Avdeling for allmenne fag

EKSAMEN

865 DATABASER II

24.05.2004

Tid:	3 timer / 4 timer
Målform:	Bokmål/nynorsk
Sidetal:	7 med framsida
Hjelpemiddel:	Ingen
Merknader:	Se første side i hver språkversjon!
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.

Bokmål

Merknader

Oppgavesettet består av 5 oppgaver. Oppgave 5 skal kun besvares av de som har fritak for underveisevaluering, og dermed får utvidet sluttprøve (4 timer). Denne oppgaven er merket: "**Skal kun besvares av de som har 4 timers sluttprøve**".

Vektingen av oppgavene i oppgavesettet går fram av tabellen under. NB! Disse vektene bestemmer kun **delkarakteren** på sluttprøva. For de som har 3 timers sluttprøve, beregnes endelig karakter ved vekting av sluttprøve og midtprøve. (Hjemmeoppgave er annulert som evalueringskomponent.) Sluttprøva teller da 80% av endelig karakter i kurset. For de med 4 timers prøve teller sluttprøva 100%.

Oppgave	Vekting 3 timer	Vekting 4 timer
1	30 %	20 %
2	20 %	15 %
3	25 %	20 %
4	25 %	20 %
5	Skal ikkje gjøres	25 %

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 i et enkelt fakturasystem for en salgsbedrift. Databasen inneholder opplysninger om bedriftens *kunder,* og hvilke *produkter* bedriften selger. Dessuten inneholder databasen opplysninger om utgående *fakturaer* som bedriften sender til sine kunder. Databasen inneholder følgende tabeller og kollonner:

kunder (kundenr, kundenavn, kontaktperson, telefonnr, betalingsdager)
produkter (produktkode, beskrivelse, stk_pris)
faktura (fakturanr, kundenr, fakturadato, forfallsdato)*
fakturalinjer (fakturanr, produktkode*, antall, stk_pris, samlet_pris)*

Primærnøkler er understreket og fremmednøkler marker med *. Tabellen *faktura* inneholder en rad for hver faktura. Tabellen *fakturalinjer* har et mange-til-en forhold mot *faktura* og inneholder en rad for hvert linje på fakturaen (dvs. for hvert produkt som er solgt på fakturaen).

Oracle databasen inneholder også en sekvens som er definert slik:

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

a)

Feltet *betalingsdager* i tabell kunder angir hvor mange dager (etter *fakturadato*) kunden har på å betale fakturaen. **Skriv en lagret funksjon** *sf_betfrist* med to parametre: *p_kundenr* og p*_fakturadato*. Funksjonen skal returnere en dato som angir betalingsfristen for den aktuelle kunden, dvs. *p_fakturadato* + antall betalingsdager for kunden. (Tips: I Oracle kan antall dager "legges til" en dato direkte med + operatoren.) Dersom funksjonen kalles med et kundenr som ikkje er registrert i databasen, skal funksjonen returnere NULL.

b)

Skriv én trigger som utfører følgende:

- automatisk gir nye fakturaer et nytt fakturanr fra sekvensen *sq_fakturanr*.
- sørger for at nye fakturaer automatisk får korrekt forfallsdato. Dvs. at feltet *forfallsdato* gis korrekt verdi basert på feltet *fakturadato* og kundens betalingsdager.

c)

Skriv en lagret prosedyre *sp_skriv_faktura* som skriver ut en faktura for et gitt *fakturanr*. Prosedyren skal skrive ut *fakturanr, fakturadato, forfallsdato* samt *kundenr* og *kundenavn* for den kunden fakturaen gjelder. Deretter skal alle fakturalinjene på den aktuelle fakturaen skrives ut sortert på *produktkode*. Hver fakturalinje skal vise *produktkode, beskrivelse, antall, stk_pris* og *samlet_pris*. Til slutt skal fakturaens samlede totalbeløp (summen av *samlet_pris* for alle fakturalinjene) skrives ut på en egen linje. Utskrift skal skje ved hjelp av DBMS_OUTPUT og lay-out på fakturaen spiller ingen rolle i denne oppgaven!

Oppgave 2

Tema: Optimalisering og aksessteknikker

a) Forklar hvilke oppgaver *optimalisator* (optimizer) har i et databasesystem og hvilke mekanismer den benytter for å utføre disse oppgavene. Bruk evt. Oracle som eksempel.

b) Anta at tabellen *kunder* fra oppgave 1 inneholder 100.000 rader og at tabellen er indeksert på feltet *kundenr* med en B-tre indeks. Følgende SQL setning kjøres:

```
SELECT * FROM kunder WHERE kundenr = 100;
```

Beskriv kort to alternative aksessmetoder/aksessteknikker som databasesystemet kan benytte for å utføre denne spørringen, hva som skiller de to og hvilken av dem som trolig vil være mest effektiv.

Oppgave 3

Tema: Datavarehus

a) Forklar forskjellen på et *operasjonellt system* og et *beslutningsstøttesystem* (informasjonssystem), og hvorfor det er hensiktsmessig å skille disse i separate databaser.

b) Tegn en skisse som viser hvilke komponenter som kan inngå i et beslutningsstøttesystem (informasjonssystem) basert på et **datavarehus**. Forklar hvilken oppgave de enkelte komponentene har og hvordan de henger sammen. Forklaringen bør som minimumum omfatte begrepene *datavarehus*, *datamarked*, *ETL*, *OLAP*.

Oppgave 4

Tema: Objektrelasjonelle databaser

a) Nevn to eksempler på databaseanvendelser der objektrelasjonelle databaser er bedre egnet enn relasjonsdatabaser. Forklar også kort hvorfor.

b) SQL standarden SQL3 (SQL-99) utvider relasjonsmodellen med objektorienterte mekanismer. Beskriv **tre** av disse mekanismene. Gi evt. også eksempler i SQL3 eller Oracle syntaks.

Slutt på oppgaveteksten for de som har 3 timers prøve!

Oppgave 5 - skal kun besvares av de som har 4 timers sluttprøve!

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

Tegn en skisse som viser oppbygging av en Oracle-server og hvilke fysiske komponenter som inngår. Forklar oppgavene til de enkelte komponentene i serveren og sammenhengen mellom dem. Du bør som minimum få med følgende begreper: *System Global Area, bakgrunnsprosesser, database, lytteprosess, tjenerprosess.*

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

Se beskrivelsen av databasen i oppgave 1. Foreslå hvilke indekser som bør opprettes i denne databasen. Gi en kort bekrunnelse for svaret. Skriv SQL kode for å oppretten én av de indeksene du foreslår, i Oracle. SQL koden skal sørge for at indeksen lagres fysisk i et tabellrom med navn FAKTURA.

c) Skal kun besvares av de som har 4 timers sluttprøve

Det skal lages en PL/SQL pakke *pk_faktura* som inneholder funksjonen *sf_betfrist* og prosedyren *sp_skriv_faktura* fra oppgave 1. Skriv PL/SQL koden for **pakkespesifikasjonen (pakkehodet)** til denne pakken. (Du skal altså ikke skrive koden for pakkekroppen.)

d) **Skal kun besvares av de som har 4 timers sluttprøve**

Skriv et **SQL*Plus klientprogram** som leser *fakturanr* fra bruker, og som deretter skriver ut fakturaen ved hjelp av prosedyren *sp_skriv_faktura* fra oppgave 1.

Slutt på oppgaveteksten for de som har 4 timers prøve!

Nynorsk

Merknader

Oppgåvesettet inneheld 5 oppgåver. Oppgåve 5 skal berre løysast av dei som er fritekne for undervegs-evaluering, og som difor har utvida sluttprøve (4 timar). Denne oppgåva er merkt: "**Skal berre løysast av dei som har 4 timars sluttprøve**".

Vektinga av oppgåvene i oppgåvesettet finn du i tabellen nedanfor. NB! Desse vektene styrer berre **delkarakteren** på sluttprøva. For dei som har 3 timars sluttprøve, vert endelig karakter rekna ut ved å vekte sluttprøva og midtprøva. (Hjemmeoppgåva er annullera som evalueringskomponent.) Sluttprøva tel då 80% av endeleg karakter i kurset. For dei med 4 timars prøve, tel sluttprøva 100%.

Oppgåve	Vekting 3 timar	Vekting 4 timar
1	30 %	20 %
2	20 %	15 %
3	25 %	20 %
4	25 %	20 %
5	Skal ikkje gjerast	25 %

Les oppgåveteksten godt, og prøv å svare på så mange deloppgåver som mogeleg. Tenk enkelt!

Lukke til!

Oppgåve 1

I denne oppgåva skal du skrive PL/SQL-kode mot ein Oracle database som vert nytta i eit enkelt fakturasystem for ei salsverksemd. Databasen inneheld opplysningar om kundane i verksemda, og kva slag *produkt* verksemda sel. Dessutan inneheld databasen opplysningar om utgåande *faktura* som verksemda sender til kundene sine. Databasen inneheld desse tabellene og kollonnene (tabell- og kollonnenamn er ikkje omsett til nynorsk):

kunder (kundenr, kundenavn, kontaktperson, telefonnr, betalingsdager)
produkter (produktkode, beskrivelse, stk_pris)
faktura (fakturanr, kundenr, fakturadato, forfallsdato)*
fakturalinjer (fakturanr, produktkode*, antall, stk_pris, samlet_pris)*

Primærnøklar er streka under og framandnøklar er merkt med *. Tabellen *faktura* inneheld ei rad for kvar faktura. Tabellen *fakturalinjer* har eit mange-til-ein forhold mot faktura og inneheld ei rad for kvar line på fakturaen (dvs. for kvart produkt som er seld på fakturaen).

Oracle databasen inneheld og ein sekvens som er definert slik:

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

a)

Feltet *betalingsdager* i tabell kunder styrer kor mange dager (etter *fakturadato*) kunden har for å betale fakturaen. **Skriv ein lagra funksjon** *sf_betfrist* med to parametre: *p_kundenr* og p*_fakturadato*. Funksjonen skal returnere ein dato som gjer betalingsfristen for den aktuelle kunden, dvs. *p_fakturadato* + antall betalingsdagar for kunden. (Tips: I Oracle kan du leggje antall dager til ein dato direkte med + operatoren.). Dersom funksjonen vert kalla med eit kundenr som ikkje er registrera i databasen, skal funksjonen returnere NULL.

b)

Skriv ein trigger som gjer dette:

- automatisk gjev nye fakturaer eit nytt *fakturanr* frå sekvensen *sq_fakturanr*.
- syter for at nye fakturaer automatisk får rett forfallsdato. Dvs. at feltet *forfallsdato* får rett verdi basera på feltet *fakturadato* og kundens betalingsdagar.

c)

Skriv ei lagra prosedyre *sp_skriv_faktura* som skriv ut ein faktura for eit *fakturanr*. Prosedyren skal skrive ut *fakturanr, fakturadato, forfallsdato*, og dessutan *kundenr* og *kundenavn* for den kunden fakturaen gjeld. Dernest skal alle fakturalinene på den aktuelle fakturaen skrivast ut sortert på *produktkode*. Kvar av fakturalinene skal syne *produktkode, beskrivelse, antall, stk_pris* og *samlet_pris*. Til slutt skal totalbeløp på fakturaen (summen av *samlet_pris* for alle fakturalinene) skrivast ut på ei eiga line. Utskrift skal gjerast med hjelp av DBMS_OUTPUT, og lay-out på fakturaen spelar inga rolle i denne oppgåva!

Oppgåve 2

Tema: Optimalisering og aksessteknikkar

c) Forklar kva for oppgåver *optimalisator* (optimizer) har i eit databasesystem og kva for mekanismer denne nytter for å løyse desse oppgåvene. Bruk evt. Oracle som døme.

d) Rekn med at tabellen *kunder* fra oppgåve 1 inneheld 100.000 rader og at tabellen er indeksera på feltet *kundenr* med ein B-tre indeks. SQL setningen nedanfor vert så kjørt:

```
SELECT * FROM kunder WHERE kundenr = 100;
```

Tal kort om to alternative aksessmetodar/aksessteknikkar som databasesystemet kan nytte for å utføre denne spørringa, kva som skil dei to, og kva for ein av dei som truleg vil vere mest effektiv.

Oppgåve 3

Tema: Datavarehus

c) Forklar skilnaden på eit *operasjonellt system* og eit *beslutningsstøttesystem* (informasjonssystem), og kvifor det er formålsteneleg å skilje desse i separate databasar.

d) Teikn ei skisse som syner kva for komponenter som kan inngå i eit beslutningsstøttesystem (informasjonssystem) som er basera på eit **datavarehus**. Tal om kva for oppgåver dei einskilde komponentane har og korleis dei heng saman. Du bør som minimum omtale omgrepa *datavarehus*, *datamarked*, *ETL*, *OLAP*.

Oppgåve 4

Tema: Objektrelasjonelle databasar

c) Nemn to døme på bruksområde databasar der objektrelasjonelle databaser er betre egna enn relasjonsdatabasar. Forklar og kort kvifor.

d) SQL standarden SQL3 (SQL-99) utvidar relasjonsmodellen med objektorienterte mekanismar. Gjer greie for **tre** av disse mekanismane. Gje evt. og døme i SQL3 eller Oracle syntaks.

Slutt på oppgåveteksten for dei som har 3 timars prøve!

Oppgåve 5 - skal berre løystast av dei som har 4 timars sluttprøve!

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

Teikn ei skisse som syner oppbygga av ein Oracle-server og kva for fysiske komponentar som inngår i denne. Gjer greie for oppgåvene til dei einskilde komponentane i serveren og samanhangen mellom dei. Du bør som minimum få med følgjande omgrep: *System Global Area, bakgrunnsprosessar, database, lytteprosess, tjenerprosess*.

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

Sjå omtala av databasen i oppgåve 1. Foreslå kva for indeksar som bør opprettast i denne databasen. Gje ei kort grunngjeving for svaret. Skriv SQL kode for å opprette **ein** av dei indeksane du foreslår, i Oracle. SQL koden skal syte for for at indeksen vert lagra fysisk i eit tabellrom med namnet FAKTURA.

c) Skal berre løysast av dei som har 4 timars sluttprøve

Det skal lagast ei PL/SQL pakke *pk_faktura* som inneheld funksjonen *sf_betfrist* og prosedyra *sp_skriv_faktura* fra oppgåve 1. Skriv PL/SQL koden for **pakkespesifikasjonen (pakkehodet)** til denne pakka. (Du skal altså ikkje skrive koden for pakkekroppen.)

d) Skal berre løysast av dei som har 4 timars sluttprøve

Skriv eit **SQL*Plus klientprogram** som les *fakturanr* frå bruker, og som dernest skriv ut fakturaen ved hjelp av prosedyra *sp_skriv_faktura* frå oppgåve 1.

Slutt på oppgåveteksten for dei som har 4 timars prøve!

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
```