Partie 2

Primary Key

Cours

Une **clé primaire (primary key)** est une colonne (ou une combinaison de colonnes) d'une table de base de données relationnelle désignée pour **identifier de manière unique chaque enregistrement de la table**.

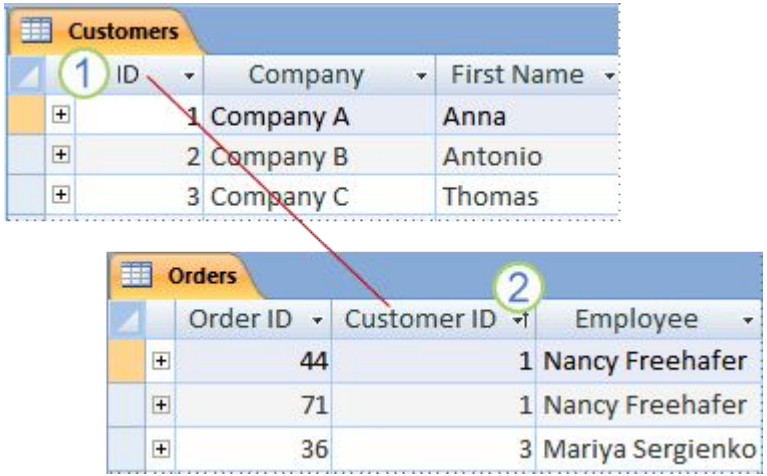
Les principales caractéristiques d'une clé primaire sont les suivantes :

- Elle doit contenir une valeur unique pour chaque ligne de données.
- Elle ne peut pas contenir de valeurs nulles.
- Une table ne peut pas avoir plus d'une clé primaire.

Une clé primaire peut être un colonne déjà présente dans le modèle de donnés qui respecte toutes les contraintes (ex: numéro de sécurité sociale) ou alors une clé primaire peut être une colonne supplémentaire créé (ex: customer_id).

Foreign Key

Une **clé étrangère (foreign key)** est une colonne ou un groupe de colonnes dans une table de base de données relationnelle qui fournit un lien entre les données de deux tables. Elle agit comme une référence croisée entre les tables car elle fait référence à **la clé primaire d'une autre table**, établissant ainsi un lien entre elles.



1 - primary key

2 - foreign key

Data Manipulation Language

Qu’est ce que CRUD ?

En termes simples, le terme CRUD résume les fonctions dont les utilisateurs ont besoin pour créer et gérer des données.

- **C**reate - Créer (ajouter, insérer)
- **R**ead - Lire (extraire, lister, consulter, interroger, rechercher)
- **U**pdate - Mettre à jour (modifier, éditer)
- **D**elete - Supprimer (effacer)

INSERT INTO

L'insertion de données dans une table s'effectue à l'aide de la commande INSERT INTO.

Insérer une ligne en spécifiant toutes les colonnes

```
INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...)
```

Cette syntaxe possède les avantages et inconvénients suivants :

- On est obligé de remplir toutes les données, tout en respectant l'ordre des colonnes
- Il n'y a pas le nom de colonne, donc les fautes de frappe sont limitées. Par ailleurs, les colonnes peuvent être renommées sans avoir à changer la requête
- L'ordre des colonnes doit rester identique sinon certaines valeurs prennent le risque d'être complétée dans la mauvaise colonne

INSERT INTO

Insérer une ligne en spécifiant seulement les colonnes souhaitées

```
INSERT INTO table (nom_colonne_1, nom_colonne_2, …)

 VALUES ('valeur 1', 'valeur 2', ...);
```

Ici il faut indiquer le nom des colonnes avant “VALUES”.

A noter : il est possible de ne pas renseigner toutes les colonnes. De plus, l’ordre des colonnes n’est pas important.

INSERT INTO

Insertion de plusieurs lignes à la fois

```
INSERT INTO client (prenom, nom, ville, age)
 VALUES
 ('Rébecca', 'Armand', 'Saint-Didier-des-Bois', 24),
 ('Aimée', 'Hebert', 'Marigny-le-Châtel', 36),
 ('Marielle', 'Ribeiro', 'Maillères', 27),
 ('Hilaire', 'Savary', 'Conie-Molitard', 58);
```

Lorsque le champ à remplir est de type VARCHAR ou TEXT il faut indiquer le texte entre guillemet simple. En revanche, lorsque la colonne est un numérique tel que INT ou BIGINT il n'y a pas besoin d'utiliser de guillemet, il suffit juste d'indiquer le nombre.

INSERT INTO

Exemples

```
INSERT INTO `products` VALUES (21,'coussin', 58, 18.99);
```

```
INSERT INTO `products` (product_id, name, quantity_in_stock, unit_price)
    VALUES (22,'couette', 43, 35.99);
```

INSERT INTO

Exercices

Utilisez le lien suivant et ajoutez 5 clients dans la table customers

https://fr.fakenamegenerator.com/gen-random-us-fr.php

UPDATE

La commande UPDATE permet d'effectuer des modifications sur des lignes existantes. Très souvent cette commande est utilisée avec WHERE pour spécifier sur quelles lignes doivent porter la ou les modifications.

Syntaxe

```
UPDATE table

SET nom_colonne_1 = 'nouvelle valeur'

WHERE condition
```

```
UPDATE table

SET colonne_1 = 'valeur 1', colonne_2 = 'valeur 2', colonne_3 = 'valeur 3'

WHERE condition
```

UPDATE

Exemples

```
UPDATE products

SET name = 'coussin douillet'

WHERE product_id = 21;
```

```
UPDATE customers

SET last_name = 'Bonjour';
```

```
SET SQL_SAFE_UPDATES = 0;
```

UPDATE

Exercices

Notre premier client à déménagé, veuillez changer son adresse à la suivante:

38 rue de Champs 69001 Lyon

Notre dernier client à perdu son téléphone, remplacez sa valeur par NULL

DELETE

La commande DELETE en SQL permet de supprimer des lignes dans une table. En utilisant cette commande associé à WHERE il est possible de sélectionner les lignes concernées qui seront supprimées.

Syntaxe

```
DELETE FROM `table`

WHERE condition
```

Attention : s'il n'y a pas de condition WHERE alors **toutes** les lignes seront supprimées et la table sera alors vide.

DELETE VS TRUNCATE

Pour supprimer toutes les lignes d'une table il convient d'utiliser la commande DELETE ou TRUNCATE sans utiliser de clause conditionnelle.

Syntaxe

```
DELETE FROM `utilisateur`
```

```
TRUNCATE TABLE `utilisateur`
```

La différence majeure étant que la commande TRUNCATE va ré-initialiser l'auto-incrémente s'il y en a un. Tandis que la commande DELETE ne ré-initialise pas l'auto-incrément.

DELETE

Exemples

```
DELETE FROM products

WHERE product_id = 21;
```

DELETE

Exercices

Supprimez les 5 derniers clients.

Data Definition Language

Les contraintes

Les contraintes sont utilisées pour spécifier les règles concernant les données dans la table. Elles peuvent être appliquées à un ou plusieurs champs d'une table SQL pendant la création de la table ou après sa création à l'aide de la commande ALTER TABLE. Les contraintes sont les suivantes :

- NOT NULL - Empêche l'insertion d'une valeur NULL dans une colonne.
- CHECK - Vérifie que toutes les valeurs d'un champ satisfont à une condition.
- DEFAULT - Attribue automatiquement une valeur par défaut si aucune valeur n'a été spécifiée pour le champ.
- UNIQUE - Assure l'insertion de valeurs uniques dans le champ.
- INDEX/KEY - Permet d'indexer un champ pour accélérer la recherche d'enregistrements.
- PRIMARY KEY - Identifie de manière unique chaque enregistrement d'une table.
- FOREIGN KEY - Assure l'intégrité référentielle d'un enregistrement dans une autre table.

Les contraintes

Exemples

```
CREATE TABLE `order_statuses` (
  `order_status_id` tinyint(4) NOT NULL,
  `status` varchar(50) NOT NULL,
  PRIMARY KEY (`order_status_id`)
);
```

Les contraintes

Exercices

Y a t'il des colonnes n'ayant pas la contrainte not null ?

Quelles sont les différentes valeurs par défaut indiqués (toutes tables et colonnes confondus) ?

A quoi correspond la commande ON UPDATE ?

A quoi correspond la commande ON DELETE ?

A quoi correspond la commande CASCADE ? Peut-on avoir d'autres options que CASCADE ?

CREATE TABLE

La commande CREATE TABLE permet de créer une table en SQL. La création d'une table sert à définir les colonnes et le type de données qui seront contenus dans chacun des colonne (entier, chaîne de caractères, date, valeur binaire ...).

```
CREATE TABLE nom_de_la_table
(
    colonne1 type_donnees contraintes,
    colonne2 type_donnees contraintes,
    colonne3 type_donnees contraintes,
    colonne4 type_donnees contraintes
)
```

CREATE TABLE

Exemples

```
CREATE TABLE `products` (
  `product_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `quantity_in_stock` int(11) NOT NULL,
  `unit_price` decimal(4,2) NOT NULL,
  PRIMARY KEY (`product_id`)
);
```

CREATE TABLE

Exercices

Recréez la table suivante que vous appellerez clients

customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thomas Ave	Orchard Park	NY	14127
2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	NY	11553
5	Charolette	Rice	(916) 381-6003	charlotte.rice@msn.com	107 River Dr.	Sacramento	CA	95820

ALTER TABLE

La commande ALTER TABLE en SQL permet de modifier une table existante par exemple pour ajouter une colonne, supprimer une colonne ou modifier le type d'une colonne existante.

Ajouter un colonne

```
ALTER TABLE nom_table

ADD nom_colonne type_donnees
```

Supprimer une colonne

```
ALTER TABLE nom_table

DROP nom_colonne
```

ALTER TABLE

Modifier une colonne

```
ALTER TABLE nom_table

MODIFY nom_colonne type_donnees
```

Renommer une colonne

```
ALTER TABLE nom_table

CHANGE colonne_ancien_nom colonne_nouveau_nom type_donnees
```

ALTER TABLE

Exemples

```
ALTER TABLE customers
ADD country VARCHAR(255)
```

```
SET SQL_SAFE_UPDATES = 0;
UPDATE customers
SET country = 'France';
```

```
ALTER TABLE customers
CHANGE country pays VARCHAR(255);
```

```
ALTER TABLE customers
DROP pays;
```

ALTER TABLE

Exercices

Ajoutez une colonne country à votre table clients et remplissez votre colonne avec les valeurs appropriés

DROP TABLE

La commande DROP TABLE en SQL permet de supprimer définitivement une table d'une base de données. Cela supprime en même temps les éventuels index, trigger, contraintes et permissions associées à cette table.

Syntaxe

```
DROP TABLE nom_table
```

Attention : il faut utiliser cette commande avec attention car une fois supprimée, les données sont perdues. Avant de l'utiliser sur une base importante il peut être judicieux d'effectuer un backup (une sauvegarde) pour éviter les mauvaises surprises.

DROP TABLE

Exercices

Supprimez la table clients

CREATE - DROP - USE DATABASES

Créer une base de données

```
CREATE DATABASE ma_base
```

Supprimer une base de données

```
DROP DATABASE IF EXISTS ma_base
```

Utiliser une base de données

```
USE ma_base
```

CREATE - DROP - USE DATABASES

Exemples

```
DROP DATABASE IF EXISTS `sql_store`;

CREATE DATABASE `sql_store`;

USE `sql_store`;
```

CREATE - DROP - USE DATABASES

Exercices globaux

Nous avons ouvert un nouveau magasin.

Créez une nouvelle base de données appelée new_store

Dans cette base de donnés créez une table “clients” avec les mêmes champs que la table customers de “sql_store”.

Nous avons 5 nouveaux clients dans notre nouveau magasin. Ajoutez les à la table “clients”.

Nos 5 clients les plus fidèles de notre 1 er magasin sont également allés visiter notre nouveau magasin. Ajoutez les à la table “clients”.

Data Query Language

UNION

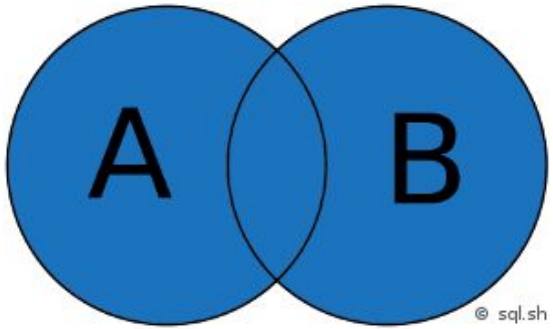
La commande UNION permet de mettre bout-à-bout les résultats de plusieurs requêtes utilisant elles-même la commande SELECT. C'est donc une commande qui permet de concaténer les résultats de 2 requêtes ou plus. Pour l'utiliser il est nécessaire que chacune des requêtes à concaténer retournes le même nombre de colonnes, avec les mêmes types de données et dans le même ordre

Syntaxe

```
SELECT * FROM table1

UNION

SELECT * FROM table2
```



UNION

Exercices

Affichez la table de tous nos clients (tous magasins confondus)

Database design

Quelles sont les étapes du database design ?

Il n'y a pas vraiment de méthodologies de design qui prédomine mais globalement vous pouvez appliquer la logique suivante:

1. **Design conceptuel de la base de données:** comprendre les besoins et les règles de l'utilisateur et de l'entreprise, comprendre l'objectif de la base de données,...
2. **Design logique de la base de données:** modélisation de la base de données, optimisation des tables, champs, contraintes, ...
3. **Conception physique de la base de données:** implémentation de la base de données

Pourquoi est ce que le design de BDD est important ?

Features	Bon design de BDD	Mauvais design de BDD
CRUD	Vous pouvez trouver les données dont vous avez besoin rapidement et facilement. La base de données empêche les modifications incohérentes.	Vous trouvez les données dont vous avez besoin très lentement ou pas du tout. Vous pouvez saisir des données incohérentes ou modifier et supprimer des données pour rendre le résultat incohérent. (Vos produits sont expédiés à la mauvaise adresse ou à la mauvaise personne).
Récupération	Vous pouvez trouver les données correctes rapidement et facilement	Vous ne pouvez pas trouver rapidement les données dont vous avez besoin. (Votre client attend en attente pendant 45 minutes pour obtenir un simple solde de compte).
Cohérence	Toutes les parties de la base de données s'accordent sur des faits communs.	Différents éléments d'information contiennent des données contradictoires. (Le compte d'Alice à été débité de 100€ mais le compte de Bob n'a pas été crédité...).
Validité	Les champs contiennent des données valides.	Des champs contiennent du charabia. Le code postal de votre entreprise a la valeur "ABCDE".
Correction des erreurs	Il est facile de mettre à jour les données incorrectes.	Les changements simples et à grande échelle ne se produisent jamais. (Des milliers de factures de vos clients vous sont retournées parce que leur code postal a changé et que la base de données n'a pas été mise à jour).
Vitesse	Vous pouvez trouver rapidement les clients par nom, numéro de compte ou numéro de téléphone	Vous ne pouvez trouver la fiche d'un client que s'il connaît son numéro de compte à 37 chiffres. La recherche par nom prend une demi-heure.

Normalization

Cours

La **normalisation** est un processus de réarrangement de la base de données pour la mettre sous une forme standard (normale) qui empêche les anomalies le plus possibles.

Des exemples d'anomalies peuvent être :

- Votre BDD peut contenir beaucoup de données dupliquées. Cela entraîne non seulement un gaspillage d'espace, mais aussi une mise à jour fastidieuse de toutes ces valeurs dupliquées.
- Votre BDD a des anomalies de mise à jour si votre BDD devient inconsistante après un changement de valeur.
- Votre BDD a des anomalies de suppression si vous perdez des informations dont vous aurez besoin plus tard en supprimant un enregistrement.
- Votre BDD a des anomalies d'insertion si vous ne pouvez pas ajouter des enregistrements dont vous avez besoin sans que cela ne transgresse des contraintes de votre BDD
- Votre BDD peut associer de manière incorrecte deux éléments de données sans rapport entre eux, de sorte que vous ne pouvez pas en supprimer un sans supprimer l'autre.
- Etc ...

1ère forme de Normalisation

Cours

Les règles que votre BDD doit respecter pour être en première forme normale (1NF) sont:

1. Chaque colonne doit avoir un nom unique.
2. L'ordre des lignes et des colonnes n'a pas d'importance.
3. Chaque colonne doit avoir un seul type de données.
4. Deux lignes ne peuvent pas contenir des valeurs identiques == Chaque table a une clé primaire.
5. Chaque colonne doit contenir une seule valeur.
6. Les colonnes ne peuvent pas contenir de groupes répétitifs.

1ère forme de Normalisation

Exemples

Le tableau suivant contient des informations sur les vols des compagnies aériennes. Il contient des données sur un groupe de deux personnes voyageant de Denver à Phoenix et un groupe de trois personnes voyageant de San Diego à Los Angeles. Les deux premières colonnes indiquent les villes de départ et de destination. La dernière colonne indique les villes de correspondance (le cas échéant) ou le nombre de correspondances. Les lignes sont ordonnées de manière à ce que les passagers premium se trouvent en haut, dans ce cas dans les trois premières lignes

City	City	Connections
DEN	PHX	1
SAN	LAX	JFK, SEA, TPA
SAN	LAX	JFK, SEA, TPA
DEN	PHX	1
SAN	LAX	JFK, SEA, TPA

1ère forme de Normalisation

Exemples

2. Chaque colonne doit avoir un nom unique.

StartCity	DestinationCity	Connections
DEN	PHX	1
SAN	LAX	JFK, SEA, TPA
SAN	LAX	JFK, SEA, TPA
DEN	PHX	1
SAN	LAX	JFK, SEA, TPA

1ère forme de Normalisation

Exemples

2. L'ordre des lignes et des colonnes n'a pas d'importance

StartCity	DestinationCity	Connections	Priority
DEN	PHX	1	1
SAN	LAX	JFK, SEA, TPA	1
SAN	LAX	JFK, SEA, TPA	1
DEN	PHX	1	2
SAN	LAX	JFK, SEA, TPA	2

1ère forme de Normalisation

3. Chaque colonne doit avoir un seul type de données.

Exemples

StartCity	DestinationCity	Connections	Priority
DEN	PHX	LON	1
SAN	LAX	JFK, SEA, TPA	1
SAN	LAX	JFK, SEA, TPA	1
DEN	PHX	LON	2
SAN	LAX	JFK, SEA, TPA	2

1ère forme de Normalisation

4. Deux lignes ne peuvent pas contenir des valeurs identiques == Chaque table a une clé primaire.

Exemples

StartCity	DestinationCity	Connections	Priority	CustomerId	Date
DEN	PHX	LON	1	4637	4/1/10
SAN	LAX	JFK, SEA, TPA	1	12878	6/21/10
SAN	LAX	JFK, SEA, TPA	1	2871	6/21/10
DEN	PHX	LON	2	28718	4/1/10
SAN	LAX	JFK, SEA, TPA	2	9287	6/21/10

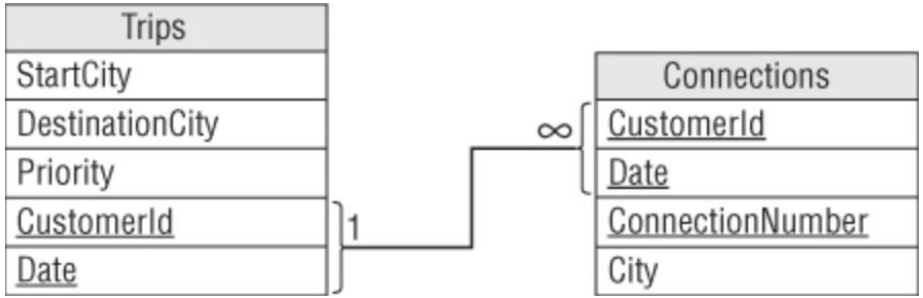
1ère forme de Normalisation

5. Chaque colonne doit contenir une seule valeur.
6. Les colonnes ne peuvent pas contenir de groupes répétitifs.

Exemples

StartCity	DestinationCity	Prioriy	CustomerId	Date
DEN	PHX	1	4637	4/1/10
SAN	LAX	1	12878	6/21/10
SAN	LAX	1	2871	6/21/10
DEN	PHX	2	28718	4/1/10
SAN	LAX	2	9287	6/21/10

CustomerId	Date	ConnectionNumber	City
4637	4/1/10	1	LON
12878	6/21/10	1	JFK
12878	6/21/10	2	SEA
12878	6/21/10	3	TPA
2871	6/21/10	1	JFK
2871	6/21/10	2	SEA
2871	6/21/10	3	TPA
28718	4/1/10	1	LON
9287	6/21/10	1	JFK
9287	6/21/10	2	SEA
9287	6/21/10	3	TPA



TECH & CODE FACTORY

2ème forme de Normalisation

Cours

Les règles que votre BDD doit respecter pour être en deuxième forme normale (2NF) sont:

1. Votre BDD est en 1NF.
2. Tous les champs non clés dépendent de **tous** les champs clés.

2ème forme de Normalisation

Exemples

Nous avons la table suivante qui recense le planning de combats contre des crocodiles

Time	Wrestler	Class	Rank
1:30	Annette Cart	Pro	3
1:30	Ben Jones	Pro	2
2:00	Sydney Dart	Amateur	1
2:15	Ben Jones	Pro	2
2:30	Annette Cart	Pro	3
3:30	Sydney Dart	Amateur	1
3:30	Mike Acosta	Amateur	6
3:45	Annette Cart	Pro	3

La table est elle en 1NF ?

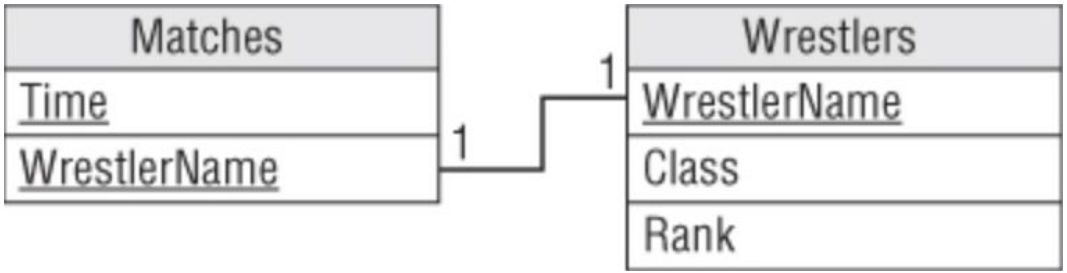
2ème forme de Normalisation

Exemples

2. Tous les champs non clés dépendent de tous les champs clés.

Matches	
Time	WrestlerName
1:30	Annette Cart
2:30	Annette Cart
3:45	Annette Cart
1:30	Ben Jones
2:15	Ben Jones
3:30	Mike Acosta
2:00	Sydney Dart
3:30	Sydney Dart

Wrestlers		
WrestlerName	Class	Rank
Annette Cart	Pro	3
Ben Jones	Pro	2
Mike Acosta	Amateur	6
Sydney Dart	Amateur	1



3ème forme de Normalisation

Cours

Les règles que votre BDD doit respecter pour être en troisième forme normale (3NF) sont:

1. Votre BDD est en 3NF.
2. Votre BDD ne contient aucune dépendance transitive.

On parle de dépendance transitive lorsque la valeur d'un champ non clé dépend de la valeur d'un autre champ non clé.

3ème forme de Normalisation

Exemples

Nous avons la table suivante qui recense les livres préférés de vos amis.

Person	Title	Author	Pages	Year
Amy	Support Your Local Wizard	Duane, Diane	473	1990
Becky	Three to Dorsai!	Dickson, Gordon	532	1975
Jon	Chronicles of the Black Company	Cook, Glen	704	2007
Ken	Three to Dorsai!	Dickson, Gordon	532	1975
Wendy	Support Your Local Wizard	Duane, Diane	473	1990

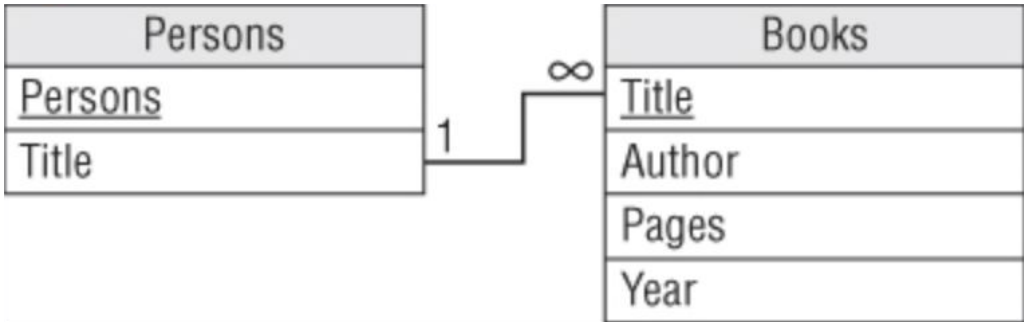
3ème forme de Normalisation

Exemples

2. Votre BDD ne contient aucune dépendance transitive.

Persons	
Person	Title
Jon	Chronicles of the Black Company
Amy	Support Your Local Wizard
Wendy	Support Your Local Wizard
Becky	Three to Dorsai!
Ken	Three to Dorsai!

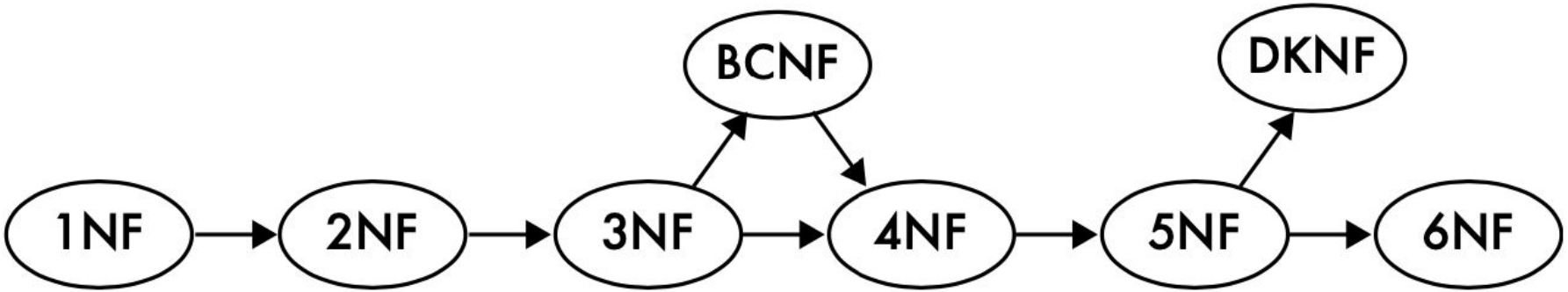
Books			
Title	Author	Pages	Year
Chronicles of the Black Company	Cook, Glen	704	2007
Support Your Local Wizard	Duane, Diane	473	1990
Three to Dorsai!	Dickson, Gordon	532	1975



TECH & CODE FACTORY

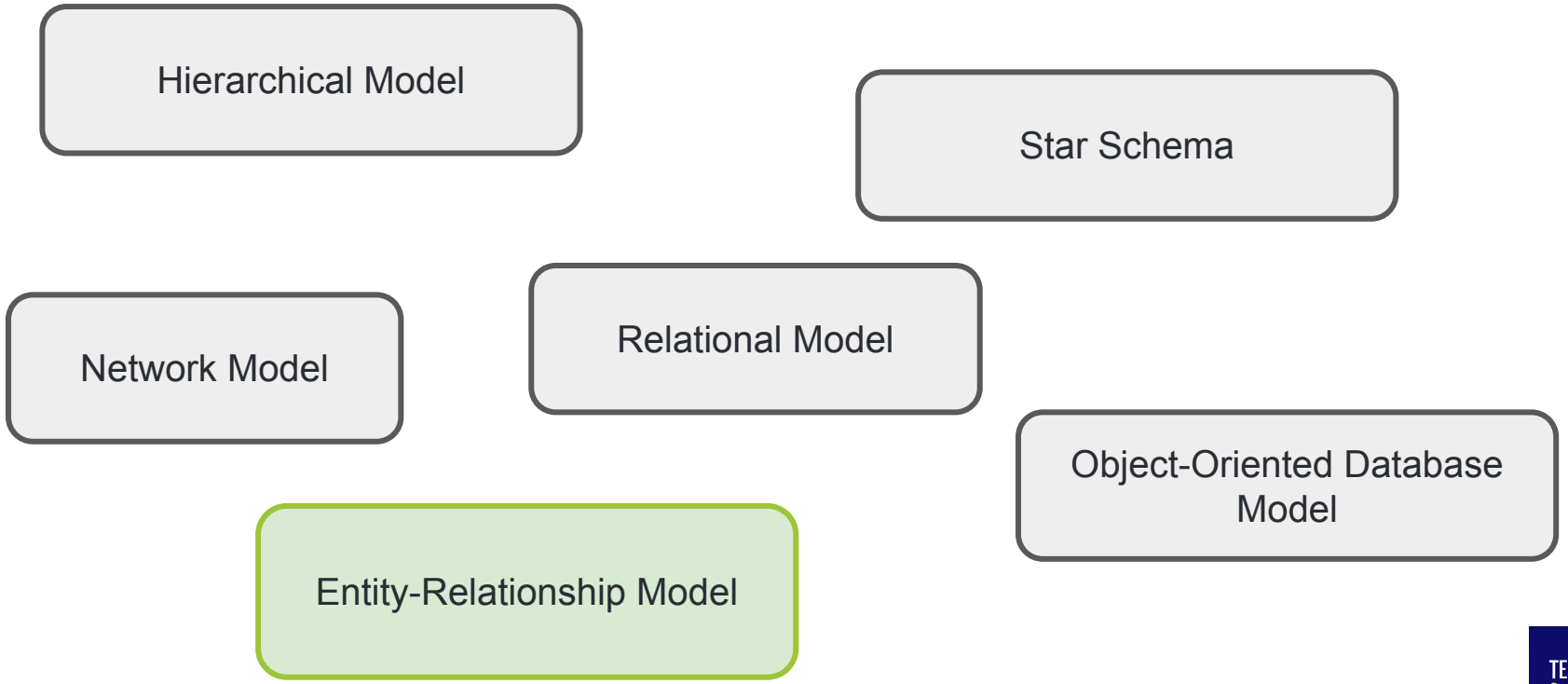
Autres formes de normalisation

De nombreux concepteurs de BDD arrêtent la normalisation de la BDD à 3NF parce que c'est celle qui offre le meilleur rapport investissement/retour. En effet il est assez facile de convertir une base de données en 3NF et ce niveau de normalisation permet d'éviter les anomalies de données les plus courantes.



Database Models

Cours



TECH & CODE FACTORY

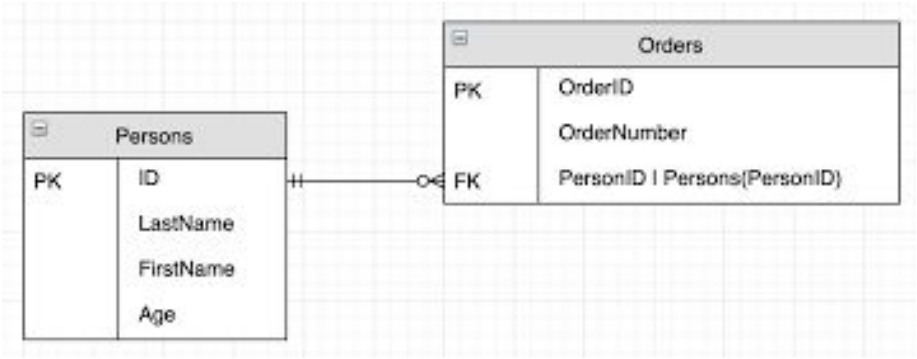
Que sont les Entity Relationship Diagrams (ERD) ?

Cours

Un diagramme entité-relation (ERD) est un schéma de la structure de notre BDD. Un ERD montre les entités (tables) d'une base de données et les relations entre les tables de cette base.

Il y a trois éléments de base dans les diagrammes ER :

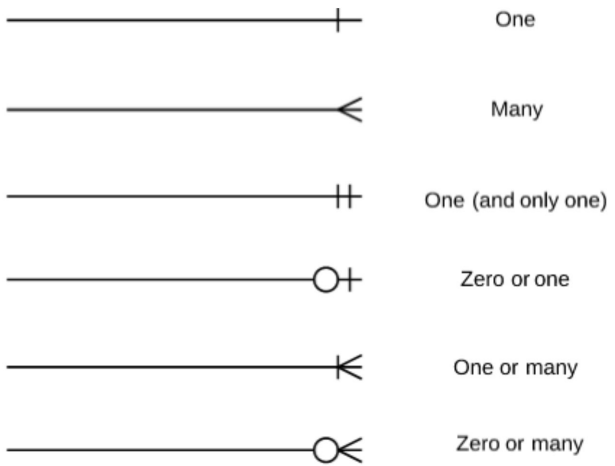
- Les entités sont les "choses" pour lesquelles nous voulons stocker des informations. Une entité est une personne, un lieu, une chose ou un événement.
- Les attributs sont les données que nous voulons collecter pour une entité.
- Les relations décrivent les relations entre les entités.



Cardinality and ordinality

Cardinality refers to the maximum number of times an instance in one entity can relate to instances of another entity. Ordinality, on the other hand, is the minimum number of times an instance in one entity can be associated with an instance in the related entity.

Cardinality and ordinality are shown by the styling of a line and its endpoint, according to the chosen notation style.



Pourquoi est ce que les ERD sont importants ?

Les ERD fournissent un point de départ visuel pour la conception de la BDD qui peut également être utilisé pour aider à déterminer les exigences du système d'information dans l'ensemble d'une organisation.

Après le déploiement d'une base de données relationnelle, un ERD peut toujours servir de point de référence, si un débogage ou une réingénierie des processus s'avèrent nécessaires par la suite.

Entity Relationship Diagrams (ERD)

Exemples

Je vous invite à regarder les deux vidéos suivantes pour bien assimiler toutes les notions des entity relationships diagrams ainsi que pour voire des exemples mis en oeuvre

https://www.youtube.com/watch?v=QpdhBUYk7Kk

https://www.youtube.com/watch?v=-CuY5ADwn24

Database Schemas / Relational Model

Exercices

Construisez le schéma de la base de donnés sql_store

A vous de jouer !

Exercices globaux

A partir du brief client suivant construisez un ERD de la BDD qu'il faudrait construire.
Construisez cette base de données et ajoutez quelques enregistrements dans chaque table

Supposons que l'on vous donne les exigences suivantes pour une base de données simple pour la Ligue nationale de hockey (LNH) :
- la NHL a de nombreuses équipes,
- chaque équipe a un nom, une ville, un entraîneur, un capitaine et un ensemble de joueurs,
- chaque joueur n'appartient qu'à une seule équipe,
- chaque joueur a un nom, une position (comme ailier gauche ou gardien de but), un niveau de compétence, et un ensemble de blessures,
- le capitaine d'une équipe est également un joueur,
- un match est joué entre deux équipes (appelées équipe_hôte et équipe_invitée) et a une date (comme le 11 mai 2017) et un score (comme 4 à 2).