Bases de données
SQL

LDD
Clés primaires et secondaires
Clés relatives et absolues
Clés artificielles ou subsidiaires

SQL_08
v120a

2022-03-24

Département d'informatique
Faculté des sciences

UNIVERSITÉ DE SHERBROOKE

Christina.Khnaisser@USherbrooke.ca
http://info.USherbrooke.ca/ckhnaisser
Luc.Lavoie@USherbrooke.ca
http://info.USherbrooke.ca/llavoie

Plan

- Clés primaires et secondaires
- Clés relatives et absolues
- Clés artificielles et naturelles
- Générateurs de clés artificielles
  - IDENTITY
  - SEQUENCE
  - SERIAL et les autres
- Quelques utilisations remarquables des clés

Clés primaires et secondaires

- Définition
- Origines
- Utilité
- Critères de choix
- Représentation sous SQL

Clés primaires et secondaires
Définition et origine

- Définition
  - Parmi les clés candidates d'une relation, une est désignée comme clé primaire; les autres (s'il en est) sont désignées comme clés secondaires.
- Origine
  - Les limites des fichiers séquentiels indexés... en 1966!

Clés primaires et secondaires
Utilité

- Pertinence du concept
  - Ce concept est inutile du strict point de vue de la théorie relationnelle.
- Toutefois
  - Le choix d'une clé candidate de référence (clé primaire) peut contribuer à la clarté et à la modifiabilité du schéma, particulièrement lors de la définition de clés étrangères, ce qui importe du point de vue du génie logiciel.
  - En dernier ressort, le choix d'une clé plutôt qu'une autre peut avoir une incidence sur la performance en fonction des tehcniques d'indexation utilisées.

Clés primaires et secondaires
Critère de choix

- Flou
  - nombre d'attributs constitutifs ?
  - taille totale ?
  - flexibilité au moment des sélections ?
  - flexibilité au moment des jointures ?

Clés primaires
Représentation sous SQL

- Clé primaire
  - PRIMARY KEY
  - les attributs la composant sont automatiquement marqués « not null »
- Clés secondaires
  - UNIQUE
  - les attributs les composant **ne sont pas** automatiquement marqués « not null »

Clés relatives et absolues

- Définition
- Pourquoi ?
- Quand ?
- Avantages (et inconvénients)
- Coexistence ?

Clé relative
définition

- Une clé est dite relative si et seulement si :
  - dont au moins un sous-ensemble de ses attributs est une clé référentielle d'une autre relation ;
  - dont au moins un attribut lui est propre (c'est-à-dire ne participant à aucune clé référentielle).
- Corolaire :
  - une clé relative comporte au moins deux attributs.

Clés relatives
Pourquoi ? Quand ?

- La représentation d'entités faibles produit naturellement des clés relatives.
- La représentation générale des associations aussi, sauf lors des optimisations pour le cas 1-1 (fusion) et pour le cas 1-N (clé référentielle).

Clé absolue

- Une clé est dite absolue si et seulement si :
  - elle ne comporte qu'un seul attribut ;
  - cet attribut lui est propre.

Clés relatives et absolues
Avantages (et inconvénients)

- Clé relative
  - Expression de la nature complète de la relation.
  - Puissance d'expression pour les jointures et les restrictions.
- Clé absolue
  - Simplification de l'expression des clés référentielles.
  - Réduction de l'encombrement ? Très rarement.
  - Réduction du temps d'accès ? Très rarement.

Coexistence des clés absolues et relatives

- Une clé relative ne devrait jamais être remplacée par une clé absolue, car elle induit une perte de validation de contrainte.
- Certains auteurs recommande doubler les clés relatives par une clé absolue, voire même d'utiliser préférentiellement cette dernière comme cible des clés référentielles.
- Cette tactique est rarement payante en pratique attendu les algorithmes contemporains d'indexation.
- En conclusion, cela ne devrait être utilisé qu'en dernier recours s'il y a un réel problème d'efficience et que le gain est vraiment significatif. Prendre en compte qu'il y aura consécutivement une augmentation de l'ensombrement associé à la relation.

Clés artificielles et naturelles

- Définition
- Pourquoi ?
- Quand ?
- Comment ?

Clé artificielle

- Clé absolue n'ayant aucune sémantique associée en regard du domaine d'application.

- Complément
  - Une clé est dite naturelle, si et seulement si elle n'est pas artificielle !

Clés artificielles
Pourquoi ? Quand ?

- Pour distinguer ce qui doit l'être (éviter les amalgames accidentels)
  - exemple : les humains, les animaux, les plantes, ...
- Pour éviter la propagation d'informations sensibles
  - exemple : le NAS (numéro d'assurance sociale)
- Pour mieux assurer l'évolutivité
  - cause : l'unicité et l'immutabilité des clés naturelles sont rarement garanties à long terme
- Pour optimiser le temps de calcul
  - exemple : clé naturelle multi-attributs ou volumineuse
- *Attention, ne jamais associer une signification à la valeur d'une clé artificielle, sinon elle n'est plus artificielle !*

Clés artificielles
Un code n'est pas une clé artificielle !

- Un code correspond à la génération « naturelle » ou « conventionnelle » d'une valeur uniquement associée à une entité.
- Un code est souvent représenté par une courte séquence alphanumérique.
- Un code a souvent une valeur mnémotechnique afin de pouvoir servir dans la programmation des requêtes.
- Un code est souvent utilisé à des fins d'affichage, par soucis de compacité dans des tableaux, par exemple.
- Un code est donc contingent à des règles externes et ne peux être modifié sans accord avec les experts du domaine.

- Le matricule de la RAMQ
- Le numéro d'assurance sociale
- Le CIP de l'Université de Sherbrooke

- Constat
  - En pratique, il n'est guère de bonne clé artificielle que celle qui ne sont pas fourni par le domaine d'application mais, plutôt, engendré par le SGBD lui-même.
- Note
  - Si ce ne sont pas de bonnes clés artificielles, cela n'empêche pas que celui puisse être (ou pas) de bons codes d'identification ☺

Clés artificielles
Comment les générer ?

- En SQL
  - SEQUENCE
- Dans de nombreux dialectes SQL
  - SMALLSERIAL
  - SERIAL
  - BIGSERIAL
  - UID
  - GUID
  - UUID

GÉNÉRATION DE CLÉS ARTIFICIELLES EN SQL

- Caractéristiques souhaitées
- Les solutions
  - IDENTITY
  - SEQUENCE
    - générateurs
    - fonctions
  - SERIAL, SMALLSERIAL, BIGSERIAL
  - UID, GUID, UUID

Génération de clés artificielles en SQL
Caractéristiques souhaitées

- Artificialité garantie lors de la génération
- Unicité garantie lors de la génération
- Unicité garantie dans un contexte transactionnel
- Automatisation de la génération lors de l'insertion

Clés artificielles en SQL ISO
Solution 1 : Par annotation de type (IDENTITY)

type_identitaire ::=
    type GENERATED [ALWAYS | AS DEFAULT]
    AS IDENTITY *séquence*

séquence ::=
    nom_de_sequence | *générateur*

Notes relatives à PostgreSQL

- Le *type* est limité aux seuls types numériques entiers (SMALLINT, INTEGER, BIGINT).
- La mise à disposition du mécanisme ISO a été graduelle, depuis la version 9.6 jusqu'à la 12.4.

Clés artificielles en SQL ISO
Solution 2 : Création d'un générateur (SEQUENCE)

création_de_séquence ::=

CREATE [TEMPORARY | TEMP]
SEQUENCE *nom_de_séquence* [AS *type*]
générateur
[OWNED BY { *nom_table.nom_colonne* | NONE }]

générateur ::=

[INCREMENT [BY] *incrément*]
[MINVALUE *valeur_min* | NO MINVALUE]
[MAXVALUE *valeur_max* | NO MAXVALUE]
[START [WITH] *début*]
[CACHE *cache*]
[[NO] CYCLE]

Clés artificielles en SQL ISO
Solution 2 : Création d'un générateur (SEQUENCE)... suite

- Notes relatives à PostgreSQL
  - L'expression standard AS *type* n'est pas offerte.
  - La clause OWNED BY est une extension PostgreSQL.

Clés artificielles en SQL ISO
Solution 2 : Utilisation d'un générateur (SEQUENCE)

- Pour obtenir *implicitement* la prochaine valeurs, lors d'une insertion, ne pas donner de valeur à l'attribut !
- Pour obtenir *explicitement* la prochaine valeur, utiliser la fonction standard NEXT VALUE FOR.

- Le mécanisme implicite est le même en postgreSQL.
- Pour le mécanisme explicite, voire la diapositive suivante.

Clés artificielles en SQL ISO
Solution 2 : Utilisation d'un générateur (SEQUENCE)... PostgreSQL

currval (regclass)	bigint	Renvoie la valeur la plus récemment obtenue avec nextval pour la séquence indiquée
lastval ()	bigint	Renvoie la valeur la plus récemment obtenue avec nextval (quelle que soit la séquence)
nextval (regclass)	bigint	Incrémente la séquence et renvoie la nouvelle valeur
setval (regclass, bigint)	bigint	Positionne la valeur courante de la séquence
setval (regclass, bigint, boolean)	bigint	Positionne la valeur courante de la séquence et son statut (booléen)

Les objets sequence sont de type regclass
voir https://docs.postgresql.fr/12/functions-sequence.html

SEQUENCE
Fonctions PostgreSQL

nextval (s)

- Avance l'objet séquence à sa prochaine valeur et renvoie cette valeur. Ce fonctionnement est atomique : même si de multiples sessions exécutent nextval concurrentiellement, chacune obtient sans risque une valeur de séquence distincte.

currval (s)

- Renvoie la valeur la plus récemment retournée par nextval pour cette séquence dans la session courante. (Une erreur est rapportée si nextval n'a jamais été appelée pour cette séquence dans cette session.) Parce qu'elle renvoie une valeur locale à la session, la réponse est prévisible, que d'autres sessions aient exécuté ou non la fonction nextval après la session en cours.

lastval ()

- Renvoie la valeur la plus récemment retournée par nextval dans la session courante. Cette fonction est identique à currval, sauf qu'au lieu de prendre le nom de la séquence comme argument, elle récupère la valeur de la dernière séquence utilisée par nextval dans la session en cours. Si nextval n'a pas encore été appelée dans la session en cours, un appel à lastval produit une erreur.

setval (s, v [, b])

- Réinitialise la valeur du compteur de l'objet séquence. La forme avec deux paramètres initialise le champ last_value de la séquence à la valeur précisée et initialise le champ is_called à true, signifiant que le prochain nextval avance la séquence avant de renvoyer une valeur. La valeur renvoyée par currval est aussi configuré à la valeur indiquée. Dans la forme à trois paramètres, is_called peut être initialisé à true ou à false. true a le même effet que la forme à deux paramètres. Positionné à false, le prochain nextval retourne exactement la valeur indiquée et l'incrémentation de la séquence commence avec le nextval suivant. De plus, la valeur indiquée par currval n'est pas modifiée dans ce cas. Par exemple,
  - SELECT setval('foo', 42); *Le nextval suivant retourne 43*
  - SELECT setval('foo', 42, true); *Comme ci-dessus*
  - SELECT setval('foo', 42, false); *Le nextval suivant retourne 42*

SEQUENCE
REMARQUE IMPORTANTE

- Pour éviter le blocage de transactions concurrentes qui obtiennent des nombres de la même séquence, une opération *nextval* n'est jamais annulée ; c'est-à-dire qu'une fois la valeur récupérée, elle est considérée utilisée, même si la transaction qui exécute *nextval* avorte par la suite. Cela signifie que les transactions annulées peuvent laisser des « trous » inutilisés dans la séquence des valeurs assignées.

SEQUENCE
SERIAL - DÉFINITION

- Le type SERIAL n'est pas un vrai type, mais plutôt un raccourci pour créer des attributs d'identifiants uniques :

```
CREATE TABLE nom_de_table (
    nom_de_colonne SERIAL,
    ...
);
```

- est équivalent à écrire :

```
CREATE SEQUENCE nom_de_table_nom_de_colonne_seq;
CREATE TABLE nom_de_table (
    nom_de_colonne INTEGER
        GENERATED AS DEFAULT AS IDENTITY nom_de_table_nom_de_colonne_seq,
    ...
);
ALTER SEQUENCE nom_de_table_nom_de_colonne_seq
    OWNED BY nom_de_table.nom_de_colonne;
```

- Un attribut de type entier est ainsi créée dont la valeur par défaut est obtenue par un générateur de séquence propre à l'attribut.
- Enfin, la séquence est marquée OWNED BY (possédée par) l'attribut pour qu'elle soit automatiquement supprimée si l'attribut ou la table est supprimé.

SEQUENCE
SERIAL – NOTES

- SMALLSERIAL, SERIAL et BIGSERIAL sont mis en oeuvre en utilisant des séquences correspondant respectivement à SMALLINT, INTEGER et BIGINT; ils en ont donc les caractéristiques.
- Dans la plupart des cas, une contrainte UNIQUE ou PRIMARY KEY est ajoutée pour interdire que des doublons soient créés par accident, mais ce n'est pas automatique.
- Pour affecter la valeur suivante de la séquence à un attribut, il faut préciser que la valeur par défaut de l'attribut doit être utilisée. Cela peut notamment se faire en excluant cet attribut de la liste des attributs de la commande INSERT.

SEQUENCE
LES UID, GUID ET AUTRES UUID

- Hors norme SQL.
- Spécifiés par l'ISO, la CEI et l'IETF.
- Disponibles par l'entremise de « packages » ou d'«extensions».

Contraintes de domaine applicables aux clés

- Clés candidates
- Clés référentielles

Contraintes de domaine applicables aux clés
Contraintes sur les valeurs

- Contraintes sur les clés candidates
  - au plus juste
    (tenter d'exclure *a priori* toute valeur non légitime)
- Contraintes sur les clés référentielles
  - ne pas répéter celles de la clé candidate de référence
  - au besoin, mettre des contraintes supplémentaires

RÉFÉRENCES

- Loney, Kevin ;
  Oracle Database 11g: The Complete Reference.
  Oracle Press/McGraw-Hill/Osborne, 2008.
  ISBN 978-0071598750.
- Date, Chris J. ;
  SQL and Relational Theory: How to Write Accurate SQL Code.
  2nd edition, O'Reilly, 2012.
  ISBN 978-1-449-31640-2.
- Le site d'Oracle (en anglais)
  - http://docs.oracle.com/cd/E11882_01/index.htm
- Le site de PostgreSQL (en français)
  - http://docs.postgresqlfr.org