Bases de données

B

Toute application, quelle qu'elle soit, manipule des données. Les systèmes de gestion de bases de données (SGBD) jouent alors un rôle essentiel : fournir un service efficace de stockage et d'extraction des données. Le choix du SGBD et de ses outils, le schéma de la base et la configuration de cet ensemble sont autant de sujets stratégiques qui requièrent la plus grande attention. Comprendre les notions essentielles liées aux bases de données est un point de départ nécessaire pour garantir performance, stabilité et durabilité à vos applications.

SOMMAIRE

MOTS-CLÉS

- base de données
- SGBD
- CRUD
- ORM
- PDO
- SQL

Référence **Best practices PHP 5**

Le chapitre 7 de l'ouvrage français *Best practices PHP 5* est un bon support pour approfondir votre connaissance des supports de données. Il détaille la plupart des notions abordées dans ce chapitre.

📖 G. Ponçon, *Best practices PHP 5*, Eyrolles, 2005

Cette annexe balaye les notions essentielles liées aux bases de données pour PHP, à commencer par comprendre ce qu'est réellement un SGBD, puis comment se fait la connexion entre un SGBD et PHP. D'autre part, il existe de nombreux outils en PHP pour simplifier les manipulations de données stockées dans des bases. Ces outils sont-ils performants ? Garantissent-ils la durabilité de l'application ? Leur choix est-il pertinent ? Nous vous proposons ici de quoi juger par vous-mêmes.

En revanche, cette section n'a pas pour vocation de vous apprendre SQL et d'entrer dans les détails d'utilisation d'un SGBD. Il fait simplement figure de rappel des notions essentielles.

Qu'est-ce qu'un SGBD ?

Cette première partie expose ce qu'est un SGBD et comment on l'utilise avec PHP :

- *l'architecture d'un SGBD* : de quoi se compose un système de gestion de bases de données ? Comment fonctionne-t-il ?
- *les principaux SGBD du marché* : la liste des outils à votre disposition, leurs objectifs, avantages et inconvénients, pour faire un choix pertinent ;
- *la connexion à PHP* : comment PHP peut-il collaborer avec votre SGBD ? Quelle solution d'interfaçage choisir ?

Ces connaissances de base sont surtout utiles pour faire les bons choix concernant vos projets PHP. Elles constituent une culture générale essentielle sur ce sujet important.

Architecture d'un SGBD

Un SGBD, comme son nom l'indique, permet de manipuler des bases de données. Il faut bien distinguer ces deux notions de base :

- Une *base de données* est composée d'un ensemble de données structurées. Une application peut utiliser une ou plusieurs bases de données. Physiquement, une base de données est constituée d'un ou plusieurs fichiers qui contiennent des données et des informations sur leur structure.
- Un *système de gestion de bases de données (SGBD)* est l'outil qui permet de manipuler les bases de données. C'est lui que l'on interroge pour ajouter, extraire, supprimer ou modifier des données dans une base.

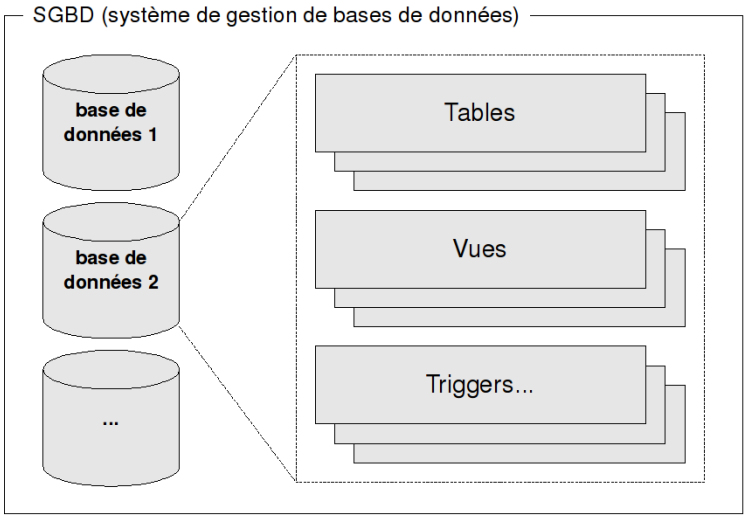

Figure B–1
Architecture d'un SGBD

La base de données

La base de données est l'élément essentiel que vous allez solliciter à chaque fois que vous voudrez manipuler des données.

Exemple simple

Voici une méthode simple pour bien comprendre la composition générale d'une base de données et construire une base cohérente. Pour cela, prenons un carnet d'adresses :

- Question 1 : *De quoi est composé un carnet d'adresses ?*
  - D'utilisateurs et d'adresses.
- Question 2 : *De quoi sont composés les utilisateurs et les adresses ?*
  - Un utilisateur possède un nom et un prénom.
  - Une adresse possède une rue, un code postal et une ville.
- Question 3 : *Quel lien y a-t-il entre les utilisateurs et les adresses ?*
  - Un utilisateur possède une adresse.
  - Une adresse peut appartenir à un ou plusieurs utilisateurs.

Notions techniques

Ces questions peuvent paraître un peu naïves, mais elles sont essentielles à la mise en place de la structure de votre base de données. Chacune d'elles est liée à des concepts techniques :

- Répondre à la question 1 permet d'identifier les *tables* de votre base. Pour répondre à cette question de manière pertinente, il est impor-

tant de savoir ce que vous voudrez faire de ces données. Ici on décide d'avoir deux notions : utilisateurs et adresses, parce qu'on veut pouvoir manipuler l'un et l'autre de manière indépendante.

- Répondre à la question 2 permet d'identifier les *champs* contenus dans chaque table. Un champ représente une donnée élémentaire. On associera chaque donnée élémentaire à un *type de données* :
  - nom : `chaîne de caractères` ;
  - prénom : `chaîne de caractères` ;
  - rue : `chaîne de caractères` ;
  - code postal : `numéro` ;
  - ville : `chaîne de caractères`.
- Répondre à la question 3 permet d'avoir des informations sur l'*intégrité* qu'il faut donner aux données. Bien que ce soit facultatif, il est souvent possible de spécifier ces liens de manière à prévenir les incohérences. On appelle ces paramètres des *contraintes d'intégrité* :
  - je veux qu'une personne ne puisse être liée qu'à une seule adresse ;
  - j'admets que plusieurs personnes puissent avoir la même adresse.

CONSEIL Outil de conception

En cherchant sur Internet, vous trouverez plusieurs outils de conception de schémas comme on le voit sur la figure B-2. Mais il est aussi possible et même recommandé de se passer de ces outils. Un tableau blanc, quand on est plusieurs à réfléchir sur un schéma, reste le meilleur des supports.

Représentation graphique

Pour plus de clarté, il est d'usage de représenter graphiquement ces différents éléments : *tables*, *champs* et *relations*. La figure B-2 représente la base de données, contenant utilisateurs et adresses. On appelle ce type de schéma un modèle conceptuel de données (MCD).

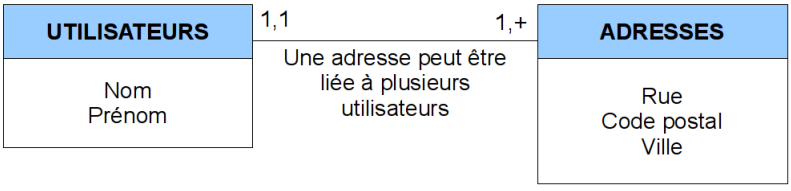

Figure B–2
Modèle conceptuel de données (MCD)

Types de données

Il est important, pour chaque champ d'une table, de déterminer son *type de donnée*, c'est-à-dire la syntaxe que doit respecter chaque donnée. Par exemple, le champ `Rue` et le champ `Ville` seront associés au type de donnée `VARCHAR` (chaîne de caractères) et le champ `Code postal` au type de donnée `INTEGER` (nombre entier). Ainsi, il sera impossible de mettre une valeur autre qu'un nombre dans le code postal et ainsi de suite.

Voici les types de données que l'on retrouve dans les SGBD courants :

- `CHAR(X)` (chaîne de caractères de longueur X fixe) ;
- `VARCHAR(X)` (chaîne de caractères de longueur X variable) ;

- `INTEGER` (nombre entier) ;
- `FLOAT, DECIMAL, DOUBLE` (nombre à virgule) ;
- `DATE, DATETIME, TIMESTAMP` (dates et heures) ;
- `BLOB, CLOB` (données binaires ou chaînes de caractères longues).

Clés et contraintes d'intégrité

Une table peut comporter des champs spéciaux appelés `clés`. Ces clés permettent de garantir l'intégrité des données et/ou d'accélérer les performances. Voici les différents types de clés que l'on peut associer à un champ :

- la clé primaire, souvent obligatoire, permet de garantir l'unicité de chaque enregistrement. La valeur d'une clé primaire est obligatoire et unique. On choisit souvent comme clé primaire un identifiant numérique auto-incrémenté ou un code (code produit) ;
- la clé unique permet de faire en sorte que la valeur d'un champ soit unique. En d'autres termes, que celui-ci ne comporte pas de doublons. Par exemple, on peut définir le champ e-mail comme unique pour éviter que deux utilisateurs déclarent avoir le même e-mail. Plusieurs champs peuvent avoir des clés uniques indépendantes (contrairement à la clé primaire) et une même clé unique peut être définie sur un ou plusieurs champs (comme la clé primaire) ;
- la clé index n'ajoute pas de contrainte. Elle permet juste d'accélérer la recherche sur le champ indexé.

Les clés ne sont pas les seules contraintes que l'on peut déclarer. Imaginons que nous souhaitons faire en sorte qu'un utilisateur ne soit lié qu'à une seule adresse à la fois, tout en acceptant qu'une adresse soit liée à plusieurs utilisateurs.

On peut pour cela définir une liaison que l'on appellera contrainte d'intégrité. Les SGBD qui sont capables de gérer ce genre de contraintes sont appelés SGBD relationnels ou SGBDR. Avec MySQL, le moteur InnoDB est capable de maintenir l'intégrité des données par l'intermédiaire de ces liaisons fortes.

CULTURE Indexation des clés

Toute clé, qu'elle soit primaire, unique ou simplement définie comme un index, est indexée. Une recherche sur un index est rapide. En revanche, plus il y a de champs indexés dans une table, plus les insertions, modifications et suppressions sont lentes.

Les principaux SGBD du marché

Il existe de nombreux SGBD utilisables avec PHP. Seul un petit groupe fait partie du cercle des SGBD courants. Nous nous limiterons ici à ces produits.

MySQL

MySQL est historiquement lié à PHP. Si les SGBD courants privilégient avant tout l'intégrité des données, MySQL a vu le jour pour privilégier les performances. Aujourd'hui, cet aspect performance est toujours au rendez-

vous, mais l'utilisateur peut choisir plusieurs moteurs en fonction de ses besoins et ainsi répondre pleinement à des problématiques web ou non web.

On utilisera MySQL pour tout type d'application web, de petite taille, de taille moyenne et même de grande taille. Il s'agit d'un bon choix dans tous les cas.

Voici les principaux moteurs que l'on peut utiliser avec MySQL :

- MYISAM : optimisé pour les performances, il s'agit du successeur du moteur original de MySQL, anciennement ISAM. Ce moteur ne permet pas d'avoir de fortes contraintes d'intégrité, ni non plus d'annuler une ou plusieurs opérations (gestion des transactions, avec *commit* et *rollback*). L'absence de ces mécanismes offre des performances améliorées ;
- INNODB : moteur transactionnel très populaire et le plus apprécié des entreprises. Contrairement à MYISAM, il permet de mettre en place des contraintes d'intégrité fortes. Il offre aussi la possibilité de faire des requêtes transactionnelles, avec validation ou annulation (*commit*, *rollback*). Ce moteur est actuellement la propriété de la société Oracle, il est sous licence propriétaire ;
- ARCHIVE : moteur optimisé pour l'écriture et non pour la lecture. Il est utile pour stocker des logs ou des données de sauvegarde ;
- MEMORY : moteur qui permet de créer des tables dans la mémoire, utilisé pour accélérer toute opération. Bien entendu, toute table créée avec MEMORY est volatile, un arrêt du SGBD ou du serveur supprimant tout le contenu.

Il existe d'autres moteurs pour MySQL que vous pouvez étudier sur la documentation officielle en ligne. La liste que nous vous avons présentée regroupe les plus populaires.

DÉFINITION Commit, Rollback, Transaction

Une *transaction* est l'action de rendre unique un ensemble de requêtes SQL. Par exemple, si vous voulez insérer un utilisateur et son adresse, il vous faudra peut-être deux requêtes SQL. Une transaction est liée à deux opérations fondamentale : la validation (*commit*) et l'annulation (*rollback*). L'annulation de l'ensemble des requêtes (*rollback*) intervient si au moins l'une d'entre elles échoue, et la validation générale de l'ensemble des requêtes (*commit*) intervient si elles ont toutes été exécutées avec succès. Ainsi, pour reprendre notre exemple, il ne peut y avoir d'utilisateur sans adresse ou d'adresse sans utilisateur. La transaction assure de ce fait l'intégrité de nos données.

Oracle

La réputation d'Oracle n'est plus à faire, en particulier dans le monde de l'entreprise. Ce SGBD est choisi généralement pour mettre en place de grosses bases de données tels des systèmes d'information stratégiques. Si aujourd'hui nous en voyons de plus en plus avec MySQL, Oracle reste une valeur sûre.

Oracle est un produit payant. Il existe une version de développement gratuite que l'on peut utiliser dans un cadre non-commercial, appelée Oracle XE.

Oracle doit être choisi dans le cadre de projets d'envergure. Ce SGBD peut s'avérer difficile et long à maîtriser. Bien souvent, l'intervention d'un administrateur spécialisé est requise pour optimiser une base de données Oracle.

Aussi, le problème d'Oracle réside dans la lenteur de l'ouverture d'une connexion, ce qui le rend peu adapté à une utilisation avec PHP, incapable de maintenir un *pool* de connexion entre deux requêtes.

SQLite

SQLite est un petit SGBD embarqué. Le terme « embarqué » signifie qu'il ne nécessite pas la présence d'un serveur, contrairement à Oracle ou MySQL. Une base de données est stockée dans un fichier qui peut être inclus dans l'application elle-même.

SQLite est également un SGBD très permissif. Un champ déclaré avec le type `INTEGER` pourra par exemple contenir des chaînes de caractères, même s'il n'est pas conseillé de le faire.

Enfin, on utilisera SQLite pour gérer des données non critiques. On peut l'utiliser par exemple pour faire du cache ou comme base intermédiaire. La caractéristique embarquée permet aussi de simplifier la maintenance, car nul besoin d'identifiant et mot de passe ni d'un quelconque paramétrage pour faire fonctionner une base SQLite. Par exemple, SQLite est présent dans de nombreux systèmes embarqués nécessitant un accès rapide à des données : téléphones portable, « box » ADSL...

Le problème majeur de SQLite est son mode de verrouillage. En effet, il verrouille ses bases intégralement lors d'une opération d'écriture. Si ses performances en lecture sont très élevées, c'est loin d'être le cas en écriture. Il faudra donc privilégier SQLite dans des environnements où les accès en lecture sont très largement majoritaires sur les accès en écriture.

REMARQUE Tests avec SQLite

SQLite est très pratique en PHP pour lancer des tests. Il assure alors la gestion des données éphémères nécessaires pour tester les programmes.

Connexion à PHP

Pour accéder à une base de données avec PHP, une extension spécifique est nécessaire. Aujourd'hui il existe deux sortes d'extensions : les extensions indépendantes et les pilotes PDO (*PHP Data Objects*).

- Les extensions *indépendantes* sont les toutes premières à avoir vu le jour. Elles proposent des fonctions et parfois des classes permettant d'effectuer des opérations sur le SGBD.
- Les extensions *PDO* permettent de lier le SGBD à PDO, qui propose des classes de manipulation standard, quel que soit le SGBD utilisé. L'utilisation de PDO est intéressante aussi bien pour simplifier la maintenance que pour assurer la durabilité des développements.

RÉFÉRENCE Documentation en ligne

L'ensemble des extensions disponibles pour l'accès aux bases de données est fourni dans la documentation en ligne de PHP. La page suivante propose la liste des pages utiles :

- http://www.php.net/manual/fr/refs.database.php

Voici les extensions utilisables pour se connecter aux SGBD les plus courants :

MySQL

- `mysql` : l'extension `mysql` standard est aujourd'hui obsolète. Elle est encore disponible pour la compatibilité avec les applications qui ont été développées avec elle, mais elle sera supprimée un jour (lointain) de PHP. Elle n'est actuellement plus maintenue, au profit de `mysqli`.
  - http://www.php.net/manual/fr/book.mysql.php
- `mysqli` : cette extension permet de se connecter à MySQL en mode procédural ou objet. Elle est compatible avec les dernières versions de MySQL et possède bien plus de fonctions que l'extension `mysql`, le *i* signifiant *improved* (amélioré). Il est fortement conseillé de privilégier cette extension par rapport à l'extension `mysql` standard.
  - http://www.php.net/manual/fr/book.mysqli.php
- `pdo_mysql` : le driver PDO pour MySQL. Aujourd'hui, cette extension est préconisée dans la plupart des cas.
  - http://www.php.net/pdo

Oracle

- `oracle` : l'ancienne extension Oracle n'est plus utilisée, elle est dépréciée et n'est pas recommandée.
- `oci8` : cette extension est la plus couramment utilisée jusqu'ici. Elle permet entre autres d'effectuer de nombreuses opérations spécifiques à Oracle.
  - http://www.php.net/manual/fr/book.oci8.php
- `pdo_oci` : le driver PDO pour Oracle. S'il est encore expérimental aujourd'hui, il répond à la plupart des besoins et s'avère de bonne facture.
  - http://www.php.net/manual/fr/ref.pdo-oci.php

SQLite

- `sqlite` : extension procédurale et objet permettant de se connecter à une base SQLite et d'effectuer au besoin des opérations avancées.
  - http://www.php.net/manual/fr/book.sqlite.php
- `pdo_sqlite` : le driver PDO permettant de se connecter à une base SQLite. Il existe deux versions : une pour la version 2 de SQLite et une autre pour la version 3 qui s'avère plus performante, selon la documentation.
  - http://www.php.net/manual/fr/ref.pdo-sqlite.php

PDO

PDO (*PHP Data Objects*) est une extension un peu particulière qui permet d'utiliser la même interface (classes et méthodes) quel que soit le SGBD sous-jacent. Les extensions qui font la liaison entre PDO et les SGBD s'appellent des pilotes (*drivers*) : `pdo_mysql`, `pdo_oci`, `pdo_sqlite`, etc.

PostgreSQL

- `pgsql` : une extension procédurale permettant la connexion et le pilotage d'un serveur PostgreSQL.
  - http://www.php.net/manual/fr/book.pgsql.php
- `pdo_pgsql` : le driver PDO de PostgreSQL.
  - http://www.php.net/manual/fr/ref.pdo-pgsql.php

Notions avancées

Le monde des SGBD est vaste et peut devenir compliqué dès qu'on s'intéresse à des architectures complexes ayant de fortes contraintes de performances. Voici une liste de cas pour lesquels il sera nécessaire d'avoir des connaissances avancées :

- besoin d'une *architecture répartie* – c'est-à-dire une base de données constituée de plusieurs nœuds, avec des réplications ou des partages qui sont paramétrés pour assurer un bon équilibre entre performances, intégrité et mises à jour des données ;
- une *forte charge* qui nécessite d'avoir une architecture répliquée, et des tampons qui permettent d'absorber les nombreuses demandes simultanées en lecture ou en écriture ;
- une *grande complexité* – un système d'information qui doit stocker de nombreuses données différentes, réparties en plusieurs bases ayant des relations entre elles. Une architecture qui n'a pas été bien pensée au départ peut devenir très difficile à maintenir ou à déboguer.

Les ORM

Le mapping objet-relationnel (ORM – *Object-Relational Mapping*) est un outil pratique qui permet au développeur PHP de manipuler une base de données avec des objets (le plus souvent générés), créant l'illusion d'une base objet.

L'ORM est généralement utilisé lorsque l'on doit effectuer de nombreuses petites requêtes paramétrées. En fonction des implémentations, l'ORM peut proposer une mise en cache automatique des résultats de requêtes, car le principal inconvénient d'un ORM est son impact important sur les performances.

Il existe en PHP plusieurs solutions ORM, parmi lesquelles :

- pdoMap (génération de classes à partir d'un fichier XSD) ;
- DB_DataObject (PEAR) ;
- Propel (l'un des générateurs d'objets les plus connus) ;

- Doctrine (l'un des plus complets à ce jour) ;
- Jelix (framework qui intègre son propre ORM).

Couches d'abstraction

Avec une couche d'abstraction de bases de données, il est possible de changer de SGBD sans modifier une ligne de code. L'outil le plus connu en PHP s'appelle ADOdb.

Les couches d'abstraction, si elles permettent une migration rapide d'un SGBD à un autre, manipulent les requêtes SQL afin de les rendre compatibles avec la base sous-jacente, ce qui n'est pas sans inconvénients :

- *des performances moindres*, bien que relatives... une requête mal écrite aura davantage d'impact sur les performances que la couche d'abstraction ;
- *l'impossibilité d'utiliser des fonctionnalités spécifiques* du SGBD sous-jacent, sous peine de ne pas être compatible avec les autres SGBD, ce qui rendrait la présence de la couche d'abstraction un peu obsolète.

RESSOURCE **ADOdb**

Quelques informations sur ADOdb :
▸ http://adodb.sourceforge.net/

Réplication et clustering

Ces opérations répondent souvent à des besoins de performances et, dans une moindre mesure, permettent d'effectuer des sauvegardes en temps réel.

La *réplication* est un mécanisme qui consiste à reproduire en temps réel les opérations d'écriture, de modification ou de suppression effectuées sur une base maître dans une base esclave. Elle permet d'obtenir des tables ou des bases de données identiques, sous réserve du temps de propagation des opérations de réplication.

La réplication est utile lorsque l'on a beaucoup de requêtes en lecture. Liées à un répartiteur de charge, deux bases de données, ou plus, peuvent se partager le traitement des requêtes de manière équilibrée.

Le *clustering* est quant à lui un système permettant de ne voir qu'une seule base de données lorsqu'il y en a plusieurs. L'objectif du cluster est d'augmenter la puissance de calcul en utilisant plusieurs ordinateurs qui n'en représentent qu'un seul.

Des SGBD comme Oracle ou MySQL proposent des solutions de réplication et de clustering. Pour avoir plus d'informations sur ces concepts avancés, nous vous conseillons la lecture de la documentation en ligne:

- clustering avec Oracle : http://www.oracle.com/technology/products/database/clustering/index.html ;
- clustering avec MySQL : http://www.mysql.com/cluster.