nature à produire l'effet inverse voulu. Il faut donc commencer par le *scale up*, c'est-à-dire augmenter les ressources du serveur (RAM, disque, CPU…), à condition d'avoir prévu dès le départ une machine évolutive ! Si tel n'était pas le cas, il faudrait alors effectuer une migration sur une nouvelle machine.

Néanmoins, il est possible d'élargir la surface d'attaque d'une application en répartissant la base de données sur plusieurs serveurs, à conditions que certaines données ne figurent que sur certains serveurs et que l'on dispose de divers mécanisme de réplication pour les données communes.

Prenons l'exemple d'un site web marchand. Étudions une solution à 4 serveurs de bases de données. Dans une telle application on trouve des produits, des clients, des commandes, des factures et des commentaires de clients.
Tant qu'un utilisateur navigue anonymement, il peut lire les offres sur n'importe quel serveur et un mécanisme d'équilibrage de charge (*load balancing*), va le diriger vers le serveur le moins chargé à l'instant *t*.
Dès que cet internaute s'authentifie (ou crée un compte) une fonction de hachage appliquée à son identifiant (un mail par exemple), associée à un calcul de modulo (ici 4) va le conduire sur l'un des serveurs numéroté de 0 à 3.
On ne trouvera donc sur chaque serveur qu'un quart des clients, des commandes et des factures. Quant aux commentaires et aux produits ils seront communs. Mais de façon à ne pas surchager les serveurs on pourra les répliquer de manière asynchrone : par exemple toutes les nuits, les nouveaux produits seront répliqués depuis un serveur de « *back office* » tandis que les commentaires clients seront répliqués de manière croisée entre les différents serveurs du « *front office* » c'est-à-dire de la boutique web.

Certains SGBDR comme Oracle, favorise ce genre de technique notamment par le biais d'un outil comme Oracle RAC (*Oracle Real Application Cluster*) qui s'approche de la notion de *Grid Computing*. Mais il faut que les applications aient été spécialement conçues pour en tirer pleinement parti, notamment au niveau du modèle de données.

# 6 - Le Plan de Reprise d'Activité

Le plan de reprise d'activité (ou PRA) consiste à mettre en œuvre différents mécanismes pour faire en sorte qu'un sinistre majeur, externe ou interne, comme un incendie, une inondation, un cyclone, un tremblement de terre, etc. ne mettent pas en péril l'activité de l'entreprise, dont le système informatique et en particulier le service des données, doit être en mesure de redémarrer dans un délai assez court, quelques heures en général.
Ce système est assuré par des mécanismes de sauvegardes, d'envoi de journaux de transaction, de réplication effectués directement par le SGBDR, mais peut aussi être réalisé par des moyens externes : duplication des données d'une baie, virtualisation de serveurs...

## 6.1 - Les sauvegardes de bases de données

L'opération de sauvegarde consiste à capturer toutes les données de la base, par un moyen quelconque et fournir une copie restaurable (par un moyen inverse de celui mise en œuvre lors de la sauvegarde) capable de rétablir une base de données cohérente, garantissant l'intégrité des données. Cette opération doit pouvoir être effectuée même lorsque les utilisateurs continuent d'alimenter la base.

Notons que la fonction de sauvegarde ne se cantonne pas qu'au PRA, mais aussi à l'archivage, par exemple pour des données comptables.

Il existe six principaux modes de sauvegardes : les sauvegardes externes par copie de fichier ou extraction des données et métadonnées qui s'effectuent à froid et les sauvegardes internes de base de données qui s'effectuent à chaud et peuvent être réalisées en mode complet, différentiel, incrémental, ou transactionnel.

### 6.1.1 - Sauvegarde à froid par copie de fichiers

Le principe est de copier l'ensemble des fichiers de la base comme on le ferait pour n'importe quels autres fichiers.
Une telle sauvegarde ne peut pas s'effectuer si la base est en production et accessible en écriture. En effet, en fonction de la volumétrie, cette copie va durer un certain temps et pendant ce temps, des utilisateurs pourraient rajouter des clients, des factures des commandes... Pour peu que l'on sauvegarde les données des clients avant les données de commandes ou de facturation, il est possible qu'un nouveau client et sa commande soient insérés après la copie des données de la table des clients, mais avant la copie de la table des commandes. A la restauration, la base est devenue incohérente. C'est pourquoi il convient pour le moins d'interdire toute mise à jour le temps d'effectuer cette copie, voire d'arrêter le service des données.
Pour les SGBDR dotés d'une gestion interne des espaces de stockage, c'est à dire ayant de leurs propre routines de lecture et d'écriture de données comme IBM DB2, Oracle ou SQL Server, il n'est pas possible d'accéder aux fichiers composant la base, tant qu'elle est en ligne, ce qui garantit que la copie sera cohérente.
Pour les bases de données qui utilisent le système de fichiers du système d'exploitation, sans gérer directement les opérations de lecture et d'écriture (MySQL et PostGreSQL en particulier), il est parfaitement possible de copier les fichiers alors que la base est exploitée, au risque de la rendre totalement incohérente, sans pour cela que l'on s'en aperçoive de prime abord...
Il faut donc être très vigilant sur cette technique qui est peu recommandable et souvent lente. Elle possède néanmoins un avantage : d'être généralement la solution la plus rapide pour la restauration.

### 6.1.2 - Sauvegarde à froid par extraction des données et métadonnées (dump)

Ce mode de sauvegarde ancestral consiste à extraire d'une part le code SQL de création des objets (SQL DDL) constitué par les schémas SQL, les domaines, les tables, les contraintes, les index, les vues, les utilisateurs SQL et leurs privilèges et l'ensemble des routines SQL (fonctions utilisateur, procédures stockées et déclencheurs), et d'autre part les données, soit par script SQL de rétro insertion (INSERT INTO ...) soit par chargement de fichiers.
Plus encore que pour la précédente méthode, il est indispensable ici d'arrêter le service des données. En effet, le scénario d'insertion évoqué à l'item précédent est tout aussi problématique dans ce cas de figure, mais empêcherait la restauration du fait des contraintes d'intégrité référentielles liées à la gestion des clefs étrangères !
Ce mode de sauvegarde est encore plus lent et beaucoup moins recommandable encore que le mode précédent...

### 6.1.3 - Un mot sur les sauvegardes à froid

Comme nous l'avons vu, toutes les sauvegardes à froid possèdent le même inconvénient : l'obligation de restreindre ou d'arrêter le service des données, qui peut se faire base par base dans

un système multibase (SQL Server, PostGreSQL ou MySQL) ou bien au niveau de l'instance pour un SGBDR monobase (Oracle). Or l'arrêt d'une base, ou pire encore d'un serveur, vide le cache ce qui a pour conséquent de rendre très lent le SGBDR au redémarrage. Une telle méthode n'est donc jamais à conseiller si l'on veut conserver des performances uniformes.
Mais c'est hélas encore le seul mode de sauvegarde disponible pour certains SGDBR comme MySQL ou SQL Lite pour faire des sauvegardes consistantes, c'est-à-dire conservant cohérence et intégrité.

### 6.1.4 - Sauvegarde complète de la base à chaud

La sauvegarde d'une base de données à chaud consiste à capturer toutes les pages de la base ainsi que toutes les transactions effectuées depuis le début de la sauvegarde.
Le processus commence par insérer une marque dans le journal de transactions, puis parcoure la mémoire pour lire les pages à écrire dans le fichier de sauvegarde et complète par les pages lues directement sur le disque pour celles ne figurant pas dans le cache. Enfin, il recopie les transactions ayant eu lieu depuis le démarrage de la sauvegarde dans l'espace de sauvegarde.

### 6.1.5 - Sauvegarde incrémentale de la base à chaud

La sauvegarde incrémentale est le fait de ne sauvegarder que les pages ayant évoluées depuis la dernière sauvegarde complète si une sauvegarde incrémentale n'a jamais eu lieu, ou bien depuis la précédente sauvegarde incrémentale. C'est le serveur lui-même qui indique, soit par un tag binaire dans les pages si elles ont été modifiées depuis lors, soit par des pages spéciales dans la base indiquant quelles pages ont évolué.
De la même manière que pour la complète, les transactions ayant eu lieu pendant le processus de sauvegarde sont copiées dans l'espace de sauvegarde.

### 6.1.6 - Sauvegarde différentielle de la base à chaud

La sauvegarde différentielle est le fait de ne sauvegarder que les pages ayant évoluées depuis la dernière sauvegarde complète et les transactions ayant eu lieu pendant le déroulement du processus. Chaque sauvegarde différentielle contient donc l'ensemble des mises à jour ayant eu lieu depuis la dernière sauvegarde complète. C'est à nouveau le serveur lui-même qui indique quelles pages ont évolué.

### 6.1.7 - Sauvegarde transactionnelle de la base à chaud

Une telle sauvegarde consiste à ne copier dans l'espace de sauvegarde que l'ensemble des transactions ayant eu lieu depuis la dernière sauvegarde quelle qu'elle soit (complète, incrémentale, différentielle ou transactionnelle).

Pour que le serveur puisse s'y retrouver dans l'enchaînement de toutes ces sauvegardes, chacune d'elle incorpore une marque transactionnelle (LSN ou Log Segment Number) afin de permettre au serveur de vérifier la continuité transactionnelle. Ainsi l'oubli d'un des fichiers de sauvegarde au moment de la restauration se traduirait par une erreur indiquant un chainage impossible des opérations.

### 6.1.8 - La sauvegarde partielle... Une hérésie ?

Le concept de sauvegarde partielle n'existe pas en base de données. Une base formant un tout et incorporant des règles de cohérence, il est évident que la sauvegarde d'une table n'a pas de sens puisque la restauration ne sera pas possible si les contraintes de cohérence (notamment intégrité référentielle) ne sont pas respectées (notons qu'une restauration partielle n'est pas possible par ce biais). Néanmoins, certains SGBDR offrent la possibilité d'effectuer des sauvegardes partielle. Schématiquement il s'agit de sauvegarder soit complétement, soit incrémentalement, soit différentiellement tout un espace de stockage à condition que cet espace de stockage ne contienne que des objets n'ayant aucun lien d'aucune sorte avec d'autres objets stockées dans un autre espace. Par exemple si l'on considère une table de code postaux servant simplement d'aide à la saisie du code postal, mais sans nécessairement l'implémentation de quelconques contraintes logiques, alors, si cette table figure dans un espace de stockage à part, on pourra effectuer des sauvegardes partielles de la base à des fréquences très diverses, par exemple une fois par jour pour les données dynamiques de production et une fois par an pour les données quasi statiques comme les codes postaux.

### 6.1.9 – Techniques avancées

Les poids lourds des SGBDR que sont Oracle, MS SQL Server ou IBM DB2, peuvent effectuer des sauvegardes avec des options avancées tel que :

- placement de la sauvegarde sur plusieurs fichiers : une même sauvegarde est ventilée dans plusieurs espaces de stockage (par exemple plusieurs lecteurs de bande) en parallèle, ce qui permet de réduire notablement le temps de la sauvegarde;
- la duplication de la sauvegarde : la même sauvegarde est envoyée sur plusieurs destinations, par exemple sur un lecteur de bande local et en même temps sur un disque distant afin de multiplier les possibilités de restauration ;
- la compression de la sauvegarde : lors des opérations de lecture, une compression est effectuée à la volée ce qui permet de réduire le volume de la sauvegarde, mais aussi et par conséquence, le temps de la sauvegarde, car l'écriture sur le support de destination est plus concentrée ;
- le chiffrement de la sauvegarde : il s'agit de crypter à la volée les informations à sauvegarder afin de conserver une sauvegarde indéchiffrable ;
- la vérification binaire de la sauvegarde au cours du processus : chaque écriture est relue et comparée en mémoire afin de savoir si la sauvegarde est intègre ;

La plupart du temps, plusieurs modes peuvent être combinés.

**Exemple 11.22** - sauvegarde compressée, répartie sur deux fichiers et deux destinations avec MS SQL Server

```
BACKUP DATABASE DB_SYNTHEX
TO        DISK = 'C:\MesSauvegardes\DB_SYNTHEX.bak',
          DISK = 'D:\MesSauvegardes\DB_SYNTHEX.bak'
MIRROR TO DISK = '\\FILER_BAK\DATABASES\SQL_Server\DB_SYNTHEX1.bak',
          DISK = '\\FILER_BAK\DATABASES\SQL_Server\DB_SYNTHEX2.bak'
WITH COMPRESSION;
```

### 6.1.10 - Élaboration d'un plan de sauvegarde

Un plan de sauvegarde est l'ensemble des techniques de sauvegardes et leur planification pour une même base de données.

**Exemple 11.23** – plan de sauvegarde pour une base de production

- Chaque lundi à 1h du matin, lancement d'une sauvegarde complète. Durée estimée : 4 h ;
- Du mardi au dimanche à 1 h du matin, lancement d'une sauvegarde différentielle. Durée estimée : 20 minutes ;
- Tous les jours de la semaine, de 8h à 20h, toutes les 30 minutes, lancement d'une sauvegarde transactionnelle. Durée estimée : 40 secondes.

Dans un tel cas de figure, la restauration d'une base le jeudi 7 juin 2012 à 11h 28 consiste à effectuer successivement :

- La restauration de la complète de lundi 4 juin 2012 ;
- La restauration de la différentielle de jeudi 7 juin 2012 ;
- La restauration dans l'ordre chronologique des 7 transactionnelles de 8h, 8h30, 9h, 9h30, 10h, 10h30 et 11h du jeudi 7 juin 2012.

## 6.2 - La restauration

Il n'y a aucun intérêt à effectuer des sauvegardes, si l'on n'a pas une fois dans sa vie testé la restauration. En effet, la sauvegarde est une opération relativement banale, tandis que la restauration est souvent une opération d'urgence qui s'effectue généralement dans la douleur !
Pour l'avoir vécu plus d'une fois, le scénario est le suivant : la base vient d'être victime d'une erreur logique : après la mise en production d'une version mal testée, la colonne prix de la table des produits a été mise à zéro. Les utilisateurs s'en donnent à cœur joie pour piller le site web. Il faut donc l'arrêter et revenir dans un état normal des données en recherchant dans une sauvegarde les prix originaux. Voici une liste de quelques-uns des problèmes auxquels vous allez être confronté dans une telle situation :

- Ou sont mes fichiers de sauvegarde ?
- Quelles versions de mes fichiers de sauvegarde dois-je prendre en considération ?
- Combien de temps est-il nécessaire pour rapatrier les sauvegardes sur le serveur ?
- Ais-je assez de place sur le serveur pour y placer tous les fichiers de la restauration ?
- Les fichiers de sauvegardes sont-ils intègres (ont-ils été vérifiés) ?
- Le chaînage des sauvegardes est-il sans discontinuité ?
- Combien de temps va mettre la restauration ?
- Dois-je écraser la base actuelle ou faire une restauration à côté ? (pour les serveurs monobase comme Oracle ou PostGreSQL ceci est un véritable casus belli)
- Ou dois-je placer la nouvelle base restaurée si l'ancienne doit coexister sur le serveur ?

Pour couronner le tout, dans le cas de figure précité, votre mail va rapidement être saturé de demandes d'explications, de coup de fils interrogateurs et votre bureau va être fréquenté par tous les acteurs important de l'entreprise qui vont venir vous voir tour à tour pour savoir ce qui se passe. Le directeur commercial comme le PDG, vont vous mettre sous pression et le stress sera maximum. Bref, si c'est à cette occasion que pour la première fois de votre carrière de DBA, vous commencez à vous intéresser au manuel de restauration, votre job d'administrateur de base de données a toutes les chances de finir à cet instant !

### 6.2.1 – Un cas pratique

Voici par exemple et pour ce cas de figure le processus de restauration à effectuer lorsqu'un plan de sauvegarde a été établi avec une sauvegarde complète hebdomadaire, une différentielle tous les jours et des sauvegardes transactionnelles toutes les heures, avec l'incident se produisant à 11h13 alors que vous en êtes informé à 11h20, sur un serveur base multibase comme MS SQL Server :

- sauvegarder le journal de transaction actuel de la base fautive ;
- rapatrier la sauvegarde complète la plus récente ;
- rapatrier la sauvegarde différentielle la plus récente ;
- rapatrier toutes les sauvegardes transactionnelles entre l'heure de la dernière sauvegarde différentielle et maintenant ;
- restaurer la complète avec un nom de base différent et un emplacement de stockage autre que celle de la base d'origine, en mode d'attente (NO RECOVERY);
- restaurer la différentielle en mode d'attente (NO RECOVERY) ;
- restaurer toutes les sauvegardes des journaux de transaction, sauf le dernier, en mode d'attente (NO RECOVERY) ;
- restaurer le dernier journal de transaction (celui que vous venez de sauvegardé au début de ce processus) en mode final (RECOVERY) avec arrêt à 11h12 (STOPAT) ;
- effectuer une requête UPDATE interbase pour retrouver les prix originaux depuis l'ancienne base pour mettre à jour la base de production ;

À ce stade, la base ou le site web peut être remis en ligne…

### 6.2.2 – Quelques options particulières

Dans cet exemple, nous avons évoqué quelques options particulières… Décrivons-en quelques-unes :
**Remplacement** (REPLACE) : remplace par écrasement des fichiers, la base existante, si tel est le cas.
**Non récupération** (NO RECOVERY) : spécifie qu'après la restauration, la base est en mode d'attente, c'est-à-dire que la base n'est pas en ligne car on a prévu d'appliquer des sauvegardes complémentaires, par exemple transactionnelles. En principe, à défaut, le mode est RECOVERY, c'est-à-dire que la base est mise en ligne immédiatement et devient opérationnelle pour toutes les transactions (récupération).
**Arrêt à** … (STOPAT / UNTIL) : spécifie que l'on va arrêter la sauvegarde transactionnelle à une date/heure précise (en général juste avant un incident) ou à une marque transactionnelle (Log Segment Number par exemple). Cette technique particulière, qui permet d'arrêter la restauration avant un incident s'appelle génériquement la récupération à un point dans le temps ou PITR (*Point In Time Recovery*).
**Partielle** (PARTIAL) : spécifie que chaque fichier de restauration est mise en ligne immédiatement avant de poursuivre le mécanisme de restauration des autres espaces de stockage (une restauration doit en principe être complète). Ce qui permet de « voir » une partie des données avant même que la restauration soit terminée.

**Exemple 11.24** – restauration à un point dans le temps avec Oracle. La base doit préalablement été positionnée en journalisation ARCHIVELOG et être à l'arrêt au moment de l'opération :

```
sql "alter session set nls_date_format = 'DD-MON-YYYY HH24:MI:SS'";
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
RUN
    {
      SET UNTIL TIME '21-OCT-2012 11:22:00';
      RESTORE DATABASE;
      RECOVER DATABASE;
    }
ALTER DATABASE OPEN RESETLOGS;
```

### 6.2.3 – Restauration par rapiéçage

Sur les grands SGBDR comme Oracle ou MS SQL Server il est possible d'effectuer des opérations de restauration par « rapiéçage » (*piecemeal restoration*), c'est-à-dire remplacer par exemple une page défaillante. C'est utile dans le cas d'une base de données de plusieurs Tera octets, car le temps de

restauration de l'ensemble de la base serait très supérieur à celui du remplacement d'une page. La technique est alors la suivante :

- identification de la page défaillante (par vérification des espaces de stockage);
- mise hors ligne de la base ;
- sauvegarde du journal de transaction ;
- recherche de la dernière sauvegarde complète et application de la restauration de la page en mode d'attente ;
- recherche de la dernière sauvegarde différentielle et application de la restauration de la page en mode d'attente ;
- recherche de toutes les sauvegardes transactionnelles intermédiaires et application de la restauration de la page en mode d'attente ;
- application de la dernière sauvegarde transactionnelle effectuée en début de processus et application de la restauration de la page en mode final ;
- remise en ligne de la base.

### 6.3 - L'envoi de journaux par « log shipping »

Moyen ancestral et simplissime, le *log shipping* (de *to ship* expédier et *log* journal de transaction, appelé couramment envoi des journaux de transaction en français) consiste à restaurer sur un serveur annexe une base de données par application des sauvegardes de journaux de transaction successives.
La technique est la suivante :

- On restaure une sauvegarde complète sur un serveur de secours ;
- On met en place une sauvegarde transactionnelle planifiée à intervalle régulier sur le serveur de production (par exemple toutes les 20 minutes) ;
- On envoi (par exemple à l'aide de FTP – File Transfer Protocol) les fichiers de ces sauvegardes dès qu'elles ont été effectuées ;
- Le serveur de secours intègre les sauvegardes transactionnelles en continu en mode d'attente (NORECOVERY), sauf la toute dernière ;
- En cas de perte du serveur de production le DBA restaure la dernière sauvegarde transactionnelle en mode de récupération (RECOVERY)

Le serveur de secours est alors en état de produire. Il ne reste plus qu'à rerouter les applications.

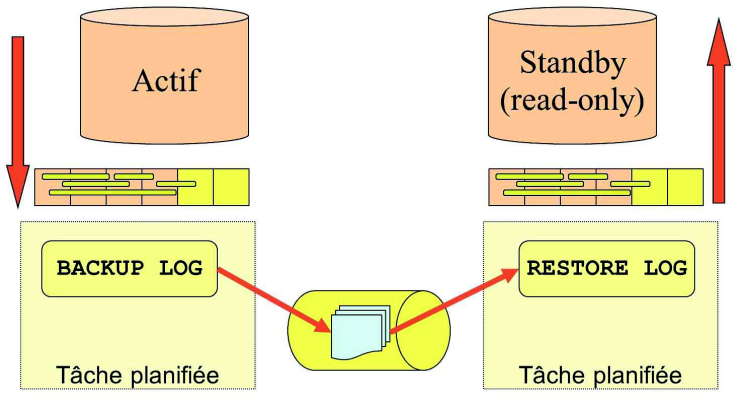


*Figure 11.11 – Principe du log shipping*

Dans certains cas, la base sur le serveur de secours peut être utilisée en lecture. Sur les serveurs multibases, d'autres bases peuvent coexister.
Le seul problème de cette méthode est qu'elle ne doit pas interférer avec le plan de sauvegarde ordinaire, car le mélange des genres pourrait conduire un jour ou l'autre à une problématique de