# Transaktioner i SQL

En logisk operation såsom

- sælg 10 stk. af vare 17

- indebærer
  - læs L17 = ”lagerbeholdning for vare 17”
  - sæt L17 = L17 -10
  - skriv L17

Hvis flere sælgere har salg af vare 17 kan der opstå et problem:

| sælger 1 | sælger 2 |
|---|---|
| læs L17=11 | |
| | læs L17=11 |
| skriv L17 = L17-10=11-10 | |
| | skriv L17 = L17-10=11-10 |

Serializability

- at f.eks. salg som disse gennemføres serielt
- eller i det mindste gennemføres på en måde, der svarer til seriel udførelse

# Atomicity

**Seriel udførelse af logiske operationer er ikke nok**

**En logisk operation såsom**

- **overfør 10.000 kr. fra Hanne til Peter**

- **indebærer (HS: Hannes saldo, PS: Peters saldo)**
  - **sæt HS = HS – 10.000**
  - **sæt PS = PS + 10.000**

**Hvis nogen slukker for computeren undervejs så har vi potentielt en inkonsistent databasetilstand**

**Atomicity**

- **enten ingen del-operationer**
- **eller alle del-operationer**

# Transaktion

## Transaktion

- samling af operationer, der udføres *atomart* og
- evt. også på en måde der svarer til *seriel* udførelse (f.eks. låsning)

## SQL2-transaktion

- serilizability er default
- mindre restriktive begrænsninger kan vælges
- ”transaction begin”
  - er implicit
- ”transaction end” er
  - COMMIT for succes – alle ændringer bliver effektueret i databasen
  - ROLLBACK for fortryd – ingen ændringer bliver effektueret i databasen

## ”Dirty Read”

- læsning af data, der endnu ikke er committed
- kan være et alvorligt problem
  - f.eks. ved posteringer, der opdaterer relativt til ”dirty” værdier
- kan være et middel til optimering
  - f.eks. ved gentagne forsøg i forbindelse med reservation

# SET TRANSACTION

## SET TRANSACTION

- er en SQL2 explicit ”transaction begin” operation,

## Form

- SET TRANSACTION READ WRITE
  - er default
- SET TRANSACTION READ ONLY
  - mindre restriktivt – tillader nogen parallelitet, der kan udnyttes af systemet

## Generel form

- omfatter også isolation level
- SET TRANSACTION [READ ONLY]  [ISOLATION LEVEL ...]

# 4 SQL2 Isolation levels

**SET TRANSACTION ISOLATION LEVEL SERIALIZABLE**

- krav om isolation svarende til seriel udførelse
- er default

**SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED**

- ”dirty read” er tilladt

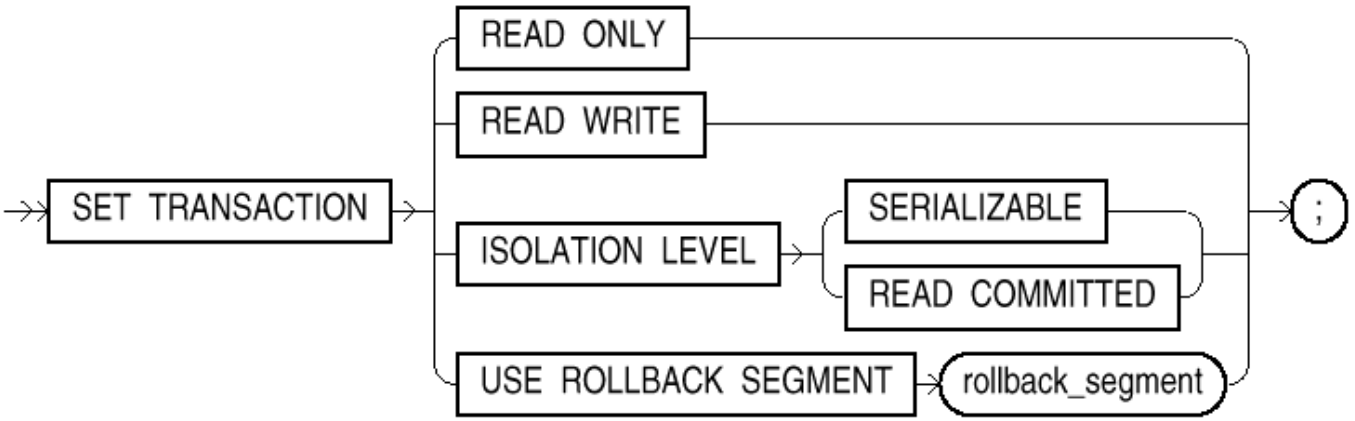
**SET TRANSACTION ISOLATION LEVEL READ COMMITTED**

- ”dirty read” er ikke tilladt
- men gentagen læsning må gerne give forskelligt resultat

**SET TRANSACTION ISOLATION LEVEL REPEATABLE READ**

- gentagen læsning skal give samme resultat,
- bortset fra phantom tupler (indsatte under transaktionen af andre transaktioner)

# Oracle Transaction



# Oracle Transaction

**SET TRANSACTION**

- kan undværes som i SQL2
- READ WRITE
  - default som i SQL2
- READ ONLY
  - After a transaction has been established as read-only, all subsequent queries in that transaction only see changes committed before the transaction began. Read-only transactions are very useful for reports
- SERIALIZABLE
  - som SQL2
  - If a serializable transaction contains data manipulation language (DML) that attempts to update any resource that may have been updated in a transaction uncommitted at the start of the serializable transaction, then the DML statement fails.
- READ COMMITTED
  - Default som i SQL2
  - If the transaction contains DML that requires row locks held by another transaction, then the DML statement waits until the row locks are released.
- USE ROLLBACK SEGMENT
  - blot til tilpasning alternative valg efter behov for størrelse
  - assigns the current transaction to the specified rollback segment. This option also implicitly establishes the transaction as a read-write transaction.

# ACID Transaction

**ACID: Atomicity- Consistency- Isolation- Durability**

- **Atomicity**
  - **alle eller ingen oprationer i T gennemføres**
  - **Opnås ved: resultat af T beregnes og commit'tes (T logges og resultat realiseres i DB)**
- **Consistency**
  - **defineret ved regler (constraints), kræves før og efter T, men ikke under T**
  - **Opnås ved: understøttelse af kontrol af konsistens og håndtering af inkonsistens**

    systemets opretholdelse af reglerne
- **Isolation**
  - **samtidige T1 og T2 må ikke have indflydelse på hinanden (F.eks. "at tømme sin konto i to banker samtidigt")**
  - **Opnås ved: låsning ved opdatering**
- **Durability**
  - **resultat af T må ikke gå tabt**
  - **Opnås ved: logging og genetablering af data fra log-filen:**

    Ved normal brug af databasen
    Opdaterende transaktioner logges i en log-fil.
    Ved database-backup
    log-filen tømmes
    Efter systemnedbrud
    databasen overskrives med kopi fra sidste database-backup
    transaktionerne fra den aktuelle log-fil udføres på databasen

# SQL2 Sikkerhed og Bruger-id

## Autorisations id

- en slags brugernavn
- PUBLIC er et special tilfælde
- kan gives privilegier
    - der er komplekse i et DBMS

## SQL2 privilegier

- SELECT, INSERT, DELETE, UPDATE
- REFERENCES
    - ret til at definere integritetsbegrænsning
- USAGE
    - på domæne – retten til at bruge det
- evt. på attribut-niveau:
    - INSERT, UPDATE,  REFERENCES

## Eksempler

- INSERT INTO R  values(...)
- INSERT INTO R(A)  values(...)
- INSERT INTO R SELECT * from B
- INSERT INTO R SELECT * from R WHERE ...

- kræver privilegier?

# GRANT, REVOKE

**Skemaer har ejer**

- ejer har alle privilegier
- Ejerskab opstår primært ved: den der skaber, ejer

**SQL operation har**

- database elementer den virker på
- bruger (agent, hvis autorisations id) der skal checkes for privilegier

**GRANT**

- at give privilegier videre
- Form
  - GRANT <privilegier> ON <db-element> TO <brugere> [WITH GRANT OPTION]

**REVOKE**

- at tage privilegier tilbage
- Form
  - REVOKE <privilegier> ON <db-element> TO <brugere> [CASCADE, RESTRICT]

**REVOKE GRANT OPTION**

- at tage GRANT privilegier tilbage
- Form
  - REVOKE GRANT OPTION <priv> ON <db-el> TO <brg> [CASCADE, RESTRICT]

# Eksempel

**Figur 7.15**

**REVOKE SELECT, INSERT ON studio FROM picard CASCADE;**

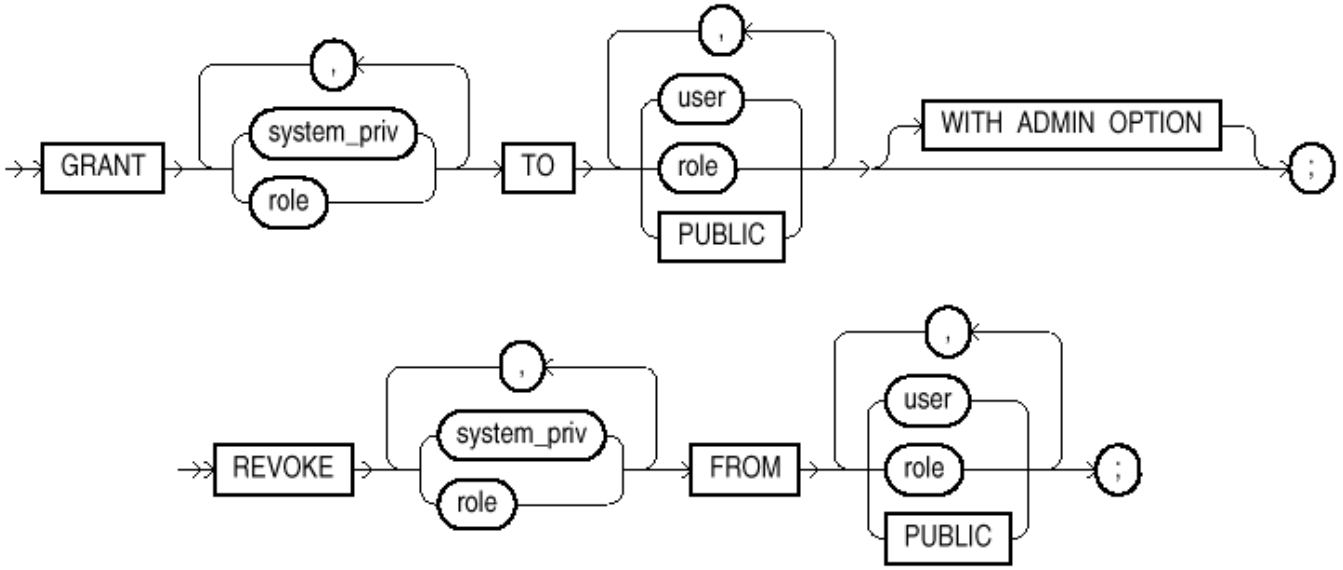
**REVOKE SELECT ON movie  FROM picard CASCADE;**

**Figur 7.16**

**Eksempel**

- **Fig 7.17**

**Eksempel**

- **Fig 7.18**

# Oracle Sikkerhed og Bruger-id