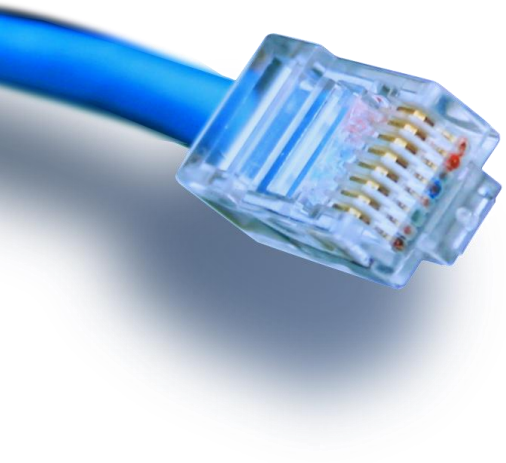
HOUSE OF TECHNOLOGY

– en del af mercantec+



Views etc.

Databaser

Views

Med Views kan vi gemme nogle af de lange select sætninger.

I vores eksempel fra tidligere er det f.eks. forbundet med en del besvær at finde telefon nr og bilmærker for en sælger da det kræver to JOINs.

Ved at lave et View kan man gemme SELECT sætningen en gang for alle sådan her

```
Create View SaelgerView AS
SELECT MedarbNr,Navn,StartDato,Email,Nr AS TelefoonNr, RegNr AS Bil,Maerke,RegAar AS BilRegAar FROM Saelger
JOIN Bil ON Saelger.MedarbNr = bil.Saelger
JOIN TelefonNr ON TelefonNr.Saelger = Saelger.MedarbNr;
```

Sælger
Navn
AnsættelsesDato
Email
MedarbejderNr
TelefonNr

Bil
RegistreringsNr
Type
RegistreringsÅr
Sælger (FK)

TelefonNr
Nr
Sælger (FK)

Efterfølgende gør man blot sådan for at anvende viewet

```
Select * from SaelgerView;
```

Resultat

	MedarbNr	Navn	StartDato	Email	TelefoonNr	Bil	Maerke	BilRegAar
1	1	Anders	2001-05-01 00:00:00.000	anders@mail.dk	23546476	AA12345	Opel	2003-04-02 00:00:00.000
2	2	Charly	2003-05-01 00:00:00.000	charly@mail.dk	45653465	BB12345	Bentley	1996-04-12 00:00:00.000
3	1	Anders	2001-05-01 00:00:00.000	anders@mail.dk	54674323	AA12345	Opel	2003-04-02 00:00:00.000

Insert via View

HOUSE OF TECHNOLOGY

- Bemærk at et View kun giver et udtræk af databasens tabeller og ikke er en tabel i sig selv
- Derfor er det normalt ikke muligt at anvende insert og delete på views, dog er der visse undtagelse hvor det er muligt, men det vil jeg ikke komme nærmere ind på her

Opgave

- Lav views som viser Kunder incl. oplysninger om hvilken sælger de betjenes af og sælgerens navn og telefon nummer
- Lav view som viser varer og hvilken kunde de købes af

ON DELETE CASCADE

```
CREATE TABLE Saelger(
        Navn NVARCHAR(20)NOT NULL,
        StartDatoDATETIMENOT NULL,
        EmailNVARCHAR(20)NOT NULL,
        MedarbNrINTNOT NULL,
        PRIMARY KEY(MedarbNr)
);
CREATE TABLE TelefonNr(
        Nr NVARCHAR(8) NOT NULL,
        SaelgerINT NOT NULL,
        PRIMARY KEY(Nr),
        FOREIGN KEY(Saelger) REFERENCES Saelger(MedarbNr) ON DELETE Cascade
);

INSERT INTO Saelger (Navn, StartDato,Email,MedarbNr) VALUES ('Hans','2001-1-
1','Hans@mail.dk',1);
INSERT INTO TelefonNr (Nr, Saelger) VALUES ('12345678',1);

DELETE FROM Saelger WHERE MedarbNr = 1;
```

I eksemplet her laves først tabellen sælger og derefter TelefonNr der har Saelger som FK.

Efterfølgende indsættes en sælger og en telefon med reference til sælgeren hvorefter vi forsøger at slette sælgeren.

Normalt vil dette ikke være muligt, men fordi vi har indsat ON DELETE Cascade vil den blot slette de numre som referere til Sælgeren når Sælgeren slettes.

Opgave

- Forsøg på sælger databasen at slette en Vare
- Ret i databasen ved at tilføje de nødvendige ON DELETE sådan at en Vare kan slettes uden problemer
- Ret derefter også sådan at en Sælger kan slettes uden problemer. Dette er nok ikke praktisk i virkeligheden men prøv alligevel

SELECT DISTINCT

Afprøv selv forskellen på

```
SELECT Navn FROM KoebesAf
JOIN Kunde ON Kunde.KundeNr = KoebesAf.Kunde
```

OG

```
SELECT DISTINCT Navn FROM KoebesAf
JOIN Kunde ON Kunde.KundeNr = KoebesAf.Kunde
```

IDENTITY, AUTO_INCREMENT

```
/* Opret tabellen Saelger */
CREATE TABLE Saelger
(
    Navn        NVARCHAR(20)    NOT NULL,
    StartDato   DATETIME        NOT NULL,
    Email       NVARCHAR(20)    NOT NULL,
    MedarbNr    INT             IDENTITY ,//Hedder AUTO_INCREMENT i MySQL
    PRIMARY KEY(MedarbNr)
);

INSERT INTO Saelger (Navn,StartDato,Email) VALUES ('Ole','1-1-1','ole@mail.dk');
```

Med IDENTITY kan man bestemme at databasen selv skal tælle værdien MedarbNr op hver gang der indsættes en ny.

Bemærk at MedarbNr nu ikke længere er med.

Unique

```
/* Opret tabellen Saelger */
CREATE TABLE Saelger
(
    Navn            NVARCHAR(20)    NOT NULL,
    StartDato       DATETIMENOT     NULL,
    Email           NVARCHAR(20)    NOT NULL,
    MedarbNr        INT             NOT NULL,
    UNIQUE (Navn),
    PRIMARY KEY(MedarbNr)
);

/* Opret tabellen TelefonNr */
CREATE TABLE TelefonNr
(
    Nr              NVARCHAR(8)     NOT NULL,
    SaelgerNavn     NVARCHAR(20)    NOT NULL,
    PRIMARY KEY(Nr),
    FOREIGN KEY(SaelgerNavn) REFERENCES Saelger(Navn)
);
```

Med UNIQI kan vi bestemme at en attribut skal være unik selv om den ikke er nøgle attribut. Derved kan vi anvende den som reference til foreign key

Stored procedure

```
CREATE PROCEDURE InsertSaelger @navn NVARCHAR(20), @email NVARCHAR(20),
@startDato DATETIME,  @saelgerNr INT, @telefonNr NVARCHAR(8)
AS
    IF NOT EXISTS (SELECT Navn FROM Saelger WHERE Saelger.MedarbNr = @saelgerNr)
    BEGIN
      INSERT INTO
      Saelger(Navn,Email,StartDato,MedarbNr)VALUES(@navn,@email,@startDato,@saelgerNr);
    END
    INSERT INTO TelefonNr(Nr,Saelger) VALUES (@telefonNr,@saelgerNr);
GO

EXECUTE InsertSaelger 'Jens', 'jens@mail.dk', '2001-1-1',4,'67891234';
EXECUTE InsertSaelger 'Jens', 'jens@mail.dk', '2001-1-1',4,'67891235';
```

Med stored procedure kan man automatiserer nogle processer. Her er f.eks. vist hvordan vi indsætter Saelger og telefon nr i en og samme arbejdsgang.