## Normalisering

Metode for å designe relasjonsdatabaser via funksjonelle avhengigheter for å unngå unødvendig dobbeltlagring av data (dobbeltlagring er nødvendig for å koble tabeller sammen via primær og fremmed nøkkel). Jo strengere krav om normal form desto høyere normalform. 1NF 2NF 3NF BCNF.

**Supernøkkel:** Er en mengde av attributter der verdiene i disse attributtene er unike i en relasjon.

**Kandidatnøkkel:** En minimalmengde attributter der verdiene er unike. Delmengde av supernøkkel

**Primærnøkkel:** En spesielt utpekt kandidatnøkkel der ingen verdier kan være NULL. Delmengde av kandidatnøkkel

**Nøkkelattributt:** Attributter som er med i en kandidatnøkkel.

**Referanseintegritet:** Hvis fremmednøkkelen ikke er null, så skal de finnes et tuppel i den refererte relasjonen hvor primærnøkkelen har samme verdi som fremmednøkkelen.

**INSERT anomali:** f.eks. kan ikke starte en tom bestilling uten antall produkter, kan ikke legge inn adresse uten en bestilling.

**UPDATE anomali:** oppdatering må gjøres flere steder.

**DELETE anomali:** sletting av den ene kan føre til sletting av den andre. F.eks sletting av bestilling sletter kundeinformasjon.

## 1NF

Alle attributter inneholder bare atomære verdier, og ingen null verdier i primær nøkkel

## Table Description automatically generatedText, chat or text message Description automatically generated2NF

Table

Description automatically generatedText

Description automatically generated

## 3NF BCNF

A screenshot of a computer

Description automatically generated with medium confidence

A->B, der A må være en supernøkkel.

BCNF er innfridd hvis 3NF og hvis tabellen ikke har minst 2 sammensatte minimale kandidatnøkler, dvs. BCNF er bare et potensielt problem hvis du har en sammensatt primærnøkkel

**Transitive avhengigheter**: avhengighet mellom flere ikke nøkkel attributter.

# Transaksjoner

**AUTOCOMMIT:** oppdateringer skjer umiddelbart etter hvert sql setning (ikke mulig å angre ved rollback)

**Leselås:** kun lov å lese, ikke skrive. Også kalt delt lås. (flere trans. Kan ha delt lås på samme objekt)

**Ekslusivtlås**: andre trans. har ingen tilgang. (en av gangen). Forutsetter ingen andre låser på objektet.

**Tofaselåsing**: korrekt utførelse.Utvidelses (sette lås) og avviklingsfasen (Commit/Rollback)

**Vranglås:** trans. som venter stående på hverandre. Løsn timeout tilbakerull, premonition, lås på begynnelsen av transaksjonene(ueffektivt)

**Serializable fullt isolasjonsnivå:** select deltlås, update ekslusivt lås.

**READUNCOMMITTED:** kan lese uncommitted verdi. **Ingen lås**

**READCOMMITTED:** kan lese bare committed verdier

**REPEATABLEREAD:** ALeser samme verdier med hver lesing, B kan endre andre rader (**rad nivå lås**)

**SERIALIZABLE:** En annen transaksjon kan ikke lese eller skrive når A holder på med å lese (Tabell nivå lås)

## Samtidighetsproblemer

Example: user A runs the same query (spørring) twice. B runs a transaction and commits in between

**DIRTYREAD:** Transaksjonen kan se på uncommitted endringer. Siste spørring for A fører til ulik verdi.

**NONREPEATABLEREAD:** ser committed, menverdien kan endre etter lesing (som kan påvirke slutt resultat).

**Text

Description automatically generatedTable

Description automatically generatedPHANTOMREAD:** A får samme verdi på begge spørringene. Hvis B har legget inn eller slettet rader, får vi ikke de samme radene (Phantom read).

Text

Description automatically generated

# Algebraopersasjoner

A screenshot of a computer

Description automatically generated with low confidence**Seleksjon**: velg ut rad

**Projeksjon:** velg ut kolonne

**Union:** slå sammen to tabeller (forutsetter like mange attributter og samme datatype)

**Snitt:** innhold som finnes i begge tabeller (like rader). (like mange attributter og samme datatype)

**Differanse**: Rader i A som ikke finnes i B (like mange attributter og samme datatype ).

**Naturlig forening:**  Forener to tabeller mhp samme attributt navn og datatyper. Resultatet har bare en kopi av felles kolonner fra to tabeller, de andre ulike kolonner blir også tatt med.

**Kryssprodukt:** alle kombinasjoner av A og B

**ACID:** Atomic (hele eller ingenting), Consistency (fra stabil tilstand til en annen stabil tilstand), Isolation (Data berørt av en transaksjon må isoleres slik at andre transaksjoner kan arbeide med de samme dataene), Durability (Etter transaksjon er ferdig må endringene lagres og bli synlig for andre brukere)

## Gjennopretting av en database («recovery»)

# Timeline Description automatically generated

# SQL Join types explained with 1 picture – Chris DaleSql spørringer

1. Lag en liste over alle bedriftene. Navn, telefon og epost til bedriften skal skrives ut.

*SELECT navn, tlf, epost FROM bedrift*;

2) Lag en liste over alle oppdragene. Om hvert oppdrag skal du skrive ut oppdragets nummer samt navn og telefonnummer til bedriften som tilbyr oppdraget.

*SELECT o.oppdrag\_id, b.navn, b.tlf FROM oppdrag o, bedrift b*

*WHERE o.bedrift\_id = b.org\_nr*;

4) Som oppgave 3d), men få med de kandidatene som ikke er registrert med kvalifikasjoner.

*SELECT kk.kvalifikasjon\_id, ka.kandidat\_id, ka.fornavn, ka.etternavn, kv.info*

*FROM kandidat ka LEFT JOIN kval\_kandidat kk ON (ka.kandidat\_id = kk.kandidat\_id)*

*LEFT JOIN kvalifikasjon kv ON (kv.kvalifikasjon\_id = kk.kvalifikasjon\_id);*

## SQL SPØRRINGER MED COUNT og stoler

1) Finn **hvor mange (antallet)** stolmodeller som finnes av hver stoltype

*SELECT t.navn AS "Type navn", COUNT(\*) AS "ant\_modeller" FROM type t LEFT JOIN model m*

*ON (m.type\_navn = t.navn)*

*GROUP BY t.navn;*

2) Ut fra alle registrerte ordre (bestillinger): Finn **gjennomsnittlig** antall bestilte stoler av hver stoltype.

*SELECT t.navn AS "type navn", AVG(t.antall) AS "gjen. bestilte stoler" FROM*

*((type t LEFT JOIN modell m ON (t.navn = m.type\_navn) LEFT JOIN stol s ON (s.model = m.navn))* JOIN stolordre so ON(so.stol\_id = s.stol\_id)

3) Finn det **totale antallet** stoler som finnes i bestilling, og som enda **ikke** er levert kunder.

Tips: Sjekk på reell leveringsdato (dvs. om ordren er effektuert).

*SELECT SUM(antall) AS "bestilte ikke-leverte stoler" FROM stolordre so JOIN ordre o ON (o.reell\_leveringsdato IS NULL AND o.ordre\_id = so.ordre\_id);*

4) Finn ut **hvor mange (antallet**) av stolene i spørring 3 over som er standardstoler.

*SELECT SUM(antall) AS "tot standard stoler" FROM (stolordre so JOIN ordre o ON (o.ordre\_id = so.ordre\_id AND o.reell\_leveringsdato is NULL) JOIN stol s ON (s.stol\_id = so.stol\_id))*

*WHERE s.stol\_id IN (SELECT stolnr FROM standardsstol);*