UNIVERSITETET I OSLO

Integritetsregler i SQL

Integritetsregler i SQL

- Kandidat- og primærnøkler
- Referanseintegritet - fremmednøkler
- Domenebegrensende integritetsregler – skranker på attributter og tupler
- Interrelasjonsskranker – assertions
- Triggere

Primærnøkler

- Kan deklareres i **create table** sammen med primærnøkkelattributtet
  (bare hvis attributtet utgjør primærnøkkelen alene)

```
create table MovieStar(
  name char(30) primary key,
  ... );
```

- Kan deklareres separat i **create table** etter attributtdeklarasjonene

```
create table MovieStar(
  name char(30),
  ...
  primary key (name)
);
```


Regler for primærnøkler

- Maks én primærnøkkeldeklarasjon pr. relasjon
- Konsekvenser av deklarasjonen:
  - To tupler i relasjonen får ikke stemme overens i alle attributtene i primærnøkkelen (selv ikke med bagrepresentasjon!)
    Forsøk på brudd ved **insert** eller **update** skal avvises av DBMSet
  - Attributtene i primærnøkkelen får ikke inneholde **null**
- Primærnøkkelskranken må sjekkes ved hver **insert** og hver **update**

Kandidatnøkler

- Kan deklareres i **create table** sammen med nøkkelattributtet (bare hvis attributtet utgjør kandidatnøkkel alene)

```
create table MovieStar(
  address varchar(255) unique,
  ... );
```

- Kan deklareres separat i **create table** etter attributtdeklarasjonene

```
create table MovieStar(
  name varchar(40),
  address varchar(255),
  ...
  unique (name,address)
);
```


Kandidatnøkler, regler

- Flere kandidatnøkkeldeklarasjoner er tillatt pr. relasjon
- Konsekvenser av deklarasjonen:
  - To tupler i relasjonen får ikke stemme overens i alle attributtene i kandidatnøkkelen
  - Kan brytes hvis ett eller flere av attributtene i kandidatnøkkelen inneholder **null**
- Kandidatnøkkelskranken må sjekkes ved hver **insert** og hver **update**

Indekser på kandidatnøkler

- DBMSer bygger vanligvis indekser automatisk på primærnøklene.
- For hver kandidatnøkkel må man vurdere spesielt om det bør deklareres indeks på nøkkelen
  Syntaks avhenger av DBMSet
  Noen SQL-implementasjoner tillater deklarasjon av kandidatnøkkel + indeks i en og samme setning:

```
create unique index AddressIndex
on MovieStar(address);
```

- Hvis det er opprettet indeks på en nøkkel, benyttes denne under sjekk av flerforekomster
  Ellers: Må i verste fall søke gjennom hele relasjonen

Referanseintegritet = fremmednøkler

- Deklarasjon av fremmednøkler:

```
create table Studio(
  ..
  presC# int
    references MovieExec(cert#),
  ..);
```

- Alternativt:

```
create table Studio(
  ..
  presC# int,
  ..
  foreign key (presC#) references MovieExec(cert#)
  ..);
```

Fremmednøkler, regler

- Konsekvenser av deklarasjonen:
  - De refererte attributtene må være deklarert **unique** eller **primary key**
  - Verdier (≠ **null**) som opptrer i fremmednøkkelens refererende attributter *må* opptre i de refererte attributtene
- Referanseintegritet må sjekkes både ved **insert**, **update** og **delete**

Opprettholdelse av referanseintegritet

Tre strategier (aktuelle bare ved **update**, **delete**):

- **Reject** (= default strategi):
  Avvis modifikasjoner som bryter regelen
- **Cascade**: Endringer i referert verdi påtvinges også refererende verdi (Hvis du sletter et tuppel t, sletter du alle tupler som refererer t (rekursivt))
- **Set-null**: Sletting av referert verdi forårsaker endring av refererende verdi til **null**

Strategiene velges separat for hver **update**, **delete** på den enkelte fremmednøkkel

Midlertidige brudd på referanseintegriteten

Nødvendig ved sirkulære avhengigheter mellom relasjoner

1. Grupper flere SQL-setninger i en *transaksjon*
2. Kan instruere SQL-systemet om ikke å sjekke referanseintegritet før transaksjonen er endelig avsluttet (committed)
   - i. **not deferrable** (default): Skranken skal sjekkes umiddelbart når sjekk er påkrevd
   - ii. **deferrable**: Sjekk (kan) utsettes
     - a. **initially deferred**: Sjekk utsettes i utgangspunktet til slutten av inneværende transaksjon
     - b. **initially immediate**: Sjekk gjøres i utgangspunktet før modifikasjon av databasen

Deklarasjon av deferrable

- I **create table**:
  (i) Deklarer deferrable
  (ii) Navngi skranken (kan sløyfes):

```
presC# int
  unique
  constraint PresC#ForeignKey
    references MovieExec(cert#)
      deferrable initially deferred
```

- Kan senere endre fra **deferred** til **immediate** og omvendt:

```
set constraint PresC#ForeignKey immediate;
..
set constraint PresC#ForeignKey deferred;
```

Skranker på attributter og tupler

- Bidrar til å begrense lovlige (kombinasjoner av) verdier i attributtene innen en relasjon
- Tommelfingerregel:
  Bare *svært enkle* skranker bør formuleres på denne måten!

Skranke på ett attributt: **not null**

- I **create table**:
  presC# **int**
  references MovieExec(cert#) **not null**
- Konsekvenser:
  – Kan ikke sette inn tuppel med verdien **null** i attributtet
  – Kan ikke endre verdien til **null** senere
  – **set-null**-policyen kan ikke benyttes (dette innvirker på hvilke views som blir oppdaterbare)

Skranke på ett attributt: **check**

- I **create table**:

```
presC# int
  references MovieExec(cert#)
  check (presC# >= 100000 and
         presC# < 1000000);
```

- Angir en betingelse på attributtet
- Betingelsen kan henvise til attributtet selv, og til andre attributter – via subquery
- Sjekkes ved hver endring av attributtets verdi
- **NB** Sjekkes *ikke* hvis **check** refererer til andre attributter og disse endres, mens attributtet med **check**-skranke ikke endres!

Skranke på et tuppel: **check**

- I **create table**:

```
...
check ((gender = 'F' or name not like 'Ms.%')
    and (gender = 'M' or name not like 'Mr.%'))
...
```

- Angir en betingelse på et tuppel i relasjonen
- Betingelsen kan henvise til attributter i relasjonen, og til attributter i andre relasjoner – via subquery
- Sjekkes hver gang et tuppel settes inn eller oppdateres
- **NB** Sjekkes *ikke* hvis **check** refererer til attributter i andre relasjoner og disse endres, mens det ikke er endringer i inneværende relasjon!

Endring av skranker

- Skranker kan legges til, endres, slettes, når som helst
  Man må i så fall navngi skrankene sammen med deklarasjonen av dem
- Endring av tabellskranker:
  - **immediate** ↔ **deferred** ved **set constraint** ...
  - **alter table**: Legg til eller slett skranker
    - **alter table** MovieStar **drop constraint** NameIsKey;
    - **alter table** MovieStar **add constraint** MyKey **primary key** (name);
    - **alter table** MovieStar **add constraint** NoAndro **check** (gender **in** ('F', 'M'));

Assertions–I

- Assertions er en del av databaseskjemaet, på lik linje med relasjonsskjemaer og views:

```
create assertion SumLength
  check (10000 >= all
    (select sum(length)
     from Movie
     group by studioName));
```


Assertions–II

- Assertions formuleres som et boolsk uttrykk som alltid må evalueres til **true**
- Kan involvere flere relasjoner
- Sjekkes når en av de involverte relasjonene endres
- Vanskelig å implementere effektivt
- Kan fjernes igjen:
  drop assertion SumLength

Triggere

- Serie aksjoner assosiert med angitte hendelser
  Utføres når hendelsen inntreffer
- Vekkes når en *hendelse* skal til å inntreffe
  (typisk: **insert**, **update**, **delete** mhp. gitt relasjon)
- Deretter testes en *betingelse*
  Hvis betingelsen ikke holder, avbrytes triggingen
- Hvis betingelsen er oppfylt, utføres en *aksjon*
  Aksjonen kan hindre hendelsen i å inntreffe eller omgjøre den mm.
  Generelt er aksjonen en sekvens av database-operasjoner

Valgmuligheter for triggere i SQL

1. Aksjonen kan eksekveres før eller etter den triggende hendelsen
2. Aksjonen kan referere til gamle og/eller nye tuppelverdier som blir satt inn, slettet eller oppdatert i den triggende hendelsen
3. Oppdateringshendelser kan begrenses til et gitt attributt/en gruppe attributter
4. En betingelse kan angis; da trigges det bare hvis hendelsen inntreffer og betingelsen evaluerer til **true** Dette gjøres i en **when**-klausul
5. Programmereren kan spesifisere at aksjonen skal utføres
   - enten én gang for hvert modifisert tuppel
   - eller én gang for samtlige modifiserte tupler

Eksempel 1 på trigger

Skjema: MovieExec(name, address, cert#, netWorth)

```
create trigger NetWorthTrigger
after update of netWorth on MovieExec
referencing
  old row as OldRow,
  new row as NewRow
for each row
when (OldRow.netWorth > NewRow.netWorth)
  update MovieExec
  set netWorth = OldRow.netWorth
  where cert# = NewRow.cert#;
```


Eksempel 2 på trigger

```
create trigger AvgNetWorthTrigger
after update of netWorth on MovieExec
referencing
  old table as OldStuff,
  new table as NewStuff
for each statement
when (500000 > (select avg(netWorth) from MovieExec))
begin
  delete from MovieExec
  where (name, address, cert#, netWorth) in NewStuff;
  insert into MovieExec (select * from OldStuff);
end;
```


Øvingsoppgave 4

- Skjema:
  Prosjekt(P#,Pnavn,Kunde,Pleder,StartDato)
  Ansatt(A#,Navn,Tittel,Fdato,Pnr,AnsDato)
  Timeliste(A#,Dato,P#,Timer)
  Kunde(K#,Knavn,Adresse)

- Oppgave:
  Finn navn og tittel på de ansatte som har arbeidet på alle prosjekter som er påbegynt i dette århundret og er bestilt av kunden «Pust og pes AS».