Introduction aux Bases de données

Cours 8 : Création des schémas

Objectifs

Commandes SQL pour :

1. Création de schémas relationnels
   - Création des tables
   - Définition des types/domaines
2. Définition de contraintes d'intégrité

Déclinaison syntaxique différente selon systèmes

Syntaxe H2 pour ce cours

Rappel : schéma d'une BD relationnelle

- Ensemble des schémas de relation
  S = {R1, R2, …,Rn} où **Ri** schéma de relation

- Schéma d'une relation = ensemble des attributs
  avec leurs **types/domaines** respectifs
  R(A1:D1, A2:D2, …, Am:Dm) arité m

+ contraintes d'intégrité

Exemple : Contraintes

Etudiants(matricule, nom, prenom, adresse, *collaborateur**)

collaborateur fait référence à (la clé primaire de) Etudiants

Modules(code, intitule, niveau, *salle**)

contraintes d'intégrités référentielles

salle fait référence à Salles

Salles(numero, capacite, *precedente* *, *suivante**)

precedente et *suivante* font chacun référence à Salles

Syntaxe de la création de tables

create table <*nom*>

(<*Attr1*> <Dom*1*> [**not null**] [**default** <val*1*>],

<*Attr2*> <Dom*2*> [**not null**] [**default** <val*2*>],

….

<*Attrn*> <Dom*n*> [**not null**] [**default** <val*2*>],

<**contrainte*1***>,

…

<**contrainte*n***>

)

- *nom* : nom de la table
- *Atti* : nom d'un attribut
- *Domi* : type/domaine
- *Vali* : valeur par défaut
- **not null** : la valeur doit être renseignée
- contrainte*i* : contrainte d'intégrité

Types de Bases

Types alphanumériques

- *char(n)* ou *character(n)*
- *varchar(n)* ou *character varying(n)*
- n varie de 1 à 1.048.576, vaut 1 si non renseingné

Types Numériques

- Entiers *: smallint, int* (ou *integer)*, *bigint*
- Réels : *numeric(t,d)*, *real*

Types alphanumériques

char(n) vs varchar(n)

- *char(n)* réserve *n* cases dans tous les cas, allocation statique, complète par des blancs
  Ex. char(10) |C|O|U|R|S| | | | | |
- *varchar(n)* utilise **au maximum** *n* cases, allocation dynamique
  Ex. varchar(10) |C|O|U|R|S|
- Comptabiles pour la comparaison
- Egalité ignore les blancs (pas pour tous les systemes)

Types Numériques

numeric(t,d) ou decimal(t,d)

Un nombre avec virgule fixe :

- t = nombre total maximum de chiffres (sans signe)
- d = nombre de chiffres après la virgule (partie décimale)
- t-d = nombre de chiffres avant la virgule (partie entière)

Ex. numeric(5,2) peut contenir 123.45, -123.45, 0.56 ou 22, mais pas 1200 ni 12.345 (mais accepte l'insertion de son nombre arrondi 12.35)

real, **double précision**

Nombre à virgule flottante, précision de 1 à 24

Types temporels : date, time, timestamp

date

- Date calendaire (année, mois et jour du mois)

time

- Temps d'un jour (heure, minutes, secondes)
- Avec/sans fuseau horaire

timestamp

- Combinaison des types date et time
- Avec/sans fuseau horaire

Exemples

date

'01-01-2015'

time

'00:30:00'

timestamp

'01-01-2015 0:30:12.50'

Manipulation des types temporels

- Création
  - Fonctions
    - FORMATDATETIME, PARSEDATETIME
    - Arguments : 'chaîne' et 'masque'
    - Ex. PARSEDATETIME('01-02-2022', 'dd-MM-yyyy')
  - Mots-clés
    - Argument : chaine respectant un standard
    - Date, Time, TIME WITH TIME ZONE
    - Ex. date '2022-02-01'

Manipulation des types temporels

- Manipulation
  - Ajout
    - DATEADD(granularité, nombre, date)
    - ex. Ex. DATEADD(MONTH, 1, DATE '2022-02-01')
  - Différence
    - DATEDIFF(granularité, dateA, dateB)
    - Positif ou nulle si dateA avant dateB
- Comparaisons
  - <, >, >=, <=, =, !=,
  - date '2022-02-01' < date '2022-03-01' → true

Définition des domaines

- Définition d'un nouveau type à partir d'un type de base
- y associer, éventuellement, une contrainte devant être évalué à VRAI ou INCONNU pour être verifiée

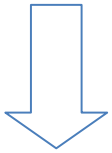
Syntaxe

create domain <*nom*> as <*dom*>[<contrainte>];

create domain fin_semaine **as** text **check**(**value** in ('sam','dim'));
create domain sal_min **as numeric(7,2) check**(**value** >=10 000);

Création d'une table en SQL : exemple

Compte(numero, titulaire, lieu, ouverture, agence)



```
create domain codeP as numeric(5);
create domain codeAg as numeric(5);
create table Compte(
    numero numeric(10) not null,
    titulaire varchar2(10) not null,
    lieu codeP default 75001,
    ouverture date,
    agence codeAg);
```

Syntaxe de la création de tables

create table <*nom*>

(<*Attr1*> <Dom*1*> [**not null**] [**default <**val*1*>],
<*Attr2*> <Dom*2*> [**not null**] [**default <**val*2*>],
….
<*Attrn*> <Dom*n*> [**not null**] [**default <**val*2*>],
<contrainte*1*>,
…
<contrainte*n*>
)

- *nom* : nom de la table
- *Atti* : nom d'un attribut
- *Domi* : type/domaine
- *Vali* : valeur par défaut
- **not null** : la valeur doit être renseignée
- Contrainte*i* : contrainte d'intégrité

Contraintes d'intégrité

Conditions devant être vérifiées par toutes les données → Cohérence des données

- Un numéro d'étudiant est unique
- Un salaire ne peut être inférieur au salaire minimum

Cohérence et mises à jour

- Toute MAJ s'applique sur une base cohérente et préserve cette cohérence

Types de contraintes d'intégrité

Contraintes de clés

- Rôle des clés : identification des n-uplets
- Chaque table possède au moins une clé (la primaire)
- Vérification : efficace avec des index

Contraintes générales

- Conditions non-exprimables avec des clés
- Conditions simples (>, <, =,..) ou complexes nécessitant des requêtes
- Vérification : potentiellement coûteuse

Contraintes de clés : rappel

Clé candidate : attributs dont les valeurs sont distinctes pour tous les n-uplets (valeurs null possibles)

Clé primaire : clé candidate dont chacun des attributs est renseigné

Clé étrangère : attributs dont les valeurs proviennent d'une clé candidate ou d'une clé primaire définie dans la même table ou dans une autre table

Contraintes de clés : syntaxe

Clés candidates

unique(*a1,..,an***)**

Clé primaire

primary key(*a1,..,an***)**

ai et *bj* sont des attributs

Clés étrangères

foreign key(*a1,..,an***) references** *table*
foreign key(*a1,..,an***) references** *table* (*b1,..,bn*)

exemple

Module(code : varchar, intitule : varchar, niveau : varchar,
suit*: varchar, salle* : number)

^ code *clé primaire;* suit ***et*** salle *référencent les tables* Module *et* Salle *respectivement.*

Salles(numero: number, nbPlaces: number)

^ numero *clé primaire*

```
create table Module(
    code varchar(10) ,
    intitule varchar(20),
    niveau char(2),
    suit varchar(10),
    salle numeric(5),
    primary key(code),
    foreign key suit references Module,
    foreign key salle references Salles
    );
```

```
create table Salles(
    numero numeric(5),
    nbPlaces numeric(3),
    primary key(numero)
    );
```

exemple

Compte(numero : number, titulaire : varchar, lieu : codeP,
ouverture : date, numAg* : codeAg)

^ numero *clé primaire;* numAg *référence la table* Agence*;* (titulaire, lieu) *clé candidate*

Agence(code :codeAg, nbEmp : number, agPlusProche* : codeAg)

^ code *clé primaire;* agPlusProche *référence la table* Agence

```
create table Compte(
    numero numeric(10) ,
    titulaire varchar2(10),
    lieu codeP,
    ouverture date,
    numAg codeAg,
    primary key(numero),
    foreign key numAg references Agence,
    unique(titulaire, lieu)
    );
```

```
create table Agence(
    code codeAg,
    nbEmp numeric(3),
    agPlusProche codeAg,
    primary key(code),
    foreign key agPlusProche
references Agence
    );
```

exemple

A(cle : number, chaine : varchar, autre : varchar)

^ cle *clé primaire,* (chaine, autre) *clé candidate*

B(cle : number, chaine1*: varchar, chaine2* : varchar)

^ cle *clé primaire,* (chaine1, chaine2) *référence la clé* (chaine, autre) de *Agence*

```
create table A(
        cle numeric(10) ,
        chaine varchar2(10),
        autre varchar2(10),
        primary key(cle),
        unique(chaine , autre )
);
```

```
create table B(
        cle numeric(10) ,
        chaine1 varchar2(10),
        chaine2 varchar2(10),
        primary key(cle),
        foreign key (chaine1, chaine2)
                        references A(chaine, autre)
);
```

Contraintes générales

Contraintes sur une seule table

- **check** <*predicat*> où *predicat* est une Condition n'utilisant qu'une seule table

Contraintes sur plusieurs tables

- **create assertion** <nom>
  check <*predicat-complexe*>
  où *predicat-complexe* est une Condition faisant appel à plusieurs tables

Contraintes sur une seule table : exemple

Empêcher qu'il y ait plus de 7 modules par niveau d'études

```
create table Module(
    code varchar(10) ,
    intitule varchar(20),
    niveau char(2),
    suit varchar(10),
    …
    constraint modules_max
        check (all (select count(*)
                    from Module
                group by niveau)<8)
     );
```

Contraintes sur plusieurs tables : exemple

```
create table Etudiant(
    eid numeric(8),
    nom varchar(10),
    moy numeric(4,2)
    …);
```

```
create table Eval(
    eid numeric(8), /*etud*/
    mid char(3), /*module*/
    note numeric(4,2) /*note*/
    …);
```

```
create assertion verif_moy
    check ( not exists (
            select * from Etudiant
            where moy <> ( select avg(note)
                           from Eval
                           where Eval.eid=Etudiant.eid ) ) );
```

L'attribut Etudiant.moy = moyenne des notes (Eval.note) de l'étudiant

Modifier le schéma d'une table

- Possible sous certaines conditions
  - Tout dépend de l'état de la base, des références, impact sur tuples existants
- Renommer, rajouter/retirer/redéfinir un attribut/une contrainte
  - ALTER TABLE <name> DROP <NAME>
  - ALTER TABLE <name> ADD <NAME><DEF>
  - ALTER TABLE <name> DROP CONSTRAINT <NAME>
  - ALTER TABLE <name> ADD CONSTRAINT<NAME><DEF>
- Désactiver les contraintes référentielles
  - ALTER TABLE <name> SET REFERENTIAL_INTEGRITY <bool>

Conditions avec NULL

Table 3–20 Conditions Containing Nulls

Condition	Value of A	Evaluation
a IS NULL	10	FALSE
a IS NOT NULL	10	TRUE
a IS NULL	NULL	TRUE
a IS NOT NULL	NULL	FALSE
a = NULL	10	UNKNOWN
a != NULL	10	UNKNOWN
a = NULL	NULL	UNKNOWN
a != NULL	NULL	UNKNOWN
a = 10	NULL	UNKNOWN
a != 10	NULL	UNKNOWN

Source doc. Oracle

Conditions avec NULL

- On associe à VRAI la valeur 1, à FAUX la valeur 0 et à INCONNU la valeur 1/2
- x AND y = min (x , y)
- x OR y = max (x , y)
- NOT x = 1 – x

Pour une requête, la condition du where doit être VRAIE pour retourner les nuplets du from

Sémantique tri-valuée : Table de vérité

X	Y	X and Y	X or Y	not x
Vrai	Vrai	Vrai	Vrai	faux
Vrai	Faux	Faux	Vrai	faux
Faux	Faux	Faux	Faux	Vrai
Vrai	Inconnu	Inconnu	Vrai	Faux
Faux	Inconnu	Faux	Inconnu	vrai
Inconnu	Inconnu	Inconnu	Inconnu	inconnu

Requêtes et valeurs nulles

1. SELECT Pname From Projet Where StartingD<01/02/15;
2. SELECT Pname From Projet Where StartingD < 01/02/15 AND Budget > 10000;
3. SELECT Pname From Projet Where StartingD < 01/02/15 OR Budget < 10000;
4. SELECT Pname From Projet Where StartingD ≤ 01/02/15 OR StartingD > 01/02/15;

Projet	Pno	Pname	City	StartingD	Budget
	1	Figue	Paris		28004
	2	Lavande	Londres	24/01/15	

Contraintes d'unicité et NULL

Exemple 1 :
create table Test(num numeric, a varchar2(10), unique(a));

Insérer : (1, "abc"), (2, "abc") (3,Null), (4, Null)

Exemple 2 :
create table Test(num numeric, a varchar2(10), b varchar2(10), unique(a,b));

Insérer :

(1,'abc', null), (2,'abc','def'), (3,null, 'def'), (4,null, 'def'), (5,'abc',null) (6,null,null),(7,null, null)

Réponses

Exemple 1 :
create table Test(num numeric, a varchar2(10), unique(a));

Insérer : (1, "abc"), (2, "abc") (3,Null), (4, Null)

num	a
1	abc
3	
4	

Exemple 2 :
create table Test(a varchar2(10), b varchar2(10), unique(a,b));

Insérer :

(1, 'abc', null), (2, 'abc','def'), (3, null, 'def'), (4, null, 'def'), (5, 'abc',null) (6, null,null),(7, null, null)

num	a	b
1	abc	
2	abc	def
3		def
6		
7		