



Avdeling for allmenne fag

EKSAMEN

865 DATABASAR II

28.05.2003

Tid: 3 timar / 4 timar

Målform: Bokmål/nynorsk

Sidetal: 7 med framsida

Hjelpemiddel: Ingen

Merknader: Ingen

Vedlegg: *Forenklet syntaks, SQL og Forenklet syntaks, PL/SQL*

Eksamensresultata blir offentliggjort på lister med kandidatnummer, lagt i hyllene ved eksamensoppslagstavla, eller du kan gå inn på Arena høgskole – da trenger du heller ikkje kandidatnummeret.

Oppgavesettet består av tre deler. Noen deloppgaver skal kun besvares av de som har fritak for midtprøven, og dermed får utvidet sluttprøve (4 timer). Disse deloppgavene er merket: "Skal kun besvares av de som har 4 timers sluttprøve".

Vekting av deloppgaver i forhold til endelig karakter i kurset går fram av følgende tabell:

Deloppgave	Vekting 3 timer	Vekting 4 timer
1-a	10 %	10 %
1-b	15 %	15 %
1-c	15 %	15 %
2-a	10 %	10 %
2-b	10 %	10 %
2-c	10 %	10 %
2-d	10 %	10 %
3-a	Skal ikke gjøres	10 %
3-b	Skal ikke gjøres	10 %
Midtprøve	20 %	Fritatt

Les oppgaveteksten nøye, og prøv å svare på så mange deloppgaver som mulig. Lykke til!

Oppgave 1

I denne oppgaven skal du skrive PL/SQL-kode mot en database som inneholder informasjon om ansatte i en bedrift, hvilken kompetanse hver ansatt har og hvilke aktiviteter ansatte er tilordnet på forskjellige dager. Bedriften utfører oppdrag knyttet til vedlikehold av veier og jernbanelinjer. Databaseen har følgende tabellstruktur, der primærnøkler er understreket og fremmednøkler merket med en stjerne:

- ansatt (ansattnr, fornavn, etternavn, avdeling
- kompetanse (ansattnr*, kompkode)
- bemanning (ansattnr*, dato, aktivitet)

Tabellen *kompetanse* representerer et mange-til-mange forhold mellom ansatte og kompetansekoder. En kompetansekode kan for eksempel være "KL 2" og representere at en ansatt har førerkort klasse 2. I tabellen *bemanning* inneholder attributtet *aktivitet* en kode som identifiserer aktiviteten. En aktivitet kan for eksempel være vedlikeholdsarbeid på en bestemt veistrekning.

1-a)

Skriv en lagret prosedyre som registrerer en ny forekomst (rad) i tabellen *kompetanse*. Verdier for *ansattnr* og *kompkode* skal være parametere til prosedyren. Prosedyren skal sjekke brudd på entitetsintegritet (primærnøkler) og referanseintegritet (fremmednøkler), og i så fall skrive ut en feilmelding og avbryte innsetningen.

1-b)

Skriv en trigger som sørger for at ansatte ikke kan tilordnes aktiviteter mer enn 3 måneder fram i tid. *Tips:* Funksjonsskallet *months_between(d1, d2)* returnerer antall måneder mellom datoene *d1* og *d2*. SYSDATE gir dagens dato. Forklar kort effekten av triggeren.

1-c)

Skriv en lagret prosedyre som skriver ut en bemanningsplan for mai 2003. For hver ansatt skal prosedyren skrive ut navn og avdeling, og deretter listen med datoer og aktiviteter denne ansatte er tilordnet i perioden. Rapporten skal være sortert på etternavn.

Oppgave 2

Denne oppgaven tar for seg stoff beskrevet i kompendiet.

2-a)

Tema: B-trær.

Hva brukes B-trær til?

Anta tabellen *ansatt* fra oppgave 1 er fysisk sortert på *ansattnr*. Vis med en figur hvordan et B-tre på *etternavn* er organisert.

Tabellen inneholder 1000 rader, etternavn krever 30 byte, pekere i B-treet krever 8 byte og blokkstørrelsen er 4096 byte. Hvor mange diskaksesser kreves i et søk i B-treet for å hente ut en ansatt med et bestemt etternavn. Begrunn svaret.

2-b)

Tema: spørreoptimalisering. Du er gitt følgende spørring mot databasen fra oppgave 1.

```
SELECT ansatt.ansattnr, ansatt.fornavn, ansatt.avdeling
FROM ansatt, kompetanse
WHERE ansatt.etternavn = 'Hansen'
AND kompetanse.kompkode = 'KL 2'
AND ansatt.ansattnr = kompetanse.ansattnr
```

Vis hvordan spørringen blir representert i relasjonsalgebra når vi først utfører likekoblingen, deretter likhetene på *etternavn* og *kompkode*, og til slutt projeksjonen på attributtene *ansattnr*, *fornavn* og *avdeling*.

Vis hvordan uttrykket kan skrives om ved hjelp av heuristiske regler til en mer effektiv form. Begrunn hvorfor det transformerte uttrykket utgjør en mer effektiv eksekveringsplan.

2-c)

Tema: Distribuerte databasesystemer.

Hva menes med fragmentering og replikering? Hvilke to hovedtyper av fragmentering finnes?

Tabellen *ansatt* er horisontalt fragmentert, med 30% av datamengden lagret i Oslo, 20% lagret i Stockholm og 50% lagret i København. Tabellen *kompetanse* er replikert på samtlige steder. Spørringen fra oppgave 2-b blir utført i Oslo. Gi to eksempler på utførelse av spørringen i henhold til strategien "Send alt til ett sted" (Ship to one site), og begrunn hvilken framgangsmåte som er den beste.

Forklar kort hvordan protokollen 2-fase commit fungerer.

2-d)

Svar kort på følgende spørsmål:

Nevn de viktigste forskjellene og likhetene mellom Data Type Definition (DTD) og XML Schema.

Objektrelasjonelle databasesystemer gjør det mulig å definere tabeller som kan sies å bryte med første normalform. Gi et eksempel på dette.

Datavarehus blir ofte utformet i henhold til et stjerneskjema. Forklar med et eksempel hva et stjerneskjema er.

Oppgave 3

Denne oppgaven tar for seg stoff beskrevet i læreboka.

3-a) Skal kun besvares av de som har 4 timers sluttprøve

Gjør rede for oppbyggingen av en Oracle-server. Bruk gjerne figurer, og legg vekt på å få med følgende begreper: System Global Area, bakgrunnsprosesser, kontrollfiler, loggfiler, arkivloggfiler, parameterfil, passordfil, datafiler, tabellrom, segment, temporærsegment, tilbakestillingssegment og blokk.

3-b) Skal kun besvares av de som har 4 timers sluttprøve

Gjør rede for begrepene privilegium, rolle og profil. Få med forskjellen på systemprivilegier og objektprivilegier. Skriv SQL-kode for å tildele brukeren *ola* leserettigheter på tabellen *bemannig* fra oppgave 1. Hvilke brukere kan utføre denne kommandoen?

Slutt på oppgaveteksten

Oppgåva inneheld tre delar. Nokre deloppgåver skal berre løysast av dei som har fritak for midtprøve, og av den grunn får ei utvida sluttprøve (4 timar). Desse deloppgåvene er merkte: "Skal berre løysast av dei som har 4 timars sluttprøve".

Vekting av deloppgåver i høve til endeleg karakter i kurset går fram av følgjande tabell:

Deloppgåve	Vekting 3 timar	Vekting 4 timar
1-a	10 %	10 %
1-b	15 %	15 %
1-c	15 %	15 %
2-a	10 %	10 %
2-b	10 %	10 %
2-c	10 %	10 %
2-d	10 %	10 %
3-a	Skal ikkje gjerast	10 %
3-b	Skal ikkje gjerast	10 %
Midtprøve	20 %	Fritatt

Les oppgåva nøye, og freist å svare på så mange deloppgåver som mulig. Lukke til!

Oppgåve 1

I denne oppgåva skal du skrive PL/SQL-kode mot ein database som inneheld informasjon om tilsette i ei bedrift, kva for kompetanse kvar tilsett har og kva for aktivitetar tilsette er tilordna på forskjellige dagar. Bedrifta tek på seg oppdrag knytt til vedlikehald av vegar og jernbaneliner. Databasen har følgjande tabellstruktur, der primærnyklar er understreka og framandnyklar merkte med ei stjerne:

- tilsett (tilsettnr, fornamn, etternamn, avdeling
- kompetanse (tilsettnr*, kompkode)
- bemanning (tilsettnr*, dato, aktivitet)

Tabellen *kompetanse* representerer eit mange-til-mange forhold mellom tilsette og kompetansekodar. Ei kompetansekode kan til dømes vere "KL 2" og representere at ein tilsett har førarkort klasse 2. I tabellen *bemanning* inneheld attributtet *aktivitet* ei kode som identifiserer aktiviteten. Ein aktivitet kan til dømes vere vedlikehaldsarbeid på ein bestemt vegstrekning.

1-a)

Skriv ein lagra prosedyre som registrerer ein ny førekomst (rad) i tabellen *kompetanse*. Verdier for *tilsettnr* og *kompkode* skal vere parameterar til prosedyren. Prosedyren skal sjekke brot på entitetsintegritet (primærnyklar) og referanseintegritet (framandnyklar), og i så fall skrive ut ei feilmelding og avbryte innsettinga.

1-b)

Skriv ein trigger som syter for at tilsette ikkje kan bli tilordna aktivitetar meir enn 3 månader fram i tid. *Tips:* Funksjonskallet *months_between(d1, d2)* returnerer talet på månader mellom datoane *d1* og *d2*. SYSDATE gir dagens dato. Forklar kort effekten av triggeren.

1-c)

Skriv ein lagra prosedyre som skriv ut ein bemanningsplan for mai 2003. For kvar tilsett skal prosedyren skrive ut namn og avdeling, og deretter lista med datoar og aktivitetar denne tilsette er tilordna i perioden. Rapporten skal vere sortert på etternamn.

Oppgave 2

Denne oppgave tek for seg stoff omtala i kompendiet.

2-a)

Tema: B-tre.

Kva nyttast B-tre til?

Tabellen *tilsett* frå oppgave 1 er fysisk sortert på *tilsettnr*. Syn med ein figur korleis eit B-tre på *etternamn* er organisert.

Tabellen *tilsett* inneheld 1000 rader, *etternamn* krev 30 byte, peikarane i B-treet krev 8 byte og blokkstorleiken er 4096 byte. Kor mange diskaksessar er naudsynt i eit søk i B-treet for å hente ut ein *tilsett* med eit bestemt *etternamn*. Grunnge svaret.

2-b)

Tema: spørjeoptimalisering. Du er gitt følgjande spørjing mot databasen frå oppgave 1

```
SELECT tilsett.tilsettnr, tilsett.fornamn, tilsett.avdeling
FROM tilsett, kompetanse
WHERE tilsett.etternamn = 'Hansen'
AND kompetanse.kompkode = 'KL 2'
AND tilsett.tilsettnr = kompetanse.tilsettnr
```

Syn korleis spørjinga blir representert i relasjonsalgebra når vi først gjer likekoblinga, deretter likningane på *etternamn* og *kompkode*, og til slutt projeksjonen på attributta *tilsettnr*, *fornamn* og *avdeling*.

Syn korleis uttrykket kan skrivast om ved hjelp av heuristiske reglar til ein meir effektiv form. Grunnge kvifor det transformerte uttrykket utgjer ein meir effektiv eksekveringsplan.

2-c)

Tema: Distribuerte databasesystem.

- Kva meiner ein med fragmentering og replikering? Kva for to hovudtypar av fragmentering finst det?
- Tabellen *tilsett* er horisontalt fragmentert, med 30% av datamengda lagra i Oslo, 20% lagra i Stockholm og 50% lagra i København. Tabellen *kompetanse* er replikert på alle stadane. Spørjinga frå oppgave 2-b blir utført i Oslo. Gje to døme på utføring av spørjinga etter strategien "Send alt til ein stad" (Ship to one site), og grunnge kva for framgangsmåte som er den beste. Forklar kort korleis protokollen 2-fase commit fungerer.

2-d)

Svar kort på følgjande spørsmål:

Nemn dei viktigaste ulikskapane og likskapane mellom Data Type Definition (DTD) og XML Schema. Objektrelasjonelle databasesystem gjer det mogleg å definere tabellar som kan seiast å bryte med første normalform. Gje eit døme på dette.

Datavarehus blir ofte utforma etter eit stjerneskjema. Forklar med eit døme kva eit stjerneskjema er.

Oppgave 3

Denne oppgåva tek for seg stoff omtala i læreboka.

3-a) Skal berre løysast av dei som har 4 timars sluttprøve

Gjer greie for oppbygginga av ein Oracle-server. Nytt gjerne figurer, og legg vekt på å få med følgjande omgrep: System Global Area, bakgrunnsprosessar, kontrollfiler, loggfiler, arkivloggfiler, parameterfil, passordfil, datafiler, tabellrom, segment, temporærsegment, tilbakestillingssegment og blokk.

3-b) Skal berre løysast av dei som har 4 timars sluttprøve

Gjer greie for omgrepa privilegium, rolle og profil. Få med ulikskapen mellom systemprivilegium og objektprivilegium. Skriv SQL-kode for å tildele brukaren *ola* leserettar på tabellen *bemannig* frå oppgåve Kva for brukarar kan utføre denne kommandoen?

Slutt på oppgåva

Appendiks F – Forenklet syntaks, SQL

Appendiks F – Foreklet syntaks, SQL

```
ALTER TABLE [skjema.]tabellnavn {
  ADD ((kolonnenavn datatype [DEFAULT uttrykk] [kolonnefø-
    ring]|tabellføring)[,...])
  |MODIFY ((kolonnenavn [datatype] [DEFAULT uttrykk] [kolonnefø-
    ring])[,...])
  |DROP {PRIMARY[CASCADE]|UNIQUE(kolonnenavn [...])
    [CASCADE]|CONSTRAINT føring}
  |DROP COLUMN kolonnenavn
  |DISABLE {CONSTRAINT føring/PRIMARY KEY/UNIQUE(kolonne
    [,kolonne],...)}| [CASCADE]}
  |ENABLE{VALIDATE|NOVALIDATE}{CONSTRAINT føring/PRIMARY
    KEY/UNIQUE(kolonne [,kolonne],...)}

ALTER USER brukernavn
  IDENTIFIED {BY passord  EXTERNALLY}

COMMIT

CREATE [UNIQUE] INDEX [skjema. indeksnavn] ON [skjema. tabell-
  navn (kolonnenavn [ASC|DESC [...])

CREATE TABLE [skjema.]tabellnavn ({(kolonnenavn datatype
  [DEFAULT uttrykk] [kolonneføring]
  |tabellføring}[,...])}|AS subspørring};

CREATE USER brukernavn
  IDENTIFIED {BY passord  EXTERNALLY}

CREATE [OR REPLACE] [FORCE|NO_FORCE] VIEW [skjema.]viewnavn
  {(kolonnealias ...)}
  AS subspørring [WITH CHECK OPTION] [CONSTRAINT føringnavn] WITH
  READ ONLY]

DELETE [FROM] {[skjema.]tabellnavn|[skjema.]view-
  navn|(subspørring)}[alias] [WHERE betingelse]

DROP {INDEX [skjema. indeksnavn]|SEQUENCE [skjema.]sekvens-
  navn|TABLE [skjema. tabellnavn] [CASCADE CONSTRAINTS]|VIEW
  [skjema.]viewnavn}

GRANT {objektprivilegium ( kolonnenavn[,...])|ALL [PRIVI-
  LEGES] {(kolonnenavn[,...])}|ALL [PRIVI-
```

```
ON [skjema.] {tabellnavn|viewnavn} TO
  {bruker|rolle|PUBLIC} [,...] [WITH GRANT OPTION]
```

```
INSERT INTO {[skjema.]tabellnavn|[skjema.]view-
  navn|(subspørring1)} [(kolonnenavn [...])
  |VALUES (uttrykk [...])|(subspørring2)]
```

```
REVOKE {objektprivilegium|ALL [PRIVILEGES]} [...] ON
  [skjema.] {tabellnavn|viewnavn}
  FROM {bruker |rolle|PUBLIC} [...] [CASCADE CONSTRAINTS]
```

ROLLBACK

```
SELECT [ALL|DISTINCT] {*|{[skjema.] {tabellnavn|view-
  navn}.*/uttrykk [(AS) kolonnealias]}[,...]}
  FROM {[skjema.] {tabellnavn|view-
  navn} [alias] |(subspørring) [alias]}[,...]
  [WHERE betingelse]
  [GROUP BY uttrykk [...]]
  [HAVING betingelse]
  [(UNION|UNION ALL|MINUS|INTERSECT) selectsetning]
  [ORDER BY {kolonnenavn|kolonneposisjon|kolonnealias} [ASC|DESC]
    [...]]
```

```
UPDATE {[skjema.] {tabellnavn|viewnavn} |(subspørring1)} [alias]
  SET {kolonnenavn={ (subspørring2) |uttrykk} |(kolonne-
  navn[,...])=(subspørring3)}[,...]
  [WHERE betingelse]
```

```
betingelse ::= uttrykk {=|>|<|<>|<=>|>=}
  {uttrykk|(subspørring)} | uttrykksliste {=|<>} (subspørring)
  | uttrykk {=|>|<|<>|<=>|>=} {ANY|ALL|SOME} {uttrykks-
  liste|(subspørring)}
  | uttrykksliste {=|<>} {ANY|ALL|SOME} {(subspørring|uttrykksliste
    [...])}
  | uttrykk [NOT] IN {uttrykksliste|(subspørring)}
  | uttrykksliste [NOT] IN {(subspørring|uttrykksliste [...])}
  | uttrykk [NOT] BETWEEN uttrykk AND uttrykk | uttrykk IS [NOT]
  NULL|EXISTS (subspørring)
  | streng1 [ NOT ] LIKE streng2 | (betingelse) | NOT betingelse |
  betingelse {AND|OR} betingelse
  | tabell1.kolonne = tabell2.kolonne (+) | tabell1.kolonne (+) =
  tabell2.kolonne
```

datatype: = CHAR|CHAR(n)|VARCHAR2(n)|NUMBER(p,s)|DATE|LONG

```
funksjoner ::=
  ADD_MONTHS(date,months_to_add) | AVG([DISTINCT|ALL] uttrykk) |
  COUNT({*|[DISTINCT|ALL] uttrykk}) | INITCAP
  (streng) | LENGTH(streng) | LOWER(streng) | MAX([DISTINCT|ALL]
  uttrykk) | MIN([DISTINCT|ALL] uttrykk|NVL(uttrykk1,uttrykk2)
  SUM([DISTINCT|ALL] uttrykk) | SYSDATE | UPPER(streng) | USER
```

```
kolonneføring ::= [CONSTRAINT føringnavn] { [NOT]
  NULL|UNIQUE|PRIMARY KEY|
```

Appendiks G – Forenklet syntaks, PL/SQL

Appendiks G Forenklet syntaks, PL/SQL

```
REFERENCES [skjema.]tabellnavn[(kolonnenavn)] [ON DELETE  
CASCADE] [CHECK (betingelse)] [DISABLE]  
  
objektprivilegier ::= ALTER|DELETE|EXECUTE|INDEX|INSERT|REFE-  
RENCES|SELECT|UPDATE  
  
tabellføring ::= [CONSTRAINT føringnavn]({UNIQUE|PRIMARY  
KEY}(kolonnenavn [...]))  
|FOREIGN KEY(kolonnenavn[,...]) REFERENCES [skjema.]tabellnavn  
[(kolonnenavn [...])]  
[ON DELETE CASCADE] [CHECK (betingelse)] [DISABLE]  
  
uttrykk ::= NULL|[(skjema.)(tabellnavn.|viewnavn. kolonne-  
navn)|'tekst'|tall|(uttrykk [...])]  
|funksjon [(DISTINCT|ALL) uttrykk [...])|(uttrykk)  
uttrykk|uttrykk {*/|+|-| ||}uttrykk
```

```
basic scalar datatypes:  
VARCHAR2 (maximum_length)  
| NUMBER[(precision,scale)]  
| DATE | BOOLEAN | BINARY_INTEGER
```

```
datatypes:  
scalar datatype[ datatype_option] | variable_name%TYPE [  
datatype_option] | table.column_name%TYPE [ datatype_option]  
| table%ROWTYPE [ datatype_option]
```

```
datatype_option:  
(NOT NULL:=expression|:=expression)
```

```
variable declaration:  
variable_name datatype;
```

```
record:  
TYPE record_name  
IS RECORD (field_name datatype ;  
[field_name datatype; ...] );
```

```
pl/sql-block:  
BEGIN  
    statement; [statement; ..]  
[EXCEPTION  
    statement; [statement; ]  
END;
```

```
valg og løkker:  
IF condition THEN  
    statement; [statement; ...]  
[ELSIF condition THEN  
    statement; [statement; ..  
[ELSE  
    statement; [statement; .  
END IF;
```

```
LOOP  
    statement; [statement;  
EXIT [WHEN condition];  
[statement; [statement;  
END LOOP;
```

```

FOR counter IN [REVERSE]
  lower_bound .. upper_bound LOOP
  statement; [statement; ...]
END LOOP;

```

```

WHILE condition LOOP
  statement; [statement; ...]
END LOOP;

```

Definisjon av databaseobjekter:

```

CREATE [OR REPLACE] PROCEDURE prosedyrenavn
[(argument [IN|OUT|IN OUT] datatype
[,argument [IN|OUT|IN OUT] datatype ...])]
IS|AS
pl/sql-block

```

```

CREATE [OR REPLACE] FUNCTION function_name [(argument [IN]
  datatype
[,argument [IN] datatype ...])]
RETURN datatype
IS|AS
pl/sql-block

```

```

CREATE [OR REPLACE] PACKAGE package_name
IS|AS
<public type and item specifications>
<subprogram specifications>
END package_name;

```

```

CREATE [OR REPLACE] PACKAGE BODY package_name
IS|AS
  <private type and item specifications>
  <subprogram bodies>
END package_name;

```

select i PL/SQL:

```

SELECT select_list
  INTO {variable_name ,variable_name
        |record_name}
  FROM table
  WHERE condition;

```

trapping exceptions

```

EXCEPTION
  {WHEN exception [OR exception
    statement; [statement; ...]
  WHEN exception [OR exception
    statement; [statement; ...]
  WHEN OTHERS THEN
    statement; [statement; ...]
  WHEN OTHERS THEN statement;
    [statement; ...] }
    THEN
    THEN

```

```

user-defined exception (declaration)
exception_name EXCEPTION;

```

```

user-defined exception (raising):
RAISE exception_name;

```

```

associate non-predefined Oracle-server error and user_defined
exception:
exception_name EXCEPTION;

```

```

PRAGMA EXCEPTION_INIT(exception_name,
                        error_number);

```

Innebygd prosedyre som kan kommunisere en serverfeil til klienten (får en PL/SQL-blokk til å feile):

```

PROCEDURE RAISE_APPLICATION_ERROR (
  feilnummer IN NUMBER,
  feilmelding IN VARCHAR2 );

```

der feilnummer er et valgfritt tall fra -20 999 til 20 000

some predefined Oracle-server errors:

```

NO_DATA_FOUND|TOO_MANY_ROWS|INVALID_CURSOR|
ZERO_DIVIDE|DUP_VAL_ON_INDEX

```

markør:

```

CURSOR markoer_name
[(parameter_name datatype
[,parameter_name datatype ...])]
IS select_statement;

```

OPEN markoer_name;

```

FETCH markoer_name
  INTO {variable_name ,variable_name
        |record_name};

```

CLOSE markoer_name;

cursor attributes:

```

%ROWCOUNT|%FOUND|%NOTFOUND|%ISOPEN

```

event:

```

INSERT|UPDATE [OF column]|DELETE

```

timing:

```

BEFORE|AFTER

```

statement trigger:

```

CREATE [OR REPLACE] TRIGGER trigger_name
  timing event1 [OR event2 OR event3]
  ON table_name
  [DECLARE deklarasjoner ]
pl/sql-block;

```

row trigger:

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing event1 [OR event2 OR event3]
ON table_name
[REFERENCING OLD AS old /NEW AS new]
FOR EACH ROW [ WHEN condition]
[DECLARE deklarasjoner ]
pl/sql-block
```