1/3/2023

Multivalued dependancy (4NF)

$X \twoheadrightarrow Y$ is defined on R

X → multideterminos Y

$X, Y \subset R \Rightarrow Z = R - (X \cup Y)$

$X \twoheadrightarrow Y$ if the following situation exists.

t_1, t_2, t_3, t_4 are 4 tuples in $r(R)$ such that

$t_1[x] = t_2[x] = t_3[x] = t_4[X]$

and $t_1[Y] = t_3[Y]$, $t_2[Y] = t_4[Y]$

and $t_1[Z] = t_4[Z]$, $t_2[Z] = t_3[Z]$

(tuple orders may change, but config is OK)

Let $X \simeq x_1$, $Y \Rightarrow y_1, y_2$

∴ for ~~a distinct~~ given X, there are number of distinct values of Y

⇒ tuples are to be formed for every distict value of Z corresponding to each x, y combination.

↳ we can sy, $x \twoheadrightarrow y \Rightarrow x \twoheadrightarrow z$

∴ $\boxed{x \twoheadrightarrow y | z}$

X	Y	Z
x_1	y_1	z_1
x_1	y_1	z_2
x_1	y_2	z_1
x_1	y_2	z_2

(check)

$\boxed{\text{trivial}}$ $X \twoheadrightarrow Y$ is trivial if

$Y \subseteq X$

(or) $X \cup Y = R$

→ no need to make it more 4NF

Roll $\twoheadrightarrow$ ph_no
Roll $\twoheadrightarrow$ SCODE

A non trivial $X \twoheadrightarrow Y$, if holds on a relation, then it violates 4NF

↓ $\boxed{\text{Non trivial}}$

Remove Y from R, put in new relation

↳ copy X (∴ pk = x, y)

$R \longrightarrow R_1, R_2 .. R_n$

↳ what about NULLs ??

~~DER~~ DEPT (DCODE, DNAME, ...)

STUDENT (ROLL, ..., DCODE)

→ STUDENT * DEPT

Student with DCODE NULL will not appear

Go for Left outer join

Student ⟕ DEPT
$\quad$ D.DCODE = S.~~D~~DCODE

What if I don't want to have NULL ??

↳ Loosing Info !!

↳ Student (ROLL, ...)

Student Dept (ROLL, DCODE) ✓✓ (if dcode not known, tuple doesn't exist)

Student * Student Dept may not give same as original one !!

↳ those which are missed if natural / inner join. → $\boxed{\text{dangling tuple}}$ (Reason, extreme dept)

$\boxed{\text{SQL}}$ Left outer join $\underline{R_1 ⟕ R_2}$

$A_1 A_2 A_3 \quad B_1 B_2 A_1$

SELECT ~~A1~~ R1.A1, A2, A3, B1, B2 FROM R1, R2,
WHERE R1.A1 = R2.A1 (+)

(+ on both side → full outer)

PL/SQL (specific to orach)

DML ⟶ Non procedural

APPLN ⇒ procedural aspects

↳ provides procedural aspects.

PL/SQL BLOCK

DECLARE
  {My own var & constant.

BEGIN
  ≡
END;

Executable part ← DML Statement, function

Exception : (optional)

$$\begin{tikzcd} A \arrow[r] & B \end{tikzcd}$$

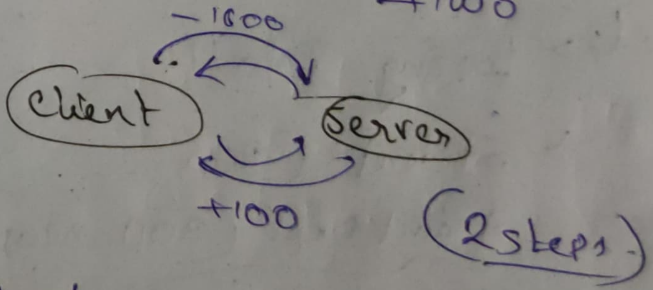
1000 ₹

```
SQL> UPDATE __ SET BAL = BAL - 1000
     WHERE AC_NO = __

SQL> _ _ _
```

2nd may not work

↳ MOD1



PL/SQL : Single ~~contae~~ shot to server.

↳ Net traffic reduce

↳ Each tuple processed individually.

→ Either complete job, or none.

STUDENT (ROLL, NAME, ...)

DECLARE
    SNAME CHAR (30);   EMP. ENAME %. TYPE
                                  DYNAMIC DTYPE DB

BEGIN
    SELECT $\overset{A1, A2}{\boxed{NAME}}$, FROM STUDENT WHERE ROLL = 1

        INTO V₁, V₂

all types allowed

(VARCHAR, VARCHA 2)

Assignment : NCCONST   NUMBER (5,2) := 3.14

    N NUMBER (2) $\bigcirc\!\!\!\!\!:=$ 5

    ↖ assg.

    ECHO SNAME

~~SN~~END

X. SQL

SQL> @ X1.

PL/SQL

6th march

```
Declare
    ===        EN CHAR(30); / EN  EMP.ENAME%.TYPE   → dynamic declaration
Begin
               SELECT ENAME INTO EN FROM EMP
Exception      WHERE ECODE = 'E001';
    ===        INSERT INTO DUMMY(——)
               (or) try ECHO statement
End            ECHO "ENAE IS" | EN      ← debugging for checking if this actually happened.
```

If i want to have complete row

```
Declare
Rec EMP%.ROWtype        Select * INTO REC
                        FROM EMP
                        where ECODE = 'E001'

        → Rec.ECODE / REC.Basic ... etc..
```

```
>>  X Number(3,0)         >> table.col%.type    }  different
    Y X%.Type.               table%.Row type    }  types of copying
                             value%.type        }  type.
```

```
DECLARE

    EN   EMP.ENAME%.type
    EC   EMP.ECODE %.type.                old 5: EC:='&EC'
                                          New 5: EC:='E001'
BASIC
    EC := '&EC'     (For user input)                ~~old 5~~
                      ↳ doesnt work
                        next time
    SELECT ENAME INTO EN
    FROM END
    WHERE ECODE = EC    ← If not found
    ;                        ↳ exception
END
```

UPDATE } → May update / delete any number of rows.
DELETE }

```
SELECT COUNT(*) INTO C
  FROM EMP
  WHERE DCODE = 'DC'
```

Aggregate fn

→ Even if no rows satisfy the condition. Still it returns something

∴ NO EXCEPTION

C NUMBER(5,0)

```
IF Condition exp THEN
    ═══
~~END IF~~ ELSE
    ═══
END IF
```

Exception Handling

Cursor → to hold number of ~~data~~ rows.

```
DECLARE
   CURSOR C1 IS
      SELECT * FROM EMP
      WHERE BASIC > B ;
BEGIN :
   B := '4 B'
   OPEN C1;
   B := B+500
FETCH C1
 INTO R
```

hold set of rows in mem

define it

set of ret rows set in cursor

query in its definition (c1) is executed.

No feats given → already opend

R c1 %Row type

```
LOOP
  ═══
END LOOP
```

Infinite loop

```
LOOP
  ═══
 IF CONDITION THEN
    EXIT
 END IF
  ═══
END LOOP
```

```
WHILE conditional expr.
 LOOP
  ═══
END LOOP
```

I NUMBER(3,0)

OR DONT DECLARE

```
FOR I IN 1... 10
         a.... b
 LOOP
  ═══
END LOOP
```

~~REVERSE(10..1)~~
REVERSE(1..10)

```
FOR I IN REVERSE 1... 10
  ═══
```

```
BEGIN
    B := AB;
LOOP  OPEN C1
    FETCH  C1  INTO R;
       ⋮
END LOOP
```

Cursor & attribute

- Explicit
  - (what we saw)
- implicit
  - SQL
    - SELECT —
    - UPDATE —
    - DELETE —
  - (SQL% ISOPEN) → False → (False) (always) closed after execution

CURSOR ATTRIBUTES

~~Customername % ISOP~~

customername % ISOPEN

% FOUND
% NOTFOUND
% ROWCOUND

Last fetch was successful

SELECT — —

SQL% FOUND [True if prev stmt executed] ~~EXIT WHEN~~

% ROWCNT → Cumulative value of no of rows fetched so far

```
WHILE C1% found
   —
   =
```

```
FOR I IN C1
LOOP
   —   I. —
   ≡   I. —
   —
END LOOP
```

• atmanirbhar loop

update

```
Cursor C1 IS
    select * FROM EMP
    WHERE ____
    FOR UPDATE OF BASIC
```

```
FOR I IN C1
LOOP
  IF ____
    UPDATE EMP SET
           BASIC = ____
    WHERE CURRENT ~~of~~ OF C1
  END IF
END LOOP
```

Exception ⇒ Built in exception.

CURSOR_ALREADY_OPEN : Trying to reopen

INVALID_CURSOR : ~~IF~~ Trying to close a cursor which is not open.

INVALID NO
VALUE_ERROR } if char to int conversion in SQL stmt only

(excepting to case for invalid_number) all type of Type conversion / constraint violation or transaction of data

~~Al'~~

NO_DATA_FOUND : Single row select statement doesnot return any row with cursor, if fetch doesn't bring any data & trying to work with row.

TOO_MANY_ROWS : IF single row select statement returns multiple rows

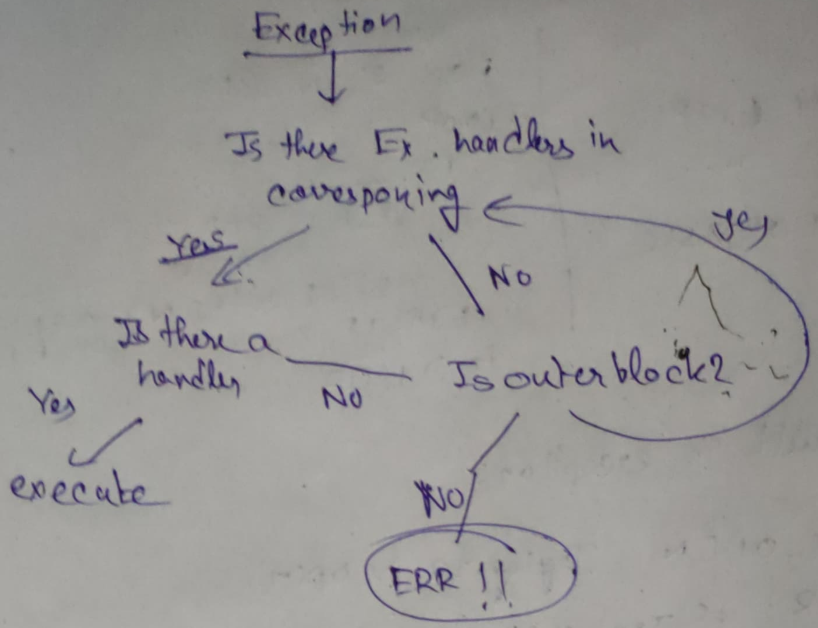
ZERO_DIVIDE :

```
EXCEPTION
   WHEN NO_DATA_FOUND THEN
        ≡
   WHEN OTHERS THEN
        =
END
```

Nesting

```
DECLARE
   ≡
BEGIN
   — DEC
     BEG
     EXP
     END
EXCEP
```

first check here then next



If exception in exception handler ⟶ check outside block

```
Declare
   myexception Exception.                    SQL.ERRCODE
Begin                                        SQL.ERRMSG
   If ____ then
      Raise myexception
   END my-exception.
```

Trigger is a subprogram which is stored as a part of ~~datase~~ database.

⟶ It is associated with an event & automatically invoked when the event occurs (can't be invoked explicitely).

Create or replace TRIGGER
⤷ optional, if exists → replaced.

Event { BEFORE | AFTER (do it before or after)
INSERT | DEL --- OF COL1 COL2 -- ON Table name.

optional ⟶ For each Row
PL/SQL

Whyne

Raise-application-error (errno, msg)

errno from -20499 to -20000

→ to cancel error

```
CREATE TRIGGER T1
 BEFORE
 UPDATE OF BALANCE ON ACCOUNT
 BEGIN
     IF :NEW.BALANCE < 0 THEN
           RAISE ( —— )
     END IF

 END
```