Høgskolen i Telemark

EKSAMEN

865 DATABASER II

9.05.2005

Tid:	3 timer / 4 timer
Målform:	Norsk bokmål
Sidetall:	8 med framsida og vedlegg
Hjelpemiddel:	Ingen
Merknader:	Se neste side
Vedlegg:	*Forenklet syntaks, SQL* og *Forenklet syntaks, PL/SQL*

Eksamensresultata blir offentliggjort på nettet, via Arena høgskole. Passordet til Arena har alle studenter fått i eget brev ved semesterstart. I tillegg finn du eksamensresultatslister på utsiden av eksamenskontoret, men da trenger du kandidatnummeret ditt, så du bør notere dette på en lapp og legge den i lommeboka.

Avdeling for allmenne fag

Merknader

Oppgavesettet består av 4 oppgaver. Disse skal besvares av **alle** kandidater. Kandidater som har utvidet sluttprøve (4 timer) skal i tillegg besvare en flervalgsprøve. Denne deles ut separat.

Vektingen av oppgavene går fram av tabellen nedenfor. NB! Disse vektene bestemmer kun **delkarakteren** på sluttprøva. For studenter som har 3 timers sluttprøve, beregnes endelig karakter ved vekting av sluttprøve, obligatorisk oppgave og midtprøve. Sluttprøva teller da 60% av endelig karakter. For studenter med 4 timers utvidet prøve, teller sluttprøva 100%.

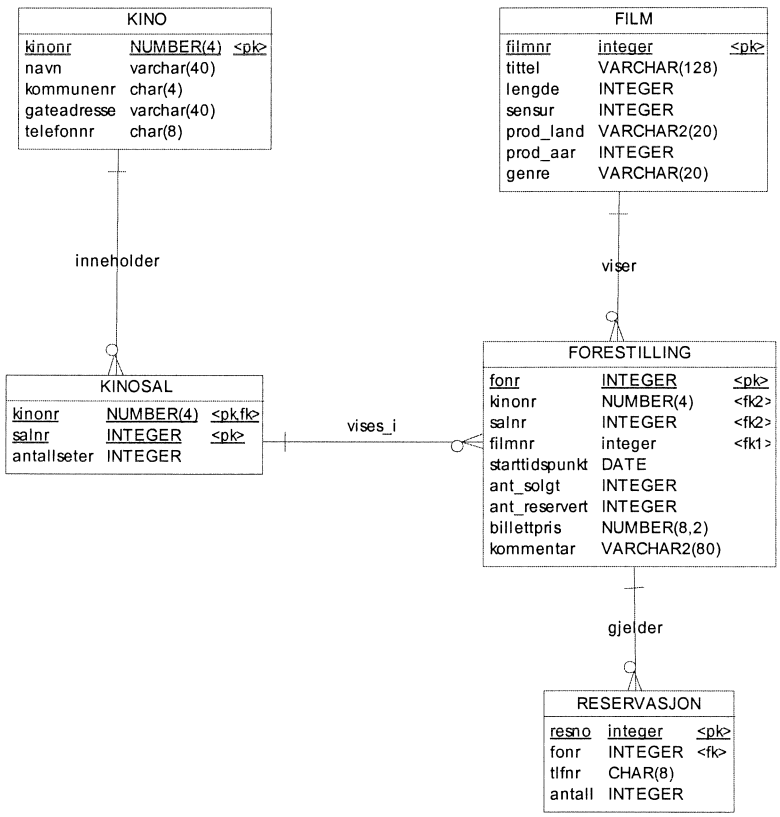
Oppgave	Vekting 3 timer	Vekting 4 timer
1	40 %	30 %
2	28 %	20 %
3	18 %	15 %
4	14 %	10 %
Flervalgsprøve	Skal ikkje gjøres	25 %
Sum	100%	100%

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

Lykke til!

Oppgave 1

I denne oppgaven skal du skrive PL/SQL-kode mot en Oracle database som benyttes for bestilling og salg av kinobilletter. Databasen inneholder opplysninger om *kinoer* og *filmer*, samt opplysninger om *forestillinger* og *kinosaler* filmene vises i. Til slutt inneholder databasen en tabell med *billettreservasjoner*. Databasen er modellert i følgende ER diagram:



ER-diagrammet over er omgjort til følgende tabeller:

- KINO(kinonr, navn, kommunenr, gateadresse, telefonnr)
- KINOSAL(kinonr*, salnr, antall_seter)
- FILM(filmnr, tittel, lengde, sensur, prod_land, prod_aar, genre)
- FORESTILLING(fonr, kinonr*, salnr*, filmnr*, starttidspunkt, ant_solgt, ant_reservert, billettpris, kommentar)
- RESERVASJON(resno, fonr*, tlfnr, antall)

Primærnøkler er understreket (og merket <pk> i ER modellen). Fremmednøkler merket med en stjerne (og merket <fk> i ER modellen). Datatypen DATE i Oracle lagrer både en dato og et klokkeslett.

Oracle databasen inneholder også en sekvens som er definert slik:

```
CREATE SEQUENCE sq_resnr INCREMENT BY 1 START WITH 1;
```

Deloppgaver

a) Regel: *Summen av antall solgte og reserverte billetter på en forestilling kan ikke overstige antall seter i kinosalen*. Skriv en trigger *tr_kinosal* som sikrer at denne regelen alltid overholdes dersom antall seter i en kinosal endres. Forklar hvorfor det ikke er fare for at denne regelen kan brytes ved registrering av nye kinosaler

b) Skriv en lagret funksjon *sf_reserver_plasser* som skal benyttes for å reservere plasser til en forestilling. Funksjonen skal ha datatype INTEGER, og den skal ha tre inn-parametre: *forestillingsnummer*, *telefonnr*, og *antall seter* man ønsker å reservere.

Før reservasjonen registreres i databasen, skal funksjonen først sjekke at det er nok ledige seter igjen på forestillingen. Hvis reservasjonen er gjennomførbar, skal en ny reservasjon registreres og tildeles reservasjonsnummer. Dette skal hentes automatisk fra Oracle sekvensen *sq_resnr*. Funksjonens returverdi skal være reservasjonsnummeret til den nylig registrerte reservasjonen. Hvis det ikke er nok ledige plasser på forestillingen, skal det ikke gjøres noen reservasjon. I stedet skal funksjonen returnere verdien -1. Denne verdien skal også returneres dersom det oppstår andre databasefeil under kjøring av funksjonen.

c) I billettluken skal personalet kunne selge billetter til forestillinger selv om man ikke har reservert på forhånd. Lag en prosedyre *sp_selg_billetter* som kan benyttes for dette. Prosedyren skal øke antall solgte billetter på forestillingen med det antallet som er solgt. Du må selv bestemme parametre til prosedyren. Prosedyren skal ha robust feilhåndtering, og må bl.a. sørge for at det ikke selges flere billetter enn antall ledige plasse til forestillingen på salgstidspunktet, dvs. plasser som ikke er solgt eller reservert. Du kan anta at reservasjoner slettes og registreres som solgte billetter i *forestilling*, etterhvert som de hentes.

e) Foreslå hvilke indekser som bør opprettes i denne databasen. Gi en kort begrunnelse for svaret ditt. Skriv SQL kode for å opprette **én av** de indeksene du foreslår. SQL koden skal sørge for at indeksen lagres fysisk i et tabellrom med navn KINO.

Oppgave 2

NB! Spørsmålene i **oppgave 2 skal besvares meget kort**, dvs. i stikkordsform eller med 1-2 setninger.

a) Skriv en SQL setning for å opprette en Oracle bruker med navn *peter* og passord *pan.*
b) Gi eksempel på ett *systemprivilegie* og ett *objektprivilegie* i Oracle.
c) Skriv en SQL setning for å gi brukeren *peter* rettighet til å lese data fra tabellen *kino* fra databasen i oppgave 1.
d) Du kjører SQL setningen **SELECT * FROM kinosal WHERE kinonr = 1**. Forklar hvordan Oracles aksessmetode TABLE ACCESS FULL finner dataene når denne setningen utføres.
e) I forbindelse med datavarehus benyttes begrepet ETL. Forklar stikkordsmessig hva forkortelsen står for og hvilke oppgaver som gjøres i ETL prosessen.
f) Forklar hva som menes med begrepet *Drill-down* i sammenheng med datavarehus.
g) Nevn to objektorienterte mekanismer i SQL-99 (SQL-3)
h) Forklar forskjellen på kommandoen SHUTDOWN NORMAL og SHUTDOWN ABORT i Oracle.

Oppgave 3

a) Strukturerte data fra en relasjonsdatabase (for eksempel fra en tabell eller et spørreresultat), kan lagres som et XML dokument. Det er to alternative metoder som kan benyttes for å representere kolonneverdier i XML. Beskriv kort disse to metodene og hvilke fordeler/ulemper de har.
b) Tabellen under viser data fra én rad i tabellen *kino*. Skriv et XML dokument som representerer data i denne ene raden. Bruk én av representasjonsmåtene fra pkt. a).

Kinonr	Navn	Kommunenr	Gateadresse	telefonnr
1	Colosseum	0301	Fr. Nansens vei 6	82030000

c) Forklar hvorfor det kan være hensiktsmessig å lagre XML dokumenter i en database, og hvordan dette kan gjøres.

Oppgave 4

a) Forklar kort forskjellen på *instansfeil* og *mediefeil* i Oracle. Forklar bl.a. hva som kan utløse disse feilene, og hvordan feilene kan resultere i tapte data.

b) Forklar hvordan gjenoppretting (recovery) av en database kan gjøres etter en *mediefeil*. Forklar spesielt hvordan datafiler, loggfiler og arkivlogfiler inngår i en slik gjenopprettingsprosess.

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[,...])]} [,...]
  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

datatyper::= 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 føringsnavn]{[NOT]
  NULL|UNIQUE|PRIMARY KEY|
```

```
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
```

Appendiks G – Forenklet syntaks, PL/SQL

Appendiks G – Forenklet syntaks, PL/SQL

```
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 ...] THEN
      statement; [statement; ...]
   [WHEN exception [OR exception ...] THEN
      statement; [statement;...] ...]
   [WHEN OTHERS THEN
      statement;[statement;...]]
   |WHEN OTHERS THEN statement;
      [statement; ...] }
```

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