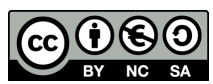
BDW - Conception des BD - évolution

Fabien Duchateau

fabien.duchateau [at] univ-lyon1.fr

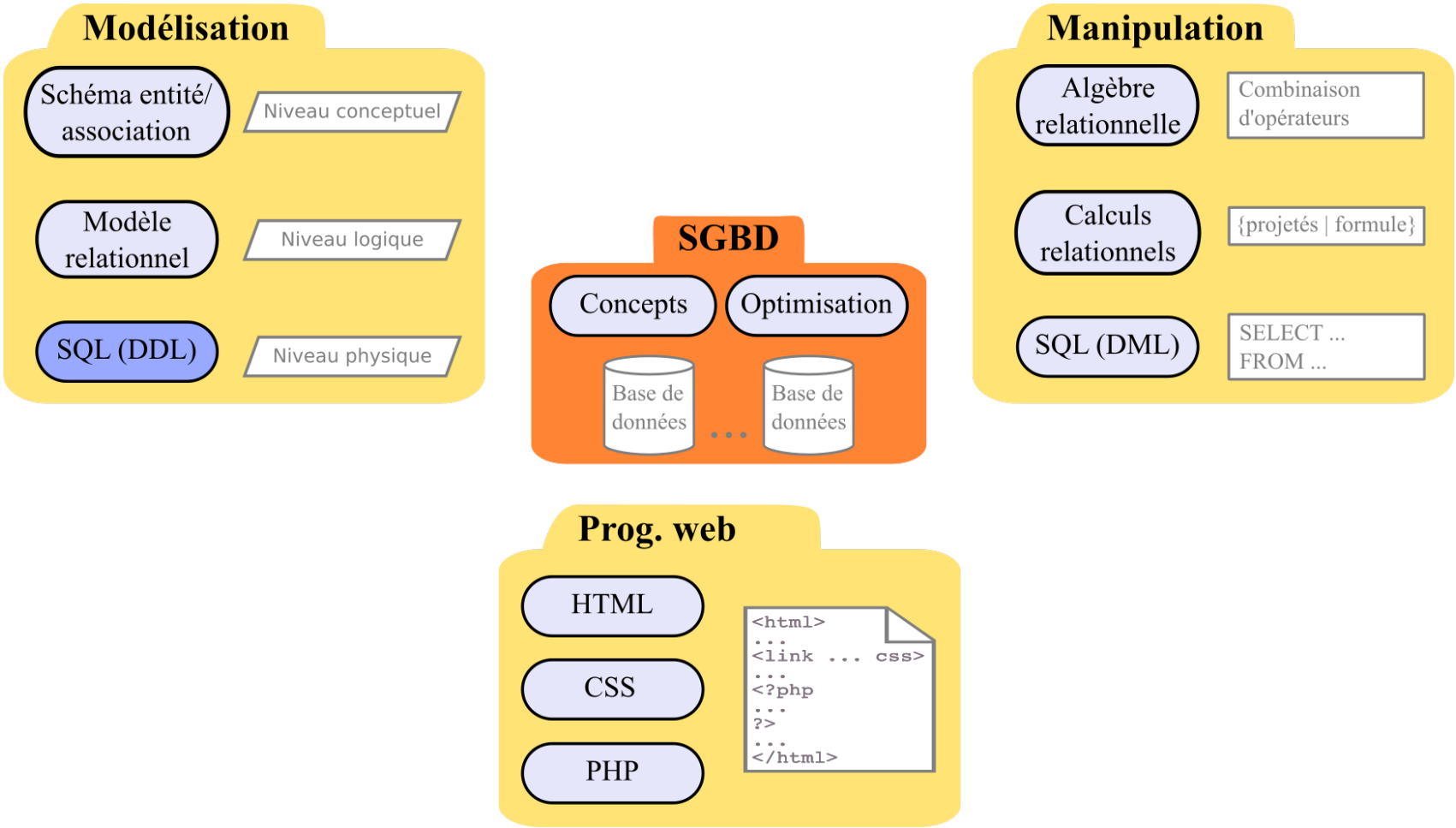
Université Claude Bernard Lyon 1

2023 - 2024



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

Motivation (2)

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

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

http://wumo.com/wumo

Plan

Généralités

Opérations fréquentes de mise à jour :

- Ajout d'information (dans le schéma d'une table)
- Modification, dont renommage

Dans la suite, mises à jour pour :

- Base de données (ALTER DATABASE)
- Table (ALTER TABLE)
- Attribut (ALTER TABLE)
- Contrainte (ALTER TABLE)
- N-uplet (UPDATE)

Mise à jour de base de données

ALTER DATABASE *nom_bd* **CHARACTER SET** = *nom_jeu* **;**
ALTER DATABASE *nom_bd* **COLLATE** = *nom_coll* **;**

- La commande ALTER DATABASE permet de modifier le jeu de caractères ou la collation de la base de données *nom_bd*

---

https://mariadb.com/kb/en/mariadb/alter-database/
https://mariadb.com/kb/en/mariadb/character-sets/

Mise à jour de table

La table est un élément crucial d'une base de données. De nombreuses mises à jour sont possibles concernant :

- Des caractéristiques générales (jeu de caractère, moteur de stockage, etc.)
- Les attributs et contraintes
- Les index
- Les partitionnements
- La désactivation des contraintes (sauf celles d'unicité)
- ...

---

https://mariadb.com/kb/en/mariadb/alter-table/

Mise à jour de table - moteur de stockage

ALTER TABLE *nom_table* **ENGINE** = *moteur* **;**

- Cette commande permet de modifier le moteur de stockage de la table *nom_table* en *moteur*

http://www.richskipworth.co.uk/

Mise à jour de table - renommage

ALTER TABLE *ancien_nom* **RENAME TO** *nouveau_nom* **;**

- Cette commande permet de renommer la table *ancien_nom* en *nouveau_nom*

Exemples de mise à jour de table

Mise à jour de la table Séries : son moteur de stockage (initialement InnoDB) devient MyISAM

Mise à jour de la table Séries : son nom devient Series_TV

Mise à jour d'attribut - ajout

ALTER TABLE *nom_table* **ADD** *att type* [*options*] **;**

- Cette commande permet d'ajouter un attribut *att* de type *type* à la table *nom_table*
- Les options de l'attribut sont identiques à celles de la création de table (e.g., NOT NULL, UNIQUE, AUTO_INCREMENT)
- Une autre option permet de spécifier la position de l'attribut dans le schéma de la table, avec les valeurs FIRST ou AFTER *autre_att*

L'ajout d'attribut concerne une table (d'où la commande ALTER TABLE)

Mise à jour d'attribut - modification

ALTER TABLE *nom_table*
MODIFY *att nouveau_type* [*options*] **;**

- Cette commande permet de modifier la définition d'un attribut, notamment de changer son type initial en *nouveau_type*
- Les options de l'attribut sont identiques à celles de la création de table (e.g., NOT NULL, UNIQUE, AUTO_INCREMENT)
- Une autre option permet de spécifier la position de l'attribut dans le schéma de la table, avec les valeurs FIRST ou AFTER *autre_att*

Mise à jour d'attribut - renommage

ALTER TABLE *nom_table*
CHANGE COLUMN *att nouveau_nom type* [*options*] **;**

- Cette commande a le même comportement que ALTER TABLE ...MODIFY, mais permet en plus de renommer l'attribut *att* en *nouveau_nom*
- La définition de l'attribut doit être complète (même si l'on veut juste renommer)
- Les options de l'attribut sont identiques à celles de la création de table (e.g., NOT NULL, UNIQUE, AUTO_INCREMENT)
- Une autre option permet de spécifier la position de l'attribut dans le schéma de la table, avec les valeurs FIRST ou AFTER *autre_att*

Exemples de mise à jour d'attribut

```
20 /*DESC Serie;*/
```

Ajout d'un attribut producteur (de type chaine de caractères variable de longueur 100) dans la table Séries

```
26 /*DESC Serie;*/
```

Modification de l'attribut producteur : son type passe à 200 caractères, il n'accepte pas les valeurs nulles et est placé en première position dans le schéma de la table

```
30 /*DESC Serie;*/
```

*Renommage de l'attribut producteur en prod. Il est nécessaire de spécifier à nouveau le type (*VARCHAR(200)*) et les options (*NOT NULL*) de l'attribut*

Mise à jour de contrainte - ajout

ALTER TABLE *nom_table*
ADD CONSTRAINT *nom_contr def_contr* **;**

- Cette commande permet d'ajouter une contrainte à la table *nom_table*
- Le nom de la contrainte *nom_contr* est optionnel (au besoin, un nom sera généré par le SGBD)
- La définition de la contrainte *def_contr* est identique à celle de création de table

Exemples de mise à jour de contrainte

Ajout d'une contrainte NOT NULL *en modifiant la définition de l'attribut prod (et son type également en* INTEGER)

Ajout d'une contrainte CHECK *sur le salaire de la table Jouer*

Ajout d'une contrainte FOREIGN KEY *sur l'attribut prod (qui doit désormais être dans la table Actrices)*

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 requête à condition qu'elle renvoie un unique résultat et que *nom_table* n'apparaisse pas dans un FROM de cette requête
- Si le WHERE est absent, tous les n-uplets de *nom_table* sont modifiés

Exemples de mise à jour de n-uplet

```
47 SET salaire = 3000;
```

Mise à jour de tous les n-uplets de la table Jouer (tous les salaires sont mis à 3000)

```
51 SET salaire = 4564
52 WHERE numINSEE = 222;
```

Mise à jour des n-uplets de Jouer qui concernent l'actrice possédant le numINSEE 222 : ses salaires sont mis à 4564

En résumé

Des commandes pour modifier :

- Une base de données avec ALTER DATABASE
- Une table, y compris son schéma (attribut, contrainte), avec ALTER TABLE
- Des n-uplets avec UPDATE

Démo avec MariaDB (script SQL en ligne)

Plan

Généralités

Dans la suite, suppression pour :

- Base de données
- Table
- Attribut
- Contrainte
- N-uplet

Suppression de base de données

DROP DATABASE [**IF EXISTS**] *nom_bd* **;**

- Cette commande supprime la base de données *nom_bd* (et donc toutes les tables qu'elle contient)
- L'option IF EXISTS permet d'éviter une erreur si la base n'existe pas
- Conseil : créer une sauvegarde de la base (export d'un dump) avant de la supprimer

---

https://mariadb.com/kb/en/mariadb/drop-database/

Suppression de table

DROP TABLE [**IF EXISTS**] *nom_table* **;**

- Cette commande permet de supprimer la table *nom_table*
- L'option IF EXISTS permet d'éviter une erreur si la base n'existe pas
- Si une clé étrangère d'une autre table référence la table à supprimer, impossible de supprimer la table sans supprimer d'abord la contrainte de clé étrangère
- Le mot clé CASCADE permet de déclencher automatiquement la suppression des clés étrangères (supporté sous MariaDB, mais non fonctionnel)

---

https://mariadb.com/kb/en/mariadb/drop-table/

Suppression d'attribut

ALTER TABLE *nom_table* **DROP COLUMN** *att* **;**

- Cette commande supprime l'attribut *att* de la table *nom_table*
- C'est une modification du schéma de la table, d'où la commande ALTER TABLE

https://mariadb.com/kb/en/mariadb/alter-table/

Suppression de contrainte

ALTER TABLE *nom_table* **DROP PRIMARY KEY ;**

- Cette commande permet de supprimer la contrainte de clé primaire de la table *nom_table*

ALTER TABLE *nom_table* **DROP FOREIGN KEY** *nom_cle* **;**

- Cette commande permet de supprimer la contrainte de clé étrangère *nom_cle* de la table *nom_table*

---

https://mariadb.com/kb/en/mariadb/alter-table/

Suppression de contrainte (2)

ALTER TABLE *nom_table* **DROP CONSTRAINT** *nom_contr* **;**

- Cette commande permet de supprimer la contrainte *nom_contr* (i.e., le nom donné à la contrainte lors de sa création avec CONSTRAINT)
- Si c'est le SGBD qui avait affecté un nom à la contrainte, on peut retrouver ce nom en fouillant les tables systèmes (BD *information_schema*, et en particulier la table *TABLE_CONSTRAINTS*)

---

https://mariadb.com/kb/en/mariadb/information-schema-tables/

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://mariadb.com/kb/en/mariadb/delete/

Exemples de suppression

```
56 ALTER TABLE Series DROP FOREIGN KEY fk_prod;
57 /* ALTER TABLE Jouer DROP check_salaire; */
```

Suppression de contraintes de clé primaire et de clé étrangère dans la table Séries

```
61 /*DESC Serie;*/
```

Suppression de l'attribut prod de la table Séries

Suppression de la table Séries

Exemples de suppression (2)

```
68 WHERE numINSEE = 111;
```

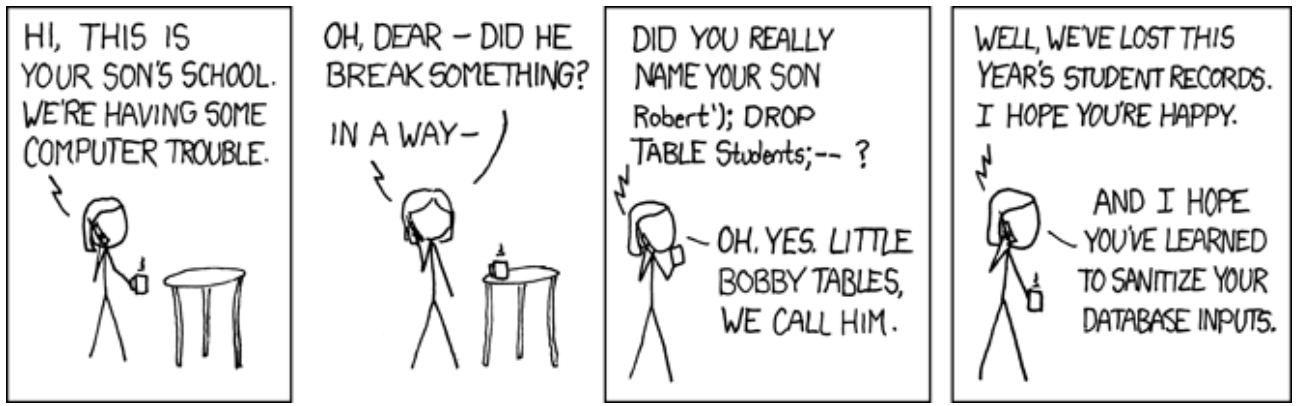
Suppression du n-uplet identifié par un numINSEE valant 111 de la table Actrices

Suppression de tous les n-uplets de la table Jouer

En résumé

Des commandes pour supprimer :

- Une base de données avec DROP DATABASE
- Une table avec DROP TABLE
- Un attribut ou une contrainte avec ALTER TABLE
- Des n-uplets avec DELETE FROM



http://xkcd.com/

Plan

Privilèges

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

- Un privilège est une autorisation d'action (e.g., créer une table, modifier une table, supprimer des n-uplets) à un niveau défini (e.g., global, sur une base de données, sur une table)
- Un rôle regroupe un ensemble de privilèges
- Une utilisatrice se voit accordée un ou plusieurs rôles/privilèges, qui définissent ce qu'elle peut faire

---

https://mariadb.com/kb/en/mariadb/account-management-sql-commands/
https://mariadb.com/kb/en/mariadb/grant/
https://mariadb.com/kb/en/mariadb/roles-overview/

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
SELECT *
FROM Saisons s NATURAL JOIN Episodes e
WHERE s.nomSerie = 'The Big Bang Theory' AND YEAR(s.
    dateLancement) = 2012;
```

Création d'une vue nommée tbbt2012, et qui contient les épisodes de la série The Big Bang Theory pour la saison 2012

Utilisation de la vue tbbt2012

Requête SQL: SELECT * FROM `tbbt2012` LIMIT 0, 30 ;
Lignes: 1

nomSerie	idSaison	numero	titre
The Big Bang Theory	2	1	The Date Night Variable

En résumé

SQL comme langage de définition de données :

- CREATE, DROP, ALTER

SQL comme langage de manipulation de données :

- SELECT, INSERT, DELETE, UPDATE