Définition d'une base de données avec Oracle

LES ELEMENTS FONCTIONNELS

- Schéma: Ensemble d'objets (tables, vues, séquence, index, procédure,…) à un utilisateur et qui porte son nom.
- Table schéma de relation + relation
- Vue table non matérialisée définie par une requête SQL sur d'autres tables et/ou vues.
- Snapshot table matérialisée définie par une requête SQL sur d'autres tables et/ou vues.
- Snapshot log Table associée à la table maître d'un snapshot dans laquelle Oracle sauvegarde les mises à jour effectuées sur la table maître afin de rafraîchir le snapshot.
- Index Structure d'accès pour améliorer l'efficacité des requêtes
- Séquence Permet de définir des entiers tous différents
- Synonym Identification secondaire d'une table ou vue.
- Cluster Regroupement de tables ayant des colonnes communes.

LES ELEMENTS FONCTIONNELS (cont.)

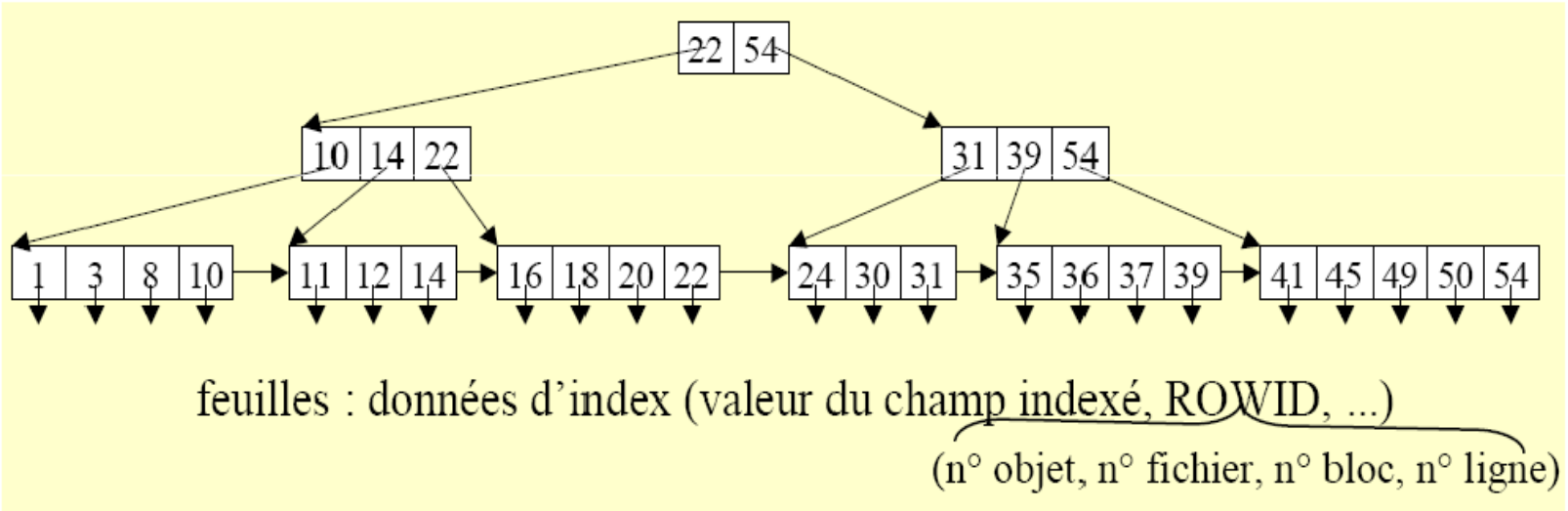
- Fonction — Ensemble nommé de commandes PL/SQL renvoyant une valeur à l'appelant.
- Procédure — Ensemble nommé de commandes PL/SQL.
- Trigger — Procédure associée à un événement de MAJ.
- Package — Collection de fonctions, procédures et objets stockés dans une même base.
- Profil — Ensemble de limitations de ressources affecté à un utilisateur pour le restreindre à ses limites.
- Rôle — Ensemble de privilèges attribuées à un utilisateur ou à d'autres rôles.
- Tablespace — Structure logique de stockage. Allocation d'espace disque

L’index B-tree

- Index par défaut sous ORACLE et la plupart des SGBD relationnels.
  - **CREATE [UNIQUE] INDEX index ON table (attributs);**

- Index intéressants pour les attributs servant fréquemment d’attribut de jointure ou de critère de recherche.
- Avantages :
  - procurent de bonnes performances pour une large gamme de requêtes avec des restrictions par égalité ou par intervalle.
  - les performances ne se dégradent pas trop lorsque la taille de la table augmente.
- Inconvénients :
  - Occupation d’espace.
  - Temps de mise à jour.

L’index B-tree



Cluster

- Ensemble de tables rangées ensemble car
  - elles partagent une ou plusieurs colonnes (clé primaire et clé étrangère)
  - elles sont souvent utilisées ensemble dans des jointures
  - Les lignes de plusieurs tables relatives à une même clé sont stockées dans le même block.

- clé de cluster
  - l'ensemble de colonnes mises en commun
  - stockées une seule fois pour toutes les tables

•Exemple :

– Déclaration du cluster et de la clé de cluster

create cluster DeptEmp (deptno **Number**) **index**;

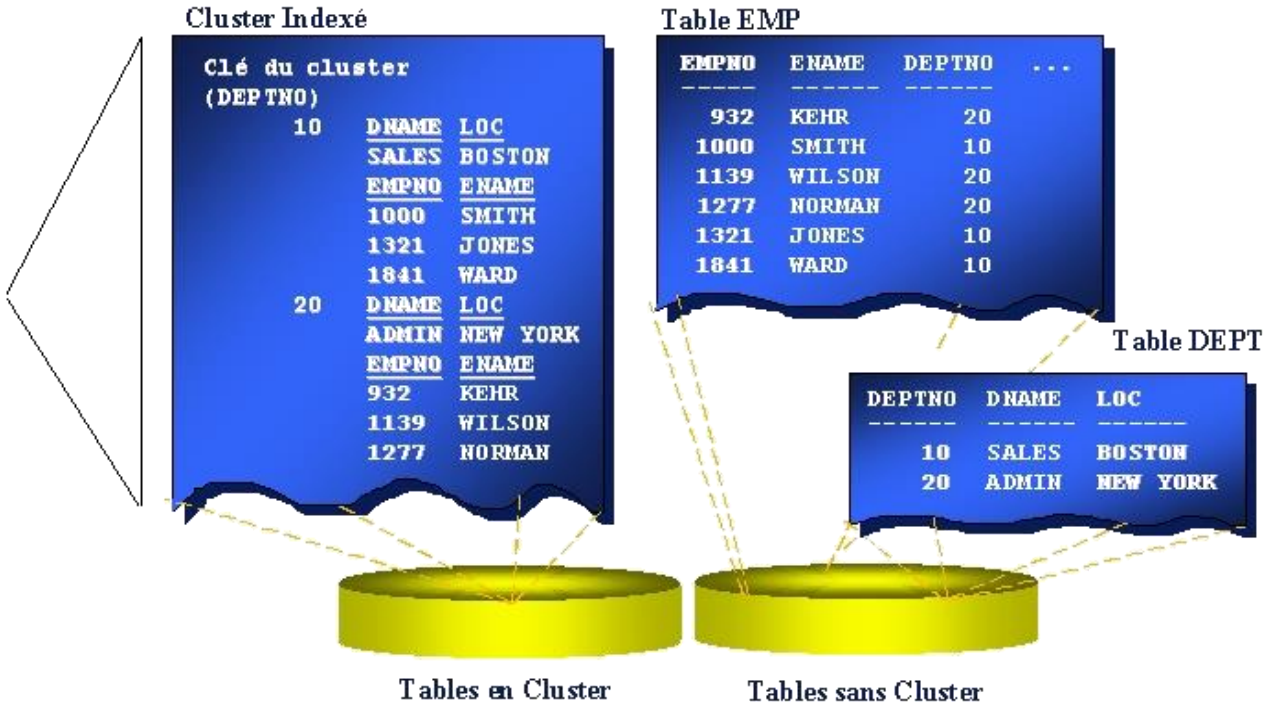
– Déclaration des tables et de leur appartenance au cluster avec attribut de cluster

create table Dept (deptno **Number Primary Key**, ...) **cluster** DeptEmp (deptno);

create table Emp (deptno **Number references** Dept(deptno), ...)

cluster DeptEmp (deptno);

Clusters Indexés



LA SEQUENCE

- Générateur de séquence

  OBJECTIF: Permettre de créer automatiquement des valeurs de clés différentes à chaque nouvelle insertion de tuple dans la relation concernée.

  La séquence sera conservée dans le dictionnaire comme tout autre objet.

- Vues du dictionnaire

  USER_SEQUENCES pour les séquences utilisateurs

  ALL_SEQUENCES pour les séquences publiques

Syntaxe:

CREATE SEQUENCE nom_séquence
[INCREMENTED BY val1]
[START WITH val2]
[MINVALUE val3 | NOMINVALUE]
[MAXVALUE val4 | NOMAXVALUE]
[CYCLE | NOCYCLE]

val2: Valeur initiale (1 par défaut)
val1: Valeur d ’incrémentation (1 par défaut)
val3 et val4: Valeurs minimale et maximale que peut prendre la séquence (1 et 10e27 par défaut)
CYCLE: Reprise de la valeur initiale dès que la valeur maximale est atteinte.

- Modification et suppression d séquence:

  ALTER SEQUENCE nom_séquence tous les paramètres sauf **START WITH**.

  DROP SEQUENCE nom_séquence

- Ex) Création :

```
Create sequence es_pilote
Start with 100
Increment by 10
Nomaxvalue
Nocycle ;
```

- Utilisation :

```
Insert into Pilote (no_pilote) VALUES
  (es_pilote.NEXTVAL) ;
```

Les relations (tables)

Table sous Oracle

Table = schéma de relation + relation

- Création/ Mise à jour/Suppression

  CREATE TABLE ma_table …;

  ALTER TABLE ma_table …;

  DROP TABLE ma_table…;
- Insertions/modifications/suppression de lignes/tuples
  - insert/update/delete
- Spécification des contraintes
  - Primary key, unique, foreign key, check …

Attribut et domaine

- Domaine = TYPES DE DONNEES
  - Les valeurs d'une colonne
  - Type choisi lors de la création de la table.
  - Spécifie l'espace de stockage
- Quelques types supportés par Oracle
  - char(n) : Un type chaîne de caractères à longueur fixe (n caractères).
  - varchar2(n) : Un type chaîne de caractères à longueur variable mais ne dépassant pas n caractères.
  - number(n,d) : Un type numérique à n chiffres et à d décimales
  - date : Un type permettant de représenter des dates. Attention la syntaxe diffère selon le pays ('jj/mm/aa' ou 'jj-mmm-aa').
  - long : Un type texte de taille maximum 2 Go.
  - raw : Un type binaire (256 octets maximum).
  - long raw : Idem mais jusqu'à 2 Go.

CARACTERISTIQUES DES TYPES DE DONNEES

- LES CHAINES DES CARACTERES
  - données alphanumériques
  - représentées entre quotes.

Exemple:

- 'Bonjour L‘ ‘ honnête homme'

Note) Des guillemets permettent de représenter une quote dans un mot.

- CHAR(n) : longueur maximale 2000 caractères

  VARCHAR2(n) : 4000 maxi. blancs en fin non stockés

Note) Le type VARCHAR est actuellement utilisé comme synonyme de VARCHAR2.

- LES NOMBRES
  - NUMBER, NUMBER(taille), NUMBER(taille,prec)

  Cf) INTEGER, FLOAT, DECIMAL(taille,prec)

  - NUMBER(38) identique à INTEGER, SMALLINT

    NUMBER identique à FLOAT
  - Ex) sal NUMBER(8,2) : une donnée de 8 chiffres dont deux après la virgule (entre –999 999, 99 et +999 999, 99)

- LES DATES
  - Manipulation non standardisée
  - DATE Par défaut : DD-MON-YYYY ex) 12-DEC-1995
  - Manipulation sous Oracle se fait via une fonction.

  Exemple:

  TO_DATE('05-02-93', 'DD-MM-YY')

* Oracle ne distingue pas au niveau des identifiants et des mots clés les majuscules des minuscules.

Quelques fonctions prédéfinies : Les fonctions numériques

Ce groupe de fonctions manipule des valeurs numériques et chacune d'elles renvoie des valeurs numériques.

- ABS(n) : retourne la valeur absolue de n.
- SIGN(n) : retourne le signe de n (-1, 0 ou 1).
- CEIL(n) : retourne l'entier par défaut.
- FLOOR(n) : idem mais avec l'entier par excès
- MOD(m,n) : calcule le reste de la division entière de m par n.
- POWER(m,n) : calcule la valeur de m élevé à la puissance n
- SQRT(n) : retourne la racine carrée de n.

Les fonctions sur chaînes de caractères

- LENGTH(ch) : retourne la longueur de la chaîne.
- UPPER(ch) : met la chaîne de caractères en majuscule.
- LOWER(ch) : met la chaîne de caractères en minuscule.
- INITCAP(ch) : met le première lettre de la chaîne en majuscule, le reste en minuscule.
- LPAD(ch,l,sch) : complémente la chaîne à gauche
  ex) LPAD('ESSAI',10,"#@") = '#@#@#ESSAI'
- RPAD(ch,l,sch) : complémente la chaîne à droite
  ex) RPAD('ESSAI',10,"_") = 'ESSAI_____'
- SUBSTR(ch,d,l) : renvoit la sous chaîne spécifiée
  ex) SUBSTR('Dominique',2,4) = 'omin'

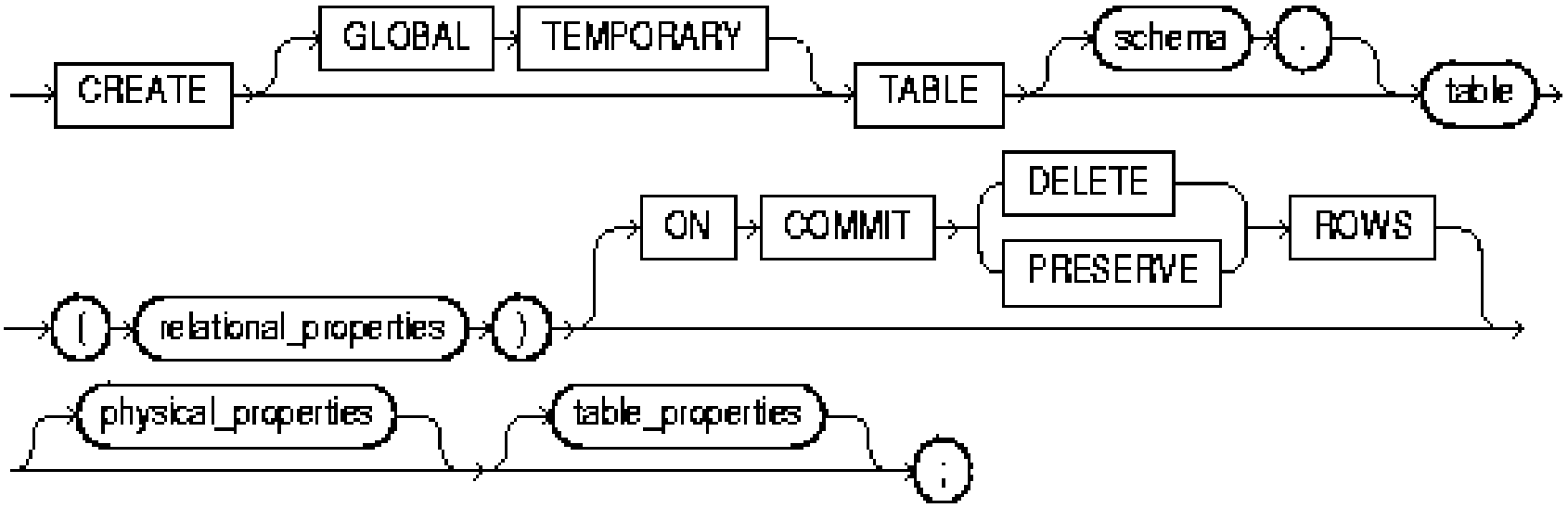
Les fonctions sur les dates

- NEXT_DAY(date,'jour') : retourne la date du prochain jour après *date*, où 'jour' est un jour de la semaine : 'Lundi', 'Mardi', etc.
- LAST_DAY(date) : retourne le dernier jour dans le mois.
- MONTHS_BETWEEN(d1,d2) : calcul le nombre de mois entre deux dates.
- ADD_MONTHS(date,n) : ajoute n mois à la date.

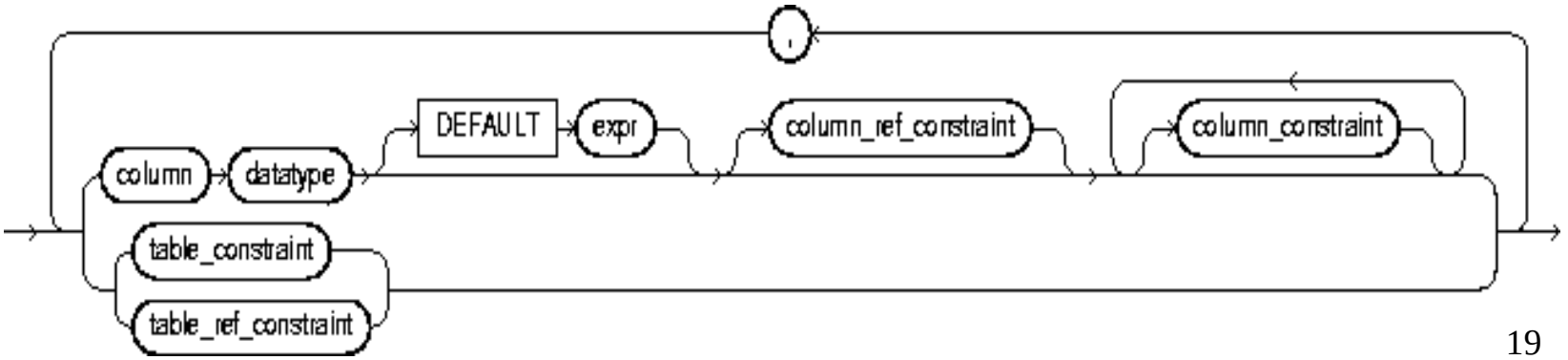
Ex) SQL> CREATE TABLE t (d date) ; DESCRIBE t ;

```
Nom de la colonne          Null?    Type
 -------------------------- -------- ----------
 D                                   DATE
SQL> INSERT INTO t VALUES (TO_DATE('15-01-04', 'DD-MM-YY'))
SQL> SELECT NEXT_DAY(d, 'Vendredi') FROM t;
NEXT_DAY( )
--------------------
16-01-04
```

Syntaxe de création d'une table (V8i)



relational_properties::=



Exemple (cont.)

```
CREATE TABLE Emp_tab (
  Empno      NUMBER(5) PRIMARY KEY,
  Ename      VARCHAR2(15) NOT NULL,
  Job        VARCHAR2(10),
  Mgr        NUMBER(5),
  Hiredate   DATE DEFAULT (sysdate),
  Sal        NUMBER(7,2),
  Comm       NUMBER(7,2),
  Deptno     NUMBER(3) NOT NULL   CONSTRAINT dept_afkey
                    REFERENCES Dept_tab(Deptno) )
  PCTFREE 10
  PCTUSED 40
  TABLESPACE users
  STORAGE (  INITIAL 50K
             NEXT 50K
             MAXEXTENTS 10
             PCTINCREASE 25 );
```

Spécifications physiques

Modification de la structure d'une table

- Ajout de colonne à une table:

  ALTER TABLE NomTable

  ADD(NomColonne[Type][DEFAULT expr] [ContrainteCol] [,…]);

  – Exemple:

  Ajouter à la table ajout la colonne secteur

  ALTER TABLE ajout

  ADD (secteur CHAR(6))
- Agrandir la taille d'une colonne:

  ALTER TABLE NomTable

  MODIFY (NomCol TYPE(Taille) [,…]);

  – Exemple:

  Agrandir la taille de la colonne lib qui passe à 20 caractères.

  ALTER TABLE ajout MODIFY(lib CHAR(20));
- Suppression de Table (DDL)

  – DROP TABLE NomTable ;

Gestion des vues

Notion de vue

- Lorsque le schéma de la base est trop complexe ou mal adaptée aux besoins d'un utilisateur
- Création de relations "virtuelles" dites **vues**
  - la vue n'est pas stockée dans la base,
  - se manipule comme une relation normale
  - peuvent être fabriquées à partir de plusieurs tables (appelées *tables de base)*
- Les opérations sur les vues affectent les tables de base de la vue
- L'utilisateur de la vue a l'impression de travailler sur une table faite pour lui alors que ce n'est qu'une fenêtre dynamique sur la base.
- Création des vues
  - Create view AS …
  - Create or Replace view AS ...
  - Create Force view as … (création de vues avec erreurs)
- Suppression des vues
  - DROP VIEW ...

Vue - Exemple

- 2 relations
  - Etudiant (N° étudiant, nom, n°projet)
  - Projet (n°projet, nom_ prof_responsable)

Create View Affectation
As Select n°étudiant, nom, nom_prof_responsable
From Etudiant, Projet
Where étudiant.n° projet = projet.n° projet;

- Interrogation à travers une vue
  - Ex) Select * from Affectation ;
  - Tout se passe comme s'il existait une table Affectation. En réalité, cette table est recomposée à chaque appel de la vue.

Vue –Mis à jours

- Il est possible d'effectuer des modifications de données par INSERT, DELETE et UPDATE à travers une vue, en tenant compte des restrictions suivante :
  - La vue doit être construite sur une seule table
  - L'ordre SELECT utilisé pour définir la vue ne doit comporter ni jointure, ni clause GROUP BY, CONNECT BY ou START WITH.
  - Les colonnes résultats de l'ordre SELECT doivent être des colonnes réelles d'une table de base et non des expressions. Ex) AVG, MAX, MIN, etc …
  - La vue contient toutes les colonnes ayant l'option NOT NULL de la table de base.

- EX) Create View vue_pilote As Select * From pilote Where adresse = 'Paris';

  UPDATE vue_pilote SET sal = sal * 1.5 ;

  => Toute les lignes de la table de base *pilote* ayant Paris sont modifiées.

Informations sur les colonnes modifiables

Vues du dictionnaire

- USER_UPDATABLE_COLUMNS
  - Shows all columns in all tables and views in the user's schema that are modifiable
- DBA_UPDATABLE_COLUMNS
  - Shows all columns in all tables and views in the DBA schema that are modifiable
- ALL_UPDATABLE_VIEWS
  - Shows all columns in all tables and views that are modifiable

Vue – Contrôle d'intégrité

- Une vue peut être utilisé pour contrôler d'intégrité des données, grâce à la clause WITH CHECK OPTION qui interdit :
  - D'insérer à travers la vue des lignes qui ne seraient pas affichées par la vue.
  - De modifier une ligne de telle sorte qu'avec les nouvelles valeurs, elle ne soit plus sélectionnée par la requête de définition de la vue.
- Ex) Lors de la modification ou de l'insertion d'un pilote dans la table pilote, on veut s'assurer qu'un pilote qui habite Paris à toujours une commission et qu'un pilote qui n'habite pas à Paris n'en a jamais.
- Create View cr_pilote As Select * From pilote Where (adresse = 'Paris' And comm IS NOT NULL) OR (adresse != 'Paris' And comm IS NULL) WITH CHECK OPTION ;

- Avion (nuavion, type, annserv, nom)
- Affectation (vol, date_vol, pilote, avion)
  - Pilote : num. du pilote conduisant l'avion pour le vol
  - Avion : num. de l'avion affecté au vol.
- Pilote (nopilot, nom, adrese, sal, comm, embauche)

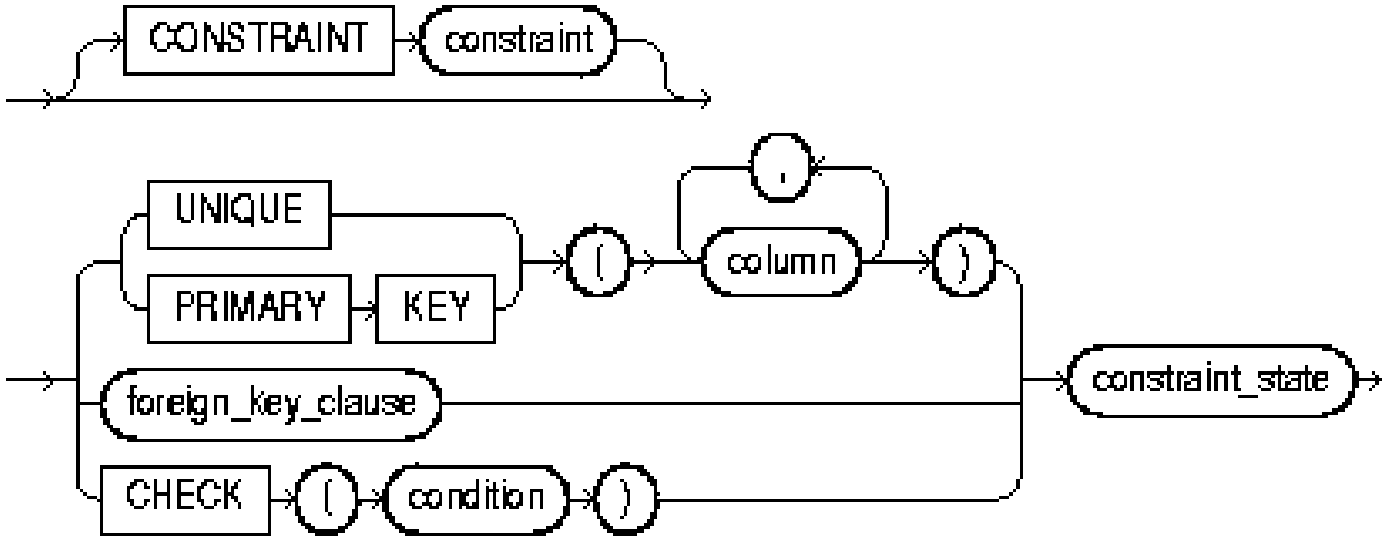
- Créer une vue qui donne la date du dernier vol réalisé par chaque avion.

- Créer une vue qui permettra de valider, en saisie et en mise à jour, qu'une commission n'est attribuée que si le pilote est affecté à un moins un vol.

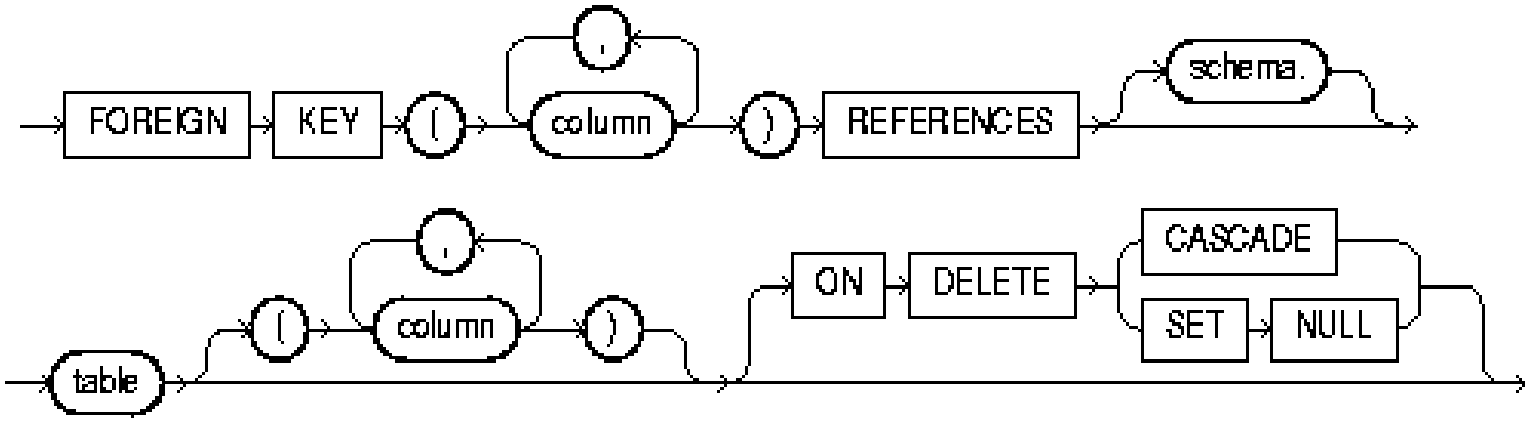
Gestion des contraintes d'intégrité

Syntaxe (V8i)

table_constraint::=



foreign_key_clause::=



Clés, Clés étrangères, Null

- Clés
  - contrainte Unique et NOT NULL ou Primary key
- Clés étrangères
  - contraintes Foreign key
- Null
  - par défaut, une colonne peut admettre des valeurs nulles.

Nommage des contraintes

- Intérêts de nommer les contraintes
  - **Activer ou désactiver certaines contraintes**

- Ajout/Modification/Suppression de contraintes existantes
  - **Ajouter une contrainte à une colonne existante**

    ALTER TABLE NomTable ADD *définition_contrainte* ;

    ex) ALTER TABLE NomTable ADD UNIQUE Colonne

- Modification et suppression : Il est possible d'activer, de déactiver ou de supprimer une contrainte par :

  ALTER TABLE NomTable

  [DROP { Primary Key | Unique (colonne [, colonnne]) | CONSTRAINT NomContrainte }]

  [ENABLE NomContrainte | DISABLE NomContrainte];

LES CONTRAINTES

Contrainte de colonne

Format:

[CONSTRAINT nom_contrainte]
[NOT] NULL
UNIQUE | PRIMARY KEY
REFERENCES [schéma.]table [(colonne)] où schéma indique le nom du créateur de la table.
[ON DELETE CASCADE]| CHECK (condition]
[USING INDEX [PCTFREE entier]
[INITRANS entier] [MAXTRANS entier]
[TABLESPACE nomts]
[STORAGE clause]]
[EXCEPTIONS INTO [schéma.]table
DISABLE UNIQUE (colonne [, colonne]…)
PRIMARY KEY
CONSTRAINT contrainte
[CASCADE]
ALL TRIGGERS

Le paramètre **STORAGE** caractérise l ’allocation d ’espace disque.

Si omis: Celui de la Tablespace par défaut.

Le mot-clé **CONSTRAINT** est facultatif et sert à donner un nom à la contrainte, nom qui sera mémorisé au dictionnaire.

Par défaut, **ORACLE** attribue un nom de la forme **SYS_Cn** avec n: Numéro unique dans la base.

DISABLE:	Désactive la contrainte, par défaut celle-ci est activée.
EXCEPTIONS INTO:	Table d 'exceptions pour copie des lignes violant les contraintes. Cette table doit être définie auparavant.
NULL:	Autorise la valeur NULL.
NOT NULL:	Interdit la valeur NULL.
PRIMARY KEY:	Interdit deux valeurs identiques dans la colonne. Un index est créé automatiquement sur la colonne et la valeur **NULL** est interdite. Cette colonne pourra être référencée dans une table dépendante par une **FOREIGN KEY**.
UNIQUE:	Interdit deux valeurs identiques dans la colonne. Un index est créé automatiquement sur la colonne. Idem primary key sauf qu'il y a acceptation de la valeur NULL.

REFERENCES: Définit une contrainte d ’intégrité référentielle par rapport à une **PRIMARY KEY** ou **UNIQUE.**

ON DELETE CASCADE: La suppression d’une ligne parent entraîne automatiquement la suppression des lignes dépendantes. Par défaut, la ligne parent ne peut être supprimée avant toutes les lignes dépendantes.

Exemple:

La suppression du département informatique dans la table TDEPT entraîne la suppression des employés affectés à ce département dans la table TEMPL.

CHECK: Condition pour insertion ou mise à jour.

Exemples:

– **Création d ’une table sans contrainte particulière:**

```
CREATE TABLE client
    (idcli      NUMBER,
    nom         CHAR(20),
    adresse     CHAR(80),
    ville       CHAR(10),
    codepost    NUMBER(5),
    adhésion    DATE
);
```

- **Création d ’une table avec idcli toujours renseigné et contrôle de domaine pour le code postal:**

```
CREATE TABLE client
(idcli      NUMBER NOT NULL,
nom         CHAR(20),
adresse     CHAR(80),
ville       CHAR(10),
codepost    NUMBER(5),
CHECK       codepost BETWEEN 10000 AND 99999),
adhésion    DATE);
```

- **Création d ’une table avec clé primaire et contrainte NULL:**

```
CREATE TABLE client
(idcli      NUMBER CONSTRAINT u_id PRIMARY KEY,
nom         CHAR(20) CONSTRAINT n_nom NOT NULL,
adresse     CHAR(80),
ville       CHAR(10),
codepost    NUMBER(5),
adhésion    DATE);
```

- **Création d ’une contrainte de référence:**

```
CREATE TABLE commande
(numcom NUMBER,
idclient NUMBER CONSTRAINT identif_contr  REFERENCES client(idcli));
```

La clause Check

```
Ex)  CREATE TABLE Climbers
  (CId INTEGER,
   CName CHAR(20),
   Skill CHAR(4),
   Age INTEGER,
   PRIMARY KEY (Cid),
   UNIQUE (CName,Age),
   CHECK (age>=10 AND age<=100));
```

- Vue USER_CONSTRAINTS
  - Ex) desc USER_CONSTRAINTS

- Pour connaître le nom de la contrainte
  - Ex) Select owner, constraint_name from USER_CONSTRAINTS Where table_name=’EMP’ and Constraint_type = ‘R’ ;
  - Sys_c0071, Sys_c0080
- Create table emp (no number primary key, a1 number references t1 (no), a2 number references t2(no)) ;

- Une fois que l’on connaît le nom, on peut examiner les colonnes associées via la vue USER_CONS_COLUMNS.
  - Ex) Select Column_name From USER_CONS_COLUMS Where Constraint_name = ‘Sys_c0071’ ;

Exception de contraintes : EXCEPTIONS

- Lorsque on active une contrainte sur des tables qui contiennent déjà des données, on peut rencontrer des violations de contraintes.

- Oracle permet d'obtenir des info. Sur les lignes qui provoquent l'échec de création de contraintes. => Option EXCEPTIONS INTO *nom_table_recup*
  - **EX) Alter table emp add constraint pk_emp primary key(num) exceptions into *nom_table_recup***
  - ***recup***
    - **Row_id : le rowid de la ligne qui a violé la contrainte**
    - **Owner : le propriétaire de la contrainte**
    - **Table_name**
    - **Constraint : le nom de la contrainte violée par la ligne**
  - $ORACLE_HOME/rdbms/admin/utlexcpt.sql

- Déterminer les lignes de la table correspondent à ces exceptions ex) Select * from emp where ROWID in (select ROW_ID from *nom_table_recup*) ;

- Créer la table recup

- Alter table emp add constraint pk_emp primary key(num) exceptions into recup ;

- Select * from emp where rowid in (select row_id from recup) ;

-Traitement des mauvaises données

- Relancer la commande alter table …

emp

rowid	num	nom	..
a12c11	1	toto	
a12c12	2	titi	
a12c13	3	..	
a12c14	1		

recup

row_id	owner	table_n	constraint
a12c11	scott	emp	pk_emp
a12c14	scott	emp	pk_emp

SQL*LOADER:

- SQL*Loader produit
  - un fichier de diagnostic(log file)
  - un fichier d'erreur (bad file)
- SQL*Loader nécessite **un fichier de contrôle**
  - les noms des fichiers de données, leur format, les formats utilisés et les tables à charger.

```
LOAD DATA INFILE *
INTO TABLE dept
fields terminated by "/"
(deptno, dname nullif dname="t", loc nullif loc="null")
BEGINDATA
12/t/null
10/accountion/cleveland
11/FINANCE/BOSTON3
```

- Exécution de la commande **sous unix (ou dos), mais non sous sqlplus** :
  - $) *sqlldr 'login'/'mot_de_passe' charger.ctl*
    - Ex) $)*sqlldr toto/toto123 charger.ctl*

Le fichier de contrôle :

- LOAD DATA INFILE *nom_fichier_données* INTO TABLE nom_de_la_table options;
  - Il précis où se trouve les données à charger grâce à la clause INFILE (si * alors les données commence dès l'apparition de BEGIN DATA sinon les données se trouvent dans le fichier spécifié dans le *nom_fichier_données*).
  - Pour les options vous avez ,
    - Fields Terminated By ',' : séparateur de champs, ici ",", par défaut, "\t" (tabulation).
    - Optionnally Enclosed By "" : délimiteur de champs

- Les commentaires commencent par // .

- Pour utiliser la valeur **`NULL,`**
  - (deptno, dname, loc nullif loc=blanks) ou
  - (deptno, dname, loc nullif loc="n")

- - Pour remplace une table existante, « REPLACE INTO TABLE » au lieu de « INTO TABLE»

Auteurs

Num	Nom	Prénom	Pays	Tél
1	Dupont	Jacques	FR	0473151585
2	Durand	Marie	GB	
3	Dupont	Pierre		
3	Dupont			

Ouvrage

Code	Titre	Prix
001	Intro aux BD	260
002	Journal de Bolivie	
003	L'homme aux sandales	

- D'abord, créer ces tables sans contraintes et les remplir (remplir la table Auteurs avec SQL*Loader, les autres tables avec la commande insert into).
- Ajouter les contraintes de clés primaires sur ces relations en récupérant les lignes qui posent problèmes dans une table PK_violation.
- Ajouter la contrainte sur la table ouvrage qui assure que les noms des auteurs sont toujours en majuscules (exceptions into).
- Supprimer la contrainte clé primaire dans la table Auteurs.

**** Ce sont des directives uniquement disponible sous le SQL d'ORACLE:**

SQL> save fic permet de sauver l'ordre courant dans le fichier fic.sql

SQL> get fic permet de récupérer le contenu de fic.sql pour alimenter le buffer de sqlplus.

SQL> start fic lance l'exécution du contenu du buffer.

SQL> l permet de lister le contenu du buffer.

SQL> ed permet d'appeler l'éditeur du système d'exploitation (vi sous UNIX). Il vous sera plus facile d'utiliser nedit.

SQL> ! est le caractère d'échappement qui permet de lancer une commande du système d'exploitation sans sortir de sqlplus.

SQL> exit permet de sortir de sqlplus.

SQL> def variable=emp permet de définir une variable ;

SQL> select * from &variable ;

SQL> undef permet d'annuler la définition de variable : undef variable.

SQL> show all permet de lister les variables de sqlplus.

SQL> set linesize 80 SQL> set pagesize 24 SQL> set pause on

SQL> set pause " PAGE SUIVANTE " permet d'arrêter le défilement entre chaque page.

SQL> spool fichier permet de copier la sortie écran dans un fichier fichier.lst. L'opération est terminée par spool off pour permettre le vidage du buffer de sortie.