

Développement PHP

Partie 4 : Bases de données

DENIS LOEUILLET – IFA - 2017

Partie 4 : Bases de données

- Présentation des bases de données
- phpMyAdmin
- Lire les données
- Écrire des données
- Les fonctions SQL
- Les dates en SQL
- Les jointures entre tables

Partie 4 : Bases de données

Les jointures entre tables

1. Modélisation d'une relation
2. Clés primaires et étrangères
3. Qu'est-ce qu'une jointure ?
4. Les jointures internes
5. Les jointures externes
6. Résumé

Partie 4 : Bases de données

Les jointures entre tables

- MySQL permet de travailler avec plusieurs tables à la fois.
- Un des principaux intérêts d'une base de données /
 - pouvoir créer des relations entre les tables
 - pouvoir lier les tables entre elles.
- Pour le moment, nous n'avons travaillé que sur une seule table à la fois :
 - Dans la pratique :
 - ✓ plusieurs tables dans votre base
 - ✓ a plupart seront interconnectées.
 - ✓ Permet de :
 - ❖ mieux découper les informations
 - ❖ éviter des répétitions
 - ❖ Rendre vos données plus faciles à gérer.

Partie 4 : Bases de données

Les jointures entre tables

- Par exemple dans la table jeux_video :
 - on répète à chaque fois le nom du possesseur du jeu.
 - Si on souhaite enregistrer d'autres informations sur le possesseur :
 - ✓ Ça fait beaucoup d'informations redondantes
 - ✓ Il faut créer une autre table lier les deux.

Partie 4 : Bases de données

Les jointures entre tables

1. Modélisation d'une relation

- Si on veut enregistrer d'autres informations concernant les propriétaires des jeux :
 - Prénom, tél, adresse ...
 - Toutes ces informations devraient être dupliquées dans chaque entrée
 - Si une donnée change (tél par exemple) il faut effectuer la modification sur chaque entrée!!!
- ✓ Il faut éviter ces répétitions : une information ne doit être présente qu'une fois dans la base
- ✓ Solution :
 - ❖ Créer une autre table qui centralise les informations des propriétaires
 - ❖ Cette table pourrait ressembler à ceci

ID	prenom	nom	tel
1	Florent	Dugommier	01 44 77 21 33
2	Patrick	Lejeune	03 22 17 41 22
3	Michel	Doussand	04 11 78 02 00

Partie 4 : Bases de données

Les jointures entre tables : modélisation d'une relation

- Si on veut enregistrer d'autres information concernant les propriétaires des jeux :
 - ✓ Solution :
 - ❖ Il faut maintenant modifier la table jeux_video pour supprimer les répétitions des noms des propriétaires
 - On supprime les champs possesseur
 - On le remplace par un champs qui va faire référence au propriétaire dans la nouvelle tables des propriétaires :
 - On crée un champs id_propriétaire qui indique l'id du propriétaire dans la table des propriétaires

Partie 4 : Bases de données

Les jointures entre tables : modélisation d'une relation

ID	nom	ID_proprietaire	console	prix	nbre_joueurs_max	commentaires
1	Super Mario Bros	1	NES	4	1	Un jeu d'anthologie !
2	Sonic	2	Megadrive	2	1	Pour moi, le meilleur jeu au monde !
3	Zelda : ocarina of time	1	Nintendo 64	15	1	Un jeu grand, beau et complet comme on en voit rarement de nos jours
4	Mario Kart 64	1	Nintendo 64	25	4	Un excellent jeu de kart !
5	Super Smash Bros Melee	3	GameCube	55	4	Un jeu de baston délirant !

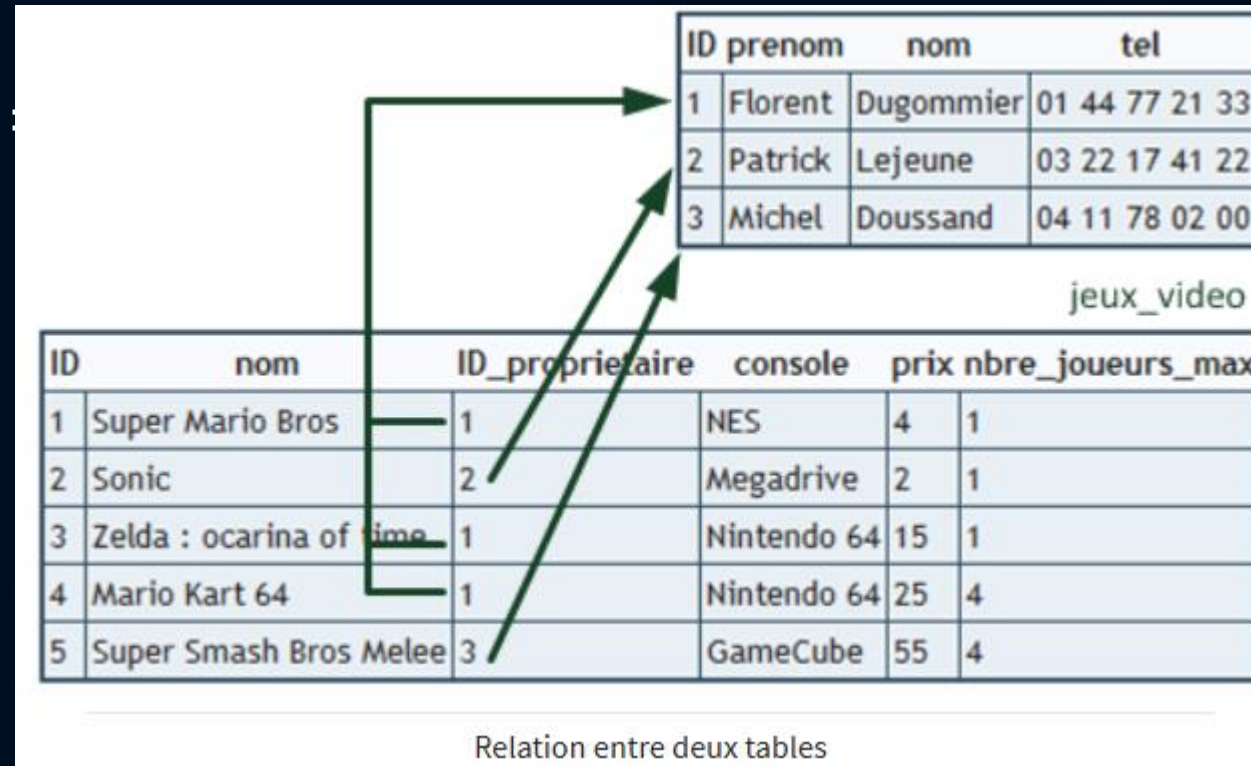
Partie 4 : Bases de données

Les jointures entre tables : modélisation d'une relation

- Si on veut enregistrer d'autres information concernant les propriétaires des jeux :

✓ Solution :

- ❖ Le nouveau champ ID_proprietaire :
 - type INT.
 - Il permet de faire référence à une entrée précise de la table propriétaires.
- ❖ Les 2 tables sont maintenant reliées à travers ces ID de propriétaires :



Partie 4 : Bases de données

Les jointures entre tables : modélisation d'une relation

- Si on veut enregistrer d'autres information concernant les propriétaires des jeux :
 - ✓ Solution :
 - ❖ MySQL ne fait aucune relation entre :
 - id de la table proprietaires
 - ID_proprietaire de la table jeux_video
 - ❖ Il va falloir lui expliquer cette relation dans une requête SQL :
 - on va faire ce qu'on appelle une jointure entre les deux tables.

Partie 4 : Bases de données

Les jointures entre tables

2. Clés primaires et clés étrangères

- Les clés sont liées aux index. Et tout comme NOT NULL et les index UNIQUE, les clés font partie de ce qu'on appelle les contraintes.
- Il existe deux types de clés :
 - les clés primaires : servent à identifier une ligne de manière unique
 - les clés étrangères :
 - ✓ permettent de gérer des relations entre plusieurs tables
 - ✓ garantissent la cohérence des données.

Partie 4 : Bases de données

Les jointures entre tables : clés primaires et étrangères

- Clés primaires
 - La clé primaire d'une table :
 - ✓ est une contrainte d'unicité : index UNIQUE
 - ✓ Composée d'une ou plusieurs colonnes
 - ✓ permet d'identifier de manière unique chaque ligne de la table : une clé primaire ne peut pas être NULL
 - Une seule clé primaire par table.
 - De même, une seule colonne peut être auto-incrémentée (la clé primaire en général).

Partie 4 : Bases de données

Les jointures entre tables : clés primaires et étrangères

- Clés étrangères
 - La clé étrangère représente un champ (ou des champs) qui pointe vers la clé primaire d'une autre table.
 - L'objectif de la clé étrangère est d'assurer l'intégrité référentielle des données :
 - ✓ En d'autres mots, seules les valeurs devant apparaître dans la base de données sont permises.

Partie 4 : Bases de données

Les jointures entre tables

3. Qu'est-ce qu'une jointure ?

- Nous avons donc maintenant deux tables :
 - jeux_video
 - proprietaires
- Les informations sont séparées dans des tables différentes :
 - Cela évite de dupliquer des informations
 - Cependant, lorsqu'on récupère la liste des jeux :
 - ✓ si on souhaite obtenir le nom du propriétaire :
 - ❖ il va falloir adapter la requête pour récupérer aussi les informations issues de la table proprietaires.
 - ❖ Pour cela, on doit faire ce qu'on appelle une jointure.

Partie 4 : Bases de données

Les jointures entre tables : qu'est-ce qu'une jointure

- Il existe plusieurs types de jointures, qui nous permettent de choisir exactement les données que l'on veut récupérer.
 - Voici les plus importantes :
 - ✓ les jointures internes : elles ne sélectionnent que les données qui ont une correspondance entre les deux tables
 - ✓ les jointures externes : elles sélectionnent toutes les données, même si certaines n'ont pas de correspondance dans l'autre table.
 - Il est important de bien comprendre la différence entre une jointure interne et une jointure externe.
 - Pour illustrer, imaginons que nous ayons une 4^e personne dans la table des propriétaires, un certain Romain Vipelli, qui ne possède aucun jeu

Partie 4 : Bases de données

Les jointures entre tables : qu'est-ce qu'une jointure

- Romain Vipelli est référencé dans la table propriétaires mais il n'apparaît nulle part dans la table jeux_video car il ne possède aucun jeu.
- Si vous récupérez les données des deux tables à l'aide :
 - d'une jointure interne :
 - ✓ Romain Vipelli n'apparaîtra pas dans les résultats de la requête.
 - ✓ La jointure interne force les données d'une table à avoir une correspondance dans l'autre ;
 - d'une jointure externe :
 - ✓ vous aurez toutes les données de la table des propriétaires :
 - ❖ même s'il n'y a pas de correspondance dans l'autre table des jeux vidéo ;
 - ❖ donc Romain Vipelli, qui pourtant ne possède aucun jeu vidéo, apparaîtra.

ID	prenom	nom	tel
1	Florent	Dugommier	01 44 77 21 33
2	Patrick	Lejeune	03 22 17 41 22
3	Michel	Doussand	04 11 78 02 00
4	Romain	Vipelli	01 21 98 51 01

Partie 4 : Bases de données

Les jointures entre tables : qu'est-ce qu'une jointure

- La jointure externe est donc plus complète :
 - elle est capable de récupérer plus d'informations
 - Voici par exemple les données que l'on récupérerait avec une jointure externe :
 - ✓ Romain apparaît maintenant. Comme il ne possède pas de jeu, il n'y a aucun nom de jeu indiqué (NULL).
- La jointure interne est plus stricte :
 - elle ne récupère que les données qui ont une équivalence dans l'autre table.
 - Voici par exemple les données que l'on récupérerait avec une jointure interne :
 - ✓ On obtient les jeux et leurs propriétaires, mais Romain qui ne possède pas de jeu n'apparaît pas du tout.

nom_jeu	prenom
Super Mario Bros	Florent
Sonic	Patrick
...	...
NULL	Romain

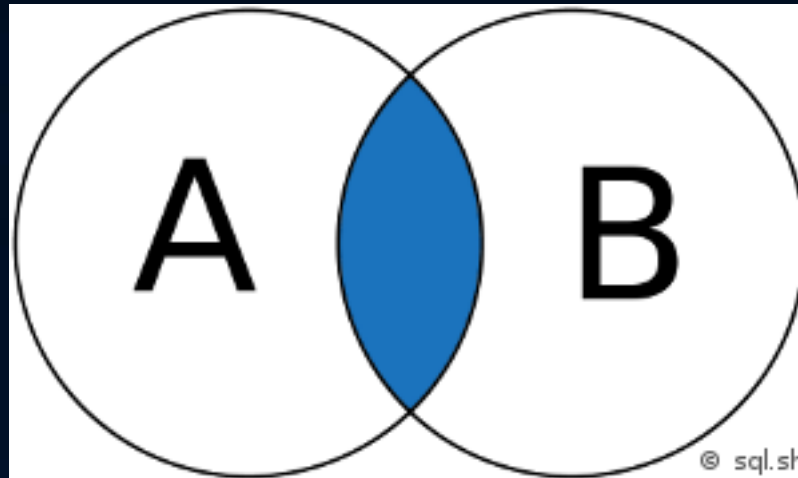
nom_jeu	prenom
Super Mario Bros	Florent
Sonic	Patrick
...	...

Partie 4 : Bases de données

Les jointures entre tables

4. Les jointures internes

- Une jointure interne peut être effectuée de deux façons différentes :
 - à l'aide du mot-clé WHERE :
 - ✓ c'est l'ancienne syntaxe, toujours utilisée aujourd'hui
 - à l'aide du mot-clé JOIN :
 - ✓ c'est la nouvelle syntaxe qu'il est recommandé d'utiliser.
 - ❖ Elle est plus efficace et plus lisible.



Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec WHERE (ancienne syntaxe)
 - Construction d'une jointure interne pas à pas
 - ✓ Pour réaliser ce type de jointure :
 - ❖ on va sélectionner des champs des deux tables
 - ❖ indiquer le nom de ces deux tables dans la clause FROM

```
1 SELECT nom, prenom FROM proprietaires, jeux_video
```

- ❖ Le champ nom apparaît dans les deux tables :
 - une fois pour le nom du propriétaire
 - une autre fois pour le nom du jeu vidéo.
 - C'est ce qu'on appelle une colonne ambiguë :
 - MySQL ne sait pas s'il doit récupérer un nom de personne ou un nom de jeu

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec WHERE (ancienne syntaxe)
 - Construction d'une jointure interne pas à pas
 - ✓ Pour réaliser ce type de jointure :
 - ❖ Il faut indiquer le nom de la table devant le nom du champ :

```
1 SELECT jeux_video.nom, proprietaires.prenom FROM proprietaires, jeux_video
```

- on demande clairement de récupérer le nom du jeu et le prénom du propriétaire
- ❖ Il reste encore à lier les deux tables entre elles :
 - les jeux et leurs propriétaires ont une correspondance via :
 - le champ ID_proprietaire (de la table jeux_video)
 - et le champ ID (de la table proprietaires).
 - On va indiquer cette liaison dans un WHERE :

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec WHERE (ancienne syntaxe)

➤ Construction d'une jointure interne pas à pas

✓ Pour réaliser ce type de jointure :

❖ Il reste encore à lier les deux tables entre elles :

○ On va indiquer cette liaison dans un WHERE :

```
1 SELECT jeux_video.nom, proprietaires.prenom  
2 FROM proprietaires, jeux_video  
3 WHERE jeux_video.ID_proprietaire = proprietaires.ID
```

- On a le droit d'écrire la requête sur plusieurs lignes :
 - Cette écriture a l'avantage d'être plus lisible.

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec WHERE (ancienne syntaxe)
 - Construction d'une jointure interne pas à pas

```
1 SELECT jeux_video.nom, proprietaires.prenom
2 FROM proprietaires, jeux_video
3 WHERE jeux_video.ID_proprietaire = proprietaires.ID
```

- ✓ On indique bien que :
 - ❖ le champ ID_proprietaire de la table jeux_video
 - ❖ correspond au champ ID de la table proprietaires.
 - ❖ Cela établit la correspondance entre les deux tables telle qu'on l'avait définie dans le schéma au début du chapitre.

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec WHERE (ancienne syntaxe)
 - Construction d'une jointure interne pas à pas
 - ✓ La requête est complète
 - ✓ On va récupérer les données suivantes :

nom	prenom
Super Mario Bros	Florent
Sonic	Patrick
...	...

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec WHERE (ancienne syntaxe)
 - Utilisez les alias !
 - ✓ Les alias nous permettait déjà de créer des « champs virtuels » pour représenter le résultat des fonctions.
 - ✓ On va également utiliser des alias lorsqu'on fait des jointures.
 - ❖ On peut utiliser des alias sur les noms de champs :

```
1 SELECT jeux_video.nom AS nom_jeu, proprietaires.prenom AS prenom_proprietaire
2 FROM proprietaires, jeux_video
3 WHERE jeux_video.ID_proprietaire = proprietaires.ID
```

- ❖ On récupère 2 champs : nom_jeu et prenom_proprietaire.
- ❖ Ces alias permettent de donner un nom plus clair aux champs que l'on récupère.

nom_jeu	prenom_proprietaire
Super Mario Bros	Florent
Sonic	Patrick
...	...

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec WHERE (ancienne syntaxe)
 - Utilisez les alias !
 - ✓ Il est également possible de donner un alias aux noms des tables :
 - ❖ recommandé pour leur donner un nom plus court et plus facile à écrire.
 - ❖ En général, on crée des alias de tables d'une lettre ou deux correspondant à leurs initiales :

```
1 SELECT j.nom AS nom_jeu, p.prenom AS prenom_proprietaire
2 FROM proprietaires AS p, jeux_video AS j
3 WHERE j.ID_proprietaire = p.ID
```

- la table jeux_video a pour alias la lettre « j » et proprietaires la lettre « p ».
- On réutilise ces alias dans toute la requête, ce qui la rend :
 - plus courte à écrire
 - plus lisible

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec WHERE (ancienne syntaxe)

➤ Utilisez les alias !

✓ Remarque :

❖ le mot-clé « AS » est facultatif :

○ On a peut le retirer de la requête :

```
1 SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
2 FROM propriétaires p, jeux_video j
3 WHERE j.ID_proprietaire = p.ID
```

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec JOIN (nouvelle syntaxe)

➤ Voici les deux mêmes requêtes :

```
1 SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
2 FROM proprietaires p, jeux_video j
3 WHERE j.ID_proprietaire = p.ID
```

```
1 SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
2 FROM proprietaires p
3 INNER JOIN jeux_video j
4 ON j.ID_proprietaire = p.ID
```

- ✓ On récupère les données depuis la table principale « proprietaires » et on fait une jointure interne (INNER JOIN) avec une autre table (jeux_video).
- ✓ La liaison entre les champs est faite dans la clause ON un peu plus loin.
- ✓ Le fonctionnement reste le même : on récupère les mêmes données qu'avec la syntaxe WHERE.

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec JOIN (nouvelle syntaxe)

➤ Voici les deux mêmes requêtes :

```
1 SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
2 FROM proprietaires p, jeux_video j
3 WHERE j.ID_proprietaire = p.ID
```

```
1 SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
2 FROM proprietaires p
3 INNER JOIN jeux_video j
4 ON j.ID_proprietaire = p.ID
```

- ✓ On récupère les données depuis la table principale « proprietaires » et on fait une jointure interne (INNER JOIN) avec une autre table (jeux_video).
- ✓ La liaison entre les champs est faite dans la clause ON un peu plus loin.
- ✓ Le fonctionnement reste le même : on récupère les mêmes données qu'avec la syntaxe WHERE.

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec JOIN (nouvelle syntaxe)
 - Pour filtrer (WHERE), ordonner (ORDER BY) ou limiter les résultats (LIMIT), il faut le faire à la fin de la requête, après le « ON j.ID_proprietaire = p.ID ».

```
1 SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
2 FROM proprietaires p
3 INNER JOIN jeux_video j
4 ON j.ID_proprietaire = p.ID
5 WHERE j.console = 'PC'
6 ORDER BY prix DESC
7 LIMIT 0, 10
```

Partie 4 : Bases de données

Les jointures entre tables : jointures internes

- Jointure interne avec JOIN (nouvelle syntaxe)
 - Pour filtrer (WHERE), ordonner (ORDER BY) ou limiter les résultats (LIMIT), il faut le faire à la fin de la requête, après le « ON j.ID_proprietaire = p.ID ».
 - ✓ On récupère le nom du jeu et le prénom du propriétaire
 - ✓ dans les tables « propriétaires » et « jeux_video »,
 - ✓ la liaison entre les tables se fait entre les champs ID_proprietaire et ID,
 - ✓ prends uniquement les jeux qui tournent sur PC,
 - ✓ trie par prix décroissants
 - ✓ ne prends que les 10 premiers

Partie 4 : Bases de données

Les jointures entre tables

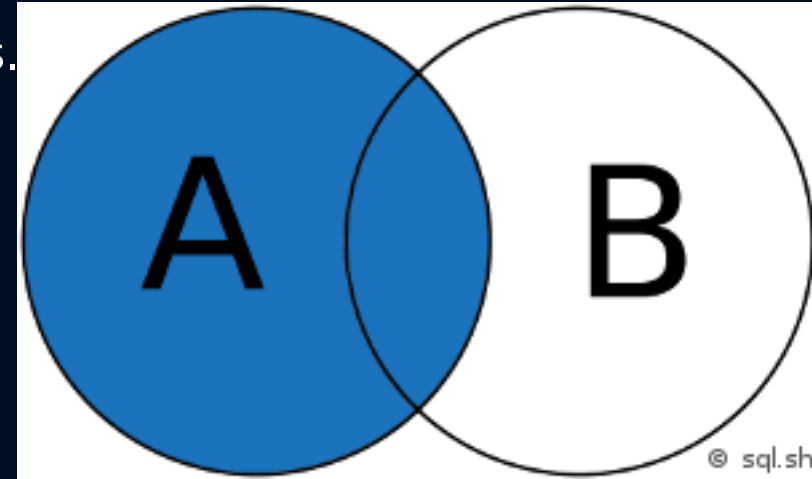
5. Les jointures externes

- Les jointures externes permettent de récupérer :
 - toutes les données,
 - même celles qui n'ont pas de correspondance.
- La seule syntaxe disponible est à base de JOIN.
- Il y a deux écritures à connaître :
 - LEFT JOIN
 - RIGHT JOIN.
 - Cela revient pratiquement au même, avec une subtile différence que nous allons voir.

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- LEFT JOIN (LEFT OUTER JOIN) : récupérer toute la table de gauche
 - la commande LEFT JOIN est un type de jointure entre 2 tables.
 - Cela permet de lister :
 - ✓ tous les résultats de la table de gauche
 - ✓ même s'il n'y a pas de correspondance dans la deuxième table.
 - Syntaxe :
 - ✓ Pour lister les enregistrement de table1, même s'il n'y a pas de correspondance avec table2, il convient d'effectuer une requête SQL utilisant la syntaxe suivante.



```
SELECT *  
FROM table1  
LEFT JOIN table2 ON table1.id = table2.fk_id
```

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- LEFT JOIN (LEFT OUTER JOIN) : récupérer toute la table de gauche
 - Exemple :

```
1 SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
2 FROM proprietaires p
3 LEFT JOIN jeux_video j
4 ON j.ID_proprietaire = p.ID
```

- ✓ « proprietaires » est appelée la « table de gauche »
- ✓ jeux_video la « table de droite ».
- ✓ LEFT JOIN demande à récupérer :
 - ❖ tout le contenu de la table de gauche,
 - ❖ donc tous les propriétaires,
 - ❖ même si ces derniers n'ont pas d'équivalence dans la table « jeux_video »

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

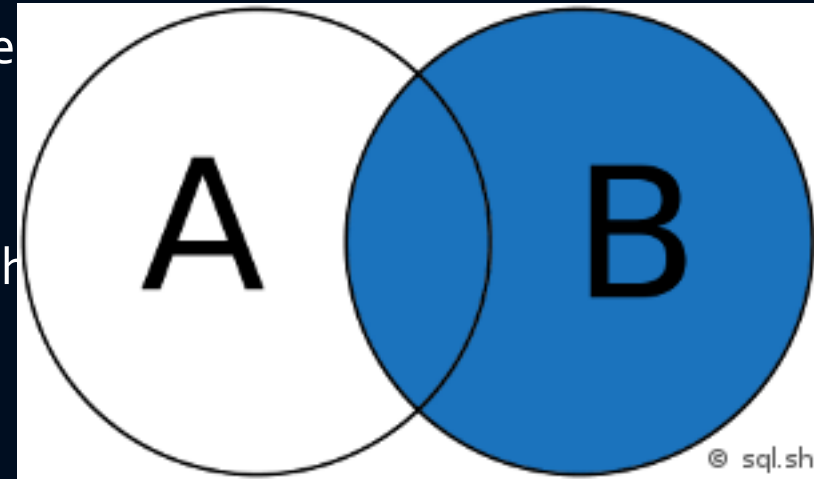
- LEFT JOIN (LEFT OUTER JOIN) : récupérer toute la table de gauche
 - ✓ Romain apparaît désormais dans les résultats de la requête grâce à la jointure externe.
 - ✓ Comme il ne possède aucun jeu, la colonne du nom du jeu est vide.

nom_jeu	prenom_proprietaire
Super Mario Bros	Florent
Sonic	Patrick
...	...
NULL	Romain

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- RIGHT JOIN (ou RIGHT OUTER JOIN) : récupérer toute la table de droite
 - la commande RIGHT JOIN est un type de jointure entre 2 tables
 - permet de retourner :
 - ✓ tous les enregistrements de la table de droite
 - ✓ même s'il n'y a pas de correspondance avec la table de gauche
 - S'il y a un enregistrement de la table de droite qui ne trouve pas de correspondance dans la table de gauche, alors les colonnes de la table de gauche auront NULL pour valeur.
 - Syntaxe :
 - ✓ Pour lister les enregistrements de table2, même s'il n'y a pas de correspondance avec table1, il convient d'effectuer une requête SQL utilisant la syntaxe suivante.



```
SELECT *  
FROM table1  
RIGHT JOIN table2 ON table1.id = table2.fk_id
```

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- RIGHT JOIN : récupérer toute la table de droite
 - RIGHT JOIN demande à récupérer :
 - ✓ toutes les données de la table dite « de droite »,
 - ✓ même si celle-ci n'a pas d'équivalent dans l'autre table.

```
1 SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
2 FROM proprietaires p
3 RIGHT JOIN jeux_video j
4 ON j.ID_proprietaire = p.ID
```

- ❖ La table de droite est « jeux_video ».
- ❖ On récupère donc tous les jeux,
- ❖ même ceux qui n'ont pas de propriétaire associé.

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- RIGHT JOIN : récupérer toute la table de droite
 - Dans ce cas, Bomberman n'appartient à personne.
 - Avec la requête RIGHT JOIN , on obtiendra toutes les lignes de la table de droite (jeux_video) même si elles n'ont aucun lien avec la table « propriétaires », comme c'est le cas ici pour Bomberman.

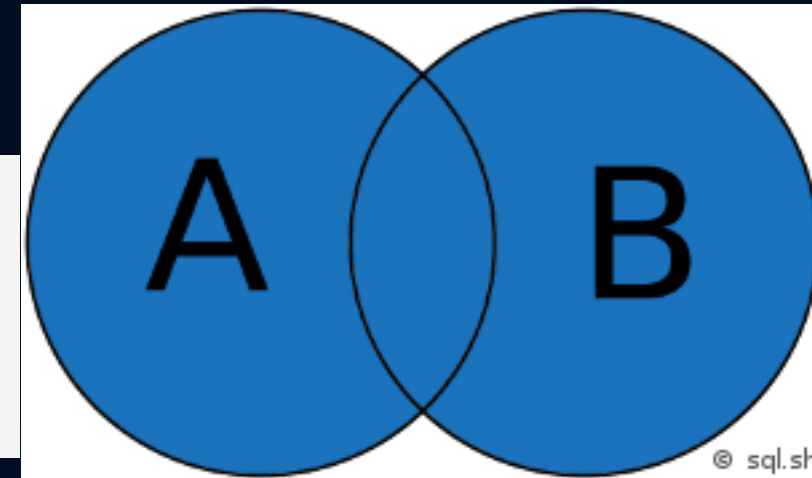
nom_jeu	prenom_proprietaire
Super Mario Bros	Florent
Sonic	Patrick
...	...
Bomberman	NULL

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - FULL JOIN (ou FULL OUTER JOIN, jointure externe) :
 - ✓ retourne les résultats quand la condition est vraie dans au moins une des 2 tables.
 - ✓ permet de combiner les résultats des 2 tables, les associer entre eux grâce à une condition et remplir avec des valeurs NULL si la condition n'est pas respectée.
 - ✓ Syntaxe :
 - ❖ Pour retourner les enregistrements de table1 et table2 :

```
SELECT *  
FROM table1  
FULL JOIN table2 ON table1.id = table2.fk_id
```



Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - FULL JOIN (ou FULL OUTER JOIN, jointure externe) :
 - ✓ Exemple :
 - ❖ Table des clients

id	prenom	nom	email	ville	actif
1	Aimée	Marechal	aime.marechal@example.com	Paris	1
2	Esmée	Lefort	esmee.lefort@example.com	Lyon	0
3	Marine	Prevost	m.prevost@example.com	Lille	1
4	Luc	Rolland	lucrolland@example.com	Marseille	1

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - FULL JOIN (ou FULL OUTER JOIN, jointure externe) :
 - ✓ Exemple :
 - ❖ Table des commandes

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
3	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - FULL JOIN (ou FULL OUTER JOIN, jointure externe) :
 - ✓ Exemple :
 - ❖ Requête :
 - Il est possible d'utiliser FULL JOIN :
 - pour lister tous les clients ayant effectué ou non une vente,
 - et lister toutes les ventes qui sont associées ou non à un utilisateur

```
SELECT id, prenom, nom, utilisateur_id, date_achat, num_facture
FROM utilisateur
FULL JOIN commande ON utilisateur.id = commande.utilisateur_id
```

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - FULL JOIN (ou FULL OUTER JOIN, jointure externe) :
 - ✓ Exemple :
 - ❖ Requête : résultat

id	pre nom	nom	utilisateur_id	date_achat	num_facture
1	Aimée	Marechal	1	2013-01-23	A00103
1	Aimée	Marechal	1	2013-02-14	A00104
2	Esmée	Lefort	2	2013-02-17	A00105
3	Marine	Prevost	3	2013-02-21	A00106
4	Luc	Rolland	NULL	NULL	NULL
NULL	NULL	NULL	5	2013-03-02	A00107

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - CROSS JOIN :
 - ✓ jointure croisée permettant de faire le produit cartésien de 2 tables.
 - ✓ En d'autres mots, permet de joindre chaque lignes d'une table avec chaque lignes d'une seconde table.
 - ✓ Attention, le nombre de résultats est en général très élevé.
 - ✓ Effectuer le produit cartésien :
 - ❖ d'une table A qui contient 30 résultats
 - ❖ avec une table B de 40 résultats
 - ❖ va produire 1200 résultats ($30 \times 40 = 1200$).
 - ❖ En général la commande CROSS JOIN est combinée avec la commande WHERE pour filtrer les résultats qui respectent certaines conditions.

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - CROSS JOIN :
 - ✓ Syntaxe :

```
SELECT *  
FROM table1  
CROSS JOIN table2
```

```
SELECT *  
FROM table1, table2
```

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :

- SELF JOIN :

- ✓ permet d'effectuer une jointure d'une table avec elle-même comme si c'était une autre table.
- ✓ très pratique dans le cas où une table lie des informations avec des enregistrements de la même table.
- ✓ Syntaxe :

```
SELECT `t1`.`nom_colonne1`, `t1`.`nom_colonne2`, `t2`.`nom_colonne1`, `t2`.`nom_colonne2`  
FROM `table` as `t1`  
LEFT OUTER JOIN `table` as `t2` ON `t2`.`fk_id` = `t1`.`id`
```

- ❖ Ici la jointure est effectuée avec un LEFT JOIN,
- ❖ il est aussi possible de l'effectuer avec d'autres types de jointures.

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - SELF JOIN :
 - ✓ Exemple : table utilisateur
 - ❖ Une application d'un intranet d'entreprise qui possède la table des employés avec la hiérarchie entre eux.
 - ❖ Les employés peuvent être dirigés par un supérieur direct qui se trouve lui-même dans la table.

id	pre nom	nom	email	manager_id
1	Sebastien	Martin	s.martin@example.com	NULL
2	Gustave	Dubois	g.dubois@example.com	NULL
3	Georgette	Leroy	g.leroy@example.com	1
4	Gregory	Roux	g.roux@example.com	2

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :

- SELF JOIN :

- ✓ Exemple : table utilisateur

- ❖ Il est possible de lister sur une même ligne les employés avec leurs supérieurs direct :

```
SELECT `u1`.`u_id`, `u1`.`u_nom`, `u2`.`u_id`, `u2`.`u_nom`  
FROM `utilisateur` as `u1`  
LEFT OUTER JOIN `utilisateur` as `u2` ON `u2`.`u_manager_id` = `u1`.`u_id`
```

u1_id	u1_prenom	u1_nom	u1_email	u1_manager_id	u2_prenom	u2_nom
1	Sebastien	Martin	s.martin@example.com	NULL	NULL	NULL
2	Gustave	Dubois	g.dubois@example.com	NULL	NULL	NULL
3	Georgette	Leroy	g.leroy@example.com	1	Sebastien	Martin
4	Gregory	Roux	g.roux@example.com	2	Gustave	Dubois

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - NATURAL JOIN :
 - ✓ jointure naturelle entre 2 tables s'il y a au moins une colonne qui porte le même nom et qui est de même type dans les 2 tables
 - ✓ Le résultat est la création d'un tableau avec autant de lignes qu'il y a de paires correspondant à l'association des colonnes de même nom.
 - ✓ Syntaxe :
 - ❖ L'avantage d'un NATURAL JOIN c'est qu'il n'y a pas besoin d'utiliser la clause ON.

```
SELECT *  
FROM table1  
NATURAL JOIN table2
```

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :

- NATURAL JOIN :

- ✓ Exemple :

- ❖ une application qui utilise une table utilisateur et une table pays.

- ❖ Si la table utilisateur contient une colonne pour l'identifiant du pays, il sera possible d'effectuer une jointure naturelle.

- ❖ Table utilisateur

user_id	user_prenom	user_ville	pays_id
1	Jérémie	Paris	1
2	Damien	Montréal	2
3	Sophie	Marseille	NULL
4	Yann	Lille	9999
5	Léa	Paris	1

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - NATURAL JOIN :
 - ✓ Exemple :
 - ❖ Table pays

pays_id	pays_nom
1	France
2	Canada
3	Belgique
4	Suisse

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - NATURAL JOIN :
 - ✓ Exemple :
 - ❖ Requête :
 - Pour avoir la liste de tous les utilisateurs avec le pays correspondant :

```
SELECT *  
FROM utilisateur  
NATURAL JOIN pays
```

Partie 4 : Bases de données

Les jointures entre tables : jointures externes

- Les autres jointures :
 - NATURAL JOIN :
 - ✓ Exemple :
 - ❖ Résultat :

pays_id	user_id	user_prenom	user_ville	pays_nom
1	1	Jérémie	Paris	France
2	2	Damien	Montréal	Canada
NULL	3	Sophie	Marseille	NULL
9999	4	Yann	Lille	NULL
1	5	Léa	Paris	France

Partie 4 : Bases de données

Les jointures entre tables

6. Résumé

- Les bases de données permettent d'associer plusieurs tables entre elles.
- Une table peut contenir les id d'une autre table ce qui permet de faire la liaison entre les deux.
- Pour rassembler les informations au moment de la requête, on effectue des jointures.
- On peut faire des jointures avec le mot-clé WHERE, mais il est recommandé d'utiliser JOIN qui offre plus de possibilités et qui est plus adapté.
- On distingue :
 - les jointures internes, qui retournent des données uniquement s'il y a une correspondance entre les deux tables,
 - les jointures externes qui retournent toutes les données même s'il n'y a pas de correspondance.

Partie 4 : Bases de données

Les jointures entre tables

- Exercice :
 - Ajouter tables :
 - ✓ « propriétaire » composée des champs :
 - ❖ id_proprietaire : clé primaire auto-incrémentation
 - ❖ nom_proprietaire
 - ✓ « console » composée des champs :
 - ❖ id_console : clé primaire auto-incrémentation
 - ❖ nom_console
 - Insérer un champs « id_proprietaire » et un champs « id_console » dans la table « jeux_video »

Partie 4 : Bases de données

Les jointures entre tables

- Exercice :
 - Créer un script pour : `script insert_table_proprietaire.php`
 - ✓ Insérer les informations des différents « possesseur » de la table « jeux_video » dans la table « proprietaire »
 - ✓ Insérer dans la table « jeux_video » l'id_proprietaire correspondant pour chaque entrée
 - ✓ Supprimer, par un script, la colonne « possesseur » de la table « jeux_video »
 - Créer un script pour : `insert_table_console.php`
 - ✓ Insérer les informations des différentes « console » de la table « jeux_video » dans la table « console »
 - ✓ Insérer dans la table « jeux_video » l'id_console correspondant pour chaque entrée
 - ✓ Supprimer, par un script, la colonne « console » de la table « jeux_video »
 - Créer un script pour ajouter une entrée dans la table « jeux_video » : `script insert_new_jeux.php`
 - ✓ Nom du jeu : jeu de l'IFA
 - ✓ Prix : moyenne des prix de tous les jeux
 - ✓ Nbre_joueurs_max : max enregistré dans la table
 - ✓ Commentaires : meilleur jeu de l'IFA
 - ✓ Date_enregistrement : date courante, heure courante
 - ✓ Id_proprietaire : nouveau proprietaire = IFA
 - ✓ Id_console : néant

Partie 4 : Bases de données

Les jointures entre tables

- Exercice :
 - Créer un script pour ajouter une entrée dans la table « console » : `script insert_new_console.php`
 - ✓ Nom_console : console IFA

Partie 4 : Bases de données

Les jointures entre tables

- Exercice :
 - Créer un script pour : script
 - ✓ Lister pour chaque entrée de la table « jeux_video » le type de console utilisé en se servant des jointures : script select_left_join.php
 - ✓ Lister pour chaque console les jeux associés : script select_left_join.php
 - ✓ Lister tous les jeux qui ont une console identifiée : script select_left_join.php

Partie 4 : Bases de données

Les jointures entre tables

- <https://openclassrooms.com/courses/faites-une-base-de-donnees-avec-uml>