

BURES Maxence  
ELAIN Kevin  
ELIE Benoît  
LEROUGE Olivier  
STROZYK Mélina

## PROCÉDURES STOCKÉES

### SOMMAIRE

1) Introduction.....	2
2) Répartition.....	2
3) Mélina.....	2
3.1) Validation syntaxique.....	2
3.1.1) Procédures :.....	2
3.2) Validation en production.....	3
3.2.1) Procédure insérerProduit.....	3
3.2.2) Procédure supprimerStation.....	4
3.2.3) Procédure insérerVelo.....	5
4) Kevin.....	6
4.1) Validation syntaxique.....	6
4.1.1) Procédures.....	6
5) Benoît.....	9
5.1) Validation syntaxique.....	9
5.1.1) Fonctions.....	9
5.1.2) Procédures.....	10
5.2) Validation en production.....	14
5.2.1) ChangerEtatVelo .....	14
5.2.2) InsérerIntervention.....	14
5.2.3) ModifierDemInter.....	15
6) Maxence.....	16
6.1) Validation syntaxique.....	16
6.1.1) Fonctions.....	16
6.1.2) Procédures.....	20
6.2) Validation en production.....	20
6.2.1) Procédure ModifierDuréeInt.....	20
6.2.2) Procédure ModifierStation.....	21
6.2.3) Procédure Modifier Etat.....	22
6.2.4) Procédure Affecter Velo.....	23
7) Olivier.....	24
7.1) Validation syntaxique.....	24
7.1.1) Fonctions.....	24
7.1.2) Procédures.....	25
7.2) Validation en production.....	26
7.2.1) InsérerTechnicien : .....	26
7.2.2) ModifierVelo : .....	27
7.2.3) SupprimerTechnicien.....	27

## 1) INTRODUCTION

Ce document a pour but de consigner les implémentations de procédures réussies.

Il s'appuie sur des captures d'écran réalisés lors de la mise en place des procédures dans le SGBDR Oracle.

## 2) RÉPARTITION

insérerProduit	Mélina
supprimerStation	Mélina
insérerVelo	Mélina
changerTech	Kevin
insérerDemInterv	Kevin
ChangerEtatVelo	Benoit
InsererIntervention	Benoit
ModifierDemInter	Benoit
ModifDuréeInt	Maxence
ModifStation	Maxence
Modif Etat	Maxence
Affecter Velo	Maxence
InsererTechnicien	Olivier
ModifierVelo	Olivier
SupprimerTechnicien	Olivier

## 3) MÉLINA

### 3.1) VALIDATION SYNTAXIQUE

#### 3.1.1) PROCÉDURES :

Nom **SUPPRIMERSTATION**Schéma **GESEMP**Statut **Valid**

Source

```
(pCodeStation IN STATION.Sta_Code%TYPE)
IS
BEGIN
    IF (nbVelo(pCodeStation)=0) THEN
        DELETE FROM STATION
        WHERE Sta_Code = pCodeStation;
    ELSE
        INSERT INTO AUDITS(Aud_Code, Aud_Libelle)
        VALUES (seq_audits.NEXTVAL, 'La station contient des velos');
    END IF;
END;
```

Nom **INSERERVELO**Schéma **GESEMP**Statut **Valid**

Source

```
(pNumVelo IN VELO.Vel_Num%TYPE)
IS
    sCodeVelo integer;
BEGIN
    IF (existeVelo(pNumVelo)=false) THEN
        sCodeVelo := nouveauCodeVelo();
        INSERT INTO VELO (Vel_Num, Vel_Station, Vel_Etat, Vel_Type, Vel_Accessoire, Vel_Casse)
        VALUES (sCodeVelo, 0, 0, 0, 0, 0);
    ELSE
        INSERT INTO AUDITS(Aud_Code, Aud_Libelle)
        VALUES (seq_audits.NEXTVAL, 'Le velo existe deja');
    END IF;
    COMMIT WORK;
END;
```

## 3.2) VALIDATION EN PRODUCTION

### 3.2.1) PROCÉDURE INSERERPRODUIT

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

SELECT \* FROM PRODUIT

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

SELECT \* FROM PRODUIT;

Exécuter Charger script Enregistrer script Annuler

PDT_CODE	PDT_LIBELLE	PDT_POIDS	PDT_PXCMP	PDT_QTESTK	PDT_NBVOLS	PDT_NBCASSES
1	VTT	0	0	0	0	0

Procédure PL/SQL terminée avec succès.

### 3.2.2) PROCÉDURE SUPPRIMERSTATION

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
SELECT * FROM STATION
```

Exécuter

Charger script

Enregistrer script

Annuler

aucune ligne sélectionnée

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
execute supprimerStation(1);
```

Exécuter

Charger script

Enregistrer script

Annuler

Procédure PL/SQL terminée avec succès.

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
SELECT * FROM STATION
```

Exécuter

Charger script

Enregistrer script

Annuler

STA_CODE	STA_NOM	STA_RUE	STA_NBATTACHES	STA_NBVELOS	STA_NBATTACDISPO	STA_NBTOTLOC	STA_NBVOLS	STA_NBDEGRAD
1	Falaise							

### 3.2.3) PROCÉDURE INSERERVELO

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
SELECT * FROM VELO
```

Exécuter

Charger script

Enregistrer script

Annuler

aucune ligne sélectionnée

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
execute insererVelo(1,1,1,1);
```

Exécuter

Charger script

Enregistrer script

Annuler

Procédure PL/SQL terminée avec succès.

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
SELECT * FROM VELO
```

Exécuter

Charger script

Enregistrer script

Annuler

VEL_NUM	VEL_STATION	VEL_ETAT	VEL_TYPE	VEL_ACCESSOIRE	VEL
1	1	1	1	0	0

## 4) **KEVIN**

### 4.1) **VALIDATION SYNTAXIQUE**

#### 4.1.1) **PROCEDURES**

Nom **CHANGERTECH**  
Schéma **GESEMP**  
Statut **Valid**

```

CREATE OR REPLACE PROCEDURE changerTech
(pNomTec IN TECHNICIEN.Tec_nom%type,
pPrenomTec IN TECHNICIEN.Tec_Prenom%type,
pDemCode IN DEMANDEINTER.DemI_Num%TYPE
)
IS

    pTechCode DEMANDEINTER.DemI_Technicien%TYPE;
    pCodeaudits number(6);
    err_tecNoExiste Exception;
    err_interNoExiste Exception;
    err_nExistePas Exception;

BEGIN

    IF ((existeDemInterv(pDemCode) = true) AND (existeTechnicien(pNomTec, pPrenomTec) = true)) THEN
        select DemI_Technicien into pTechCode
        from DEMANDEINTER
        where DemI_Num = pDemCode;

        Update DEMANDEINTER
        set DemI_Technicien = pTechCode
        where DemI_Num = pDemCode;

    ELSE IF ((existeDemInterv(pDemCode) = true) AND (existeTechnicien(pNomTec, pPrenomTec) = false)) THEN
        Raise err_tecNoExiste;

    ELSE IF ((existeDemInterv(pDemCode) = false) AND (existeTechnicien(pNomTec, pPrenomTec) = true)) THEN
        Raise err_interNoExiste;

    ELSE
        Raise err_nExistePas;
    END IF;
END IF;

EXCEPTION

    WHEN err_tecNoExiste THEN
        pCodeaudits := codeaudits() + 1;
        INSERT INTO AUDITS(AUD_CODE, AUD_LIBELLE,AUD_DATE)
        VALUES(pCodeAudits,'Technicien n existe pas',sysdate);

    WHEN err_interNoExiste THEN
        pCodeaudits := codeaudits() + 1;
        INSERT INTO AUDITS(AUD_CODE, AUD_LIBELLE,AUD_DATE)
        VALUES(pCodeAudits,'Demande intervention n existe pas',sysdate);

    WHEN err_nExistePas THEN
        pCodeaudits := codeaudits() + 1;
        INSERT INTO AUDITS(AUD_CODE, AUD_LIBELLE,AUD_DATE)
        VALUES(pCodeAudits,'Technicien et demande intervention n existent pas',sysdate);

END;

```

Nom **EXISTEDEMINTE**

Schéma **GESEMP**

Statut **Valid**

Source

```
(pDemInterv IN DEMANDEINTER.Deml_Num%TYPE)
RETURN Boolean
IS
    iNb integer;
BEGIN
    SELECT COUNT(Deml_Num)
    INTO iNb
    FROM DEMANDEINTER
    WHERE Deml_Num = pDemInterv;
    RETURN(iNb = 1);
END;
```

Nom **INSERERDEMINTE**

Schéma **GESEMP**

Statut **Valid**

Source

```
(pDeml_Date IN DEMANDEINTER.Deml_Date%TYPE,
pDeml_Motif IN DEMANDEINTER.Deml_Motif%TYPE
)
IS
    sCode DEMANDEINTER.Deml_Num%TYPE;
    err_demInterExisteDeja Exception;
    pCodeaudits number(6);

BEGIN
    sCode := nouveauCodeDemInterv();
    IF (existeDemInterv(sCode) = false) THEN
        INSERT INTO DEMANDEINTER(Deml_Num, Deml_Date, Deml_Motif)
        VALUES (sCode, pDeml_Date, pDeml_Motif);
    ELSE
        Raise err_demInterExisteDeja;
    END IF;

EXCEPTION
    WHEN err_demInterExisteDeja THEN
        pCodeaudits := codeaudits() + 1;
        INSERT INTO AUDITS(AUD_CODE, AUD_LIBELLE,AUD_DATE)
        VALUES(pCodeAudits,'La demande d intervention existe deja',sysdate);

END;
```



Nom **NOUVEAUCODEDEMINTERV**Schéma **GESEMP**Statut **Valid**

Source

```
RETURN DEMANDEINTER.Deml_Num%TYPE
IS
    sCode DEMANDEINTER.Deml_Num%TYPE;

BEGIN
    select max(Deml_Num)
    into sCode
    from DEMANDEINTER;

    IF (sCode IS NULL) THEN
        sCode := 1;
    ELSE
        sCode:= TO_NUMBER(sCode);
        sCode:= sCode + 1;
        sCode:= TO_CHAR(sCode);
    END IF;

    return sCode;

END;
```

## 5) BENOÎT

### 5.1) VALIDATION SYNTAXIQUE

#### 5.1.1) FONCTIONS

```
CREATE OR REPLACE FUNCTION existeDemande1
(
    pDemi_Num IN DEMANDEINTER.Demi_Num%TYPE
)
RETURN boolean
IS
    iNb integer;
BEGIN
    SELECT COUNT(*)
    INTO iNb
    FROM DEMANDEINTER
```

Exécuter   Charger script   Enregistrer script   Annuler

Fonction créée.

```
CREATE OR REPLACE FUNCTION MaxCode_BONINTERV
RETURN number
IS
    sCodeProd BONINTERV.BI_Num%type;
BEGIN
    SELECT MAX(Pdt_Code)
    INTO sCodeProd
    FROM PRODUIT;
    IF sCodeProd IS NULL THEN
        sCodeProd := 1;
    ELSE
```

Exécuter   Charger script   Enregistrer script   Annuler

Fonction créée.

```
CREATE OR REPLACE FUNCTION existeEtat
(
    pLibelleEta IN ETAT.Eta_Libelle%TYPE
)
RETURN boolean
IS
    iNb integer;
BEGIN
    SELECT COUNT(*)
    INTO iNb
    FROM ETAT
```

Exécuter   Charger script   Enregistrer script   Annuler

Fonction créée.

### 5.1.2) PROCÉDURES

Nom **INSERERINTERVENTION**Schéma **TEST**Statut **Valid**

Source

```

(
    pTec_Matric IN TECHNICIEN.Tec_Matricule%TYPE,
    pVel_Num IN VELO.Vel_Num%TYPE,
    pSurPlace IN BONINTERV.BI_SurPlace%TYPE,
    pReparable IN BONINTERV.BI_Reparable%TYPE,
    pDatFin IN BONINTERV.BI_DatFin%TYPE,
    pDatDebut IN BONINTERV.BI_DatDebut%TYPE,
    pCpteRendu IN BONINTERV.BI_CpteRendu%TYPE,
    pDem_Num IN DEMANDEINTER.DEMI_Num%TYPE
)
IS
    iBI_Num INTEGER := 1;
BEGIN
    IF pDem_Num <> null THEN
        iBI_Num := MaxCode_BONINTERV();
        INSERT INTO BONINTERV
            (BI_NUM, BI_Velo, BI_DatDebut, BI_DatFin, BI_Technicien, BI_SurPlace, BI_Reparable, BI_CpteRendu, BI_Demande)
        VALUES
            (iBI_Num, pVel_Num, pDatDebut, pDatFin, pTec_Matric, pSurPlace, pReparable, pCpteRendu, pDem_Num);

        UPDATE DEMANDEINTER
        SET Deml_Traite = 1
        WHERE Deml_Num = pDem_Num;
    ELSE
        iBI_Num := MaxCode_BONINTERV();
        INSERT INTO BONINTERV
            (BI_NUM, BI_Velo, BI_DatDebut, BI_DatFin, BI_Technicien, BI_SurPlace, BI_Reparable, BI_CpteRendu)
        VALUES
            (iBI_Num, pVel_Num, pDatDebut, pDatFin, pTec_Matric, pSurPlace, pReparable, pCpteRendu);
    END IF;

    IF pReparable = '0' THEN
        UPDATE VELO
        SET Vel_Casse = 1
        WHERE Vel_Num = pVel_Num;
    END IF;
COMMIT;
END;

```

Nom **CHANGERETATVELO**Schéma **TEST**Statut **Valid**

Source

```
(
  pVel_Num IN VELO.Vel_Num%TYPE,
  pEta_Libelle IN ETAT.Eta_Libelle%TYPE
)
IS
  sEta_Code ETAT.Eta_Code%TYPE;

  err_etat_nofound Exception;
BEGIN
  IF existeEtat(pEta_Libelle) THEN
    SELECT Eta_Code INTO sEta_Code
    FROM ETAT
    WHERE Eta_Libelle = pEta_Libelle;

    UPDATE VELO
    SET Vel_Etat = sEta_Code
    WHERE Vel_Num = pVel_Num;
  ELSE
    RAISE err_etat_nofound;
  END IF;

  COMMIT;

EXCEPTION
  WHEN err_etat_nofound THEN
    INSERT INTO AUDITS(Aud_Code, Aud_Libelle)
    VALUES (seq_audits.NEXTVAL, 'Etat inexistant');
END;
```

...

Nom **MODIFIERDEMINTER**Schéma **TEST**Statut **Valid**

```

Source (
    pDemi_Num IN DEMANDEINTER.Demi_Num%TYPE,
    pEta_Libelle IN ETAT.Eta_Libelle%TYPE
)
IS
    sEta_Code ETAT.Eta_Code%TYPE;
    sVel_Num VELO.Vel_Num%TYPE;

    err_DemandeL_nofound Exception;
    err_Etat_nofound Exception;
BEGIN
    IF existeDemandeL(pDemi_Num) THEN
        IF existeEtat(pEta_Libelle) THEN
            SELECT Eta_Code INTO sEta_Code
            FROM ETAT
            WHERE Eta_Libelle = pEta_Libelle;

            SELECT Demi_Velo INTO sVel_Num
            FROM DEMANDEINTER
            WHERE Demi_Num = pDemi_Num;

            UPDATE VELO
            SET Vel_Etat = sEta_Code
            WHERE Vel_Num = sVel_Num;
        ELSE
            RAISE err_Etat_nofound;
        END IF;
    ELSE
        RAISE err_DemandeL_nofound;
    END IF;

    COMMIT;
EXCEPTION
    WHEN err_Etat_nofound THEN
        INSERT INTO AUDITS(Aud_Code, Aud_Libelle)
        VALUES (seq_audits.NEXTVAL, 'Etat inexistant');

    WHEN err_DemandeL_nofound THEN
        INSERT INTO AUDITS(Aud_Code, Aud_Libelle)
        VALUES (seq_audits.NEXTVAL, 'Demande intervention inexistant');
END;

```

## 5.2) VALIDATION EN PRODUCTION

### 5.2.1) CHANGERETATVELO

```
select * from Velo
```

Exécuter

Charger script

Enregistrer script

Annuler

VEL_NUM	VEL_STATION	VEL_ETAT	VEL_TYPE	VEL_ACCESSOIRE	VEL
1	1	2	1	0	0

```
execute ChangerEtatVelo ( 1, '1');
select * from velo;
```

Exécuter

Charger script

Enregistrer script

Annuler

Procédure PL/SQL terminée avec succès.

VEL_NUM	VEL_STATION	VEL_ETAT	VEL_TYPE	VEL_ACCESSOIRE	VEL
1	1	1	1	0	0

### 5.2.2) INSERERINTERVENTION

La procédure semble avoir des problèmes avec les arguments qu'on lui passe et malgré les différentes approche je n'ai pas encore réussi à la débogué.

### 5.2.3) MODIFIERDEMINTER

On reprend l'état de la base à la fin de la procédure 1), avec ETAT = 1 :

```
execute ModifierDemInter (1, 2);  
select * from velo
```

ExécuterCharger scriptEnregistrer scriptAnnuler

Procédure PL/SQL terminée avec succès.

VEL_NUM	VEL_STATION	VEL_ETAT	VEL_TYPE	VEL_ACCESSOIRE	VEL
1	1	1	1	0	0

## **6) MAXENCE**

### **6.1) VALIDATION SYNTAXIQUE**

#### **6.1.1) FONCTIONS**





Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
CREATE OR REPLACE FUNCTION existeBON  
(pCodeBon IN BONINTERV.BI_Num%TYPE)  
Return boolean  
IS
```

```
    iNb Number;  
BEGIN  
    SELECT count(BI_Num) INTO iNb  
    FROM BONINTERV  
    WHERE BI_Num = pCodeBon;
```

Exécuter

Charger script

Enregistrer script

Annuler

Fonction créée.

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
CREATE OR REPLACE FUNCTION existeEtat  
(pCodeEtat IN ETAT.Eta_Code%TYPE)  
Return boolean  
IS
```

```
    iNb Number;  
BEGIN  
    SELECT count(Eta_Code) INTO iNb  
    FROM ETAT  
    WHERE Eta_Code = pCodeEtat;
```

Exécuter

Charger script

Enregistrer script

Annuler

Fonction créée.

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
CREATE OR REPLACE FUNCTION existeStation
(pCodeSta IN STATION.Sta_Code%TYPE)
Return boolean
IS
```

```
    iNb Number;
BEGIN
    SELECT count(Sta_Code) INTO iNb
    FROM STATION
    WHERE Sta_Code = pCodeSta;
```

Exécuter

Charger script

Enregistrer script

Annuler

Fonction créée.

### Visualiser la fonction : GESEMP.CODEAUDITS

Actions Créer comme

Exécuter

Modifier

Nom **CODEAUDITS**

Schéma **GESEMP**

Statut **Valid**

Source

```
|
Return number
IS

    iNb Number;
BEGIN
    SELECT MAX(AUD_CODE) INTO iNb
    FROM AUDITS;

    Return (iNb);
END;
```

## 6.1.2) PROCÉDURES

Modifier Visualiser Supprimer Actions Créer comme Exécuter					
Sélectionner	Schéma ▲	Nom de procédure	Création	Dernière modification	Statut
<input checked="" type="radio"/>	GESEMP	<a href="#">AFFECTERVELO</a>	16 nov. 2013 01:37:42 CET	17 nov. 2013 15:19:10 CET	VALID
<input type="radio"/>	GESEMP	<a href="#">MODIFDUREEINT</a>	16 nov. 2013 01:42:56 CET	17 nov. 2013 13:40:41 CET	VALID
<input type="radio"/>	GESEMP	<a href="#">MODIFETAT</a>	16 nov. 2013 02:03:49 CET	17 nov. 2013 13:38:08 CET	VALID
<input type="radio"/>	GESEMP	<a href="#">MODIFSTATION</a>	16 nov. 2013 02:11:35 CET	17 nov. 2013 15:01:41 CET	VALID

## 6.2) VALIDATION EN PRODUCTION

### 6.2.1) PROCÉDURE MODIFDURÉEINT

```
select * from boninterv
```

Exécuter

Charger script

Enregistrer script

Annuler

BI_NUM	BI_VELO	BI_DATDE	BI_DATFI	BI_CPTERENDU	BI_	BI_DEMANDE	BI_TECHNICIEN	BI_	BI_D
1	1				I		1	I	

```
select * from boninterv
```

Exécuter

Charger script

Enregistrer script

Annuler

Exécuter

BI_NUM	BI_VELO	BI_DATDE	BI_DATFI	BI_CPTERENDU	BI_	BI_DEMANDE	BI_TECHNICIEN	BI_	BI_DUREE
1	1				I		1	I	10

```
execute modifDureeInt('1', 20)
```

Exécuter

Charger script

Enregistrer script

Annuler

Procédure PL/SQL terminée avec succès.

### 6.2.2) PROCÉDURE MODIFSTATION

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

Effacer

```
select * from station
```

Exécuter

Charger script

Enregistrer script

Annuler

STA_CODE	STA_NOM	STA_RUE	STA_NBATTACHES	STA_NBVELO	STA_NBATTACDISPO	STA_NBTOTLOC
1			30	10	27	

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
execute modifstation('1',50,50)
```

Exécuter

Charger script

Enregistrer script

Annuler

Procédure PL/SQL terminée avec succès.

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
select * from station
```

Exécuter

Charger script

Enregistrer script

Annuler

STA_CODE	STA_NOM	STA_RUE	STA_NBATTACHES	STA_NBVELOS	STA_NBATTACDISPO
1			50	10	50

### 6.2.3) PROCÉDURE MODIF ETAT

```
select * from boninterv
```

Exécuter

Charger script

Enregistrer script

Annuler

Exécuter

BI_NUM	BI_VELO	BI_DATDE	BI_DATFI	BI_CPTERENDU	BI_	BI_DEMANDE	BI_TECHNICIEN	BI_	BI_DUREE
1	1				I		1	I	10

#### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
execute modifEtat('1','E','E')
```

Exécuter

Charger script

Enregistrer script

Annuler

Procédure PL/SQL terminée avec succès.

## Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

Effacer

```
select * from boninterv
```

Exécuter

Charger script

Enregistrer script

Annuler

BI_NUM	BI_VELO	BI_DATDE	BI_DATFI	BI_CPTERENDU	BI_	BI_DEMANDE	BI_TECHNICIEN	BI_	BI_DUREE
1	1				E		1	E	20

### 6.2.4) PROCÉDURE AFFECTER VELO

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
select * from velo
```

Exécuter

Charger script

Enregistrer script

Annuler

VEL_NUM	VEL_STATION	VEL_ETAT	VEL_TYPE	VEL_ACCESSOIRE	VEL
1	1	1	1	1	0

## 7) OLIVIER

### 7.1) VALIDATION SYNTAXIQUE

#### 7.1.1) FONCTIONS

```
CREATE OR REPLACE FUNCTION existeTechnicien
(pNomTec IN TECHNICIEN.Tec_nom%type,
pPrenomTec IN TECHNICIEN.Tec_Prenom%type)
RETURN boolean
IS
    iNbTec number(2) :=0 ;
BEGIN
    SELECT count(*)
    INTO iNbTec
    FROM Technicien
```

Exécuter   Charger script   Enregistrer script   Annuler

Fonction créée.

---



## 7.1.2) PROCÉDURES

### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
CREATE OR REPLACE PROCEDURE InserTechnicien
(pNomTech in TECHNICIEN.TEC_NOM%TYPE,
pPrenomTech in TECHNICIEN.TEC_PRENOM%TYPE)
IS
    sCode TECHNICIEN.Tec_Matricule%TYPE;
    err_doublon Exception;
BEGIN
    IF (existeTechnicien(pNomTech,pPrenomTech) = FALSE ) THEN
        SELECT MAX(TO_NUMBER(Tec_Matricule))
        INTO sCode
        FROM TECHNICIEN;
    
```

Exécuter Charger script Enregistrer script Annuler

Procédure créée.



### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
CREATE OR REPLACE PROCEDURE ModifierVelo
(pVelNum IN Velo.vel_Num%TYPE,
pStationCode IN Station.Sta_Code%TYPE,
pCodeEtat IN ETAT.Eta_Code%TYPE,
pCasseVelo IN VELO.Vel_Casse%TYPE)
IS
    err_NotExiste Exception;
BEGIN
    IF (existeVelo(pVelNum)) = TRUE THEN
        UPDATE VELO
        SET Vel_Station =pStationCode,
    
```

Exécuter Charger script Enregistrer script Annuler

Procédure créée.

### Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
CREATE OR REPLACE PROCEDURE SupprimerTechnicien
(pNomTech IN TECHNICIEN.Tec_Nom%type,
pPrenomTech IN TECHNICIEN.Tec_Prenom%type)
IS
BEGIN
    IF (existeTechnicien(pNomTech,pPrenomTech) = TRUE ) THEN
        DELETE FROM Technicien
        WHERE Tec_Nom = pNomTech
        AND Tec_Prenom= pPrenomTech;
        COMMIT;
    END IF;
    
```

Exécuter Charger script Enregistrer script Annuler

Procédure créée.

Sélectionnez un type d'objet, puis entrez éventuellement un nom de schéma et un nom d'objet pour filtrer les données affichées dans l'ensemble de résultats.

Par défaut, la recherche renvoie toutes les correspondances en majuscules commençant par la chaîne saisie. Pour lancer une recherche exacte ou avec distinction maj/min, mettez la chaîne recherchée entre guillemets. Vous pouvez utiliser le caractère générique (%) dans une chaîne entre guillemets.

Créer

<div> <div>Modifier</div> <div>Visualiser</div> <div>Supprimer</div> <div>Actions</div> <div>Créer comme</div> <div>Exécuter</div> </div>					
Sélectionner	Schéma	Nom de procédure	Création	Dernière modification	Statut
<input checked="" type="radio"/>	GESEMP	INSERTECHNICIEN	18 nov. 2013 22:12:39 CET	19 nov. 2013 15:15:56 CET	VALID
<input type="radio"/>	GESEMP	MODIFIERVELO	18 nov. 2013 20:10:37 CET	19 nov. 2013 15:49:22 CET	VALID
<input type="radio"/>	GESEMP	SUPPRIMERTECHNICIEN	18 nov. 2013 20:05:14 CET	19 nov. 2013 15:25:54 CET	VALID

### 7.2.1) INSERERTECHNICIEN :

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

Exécuter    Charger script    Enregistrer script    Annuler

	TEC_MATRICULE	TEC_NOM	TEC_PRENOM
1		jean	rostand
2		Jean	Daniel
3		Jean	claudio

## Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

Exécuter    Charger script    Enregistrer script    Annuler

3)

## Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
select*
from technicien
```

Exécuter Charger script Enregistrer script Annuler

	TEC_MATRICULE	TEC_NOM	TEC_PRENOM
1	jean	rostand	
2	Jean	Daniel	
3	Jean	claude	
4	Jean	MOULE	

## 7.2.2) MODIFIERVELO :

1)

## Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

Exécuter Charger script Enregistrer script Annuler

VEL_NUM	VEL_STATION	VEL_ETAT	VEL_TYPE	VEL_ACCESSOIRE
1	1	1	1	1

2)

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
execute ModifierVelo ('1','2','2','2')
```

Exécuter Charger script Enregistrer script Annuler

Procédure PL/SQL terminée avec succès.

3)

## Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
select *
from velo
```

Exécuter Charger script Enregistrer script Annuler

VEL_NUM	VEL_STATION	VEL_ETAT	VEL_TYPE	VEL_ACCESSOIRE
1	2	2	1	1

## 7.2.3) SUPPRIMERTECHNICIEN

1)

Espace de travail

Connecté en tant que GES

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
select *
from technicien
```

Exécuter Charger script Enregistrer script Annuler

TEC_MATRICULE	TEC_NOM	TEC_PRENOM
1	jean	rostand
2	Jean	claude

2)

Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
execute SupprimerTechnicien('Jean','claude')
```

Exécuter Charger script Enregistrer script Annuler

3)

Procédure PL/SQL terminée avec succès.

Espace de travail

Saisissez les instructions SQL, PL/SQL et SQL\*Plus.

```
select *
from technicien
```

Exécuter Charger script Enregistrer script Annuler

TEC_MATRICULE	TEC_NOM	TEC_PRENOM
1	jean	rostand

*PS: Les données rentrées dans la BDD sont totalement fausse, ils servent uniquement aux test des procedures.*