Consistance des données dans une base

Novembre 2022

Maintien opérationnel et optimisation

Rappels SQL en profondeur:

- Création d'une table
- Les requêtes d'ajout
- Les requêtes d'extraction
- Les requêtes de mise à jour
- Les requêtes de destruction
- Exemple de requêtes d'extraction

Maintien opérationnel et optimisation

- ## Création d'une table

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type [ COMPRESSION compression_method ] [ COLLATE collation ] [ column_constraint [ ... ] ]
    | table_constraint
    | LIKE source_table [ like_option ... ] }
    [, ... ]
] )
```

Example:

```
CREATE TABLE films (
    code        char(5) CONSTRAINT firstkey PRIMARY KEY,
    title       varchar(40) NOT NULL,
    did         integer NOT NULL,
    date_prod   date,
    kind        varchar(10),
    len         interval hour to minute
);
```

https://www.postgresql.org/docs/current/sql-createtable.html

Maintien opérationnel et optimisation

- ## Les requêtes SQL d'ajout

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    [ * | expression [ [ AS ] output_name ] [, ...] ]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
    [ HAVING condition ]
    [ WINDOW window_name AS ( window_definition ) [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
    [ ORDER BY expression [ ASC | DESC | USING operator ] [ NULLS { FIRST | LAST } ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start [ ROW | ROWS ] ]
    [ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } { ONLY | WITH TIES } ]
    [ FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE } [ OF table_name [, ...] ] [ NOWAIT | SKIP LOCKED ] [...] ]
```

https://www.postgresql.org/docs/current/sql-select.html

Maintien opérationnel et optimisation

- ## Les requêtes SQL d’extraction

```
[ WITH [ RECURSIVE ] requête_with [, ...] ]
INSERT INTO nom_table [ AS alias ] [ ( nom_colonne [, ...] ) ]
    [ OVERRIDING { SYSTEM | USER } VALUE ]
    { DEFAULT VALUES | VALUES ( { expression | DEFAULT } [, ...] ) [, ...] | requête }
    [ ON CONFLICT [ cible_conflit ] action_conflit ]
    [ RETURNING * | expression_sortie [ [ AS ] nom_sortie ] [, ...] ]
```

https://www.postgresql.org/docs/current/sql-insert.html

Maintien opérationnel et optimisation

- Les requêtes SQL de mise à jour

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
UPDATE [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    SET { column_name = { expression | DEFAULT } |
          ( column_name [, ...] ) = [ ROW ] ( { expression | DEFAULT } [, ...] ) |
          ( column_name [, ...] ) = ( sub-SELECT )
        } [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition | WHERE CURRENT OF cursor_name ]
    [ RETURNING * | output_expression [ [ AS ] output_name ] [, ...] ]
```

https://www.postgresql.org/docs/current/sql-update.html

Maintien opérationnel et optimisation

- ## Les requêtes SQL de destruction

[WITH [RECURSIVE] with_query [, ...]]
DELETE FROM [ONLY] table_name [*] [[AS] alias]
    [**USING** from_item [, ...]]
    [**WHERE** condition | WHERE CURRENT OF cursor_name]
    [RETURNING * | output_expression [[AS] output_name] [, ...]]

https://www.postgresql.org/docs/current/sql-update.html

Maintien opérationnel et optimisation

Les requêtes de SELECT avancé

- SELECT définit la *projection*, c'est-à-dire la liste des colonnes renvoyées par la requête
- Les colonnes sont séparées par des virgules
- Chaque colonne peut être renommée avec *AS*
- Les doublons peuvent être supprimés avec *DISTINCT*

- https://www.postgresql.org/docs/current/queries-select-lists.html

Query Editor | Query History

```
1  WITH couples AS(
2      SELECT *
3      FROM  (VALUES
4              ('un', 1),
5              ('deux', 2),
6              ('trois', 3),
7              ('deux',2)
8             ) AS t (cle, valeur)
9  )
10 SELECT DISTINCT
11     cle AS lettres,
12     valeur AS nombres
13 FROM couples
14 ORDER BY valeur;
```

Data Output

	lettres text	nombres integer
1	un	1
2	deux	2
3	trois	3

Explain | Messages | Notifications

```
Successfully run. Total query runtime: 37 msec.
3 rows affected.
```

Maintien opérationnel et optimisation

Les requêtes de SELECT avancé

- FROM fait une référence à une table *nommée* ou *dérivée*
- Une table est dérivée si elle est issue d'une *sous-requête* ou construite avec des *jointures*
- Si plusieurs tables sont présentes (séparées par des virgules), le *produit cartésien* des lignes est formé.
- https://www.postgresql.org/docs/current/queries-select-lists.html

```
SELECT *
FROM
    tableexp."TableA"
```

	a1 [PK] integer	a2 [PK] integer
1	1	10
2	2	20
3	3	30

```
SELECT *
FROM (
    SELECT a1
    FROM tableexp."TableA"
    WHERE a2 > 15
    ) t
```

	a1 integer
1	2
2	3

```
SELECT *
FROM
    tableexp."TableA",
    tableexp."TableB"
```

	a1 integer	a2 integer	b1 integer	b2 integer	b3 integer
1	1	10	1	10	100
2	2	20	1	10	100
3	3	30	1	10	100
4	1	10	2	20	200
5	2	20	2	20	200
6	3	30	2	20	200

Maintien opérationnel et optimisation

Les requêtes de SELECT avancé JOINTURE

- Une jointure est une table *dérivée* depuis plusieurs tables (nommées ou dérivées)
- Les jointures *naturelles* utilisent les noms d'attributs communs entre les tables
- Les jointures qualifiées sont *INNER* ou *LEFT | RIGHT | FULL OUTER*

```
SELECT *
FROM tableexp."TableA"a1
INNER JOIN tableexp."TableB"
ON a1 = b1
```

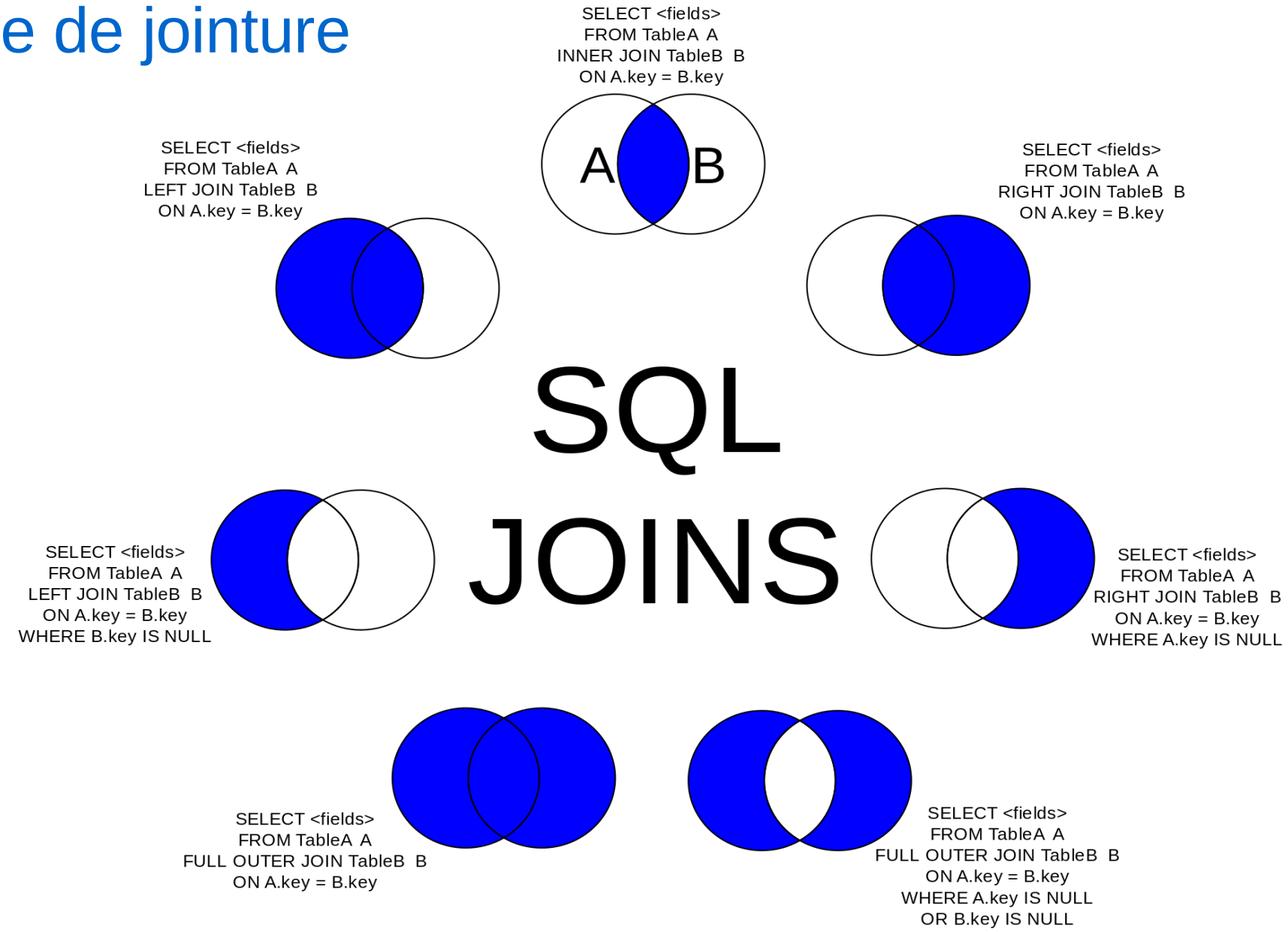
	a1 integer	a2 integer	b1 integer	b2 integer	b3 integer
1	1	10	1	10	100
2	2	20	2	20	200

```
SELECT *
FROM tableexp."TableA"a1
LEFT OUTER JOIN tableexp."TableB"
ON a1 = b1
```

	a1 integer	a2 integer	b1 integer	b2 integer	b3 integer
1	1	10	1	10	100
2	2	20	2	20	200
3	3	30	[null]	[null]	[null]

Maintien opérationnel et optimisation

Type de jointure



Maintien opérationnel et optimisation

Les requêtes de SELECT avancé

- FROM fait une référence à une table *nommée* ou *dérivée*
- Une table est dérivée si elle est issue d'une *sous-requête* ou construite avec des *jointures*
- Si plusieurs tables sont présentes (séparées par des virgules), le *produit cartésien* des lignes est formé.
- https://www.postgresql.org/docs/current/queries-select-lists.html

```
SELECT *
FROM
    tableexp."TableA"
```

	a1 [PK] integer	a2 [PK] integer
1	1	10
2	2	20
3	3	30

```
SELECT *
FROM (
    SELECT a1
    FROM tableexp."TableA"
    WHERE a2 > 15
    ) t
```

	a1 integer
1	2
2	3

```
SELECT *
FROM
    tableexp."TableA",
    tableexp."TableB"
```

	a1 integer	a2 integer	b1 integer	b2 integer	b3 integer
1	1	10	1	10	100
2	2	20	1	10	100
3	3	30	1	10	100
4	1	10	2	20	200
5	2	20	2	20	200
6	3	30	2	20	200

Maintien opérationnel et optimisation

- Les clés primaires
- Les contraintes
  - Unique
  - Inter-table
  - Multi-colonne

Maintien opérationnel et optimisation

- ## Les clés primaires

- Une clé est un ensemble d'un ou plusieurs attributs qui permet d'identifier à coup sûr *une (et pas plus)* occurrence dans la table
  - Exemple : https://docs.google.com/spreadsheets/d/1m1idjtGTuF0UUP4YUG5rJhBDXxlf8LFKPbquXE6ZqI8/edit?usp=sharing
  - Quelle est la clé cette table ?

Maintien opérationnel et optimisation

- ## Les clés primaires

- définition :
  - permet d'identifier chaque enregistrement dans une table de base de données
  - Chaque enregistrement de cette clé primaire doit être UNIQUE et ne doit pas contenir de valeur NULL.
  - Trés souvant la clé primaire est un entier autoincrementé
    - Mysql : type AUTO_INCREMENT
    - Postresql : Type Serial
    - SQLSqlite : C'est un compteur + un contraint unique

Maintien opérationnel et optimisation

- ## Les constraints
- définition :
  - Les contraintes permettent d'assurer un cohérence des données dans une base de données relationnelles.
  - Elles peuvent être de plusieurs types
    - Une contrainte dans une colonnes comme UNIQUE
    - Une contrainte entre plusieurs colonnes
    - Une contrainte entre les tables
- https://docs.postgresql.fr/10/ddl-constraints.html

Maintien opérationnel et optimisation

- ## La contrainte unique

- Définition : La contrainte permet d'avoir une valeur et une seule à l'exception de la valeur null
- Example d'utilisation :
  - Une table facture où le numéro de facture serait unique
  - Une table département où les numéros de département
- https://docs.postgresql.fr/10/ddl-constraints.html

```
CREATE TABLE facture (
    no_facture integer UNIQUE,
    montant int
    paye int
);
```

Maintien opérationnel et optimisation

La contrainte dans une colonne

- Définition : Certaines bases de données comme postgres permettent de contrôler les valeurs d'une colonne

- Example d'utilisation :
  - Une table produit à obligatoire un prix strictement positif
  - Une table produit peut avoir une promotion qui n'est pas plus supérieur aux prix.

- https://docs.postgresql.fr/10/ddl-constraints.html

```
CREATE TABLE produits (
    no_produit integer,
    nom text,
    prix numeric CHECK (prix > 0),
    prix_promotion numeric CHECK (prix_promotion > 0),
    CHECK (prix > prix_promotion)
);
```

Maintien opérationnel et optimisation

La multi colonne

- Définition : Certaines bases de données comme Postgres permettent de contrôler l'unicité des valeurs de 2 colonnes.

- https://docs.postgresql.fr/10/ddl-constraints.html

```
CREATE TABLE someTable (
    id serial PRIMARY KEY,
    col1 int NOT NULL,
    col2 int NOT NULL,
    UNIQUE (col1, col2)
)

Si la table existe:
ALTER TABLE someTable
ADD UNIQUE (col1, col2)
```

Maintien opérationnel et optimisation

La contrainte externe (clé étrangére)

- Définition : La table avec la clé étrangère est appelée la table enfant et la table avec la clé primaire est appelée la table référencée ou parente.

```
CREATE TABLE commandes (
                id_commande integer PRIMARY KEY,
                no_produit integer REFERENCES produits (no_produit),
                quantite integer
);
```

- Example utilisation :
  - Un numero de produit dans une commande doit existe dans la table produit.

- https://docs.postgresql.fr/10/ddl-constraints.html

Maintien opérationnel et optimisation

- TD

Afin que les applications ne puissent pas injecter de données incohérant modifier votre base de données via des contraintes:

Utiliser la contrainte unique
Contrainte dans une colonne
Et les contraintes externes