UNIVERSITETET I AGDER

Emnekode : IS-201
Kandidatnr. : 3134
Dato : 26.11-10
Ark nr. : 1 av 8

Oppgave 1

a) Sammenhengen mellom Database Manipulation Language og commit og rollback funksjoner er at diverse kommandoer ikke blir utført på databasen før en commit er gjort. Rollback funksjonen brukes for å gå tilbake til en tidligere versjon av databasen. Det er ikke alle databaser som støtter rollback, avhengig av database Enginen, f.eks InnoDB har rollback funksjon. Commit til databasen gjøres ikke før alle DML utsagn er sjekket at fungerer.

Insert → Delete → Alter → Commit → Delete → Commit → Rollback

b) en til mange relasjon er implementert i en relasjonell datamodell ved å definere en Forreign key i mange siden av relasjonen som peker til primær nøkkelen i en-siden av relasjonen

ORDRE(ordreNr, status, levDato, kundeNr)
KUNDE(kundeID, navn, adresse)

En enkel relasjons modell der ordre er på mange siden og har en fremmednøkkel kundeNr.

UNIVERSITETET I AGDER

Emnekode : IS-201

Kandidatnr. : 3134

Dato : 26.11 -10

Ark nr. : 2 av 8

c) En index er ~~[illegible]~~ som gule sider i databaser. Indexer brukes til å gjøre ~~søk i databasen~~ databasen raskere å finne fram i, gjør f.eks søk raskere. Man burde indexere hele tabellen, inkludert primær nøkkelen. Man må ikke indexere ~~Fremmednøkler~~ PK. Indexer lagrer informasjon om hvor de forskjellige data finnes, det finnes en del ulike ~~metoder~~ måter å indexere på alt etter størrelsen og hva som er lagra i databasen. Databasen burde ha over 1000 rader/tuples for at indexering skal være verdt det.

UNIVERSITETET I AGDER

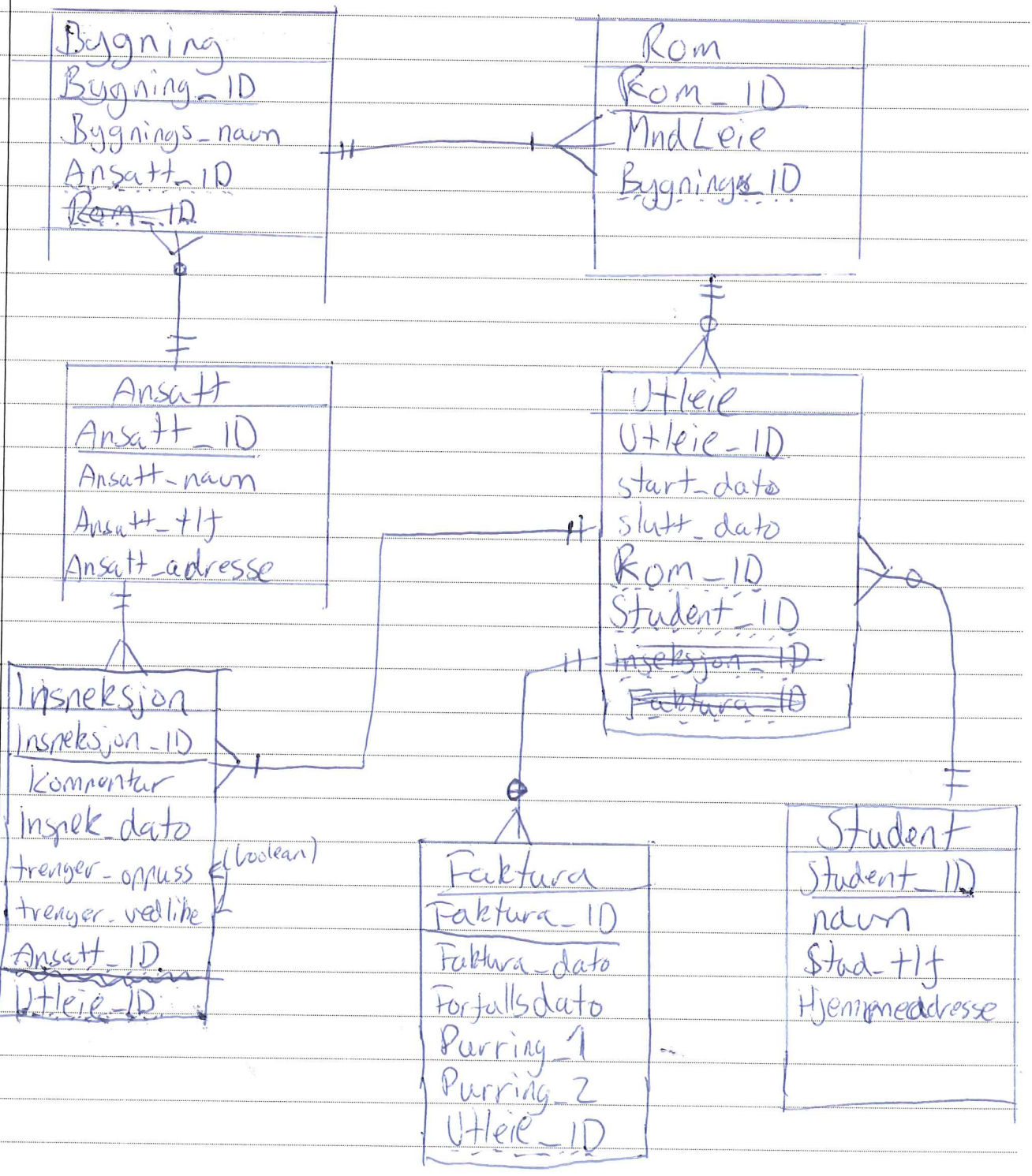
Emnekode : IS-209

Kandidatnr. : 3134

Dato : 26.11-2010

Ark nr. : 3 av 8

Oppgave 2



Ser ikke behov for sjekkeid her, ville bare vært knytta til Studenten 1:m.

Oppgave 3

Denne raporten er på UNF. Den har flerverdi atributter.

Funksjonelle avhengigheter:
Doctor-in-charge → Phone
Wardno → Doctor in charge
Patientno → Lastname, First Name, Home_city
Nurse → Phone number, Address

1NF Clinic(Wardno, Doctor, doctor_phone, nurse_name nurse_phone, nurse_address, patientno, patient_lastname, patient_firstname, patient_city, visit_date, diagnosis, referred_to)

Dele skille alle funksjonelle avhengigheter

2NF

Ward(Ward_ID, doctor_ID)
Doctor(Doctor_ID, doctor_name, doctor_phone)
Nurse(Nurse_ID, nurse_name, nurse_phone, nurse_address, Ward_ID)
Patient(Patientno, patient_lastname, patient_firstname, patient_city, visit_date, diagnosis, refered_to, Nurse_ID)

Oppgave 3 forts

3NF Det som trengs a endre er de transitive avhengigheter. Ville delt patient i

Visit/check(visit_ID, visit_date, diagnosis, refered_to, ~~patient~~ patientno)

Patient(Patientno, patient_lastname, patient_firstname, patient_city Nurse_ID)

Etter det jeg kan se i tabellen virker det som en pasient kun få forskjellige diagnoser, dvs at diagnosis og refered to peker på visit_date.

Refered to avhenger ikke av diagnosis fordi jeg ser at man kan få refered to forskjellige uavhengig av diagnose.

Regner med at Phone er tilknytta doctor, og ikke til Ward.

Regner også med at pasienter kan få flere besøk samme dag.

Oppgave 4

a)

Constraints i en database er regler mas setter for atributter, som at de skal vaare Primary key / Foregn key.

Constraints til ordredetalj:
CONSTRAIN Foregn key ~~fk_01~~ (ordrenr) ← ordredetalj_fk01
REFERENCE Ordre(ordrenr),
CONSTRAIN Foregn key fk02 (produktnr)
REFERENCE Produkt(produktnr) ← ordredetalj_fk02

For primary key i ordre:
CONSTRAIN Primary key ordre_pk(ordrenr)
Samme i Produkt bare bytte ut ordre med produkt produktnr

b) SELECT produktnavn navn, ~~produktpris~~ enhetspris pris,
antall_på_lager, bestillingsnivå
FROM produkt p
INNER JOIN ordredetalj od
ON od.produktnr = p.produktnr
INNER JOIN ordre o
ON od.ordrenr = o.ordrenr
WHERE ordre_dato BETWEEN '01.10.2010'
AND '31.10.2010'
ORDER BY navn asc;

Mulig man må bruke TO DATE her for å få riktig på Between
BETWEEN To Date ('01.10.2010', dd.mm.yyyy')
samme greia etter AND

C) SELECT kunde_navn, kunde.kundenr

FROM kunde WHERE kundenr NOT IN
(SELECT DISTINCT kundenr
FROM ordre)

antall_produkter

D) SELECT kundenr, kundenavn, sum(antall_bestilt),
sum(enhetspris*antall_bestilt)
FROM kunde k, ordre o, ordredetalj od, produkt p
WHERE k.kundenr = o.kundenr
AND o.ordrenr = od.ordrenr
AND od.produktnr = p.produktnr
GROUP BY kundenr
ORDER BY kundenavn asc;

Kunne brukt Inner Joins her også, spesielt i forhold til at det blir veldig mange AND når E) legges til. Ville da brukt inner joins som i C) + kunde

E) Legge til i WHERE
kunde.postnr BETWEEN 4600 AND 4699
AND antall_produkter > 5

Hvis det menes at man bare skal ha med produkter det er flere enn i bestilt av totalt må det gjøres via to Select spørringer som i oppg 4c. Virker ulogisk.

F) Et database view er et utsnitt av noe i databasen. Vi bruker views for å gjøre det enkelt for f.eks brukere av database sen slik at de bare ser det de trenger å se, samt bare gjøre endringer på det de har lov til å gjøre endringer på. Man kan gi privilegier til views. Man kan også gjøre spørringer på views, noe som gjør at avanserte spørringer ikke trenger å gjøres så mange ganger fordi de allerede er definert i et view.

Eksempel:

```
CREATE VIEW Kunde_krsand AS
( SELECT kundenr, kunde_navn, kunde_gate,
  postnr, kunde_tlfnr, kunde_epost, kunde_by
  FROM Kunde
  WHERE * kunde_by = 'kristiansand'
  ORDER BY kunde_navn, kunde_gate);
```