000' är samma som ovan

COBOLSKOLAN IBM-UTBILDNING 19

Returkoden från DB2 måste man alltid testa på, allt utöver 0 (noll) måste tas omhand. Efter t.ex. en returkod +100 har man inte fått något data men vad som finns i areorna är odefinierat; kan variera mellan releaser av DB2.

SQLCODE är det klassiska fältet man testar på men SQLSTATE ger samma returkoder för alla varianter av DB2. SQLCODE gäller bara för z/OS så ska koden kunna köras på andra plattformar är det bäst att använda SQLSTATE. SQLCODE ger däremot mer detaljerad information.

Vanliga SQL-koder

- `-803 AN INSERTED OR UPDATED VALUE IS INVALID BECAUSE THE INDEX IN INDEX SPACE indexspace-name CONSTRAINS COLUMNS OF THE TABLE SO NO TWO ROWS CAN CONTAIN DUPLICATE VALUES IN THOSE COLUMNS. RID OF EXISTING ROW IS X record-id`
  En rad med denna nyckel finns redan i tabellen. `indexspace-name` är ett unikt index som redan innehåller detta värde.
- `-811 THE RESULT OF AN EMBEDDED SELECT STATEMENT OR A SUBSELECT IN THE SET CLAUSE OF AN UPDATE STATEMENT IS A TABLE OF MORE THAN ONE ROW, OR THE RESULT OF A SUBQUERY OF A BASIC PREDICATE IS MORE THAN ONE VALUE`
  En select eller subselect som borde ge bara en rad ger ett svar med flera rader vilket inte kan tas emot.
- `-904 UNSUCCESSFUL EXECUTION CAUSED BY AN UNAVAILABLE RESOURCE. REASON reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name.`
  Någonting i DB2 är inte tillgängligt, det kan vara något som är stoppat.
- `-911 THE CURRENT UNIT OF WORK HAS BEEN ROLLED BACK DUE TO DEADLOCK OR TIMEOUT. REASON reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name`
  Programmet har råkat ut för en ROLLBACK p.g.a. en låsningskonflikt.
- `-913 UNSUCCESSFUL EXECUTION CAUSED BY DEADLOCK OR TIMEOUT. REASON CODE reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name`
  Programmet avbröts/kunde inte köra p.g.a. en låsningskonflikt.

COBOL Declaration for table

```
*****************************************************************
* COBOL DECLARATION FOR TABLE DRYCK                             *
*****************************************************************
 01  DCLDRYCK.
     10 DRYCKNR              PIC S9(5)V USAGE COMP-3.
     10 DRYCKNAMN            PIC X(30).
     10 UKOD                 PIC X(3).
     10 DSKOD                PIC X(2).
     10 DISTRIKT             PIC X(20).
     10 ARGANG               PIC X(4).
     10 TILLVERKARE          PIC X(20).
     10 DRBESKR              PIC X(200).
*****************************************************************

*****************************************************************
* COBOL DECLARATION FOR TABLE PRIS                              *
*****************************************************************
 01  DCLPRIS.
     10 DRYCKNR              PIC S9(5)V USAGE COMP-3.
     10 VOLYM                PIC S9(5)V USAGE COMP-3.
     10 PRIS                 PIC S9(5)V9(2) USAGE COMP-3.
     10 PRDATUM              PIC X(10).
*****************************************************************
```

COBOLSKOLAN IBM-UTBILDNING 21

Detta är en vanlig COBOL-deklaration för areorna vi använder när vi läser eller skriver tabellen DRYCK. Deklarationerna blir automatiskt rätt när vi använder DCLGEN.

DATATYPER NUMERISKA

DB2	COBOL
SMALLINT	Pic S9(4) Binary
INTEGER	Pic S9(9) Binary
BIGINT	Pic S9(18) Binary
DECIMAL(11,2)	Pic S9(9)V99 Comp-3 (Packed-Decimal)
REAL	Comp-1
DOUBLE	Comp-2

- Comp-3 / Packed-Decimal är det klassiska numeriska formatet i COBOL. Det är ett packat numeriskt format där varje siffra ligger i textformat samt en halvbyte sist för tecken.

COBOLSKOLAN IBM-UTBILDNING 22

SMALLINT är binärlagrat och kan lagra -32768 till +32767. Tar 2 byte i lagring.

INTEGER är binärlagrat och kan lagra -2 147 483 648 till +2 147 483 647. Tar 4 byte i lagring.

BIGINT är binärlagrat och kan lagra -9 223 372 036 854 775 808 till +9 223 372 036 854 775 807. Tar 8 byte i lagring.

DECIMAL(*p*,*s*) är den traditionellt vanligaste lagringen av numeriskt data i COBOL-sammanhang. Lagringen sker i ett packat textformat där varje byte rymmer två siffror plus att en halv byte används för tecknet. Antalet siffror i talet är *p* som står för precisionen och *s* som är skalningen, d.v.s. hur många av siffrorna som är decimaler. Antalet siffror kan vara 1-31 och antalet decimaler därav kan vara 0-31.

Dessutom finns det varianter på **flyttal** men dessa är ovanliga i COBOL-sammanhang.

Binary och Comp som det kallas i äldre program är samma sak, binärlagrat.

Singel SQL

- Singel SQL returnerar endast en rad
- Här hämtar vi in dagens datum till den egendefinierade arean ”Dagens-Datum”.

```
01 Ws-DagensDatum      Pic X(10).

EXEC SQL
  SELECT CURRENT DATE
    INTO :Ws-DagensDatum
  FROM SYSIBM.SYSDUMMY1
END-EXEC
```

- Current date/time/timestamp/timezone

COBOLSKOLAN IBM-UTBILDNING 23

Om man bara ska hämta en rad från en tabell kan man skriva en singel-SQL. Den får inte ge fler än en rad i svar; om detta inträffar får man felkod -811.

SYSIBM.SYSDUMMY1 är en specialtabell i DB2 som bara innehåller en rad och som kan användas i uttryck av denna typ. (Motsvarande funktion i Oracle heter DUAL.)

Hostvariabler

- En hostvariabel är ett vanligt COBOL-fält som refereras i SQL

```
01 Ws-DagensDatum      Pic X(10).
01 Ws-DagensTid        Pic X(8).

EXEC SQL
  SELECT CURRENT DATE
    INTO :Ws-DagensDatum
  FROM SYSIBM.SYSDUMMY1
END-EXEC

EXEC SQL
  SELECT CURRENT DATE, CURRENT TIME
    INTO :Ws-DagensDatum, :Ws-DagensTid
  FROM SYSIBM.SYSDUMMY1
END-EXEC
```

COBOLSKOLAN IBM-UTBILDNING 24

Vi har ju redan sett hur man deklarerar tabellens kolumner i COBOL. I SQL måste vi ange till vilken Hostvariabel vi ska få data från en kolumn till. En hostvariabel är alltså ett vanligt deklarerat fält i COBOL som refereras i SQL.

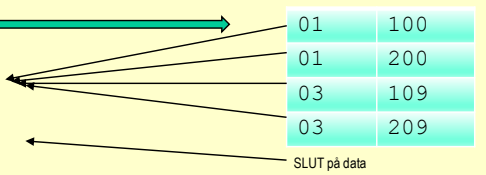
I första exemplet hämtar vi bara en kolumn till en variabel.

I andra exemplet visas hur man anger flera kolumner och flera variabler.

CURSOR

- För att läsa in mer än en rad i ett program använder vi CURSOR
- Det är en deklarerad SQL-sats som vi läser ifrån
- Hantering sker som om det var en fysisk fil

- OPEN
- FETCH (som en READ)
- RETURKOD = +100
- CLOSE



COBOLSKOLAN IBM-UTBILDNING 25

Det vanligaste sättet att hantera flera rader i svaret är att använda en CURSOR. Det är en pekare i DB2 som håller reda på vari svarstabellen man är. En Cursor deklareras i programmet, har ett namn och innehåller en SQL-sats.

Denna beskrivning är bara på en övergripande konceptuell nivå.

När man ska använda resultatet öppnar man först cursorn. Då exekveras SQL-satsen och kan få tillgång till raderna. För att läsa in raderna i programmet använder man FETCH. Varje fetch läser in en ny rad till man får reurkod -100 som visar att man har kommit till slutet. Därefter kan man stänga cursorn. Vid programslut eller annan commit stängs (normalt) cursorn.

DECLARE CURSOR, *row-positioned*

- Definierar cursorn C1_DRYCK
- Namnet är unikt i programmet och används i OPEN, FETCH etc.

```
EXEC SQL
   DECLARE C1_DRYCK CURSOR FOR
   SELECT DRYCKNR
         ,DRYCKNAMN
         ,ARGANG
         ,TILLVERKARE
   FROM   DRYCK
   ORDER BY DRYCKNR
END-EXEC
```

COBOLSKOLAN IBM-UTBILDNING 26

I DECLARE CURSOR skriver vi en SQL-sats som vi skulle kunna köra interaktivt. DECLARE CURSOR är ingen SQL som körs utan ”bara” en definition. Den kan därför även ligga i WORKING STORAGE. Ett krav är dock att den måste ligga före alla referenser till den.

FETCH

- Med FETCH läser vi in en rad i taget

```
EXEC SQL
   FETCH C1_DRYCK
   INTO :Drycknr
       ,:Drycknamn
       ,:Argang
       ,:Tillverkare
END-EXEC
```

COBOLSKOLAN IBM-UTBILDNING 27

Vid varje FETCH läser en ny rad in till de angivna hostvariablerna. Eftersom selecten och fetchen står åtskilda är det lätt hänt att man får mismatch mellan kolumnerna vid ändringar. Kan ge svårfunna fel om man har många likanande fält, t.ex. många beloppsfält med samma deklaration. Därför är det bra att skriva välstrukturerat.

DB2 – uppbyggnad

- Fysiska objekt, t.ex.
  - Tablespace – där vårt data lagras
  - Indexspace – index för att garantera unikhet och snabba upp access
  - Loggar – för att möjliggöra rollback
  - Arkivloggar – för att spara loggar längre tid
- Logiska objekt t.ex.
  - Behörigheter – för att göra något i DB2 behöver du behörighet
  - VIEW – en logisk alternativ bild av en (eller flera) tabeller
  - Katalogen – beskriver allt i DB2
  - Bufferpooler – så DB2 inte behöver läsa/skriva direkt på ”disk”

COBOLSKOLAN IBM-UTBILDNING 28

Det mesta i DB2 är logiska definitioner för objekt, t.ex. en tabell finns inte i verkligheten det är bara en definition i katalogen som gör att DB2 kan tyda fysiskt data och hantera det som om det vore en tabell.

Behörighet finns på alla nivåer och för alla objekt. Behörighet kan ges väldigt granulerat eller på ett antal övergripande nivåer. Har du fått skapa ett objekt är du ägare och har implicit behörighet på detta objekt.

Läsningar i DB2

- Tablespace scan – läser hela tabellens data
- Index scan – läser ett helt index
- Index only – allt du behöver finns i indexet
- Index filtering – läser i index för att välja vilka rader som ska läsas i tablespace
- Index -> table page – normal läsning: via index till TS

COBOLSKOLAN IBM-UTBILDNING 29

Tablespace Scan läser alltså hela tabellen från början till slut för att hitta det sökta datat.

Index Scan läser ett helt index för att hitta data eller sökta nycklar, t.ex. om filtreringen sker på kolumn som inte ligger först i indexet.

Index Only betyder att allt du vill ha som svar finns i indexet.

Index Filtering betyder att DB2 läser i index för att hitta nycklarna till data och sedan gå vidare till Tabellen.

Katalogen

Authorization options	Plans
Storage groups	Collections
Databases	Packages
Table spaces	DBRMs
Tables, views, and aliases	Schemas
Views	User defined data types
Aliases	Functions
Synonyms	Stored procedures
Indexes	Triggers
Columns	Sequences
Constraints	

EX: SYSIBM.SYSTABLES

COBOLSKOLAN IBM-UTBILDNING 30

I DB2:s katalog finns det information om allt! Det är där DB2 hanterar sitt metadata, all statistik, information om behörigheter etc.