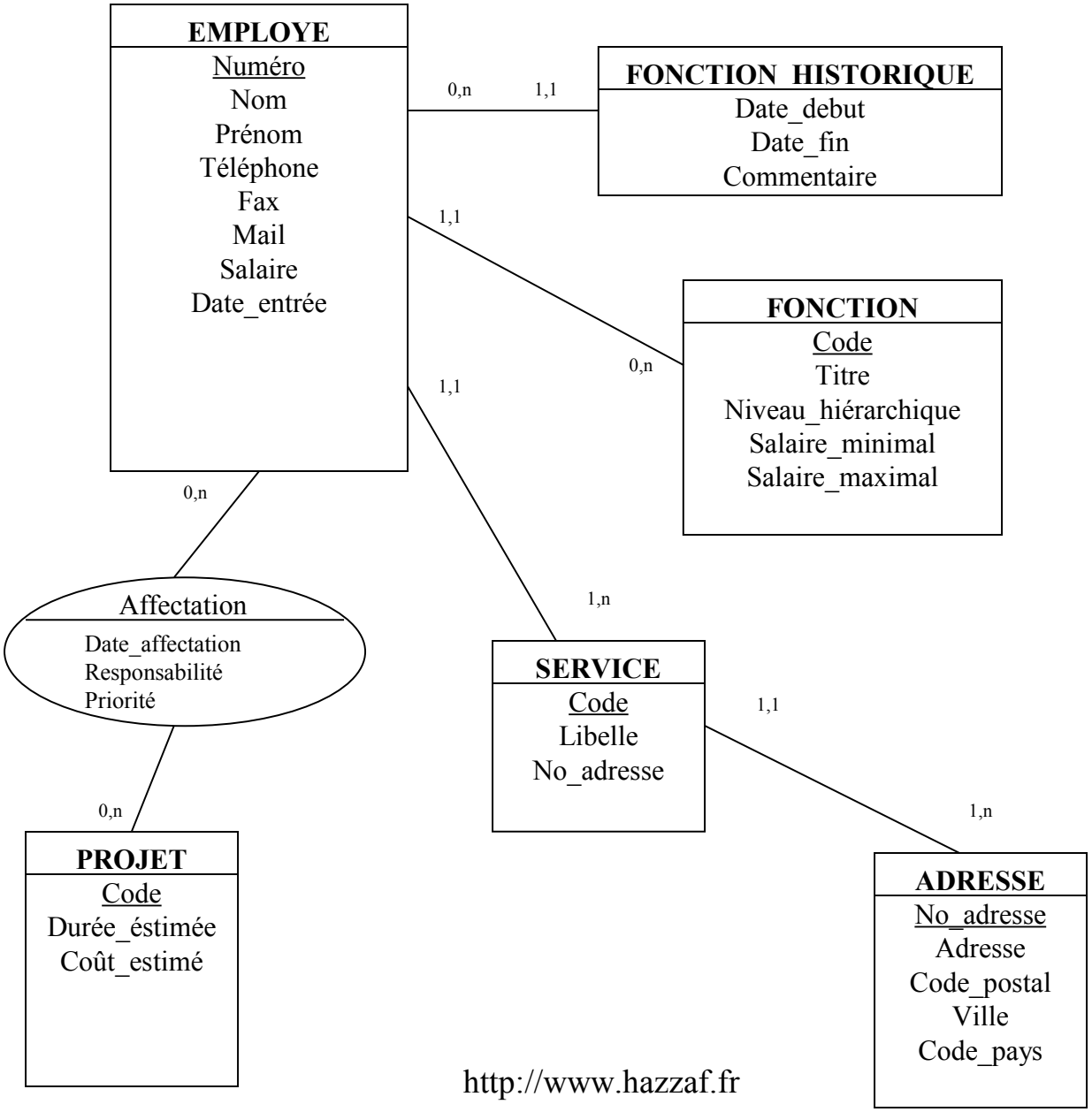
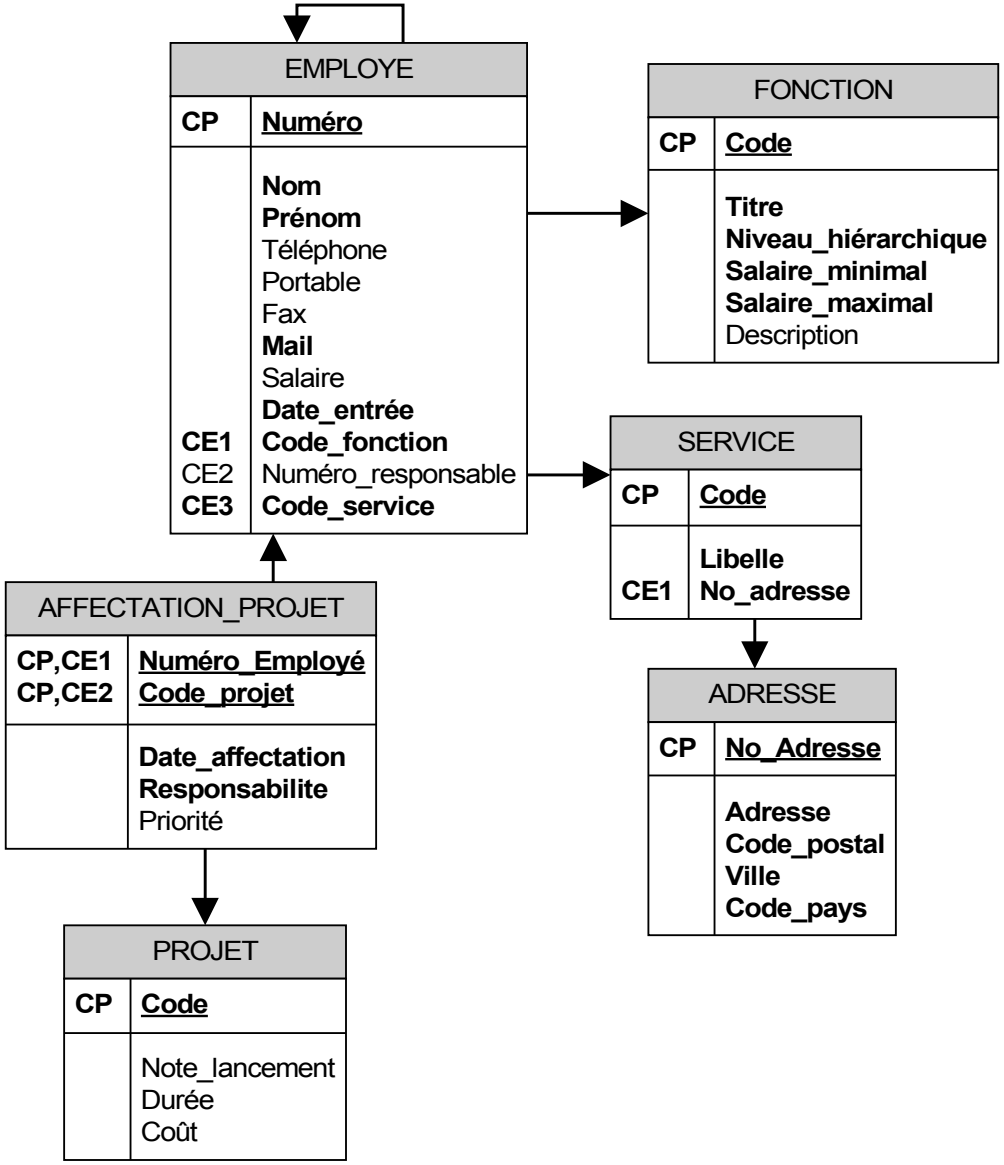
Modèle conceptuel



Modèle Relationnel



Plan

Requêtes SQL :

- Projection
- Restriction
- Conditions
- LIKE
- Traitement valeurs NULL
- Multi Conditions
- Opérateurs arithmétiques
- Utilisation des alias
- Suppression doublons
- Tri
- Opérateurs SET
- Fonctions monolignes
- Fonctions Multilignes
- Jointures
- Requêtes imbriquées
- SQL dynamique

Ordres DML :

- INSERT
- UPDATE
- DELETE

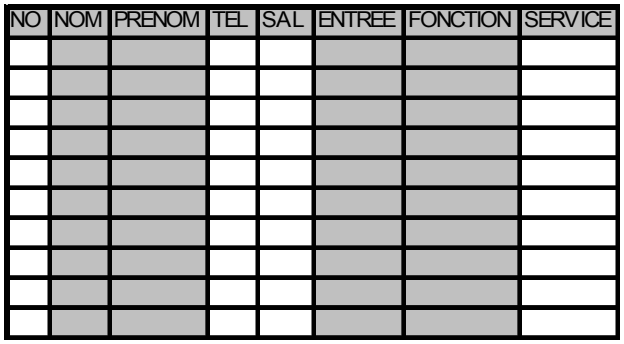
Ordres DDL :

- Table
- Contrainte
- Trigger
- Vue
- Synonyme
- Index
- Procédure
- Fonction
- Package

Environnement :

- Variables SQL+
- Commandes SQL+
- Scripts SQL
- Curseurs

Projection



SELECT NOM**,** PRENOM**,** DATE_ENTREE**,** CODE_FONCTION
FROM EMPLOYE**;**

Restriction (Sélection)

NO	NOM	PRENOM	TEL	SAL	ENTREE	FONCTION	SERVICE
						DBA	
						DBA	
						DBA	
						DBA	

SELECT *
FROM EMPLOYE
WHERE CODE_FONCTION = 'DBA';

SELECT NO, NOM, PRENOM
FROM EMPLOYE
WHERE SALAIRE >= 1500;

Condition LIKE

L'opérateur LIKE permet de comparer des chaînes de caractères. Le « % » remplace une chaîne de caractère, et « _ » remplace un caractère.

- Pour obtenir la liste des employées dont le nom commence par « F »
  SELECT NOM, PRENOM
  FROM EMPLOYE
  WHERE NOM **LIKE 'F%'**;
- Pour obtenir la liste des employées dont le nom contient « AN »
  SELECT NOM, PRENOM
  FROM EMPLOYE
  WHERE NOM **LIKE '%AN%'**;
- Pour obtenir la liste des employées dont le nom se termine par « ING »
  SELECT NOM, PRENOM
  FROM EMPLOYE
  WHERE NOM **LIKE '%ING'**;
- Pour obtenir la liste des employées dont la deuxième lettre du nom est « I »
  SELECT NOM, PRENOM
  FROM EMPLOYE
  WHERE NOM **LIKE '_I%'**;

Traitement des valeurs avec NULL

Comparaison avec NULL :

```
SELECT * FROM EMPLOYE
WHERE NO_RESPONSABLE IS NULL;
SELECT * FROM EMPLOYE
WHERE NO_RESPONSABLE IS NOT NULL;
```

Retourner une autre valeur que NULL (NVL) :

```
SELECT NVL(CODE_FONCTION, 'Non définie')
FROM EMPLOYE;
```

Multi Conditions

Opérateurs logiques :

- AND
- OR
- NOT

```
SELECT NOM, PRENOM
FROM EMPLOYE
WHERE SAL > 1500
AND CODE_FONCTION = 'DBA';
```

```
SELECT CODE
FROM PROJET
WHERE DUREE > 60
OR COUT > 500000;
```

Opérateurs arithmétiques

+ - * /

SELECT NOM, PRENOM, **SALAIRE * 1.18 AS PRIME**
FROM EMPLOYE;

AS permet de définir un alias à une colonne

Suppression des doublons

Lors d'une projection les doublons ne sont pas éliminés, C'est une différence entre l'algèbre relationnel et SQL

```
SELECT DISTINCT CODE_SERVICE
FROM EMPLOYE;
```

Tri

La clause ORDER BY permet de trier le résultat d'une requête SELECT

ASC : Tri croissant

DESC : Tri décroissant

Exemple :

SELECT * FROM EMPLOYE

ORDER BY DATE_ENTREE **DESC**, NOM **ASC**;

Dans cette requête, le résultat est trié par date d'entrée décroissante, et les lignes qui ont la même date d'entrée sont triées par nom croissant.

Si vous ne spécifier ni ASC ni DESC dans la clause ORDER BY, c'est l'ordre croissant qui est pris par défaut.

Opérateurs SET

- UNION
- UNION ALL
- INTERSECT
- MINUS

Fonctions Monolignes

Une fonction monoligne est une fonction qui s'applique enregistrement par enregistrement.

- Fonctions de chaînes de caractères
- Fonctions numériques
- Fonctions de dates
- Fonctions de conversion

Fonctions de chaînes de caractères

Une fonction de caractères prend en entrée des arguments (Nombre ou chaîne de caractères), et retourne un nombre ou une chaîne de caractères en sortie.

Quelques fonctions renvoyant une chaîne :

- **CHR** : Transforme un code ASCII en caractère
- **CONCATENATION** : L'opérateur || est utilisé pour concaténer des chaînes de caractères
- **INITCAP** : Retourne la chaîne de caractères passée en entrée, avec la première lettre en majuscule et le reste en minuscule.
- **LOWER** : Transforme toutes les lettres d'une chaîne de caractères en minuscule
- **UPPER** : Transforme toutes les lettres d'une chaîne de caractères en majuscule
- **LPAD** : LPAD(chaîne1, n, chaîne2) retourne une chaîne de caractères de longueur n, contenant chaîne1 complétée à gauche par plusieurs chaîne2 (RPAD complète à droite).
- **LTRIM** : LTRIM(chaîne1, chaîne2) enlève chaîne2 de chaîne1 si elle existe à gauche, si chaîne2 n'est pas spécifiée, cette fonction supprime les espaces à gauche (il y a aussi RTRIM, TRIM)
- **REPLACE** : REPLACE(chaîne, C1, C2) remplace dans chaîne toutes les sous chaîne C1 par C2.
- **SUBSTR** : SUBSTR(chaîne,p,l) retourne la partie de chaîne commençant à p et de longueur l.
- **REGEXP_XXXXX** : http://drisshazzaf.typepad.com/oracle/2006/06/les_expressions.html

Fonctions de chaînes de caractères (Suite)

Fonctions retournant un nombre :

- ASCII : ASCII(chaîne) retourne l'équivalent décimal du premier caractère de chaîne.
- INSTR : INSRT(chaîne1,chaîne2,n,m) recherche chaîne2 dans chaîne1 à partir de la position n, et renvoi la position du premier caractère de l'occurrence m de chaîne2 trouvée. Par défaut n=1 et m =1, INSTR(chaîne1,chaîne2) retourne la position du premier caractère de la première occurrence de chaîne2 trouvée dans chaîne1 en commençant à chercher à partir du premier caractère.
- LENGTH : Retourne la longueur d'une chaîne de caractères
- REGEXP_INSTR : http://drisshazzaf.typepad.com/oracle/2006/06/les_expressions.html

Fonctions Numériques

Les fonctions numériques prennent des nombres en entrée et retourne un nombre en sortie.

- ABS : Retourne la valeur absolue d'un nombre
- CEIL : CEIL(n) renvoi l'entier le plus proche qui est supérieur ou égale à n
- FLOOR : renvoi la partie entière d'un nombre
- TRUNC : TRUNC(n,d) Tronque n à la partie décimal d. Si d n'est pas spécifié, TRUNC retourne le même résultat que FLOOR.
- ROUND : ROUND(n,d) arrondi n à d décimales.

Il existe d'autres fonctions numériques :

ACOS,ASIN,ATAN,ATAN2,BITAND,COS,COSH,EXP,LN,LOG,MOD,NANVL,POWER,REMAINDER,SIGN,SIN,SINH,SQRT,TAN,TANH,WIDTH_BUCKET …

Fonctions de dates

Ce sont des fonctions qui manipulent des dates :

- ADD_MONTHS : Ajoute un nombre de mois à une date, ADD_MONTHS(date1, n) ajoute n mois à date1
- MONTHS_BETWEEN : Donne le nombre de mois entre deux dates, MONTHS_BETWEEN(date1,date2) retourne un décimal positif si date1 > date2, et négatif sinon.
- LAST_DAY : LAST_DAY(date1) Retourne la date du dernier jour du mois de date1

```
SQL> SELECT LAST_DAY(to_date('15-10-2006','DD-MM-YYYY')) FROM DUAL;
LAST_DAY
--------
31/10/06
```

- NEXT_DAY : LAST_DAY(date1,'JEUDI') renvoi la date du 'JEUDI' le plus proche supérieure à date1

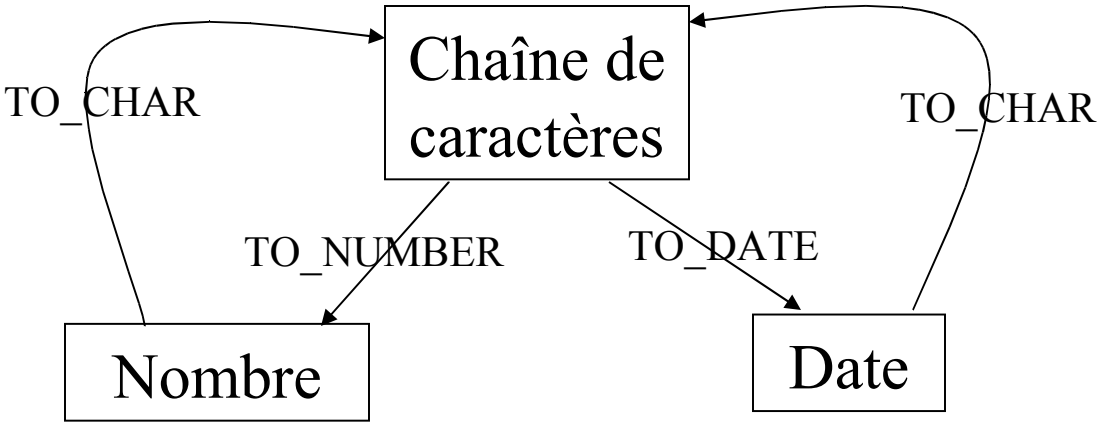
```
SQL> SELECT NEXT_DAY(to_date('01-12-2006','DD-MM-YYYY'),'JEUDI') FROM DUAL;
NEXT_DAY
--------
07/12/06
```

- SYSDATE : Retourne la date et l'heure système courante

Il existe d'autres fonctions de dates :

CURRENT_DATE, CURRENT_TIMESTAMP, DBTIMEZONE, EXTRACT, FROM_TZ, LOCALTIMESTAMP, NEW_TIME, NUMTODSINTERVAL, NUMTOYMINTERVAL, SESSIONTIMEZONE, SYS_EXTRACT_UTC, SYSTIMESTAMP, TO_TIMESTAMP, TO_TIMESTAMP_TZ, TO_DSINTERVAL, TO_YMINTERVAL,TZ_OFFSET …

Fonctions de conversion



TO_BINARY_DOUBLE et TO_BINARY_FLOAT :
http://drisshazzaf.typepad.com/oracle/2006/06/les_types_de_do.html

Il existe d'autres fonctions de conversion :

ASCIISTR,BIN_TO_NUM,CAST,CHARTOROWID,COMPOSE,CONVERT,DECOMPOSE,HEXTORAW,NUMTODSINTERVAL,NUMTOYMINTERVAL,RATOHEX,RAWTONHEX,ROWIDTOCHAR,ROWIDTONCHAR,SCN_TO_TIMESTAMP,TIMESTAMP_TO_SCN,TO_CLOB,TO_DSINTERVAL,TO_LOB,TO_MULTI_BYTE,TO_NCHAR, TO_NCHAR,TO_NCHAR, TO_NCLOB,TO_DSINTERVAL,TO_SINGLE_BYTE,TO_TIMESTAMP,TO_TIMESTAMP_TZ, TO_YMINTERVAL, TO_YMINTERVAL,TRANSLATE ...

Fonctions de groupes

Contrairement à la notion de fonction monoligne, une fonction multiligne ou fonction de groupe s'applique sur un groupe d'enregistrements.

- Clause GROUP BY : Définit le critère de groupement pour la fonction
- Clause HAVING : Permet de mettre des conditions sur les groupes d'enregistrements

- COUNT : Nombre
- SUM : Somme
- AVG : Moyenne
- MIN : Minimum
- MAX : Maximum

La liste des employés qui travaillent sur deux projets ou plus, et le nombre de projets pour chaque employé :

```
SELECT NO_EMPLOYE, COUNT(CODE_PROJET) AS NOMBRE_PROJETS
FROM AFFECTATION_PROJET
GROUP BY NO_EMPLOYE
HAVING COUNT(CODE_PROJET)  >= 2;
```

Jointures

- CROSS JOIN : Produit cartésien de deux tables

```
SELECT *
FROM EMPLOYE E
CROSS JOIN SERVICE S
```

- INNER JOIN

```
SELECT E.NO_EMPLOYE, S.LIBELLE
FROM EMPLOYE E
INNER JOIN SERVICE S
ON ( S.CODE = E.CODE_SERVICE )
```

- NATURAL JOIN

La jointure naturelle est une Equi-jointure (INNER JOIN) qui se fait par rapport aux colonnes ayant les mêmes noms entre les deux tables :

```
SELECT S.LIBELLE AS Service, A.CODE_POSTAL AS codePostal
FROM Service S NATURAL JOIN ADRESSE A;
```

OUTER JOIN

Avoir la liste des employées et leurs services, si un employé n'a pas de service, il sera affiché avec NULL à la place du service :

```
SELECT E.NO_EMPLOYE, S.LIBELLE
FROM EMPLOYE E
LEFT OUTER JOIN SERVICE S
ON ( S.CODE = E.CODE_SERVICE )
```

Avoir la liste des employées et leurs services, si un service n'a pas d'employé, il sera affiché avec NULL à la place d'employé :

```
SELECT E.NO_EMPLOYE, S.LIBELLE
FROM EMPLOYE E
RIGHT OUTER JOIN SERVICE S
ON ( S.CODE = E.CODE_SERVICE )
```

Jointure externe des deux côtés (à droite et à gauche) :

```
SELECT E.NO_EMPLOYE, S.LIBELLE
FROM EMPLOYE E
FULL OUTER JOIN SERVICE S
ON ( S.CODE = E.CODE_SERVICE )
```

Requêtes imbriquées

- Sous requêtes retournant une seule valeur
- Sous requête retournant une liste de valeurs
- Clause EXISTS

Sous requêtes retournant une valeur

Sous requête dans le SELECT :

SELECT SALAIRE – **(SELECT AVG(SALAIRE) FROM EMPLOYE) AS Résultat**

FROM EMPLOYE;

Sous requête dans le WHERE :

SELECT EMPLOYE_NO, SALAIRE

FROM EMPLOYE

WHERE SALAIRE >= **(SELECT AVG(SALAIRE) FROM EMPLOYE);**

Sous requête retournant une liste de valeur

Prédicat IN : Tous les services de PARIS

```
SELECT CODE, LIBELLE
FROM SERVICE
WHERE NO_ADRESSE IN (        SELECT NO_ADRESSE
                             FROM ADRESSE
                             WHERE VILLE = 'PARIS');
```

Prédicats ANY : Tous les Employés qui ont un salaire supérieur à celui d'un DBA

```
SELECT NOM, PRENOM
FROM EMPLOYE
WHERE SALAIRE > ANY (        SELECT SALAIRE FROM EMPLOYE
                             WHERE CODE_FONCTION = 'DBA')
```

Prédicats ALL : Tous les Employés qui ont un salaire supérieur aux salaires de tous les DBA

```
SELECT NOM, PRENOM
FROM EMPLOYE
WHERE SALAIRE > ALL ( SELECT SALAIRE FROM EMPLOYE
                             WHERE CODE_FONCTION = 'DBA')
```

EXISTS

Teste si une sous requête retourne des données ou non.

La liste de tous les employés participant à des projets :

```
SELECT E.NOM, E.PRENOM
FROM EMPLOYE E
WHERE EXISTS (SELECT NULL
              FROM AFFECTATION_PROJET A
              WHERE A.NO_EMPLOYE = E.NO_EMPLOYE);
```

La liste de tous les employés ne participant pas à des projets

```
SELECT E.NOM, E.PRENOM
FROM EMPLOYE E
WHERE NOT EXISTS (SELECT NULL
                  FROM AFFECTATION_PROJET A
                  WHERE A.NO_EMPLOYE = E.NO_EMPLOYE);
```

SQL dynamique

Le SQL dynamique sert à fabriquer une requête SQL à l'intérieur d'un programme.

Par exemple, c'est l'applicatif qui détermine avec un paramètre l'ordre dans le quel les données sont ramenées de la base de données.

Exemple :

Une procédure stockée qui ordonne la liste des employés dans un ordre defini dynamiquement.

```
CREATE OR REPLACE PROCEDURE TEST
( P_ORDRE VARCHAR2)
BEGIN
EXECUTE IMMEDIATE 'SELECT * FROM EMP ORDER BY :1' USING P_ORDRE;
END;
```

http://drisshazzaf.typepad.com/oracle/2006/04/quelques_mots_s.html

Ordres DML

- INSERT
- UPDATE
- DELETE

INSERT

```
INSERT INTO NOM_TABLE
(COLONNE_1, COLONNE_2,…COLONNE_n)
VALUES
(VALEUR_1, VALEUR_2,…VALEUR_n) ;

INSERT INTO TABLE_1
(COLONNE_11, COLONNE_12,…COLONNE_1n)
 (SELECT
COLONNE_21,COLONNE _22,…COLONNE _2n
FROM TABLE_2) ;

INSERT INTO CLIENT
(NOM,PRENOM)
(SELECT
NOM,PRENOM
FROM EMPLOYE
WHERE TYPE = 'CLIENT')
```

UPDATE

```
UPDATE TABLE_1
SET COLONNE_1 = VALEUR_1, COLONNE_2 = VALEUR_2
WHERE COLONNE_3 = VALEUR_3 ;

UPDATE EMPLOYE
SET SALAIRE = SALAIRE * 1.1
WHERE CODE_FONCTION = 'DBA' ;
```

DELETE

```
DELETE FROM TABLE_1
WHERE COLONNE_1 = VALEUR_1
OR COLONNE_2 = VALEUR_2;

DELETE FROM EMPLOYE
WHERE CODE_FONCTION IS NULL;
```

Gestion des transactions

- COMMIT
- ROLLBACK
- SAVEPOINT
- Lecture cohérente

Ordres DDL

- Table
- Contrainte
- Trigger
- Vue
- Synonyme
- Index
- Procédure
- Fonction
- Package

Création de table

```
CREATE TABLE ADRESSE
  (NO_CLIENT VARCHAR2(10) NOT NULL,
   TYPE_ADRESSE VARCHAR2(10) NOT NULL,
   ADRESSE VARCHAR2(100) NOT NULL,
   CODE_POSTAL CHAR(5) NOT NULL,
    CONSTRAINT ADRESSE_PK PRIMARY KEY
   (NO_CLIENT, TYPE_ADRESSE),
    CONSTRAINT "CLIENT_ADRESSE_FK1" FOREIGN
   KEY (NO_CLIENT) REFERENCES CLIENT
   (NO_CLIENT)) TABLESPACE CNAM;

CREATE TABLE CLIENT2
AS
(SELECT * FROM CLIENT);
```

Création de contrainte

- Supprimer une contrainte :

ALTER TABLE ADRESSE

DROP CONSTRAINT CLIENT_ADRESSE_FK1;

- Ajouter une contrainte :

ALTER TABLE ADRESSE

ADD CONSTRAINT CLIENT_ADRESSE_FK1 **FOREIGN KEY** (NO_CLIENT) **REFERENCES** CLIENT (NO_CLIENT);

- Contrainte de domaine (CHECK) :

ALTER TABLE CLIENT**ADD** (CONSTRAINT SITUATION_CK **CHECK** (CODE_SITUATION **IN** (‘OK',‘CONTENTIEUX',‘INTERDIT')) ENABLE VALIDATE) ;

Création de trigger

Un programme qui est déclencher par une action sur les données :

- Before ou After un ordre DML sur une tables
- Pour chaque ligne ou pour la table entière

```
CREATE OR REPLACE TRIGGER
   CLIENT_BEFORE_INSERT
BEFORE
INSERT ON CLIENT
FOR EACH ROW
BEGIN
   IF :NEW.CODE  IS NOT NULL THEN
         :NEW.DATE := SYSDATE;
   END IF;
END;
```

Création de vue

Un objet qu'on peut interroger pour extraire des données comme une table. Mais la vue n'est que l'exécution d'une requête. Une vue permet de :

- Simplifier l'extraction des données des tables
- Participer à l'organisation de la gestion des droits utilisateurs. En ne donnant à certains utilisateurs que l'accès aux vues.

```
CREATE OR REPLACE VIEW EMPLOYES_SERVICES
AS
SELECT  E.NOM AS Nom, E.PRENOM AS Prénom, S.LIBELLE AS Service FROM EMPLOYE E
INNER JOIN SERVICE S ON E.CODE_SERVICE = S.CODE_SERVICE;
```

Création de Synonyme

un alias d'une table ou d'une vue. Il sert à simplifier ou à donner un nom plus significatif à l'objet auquel il est attaché.

CREATE SYNONYM DETAIL_PRESTATION **FOR** CMD_LIGNE_PRESTATION;

Ou pour un synonyme public :

CREATE PUBLIC SYNONYM DETAIL_PRESTATION **FOR** CMD_LIGNE_PRESTATION;

Création d'index

Il existe principalement deux types :

- Index B*TREE
- Index BITMAP

CREATE UNIQUE INDEX ADRESSE_PK **ON ADRESSE**
(**NO_CLIENT**, **TYPE_ADRESSE**);

Création de procédure stockée

```
CREATE OR REPLACE PROCEDURE
    EMPLOYE_INFO
(P_NUMERO IN NUMBER,
P_NOM OUT VARHCAR2,
P_PRENOM OUT VARCHAR2)
AS
BEGIN
SELECT NOM, PRENOM INTO P_NOM, P_PRENOM
FROM EMPLOYE
WHERE NUMERO = P_NUMERO;
END ;
```

Création de package

Un package est un ensemble de procédures et de fonctions. Il est composé de :

- Spécification de Package, il ne contient que les signatures des procédures et des fonctions.
- Corps de Package, il contient le code des procédures et fonctions.

Spécification de package

```
CREATE OR REPLACE PACKAGE PKG_CLIENT
AS
FUNCTION CLIENT_SELECT
(P_NO_CLIENT IN VARCHAR2)
RETURN VARCHAR2;
PROCEDURE CLIENT_PYSIQUE_INSERT (
P_NO_CLIENT IN VARCHAR2,
P_TYPE_CLIENT IN VARCHAR2,
P_NOM IN VARCHAR2,
P_CODE_SITUATION IN VARCHAR2,
P_CODE_RETOUR OUT NUMBER);
END;
```

Corps de Package

```
CREATE OR REPLACE PACKAGE BODY PKG_CLIENT AS
FUNCTION CLIENT_SELECT
(P_NO_CLIENT IN VARCHAR2)
RETURN VARCHAR2 AS
W_NOM_PRENOM VARCHAR2(100);
BEGIN
SELECT NOM||' '||PRENOM INTO W_NOM_PRENOM FROM
    CLIENT WHERE NO_CLIENT = P_NO_CLIENT;
RETURN W_NOM_PRENOM;
END;
```

Corps de Package (2)

```
PROCEDURE CLIENT_PYSIQUE_INSERT (
P_NO_CLIENT IN VARCHAR2,
P_TYPE_CLIENT IN VARCHAR2,
P_NOM IN VARCHAR2,
P_CODE_SITUATION IN VARCHAR2,
P_CODE_RETOUR OUT NUMBER)
AS
BEGIN
INSERT INTO CLIENT
(NO_CLIENT,
TYPE_CLIENT,
NOM,
CODE_SITUATION)
VALUES
(P_NO_CLIENT,
'PP',
P_NOM,
P_CODE_SITUATION);
SELECT 1 INTO P_CODE_RETOUR FROM DUAL;
EXCEPTION
WHEN OTHERS THEN
BEGIN
 SELECT 0 INTO P_CODE_RETOUR FROM DUAL;
END;
END;
END;
```