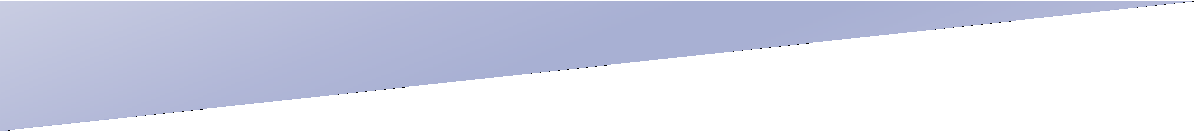
1. ***PLSQL 2***
2. ***Select into***
3. ***Anchored declaration***
4. ***Cursorvariabler***
5. ***Olika typer av cursors***
6. ***Cursorattribut***
7. ***Cursorloop***
8. ***Datatypen record: %rowtype***
9. ***Cursor for loop***
10. ***Några kodexempel***
11. ***Felhantering***
12. ***Olika typer av undantag***
13. ***no\_data\_found***
14. ***raise\_application\_error***

##### Pär Douhan, [pdo@du.se](mailto:pdo@du.se)



Vi börjar med cursorvariabler

|  |  |  |
| --- | --- | --- |
|  | **# knr: fnamn: enamn:**  **email:** | **number(8) varchar2(40) varchar2(60)**  **varchar2(30)** |
| **mobil:** | **varchar2(10)** |
| **Maria exekverar följande anonyma PL/SQL‐block vi tidpunkt T1:**  **declare** |  |  |

v\_email varchar2(30); v\_mobil varchar2(10);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kundnr** | **fnamn** | **enamn** | **email** | **mobil** |
| 56482 | Rolf | Björk | [rb@gmail.com](mailto:rb@gmail.com) | 0730211142 |
| 89658 | Malin | Ek | [maek@du.se](mailto:maek@du.se) | 0703658974 |
| 58476 | Jossef | Mdoud | [josse](mailto:jossef@du.se)[f@du.se](mailto:f@du.se) | 0730533347 |

begin



"select into" fungerar bara med "singelton select"

select email,mobil into v\_email,v\_mobil from kund

where knr = 56482;

dbms\_output.put\_line('mobil: '||v\_mobil||','||'email: '||v\_email);

end;

/

mobil: varchar2(15)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **kundnr** | **fnamn** | **enamn** | **email** | **mobil** |
| 56482 | Rolf | Björk | [rb@gmail.com](mailto:rb@gmail.com) | 0730211142 |
| 89658 | Malin | Ek | [maek@du.se](mailto:maek@du.se) | 0703658974 |
| 58476 | Jossef | Mdoud | [josse](mailto:jossef@du.se)[f@du.se](mailto:f@du.se) | 0730533347 |

###### Någon annan skriver följande kod vid tidpunkt T2:

alter table kund

modify mobil varchar2(15);

update kund

set mobil = '+46730277742' where knr = 56482; commit;

### Vad händer vid tidpunkt T3, när Maria exekverar sitt anonyma PL/SQL‐block igen?

declare

v\_email varchar2(30); v\_mobil varchar2(10);

begin

select email,mobil into v\_email,v\_mobil from kund

where knr = 56482;

dbms\_output.put\_line('mobil: '||v\_mobil||','||'email:

end;

/

ORA-06502: PL/SQL: numeric or value error

select length('+46730277742') from dual;



'||v\_email);

LENGTH('+46730277742')

‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

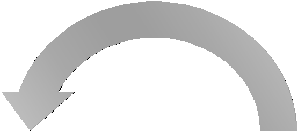
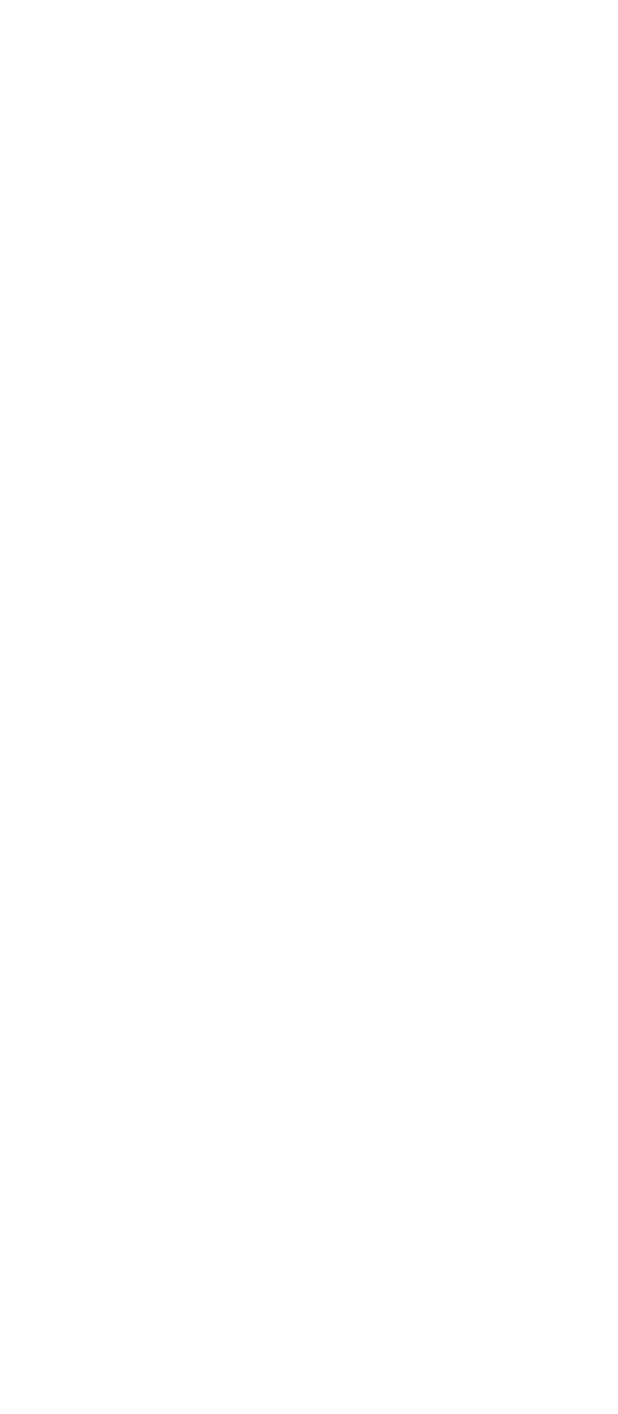
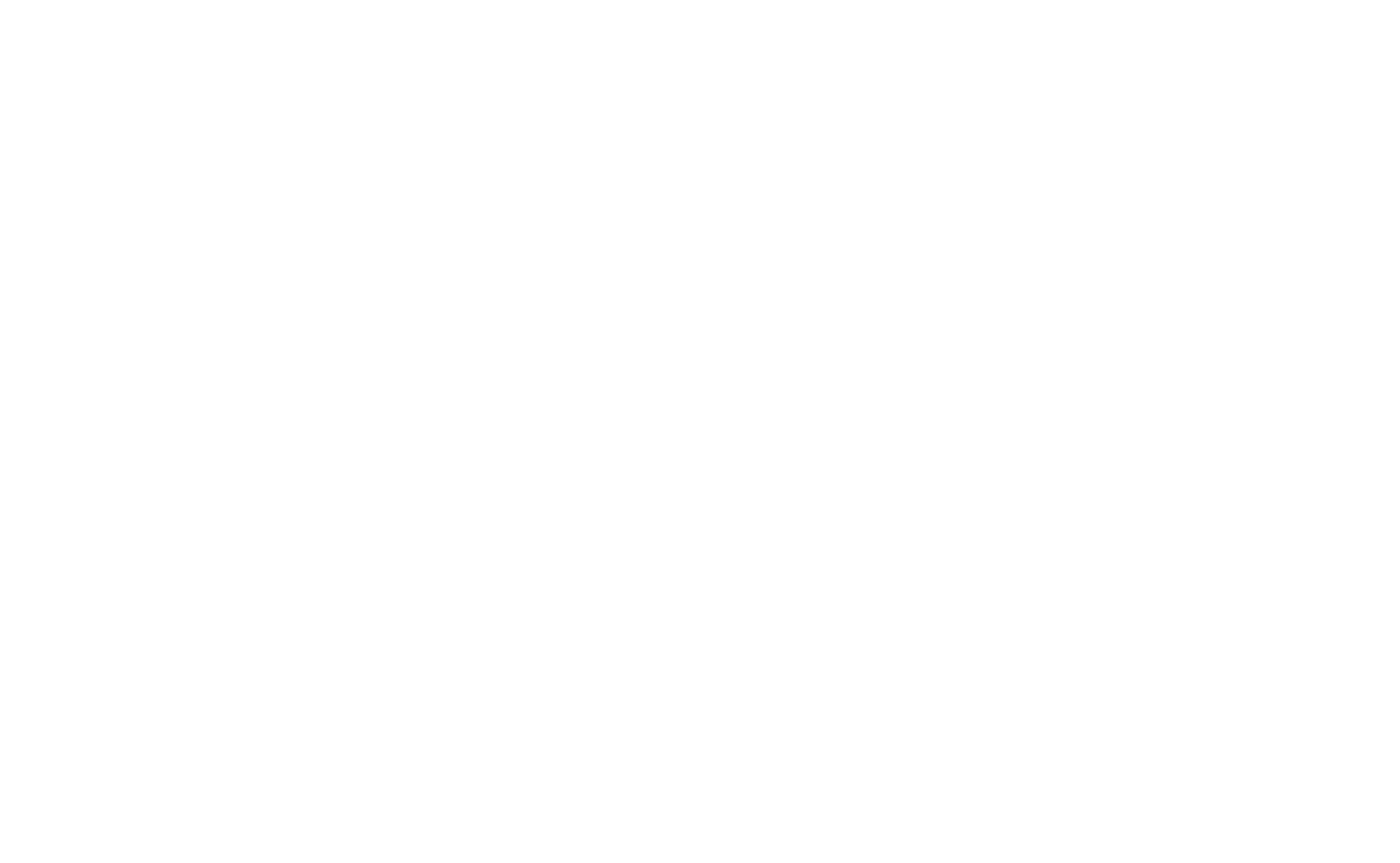
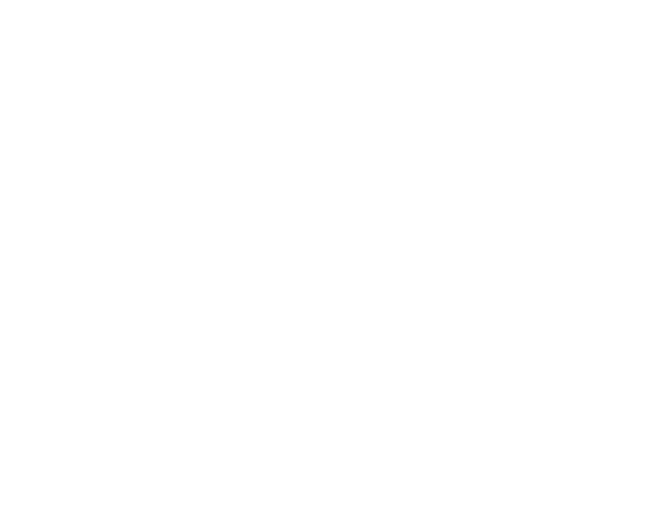
12

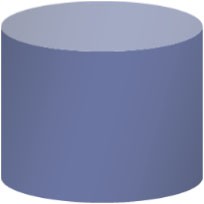
v\_mobil varchar2(10);

**Lösningen på problemet är *anchored declaration***

# %type

declare



v\_email kund.email%type; v\_mobil kund.mobil%type;

**tabell kolumn**

***Hämtar datatypen från databasen.***

# Synonymer

### cursor ~ recordset ~ recordgroup ~ resultset ~ länkad lista

***"en packe med rader"*, d.v.s. resultatet av en *select‐sats*.**

### cursor‐deklaration

select kursnr,poäng,nivå from kurs;

KURSNR POÄNG NIVÅ

***first record***

### cursor = resultatet av sql‐satsen

**100**

**101**

**102**

**103**

**104**

**105**

**5**

**5**

**5**

**10**

**5**

**5**

**A A B C D B**

***last record***

# När behöver vi använda oss av en *cursor*?

#### "en packe med rader"



***När vår SQL‐sats reurnerar fler än en rad.***

##### Om vi gör en ***singelton select***, d.v.s. where‐villkoret matchar en kolumn som innehåller unika värden (PK eller unique) så kan vi använda ***select‐into***.

Får vi tillbaka ***fler än en rad***, så kan vi ***inte*** använda ***select‐into***, utan vi *måste* använda en ***cursor***.

# Statiska cursors

*‐ Refererar alltid till samma SQL.*

# Dynamiska cursors

*‐ Kan referera till olika SQL‐satser vid olika tidpunkter.*

# Statiska cursors ‐ två olika typer

## Implicit cursor

*‐ Hanteras av DBMS.*

## Explicit cursor



1. *Deklarera den*
2. *Öppna den*
3. *Loopa igenom raderna*
4. *Stänga den*

declare

cursor c\_kunder is select fnamn,enamn,adress

from kund;

begin

null; end;

/

**namnet på cursorn**

declare

cursor c\_storkunder is

select fnamn,enamn,adress from kund

where knr in(select knr

from kundorder

where ordnr in(select orderrad.ordnr

from orderrad,artikel

where orderrad.artnr = artikel.artnr

having sum(artikel.pris \* orderrad.antal) > 20000 group by orderrad.ordnr));

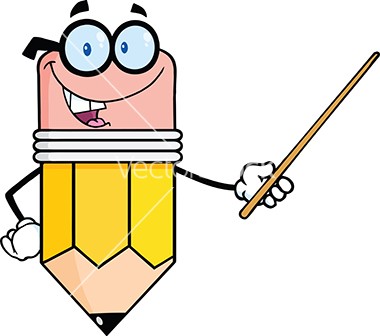
# %isopen

**%found**

# %notfound

**%rowcount**

### true, false true, false true, false number

select fnamn,enamn from kund;

FNAMN ENAMN

NULL

%rowcount = 1

%rowcount = 2

**olof maria tomas hans**

**andersson andersson kvist rosenboll**

%rowcount = 3

%rowcount = 4

EOF, %notfound = true

declare

O L F E E C

cursor c\_kundmobil is select fnamn,enamn,mobil

from kund; v\_fnamn kund.fnamn%type;

v\_enamn kund.enamn%type;

v\_mobil kund.mobil%type;

begin

if not c\_kundmobil%isopen then open c\_kundmobil;

end if;

-- öppna cursorn

loop -- loopa igenom resultatet av SQL-satsen

fetch c\_kundmobil -- ställ pekaren på första raden

into v\_fnamn,v\_enamn,v\_mobil; -- hämta data från aktuell rad till variablerna

exit when c\_kundmobil%notfound; -- sluta loopa när EOF inträffar dbms\_output.put\_line(v\_fnamn||', '||v\_enamn||', mobil: '||v\_mobil);

end loop;

close c\_kundmobil; -- stäng cursorobjektet end;

/

declare

cursor c\_kundmobil is select

fnamn,enamn,mobil

from kund;

v\_fnamn kund.fnamn%type;

v\_enamn kund.enamn%type; v\_mobil kund.mobil%type;

rec c\_kundmobil%rowtype;

begin

irriterande med

flera variabler!

**rec**

|  |  |  |
| --- | --- | --- |
| fnamn | enamn | mobil |

if not c\_kundmobil%isopen then open c\_kundmobil;

end if; loop

fetch c\_kundmobil

into rec;v\_fnamn,v\_enamn,v\_mobil; exit when c\_kundmobil%notfound;

dbms\_output.put\_line(rec.fnamn||', '||rec.enamn||', mobil: '||rec.mobil); end loop;

close c\_kundmobil;

end;

/

**For loop innebär minst kod!**

### Cursorn behöver varken öppnas eller stängas. Detta sker implicit.

declare

cursor c\_kundmobil is select fnamn,enamn,mobil

from kund;

begin

for rec in c\_kundmobil loop

dbms\_output.put\_line(rec.fnamn||', '||rec.enamn||', mobil: '||rec.mobil); end loop;

end;

/

###### Uppgift:

Skriv ett *PLSQL‐program* som kopierar över alla artiklar, som kostar ***mer än*** 2000 kr, från tabellen ARTIKEL till tabellen LYXARTIKEL.

### ARTIKEL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **artnr** | **artnamn** | **varumärke** | **lagerantal** | **pris** |
| 1 | Såskastrull | Sofoni | 5 | 498 |
| 2 | Traktörpanna | Hackman | 13 | 795 |
| 3 | Atlantis kastrull | Demeyere | 13 | 2999 |
| 4 | Opera Såskastrull i koppar 18 cm | Ruffoni | 2 | 2599 |
| 5 | Prima Matera Såskastrull i koppar | de Buyer | 2 | 2595 |

**LYXARTIKEL**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **artnr** | **artnamn** | **varumärke** | **lagerantal** | **pris** |
|  |  |  |  |  |

**declare**

**cursor c\_lyxartiklar is select artnr,artnamn,varumärke,lagerantal,pris from artikel**

O L F E E C

**where pris > 2000;**

**v\_rec c\_lyxartiklar%rowtype; begin**

**if not c\_lyxartiklar%isopen then open c\_lyxartiklar;**

**end if;**

**loop**

**fetch c\_lyxartiklar into v\_rec;**

**exit when c\_lyxartiklar%notfound;**

**insert into lyxartikel(artnr,artnamn,varumärke,lagerantal,pris) values(v\_rec.artnr,v\_rec.artnamn,v\_rec.varumärke,v\_rec.lagerantal,v\_rec.pris);**

**end loop;**

**close c\_lyxartiklar;**

**commit;**

**dbms\_output.put\_line('Kopieringen är klar!');**

**end;**

##### Samma sak men med en for loop:

**declare**

**cursor c\_lyxartiklar is select artnr,artnamn,varumärke,lagerantal,pris from artikel**

**where pris > 2000;**

**begin**

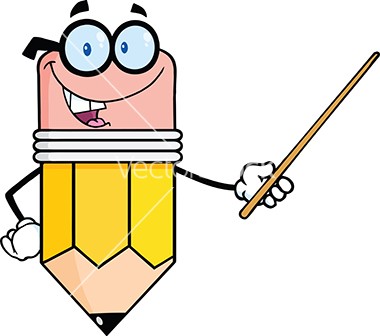
**for v\_rec in c\_lyxartiklar loop**

**insert into lyxartikel(artnr,artnamn,varumärke,lagerantal,pris) values(v\_rec.artnr,v\_rec.artnamn,v\_rec.varumärke,v\_rec.lagerantal,v\_rec.pris);**

**end loop; commit;**

**dbms\_output.put\_line('Kopieringen är klar!');**

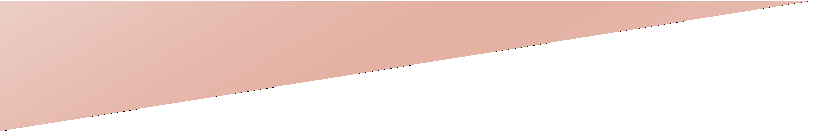
**end;**

anonymous block completed Kopieringen är klar!

select \*

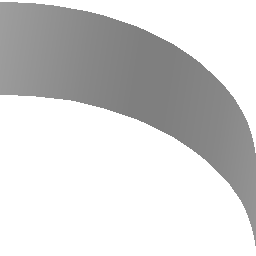
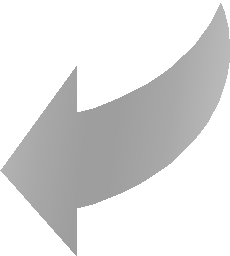
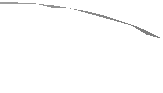
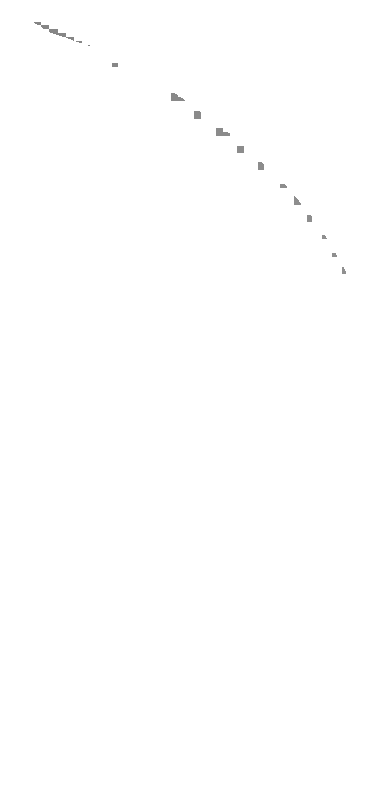
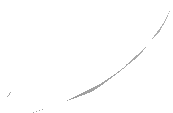
from lyxartikel;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ARTNR** | **ARTNAMN** | **VARUMÄRKE** | **LAGERANTAL** | **PRIS** |
| ‐‐‐‐‐ | ‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐ | ‐‐‐‐‐‐‐‐‐‐‐ | ‐‐‐‐‐‐‐‐‐‐ | ‐‐‐‐‐‐‐‐ |
| 3 | Atlantis kastrull | Demeyere | 13 | 2999 |
| 4 | Opera Såskastrull i koppar 18 cm | Ruffoni | 2 | 2599 |
| 5 | Prima Matera Såskastrull i koppar | de Buyer | 2 | 2595 |



PLSQL: Felhantering

### Felhantering exceptions = errors



#### När ett fel inträffar i programblocket, så förflyttas kontrollen av programblocket till felhanteringsdelen.

declare

v\_mobil kund.mobil%type; begin

select mobil

into v\_mobil \*\*\* E R R O R \*\*\* inträffar här from kund

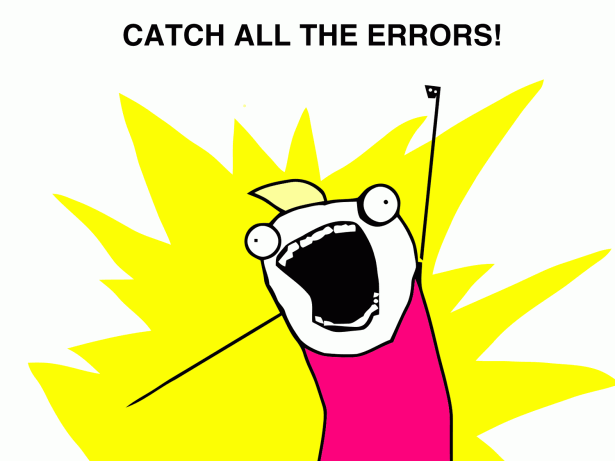
where knr = 3;

exception -- MÅSTE ligga sist i programblocket

\*\*\* E R R O R \*\*\* hanteras här

end;

/

Rätt ***exception handler*** (felhanterare) spåras via en namnreferens till det inträffade felet.

Om ett exception (undantag) inträffar och det inte finns någon exception handler, så kommer ***others*** att ta hand om det.

Om inte others heller finns, så kommer en ***felkod*** (error code) och ett ***felmeddelande*** (error message) att visas.

# Tre olika typer av exceptions:

#### Fördefinierade Oracle Server Exceptions

Har ett fördefinierat namn, t. ex. ***no\_data\_found***, ***zero\_divide***, ***to\_many\_rows, cursor\_allready\_open, value\_error*** etc. ~ 20 st, "raised implicitly" av servern.

#### Icke fördefinierade

Standard Oracle Server‐fel som inte är namngivna av systemet. Har en felkod och ett felmeddelande.

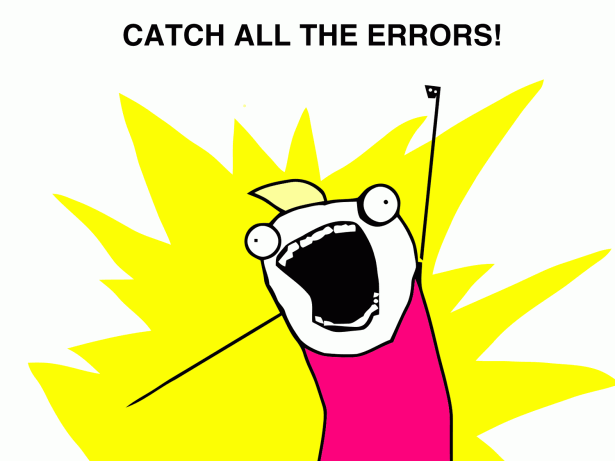
"raised implicitly" av servern.

#### Användardefinierade

Deklareras och namnges av användaren (programmeraren). "raised explicitly" av användaren.

**sqlcode sqlerrm**

**felkoden number**

**felmeddelandet varchar2**

declare

v\_mobil kund.mobil%type; begin

select mobil into v\_mobil from kund

det finns ingen kund med PK = 138

v\_mobil innehåller således null

where knr = 138; exception

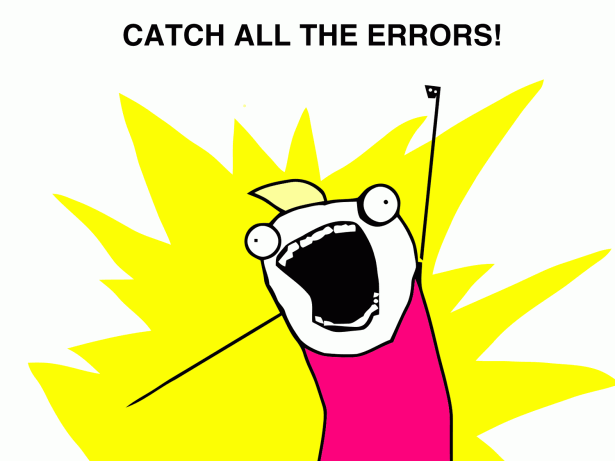
when others then

dbms\_output.put\_line('fel: '||sqlerrm||', felkod: '||sqlcode);

end;

/

fel: ORA-01403: data saknas, felkod: 100

Undantaget ***no\_data\_found*** "kastas" när en ***select into‐sats inte***

##### returnerar någon data.

Detta gäller bara när ***select into‐satsen inte*** jobbar mot en

***aggregatfunktion***.

studera:

declare

v\_antal number(1); begin

returnerar alltid 1 eller 0

select count(\*) into v\_antal from kund

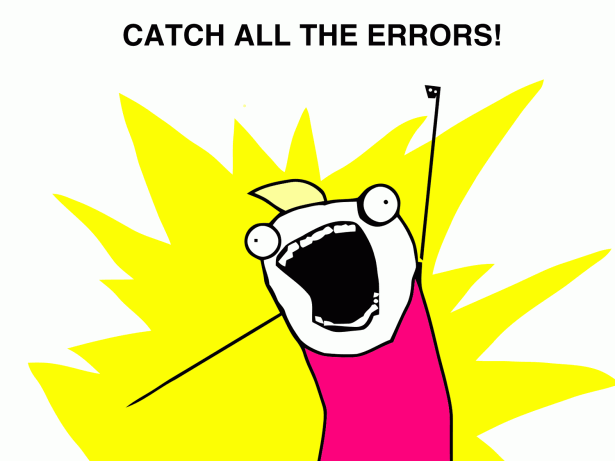
where knr = 138; exception

inget kommer att fångas här

when no\_data\_found then dbms\_output.put\_line('Data saknas!');

end;

/

Undantaget ***no\_data\_found*** fungerar ***inte*** i en cursor!

### Här måste vi hantera undantaget själva:

studera:

declare

cursor c\_kundmobil is select mobil

from kund

begin

where knr = 2;

for rec in c\_kundmobil loop if rec.mobil is null then

raise\_application\_error(-20001,'Mobilnummer saknas!'); end if;

dbms\_output.put\_line(', mobil: '||rec.mobil); end loop;

end;

/

**Vi kan anropa proceduren raise\_application\_error**

raise\_application\_error(-20001,'ERROR!, here is some text about it');

**Egen felkod: ‐20001 till ‐20999**

**Eget felmeddelande**



## Vi kan skapa s.k. användardefinerade undantag:

Studera:

declare

cursor c\_kundmobil is select mobil

from kund where knr = 2;

egendefinierat undantag

no\_mobil exception; begin

for rec in c\_kundmobil loop if rec.mobil is null then

undantaget kastas explicit

raise no\_mobil; end if;

dbms\_output.put\_line('mobil: '||rec.mobil); end loop;

exception

undantaget hanteras

when no\_mobil then

dbms\_output.put\_line('Kunden saknar mobiltelefon!');

end;

/

