Utiliser des bases de données de Firebird dans le système d'informations des entreprises

Helen Borrie
Traduction en français: Philippe Makowski

15 Juillet 2006 Document version 1.2-fr

Table of Contents

Qu'est-ce que Firebird?

Firebird est système de base de données relationnel, comparable à des produits comme DB2 d'IBM, Oracle, SQL Server de Microsoft et le produit open source PostGreSQL. Le logiciel a deux principaux composants : le serveur de bases de données, qui est installé sur la même machine que les bases de données et l'interface applicative, communément appelée la “bibliothèque client”. La bibliothèque client est un composant — une DLL sous Windows ou un objet partagé (.so) sur les autres plates-formes — nécessaire sur chaque station cliente dans le cadre d'un déploiement deux-tiers. Pour les déploiements multi-tiers, quand les utilisateurs accèdent aux bases de données à travers un middleware depuis un navigateur web ou autre “client léger”, la bibliothèque cliente Firebird n'est pas déployée sur les stations des utilisateurs mais uniquement au sein du middleware.

Le serveur Firebird laisse une faible “empreinte” dans le système de fichiers quand il est installé sur la machine serveur. L' exécutable fait moins de 1.5 Mb et une installation complète, avec les outils et la documentation, prend moins de 10 Mb. L'occupation mémoire variera en fonction du déploiement, qui peut aller d'une application mono utilisateur utilisant une seule base de données à des centaines de connexions concurrentes vers de multiples bases de données servant des centaines d'utilisateurs au sein d'un large réseau.

Firebird est maintenu et développé par une communauté de développeurs du monde entier. C'est un projet open source, non commercial appartenant aux développeurs. Étant distribué complètement libre d'honoraires, le droit d'exploitation n'est source de revenus pour personne.

Historique de Firebird

Les versions de production de Firebird sont disponibles depuis début 2002, mais l'ascendance de ce logiciel est bien plus ancienne. C'était InterBase en 1985, pour la plate-forme Unix VMS de cette époque. Il est devenu la propriété de Borland Software Corporation au début des années 1990 par le biais d'acquisitions et a évolué au fil des années jusqu'à la version 5.6. Fin 1999, la situation financière de Borland a causé l'arrêt du développement de la version 6. L'année suivante, le code source d'InterBase 6 a été rendu publique sous licence open source en Juillet 2000. Deux développeurs australiens ont téléchargé le code fraîchement rendu disponible et ont mis en place le Projet Firebird sur Sourceforge, une importante ferme de serveurs qui procure des outils sophistiqués, gratuitement, pour les projets open source.

La livraison initiale

La période qui s'est écoulée entre l'arrêt du développement d'Interbase chez Borland et la mise à disposition du code source a permis de transformer la base des anciens développeurs et utilisateurs d'InterBase en une vivante et enthousiaste communauté Firebird constituée de concepteurs, testeurs et développeurs d'outils et d'interfaces et de gourous de support. Quand le code source a été libéré, une équipe conséquente était prête à travailler dessus. Firebird n'a jamais regardé en arrière. Firebird 1.0 — essentiellement un nettoyage du code en langage C d'IB 6 avec quelques importantes corrections pour stabiliser la production des binaires et quelques anciens bugs — a été livré en 2002 et a connu quatre sous-versions.

Version stable actuelle

Firebird 1.5, livré la première fois en mars 2004, était une révision complète du code C vers C++ pour préparer la voie à des améliorations architecturales essentielles prévues pour Firebird 2. La version de production la plus récente, Firebird 1.5.3, est très stable et a bénéficié d'améliorations en provenance du développement de Firebird 2.

En développement

Firebird 2

Firebird 2, qui bénéficie d'améliorations importantes dans de nombreux de ses sous-systèmes, y compris l'optimiseur SQL, est actuellement en version RC3. La version finale est prévue pour l'été 2006.

"Vulcan"

Un "fork", une branche, a été créé depuis le premier code alpha de Firebird 2 en décembre 2003 afin de reconcevoir l'architecture de threading du moteur de base de données. Le projet, implémenté par le développeur originel d'InterBase (Jim Starkey) était commandé par SAS Institute, le leader du marché des applications logicielles de statistiques commerciales et médicales. SAS a pris la décision en 2003 de migrer plusieurs de ses applications commerciales d'Oracle vers Firebird. Les sources, nom de code "Vulcan", ont été formellement reversées au projet Firebird en 2005 et continuent à être développées en parallèle avec Firebird 2. La première livraison publique d'une version beta de Vulcan est prévue pour mi 2006.

Firebird 3

La fusion des codes de Firebird 2 et Vulcan a commencé, avec l'objectif de livrer Firebird 3 début 2007. Firebird 3 aura un support complet de gestion des thread sur des machines multi-core et multi-CPU, une utilisation complète des fonctionnalités des systèmes 64-bit et de nombreuses options de configuration du niveau de sécurité du serveur et des bases de données.

Firebird est il “Prêt pour l'entreprise”?

Définir l'expression “Prêt pour l'Entreprise” est beaucoup plus difficile que compter le nombre de réponses sur Google. La presse en ligne utilise cette expression comme si elle était clairement définie, comme “nouveau né” ou “détaxé”. On en conclut que cette chose éphémère, quelque chose que tout le monde désire pour un logiciel de bases de données, est, soit présente, soit absente, et ne peut être obtenue que dans des produits commerciaux.

Pour aborder la question de manière constructive, la première chose est de trouver un contexte d' “entreprise” qui corresponde au cas traité. Les deux questions rationnelles à ce poser sont : Quels besoins, présents et futurs, de l'Entreprise X doivent être satisfaits par les fonctionnalités du système de gestion de bases de données ? Le SGBD Y y répond-il?

Ceux d'entre nous qui sommes du côté des utilisateurs plutôt que du côté des critiques de la presse ex-

amineront les fonctionnalités nécessaires pour l'entreprise à travers six filtres : stabilité, extensibilité, disponibilité, capacité, interopérabilité et autonomie.

Stabilité

Nous voulons un système de gestion de bases de données qui enregistre les données nécessaires à notre activité et qui les protège contre les dégradations possibles en provenance soit de problèmes liés à notre environnement système soit d'erreurs humaines. Notre système doit être capable de délivrer les données aux applications de manière consistante et flexible, avec certitude et à une vitesse raisonnable et doit être capable de gérer d'éventuels conditions de conflits. Il doit pouvoir faire tout cela en même temps, sans interruptions, dégradations, crashes ni temps d'attente excessif.

“Conformité ACID”

Une telle stabilité est évidente pour un SGBD, quelques soient les autres facteurs désirés par l'entreprise. Un ensemble de principes a évolué au cours du temps pour définir quatre points essentiels qui ne doivent pas manquer à un système de gestion de bases de données pour être pris au sérieux. “ACID” est l'acronyme de ces quatre principes : Atomicité, Cohérence, Isolement, Durabilité.

La conception et l'architecture de Firebird respectent complètement la norme “ACID”. Les concepts ACID sont décrits plus bas, dans la section Conformité ACID et Firebird, avec des commentaires sur comment Firebird respecte ces principes.

Tout dans Firebird est fait dans le contexte d'une transaction ACID. Une transaction dans Firebird peut mettre en oeuvre de nombreuses phases, complexes et inter-dépendantes d'une tâche d'entreprise mais ne permettra jamais la violation d'aucun principe ACID.

Antécédents et support

Ayant en tête la stabilité de notre environnement de production, nous ne choisirons pas forcément un tout nouveau SGBD pour être le coeur de notre système d'information. En général, nous préferons commencer avec un produit qui a prouvé ses capacités et dont les forces et faiblesses sont connues et bien comprises. Si un utilisateur final doit être en charge de répondre aux problèmes qui peuvent arriver, nous devons savoir à qui ils peut demander de l'assistance. Si un distributeur de solution logicielle doit assurer le support technique, est-ce que ce distributeur peut être en mesure d'obtenir le support adéquat ?

La communauté des utilisateurs de Firebird est bien aidée par des développeurs d'outils et d'applications très compétents. En général, ces développeurs sont proches du code source développé depuis six ans. Leur expérience et leur expertise remonte même au delà, au début des outils de développement de Borland, qui ont toujours été livrés avec la version d'InterBase "Développeurs" dans leur version Entreprise.

La communauté de développeurs Firebird est renommée pour ses forums de support, où l'expertise y est constamment partagée.

Des contrats de support pour utilisateur finaux, même s'ils sont offert par plusieures entreprises, sont rarement demandés. Des contrat de support pour les développeurs sont disponibles dans de nombreux pays, y compris la France. Toutefois, il doit être signalé que la demande de support pour les développeurs, mise à part peut être la phase d'installation sur une plate-forme non familière, est faible dans la plus part des pays.

Développements futurs

D'un autre coté, parce que l'activité, les demandes et les configurations matérielles évoluent constamment et rapidement, nous voulons aussi savoir si le SGBD est dans une phase active de développement ou s'il est proche de la fin de sa vie. Nous espérons des mises à jour régulières de versions intermédiaires et d'avoir des signes qu'une nouvelle version majeure est en cour de réalisation. Quand nous serons en production, sera t-il facile de faire la mise à jour ? Nos bases de données existantes pourront-elles facilement être intégrées dans une nouvelle version, ou bien ce processus sera t-il pénible, ou long, ou presque impossible ?

Maintenant dans sa sixième année, l'équipe du Projet Firebird continue en toute confiance le développement des versions prévues qui vont amener les améliorations architecturales pour profiter des améliorations matérielles et satisfaire les besoins exprimés de la communauté des développeurs, utilisateurs et supporteurs techniques. La transparence du code et la célèbre volonté de la communauté de partager sa connaissance assure une maîtrise parfaite du code source et des possibilités du logiciel.

L'indépendance du projet vis à vis des entités commerciales assure la continuité du projet et assure que la discipline technique prévaut sur une quelconque volonté marketing de peser sur l'évolution du produit.

Extensibilité

Par extensibilité on désigne le fait que le SGBD soit capable non seulement de répondre aux besoins actuels de l'entreprise, mais aussi capable de grandir avec elle, de répondre à des accroissements de charge mais aussi de répondre à des besoins moindres (applications mono utilisateur, modèle déconnecté ou embarqué par exemple).

Les considérations liées à l'extensibilité à prendre en compte sont les nombres minimum et maximum d'utilisateurs actifs, l'effet de la croissance de la taille de la base et du nombre d'utilisateurs sur les performances ou la stabilité, les possibilités de migration des données en cas de changement d'échelle. Pour certaines entreprises, les facteurs d'extensibilité prendront le pas sur toues les autres considérations, comme la disponibilité ou l'interopérabilité.

L'extensibilité quelle que soit sa direction est une des forces de Firebird. Contrairement à beaucoup d'autres systèmes concurrents, Firebird, depuis même ses premiers jours sous le nom d' InterBase, a toujours été un logiciel conçu pour le fonctionnement en réseau, et sa structure de stockage sur disque des bases de données a toujours été gérée indépendamment du système de fichier hôte, par le moteur de base de données lui même.

A la différence de certains systèmes très valorisés sur le marché, supposés être “prêts pour l'entreprise”, qui répondent à la demande d'extensibilité en cas de croissance des besoins par des ajouts logiciels et des mécanismes nombreux et lourds, la question de la croissance avec Firebird est essentiellement seulement une question de modification de l'environnement. Le même moteur gère confortablement toutes les situations depuis un système embarqué avec un seul utilisateur, en passant par le système classique d'une architecture deux tiers client/serveur en réseau avec environ 750 utilisateurs potentiels, jusqu'à son implication dans une solution multi-tiers pour des centaines de clients potentiels. La croissance des bases de données est seulement limitée par la capacité de stockage disque disponible et une base peut être découpée pour être placée sur plusieurs disques durs.

A l'aide d'une réplication intelligente et une bonne gestion des connexions d'accès, la charge d'un système très chargé peut être distribuée sur de multiples serveurs. Par exemple, un serveur central bien dimensionné peut servir les demandes d'accès du réseau, intranet ou extranet (ou les deux ensemble) pendant qu'un serveur répliqué prend en charge les traitements longs qui demandent un cliché

isolé des données pendant une longue période.

Disponibilité

Le maximum de disponibilité est parfois mesuré par le critère des “cinq neuf” : le serveur est il capable d'offrir une disponibilité à 99.999 pour cent pendant les heurs de service ? Parmis ces utilisateurs, Firebird a la réputation d'être parfaitement disponible.

Certains des systèmes de gestion de données demandent une coûteuse expertise sur site pendant tout le temps de fonctionnement du système de bases de données. Le déploiement typique de Firebird est celui des entreprises qui n'ont pas d'administrateur de bases de données (DBA).

Comme pour tout système complexe, un déploiement de Firebird doit être bien planifié et bien conçu, mais un déploiement de Firebird configuré convenablement dans une infrastructure réseau efficace “fonctionne tout seul”. Il utilise un système de verrous optimiste au niveau des enregistrements, qui réduit considérablement le temps d'attente en comparaison avec d'autres systèmes où des transactions en lecture-écriture verrouillent des ensemble entiers, voire des tables, de manière préventive. Aucun réglage fin n'est requis pour gérer les différences de charge au cours du temps. Une base n'a pas besoin d'être arrêtée pour être sauvegardée. Elle peut être répliquée ou dupliquée pour prévenir les coupures dues à un crash du disque. Firebird est robuste et se remet en route immédiatement après une coupure de courant, sans dommage pour l'intégrité des bases.

Firebird est un choix populaire pour les entreprises ayant besoin d'une continuité de service 24h sur 24. Des outils en ligne de commande sont inclus dans la distribution pour toutes les tâches administratives, permettant d'automatiser la maintenance régulière dans des tâches programmées. Une API de services est aussi disponible pour inclure des tâches d'administrations dans un autre programme.

Les sauvegardes en ligne (gbak) ne créent pas des bases de données mais des fichiers neutres vis à vis de la plate-forme qui contiennent les métadonnées et les données de manière séparée dans un format texte compressé. Firebird 2 dispose d'un outil de sauvegardes incrémentales, qui peut être planifié afin de s'adapter à la charge du serveur.

Capacité

La plus grande base Firebird que nous connaissons fait environ 11 Teraoctets et continue de grossir. Les tables sont limitées à environ 2.000.000.000 lignes et, jusqu'à la version 1.5.x, à un maximum d'environ 30 Gigaoctets par table. Cette limite maximum de taille de table n'existe plus dans la version 2.0, disponible mi-2006.

Interopérabilité

Conformité aux Standards

Firebird fait partie des SGBD qui respectent le plus les normes internationales (ISO) et U.S. (ANSI) pour le langage SQL, faisant même mieux que son cousin InterBase. Les normes évoluent continuellement et, en parallèle, les développeurs de Firebird font évoluer le produit en concordance.

Les implémentation existantes des fonctionnalités du langage qui font l'objet d'une nouvelle norme sont gardées comme des "options dépréciées" pour garantir la compatibilité descendante.

Indépendance

Firebird est conçu pour l'interopérabilité. Il n'est lié à aucune “solution intégrée” qui lie les bases de données à un environnement spécifique. La décision de déplacer une base de données Firebird depuis Windows vers une machine Linux ou Unix, et vice versa, peut littéralement se mettre en oeuvre du jour au lendemain. Tout ce qu'il faut, c'est une sauvegarde au format transportable sur l'ancienne machine et une restauration sur la nouvelle, et vous voilà de nouveau opérationnel.

Les distributeurs d'applications qui sont conscients de cette facilité de migration sont attentif au fait que les serveurs Firebird peuvent être facilement utilisés par des logiciels clients qui tournent sur un système d'exploitation qui peut être différent de celui sur lequel tourne le serveur et que cette situation peut facilement évoluer.

Firebird est capable de fonctionner dans des environnement réseaux hétérogènes. Une application écrite pour un système d'exploitation peut facilement se connecter à une base tournant sur un autre système d'exploitation, sans modification. Une application peut se connecter à de multiples serveurs sur des hôtes avec des systèmes d'exploitations différents simultanément et peut utiliser des transactions sur des bases multiples sur des serveurs différents.

Il est fréquent, par exemple, pour des sites très utilisés, de faire tourner plusieurs serveurs Firebird, de mettre en place une réplication depuis un serveur principal important vers des serveurs moins coûteux et moins surchargés fonctionnant sous Linux avec des systèmes disques rapides pour mettre en place une solution de délestage (load-balancing) et de prévention des pannes.

Les serveurs Firebird peuvent aussi être intégrés à des solutions hétérogènes de type MTS et Citrix, dans des environnement pass-through et des réseaux privés virtuels (VPN).

Connectivité

Firebird n'a pas besoin de suites ou modules spéciaux pour le lier à une application externe. Toutes les interfaces d'accès dialoguent avec le serveur à l'aide des deux interfaces publiées (APIs), une pour les opérations au niveau des bases de donnée et l'autre pour les opérations serveur comme les sauvegardes et l'authentification des utilisateurs.

Des pilotes sont disponibles pour une grand nombre de langages et d'interfaces, y compris Java/JDBC, ODBC, .NET, Delphi, Python, PHP et Perl.

Autonomie

L'autonomie se mesure par le degré avec lequel une base stocke et gére les données en ayant “sa propre vie”, c'est à dire sa capacité à exister et rester cohérente indépendamment du matériel et du système d'exploitation,des programmes externes, des environnement spécifiques de programmation.

“Fonctionnalités d'autonomie”

“Les fonctionnalités d'autonomie” comportent des point comme

- indépendance du système de fichiers
- intégrités référentielles déclaratives

- la possibilité de mettre en oeuvre des validation et/ou des règles de gestion à l'aide de déclencheurs et des contraintes CHECK
- les procédures stockées, pour intégrer des procédures de gestion et les mettre en action de manière interne, indépendamment du langage des applications ou de l'interface utilisateur
- le contrôle d'accès des utilisateurs au niveau de la base
- la possibilité d'interroger des données externes comme des structures “virtuelles” sans avoir besoin de stocker des données
- la possibilité de gérer des ensembles de données temporaires par programmation sans avoir à matérialiser des tables temporaires
- garder des fonctionnalités dépréciées pour assurer la rétro compatibilité

Firebird prend en charge toutes ces fonctionnalités.

Complexité de migration et mise à jour

Un point supplémentaire d'autonomie qui est devenu un problème pour les utilisateurs de nombreux SGBD évolués dans les temps récents est le degré de transparence avec lequel des anciennes bases de données peuvent être administrées et migrées quand le logiciel serveur est mis à jour ou que les bases sont déplacées vers une nouvelle plate-forme matérielle ou logicielle. Souvent ces problèmes deviennent des opérations très coûteuses en temps et en logistique.

Migration

Au contraire d'autres SGDB significatifs, à part le cousin de Firebird, InterBase, la migration de bases est sans heurts. Contrairement aux autres, qui ont une architecture dur disque différente selon les plate-formes, ou qui ne gèrent leurs bases que sur une seule plate-forme, la structure des bases Firebird est stable quelle que soit la plate-forme. Cela veut dire, par exemple, que si le choix d'un système d'exploitation pour le serveur se révèle être une solution sous-optimale, vous pouvez changer pour un autre serveur avec simplement le temps qu'il faut pour faire une sauvegarde et une restauration.

Compatibilité ascendante

Une nouvelle version de Firebird se connecte à n'importe quelle base de données qui fonctionnait sous une version précédente et une sauvegarde transportable d'une base faite sur une autre plateforme matérielle ou logicielle peut être restaurée, prête à fonctionner, sur n'importe quelle autre plateforme supporté par la nouvelle version.

Quand une mise à jour est faite vers une nouvelle version majeure, il est hautement recommandé, même si ce n'est pas essentiel, de faire passer les bases par le cycle de sauvegarde restauration, afin de mettre à jour la structure de stockage sur disque et rendre les nouvelles fonctionnalités disponibles. La question qui se pose pour savoir s'il faut le faire où non est uniquement liée au fait de savoir si le fournisseur de l'applicatif a déjà mis à jour son application pour utiliser les nouvelles fonctionnalités.

Sous-versions

Normalement, les sous-version ne changent pas la structure de stockage sur disque, toutefois il peut être intéressant de faire un cycle de sauvegarde restauration si la sous-version améliore des fonctionnalités existantes.

Conformité ACID et Firebird

Les quatre principes ACID sont l'atomicité, la cohérence, l'isolement et la durabilité.

Atomicité

L'atomicité garanti qu'il n'y a que deux résultats possibles pour une tâche (que l'on nomme transaction) qui implique des changements de multiples ensembles de données interdépendants : soit toutes les étapes ont été réalisées correctement et le résultat peut être validé dans la base comme une seule entité ou, si l'une des étapes n'a pas été réalisée, toutes les autres étapes doivent être défaites (“rolled back”), laissant l'état de la base inchangé.

L'atomicité est manifestement d'une extrême importance dans les systèmes financiers, où des balances non équilibrées du fait d'échecs partiels seraient une catastrophe. Les tableurs et les bases de données qui ne supportent pas les transactions ne peuvent pas être atomiques.

Firebird est entièrement “piloté par les transactions”: rien ne peut arriver en dehors du contexte d'une transaction.

Cohérence

La Cohérence est la capacité du moteur de bases de données à protéger la base de tous les changements d'état qui pourraient laisser un ensemble de données désynchronisé par rapport à un autre ensemble de données. Par exemple, le système doit être capable de reconnaître qu'une facture est liée à un client et aux éléments facturés. Il doit être capable d'éviter, par exemple, la suppression d'un client s'il existe encore des factures pour ce client, et la suppression d'une facture qui a des éléments associés.

L'implémentation pratique de telles dépendances est faite par la déclaration de contraintes d'intégrité référentielles (“foreign keys”) renforcée par des déclencheurs automatiquement générés. (Les déclencheurs sont automatiquement générés ou définis dans des blocs de code qui s'exécutent quand un enregistrement est inséré, modifié ou supprimé.) Le système de bases de données qui dépend du code d'une application pour gérer des règles de cohérence ne sont pas conformes à la règle de cohérence ACID. Firebird respecte parfaitement les règles d'intégrité définies par la norme.

Firebird garanti aussi la cohérence quand une seule transaction est exécutée pour effectuer des changements sur plusieurs bases à la fois, c'est ce que l'on appelle le “two-phase commit”. Les systèmes qui peuvent accéder à plusieurs bases à la fois de manière concurrente sans la possibilité de synchroniser les changement dans toutes les bases ne respectent pas le principe de cohérence.

> **Note**
>
> Firebird ne gère pas les déclaration d'intégrité référentielle multi-bases. Chaque base impliqué dans une transaction multi-bases est responsable de ses propres contraintes référentielles.

Isolement

L'isolement décrit la possibilité du moteur de bases de données de permettre à chaque utilisateur (ou transaction) d'opérer comme s'il était le seul utilisateur (ou la seule transaction). Le mécanisme d'isolement doit être capable de garder la base cohérente pour chaque transaction tant qu'elle est en cours, indépendamment de tout changement intervenant dans d'autres transactions. Les systèmes de gestion de bases de données qui sont conformes à ce principe offre en général différents niveaux d'isolement, qui sont définis dans les normes ISO/ANSI SQL.

En plus du niveau d'isolement décrit ci-dessus (Concurrency ou Repeatable Read), qui doit être supporté, Firebird supporte aussi Read Committed (quand une transaction peut voir le travail validé par les autres transactions) et, à contrario, Consistency ou Table Stability (quand une transaction qui peut écrire dans la base bloque l'accès aux tables qu'elle utilise aux autres transactions capables d'écrire).

Durabilité

La durabilité garanti que la base va garder trace des changements en cours de manière que l'état de la base ne soit pas affecté si une transaction est interrompue. De sorte que, même si le serveur de base de données est débranché en plein milieu d'une transaction, les serveurs de base de données conformes à la norme ACID doivent remettre la base dans un état cohérent à leur redémarrage.

Journal des Transactions

La durabilité est l'un des principes les plus difficile à respecter. Les autres systèmes de base de données qui se disent ACID gèrent traditionnellement cela en enregistrant les transactions non validées dans un journal des transactions. Cependant, l'utilisation d'un journal de transaction n'a jamais totalement garanti la durabilité, puisque le fichier journal lui même peut être lui même corrompu logiquement ou physiquement par l'événement qui a interrompu la transaction.

Certains des SGBD qui reposent sur une solution de fichier journal pour satisfaire à la durabilité essaient de réduire les risques en utilisant des “write-ahead log” pour enregistrer les informations avant d'essayer de valider les changements. Si ce type de journal survit sans dommage, il peut être possible de retrouver le travail non validé quand le système se remet en marche et utiliser cette information pour remettre la base dans un état stable et la remettre dans l'état où elle était avant le problème. De tels systèmes sont caractérisés par la nécessité d'avoir de longues “procédure de remise en état” après des erreurs réseau ou des pannes électriques.

Certains produits SGBD sophistiqués sont notoirement connus pour leurs problèmes liés à la restauration à l'aide des journaux générés lors de l'interruption des transactions. L'instabilité de ces moteurs de bases de données est telle que, même sur des sites avec des équipements moyens, il est nécessaire d'employer des équipes pour surveiller en permanence le serveur pour éviter les problèmes et corriger les incohérences avant que les problèmes ne se propagent trop loin et assurer l'intégrité des données.

L'Architecture Multi-Générationelle de Firebird

L'architecture de Firebird permet de se passer de fichiers journaux car elle garde la version précédente de chaque enregistrement modifié ou supprimé, pas seulement le temps que la transaction existe, mais tant que toutes les transactions qui étaient “intéressées” par cet enregistrement, quelle qu'en soit la

raison, aient fini leur travail. Le terme utilisé pour cela est “architecture multi-générationelle”, ou MGA.

MGA était une spécificité unique d'InterBase pendant environ 10 ans jusqu'à ce que cela soit imité par Oracle. Une fois que le code source de Firebird a été disponible, PostGreSQL l'a copié. Plus récemment, Microsoft a introduit MGA dans la dernière évolution de SQL Server.

Journal des transactions et Firebird

Firebird n'a pas besoin de journal de transaction pour effectuer des réparations et il n'a pas de fonctionnalité de journaux de transactions dans son moteur. Toutefois, si une entreprise à besoin d'enregistrer des journaux pour des besoins d'audit, d'excellents outils existent et sont disponibles chez des distributeurs tiers de solutions logicielles.

Qui utilise Firebird?

Parce que Firebird est gratuit, il n'y a pas de licences, pas de ventes à compter. On sait, par des sondages d'entreprises réputées, que Firebird fait son travail dans des centaines de milliers de sites de production dans le monde entier. La sélection suivante est faite d' entreprises et d'organisations qui sont publiquement connues pour utiliser Firebird :

- Broadview Software Ltd, Toronto, Canada, distributeur de systèmes d'information, de contrôle et de services en ligne pour les chaînes télévisées dans le monde entier.
- Morfik P/L, Hobart, Tas., développeurs et distributeurs de WebOS, suite de développement pour la réalisation et maintenance de sites web interactifs, stocke les objets web dans des bases Firebird, de même que les données utilisateurs.
- Communicare Systems Pty Ltd, Perth, WA, distributeur de solutions de gestion des patients pour les hôpitaux, cliniques, praticiens médicaux et unités de santé mobiles dans toute l'Australie.
- “The Examiner” newspaper, Launceston, Tas., système haute disponibilité (24/7) pour la gestion des informations.
- U.S. Navy, large usage dans des systèmes de logistique et d'organisation.
- Frontrange Solutions USA Inc., Colorado Springs, U.S.A, comme support d'un outil de CRM complet, puissant et compétitif, adaptable à toutes les organisations : “Goldmine” software suite.
- British Rail, U.K., système de gestion du temps, réservations, facturation et d'information pour les transports ferroviaires de passagers.
- Deutsche Presse-Agentur GmbH, HQ in Hambourg, Allemagne, la plus grande agence de presse d'Allemagne, propose un service planétaire pour les journaux, magazines, TV et radios.
- KIMData, Munich, Allemagne, système de solutions décisionnelles pour les hôpitaux allemands.

Exemples de déploiement

Distributel, Fournisseur de services de télécommunications

Lieux: Canada

Contact: Dalton Calford (CTO)

Distributel est un fournisseur de services pour les appels téléphoniques longue distance avec trois principaux établissements situés dans trois villes différentes et deux provinces. Firebird est le support utilisé pour le système d'information interne, utilisé par une moyenne d'environ 500 utilisateurs différents. Toutefois, le système d'information interne n'est pas le domaine pour lequel nous utilisons Firebird le plus intensément.

Le vrai stress vient de la charge générée par nos clients. Nous fournissons une large variété de services, pour des centaines de milliers de clients, qui génèrent 2 millions de transactions par jour. Le tout étant géré par une base de données, efficacement.

Nous avons trois Points de Présence réseau (“PoPs”), chacun dans une ville différente, plus un PoP de développement dans notre laboratoire de tests. Chaque PoP a un double switch de télécommunication qui fait du délestage et sert de secours en cas de défaillance d'un autre.

Cet équipement est vraiment spécifique pour nos besoins, mais nous le contrôlons en utilisant Firebird. Chaque switch est connecté à deux processeurs de contrôle de signal (SCPs), qui sont des petits ordinateurs utilisant Firebird. Cela veux dire que chaque PoP a quatre SCPs. Chaque SCP héberge deux bases de données différentes, ce qui veut dire que nous avons 32 bases de données différentes, toutes ayant la même structure et contenant les mêmes données.

Si nous perdons un PoP, les autres PoPs prennent en charge l'appel, en fonction de l' “état de l'appel”. Parmis les 96 états d'appels différents, seulement quatre ne sont pas récupérables. Chaque ville garde les dialogues redondants avec les SCPs des autres villes et les dialogues de réponses sont analysés en temps réel.

Pour donner une indication des temps de réponses nécessaires, quand vous décrochez le téléphone, un signal va vers le switch de télécommunication, qui demande à un SCP ce qu'il doit faire. La réponse habituelle est 'donne la tonalité et attend les chiffres composés'. La tonalité n'est pas automatique — c'est la réponse d'une requête dans la base Firebird qui vérifie l'identifiant de la ligne, le statut du client, les indications de services (comme une réponse à un appel) de même que les garanties gouvernementales et les données privées.

Du fait de la loi, nous n'avons droit qu'à un échec pour 99,999 appels, ce que l'on appelle les “cinq neuf” qui définissent le fait d'être “prêt pour l'entreprise”.

Notre service client est aussi impliqué dans la mise à jour des bases de données. Ces bases de données fournissent des informations en temps réel à notre support client, ainsi qu'à notre système de facturation, une base de données Firebird. En plus de stocker et générer les données des comptes clients, comme on peut s'en douter, la base de données est aussi continuellement interrogée par notre unité de relation clientèle, dont les employés sont à la disposition des clients pour les informer sur l'état de leur compte et leur historique des appels.

Bibliothèque municipale de Prague

Lieux: République Tchèque

Contact: Ondrej Cerny, IT department manager

La bibliothèque municipale de Prague gère environs 3.000.000 de publications et a environ 120.000 utilisateurs réguliers (enregistrés). La bibliothèque centrale dispose de nombreuses annexes dans la ville, 20 d'entre elles sont actuellement connectées au site central. Le déploiement est en cours, avec deux ou trois annexes supplémentaires connectées par mois, chacune représentant environs 5-10 nouveaux utilisateurs utilisant 10-50 nouvelles connexions.

La bibliothèque utilise 20 applications avec une seule base de données. Cinq sont considérées comme des applications principales utilisées par presque tous les utilisateurs, gérant les opérations usuelles d'une bibliothèque, terminaux publiques d'accès (dans les bibliothèques) et accès publique via Internet aux collections. Ces applications sont utilisées par 300-350 utilisateurs concurrents pendant les heures d'ouverture, utilisant entre 400 et 600 connexions à la base, qui a actuellement une taille d'environ 30GB. Firebird gère 3-5 millions de transactions chaque jour. Les autres applications sont spécialisées ou sont des tâches de fond (envoi d' e-mails au sujet des livres demandés etc.).

Nous utilisons Linux Classic Firebird 1.5.2 sur Red Hat 9 avec un noyau préparé pour gérer les 400-600 instances. Le matériel est un 4-CPU Xeon avec 16Gb RAM et 120Gb RAID 10 de stockage. Nous avons aussi un 8-CPU Xeon avec 20Gb RAM et 500Gb RAID 10 pour les besoins futurs — précisons que tout cela est encore en déploiement, donc les besoins grandissent tout le temps. La bibliothèque ne fonctionne pas 24h sur 24 7 jours sur 7. Il y a des plages de maintenance de minuit à 5 heures tous les jours, mais le système doit fonctionner sans arrêts tout le reste du temps. Les arrêts potentiels ne sont pas critiques et donc il y a un plan de secours en place pour restaurer les données en moins de deux heures si le système principal subissait un problème critique.

Bien que très satisfait de Firebird, il reste des problèmes avec les anciennes applications qui utilisent le Borland Database Engine (BDE), une couche d'accès aux données obsolète conçue pour accéder à des bases bureautiques comme Access, qui n'a jamais très bien fonctionné en réseau et est particulièrement faible pour gérer les données de manière transactionnelle. Le plus gros problème est le blocage du nettoyage des données périmées qui oblige à faire des sauvegardes/restaurations toutes les nuits et oblige à surdimensionner le matériel pour compenser les dégradations de performances du fait des données périmées accumulées tous les jours.

En dépit de l'architecture non optimale actuelle des applications, la machine 4x Xeon n'est utilisée qu'à 50 pour cent. Ces vieilles applications BDE doivent être remplacées cette année par de nouvelles applications conçues en architecture trois-tiers avec des poll de connexion, utilisant une interface d'accès direct aux données dans le middleware Delphi . Nous espérons avec cette mise en place, des gains substantiels de réserve matérielle avec l'équipement actuel, assez pour garantir la croissance des besoins d'accès pour les cinq prochaines années.

OneDomain, niche de système de business intelligence

Lieux: Birmingham, Al, U.S.A.

Contact: Ed Salgardo Snr

OneDomain est une entreprise à croissance rapide implantée à Birmingham, Alabama qui développe et vend des logiciels de media planning, recherche, et de business intelligence aux chaînes de télévision dans tous les Etats Unis. Le principal produit, appelé ClearView, permet aux vendeurs d'espaces publicitaires d'analyser les audiences TV et définir les cibles pour les annonceurs. Après

notre création en octobre 2001, les deux premières années ont été utilisées pour le développement du produit. Seulement 24 mois après la première version finale en novembre 2003, OneDomain avait 20% de parts de marché et en a près de 30% maintenant.

Par conception, notre architecture d'application est réellement client/serveur mais nous utilisons Citrix [Metaframe Terminal Server] pour créer des clients légers et utiliser la mémoire du serveur et non pas celle du client.

Nous avons un serveur avec Firebird 1.5.2 et plusieurs (jusqu'à six) serveurs Citrix qui le sollicitent. Nous utilisons un client Win32, écrit en Delphi, qui fonctionne sur les serveurs Citrix et alimente les utilisateurs qui accèdent à la base sur la machine serveur Citrix, une machine avec quatre processeurs, utilisant l'hyperthreading pour agir comme huit processeurs, avec 3.5Gb de RAM disponible.

Nous avons commencé avec Firebird SuperServeur mais nous avons rapidement fait le choix d'utiliser la version Classic pour plusieures raisons, dont les limites d'utilisation de la mémoire qui dégradait considérablement les performances quand la charge devenait importante.

Avec près de 800-900 utilisateurs théoriques, dont moins de 300 connectés simultanément, nous avons toujours tenu la charge. Le bon point avec cette architecture, est que vous pouvez facilement monter en charge, en ajoutant seulement plus de serveurs.

Moscow Interbank Currency Exchange Bank

Lieux: Russie

Contact: Sergey Korotkikh

En dépit de son nom, MICEX est la plus grande bourse d'échange de devises et dérivés de Russie. Le chiffre d'affaires moyen quotidien dépasse les 6 milliards de $. En tant que bourse d'échange entièrement électronique, MICEX a utilisé InterBase et, plus tard, Firebird, depuis 1994 en tant que principal serveur de base de données pour les données de marché, ordres et échanges.

Le MICEX Trading System a plus de 2000 utilisateurs implantés sur tout le territoire de la Russie dans huit fuseaux horaires différents. Les échanges sont en temps réel avec une charge moyenne de plus d'un quart de millions d'ordres par jour, avec plus de 180.000 échanges conclus par jour.

De plus, plus de 300 systèmes de courtage électronique sont connectés au système d'échanges à travers une API de connexion. Les systèmes d'échanges lui même est “semi-détaché” de la base de données elle même, fournissant un middleware lourd d'accès à la base, de laquelle et vers laquelle il récupère et maintiens toutes les informations nécessaires pour les échanges.

En plus des fonctions d'échanges, la base de données Firebird est intensément utilisée par nos activité de clearing et de reporting. Nous générons des états quotidiens d'échanges et de clearing pour nos 1000 membres et les envoyons par e-mail.

Élément clé des échanges financiers en Russie, MICEX est tenu d'offrir un haut niveau de disponibilité. D'après un audit fait par le Gartner Group, nous avons un niveau de disponibilité de 99.999 pour cent.

Star-airlines

Lieux: France

Contact: Fabien Campos

STAR AIRLINES est une compagnie aérienne française. Dédiée au trafic touristique, STAR AIRLINES exploite des vols réguliers principalement sur des destinations long courrier, en Afrique et au Proche Orient, et des vols affrétés sur les destinations moyen courrier du bassin méditerranéen.

Firebird est le serveur de base de données de notre ERP et gère notre budget, le planning des avions et des équipages, l'exploitation et la régulation, la facturation et le contrôle.

Firebird est installé sur un serveur Windows 2000 utilisé par 125 utilisateurs (administratifs) déclarés et cinq d'entre eux en utilisation 24h/24 et 7j/7.

Environ 300 personnes (nomades) consultent régulièrement leur planning et messages à l'aide d'une application web.

Deux services tournant 24h/24 et 7j/7 importent automatiquement et régulièrement des données en provenance de l'extérieur.

Des extractions quotidiennes et mensuelles sont faites vers notre système de paie et notre comptabilité.

Facteurs influant l'extensibilité

Différents facteurs doivent être pris en compte quand on prévoit d'étendre les capacités pour permettre plus de connexions concurrentes des utilisateurs.

Nombre d'utilisateurs

La première chose à identifier est de savoir si l'on parle du nombre de connexions (le nombre maximum d'utilisateurs connectés au même moment au plus haut de la charge) ou de la taille maximum de la base des utilisateurs ?

Limites matérielles, logicielles et réseaux

Firebird peut être déployé avec différents “modèles”. Ce qui nous intéresse ici sont les deux modèles “serveur complet” , Superserveur et Classic. Ces deux modèles sont disponibles sous Windows, Linux et d'autres plate-formes. Bien que le moteur du serveur soit le même, quel que soit le modèle, le choix du modèle a une influence croissante au fur et à mesure que le nombre d'utilisateurs potentiels connectés augmente. Le matériel et le système d'exploitation jouent un rôle aussi.

Superserveur

Chaque Firebird Superserveur est limité en pratique à un maximum de 150 à 400 connexions concur-

rentes. Cela vient du fait que Superserveur et toutes ses connexions à toutes ses bases de données sont encapsulées dans un seul processus 32-bit. Un processus 32-bit ne peut adresser plus de 2 Gb de mémoire. Chaque connexion à une base crée un ou plusieurs threads pour gérer les requêtes de cette connexion; Superserveur utilise aussi des threads pour ces propres tâches de fond de collecte des données périmées et d'autre tâches en ligne.

Superserveur a aussi besoin de mémoire pour d'autres tâches, bien sûr.

Un cache de page configurable — un pour chaque base ayant une connexion active — est maintenu pour gérer les pages de base et d'index fréquemment utilisées en mémoire. Pour Superserveur, ce cache est partagé par toutes les connexions à la base en question. La taille du cache de page par défaut est de 2048 pages, donc, pour une base ayant une taille de page par défaut de 4Kb, 8 Mb de RAM doivent être disponibles pour un usage efficace de ce cache. (Si le a besoin d'être paginé dans la mémoire virtuelle, son utilité serait nulle!).

En général, les bases sont créées avec une taille de page de 8Kb et c'est un fréquent gâchis de ressources quand les développeurs et les DBA surdimensionnent la taille du cache, en pensant que “plus c'est mieux” (c'est ce que l'auteur appelle “le syndrome Lamborghini”: si vous roulez sur le périphérique parisien aux heures de pointe dans une Lamborghini, pensez-vous que vous irez plus vite que moi avec ma Mazda 121 ?).

Quelques fois cette mesure est prise pour améliorer le temps de réponse qui a été dégradé par une mauvaise gestion des transactions dans le code client et par une mauvaise gestion des connexions. C'est inutile. De tels problèmes n'ont rien à voir avec la taille du cache mais bien avec une mauvaise libération des ressources. Augmenter le cache ne fait qu'aggraver la situation.

Pensez à l'effet sur les ressources quand, par exemple, le cache pour une de page classique de 8Kb est porté à 20.000 pages. Cela représente 160 Mb qui doit être préservé en mémoire pour rendre le cache utile. Le serveur maintien des fichiers de verrous dans la RAM et cela grossi dynamiquement (jusqu'à un maximum configurable) quand plus de connexions sont créées. Firebird 1.5 va utiliser la RAM pour les opérations de tri si elle est disponible. C'est parfait pour les temps de réponse, mais beaucoup d'applications mal écrites maintiennent ouvert des ensembles de données triées comportant des centaines de lignes pendant de longues périodes.

Les serveurs de bases de données aiment la RAM. Plus il y en a de disponible pour le serveur, plus il est rapide. Toutefois, la barrière de 2 Go pénalise le Superserveur dans une environnement surpeuplé parce que chaque connexion Superserveur utilise environ 2 Mo de RAM pour instancier son processus et maintenir ces ressources. Donc avec 500 connexions Superserveur vous utilisez un gigaoctet, laissant en gros moins d'un gigaoctet de RAM totale adressable disponible pour que le serveur travaille, trie, gère les transactions, les verrous des tables, le stockage en mémoire et les nouvelles connexions.

Classic

Le “serveur” Classic n'est en fait pas un serveur de bases de données, mais un service en tâche de fond (ou, sous Linux, un daemon xinetd) qui écoute les demandes de connexion. Pour chaque connexion réussie, le service instancie un processus serveur Firebird qui est la propriété exclusive de cette connexion. Le résultat est que chaque connexion utilise plus de ressources, mais le nombre de connexions devient uniquement une question de RAM disponible sur la machine serveur. En plus des ~2Mb pour instancier la connexion, chacun maintient son propre cache de page. Toutes les connexions obtiennent le même cache de départ et, puisque que le cache n'est pas partagé, il peut (et doit) être plus petit. La taille recommandée est d'environ 2Mb (512 pages pour une taille de page de 4Kb) sur une machine dédiée de 4Gb, mais peut être beaucoup plus petite. Il est important de garder assez de RAM disponible pour que le gestionnaire de verrous puisse “grossir” lors des pics de connexions.

Adressage 64-bit

Une manière de lever les limitations de la RAM 32-bit est de compiler un Superserveur 64-bit. Un Superserveur 64-bit est possible sur les plate-formes POSIX où l'OS a un support stable pour le matériel 64-bit. Un Superserveur 64-bit Firebird 1.5.2 expérimental sur AMD64/Linux i64 a été livré il y a plus d'un an mais il a été déclaré instable. Une version 64-bit AMD64/Linux i64 existe dans les phases de test de Firebird 2.0. Sous Windows, il n'est pas encore possible de compiler un Superserveur 64-bit du fait de problèmes avec l'actuelle version de du compilateur de Microsoft Visual Studio 7 C++, l'environnement standard de compilation de Firebird 2 sous Windows. Des binaires privés des deux versions v.1.5.x et v.2.0 sur des compilateur non Microsoft fonctionnent de manière stable dans des environnements de production et l'équipe de développement a signalé son intention de réaliser une version 64-bit Windows pour Firebird 2.0.

Pool de connexions

Utiliser un middleware pour garder et gérer des ressources pré allouées pour garder un nombre fini de connexions prêtes pour des demandes de connexions est ce qui est appelé un poll de connexions. Il y a plusieures manières d'implémenter un poll de connexions et qui oblige à agir de la sorte quand on a une base d'utilisateurs qui augmente. Il ne serait pas réaliste de mettre en place un système extensible multi utilisateurs sans cela. Un pool de connexions peut être combiné avec une gestion des files d'attente si les ressources ne sont pas suffisantes pour gérer les pics de charge.

Les développeurs de ces middleware doivent prendre soin de surveiller les connexions et déconnexions, surtout pour les clients “stateless” comme les navigateurs web, pour éviter les manques de ressource. Le middleware doit éviter les flux récursifs qui pourraient monopoliser le pool et, pour des raisons d'architecture, interdire complètement les connexions croisées.

La compétition pour la RAM

Essayer de faire fonctionner un serveur de base de données sur un système qui utilise aussi des services entrant en compétition avec lui, comme un serveur web, un serveur Exchange, un serveur de domaine, etc., risque de voir ces autres services voler les ressources (RAM et temps CPU) au serveur de base de données, empêchant le même le Superserveur d'utiliser les 2 Gb qu'il est capable de gérer ou limiter le nombre de connexions au serveur Classic, et ralentir les temps de réponses des bases de données.

Support SMP

L'incompatibilité de l'implémentation des threads de Superserveur avec l'implémentation Windows SMP doit être vue comme une “limite à l'extensibilité”. C'est à cause de la manière totalement arbitraire que Windows utilise pour passer l'intégralité de l'affinité d'un processus entre les processeurs qui résulte en un “effet de bascule”, par lequel les performances vont se dégrader continuellement aux heures de pointe tandis que le système attend que l'OS se décide à transférer toutes les resources du Superserveur actives en mémoire d'un processeur à l'autre quand il détecte que la charge d'utilisation des processeurs est inégale. Superserveur est configuré par défaut pour n'être lié qu'à un seul processeur pour éviter cela.

Avec Linux, la gestion multiprocesseurs ne crée pas cet effet de bascule avec les noyaux 2.6 et supérieurs, il a toutefois été rapporté avec certains noyaux 2.4 sur de l'ancien matériel SMP. Cependant, SMP n'apporte pas non plus de gains significatifs de performance pour Superserveur sous Linux.

Le support configurable pour différent niveaux de SMP, la gestion fine du multi-threading a été

conçue pour le moteur Vulcan et deviendra une fonctionnalité de Firebird 3. Son efficacité sur les opérations en mémoire a été suffisamment démontrée pour que cela soit quelque chose de très intéressant pour le futur.

Dans le même temps, au vu de l'importance inhérente aux temps de réponses des E/S des disques sur les processus en cours de transaction, tout le bruit fait autour des possibilités de SMP peut être rapproché du syndrome Lamborghini. Dans un environnement approprié, Superserveur sur un système mono processeur est très efficace ! Classic — un seul processus par connexion — ne souffre pas d'inhibition avec SMP, même sous Windows. Avant Firebird 3, quand la distinction entre les deux modèles de ressources aura disparue, pour être remplacée par une gestion configurable des ressources, Classic est le meilleur choix pour un plan de monté en charge puisqu'il est capable de répondre à demande croissante de montée en charge par une simple augmentation des ressources du système.

> **Note**
>
> Il a été rapporté quelques problème au niveau du noyau avec SMP sur certaines versions de Linux avec 8 CPUs et l'hyperthreading activé, qui n'apparaissent pas avec des systèmes 4X. Comme le support complet de l'affinité CPU sous Linux n'existe pas avant le noyau v.2.6, définir l'affinité CPU avec firebird.conf n'est pas supporté par Linux dans la version actuelle de Superserveur.

"Réseaux Windows"

Un réseau TCP/IP avec les ressources adéquates en utilisant le modèle approprié de Firebird est parfaitement extensible. Un hôte Linux sans application entrant en compétition (y compris Xserver!) fonctionnant dessus va délivrer le maximum de performance à l'autre bout de l'échelle. Comme serveur de base de données, un hôte Windows souffre de charges inutiles inhérente au système d'exploitation qui ont un effet notable négatif sur les performances dans des conditions de forte charge.

Toutefois, il y a un autre “piège” lié au réseaux Windows. Le protocole natif de canaux de communication nommés de Windows (“NetBEUI”), que l'interface réseau de Firebird supporte pour des raisons historiques, a une limite absolue d'un maximum de 930 connexions pour tout serveur. Les canaux de communication nommés doivent être proscrits pour les environnements autre que des petits réseaux workgroup, car c'est un protocole très “bavard” et l'est de plus en plus en cas de croissance du réseau.

Soit dit en passant, les bases de données sous Windows devraient toujours être stockées sur des partitions NTFS qui, si elles sont suffisamment protégées, offrent de meilleures performances et sont moins exposées aux risques que FAT32. Les anciens disques durs qui ont été utilisés sur des systèmes fonctionnant auparavant avec NT 4.0 doivent être aussi considérés comme exposés, l'ancien NTFS n'apportant pas les garanties de protections qui ont été ensuite ajoutées dans les version actuelles.

Conception des bases et des logiciels

La disponibilité et l'extensibilité de tout système logiciel peut être négativement affecté par une mauvaise conception de la base de données, des requêtes mal écrites, des flux de données inappropriés et, particulièrement, une mauvaise gestion des transactions.

L'architecture multi générationnelle assure une gestion des tâches optimiste robuste et excellente et une forte capacité de traitement. Une base de données performante est le résultat d'une conception attentive, une bonne normalisation, de bons index, et, du coté client, de requêtes bien écrites et une attention particulière portée à la gestion des transactions. Au contraire, une mauvaise conception de la base amène des requêtes inutilement complexes, des mauvais index qui peuvent bloquer les capacité de l'optimiseur à utiliser les bons plans de requête. Des requêtes lentes sont généralement le résultat

de ces défauts.

Collecte des données périmées

MGA accumule des “données périmées”, sous la forme d'anciennes versions des enregistrements. Le moteur nettoie en ligne ces données périmées. Toutefois, il ne permet pas le nettoyage des données concernant des anciennes versions d'enregistrements qui “intéressent” encore des transactions actives. Les transactions qui restent intéressées longtemps maintiennent des versions anciennes des enregistrements qui peuvent être pertinentes pour des transactions commencées après, ce qui fait que l'on peut arriver à un niveau de données périmées difficilement gérable. Il est essentiel que les développeurs qui utilisent Firebird comprennent comment cela peut arriver et d'écrire des logiciels qui évitent cet état, en gardant un niveau raisonnable de données périmées afin que le sous système de nettoyage puisse le gérer.

Commit Retaining et “Autocommit”

Commit Retaining est une “fonctionnalité” de Firebird qui a été héritée de son ancêtre, InterBase. Cela a été implémenté pour conserver des ressources coté serveur et les laisser disponibles pour les utilisateurs utilisant les outils de développement rapide de Borland et le BDE, le cadre d'accès générique d'accès aux données qui a été créé pour permettre aux applications graphiques Windows de se connecter aux bases de données. Son but était de présenter un niveau uniforme pour le développement RAD, quelque soit la base de données fonctionnant à l'autre bout de la chaîne.

Autocommit—- un mécanisme coté client qui joue le rôle d'un Post au niveau instruction et d'un Commit au niveau transaction en une seule étape -— a été fournit pour permettre aux développeurs de ne pas ce soucier des transactions. Dans les composants RAD, Commit Retaining et Autocommit ont été liés. Cela convient très bien à des base de données bureautique comme dBase et Paradox (lesquels moteurs sont, de fait, le BDE!), qui n'ont pas de transactions.

Avec Firebird (et InterBase), Commit Retaining fait que les transactions restent intéressées indéfiniment. La collecte des données périmées cesse de fait avec les applications “standard” créées avec les outils Borland RAD et tout autre application qui utilise Commit Retaining. De tels systèmes connaissent des problèmes de dégradations progressives de performances qui ne peuvent être résolus sans un arrêt des bases afin de permettre à ces vieilles transaction de mourir.

Autocommit et Commit Retaining ne sont pas le seul apanage des outils Borland, bien sûr. Il sont utilisés par beaucoup d'interfaces d'accès aux données et Commit Retaining est disponible dans la norme SQL, donc il revient aux développeurs de bien comprendre le mécanisme et ses effets afin d'utiliser ces fonctionnalités avec une extrême prudence et un bon degré de contrôle.

A qui appartient et qui gère Firebird?

Aucune organisation commerciale ne “possède” Firebird. C'est la propriété commune des membre du projet.

Il est légalement et financièrement soutenu par la Fondation Firebird, une association à but non lucratif enregistrée et administrée en Nouvelles Galles du Sud, en Australie.

Gestion

L'équipe de développement est auto-gérée. Les décisions sur ce qui doit être livré dans les nouvelles versions sont prises par consensus, dans un forum privé constitué des développeurs actifs, les responsables de la construction des binaires, (ceux qui compilent les versions livrées et les utilitaires d'installation), le coordinateur de la documentation et le responsable des livraisons de versions.

La Fondation Firebird, en tant que telle, n'est pas impliquée dans la direction ou la gestion de l'effort de développement, même s'il existe des interconnexions (des membre du projets qui sont aussi des membres de la Fondation).

Maintenance du code

Firebird a derrière lui un code de base solide, bien maintenu avec constamment de nombreux regards attentifs posés sur lui à tous niveaux, de l'architecture, à la conception, à l'implémentation et aux tests de qualité.

Politique de versions

L'équipe de développement tient bon le principe de ne pas faire de version de production tant qu'au moins tous problèmes relevés dans les tests de qualité (QA) n'aient été résolus de manière jugée satisfaisante par les administrateurs du projet. Les version majeures et sous versions passent par de multiples cycles de tests de beta et pré-versions. L'inconvénient d'une telle rigueur dans la production est que les dates finales de livraisons ne sont jamais fermes.

Les versions majeures représentent des améliorations majeures et des nouvelles fonctionnalités. Les sous-versions arrivent périodiquement entre les versions majeures pour corriger des bug ou pour apporter des améliorations mineures à des fonctionnalités existantes et aux outils, ou pour compléter des fonctionnalités qui auraient été partiellement implémentées dans une précédente version ou sous-version.

Pour les applications clientes, la seule modification essentielle est de mettre à jour la bibliothèque cliente en concordance avec le numéro de la version majeure. Les fournisseurs d'applications peuvent incorporer des nouvelles fonctionnalités dans leurs logiciels et synchroniser l'installation d'une nouvelle version de Firebird en installant de nouveaux modules ou versions de leurs application clientes ou serveur de leurs applications.

Licence Open Source

Le code source de Firebird est sous deux licences dérivées de la Mozilla Public License v.1.1. (MPL). Les modules en provenance du code de base d'InterBase est sous la licence InterBase Public License v.1.0, tandis que l' Initial Developer's Public License (IDPL) s'applique aux nouveaux modules.

L'IPL et l'IDPL diffèrent de la MPL 1.1 seulement par le retrait de ce qui impliquait des droits de propriété à Netscape Corporation. L'IDPL diffère de l'IPL par l'exclusion d' “Inprise Corporation” dans différents copyright du code source.

Le style MPL de licence diffère de la bien connue GNU Public Licenses (GPL) du fait qu'elle n'est pas “virale”. “Virale” est le terme qui décrit la restriction GPL qui interdit que du code sous cette licence soit compilé avec un autre code source qui ne soit pas lui même soumis à la licence GPL. Une

licence non virale permet que des modules régis sous cette licence soient compilés avec du code, fermé, propriétaire, ce qui la rend plus utilisable pour le développement de logiciels commerciaux.

Toutefois, comme la GPL et la plupart des licences open source, le style de licence MPL rend obligatoire que les modifications faites au code source lui même soient rendues librement disponibles au public et que ce code ne soit pas distribué sous une autre licence.

Financement

Les concepteurs et programmeurs qui développent Firebird sont des volontaires, principalement des travailleurs indépendants ou avec une autre activité qui travaillent à temps partiel sur le code source de Firebird. Certains sont rétribués directement ou indirectement par leur employeur, en donnant une partie de temps de travail de l'entreprise pour travailler sur Firebird.

Certains des plus actifs du “coeur” des développeurs (ceux qui travaillent sur le moteur de Firebird lui même) sont aidés par des subventions provenant des fonds levés par la Fondation Firebird, ce qui leur permet de consacrer un minimum d'heures par mois au projet. Le projet a un concepteur/developpeur pour qui la subvention est suffisante pour qu'il y travaille à plein temps. Un subventionnement est aussi possible pour ceux des membres qui développent et supportent des pilotes d'accès.

Les fonds de la Fondation proviennent des cotisations des membres, du sponsoring d'entreprises et de dons privés. Les membres, les sponsors et les dons à la Fondation proviennent de la communauté des utilisateurs et des entreprises qui utilisent Firebird pour leurs propres produits commerciaux et donnent quelque chose en retour pour le développement.

Licence