Bases de données

Cours 5
Requêtes SQL (Suite et fin)

Marie Pelleau
marie.pelleau@univ-cotedazur.fr

12 décembre 2023

Exemple fil-rouge : bibliothèque

Attributs des entités

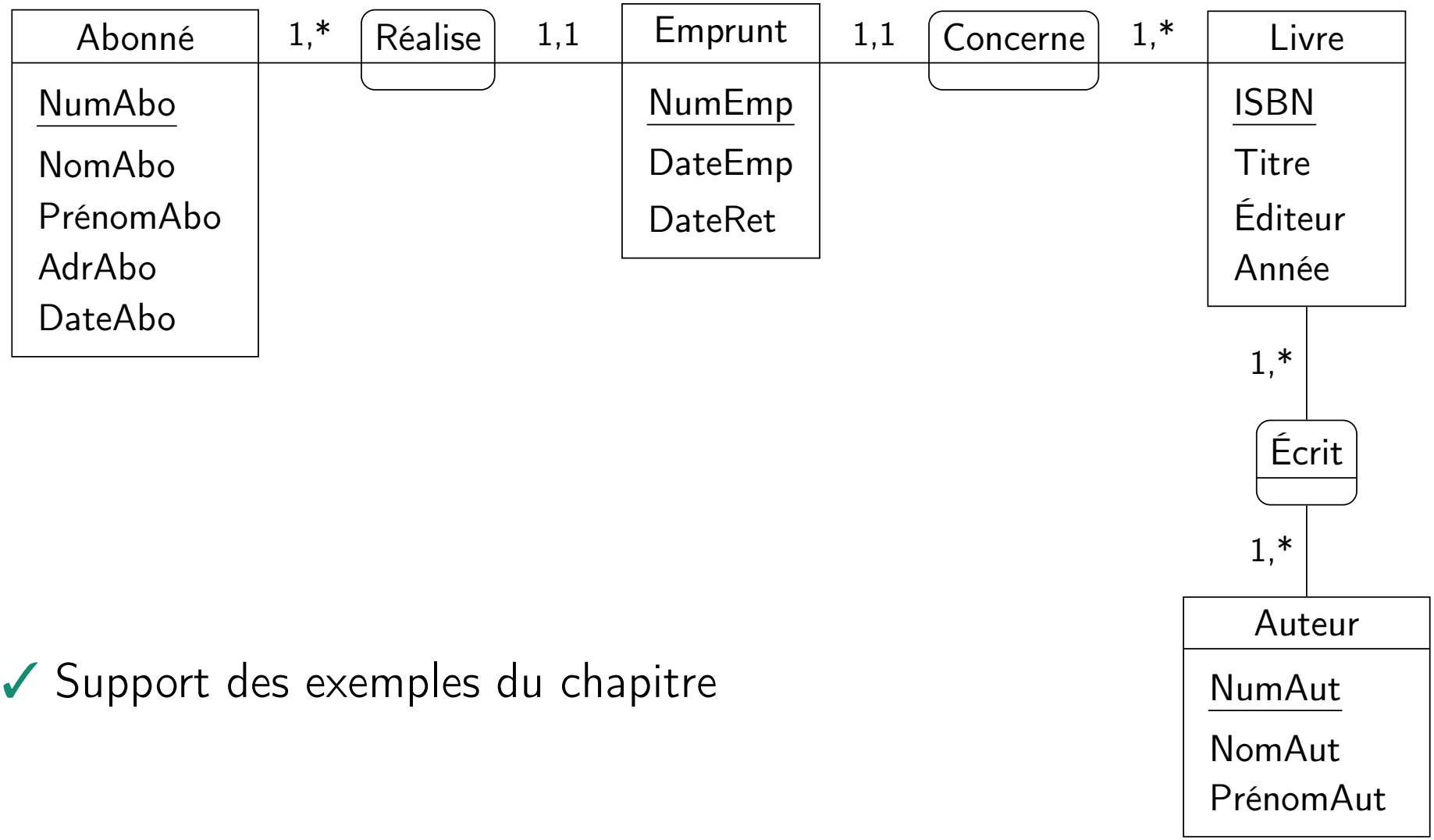
- Chaque **abonné** a un *numéro d'abonné* unique, un *nom*, un *prénom*, une *adresse* et une *date d'abonnement*.
- Les **livres** ont tous un *numéro ISBN*, un *titre*, un *éditeur* et une *année de publication*.
- Les **auteurs** qui écrivent les **livres** sont identifiés par un *numéro d'auteur*, et on stocke leur *nom* et *prénom*.
- Lorsqu'un **abonné** réalise un **emprunt** d'un **livre**, on enregistre le *numéro* et la *date de l'emprunt*.
- Lorsqu'il le restitue, on mémorise la *date de retour*.

Dictionnaire de données

On le résume dans un tableau.

Libellé	Type	Description
NumAbo	entier	Numéro de l'abonné
NomAbo	car(20)	Nom de l'abonné
PrénomAbo	car(20)	Prénom de l'abonné
AdrAbo	car(80)	Adresse de l'abonné
DateAbo	date	Date de l'abonnement (AAAA-MM-JJ)
NumAut	entier	Numéro de l'auteur
NomAut	car(20)	Nom de l'auteur
PrénomAut	car(20)	Prénom de l'auteur
ISBN	car(13)	Code ISBN identifiant un livre
Titre	car(80)	Titre du livre
Editeur	car(20)	Nom de l'editeur
Année	entier	Année de publication
NumEmp	entier	Numéro d'emprunt
DateEmp	date	Date de l'emprunt d'un livre par un abonné
DateRet	date	Date de retour d'un livre emprunté par un abonné

Exemple BD Bibliothèque : schéma E-A



✓ Support des exemples du chapitre

Exemple BD Bibliothèque : schéma relationnel

- Schéma relationnel de la BD

ABONNÉ(NumAbo, NomAbo, PrénomAbo, AdrAbo, DateAbo)
LIVRE(ISBN, Titre, Éditeur, Année)
AUTEUR(NumAut, NomAut, PrénomAut)
ÉCRIT(ISBN, NumAut)
EMPRUNT(NumEmp, NumAbo, ISBN, DateEmp, DateRet)

Exemple BD Bibliothèque : tables

- Tables (relations).

LIVRE

ISBN	Titre	Éditeur	Année
9782212112818	Bases de Données	Eyrolles	1989
9782225805158	Le Langage C	Masson	1985
9782207257357	Fondation	Denoël	2006

AUTEUR

NumAut	NomAut	PrénomAut
1	Gardarin	Georges
2	Kernighan	Brian
3	Ritchie	Dennis
4	Asimov	Isaac

ÉCRIT

ISBN	NumAut
9782212112818	1
9782225805158	2
9782225805158	3
9782207257357	4

ABONNE

NumAbo	NomAbo	PrénomAbo	DateAbo
1	Dupont	Philippe	2008-06-18
2	Durand	Arthur	2009-01-02
3	Dupont	Charlie	2015-05-03
4	Ducros	Marie	2020-07-04
5	Vernier	Alain	2021-09-15

EMPRUNT

NumEmp	ISBN	NumAbo	DateEmp	DateRet
1	9782225805158	2	2021-09-06	2021-09-20
2	9782225805158	3	2021-09-25	2021-10-11
3	9782212112818	1	2021-10-28	2021-11-10
4	9782212112818	1	2021-11-08	NULL

Conventions de notations

- Mots-clés de SQL : caractères `COURIER` majuscules
- Paramètres des requêtes : caractères `courier` minuscules
- Paramètres optionnels : `[option]`
- Valeurs multiples possibles : $\texttt{valeur}_1 \,|\, \texttt{valeur}_2$
- Options multiples : $\texttt{[option}_1 \,|\, \texttt{option}_2\texttt{]}$

Modification du schéma

- On peut être amené à créer de nouvelles tables, à ajouter des attributs ou à en modifier la définition.
- La modification d'une table peut poser des problèmes si elle est **incompatible** avec le contenu existant.
  - Passer un attribut à `NOT NULL` implique que cet attribut a déjà des valeurs pour tous les n-uplets de la table.

Supression d'une table

- Supprimer une table.

  ```
  DROP TABLE [IF EXISTS] Table;
  ```

- Supprime la table `Table`.
- Cette action est irréversible.

Exemple

```
DROP TABLE Abonné;
```

Modification d'une table

- Renommer une table.

  ```
  ALTER TABLE Table RENAME TO NouveauNom;
  ```

- Renomme la table `Table` en `NouveauNom`.

Exemple

```
ALTER TABLE Abonné RENAME TO Usager;
```

Modification d'attributs

- Ajout d'un attribut.

  `ALTER TABLE Table ADD Attribut Type;`

- Ajout de l'attribut `Attribut` de type `Type` dans la table `Table`.

Exemple

```
ALTER TABLE Livre ADD Édition SMALLINT;
```

Modification d'attributs

- Renommage d'un attribut.

  ```
  ALTER TABLE Table RENAME Attribut TO NouveauNom;
  ```

- Renomme l'attribut `Attribut` de la table `Table` en `NouveauNom`.

Exemple

```
ALTER TABLE Livre RENAME Édition TO Genre;
```

Index

- Un index offre un chemin d'accès rapide aux lignes d'une table.
- Utile quand le nombre de lignes est très élevé.
- Un index est systématiquement créé sur la clé primaire de chaque table
  - afin de vérifier rapidement, au moment d'une insertion, que la clé n'existe pas déjà,
  - car les requêtes SQL, notamment celles qui impliquent plusieurs tables se basent (le plus souvent) sur les clés des tables pour effectuer les jointures.

Création d'index (1)

```
CREATE [UNIQUE] INDEX [IF NOT EXISTS] NomIndex
ON Table (attribut[, ..., attributi]);
```

Exemple

```
CREATE INDEX idxEditeur ON Livre (Éditeur);
```

Cet index permettra d'exécuter très rapidement des requêtes SQL ayant comme critère de recherche l'éditeur d'un livre.

Remarque

Un index peut être crée sur un attribut non unique.

Création d'index (2)

```
CREATE [UNIQUE] INDEX [IF NOT EXISTS] NomIndex
ON Table (attribut[, ..., attributi]);
```

Exemple

```
CREATE INDEX idxNom ON Auteur (NomAut, PrénomAut);
```

Cet index permettra d'exécuter très rapidement des requêtes SQL ayant comme critère de recherche :

- le nom de l'auteur
- le nom et le prénom de l'auteur

Remarque

Un index peut être crée sur un ensemble d'attributs non uniques.

Création d'index (3)

- La clause `UNIQUE` indique que l'ensemble d'attributs doit être unique (mais peut être `NULL`).
- Toute insertion ultérieure violant la contrainte d'unicité échouera.

Exemple

```
CREATE UNIQUE INDEX idxNom ON Auteur (NomAut, PrénomAut);
```

Il ne peut pas y avoir d'homonymes dans la table `Auteur`.

Remarque

Une index `UNIQUE` est plus efficace qu'un index non unique.

Remarque

Il ne faut pas créer des index à tort et à travers, car cela ralenti les insertions et suppressions. À chaque fois, il faut mettre à jour tous les index portant sur la table.

Suppression d'index

- Suppression d'un index.

  ```
  DROP INDEX [IF EXISTS] IndexNom;
  ```

- Supprime l'index `IndexNom`.

Exemple

```
DROP INDEX idxEditeur;
```

Vue

- Une requête SQL produit toujours une table.
- Dans la terminologie relationnelle, une **vue** est un point de vue particulier sur la base, sous la forme d'une table calculée.
- On peut interroger des vues comme des tables.
  - Une vue induit un stockage négligeable puisqu'elle n'existe pas physiquement,
  - Une vue permet d'obtenir une représentation différente des tables sur lesquelles elle est basée,
  - Une vue permet de stocker le résultat d'une requête.
- Le contenu d'une vue est dynamique : il change si les données contenu dans les tables qu'elle référence sont modifiées.

Création d'une vue

```
CREATE VIEW [IF NOT EXISTS] NomVue
  [(attribut_1, ..., attribut_i)]
AS SELECT ... ;
```

Exemple

```
CREATE VIEW LivreAnciens AS
  SELECT * FROM Livre
  WHERE Année < 1900;
```

Vue des livres écrits avant le 20^{e} siècle.

Création d'une vue

```
CREATE VIEW [IF NOT EXISTS] NomVue
  [(attribut1, ..., attributi)]
AS SELECT ... ;
```

Exemple

```
CREATE VIEW LivreNbAuteur AS
  SELECT ISBN, COUNT(*) AS nbAuteurs
  FROM Livre, Auteur, Écrit
  WHERE Livre.ISBN = Écrit.ISBN
  AND Auteur.NumAut = Écrit.NumAut
  GROUP BY Livre.ISBN;
```

Vue des livres avec leur ISBN et le nombre d'auteurs.

Création d'une vue

```
CREATE VIEW [IF NOT EXISTS] NomVue
  [(attribut_1, ..., attribut_i)]
AS SELECT ... ;
```

Remarque

Il est possible de créer une **table** au lieu d'une **vue**.

```
CREATE TABLE [IF NOT EXISTS] NomTable
  [(attribut_1, ..., attribut_i)]
AS SELECT ... ;
```

Cependant, la table ainsi créée :

- est statique : son contenu n'évoluera pas
- induit une duplication des données (stockage, risque d'incohérences lors de modifications ultérieures . . .)

Création d'une vue

- Les vues donnent une représentation dénormalisée de la base, en regroupant des informations par des jointures.

Exemple

```
CREATE VIEW LivreAuteur (ISBN, Titre, Année, Nom, Prénom) AS
  SELECT Livre.ISBN, Titre, Année, NomAut, PrénomAut
  FROM Livre, Auteur, Écrit
  WHERE Livre.ISBN = Écrit.ISBN
  AND Auteur.NumAut = Écrit.NumAut;
```

Vue des livres avec les auteurs.

Utilisation d'une vue

- Utilisation des vues dans des requêtes.

Exemple

```
SELECT Nom, Prénom
FROM LivreAuteur
WHERE Année < 1900;
```

Suppression d'une vue

- Suppression d'une vue.

  ```
  DROP VIEW [IF EXISTS] NomVue;
  ```

- Supprime la vue `NomVue`.

Exemple

```
DROP VIEW LivreAnciens;
```

Exercice sur les vues

- Créer une vue des noms et prénoms des abonnés qui n'ont pas encore rendu un livre emprunté
- Créer une vue pour chaque abonné avec le nombre d'emprunts réalisés
- Créer une vue contenant pour chaque livre emprunté le nombre de fois qu'il a été emprunté (trié par ordre décroissant)
- Créer une vue des livres disponibles (les livres qui n'ont pas un emprunt en cours)

Trigger

Un déclencheur (**trigger**) est une procédure stockée qui se déclenche automatiquement sur certains événements.

Avantages

- Gestion des redondances
- Enregistrement automatique de certains événements
- Gestion de contraintes complexes
- Gestion de contraintes liées à l'environnement d'exécution

Inconvénient

Les triggers s'exécutent de manière cachée, et peuvent mener à des cycles sans fin.

Trigger par l'exemple

- Supposons qu'il y ait l'attribut `NbJours` dans la table `Emprunt`, qui correspond aux nombre de jours de l'emprunt
- Lors de la modification de la date de retour, ce nombre doit être calculé
- On crée donc un trigger pour mettre à jour cette valeur

Création d'un trigger

```
CREATE TRIGGER [IF NOT EXISTS] NomTrigger
[BEFORE|AFTER|INSTEAD OF] [INSERT|DELETE|UPDATE OF] ON Table
[FOR EACH ROW] [WHEN condition]
BEGIN
  ...
END;
```

- Un trigger est déclenché avant, après ou à la place d'une insertion, suppression ou modification
- La condition est testée, et si elle n'est pas satisfaite, l'exécution du trigger s'arrête
- L'action est effectuée à l'aide d'une requête (ou d'un langage procédural)
- Un trigger peut manipuler simultanément les valeurs ancienne et nouvelle du n-uplet modifié.

Création d'un trigger

```
CREATE TRIGGER [IF NOT EXISTS] NomTrigger
[BEFORE|AFTER|INSTEAD OF] [INSERT|DELETE|UPDATE OF] ON Table
[FOR EACH ROW] [WHEN condition]
BEGIN
  ...
END;
```

Exemple

```
CREATE TRIGGER MajJours
AFTER UPDATE ON Emprunt
FOR EACH ROW
WHEN (new.DateRet <> old.DateRet)
BEGIN
  UPDATE Emprunt SET
  NbJours = julianday(new.DateRet) - julianday(new.DateEmp)
  WHERE NumEmp = new.NumEmp;
END;
```

Suppression d'un trigger

- Suppression d'un trigger.

  ```
  DROP TRIGGER [IF EXISTS] NomTrigger;
  ```

- Supprime le trigger `NomTrigger`.

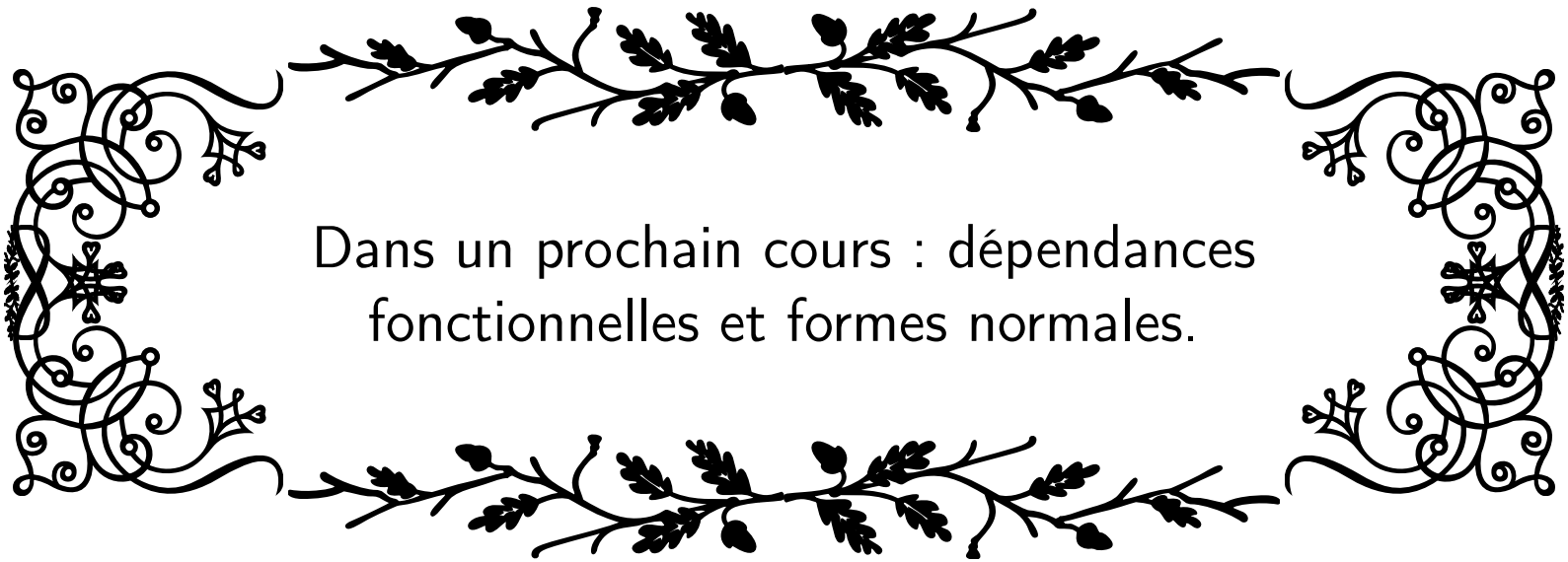
Exemple

```
DROP TRIGGER MajJours;
```

Exercice Bilan

1. Ajouter l'attribut `NbJours` à la table `Emprunt`
2. Mettre à jour l'attribut `NbJours`
3. Ajouter l'attribut `NbLivres` à la table `Auteur`
4. Créer une vue pour chaque auteur avec son nombre de livres
5. Mettre à jour la table `Auteur` avec le nombre de livres
6. Créer un trigger qui met à jour le nombre de livres d'un auteur lors de l'ajout d'un n-uplet dans la table `Écrit`
7. Créer un trigger qui met à jour le nombre de livres d'un auteur lors de la suppression d'un n-uplet dans la table `Écrit`
8. Créer un trigger qui met à jour le nombre de livres d'un auteur lors de la modification d'un n-uplet dans la table `Écrit`

À suivre


Dans un prochain cours : dépendances
fonctionnelles et formes normales.