

Définitions de contraintes

Olivier Caron

Polytech Lille
Avenue Paul Langevin Cité Scientifique Lille 1
59655 Villeneuve d'Ascq cedex

<http://ocaron.plil.fr>
Olivier.Caron@polytech-lille.fr



Les types de contraintes

- Normalisation SQL-92

Les types de contraintes

- Normalisation SQL-92
- Les contraintes de domaine définissent les valeurs prises par un attribut.

Les types de contraintes

- Normalisation SQL-92
- Les contraintes de domaine définissent les valeurs prises par un attribut.
- Les contraintes d'intégrité d'entité précisent la clé primaire de chaque table

Les types de contraintes

- Normalisation SQL-92
- Les contraintes de domaine définissent les valeurs prises par un attribut.
- Les contraintes d'intégrité d'entité précisent la clé primaire de chaque table
- Les contraintes d'intégrité référentielle assurent la cohérence entre les clés primaires et les clés étrangères

Les types de contraintes

- Normalisation SQL-92
- Les contraintes de domaine définissent les valeurs prises par un attribut.
- Les contraintes d'intégrité d'entité précisent la clé primaire de chaque table
- Les contraintes d'intégrité référentielle assurent la cohérence entre les clés primaires et les clés étrangères
- Les assertions spécifient des contraintes plus générales entre attributs quelconques.

Contrainte de domaine : NOT NULL

```
CREATE TABLE personnel (nom TEXT NOT NULL, prenom TEXT)
```

```
INSERT INTO personnel(nom) VALUES ( 'dupont ')
```

```
INSERT INTO personnel(prenom) VALUES( 'henri ' ) → ERREUR
```

Contrainte de domaine : DEFAULT

```
CREATE TABLE article (num INT NOT NULL,  
                       quantite INT DEFAULT 1,  
                       date_creation DATE DEFAULT now()  
                       )
```

- La clause NOT NULL est implicite (pour DEFAULT)
(sauf si DEFAULT NULL !)



Contrainte de domaine : UNIQUE

- Eviter les redondances , utile pour les clés :

```
CREATE TABLE article (num INT NOT NULL UNIQUE,  
                        nom TEXT ...
```

- La clé peut être constituée de plusieurs attributs :

```
CREATE TABLE reserve_par ( num_client INT NOT NULL,  
                           num_livre INT NOT NULL,  
                           UNIQUE ( num_client , num_livre )  
                           )
```

Contrainte de domaine : CHECK (1/2)

- But : spécifier une contrainte qui doit être vérifiée à tout moment par les tuples de la table :

```
CREATE TABLE personnel (  
  num INT NOT NULL UNIQUE,  
  age INT CHECK (age >= 18),  
  sexe CHAR DEFAULT 'F' CHECK (sexe IN ('M', 'F')),  
  ageFuturePromotion INT CHECK (ageFuturePromotion > age)  
)
```

Contrainte de domaine : CHECK (2/2)

- La clause CHECK peut se placer après la définition de tous les attributs.
 - Il est préférable de nommer la contrainte (facultatif)
- Utilisation de sous-requêtes SQL

CONSTRAINT moy_age

CHECK ((**select avg**(age) **from** personnel) > 35))

Déclaration d'un domaine

- Plusieurs attributs ont le même type et les mêmes contraintes
- Syntaxe :

```
CREATE DOMAIN nom [AS] type_donnees  
    [DEFAULT expression ] [ contrainte [ ... ] ]
```

- Exemple :

```
CREATE DOMAIN entier_positif  
    INT DEFAULT 0 CHECK (VALUE >=0)
```

```
CREATE TABLE personnel(  
    num INT NOT NULL UNIQUE,  
    age entier_positif  
    ...
```

Les contraintes d'intégrité d'entité

- permet de spécifier la clé primaire
- analogue à NOT NULL UNIQUE
- génère un index
- peut être spécifié à part lorsque la clé est constituée de plusieurs attributs (idem clause UNIQUE)

```
CREATE TABLE personnel (num INT PRIMARY KEY  
    . . .
```

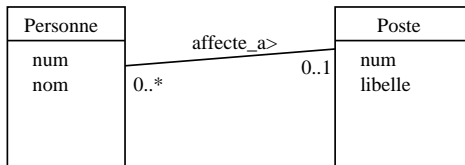
Les contraintes d'intégrité référentielle

- Définies dans la norme SQL-92

Les contraintes d'intégrité référentielle

- Définies dans la norme SQL-92
- Permettent d'assurer la cohérence des associations issues de la conception.

CIR : une cardinalité "0..1" (1/3)



CIR : une cardinalité "0..1" (2/3)

- Réalisation des tables :

```
CREATE TABLE poste (  
  num INT PRIMARY KEY,  
  libelle TEXT NOT NULL UNIQUE )
```

```
CREATE TABLE personne (  
  num INT PRIMARY KEY,  
  nom TEXT NOT NULL,  
  num_poste INT REFERENCES poste)
```

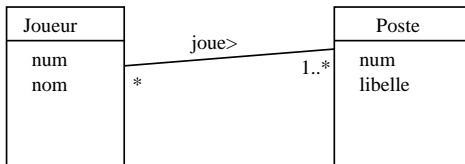
CIR : une cardinalité "0..1" (3/3)

- table poste :
- exécution :

num	libelle
1	'directeur'
2	'ingenieur'
3	'agent'

INSERT INTO personne **VALUES** (1, 'dupont', 1) → OK
INSERT INTO personne **VALUES** (2, 'durant', 4) → ERREUR
DELETE FROM poste **WHERE** num=1 → ERREUR
UPDATE personne **SET** num_poste=NULL **WHERE** num=1 → OK
DELETE FROM poste **WHERE** num=1 → OK

CIR : une cardinalité "*" (1/3)



CIR : une cardinalité "*" (2/3)

- Réalisation des tables :

```
CREATE TABLE poste (  
    num INT PRIMARY KEY, libelle TEXT NOT NULL UNIQUE )
```

```
CREATE TABLE joueur (  
    num INT PRIMARY KEY, nom TEXT NOT NULL)
```

```
CREATE TABLE joue (  
    num_joueur INT NOT NULL REFERENCES joueur ,  
    num_poste INT NOT NULL REFERENCES poste ,  
    PRIMARY KEY (num_joueur , num_poste))
```

CIR : une cardinalité "*" (3/3)

- Les tables joueur et poste :

joueur		poste	
num	nom	num	libelle
1	'blanc'	1	'goal'
2	'barthez'	2	'defenseur'
		3	'milieu'
		4	'attaquant'

- Exécution (en rouge : erreur) :

```
INSERT INTO joue VALUES (1,2)
```

```
INSERT INTO joue VALUES (1,3)
```

```
INSERT INTO joue VALUES (2,1)
```

```
INSERT INTO joue VALUES (2,1)
```

```
INSERT INTO joue VALUES (2,5)
```

```
DELETE FROM poste where num=3
```

```
DELETE FROM poste where num=4
```

Contraintes et clés étrangères

- Plusieurs modes possibles.

Contraintes et clés étrangères

- Plusieurs modes possibles.
- Objectifs communs : préserver la cohérence de la base

Contraintes et clés étrangères

- Plusieurs modes possibles.
- Objectifs communs : préserver la cohérence de la base
- Mode par défaut :
Refuser l'opération si contrainte non respectée

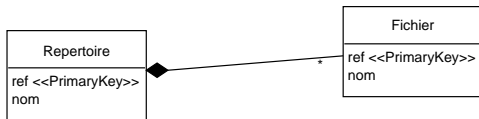
Contraintes et clés étrangères

- Plusieurs modes possibles.
- Objectifs communs : préserver la cohérence de la base
- Mode par défaut :
Refuser l'opération si contrainte non respectée
- Autre mode :
Accepter l'opération et modification en cascade pour préserver la cohérence.

Contraintes et clés étrangères

- Plusieurs modes possibles.
- Objectifs communs : préserver la cohérence de la base
- Mode par défaut :
Refuser l'opération si contrainte non respectée
- Autre mode :
Accepter l'opération et modification en cascade pour préserver la cohérence.
- Définition des modes lors de la définition de contraintes.

Contraintes sur clés étrangères : un exemple



- Valeurs initiales des tables :

Table repertoire		Table fichier		
ref	nom	ref	nom	ref_rep
0	'/'	1	'fic1'	1
1	'/rep1'	2	'fic2'	2
2	'/rep2'	3	'fic3'	1

Contraintes sur clé étrangère (1/5)

- Mode par défaut
- Le plus restrictif : refus de l'opération
- Exemple :

```
CREATE TABLE fichier (  
  ref INTEGER PRIMARY KEY, nom TEXT NOT NULL,  
  ref_rep INTEGER REFERENCES repertoire  
  ON DELETE RESTRICT  
)  
;  
...  
DELETE FROM repertoire where ref=1 ;
```

- Résultat : la suppression est refusée

Contraintes sur clé étrangère (2/5)

- Le mode permissif
- Exemple :

```
CREATE TABLE fichier (  
  ref INTEGER PRIMARY KEY, nom TEXT NOT NULL,  
  ref_rep INTEGER REFERENCES repertoire  
  ON DELETE CASCADE
```

```
) ;
```

```
...
```

```
DELETE FROM repertoire WHERE ref=1 ;
```

- Résultat : '/rep1' supprimé, 'fic1' et 'fic3' également !

Contraintes sur clé étrangère (3/5)

- Valeurs par défaut
- Exemple :

```
CREATE TABLE fichier (  
    ref INTEGER PRIMARY KEY, nom TEXT NOT NULL,  
    ref_rep INTEGER DEFAULT 0 REFERENCES repertoire  
        ON DELETE SET DEFAULT  
);  
DELETE FROM repertoire WHERE ref=1 ;
```

- Résultat : supprime '/rep1', 'fic1' et 'fic3' sont désormais dans '/'
- Si la valeur par défaut est incohérente, la suppression est refusée

Contraintes sur clé étrangère (4/5)

- Clause SET NULL
- Exemple :

```
CREATE TABLE fichier (  
  ref INTEGER PRIMARY KEY, nom TEXT NOT NULL,  
  ref_rep INTEGER REFERENCES repertoire  
    ON DELETE SET NULL  
)  
;  
...  
DELETE FROM repertoire WHERE ref=1 ;
```

- Résultat : supprime '/rep1', 'fic1' et 'fic3' ne sont pas affecté à un répertoire

Contraintes sur clé étrangère (5/5)

- Les contraintes sont également valables pour la mise à jour
- Syntaxe identique
- Exemple :

```
CREATE TABLE fichier (  
  ref INTEGER PRIMARY KEY, nom TEXT NOT NULL,  
  ref_rep INTEGER REFERENCES repertoire  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
)  
;
```

...

Conclusion

- Mécanisme de contraintes très développé
- Largement utilisé depuis SQL-92
- Syntaxe classique (contraintes nommées) :

CONSTRAINT nom [**UNIQUE** | **NOT NULL** | ...]

- Mise à jour de contraintes via `ALTER TABLE`