BDW - Conception des BD - évolution

Fabien Duchateau

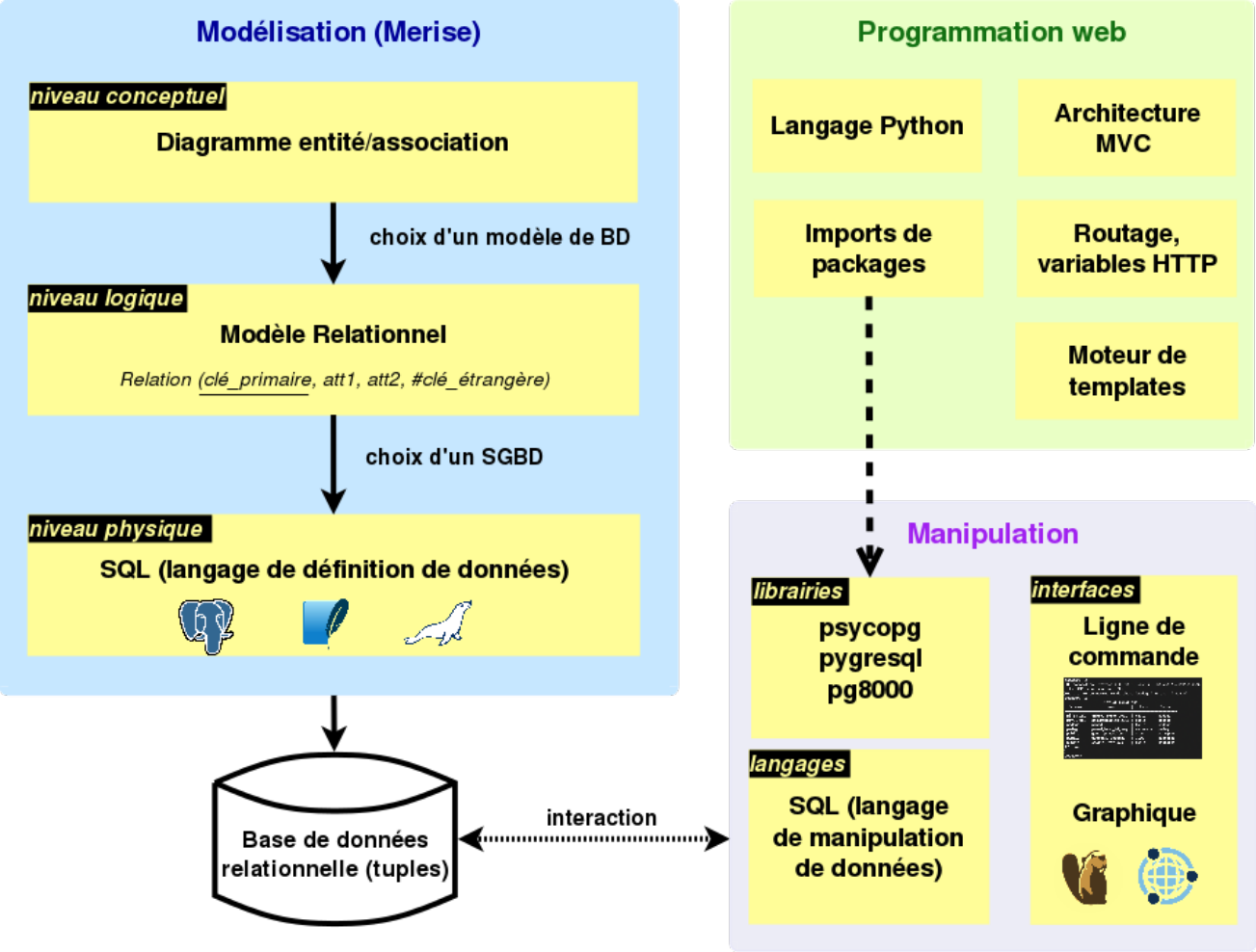
fabien.duchateau [at] univ-lyon1.fr

Université Claude Bernard Lyon 1

2024 - 2025

https://perso.liris.cnrs.fr/fabien.duchateau/BDW/

Positionnement dans BDW



Ces diapositives utilisent le genre féminin (e.g., chercheuse, développeuses) plutôt que l'écriture inclusive (moins accessible, moins concise, et pas totalement inclusive)

Motivation

Une base de données est amenée à évoluer :

- Contraintes non implémentables à la création (e.g., deux tables qui se référencent)
- Corrections en phase de maintenance
- Nouveaux besoins ou fonctionnalités, qui nécessitent une modification de l'application et de son modèle de données

Éléments qui peuvent évoluer : base de données, tables, attributs, contraintes, n-uplets, etc.

Le langage SQL permet les mises à jour (CR**U**D) et les suppressions (CRU**D**) sur ces différents éléments

Plan

Mise à jour

Suppression

Notions avancées

Généralités

Instructions SQL de mise à jour :

- ALTER DATABASE pour une base de données (e.g., renommage)
- ALTER SCHEMA pour un répertoire schéma (e.g., renommage)
- ALTER TABLE pour une table, ses attributs ou ses contraintes (e.g., renommage de la table ou d'un attribut, ajout d'une contrainte)
- UPDATE pour les n-uplets (mise à jour de valeur)

---

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

Exemples de mise à jour de table

```sql
ALTER TABLE series
RENAME TO Series_TV;
```

Modification du nom de la table

```sql
ALTER TABLE series
ADD producteur VARCHAR(50);
```

Ajout d'un attribut `producteur`

```sql
ALTER TABLE series
ALTER COLUMN producteur TYPE VARCHAR(100);
```

Modification du type de l'attribut `producteur`

Exemples de mise à jour de table (2)

```
1 ALTER TABLE series
2 RENAME producteur TO prod;
```

Renommage de l'attribut `producteur` en `prod`

```
1 ALTER TABLE series
2 ADD CONSTRAINT check_prod CHECK(prod != '');
```

Ajout d'une contrainte CHECK *sur l'attribbut `prod`*

```
1 ALTER TABLE Series
2 ADD CONSTRAINT prod_unique UNIQUE(prod);
```

Ajout d'une contrainte UNIQUE *sur l'attribbut `prod`*

Mise à jour de n-uplet

UPDATE *nom_table*
SET $att_1 = e_1$, [$att_2 = e_2$, ...]
[**WHERE** *condition*] **;**

- Cette commande met à jour les n-uplets qui vérifient la condition *condition*
- Chaque att_i prend la valeur calculée par l'expression correspondante e_i :
  - e_i peut utiliser les (anciennes) valeurs de att_i
  - e_i peut être une sous-requête qui retourne un unique résultat
- Si le WHERE est absent, tous les n-uplets sont modifiés

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

Exemples de mise à jour de n-uplet

```sql
UPDATE Jouer
SET salaire = 3000;
```

Tous les salaires mis à 3000

```sql
UPDATE Jouer
SET salaire = 4000
WHERE numINSEE = 222;
```

Les salaires de l'actrice 222 mis à 4000

```sql
UPDATE Jouer
SET salaire = 4000
FROM episodes join saisons using(idsaison)
WHERE numINSEE = numinsee and nomsérie = 'Games of Thrones';
```

Les salaires des actrices de Game of thrones mis à 4000

Plan

Mise à jour

Suppression

Notions avancées

Généralités

Instructions SQL de suppression :

- DROP DATABASE pour une base de données
- DROP SCHEMA pour un répertoire schéma
- DROP TABLE pour une table
- ALTER TABLE pour un attribut ou une contrainte
- DELETE pour un n-uplet

---

https://www.postgresql.org/docs/current/sql-dropdatabase.html
https://www.postgresql.org/docs/current/sql-dropschema.html
https://www.postgresql.org/docs/current/sql-droptable.html
https://www.postgresql.org/docs/current/sql-delete.html

Exemples de suppression

```
1 ALTER TABLE Series
2 DROP CONSTRAINT prod_unique;
```

Suppression de la contrainte `prod_unique`

```
1 ALTER TABLE Series
2 DROP COLUMN prod;
```

Suppression de l'attribut `prod`

```
1 DROP TABLE IF EXISTS Series CASCADE;
```

Suppression d'une table en cascade (à éviter)

```
1 DROP SCHEMA IF EXISTS series CASCADE;
```

Suppression du schéma complet (recommandé)

Suppression de n-uplets

DELETE FROM *nom_table*
[**WHERE** *condition*] **;**

- Cette commande supprime les n-uplets de *nom_table* qui vérifient la condition *condition*
- La condition peut être aussi complexe qu'une condition exprimée dans le WHERE d'un SELECT :
  - *condition* peut contenir des requêtes imbriquées
  - ces requêtes imbriquées ne peuvent pas faire référence à *nom_table* (car elle est en cours de modification)
- Si la clause optionnelle WHERE est absente, tous les n-uplets de *nom_table* sont supprimés

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

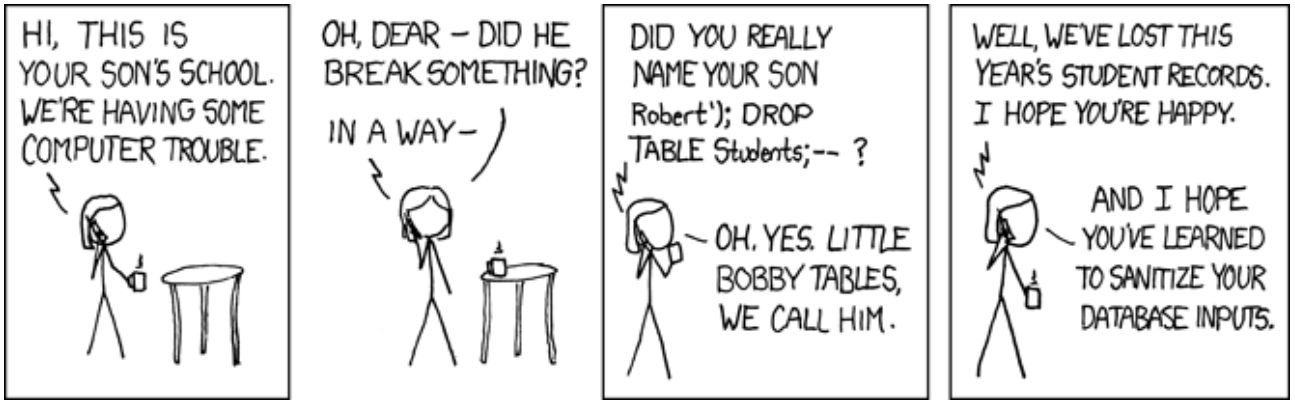
Exemples de suppression de n-uplets

```
1 DELETE FROM Actrices
2 WHERE numINSEE = 111;
```

Suppression de l'actrice 111 (effective uniquement si l'actrice 111 n'apparait plus dans la table JOUER*, contrainte de clé étrangère)*

```
1 DELETE FROM Jouer;
```

Suppression de toutes les instances JOUER



http://xkcd.com/

Plan

Mise à jour

Suppression

Notions avancées

Rôles et privilèges

Toutes les utilisatrices d'une BD n'ont pas les mêmes autorisations pour définir et manipuler des données :

- Un rôle est un utilisateur ou groupe d'utilisateurs
- Un privilège est une autorisation d'action sur un objet (e.g., créer une table, modifier une table, supprimer des n-uplets) accordée à un rôle
- Exemple : sur le serveur `bd-pedago`, vous pouvez créer des (répertoires) schémas, des tables mais pas de base de données

---

https://www.postgresql.org/docs/current/user-manag.html
https://www.postgresql.org/docs/current/ddl-priv.html

Création de vue

CREATE VIEW *nom_vue*
AS SELECT ... **;**

- Une **vue** est une requête à laquelle on donne un nom
- Elle s'utilise comme une table, via un SELECT
- La vue est recalculée à chaque utilisation
- Pas d'opération de mise à jour directement sur une vue
- La vue permet de définir des privilèges (lecture) sur certains n-uplets d'une table

Exemple de création de vue

```sql
CREATE VIEW episodes_tbbt AS
SELECT *
FROM Saisons s JOIN Episodes e USING(idsaison)
WHERE s.nomsérie = 'The Big Bang Theory'
ORDER BY idsaison, numéro;
```

Création d'une vue nommée `episodes_tbbt` contenant les épisodes de la série The Big Bang Theory

```sql
SELECT * FROM episodes_tbbt;
```

Utilisation de la vue `episodes_tbbt`

```
fduchateau=> select * from episodes_tbbt ;
 idsaison | datelancement |      nomsérie       | numéro |           titre
----------+---------------+---------------------+--------+---------------------------
        1 | 2011-09-22    | The Big Bang Theory |      1 | The Skank Reflex Analysis
        2 | 2012-09-27    | The Big Bang Theory |      1 | The Date Night Variable
(2 rows)
```

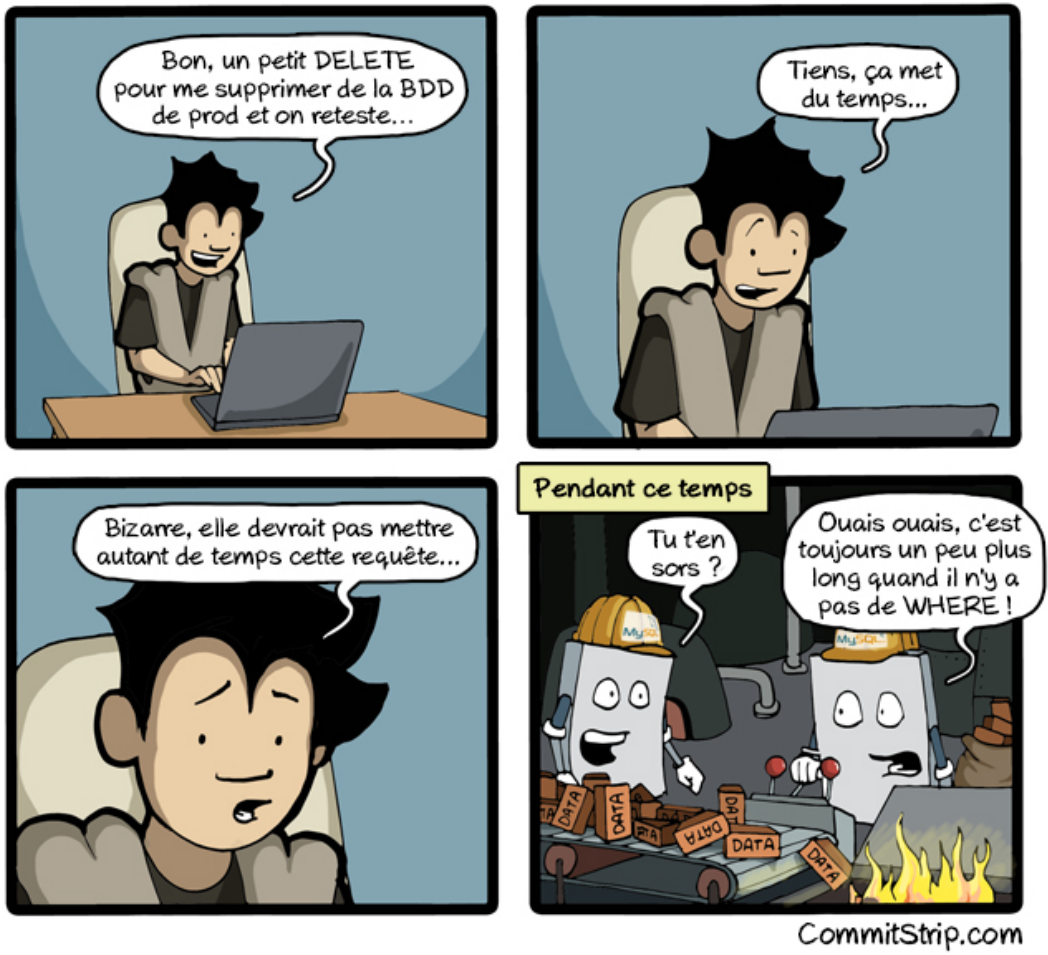
En résumé

SQL comme langage de définition de données :

- CREATE, DROP, ALTER

SQL comme langage de manipulation de données :

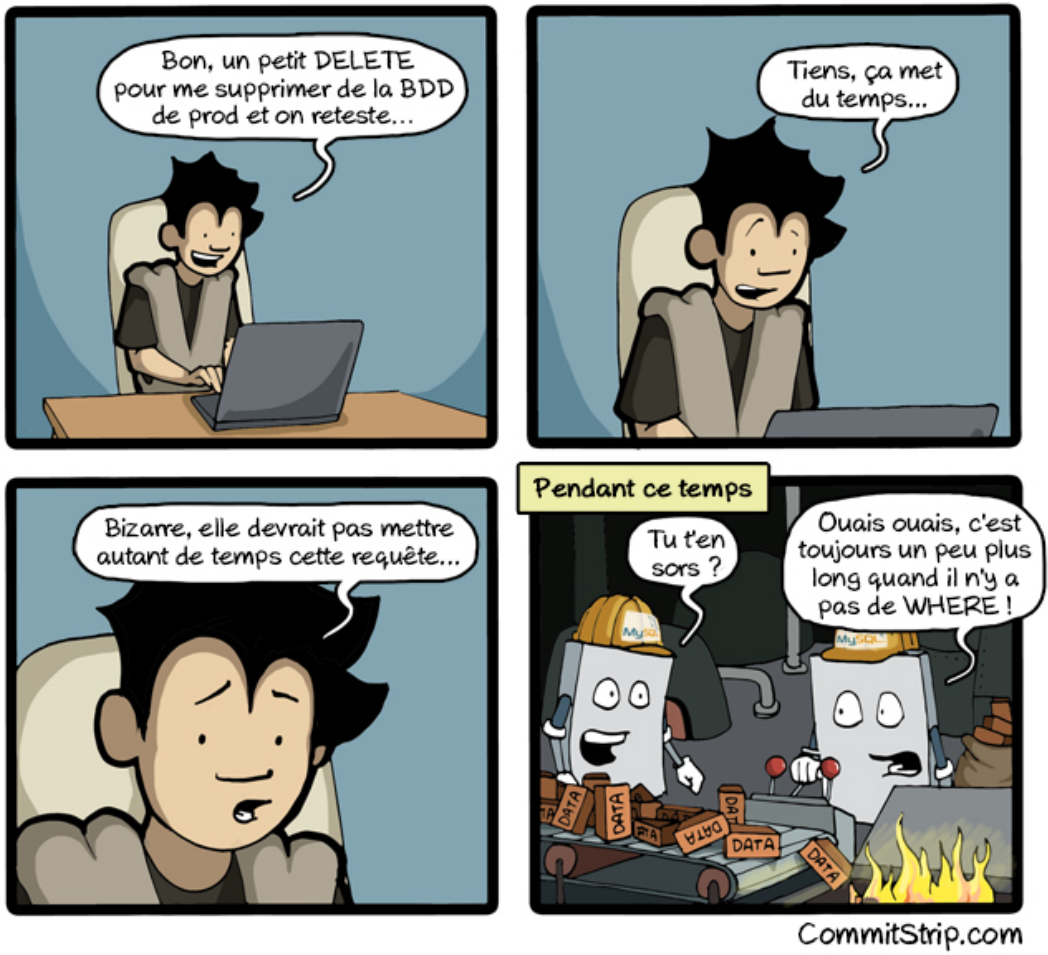
- SELECT, INSERT, DELETE, UPDATE



En résumé

SQL comme langage de définition de données :

- CREATE, DROP, ALTER

SQL comme langage de manipulation de données :

- SELECT, INSERT, DELETE, UPDATE