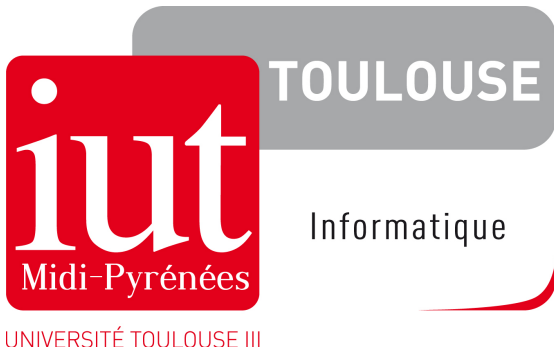
# M2106 : Programmation et administration des bases de données

## Cours 5/6 – Exceptions et curseurs

Guillaume Cabanac

guillaume.cabanac@univ-tlse3.fr


version du 9 octobre 2013

# Exceptions et curseurs

# Exceptions et curseurs

# Le mécanisme de gestion des exceptions

## Exceptions prédéfinies

### Définition

Une exception est une erreur logicielle détectée à l'exécution du programme. Le traitement de l'exception est réalisé *via* le gestionnaire d'exception. Une exception non traitée remonte la pile des appels de sous-programmes et cause l'arrêt du programme avec un code d'erreur.

En PL/SQL, chaque bloc **`begin ... end`** peut définir un gestionnaire d'exception dans la section nommée **`exception`**.

Quelques exceptions sont prédéfinies :

| Exception | Code | Description |
|---|---|---|
| `case_not_found` | `ORA-06592` | Aucun des choix d'un `case` sans `else` ne convient. |
| `invalid_number` | `ORA-01722` | Échec de conversion `(var)char2` → `number` |
| `value_error` | `ORA-06502` | Erreur d'arithmétique sur un `number`. |
| `zero_divide` | `ORA-01476` | Division par zéro. |
| `...` | `...` | ... |

# Le mécanisme de gestion des exceptions

## Traitement d'une exception levée par le système

```
declare
  vTmp number ;
  vRes number ;
begin
  -- il se trouve que vTmp prend la valeur 0...
  ...
  select count(*)/x into vRes from ... ;
  ...
exception
  when zero_divide then
    -- interception de l'exception zero_divide ici
  when ... then
    ...
  when others then
    -- traitement d'une exception non interceptée préalablement
    dbms_output.put_line('Erreur inconnue '||sqlcode||' : '||sqlerrm) ;
end ;
/
```

NB : Dans un bloc **`exception`**, deux variables système sont consultables :

`sqlcode` Le code d'erreur de l'exception prédéfinie ou personnalisée (`ORA-XXXXX`).

`sqlerrm` Le message d'erreur correspondant au `sqlcode`.

# Le mécanisme de gestion des exceptions

## Exceptions personnalisées

### Définition

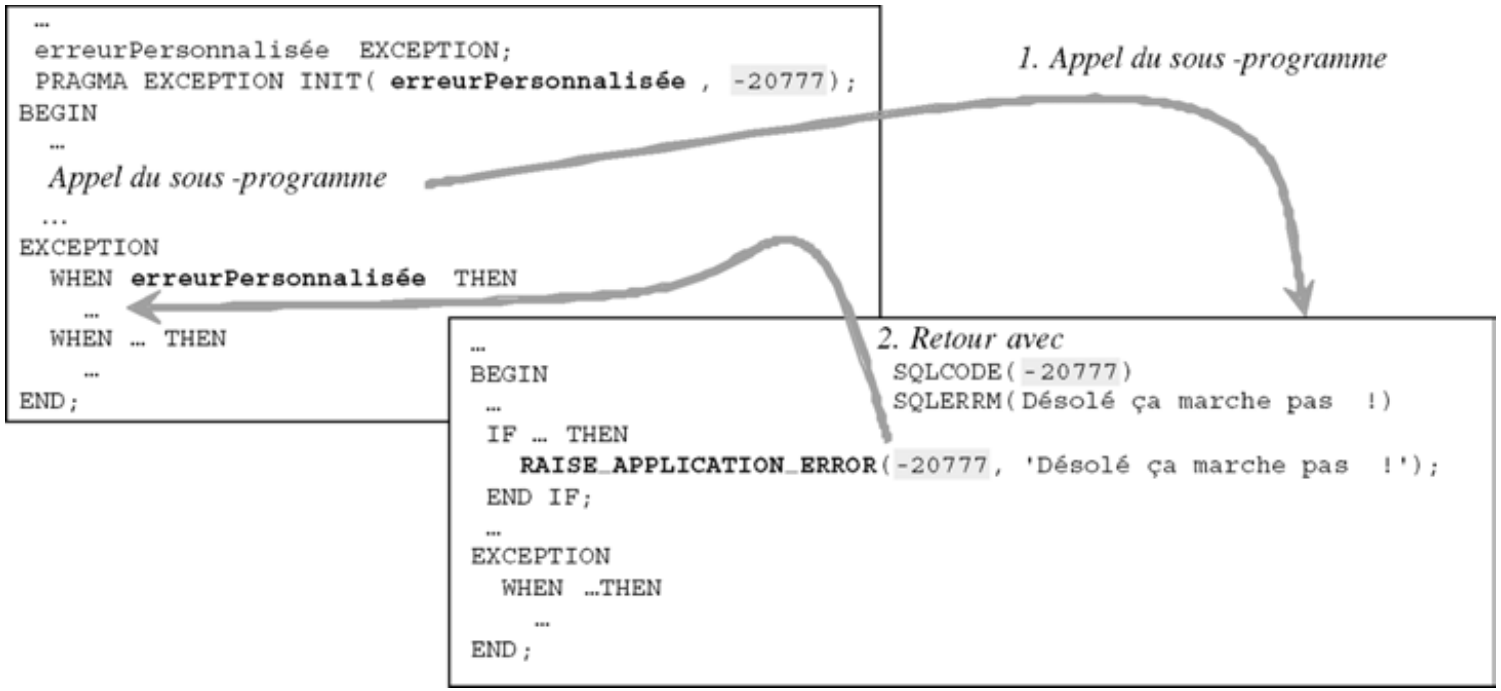
Le programmeur peut définir ses propres exceptions qui seront levées avec l'instruction **raise** ou la procédure **raise_application_error**.

```
-- Création d'une exception personnalisée et déclenchement avec raise
declare
  -- déclaration de l'exception en la nommant
  aucunInscrit exception ;
  -- association d'un code d'erreur dans [-20999 ; -20000] à l'exception
  pragma exception_init(aucunInscrit, -20001) ;
begin
  ...
  if vNb = 0 then
    -- déclenchement de l'exception
    raise aucunInscrit ;
  end if ;
  ...
end ;
```

```
-- Création et déclenchement d'une exception personnalisée avec raise_application_error
begin
  if vNb = 0 then
    -- déclenchement de l'exception
    raise_application_error(-20001, 'Aucun chien n''est inscrit') ;
  end if ;
end ;
```

# Le mécanisme de gestion des exceptions

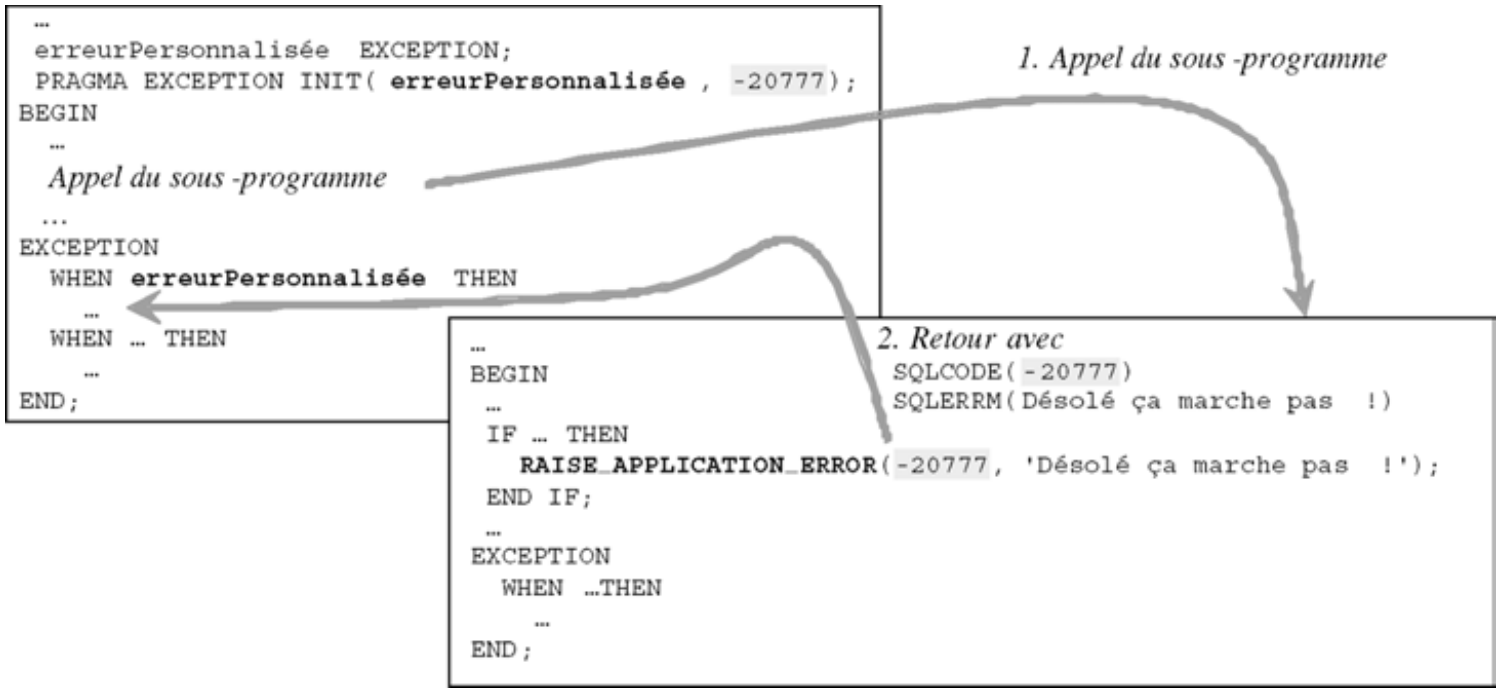
## `raise` et `raise_application_error`




Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 332)

# Le mécanisme de gestion des exceptions

## `raise` et `raise_application_error`

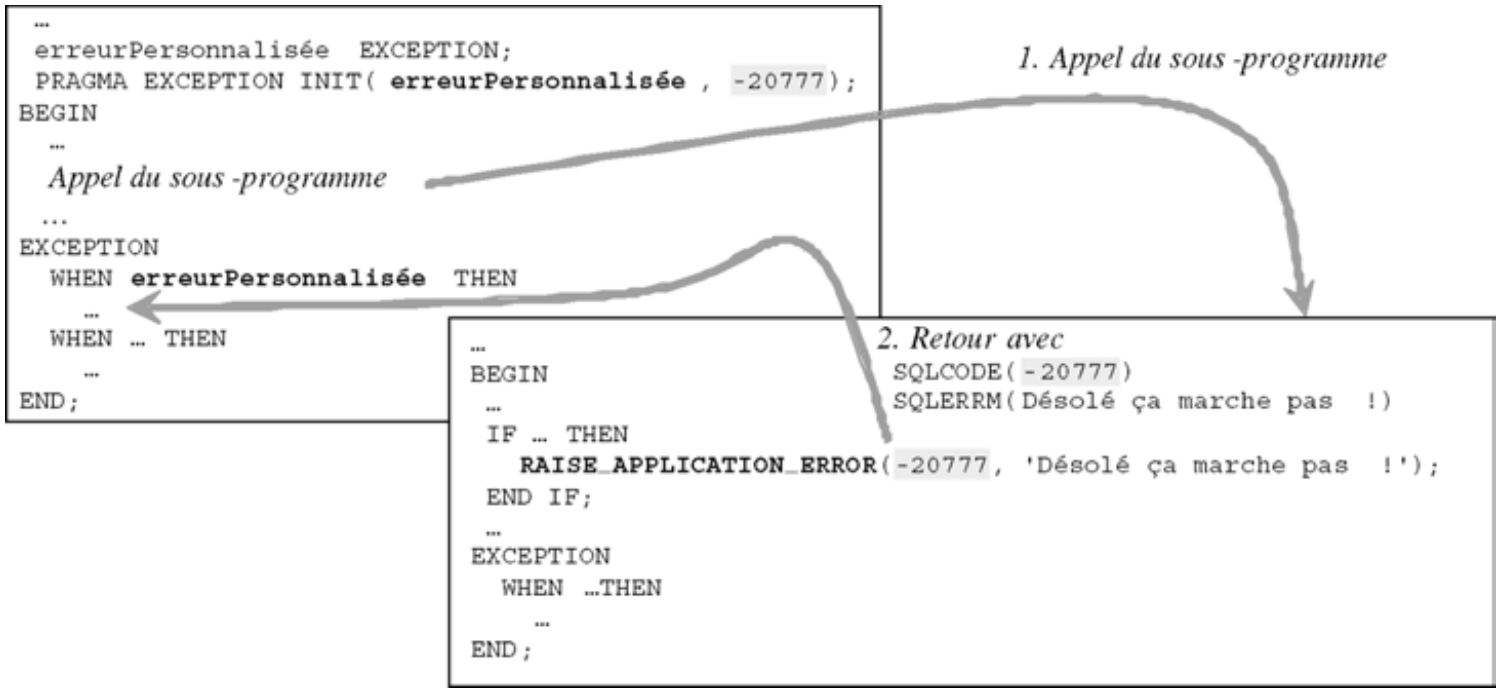



Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 332)

⚠ Attention à bien distinguer un message destiné à l'utilisateur (`put_line`) d'une erreur logicielle (`raise` et `raise_application_error`).

# Le mécanisme de gestion des exceptions

## `raise` et `raise_application_error`




Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 332)

⚠ Attention à bien distinguer un message destiné à l'utilisateur (`put_line`) d'une erreur logicielle (`raise` et `raise_application_error`).

☠ Formellement interdit : signaler une erreur par un message (`put_line`).

# Exceptions et curseurs

# Extraction de données

## Cadrage

Il existe deux façons d'extraire des données en PL/SQL :

### ← Affection

- Affectation de valeurs
- Extrait une seule ligne
- 1 syntaxe :
  `select ... into ... from`

### ↻ Curseur

- Boucle sur un résultat
- Extrait plusieurs lignes
- 3 syntaxes :
  1. manuel : `open ...`
  2. semi-autom. : `cursor ...`
  3. automatique : `for ...`

# Extraction de données

## Cadrage

Il existe deux façons d'extraire des données en PL/SQL :

### ← Affection

- Affectation de valeurs
- Extrait une seule ligne
- 1 syntaxe :
  `select ... into ... from`

### ↻ Curseur

- Boucle sur un résultat
- Extrait plusieurs lignes
- 3 syntaxes :
  1. manuel : `open ...`
  2. semi-autom. : `cursor ...`
  3. automatique : `for ...`

Ne pas utiliser une affectation pour plusieurs lignes !
Déclenche l'exception `too_many_rows` à l'exécution...

# Extraction de données

## Cadrage

Il existe deux façons d'extraire des données en PL/SQL :

### ← Affection

- Affectation de valeurs
- Extrait une seule ligne
- 1 syntaxe :
  `select ... into ... from`

### ↻ Curseur

- Boucle sur un résultat
- Extrait plusieurs lignes
- 3 syntaxes :
  1. manuel : `open ...`
  2. semi-autom. : `cursor ...`
  3. automatique : `for ...`

☠ Ne pas utiliser une affectation pour plusieurs lignes !
Déclenche l'exception `too_many_rows` à l'exécution...

⚠ Éviter l'utilisation d'un curseur pour une seule ligne.

# Exceptions et curseurs

# Affectation

## Syntaxes

```
select * into variableRowType from ...                -- var %rowtype

select A[, ..., Z] into var1[, ..., varZ] from ... -- var %type, number...
```

```
declare
  vTatouage    chien.tatouage%type default 1664 ;  -- stocke une valeur
  vUnChien     chien%rowtype ;                     -- stocke une ligne
  vMaxPrime    participation.prime%type ;
  vNbSup100    number ;
  vDernierConc concours.dateC%type ;
begin
  -- Affectation de l'intégralité d'une ligne dans un %rowtype
  select * into vUnChien from chien where tatouage = 1664 ; -- rowtype : on récupère nom & sexe
  -- Affectation d'une valeur dans un %type
  select max(prime) into vMaxPrime from participation where tatouage = 1664 ;
  -- Affectation de 2 valeurs
  select count(*), max(dateC) into vNbSup100, vDernierConc
  from participation p, concours c
  where p.idC = c.idC
    and tatouage = 1664 and prime > 100 ;
  -- Affichage du résultat
  dbms_output.put_line(vUnChien.nom||' '||'de sexe '
    || case vUnChien.sexe when 'M' then 'masculin' else 'féminin' end
    || ' dont le plus gros gain est de '||vMaxPrime||' €. Gains supérieurs à '
    || ' 100 € : '|| vNbSup100||'. Dernier bon concours : '||vDernierConc)  ;
end ;
/

-- Iench de sexe féminin dont le plus gros gain est de 1234 €. Gains supérieurs à  100 € : 1.  ✂
      Dernier bon concours : 30-SEP-13
```

# Exceptions et curseurs

# Curseur manuel

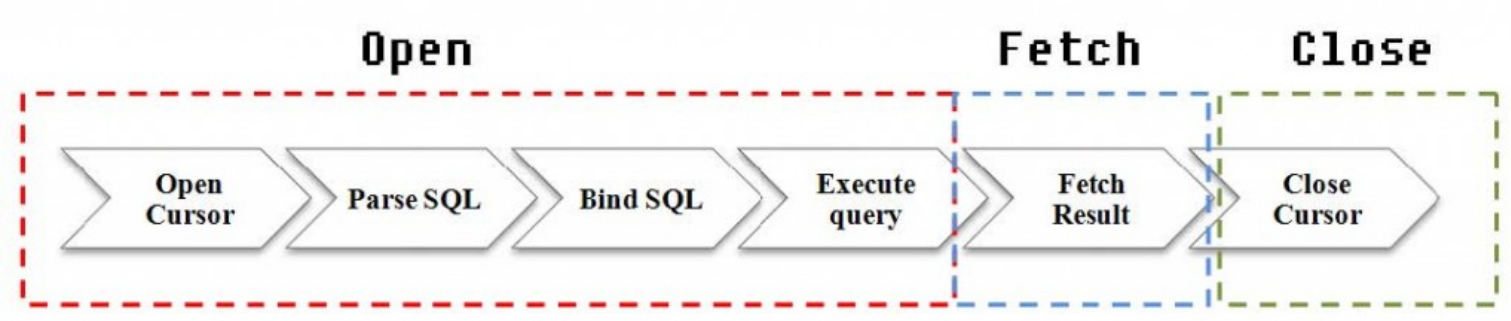
## Les instructions utilisées

| Instruction | Description |
|---|---|
| `cursor nomCur is requête ;` | Déclare le curseur. |
| `open nomCur ;` | Ouvre le curseur en chargeant les lignes en mémoire centrale. Aucune exception levée si la requête ne renvoie aucune ligne. |
| `fetch nomCur into var1, ..., varN ;` | Avance d'une ligne et affecte les valeurs de la ligne aux variables. |
| `close nomCur ;` | Ferme le curseur. |
| `nomCur%isOpen` | `true` si le curseur est ouvert, `false` sinon. |
| `nomCur%notFound` | `true` si le dernier `fetch` n'a pas renvoyé de ligne (fin de curseur), `false` sinon. |
| `nomCur%found` | `true` si le dernier `fetch` a renvoyé une ligne, `false` sinon. |
| `nomCur%rowCount` | Nombre de lignes parcourues jusqu'à présent. |

# Curseur manuel

## Cycle de vie

| | |
|---|---|
| cursor | Déclaration du nom du curseur et de la requête. |
| open | Ouverture du curseur et exécution de la requête. |
| fetch | Récupération des lignes, une par une. |
| close | Fermeture du curseur et libération des ressources. |




http://www.dbanotes.com/wp-content/uploads/2011/06/Introduction-to-Oracle-11g-Cursors_img_2-1024x221.jpg

# Curseur manuel

## Exemple 1 : curseur manuel

```
-- Exemple d'utilisation d'un curseur manuel
create procedure hallOfFame as
  cursor curTopChiens is select c.tatouage, nom, sum(classement) totClass
                         from chien c, participation p
                         where c.tatouage = p.tatouage
                           and classement is not null
                         group by c.tatouage, nom
                         order by 3, 2 ;
  vTopChien curTopChiens%rowtype ;
begin
  dbms_output.put_line('Hall of Fame de la Société Canine') ;
  open curTopChiens ;
  -- Récupérer le premier chien
  fetch curTopChiens into vTopChien ;
  while curTopChiens%found loop
    dbms_output.put('Numéro '||curTopChiens%rowCount||' : '||vTopChien.nom) ;
    dbms_output.put(' ('||vTopChien.tatouage||') avec '||vTopChien.totClass||' classement') ;
    if vTopChien.totClass > 1 then
      dbms_output.put('s') ;
    end if ;
    dbms_output.new_line ;
    -- Passer au chien suivant (NE PAS L'OUBLIER !!!)
    fetch curTopChiens into vTopChien ;
  end loop ;
  close curTopChiens ; -- NE PAS l'OUBLIER !!!
end hallOfFame ;
/

call hallOfFame() ;

-- Hall of Fame de la Société Canine
-- Numéro 1 : Iench (1664) avec 1 classement
-- Numéro 2 : Cerbère (666) avec 7 classements
```

# Curseur manuel

## Exemple 2 : affectation et curseur manuel paramétré

```
create procedure afficherParticipants(pIdC in concours.idC%type) as
  -- Définition d'un curseur paramétré avec idC
  cursor curPart(pcIdC in concours.idC%type) is select tatouage
                                                from participation
                                                where idC = pIdC
                                                order by classement nulls last, tatouage ;
  vUnChien  chien.tatouage%type ;
  vVille    concours.ville%type ;
  vDateC    concours.dateC%type ;
  vDotation number ; -- car la longueur peut dépasser concours.prime%type
begin
  select ville, dateC, sum(prime) into vVille, vDateC, vDotation
  from concours c, participation p
  where c.idC = p.idC
    and c.idC = pIdC
  group by c.idC, ville, dateC ;
  dbms_output.put_line('Concours de '||vVille||' du '||vDateC) ;
  dbms_output.put_line('Dotation : '||vDotation) ;
  -- Liste des participants
  open curPart(pIdC) ;
  -- Récupérer le premier chien
  fetch curPart into vUnChien ;
  while curPart%found loop
    dbms_output.put_line(curPart%rowCount||'. '||vUnChien) ;
    -- Passer au chien suivant (NE PAS L'OUBLIER !!!)
    fetch curPart into vUnChien ;
  end loop ;
  close curPart ; -- NE PAS l'OUBLIER !!!
end afficherParticipants ;
/

call afficherParticipants(10) ;
```

# Exceptions et curseurs

# Curseur semi-automatique

## Exemple 1 : curseur semi-automatique

### Le choc des simplifications

~~variable~~ ~~`%rowtype`~~, ~~`open`~~, ~~`fetch`~~, ~~`close`~~ → ~~`for`~~

NB : ne pas déclarer la variable utilisée dans le `for` :-)

```
-- Exemple d'utilisation d'un curseur semi-automatique
create procedure hallOfFame as
  cursor curTopChiens is select c.tatouage, nom, sum(classement) totClass
                         from chien c, participation p
                         where c.tatouage = p.tatouage
                           and classement is not null
                         group by c.tatouage, nom
                         order by 3, 2 ;
begin
  dbms_output.put_line('Hall of Fame de la Société Canine') ;
  for vTopChien in curTopChiens loop
    dbms_output.put('Numéro '||curTopChiens%rowCount||' : '||vTopChien.nom) ;
    dbms_output.put(' ('||vTopChien.tatouage||') avec '||vTopChien.totClass||' classement') ;
    if vTopChien.totClass > 1 then
      dbms_output.put('s') ;
    end if ;
    dbms_output.new_line ;
  end loop ;
end hallOfFame ;
/

call hallOfFame() ;

-- Hall of Fame de la Société Canine
-- Numéro 1 : Iench (1664) avec 1 classement
-- Numéro 2 : Cerbère (666) avec 7 classements
```

# Curseur semi-automatique

## Exemple 2 : affectation et curseur semi-automatique paramétré

```
create procedure afficherParticipants(pIdC in concours.idC%type) as
  -- Définition d'un curseur paramétré avec idC
  cursor curPart(pcIdC in concours.idC%type) is select tatouage
                                                from participation
                                                where idC = pIdC
                                                order by classement nulls last, tatouage ;
  vUnChien  chien.tatouage%type ;
  vVille    concours.ville%type ;
  vDateC    concours.dateC%type ;
  vDotation number ; -- car la longueur peut dépasser concours.prime%type
begin
  select ville, dateC, sum(prime) into vVille, vDateC, vDotation
  from concours c, participation p
  where c.idC = p.idC
    and c.idC = pIdC
  group by c.idC, ville, dateC ;
  dbms_output.put_line('Concours de '||vVille||' du '||vDateC) ;
  dbms_output.put_line('Dotation : '||vDotation) ;
  -- Liste des participants
  for vUnChien in curPart(pIdC) loop
    dbms_output.put_line(curPart%rowCount||'. '||vUnChien.tatouage) ;
  end loop ;
end afficherParticipants ;
/

call afficherParticipants(10) ;
```

# Exceptions et curseurs

# Curseur automatique

## Exemple 1 : curseur automatique

### Le choc des simplifications 2 : le retour

~~`cursor`~~ → requête dans le `for`.

```
-- Exemple d'utilisation d'un curseur automatique
create procedure hallOfFame as
  vI number := 0 ;  -- Il n'y a plus de déclaration de curseur ici
begin
  dbms_output.put_line('Hall of Fame de la Société Canine') ;
  for vTopChien in (select c.tatouage, nom, sum(classement) totClass
                    from chien c, participation p
                    where c.tatouage = p.tatouage
                      and classement is not null
                    group by c.tatouage, nom
                    order by 3, 2)
  loop
    vI := vI + 1 ; -- remplace %rowCount car on n'y a plus accès
    dbms_output.put('Numéro '||vI||' : '||vTopChien.nom) ;
    dbms_output.put(' ('||vTopChien.tatouage||') avec '||vTopChien.totClass||' classement') ;
    if vTopChien.totClass > 1 then
      dbms_output.put('s') ;
    end if ;
    dbms_output.new_line ;
  end loop ;
end hallOfFame ;
/
call hallOfFame() ;

-- Hall of Fame de la Société Canine
-- Numéro 1 : Iench (1664) avec 1 classement
-- Numéro 2 : Cerbère (666) avec 7 classements
```

# Curseur automatique

## Exemple 2 : affectation et curseur automatique paramétré par synchronisation

Le paramétrage du curseur se fait désormais *via* la synchronisation sur `pIdC`.

```
create procedure afficherParticipants(pIdC in concours.idC%type) as
  vVille    concours.ville%type ;
  vDateC    concours.dateC%type ;
  vDotation number ; -- car la longueur peut dépasser concours.prime%type
  vI number := 0 ;
begin
  select ville, dateC, sum(prime) into vVille, vDateC, vDotation
  from concours c, participation p
  where c.idC = p.idC
    and c.idC = pIdC
  group by c.idC, ville, dateC ;
  dbms_output.put_line('Concours de '||vVille||' du '||vDateC) ;
  dbms_output.put_line('Dotation : '||vDotation) ;
  -- Liste des participants
  for vUnChien in (select tatouage
                   from participation
                   where idC = pIdC
                   order by classement nulls last, tatouage)
  loop
    vI := vI + 1 ; -- remplace %rowCount car on n'y a plus accès
    dbms_output.put_line(vI||'. '||vUnChien.tatouage) ;
  end loop ;
end afficherParticipants ;
/

call afficherParticipants(10) ;
```

# Bilan sur les 3 types de curseurs

## Curseur manuel

- ☠ Code long
- ☠ Bugs potentiels dus aux divers oublis quasi-inévitables

# Bilan sur les 3 types de curseurs

## Curseur manuel

- ☠ Code long
- ☠ Bugs potentiels dus aux divers oublis quasi-inévitables

## Curseur semi-automatique

- ☺ 4 instructions en moins !
- ⚠ La requête est parfois « loin » (déclaration) de son utilisation (boucle)

# Bilan sur les 3 types de curseurs

## Curseur manuel

- ☠ Code long
- ☠ Bugs potentiels dus aux divers oublis quasi-inévitables

## Curseur semi-automatique

- ☺ 4 instructions en moins !
- ⚠ La requête est parfois « loin » (déclaration) de son utilisation (boucle)

## Curseur automatique

- ☺ Encore 1 instruction en moins !
- ☺ La requête et son utilisation sont couplées dans la boucle
- ⚠ Pas de réutilisation de requête, contrairement au curseur déclaré

# Conseil d'un ami qui vous veut du bien

## Ne réinventez pas la roue !

« Il n'y a rien de plus inutile que de faire avec efficacité
quelque chose qui ne doit pas être fait du tout. » — Peter Drucker


http://www.videouniversity.com/files/reinventing_wheel.jpg

# Conseil d'un ami qui vous veut du bien

## Ne réinventez pas la roue !

« Il n'y a rien de plus inutile que de faire avec efficacité
quelque chose qui ne doit pas être fait du tout. » — Peter Drucker


http://www.videouniversity.com/files/reinventing_wheel.jpg

Ne recodez pas avec des boucles les sélections, les jointures, les regroupements. . . Exploitez la puissance de SQL !

# Sometimes less is more !

```
-- À NE PAS FAIRE : réinventer la roue !!! Ça "fonctionne" mais c'est sous-optimal peu maintenable
create function gainsMalesToulousainsKO return number as
  pVille      adherent.ville%type ;
  vTotalGains number := 0 ;
begin
 for ch in (select * from chien) loop
  if ch.sexe = 'M' then
    -- vérifie qu'il est à Toulouse
    select ville into pVille from adherent where idA = ch.idA ;
    if pVille = 'Toulouse' then
      for pa in (select * from participation) loop
        if ch.tatouage = pa.tatouage then
          vTotalGains := vTotalGains + pa.prime ;       -- Ce code est une HORREUR !
        end if ;                                        -- Ne faites **surtout pas** ça
      end loop ;
    end if ;
  end if ;
 end loop ;
 return vTotalGains ;
end gainsMalesToulousainsKO ;
/
```

# Sometimes less is more !

```
-- À NE PAS FAIRE : réinventer la roue !!! Ça "fonctionne" mais c'est sous-optimal peu maintenable
create function gainsMalesToulousainsKO return number as
  pVille      adherent.ville%type ;
  vTotalGains number := 0 ;
begin
 for ch in (select * from chien) loop
  if ch.sexe = 'M' then
    -- vérifie qu'il est à Toulouse
    select ville into pVille from adherent where idA = ch.idA ;
    if pVille = 'Toulouse' then
      for pa in (select * from participation) loop
        if ch.tatouage = pa.tatouage then
          vTotalGains := vTotalGains + pa.prime ;       -- Ce code est une HORREUR !
        end if ;                                        -- Ne faites **surtout pas** ça
      end loop ;
    end if ;
  end if ;
 end loop ;
 return vTotalGains ;
end gainsMalesToulousainsKO ;
/
```

```
-- Exploitation judicieuse de SQL, encapulation dans une fonction
create function gainsMalesToulousainsOK return number as
  vTotalGains number ;
begin
  -- calcul du résultat en pur SQL
  select nvl(sum(prime), 0) into vTotalGains
  from chien ch, participation pa, adherent ad
  where ch.tatouage = pa.tatouage
    and ch.idA = ad.idA
    and ville = 'Toulouse'
    and sexe = 'M' ;
  return vTotalGains ;
end gainsMalesToulousainsOK ;
/
```

# Exceptions et curseurs

# Le mécanisme de gestion des exceptions

## Exceptions prédéfinies liées à l'extraction de données

Parmi les exceptions prédéfinies, on trouve également :

| Exception | Code | Description |
|---|---|---|
| `dup_val_on_index` | `ORA-00001` | Violation de contrainte unique : UN ou PK. |
| `no_data_found` | `ORA-01403` | Requête ne retournant aucun résultat. |
| `rowtype_mismatch` | `ORA-06504` | Incompatibilité de types lors d'une affectation. |
| `too_many_rows` | `ORA-01422` | Requête retournant plusieurs lignes (`select into` ☠) |
| | `ORA-01400` | Tentative d'insertion d'une valeur `null` non autorisée. |
| | `ORA-01407` | Tentative de màj d'une valeur `null` non autorisée. |
| | `ORA-02290` | Transgression d'une contrainte `check`. |
| | `ORA-02291` | Tentative d'insertion d'une valeur de FK inexistante. |
| | `ORA-02292` | Tentative de suppression d'une valeur de FK pointée. |

# Le mécanisme de gestion des exceptions

## Exceptions prédéfinies liées à l'extraction de données

Parmi les exceptions prédéfinies, on trouve également :

| Exception | Code | Description |
|---|---|---|
| `dup_val_on_index` | `ORA-00001` | Violation de contrainte unique : UN ou PK. |
| `no_data_found` | `ORA-01403` | Requête ne retournant aucun résultat. |
| `rowtype_mismatch` | `ORA-06504` | Incompatibilité de types lors d'une affectation. |
| `too_many_rows` | `ORA-01422` | Requête retournant plusieurs lignes (`select into` ☠) |
| | `ORA-01400` | Tentative d'insertion d'une valeur `null` non autorisée. |
| | `ORA-01407` | Tentative de màj d'une valeur `null` non autorisée. |
| | `ORA-02290` | Transgression d'une contrainte `check`. |
| | `ORA-02291` | Tentative d'insertion d'une valeur de FK inexistante. |
| | `ORA-02292` | Tentative de suppression d'une valeur de FK pointée. |

⚠ Certaines erreurs ne sont pas associées à des exceptions prédéfinies. On est donc forcé de les intercepter :

- soit en déclarant une exception et en la liant au code d'erreur associé (**`pragma`**),
- soit dans le cas **`others`** du bloc **`exception`** (en testant pour être sûr de bien avoir été déclenché par l'exception attendue).

# Le mécanisme de gestion des exceptions

## Exceptions prédéfinies liées à l'extraction de données : exemple

```
create procedure ajouterParticipation(pTatouage in participation.tatouage%type,
                                      pConcours in participation.idC%type,
                                      pClass in participation.classement%type,
                                      pPrime in participation.prime%type) as
  fkInexistante exception ;
  pragma exception_init(fkInexistante, -2291) ; -- associe un nom à cette exception prédéfinie
  -- ORA-02291: integrity constraint FK... violated - parent key not found
  vTestException number ;
begin
  insert into participation values(pTatouage, pConcours, pClass, pPrime) ;
exception
  when fkInexistante then  -- PB de FK... oui, mais laquelle ?
    begin
      select tatouage into vTestException from chien where tatouage = pTatouage ;
    exception when no_data_found then
      raise_application_error(-20001, 'Erreur : tatouage '||pTatouage||' inexistant.') ;
    end ;
    begin
      select idC into vTestException from concours where idC = pConcours ;
    exception when no_data_found then
      raise_application_error(-20002, 'Erreur : concours '||pConcours||' inexistant.') ;
    end ;
  when dup_val_on_index then
    raise_application_error(-20003, 'Part. au conc. '||pConcours||' déjà notée pour '||pTatouage) ;
end ajouterParticipation ;
/

call ajouterParticipation(99, 10, 2, 512) ;
-- SQL Error: ORA-20001: Erreur : tatouage 99 inexistant.

call ajouterParticipation(51, 99, 2, 512) ;
-- SQL Error: ORA-20002: Erreur : concours 99 inexistant.

call ajouterParticipation(666, 10, 2, 512) ;
-- SQL Error: ORA-20003: Part. au conc. 10 déjà notée pour 666
```

# Le mécanisme de gestion des exceptions
## Compléments

Tableau 7-20 Deux exceptions traitées

| Code PL/SQL | Commentaires |
|---|---|
| `CREATE PROCEDURE procException1 (p_comp IN VARCHAR2) IS`<br>`  var1 Pilote.nom%TYPE;`<br>`BEGIN`<br>`  SELECT nom INTO var1 FROM Pilote`<br>`  WHERE comp = p_comp;`<br>`  DBMS_OUTPUT.PUT_LINE('Le pilote de la compagnie '`<br>`  || p_comp || ' est ' || var1);` | Requête déclenchant potentiellement deux exceptions prévues. |
| **`EXCEPTION`**<br>**`WHEN`** `NO_DATA_FOUND` **`THEN`**<br>`DBMS_OUTPUT.PUT_LINE('La compagnie ' ||`<br>`p_comp || ' n''a aucun pilote!');` | Aucun résultat renvoyé. |
| **`WHEN`** `TOO_MANY_ROWS` **`THEN`**<br>`DBMS_OUTPUT.PUT_LINE('La compagnie ' ||`<br>`p_comp || ' a plusieurs pilotes!');`<br>`END;` | Plusieurs résultats renvoyés. |

Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 322)

# Le mécanisme de gestion des exceptions
## Compléments

Tableau 7-21 Une exception traitée pour deux instructions

| Code PL/SQL | Commentaires |
|---|---|
| `CREATE PROCEDURE procException2`<br>`(p_brevet IN VARCHAR2, p_heures IN NUMBER) IS`<br>`var1 Pilote.nom%TYPE;`<br>`requete NUMBER := 1;`<br>`BEGIN`<br>`SELECT nom INTO var1 FROM Pilote`<br>`WHERE brevet = p_brevet;`<br>`DBMS_OUTPUT.PUT_LINE('Le pilote de ' ||`<br>`p_brevet || ' est ' || var1);` | Requêtes déclenchant potentiellement une exception prévue. |
| `requete := 2;`<br>`SELECT nom INTO var1 FROM Pilote`<br>`WHERE nbHVol = p_heures;`<br>`DBMS_OUTPUT.PUT_LINE('Le pilote ayant ' ||`<br>`p_heures || ' heures est ' || var1);` | |
| **`EXCEPTION`**<br>**`WHEN`** `NO_DATA_FOUND` **`THEN`**<br>`IF requete = 1 THEN`<br>`DBMS_OUTPUT.PUT_LINE('Pas de pilote de brevet : '`<br>`|| p_brevet);`<br>`ELSE`<br>`DBMS_OUTPUT.PUT_LINE('Pas de pilote ayant ce`<br>`nombre d''heures de vol : ' || p_heures);`<br>`END IF;` | Aucun résultat.<br>Traitement pour savoir quelle requête a déclenché l'exception. |
| **`WHEN`** `OTHERS` **`THEN`**<br>`DBMS_OUTPUT.PUT_LINE('Erreur d''Oracle ' ||`<br>**`SQLERRM`** `|| ' (' ||` **`SQLCODE`** `|| ')');`<br>`END;` | Autre erreur. |

Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 324)

# Le mécanisme de gestion des exceptions
## Compléments

Tableau 7-22 Bloc d'exceptions imbriqué

| Code PL/SQL | Commentaires |
|---|---|
| `CREATE PROCEDURE procException3`<br>`(p_brevet IN VARCHAR2, p_heures IN NUMBER) IS`<br>`var1 Pilote.nom%TYPE;`<br>`BEGIN`<br>`BEGIN`<br>`SELECT nom INTO var1 FROM Pilote`<br>`WHERE brevet = p_brevet;`<br>`DBMS_OUTPUT.PUT_LINE('Le pilote de ' \|\| p_brevet`<br>`\|\| ' est ' \|\| var1);` | Bloc imbriqué. |
| `EXCEPTION`<br>`WHEN NO_DATA_FOUND THEN`<br>`DBMS_OUTPUT.PUT_LINE('Pas de pilote de brevet : '`<br>`\|\| p_brevet);`<br>`WHEN OTHERS THEN`<br>`DBMS_OUTPUT.PUT_LINE('Erreur d''Oracle ' \|\|`<br>`SQLERRM \|\| ' (' \|\| SQLCODE \|\| ')');`<br>`END;` | Gestion des exceptions de la première requête. |
| `SELECT nom INTO var1 FROM Pilote`<br>`WHERE nbHVol = p_heures ;`<br>`DBMS_OUTPUT.PUT_LINE('Le pilote ayant ' \|\| p_heures \|\|`<br>`' heures est ' \|\| var1);` | Suite du traitement. |
| `EXCEPTION`<br>`WHEN NO_DATA_FOUND THEN`<br>`DBMS_OUTPUT.PUT_LINE('Pas de pilote ayant ce nombre`<br>`d''heures de vol : ' \|\| p_heures);`<br>`WHEN OTHERS THEN`<br>`DBMS_OUTPUT.PUT_LINE('Erreur d''Oracle ' \|\|`<br>`SQLERRM \|\| ' (' \|\| SQLCODE \|\| ')');`<br>`END;` | Gestion des exceptions de la deuxième requête. |

# Le mécanisme de gestion des exceptions
## Compléments

Tableau 7-23 Exceptions utilisateur

| Code PL/SQL | Commentaires |
|---|---|
| `CREATE PROCEDURE saisiePilote`<br>`(p_brevet IN VARCHAR2,p_nom IN VARCHAR2,`<br>`p_nbHVol IN NUMBER, p_comp IN VARCHAR2) IS`<br>`erreur_piloteTropJeune EXCEPTION;`<br>`erreur_piloteTropExpérimenté EXCEPTION;` | Déclaration de l'exception. |
| `BEGIN`<br>`INSERT INTO Pilote (brevet,nom,nbHVol,comp)`<br>`VALUES (p_brevet,p_nom,p_nbHVol,p_comp);`<br>`IF p_nbHVol < 200 THEN RAISE erreur_piloteTropJeune;`<br>`END IF;`<br>`IF p_nbHVol > 20000 THEN`<br>`RAISE erreur_piloteTropExpérimenté;`<br>`END IF;`<br>`COMMIT;` | Corps du traitement (validation). |
| `EXCEPTION`<br>`WHEN erreur_piloteTropJeune THEN`<br>`ROLLBACK;`<br>`DBMS_OUTPUT.PUT_LINE ('Désolé, le pilote manque`<br>`d''expérience');`<br>`WHEN erreur_piloteTropExpérimenté THEN`<br>`ROLLBACK;`<br>`DBMS_OUTPUT.PUT_LINE ('Désolé, le pilote a`<br>`trop d''expérience');`<br>`WHEN OTHERS THEN`<br>`ROLLBACK;`<br>`DBMS_OUTPUT.PUT_LINE('Erreur d''Oracle ' \|\| SQLERRM`<br>`\|\| '(' \|\| SQLCODE \|\| ')');`<br>`END;` | Gestion de l'exception.<br><br>Gestion des autres exceptions. |

Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 326)

# Le mécanisme de gestion des exceptions
## Compléments

Tableau 7-24 Utilisation du curseur implicite

| Code PL/SQL | Commentaires |
|---|---|
| `CREATE OR REPLACE PROCEDURE détruitCompagnie (p_comp IN VARCHAR2) IS`<br>`erreur_ilResteUnPilote EXCEPTION;`<br>`PRAGMA EXCEPTION_INIT(erreur_ilResteUnPilote , -2292);`<br>`erreur_compagnieInexistante EXCEPTION;` | Déclaration des exceptions. |
| `BEGIN`<br>`DELETE FROM Compagnie WHERE comp = p_comp;`<br>`IF SQL%NOTFOUND THEN`<br>`RAISE erreur_compagnieInexistante;`<br>`END IF;`<br>`COMMIT;`<br>`DBMS_OUTPUT.PUT_LINE('Compagnie '\|\|p_comp\|\| ' détruite.');` | Corps du traitement (validation). |
| `EXCEPTION`<br>`WHEN erreur_ilResteUnPilote THEN`<br>`DBMS_OUTPUT.PUT_LINE ('Désolé, il reste encore un pilote à la compagnie ' \|\| p_comp);`<br>`WHEN erreur_compagnieInexistante THEN`<br>`DBMS_OUTPUT.PUT_LINE ('La compagnie ' \|\| p_comp \|\| ' n''existe pas dans la base!');`<br>`WHEN OTHERS THEN`<br>`DBMS_OUTPUT.PUT_LINE('Erreur d''Oracle ' \|\| SQLERRM \|\| '(' \|\| SQLCODE \|\| ')');`<br>`END;` | Gestion des exceptions.<br><br>Gestion des autres exceptions. |

Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 327)

# Le mécanisme de gestion des exceptions
## Compléments

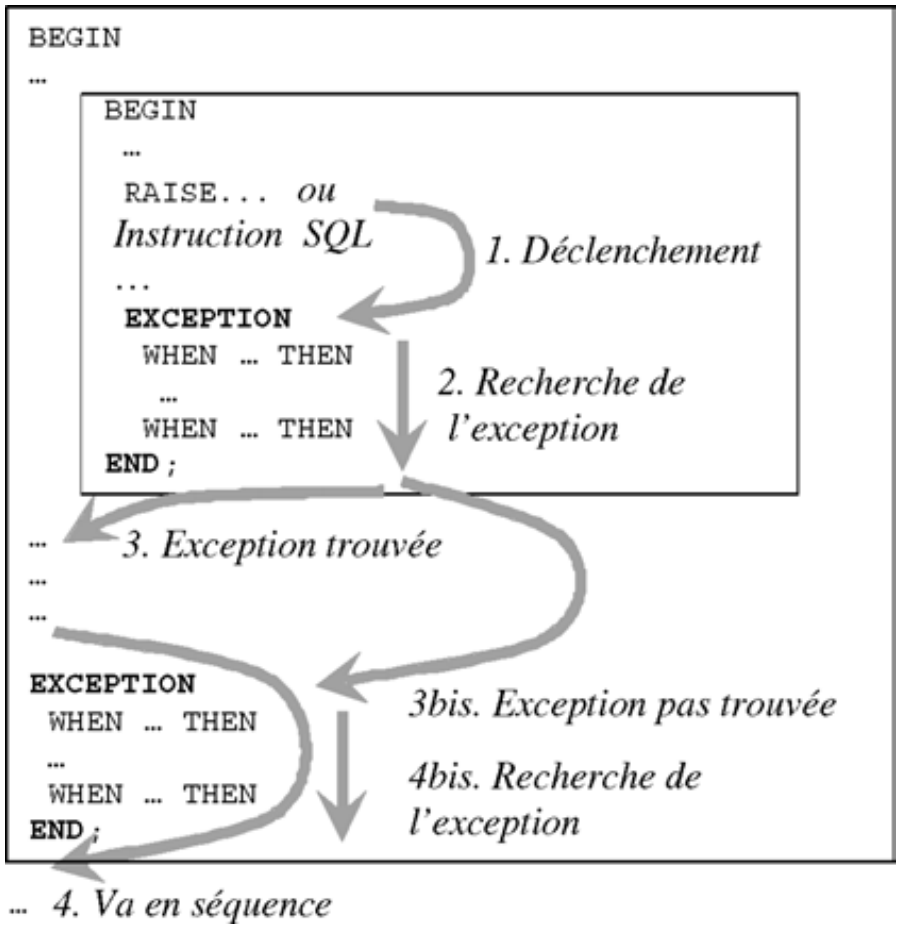
Tableau 7-25 Exception interne non prédéfinie

| Code PL/SQL | Commentaires |
|---|---|
| `CREATE PROCEDURE détruitCompagnie(p_comp IN VARCHAR2) IS`<br>`  erreur_ilResteUnPilote EXCEPTION;`<br>`  PRAGMA EXCEPTION_INIT(erreur_ilResteUnPilote , -2292);` | Déclaration de l'exception. |
| `BEGIN`<br>`  DELETE FROM Compagnie WHERE comp = p_comp;`<br>`  COMMIT;`<br>`  DBMS_OUTPUT.PUT_LINE ('Compagnie ' \|\| p_comp \|\|`<br>`  ' détruite.');` | Corps du traitement (validation). |
| `EXCEPTION`<br>`  WHEN erreur_ilResteUnPilote THEN`<br>`  DBMS_OUTPUT.PUT_LINE ('Désolé, il reste encore un`<br>`  pilote à la compagnie ' \|\| p_comp);` | Gestion de l'exception. |
| `  WHEN OTHERS THEN`<br>`  DBMS_OUTPUT.PUT_LINE('Erreur d''Oracle ' \|\| SQLERRM`<br>`  \|\| '(' \|\| SQLCODE \|\| ')');` | Gestion des autres exceptions. |
| `END;` | |

Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 330)

# Le mécanisme de gestion des exceptions
## Compléments

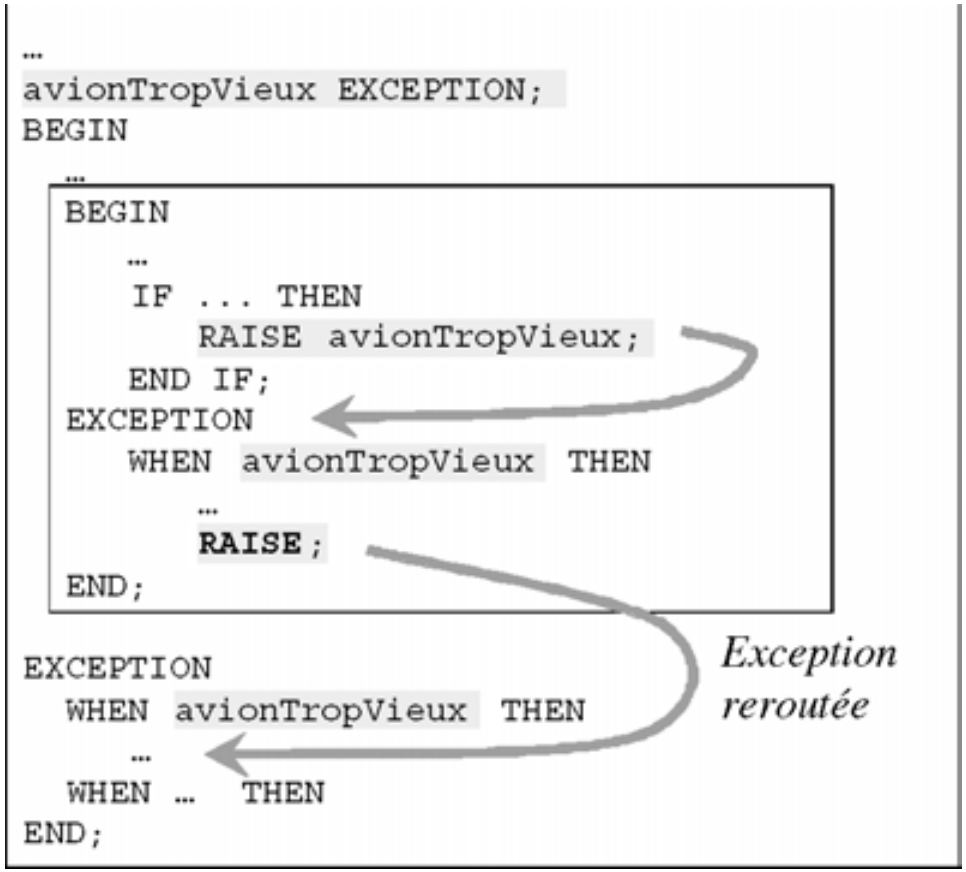
***Figure 7-8*** *Propagation des exceptions*




Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 331)

# Le mécanisme de gestion des exceptions
## Compléments

***Figure 7-9*** *Exception reroutée*




Scénario de gestion d'exceptions (Soutou & Teste, 2008, p. 332)

# Sources

Les illustrations sont reproduites à partir du document suivant :

- SQL pour Oracle – 3$^{e}$ édition, C. Soutou, Eyrolles (2008)