Att skydda databasen

Mot vad/vem?

- Fel i indata
- Fel i mjukvara som leder till korrumperad data.
- Fel i hårdvara som leder till korrumperad data.
- Brytande mot generella policyn
- Illvilliga, elaka, ondskefulla, lömska, skadelystna, infernaliska, malevolenta, illasinnade och kanske helt enkelt fientliga användare.

Hur?

- Inför säkerhetsåtgärder.
- Inför integritetsvillkor för databasens innehåll.

Säkerhet

Många av säkerhetsfrågor som förekommer i samband med databaser är inte unika till databaser utan förekommer i andra system också (t.ex. i operativsystem).

Säkerhetsåtgärder:

Identifiering av användare:
Olika användare -> olika accessrättigheter.

Auktorisering till olika **systemnivåer:**
T.ex. DBMS systemet, DBA, SQL användare

Vissa tillämpningar kräver multipla **säkerhetsnivåer** varvid olika säkerhetspolicyn tillämpas på olika nivåer:
T.ex. top secret, secret, confidential, unclassified.

Säkerhet

Skydd mot **missbruk av statistisk i databaser** så att man ej kan komma åt otillbörlig information med statistisk inferens.

Kryptering av data som transporteras över kommunikationsnätverk.

Fysiska skydd: Lås in skivminnet/backup kopior i säkra rum.

Skydd mot virus, brandväggar: Så att 'trojanska hästar' inte släpps in.

DBA

Databasadministratören är ansvarig för hantering av databasens säkerhet:

- Skapa konton, användargrupper och lösenord
- Ge privilegier
- Ta tillbaks privilegier
- Tilldela säkerhetsnivåer

Logg

Databasloggen måste innehålla användardata.

Audit trail:
Logg som innehåller information för säkerhetshantering och efteranalys. Alla DBMS loggar händelser för att kunna återskapa databasen efter krasher. DBMS loggen kan också användas för att undersöka vem som gjort en viss transaktion, etc.

SQL-99 Privilegier

Access-matris:

M(s,o) -> p

där
s är subjekt (användare, grupp, konton, program)
o objekt (relation eller vy, kolumn)
p är privilegietyp (läsning, uppdatering).

Varje relation är ägd av ett konto
T.ex. kontot som skapade relationen.

Ägaren har fullständiga rättigheter

Ägaren kan delegera rättigheter till andra subjekt.

SQL-99 Privilegier Exempel

DBA:

CREATE SCHEMA EXAMPLE **AUTHORIZATION** A1**;**

A1:

CREATE TABLE EMPLOYEE**(...)**
CREATE TABLE DEPARTMENT**(...)**

GRANT: Delegera privilegier till subjekt (dvs. sätt element i accessmatrisen)
Syntax:

GRANT privilegietyper **ON** objekt **TO** subjekt

SQL-99 Privilegier Exempel

Exempel:

A1:

GRANT INSERT,DELETE **ON** EMPLOYEE,DEPARTMENT **TO** A2

OBS: A2 kan ej vidarebefodra privilegier

GRANT SELECT **ON** EMPLOYEE,DEPARTMENT **TO** A3 **WITH GRANT OPTION;**

=> A3 kan vidarebefodra privilegier till andra konton.

SQL-99 Privilegier Exempel

A3:

GRANT SELECT **ON** EMPLOYEE **TO** A4

A2:

REVOKE SELECT **ON** EMPLOYEE **FROM** A3

=> A4 kan ej heller accessa EMPLOYEE!

- **GRANT** och **REVOKE** kan appliceras också på vyer.
- Man kan få privilegier från mer än en källa
  => faktiska privilegier = unionen av alla erhållna privilegier

Multi-level security

Säkerhetsklasser: TS (Top Secret), S (secret), C (Confidential), U (Unclassified)

Utvidgad accessmatris:

M(s,o) -> <p,c>

s är subjekt (användare, konton program)
o är objekt (relation eller vy, record, kolumn, vy konton, program)
c är säkerhetsklass

Klassificering av subjekt/objekt:

- class(s) ≥ class(o)

Vyer som säkerhetsåtgärd

- Vyer ger avancerad skyddsmekanism
- Omvandling av frågor för specifika användare
  T.ex. Lägg till en selektion och projektion till varje fråga ICA-anställda ställer:

DBA:

```
CREATE TABLE SUPPLIES(
        STORE CHAR,
        ITEM CHAR,
        PRICE DECIMAL(10,2),
    PRIMARY KEY(STORE, ITEM))

CREATE VIEW ICASUPPLIES AS
        SELECT *
        FROM SUPPLIES
        WHERE STORE = 'ICA'

GRANT SELECT, INSERT, DELETE ON ICASUPPLIES
TO ICANDER
```

Vyer som säkerhetsåtgärd

ICANDER kan ej komma åt SUPPLIES, bara ICASUPPLIES:

```
SELECT PRICE
    FROM ICASUPPLIES S
    WHERE S.ITEM = 'Tomater'
```

översätts av frågeprocessorn till

```
SELECT PRICE
    FROM SUPPLIES S
    WHERE S.ITEM = 'Tomater'
        AND S.STORE = 'ICA'
```

- Kan ha avancerade säkerhetspolicies med vyer
- Observera dock att vyer inte alltid uppdaterbara
- Nyckeln i bastabell måste inkluderas i vy definitionen

Integritetsvillkor ‘Integrity constraints’

Sec 7.2, 7.3, 8.1.2

Exempel på intgritetsvillkor på enkla attributvärden som hanteras av SQL-99:

- Format på inbyggda datatyper, t.ex. tal och tid
- Format på användardefinerade domäner (datatyper)
  T.ex.
  CREATE DOMAIN DOLLAR **AS DECIMAL(**10,2**)**
- Möjlighet att a begränsningar på domänvärden (sec 8.6).
  T.ex.
  CREATE DOMAIN SALARY **AS DECIMAL(**10,2**)**
  CHECK(SALARY > 10000**)**

Integritetsvillkor ‘Integrity constraints’

- Möjlighet att ha NULL-värden i fält
  T.ex.
  CREATE TABLE DEPT_LOCATIONS
  (DNUMBER **INT NOT NULL,**
  DLOCATION **VARCHAR(**15**) NOT NULL)**
- Möjlighet att ha defaultvärden på kolumner:
  CREATE TABLE EMPLOYEE
  (SSN **CHAR(**9**) NOT NULL,**
  DNO **INT NOT NULL DEFAULT** 1**)**

Integritetsvillkor ‘Integrity constraints’

Definition av nycklar i SQL-99:

- Nycklar kan specificaras för varje relation:
  CREATE TABLE DEPT_LOCATIONS
  (DNUMBER **INT NOT NULL,**
  DLOCATION **VARCHAR(**15**) NOT NULL,**
  PRIMARY KEY(DNUMBER, DLOCATION**))**
- Sekundärnycklar kan också specificeras:
  CREATE TABLE DEPARTMENT
  (DNAME **VARCHAR(**15**) NOT NULL,**
  DNUMBER **INT NOT NULL,**
  PRIMARY KEY (DNUMBER)
  UNIQUE(DNAME))
  Notera: UNIQUE anger secondary key.

Integritetsvillkor ‘Integrity constraints’

Referenser (‘relationships’) mellan tabeller kan defineras i SQL-99:

- Främmande nycklar:

```
CREATE TABLE DEPT_LOCATIONS
    (   DNUMBER INT NOT NULL,
        DLOCATION VARCHAR(15) NOT NULL,
    PRIMARY KEY(DNUMBER, DLOCATION),
    FOREIGN KEY(DNUMBER)
        REFERENCES DEPARTMENT(DNUMBER))
```

- *Referentail integrity* kan upprätthållas:

```
CREATE TABLE DEPT_LOCATIONS
    (   DNUMBER INT NOT NULL,
        DLOCATION VARCHAR(15) NOT NULL,
    PRIMARY KEY(DNUMBER, DLOCATION),
    FOREIGN KEY(DNUMBER)
        REFERENCES DEPARTMENT(DNUMBER)
        ON DELETE CASCADE
        ON UPDATE CASCADE)
```


Integritetsvillkor ‘Integrity constraints’

Exempel 2, namngivet referential integrity:

```
CREATE TABLE EMPLOYEE
    (   SSN CHAR(9) NOT NULL,
        DNO INT NOT NULL DEFAULT 1,
    PRIMARY KEY(SSN)
    CONSTRAINT EMPDEPTFK
        FOREIGN KEY(DNO)
        REFERENCES DEPARTMENT(DNUMBER)
        ON DELETE SET DEFAULT
        ON UPDATE CASCADE)
```

Integritetsvillkor
‘Integrity constraints’

Deklarativa integritetsvillkor (*assertions*).

T.ex. ‘Inkomsten av en anställd får aldrig överstiga inkomsten av chefen för avdelningen där han/hon arbetar’

SQL-99:

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK(NOT EXISTS
        (SELECT *
            FROM EMPLOYEE E, EMPLOYEE M,
                DEPARTMENT D
            WHERE E.SALARY > M.SALARY AND
                E.DNO = D.DNUMBER AND
                D.MGRSSN = M.SSN))
```

Mer om detta när vi diskuterar *Aktiva Databaser*