Bases de données
Contraintes et séquences

Silviu Maniu

2022/2023

Polytech App3

Contenu

Contraintes d'intégrité

Situation :

- un employé introduit des données corrompues (prix négatif, horaire impossible)

Integrité :

- vérifier l'integrité des données avant leur stockage (valeurs permises et par rapport aux autres données)

Contraintes – définitions

Contrainte : une propriété qui doit être vérifiée par la base de données (*intégrité des données*)

Langage de contraintes : sous-ensemble SQL pour définir les contraintes

Contraintes – outils SGBD

Langage de contraintes : sous-ensemble SQL pour définir l'intégrité des données

Comportement SGBD :

1. détection automatique des mises à jour qui ne respectent pas les contraintes déclarées (violation de contrainte)
2. refus de la mise à jour

Langage de contraintes

- obligation de valeur dans une colonne `not null`
- unicité d'une valeur dans une colonne `unique`
- types énumérés
- clé primaire `primary key` (= `unique`+`not null`)
- inclusion entre valeurs de deux colonnes `references`
- clé étrangère (intégrité référentielle) : contrainte d'inclusion sur une clé d'une autre relation
- formule logique sur les lignes d'une seule table `check`

Note – tout n'est pas exprimable par le langage de contraintes ; solution : autres outils SGBD (trigger, etc.)

Langage de contraintes – exemples d'ordres

```
create table t(
    a integer not null,
    b integer unique,
    c char(2),
    primary key (a,c),
    foreign key (b) references s(d)
);

alter table t add constraint somme check (a+b<12);
```

Contenu

Séquences

Situation :

- nécessite de générer des clefs numériques automatiquement

SGBD :

- type spécifique : `serial` en PostgreSQL, `autoincrement` en MySQL, etc.
- générateur de valeurs séquentielles `sequence` : génération d'une nouvelle valeur, accès a la dernière valeur générée (par exemple, Oracle n'a aucun moyen de générer de)

SERIAL – PostgreSQL

Dans la déclaration table :

```
create table t(
  a serial primary key,
  b integer
)
```

- pas un vrai type ! = type entier + séquence invisible

Utilisation :

```
insert into t(b) values(3);

insert into t values(DEFAULT, 4);
```

Séquences – ordres SQL

Création d'un générateur séquentiel `create sequence nomseq`

Utilisation :

- dans `select`, `insert`, `update`
- `nextval('nomseq')` génère la prochaine valeur (initialisée si premier appel)
- `currval('nomseq')` renvoie la valeur courante
- `setval('nomseq',val)` re-écrit la valeur de la séquence

Séquences – exemple

Table `people(num, nom, age)`, clé primaire `people(num)`

Table `owner(num, voiture)`, clé étrangère `owner(num)` réf. `people(num)`

```
create sequence numseq;

insert into people values(nextval('numseq'), 'Pierre', 21);
insert into owner values(currval('numseq'), 'Ferrari');
```

Compétences à acquérir

Analyse : savoir prédire le comportement d'une séquence d'ordres SQL en présence de contraintes

Implementation :

- savoir utiliser le langage de contraintes pour les modéliser
- savoir les limites du langage de contraintes

Remerciements

Le contenu de ce cours est grandement inspiré des cours d'Emmanuel Waller

- `https://www.lri.fr/~waller/`