UNIVERSITÉ DU QUÉBEC À MONTRÉAL

DÉVELOPPEMENT D'UNE BASE DE DONNÉES BIOINFORMATIQUE SPECIALISÉE GBANK UQAM

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
RABAH DJEMA

MAI 2008

REMERCIEMENTS

Tout d'abord je veux remercier mon directeur de recherche, monsieur Vladimir Makarenkov, pour son soutien, sa patience et sa générosité tout au long de ma maîtrise.

Ensuite, je tiens à remercier mes amis et collègues : Alix Boc, Abdoulaye Baniré Diallo, Alpha Boubacar Diallo et surtout Walid Ktata pour leurs aide précieuse et leurs points de vue sans lesquels ce travail de mémoire ne serait pas aussi riche et aussi consistant.

Enfin, tous mes remerciements à Dieu, ensuite ma femme, mes enfants, mes frères, mes sœurs et tous mes amis qui m'ont encouragé durant toute ma scolarité.

Avec tout mon cœur,
Rabah Djema

TABLES DES MATIÈRES

LISTE DES ACRONYMES

ORACLE 10 g	La base de données d'Oracle version 10 g faite pour supporter la grille informatique.
NCBI	National Centre for Biotechnology Information.
EMBL	La base du laboratoire de biologie moléculaire européen.
DDBJ	DNA Data Bank of Japan
DNA	ADN
PHYLOGÉNIE	Étude de la formation et de l'évolution des organismes vivants
GENBANK	Genetic Sequence Bank, banque de données de NCBI.
GBANK UQAM	Base de données développée à l'UQAM.
T-REX	Tree and reticulogram reconstruction.
ADN	Acide DesoxyriboNucleique.
ARN	Acide RiboNucleique.
HGT	Horizontal Gene Transfer.
SGA	Zone Globale de Système (utilisée dans les bases de données).
GI	GenInfo Identifier
MEDLINE	Medical Literature Analysis and Retrieval System Online, est une base de données bibliographiques qui couvre tous les domaines médicaux
PUBMED	Moteur de recherche dans le MEDLINE
DDL	Data Definition Language: Langage de definition et de desciption de données

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

La base de données GBank de l'UQAM a été développée afin de pallier certains problèmes majeurs posés par l'utilisation de la base de données GenBank du NCBI. En effet, les problèmes suivants ont déclenché le développement de GBank UQAM : 1- Certaines requêtes complexes utilisées par les bioinformaticiens sont lentes en raison notamment de la taille énorme et toujours croissante de la base de données. 2- Les bioinformaticiens de l'UQAM dépendent entièrement de la base de NCBI. En cas de sa panne, ils n'ont pas de possibilité d'y accéder. 3- Les utilisateurs n'ont aucun contrôle sur la base de données GenBank. En plus, ils dépendent entièrement des mises à jour du NCBI. 4- Les outils de GenBank pour le filtrage des données ne sont pas toujours adaptés aux besoins des bioinformaticiens intéressés par l'analyse phylogénétique. Ceci mène les bioinformaticiens de se soumettre au mode de fonctionnement de la base GenBank.

GBank UQAM se voit donc un sous-ensemble de la base GenBank international, qui résout en totalité ou partiellement les problèmes posés ci-dessus. Ceci a été rendu possible notamment grâce à l'utilisation de la base de données Oracle 10g qui offre plusieurs caractéristiques intéressantes. La nouvelle base de l'UQAM permettrait donc : 1- d'**Améliorer le temps de réponse** : Étant traité localement, nous pouvons offrir un temps d'accès nettement meilleur. 2- de **Mieux contrôler les données** : Nous pouvons organiser les données selon nos besoins et donc rendre la base de données plus optimale. En effet, maintenant nous sommes capables de filtrer les données selon nos besoins spécifiques ce qui augmente nettement notre productivité. 3- d'**Optimiser la base de données** : Avec des temps de réponses améliorés et une plus grande maniabilité dans la gestion de la base de données de l'UQAM, il nous est possible d'optimiser continuellement notre base de données pour la rendre plus évolutive et plus adaptée à nos besoins futurs.

Afin de mieux exploiter la nouvelle base de données, nous avons élaboré une interface utilisateur facile et conviviale qui répond à tous les besoins des utilisateurs (bioinformaticiens) d'une base de données bioinformatique.

MOTS CLÉS: GBank UQAM, Bioinformatique, Oracle10g, Performances, T-REX.

CHAPITRE I

LA BASE DE DONNÉES ORACLE 10G

1.1 Introduction

Les bases de données Oracle 10g (www.Oracle.com) sont idéales pour les projets qui comprennent des traitements transactionnels de volumes importants de données et les projets de type entrepôt de données (DATA WAREHOUSING) à requêtes intenses. Elles offrent une bonne extensibilité sur toutes les configurations matérielles et peuvent être utilisées afin de gérer une quantité d'information considérable avec la plus grande fiabilité connue aujourd'hui dans l'industrie.

Une base de données Oracle 10g offre une disponibilité unique qui pallie aux problèmes de pannes liées aux interventions humaines. Ces pannes sont le plus souvent très coûteuses. Elle augmente les capacités d'autogestion qui permettent de réduire les coûts opérationnels.

La base de données Oracle 10g, peut être utilisée afin de réduire de manière significative les coûts d'infrastructure à travers l'utilisation efficace des *'pools partagés'*. À chaque version, Oracle s'engage de plus en plus dans la grille informatique, dans l'automatisation de base de données et dans l'autogestion des ressources.

Dans ce chapitre, nous présenterons de façon générale, les notions fondamentales d'Oracle 10g et certains concepts utilisés dans le cadre de notre travail.

Dans ce chapitre nous allons aussi présenter l'architecture d'Oracle 10g. Ensuite, nous présenterons la structure physique et la structure logique de la base de données Oracle. Finalement, nous présenterons les caractéristiques de la base de données Oracle sous forme de facteurs de qualité. La figure suivante présente l'évolution de la base de données Oracle entre 1979 et 2006 [10].

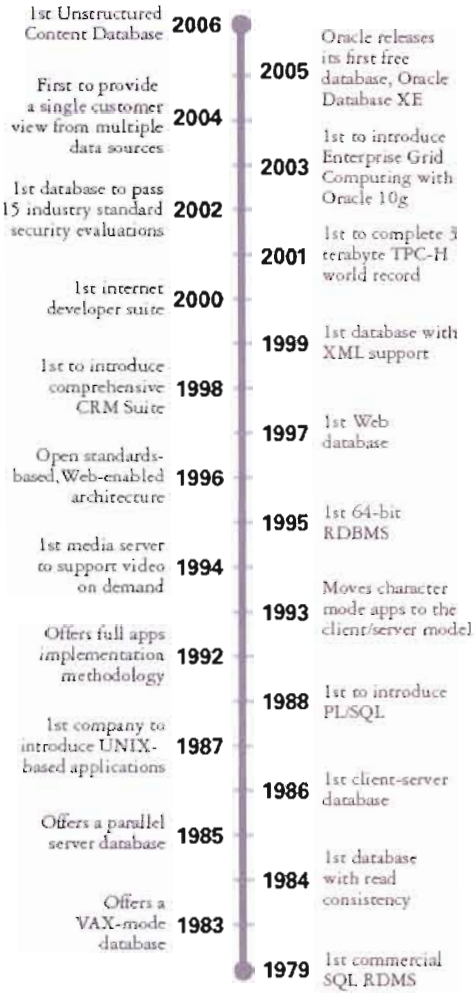

Figure 1.1 Évolution de base de données Oracle

1.2 Architecture de la base de données Oracle

La base de données Oracle est une collection de fichiers de données traités comme une seule unité, dans le but de sauvegarder et de consulter des informations. Un serveur de base de données (en terme de plateforme logiciel) est un moyen efficace pour résoudre des problèmes de gestion d'information. Généralement, un serveur gère une grande quantité de données dans un environnement multi-utilisateurs. Ainsi, plusieurs utilisateurs peuvent concurremment accéder aux mêmes données. Ceci est accompli tout en garantissant une performance très élevée. Un serveur de base de données empêche l'accès non autorisé et fournit des solutions efficaces pour le rétablissement après échec (Rollback).

La base de données possède des structures logiques et physiques. Du fait que les structures physiques et logiques sont séparées, le stockage physique des données peut être contrôlé sans affecter l'accès aux structures logiques de stockage.

1.2.1 Architecture de la grille d'Oracle

Les entreprises sont toujours à la recherche de méthodes pour réduire les coûts et augmenter l'efficacité de leurs systèmes. La grille d'Oracle offre un moyen fort intéressant notamment grâce à la réunification des ressources matérielles et logicielles. Ainsi, l'usager n'a pas besoin de savoir où se trouve une ressource pour pouvoir l'utiliser.

Oracle 10g est la première plateforme qui offre un système complet et intégré pour la grille informatique. Elle agit à la fois en tant que consommateur de ressources et fournisseur de données.

1.2.2 Architecture d'application

Il existe deux façons usuelles pour concevoir une base de données : le client/serveur et le multi niveaux. Pendant que l'Internet prend de l'ampleur dans les

environnements informatiques, plusieurs systèmes de gestion de base de données sont devenus des environnements à plusieurs niveaux. Le multi niveaux utilise plus d'un processeur pour traiter un ensemble de tâches. Le traitement distribué réduit la charge sur un processeur simple en permettant à différents processeurs de se concentrer sur un sous-ensemble de tâches reliées, améliorant ainsi la performance et les capacités du système tout entier. Un système de base de données Oracle peut facilement tirer profit du traitement distribué en utilisant son architecture client/serveur. Dans cette architecture, le système de base de données est divisé en deux parties : un client et un serveur tels qu'indiqué dans la figure 1.5 [20].

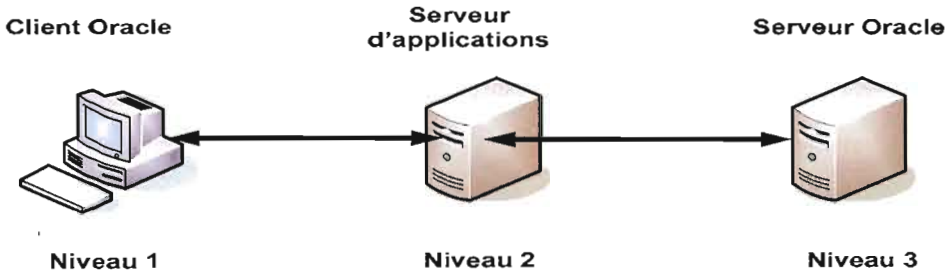

Figure 1.2 Architecture multi niveaux

Le client : le client est une application de base de données qui fait une demande d'exécution d'une opération sur le serveur de base de données. Il demande, traite, et présente des données gérées par le serveur. Le poste de travail du client peut être optimisé pour faire son travail. Par exemple, il pourrait ne pas avoir besoin de grandes capacités de disque. Souvent, le client s'exécute sur un ordinateur différent de celui du serveur de base de données, généralement sur un PC ou sur un terminal.

Le serveur : le serveur accède à la base de données Oracle et gère les fonctions nécessaires à l'accès simultané et partagé aux données. Le serveur reçoit et traite les requêtes de SQL et de PL/SQL qui proviennent des applications clientes. L'ordinateur du serveur peut être ordinateurs 'optimisé' pour ses fonctions. Par exemple, il peut y avoir une grande capacité de disque et plusieurs unités de traitements rapides.

1.2.3 Structure physique de la base de données Oracle 10g

Les sections suivantes présentent la structure physique de la base de données Oracle 10g, y compris les fichiers de données, les fichiers de journalisation de reprise et les fichiers de contrôles.

Fichiers de données

Chaque base de données possède un ou plusieurs fichiers de données physiques (voir le tableau 1.1 [16]). Ces fichiers contiennent toutes les données de la structure logique de la base de données, tels que des tables et des index qui sont physiquement stockés dans des fichiers assignés par la base de données.

Les caractéristiques principales des fichiers de données sont les suivantes :

- Un fichier de données est associé à une seule base de données.
- Un ou plusieurs fichiers de données forment une unité logique de stockage de base de données appelée un espace disque « tablespace ».

Pour visualiser tous les fichiers physiques de données, on utilise la requête SQL décrite dans le tableau 1.1. Ce même tableau présente un exemple de fichiers de données physiques.

Tableau 1.1: Fichiers de données d'Oracle 10g

```
SQL> select * from V$DBFILE;

    FILE# ---------NAME
       2     D:\ORACLENT\DATABASE\USR1ORCL.ORA
       3     D:\ORACLENT\DATABASE\RBS1ORCL.ORA
       4     D:\ORACLENT\DATABASE\TMP1ORCL.ORA
       1     D:\ORACLENT\DATABASE\SYS1ORCL.ORA
SQL>
```

Fichiers de contrôle

Chaque base de données Oracle possède un fichier de contrôle contenant les entrées qui définissent sa structure physique (tableau 1.2 [16]). Parmi ces informations figurent:

- Le nom de la base de données;
- Les noms et localisation des fichiers de données et des fichiers du journal de reprise;
- La date de création de la base de données.

Oracle peut multiplier le fichier de contrôle, c'est-à-dire, il peut maintenir simultanément un certain nombre de copies identiques de fichiers de contrôle afin de se protéger contre une défaillance impliquant le fichier de contrôle. Chaque fois qu'une instance d'une base de données Oracle est démarrée, un fichier de contrôle identifie la base de données et un fichier de journal de reprise est ouvert. Si la base de données est changée physiquement (par exemple, si un nouveau fichier de données ou un fichier de journal de reprise est créé), alors le fichier de contrôle est automatiquement modifié par Oracle pour refléter le changement.

Pour afficher les fichiers physiques de contrôles, on utilise une requête SQL (voir tableau 1.2). Ce même tableau présente un exemple de fichiers de contrôle.

Tableau 1.2: Fichiers de contrôle d'Oracle

```
SQL> select * from V$CONTROLFILE;

STATUS       NAME
------------------------------------------
             D:\ORACLENT\DATABASE\CTL1ORCL.ORA
           D:\ORACLENT\DATABASE\CTL2ORCL.ORA
SQL>
```

Fichiers du journal de reprise

Chaque base de données Oracle contient deux fichiers de journalisation de reprise ou plus. Un journal de reprise se compose de doubles des entrées. La fonction principale de cette réplication de l'action est d'enregistrer tous les changements apportés aux données. Si un échec survient, empêchant des données modifiées d'être, de manière permanente, écrite dans les fichiers de données, les changements peuvent alors être obtenus en tirant profit du dédoublement des actions ; ainsi les données sont protégées contre la perte.

Pour se protéger contre un échec impliquant la notation des doubles elle-même, Oracle permet de refaire la notation de sorte que deux copies ou plus de la notation puissent être maintenues sur différents disques. L'information dans un dossier de notation est employée seulement pour récupérer d'un système ou d'un échec de médias qui empêche des données d'une base de données d'être écrites dans les fichiers de données. Par exemple, si une panne de courant inattendue termine l'opération de base de données, les données dans la mémoire ne peuvent pas être écrites dans les fichiers de données et seront perdues. Cependant, des données perdues peuvent être récupérées quand la base de données est ré-ouverte, après que le courant soit rétabli. En appliquant les actions dans le fichier de réplication le plus récent, Oracle est capable de restaurer la base de données à l'état ou elle était avant la panne. Ce processus s'appelle 'Rollback'.

Fichiers d'archivage

Quand la base de données est en mode ARCHIVELOG, Oracle archive automatiquement les fichiers récupérés (voir le tableau suivant [16]).

Pour afficher les fichiers d'archivage d'Oracle, on utilise une requête SQL (voir tableau 1.3). Ce même tableau présente un exemple de fichiers d'archivage.

Tableau 1.3: Fichiers d'archivage d'Oracle

```
SQL> select * from V$LOGFILE;

 GROUP#  STATUS     NEMBER
---------------------------------------
    1    STALE      D:\ORACLENT\DATABASE\LOG2ORCL.ORA
    2               D:\ORACLENT\DATABASE\LOG1ORCL.ORA

SQL>
```

Fichiers d'initialisation

Le fichier d'initialisation contient la liste des paramètres de configuration des instances de la base de données Oracle 10g [15, 17]. Parmi ces informations figurent la taille de la mémoire allouée à l'instance, ainsi que la taille des différentes structures dans cette espace mémoire. Généralement, on distingue trois types de paramètres:

1. **Les paramètres d'emplacement**

   Ces paramètres décrivent l'emplacement des fichiers où l'instance doit lire et écrire ses données. Dans certains cas, on peut spécifier le chemin complet, comme dans le paramètre CONTROL_FILES.

   Quand l'instance est démarrée, Oracle 10g vérifie que les fichiers et les répertoires inscrits dans le fichier d'initialisation sont accessibles, sinon des messages d'erreurs apparaissent.

2. **Les paramètres de limites**

   Les paramètres de limites permettent de :

- Définir la quantité de ressources disponibles pour démarrer une instance (OPEN_CURSORS, OPEN_LINKS, TIMED_STATISTICS).

- Spécifier la taille des différentes structures mémoire (DB_CACHE_SIZE, CHARED_POOL_SIZE, SORT_AREA_SIZE).

3. **Les paramètres de fonctionnalités**

   Les paramètres de fonctionnalité sont des paramètres qui acceptent uniquement les valeurs : TRUE, FALSE, PARTIAL ou FULL. Il s'agit des paramètres MAX_DUMP_FILE_SIZE, ORACLE_TRACE_ENABLE, ROW_LOKING et des autres paramètres qui déterminent le degré d'applicabilité des fonctionnalités (par exemple : COMPATIBLE).

Fichiers de sauvegarde

La reconstitution d'un fichier est tout simplement son remplacement par un fichier de sauvegarde. Typiquement, on reconstitue un fichier quand un échec ou une erreur d'utilisation a endommagé ou a supprimé le fichier original. La sauvegarde et la restauration contrôlent le processus de restauration, comme l'établissement de l'ordre des sauvegardes ainsi que l'ordre des restaurations, et l'utilisation du bon fichier de sauvegarde quand une restauration est nécessaire [15, 17].

1.2.4 Structures logiques des bases de données

Tablespaces

Les tablespaces sont des structures logiques qui regroupent plusieurs fichiers de données physiques (Datafiles) d'une base de données. Les tables, les index et les autres objets d'une base de données sont placés dans ces *tablespaces,* tel qu'illustré dans la figure 1.7) [17].

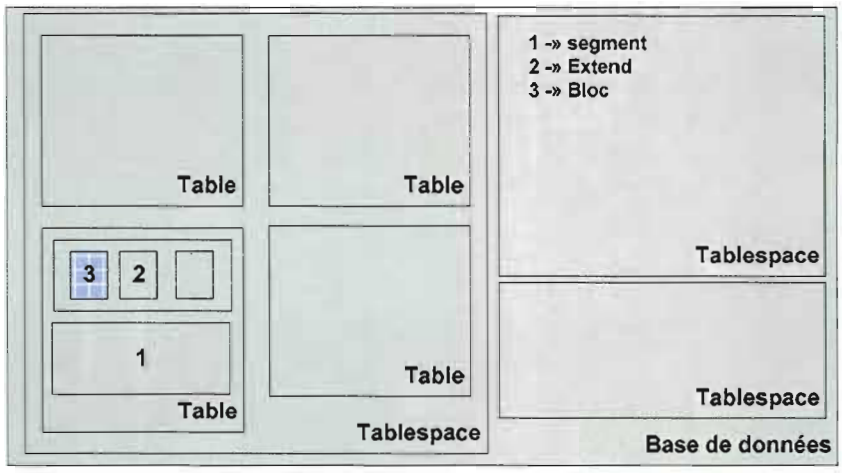

Figure 1.3 Structures logiques d'une base de données

Blocs de données d'Oracle

Un bloc de données correspond à un nombre spécifique d'octets de l'espace physique de la base de données sur le disque. La longueur du bloc standard est indiquée par le paramètre d'initialisation DB_BLOCK_SIZE. En outre, on peut indiquer jusqu'à cinq autres longueurs de bloc.

Extends

Un extend est un nombre spécifique de blocs de données contigus utilisés pour stocker un type spécifique d'information (voir le tableau 1.4) [17].

Tableau 1.4: Exemple d'extend

1 bloc	1 bloc	1 bloc	1 bloc
1 bloc	1 bloc	1 bloc	1 bloc
1 bloc	1 bloc	1 bloc	1 bloc
1 bloc	1 bloc	1 bloc	1 bloc
1 bloc	1 bloc	1 bloc	1 bloc
1 bloc	1 bloc	1 bloc	1 bloc
1 extend			

Segments

Un segment est un ensemble d'*extends* affectés à une certaine structure logique. On distingue plusieurs types de segments (les segments de données, les segments d'index, les segments temporaires et les segments de retour en arrière 'RollBack').

1.2.5 Schémas et objets de schémas [11]

Un schéma est une collection d'objets de base de données appartenant à un utilisateur. Il possède le même nom que cet utilisateur. Les objets du schéma sont des structures logiques qui se rapportent directement aux données de la base de données. Les objets du schéma incluent des structures comme des tables, des vues, et des index.

N.B : Les objets dans le même schéma peuvent être dans différents *tablespaces* et un *tablespace* peut contenir des objets de différents schémas.

Les objets d'un schéma sont :

- **Les tables** : Une table est l'unité de stockage élémentaire dans une base de données Oracle. Les tables de base de données contiennent toutes les données accessibles par un utilisateur. Chaque table possède des colonnes et des enregistrements.

- **Les index** : Les index sont des structures facultatives liées aux tables. Des index peuvent être créés pour augmenter la performance lors de l'accès aux données. Un index d'Oracle fournit un chemin d'accès aux données d'une table.

En traitant une requête, Oracle utilise les index disponibles pour localiser efficacement les enregistrements demandés. Les index sont utiles quand les applications demandent fréquemment une table pour un ensemble d'enregistrements ou d'un enregistrement spécifique.

Des index sont créés sur une ou plusieurs colonnes d'une table. Après qu'il soit créé, un index est automatiquement maintenu et utilisé par Oracle. Les changements aux données de la table (comme l'ajout de nouveaux enregistrements, la mise à jour des enregistrements, ou la suppression des enregistrements) sont automatiquement incorporés à tous les index appropriés de façon transparente pour l'utilisateur.

- **Les vues** : Les vues sont des présentations personnalisées des données contenues dans une ou plusieurs tables ou d'autres vues. Une vue peut également être considérée comme une requête stockée. Les vues ne contiennent pas réellement des données. Elles prennent leurs données dans les tables sur lesquelles elles sont basées, désignées sous le nom des tables de base des vues. Comme les tables, les vues peuvent être consultées, mises à jour, insérées, supprimées (avec quelques restrictions). Les opérations exécutées sur une vue affectent réellement les tables liées à la vue.

  Les vues fournissent un niveau additionnel de sécurité de table par l'accès limité à un ensemble prédéterminé d'enregistrements et aux colonnes d'une table. Elles cachent également la complexité de données et stockent des requêtes complexes.

- **Les clusters** : Les clusters sont des groupes de plusieurs tables physiquement stockées ensemble parce qu'elles partagent des colonnes communes et sont utilisées souvent ensemble. Étant donné que les enregistrements respectifs sont physiquement stockés ensemble, le temps d'accès au disque est grandement amélioré. Comme les index, les clusters n'affectent pas la conception de l'application.

- **Les synonymes** : Un synonyme est un alias d'une table, d'une vue, d'une séquence, d'une procédure, d'une fonction, d'un package, d'un type, d'une classe Java, ou d'un autre synonyme. Étant donnée qu'un synonyme est

simplement un nom d'alias, il n'exige aucun stockage autre que sa définition dans le dictionnaire de données.

1.2.6 Instance d'Oracle

Un serveur de base de données Oracle se compose d'une base de données Oracle et d'une instance d'Oracle. Chaque fois qu'une base de données est démarrée, une zone globale système (SGA) est assignée et Oracle lance un ou plusieurs processus. La combinaison des processus et des tampons de mémoire s'appelle **instance** d'Oracle.

La zone SGA

La zone globale de système (SGA) est une région de mémoire partagée qui contient des données et des paramètres pour une instance d'Oracle. Lorsque l'on initie le démarrage d'une instance sur le serveur de base de données, Oracle alloue une zone SGA, et libère cette zone quand l'instance s'arrête. Chaque instance a son propre SGA.

La figure 1.8 présente une structure typique des fichiers d'une base de données Oracle [11].

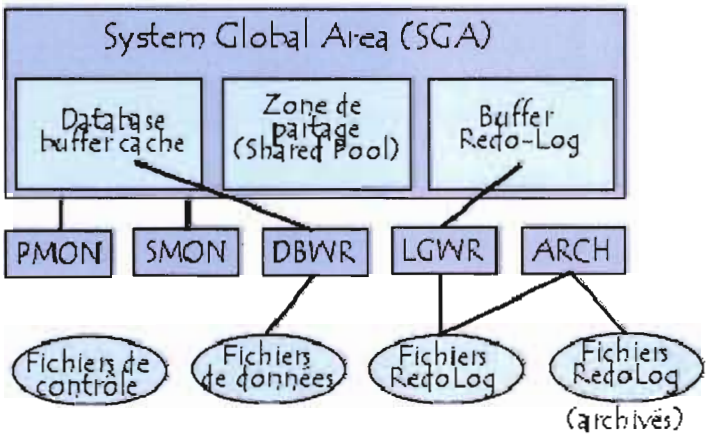

Figure 1.4 Structure des fichiers d'une base de données Oracle

Tous les utilisateurs qui sont connectés à la base de données Oracle 10g partagent la même zone SGA. Cette zone contient plusieurs structures mémoire.

L'information stockée dans la zone SGA est divisée en plusieurs types de structures de mémoire, y compris les tampons de données, le tampon du journal de reprise, et le *pool partagé*.

1. **Les tampons de données**

Les tampons de données sont regroupés dans le cache de données. Les tampons de données stockent les blocs de données utilisées en dernier (généralement celles qui sont le plus souvent demandées). Donc, il en résulte moins d'opérations d'entrée-sortie sur disque, ce qui améliore grandement la performance.

2. **Le tampon de journal de reprise**

Le journal de reprise contient toutes les modifications qui ont été apportées aux données de la base. Les entrées du journal sont placées dans les tampons du journal de reprise. Ensuite, elles sont utilisées si la restauration de la base de données s'avère nécessaire. La taille du tampon du journal de reprise est statique mais peut être augmentée via les paramètres du fichier d'initialisation.

3. **Le pool partagé**

Le pool partagé contient les constructions partagées de mémoire, telle que la zone des instructions de SQL partagée. Une zone de SQL partagée est nécessaire pour traiter chaque requête SQL soumise à une base de données. Une zone partagée de SQL contient entre autre le plan d'exécution d'analyse pour la requête correspondante.

4. Les partitions

- **Partition des tables**

La partition d'Oracle 10g permet de fractionner une table en plusieurs morceaux selon les critères de données. Ces morceaux sont aussi appelés des partitions.

La partition est transparente aux utilisateurs, elle ne nécessite pas de modifier les requêtes SQL. Cependant, après que des partitions soient définies, les scripts de DDL peuvent accéder, et manœuvrer, à des partitions précises plutôt qu'à des tables entières ou à des index.

Chaque partition d'une table ou d'un index doit avoir les mêmes attributs logiques, tels que des noms de colonnes, des types de données et des contraintes. Cependant, chaque partition peut avoir des attributs physiques séparés tels que les espaces disques logiques et fichiers de données.

La partition est utile pour plusieurs types d'applications, en particulier les applications qui contrôlent de grands volumes de données. Souvent, les systèmes d'OLTP tirent profit des améliorations d'administration et de disponibilité, alors que les systèmes d'entreposage de données tirent profit de l'exécution et de l'administration [18].

- **II- Partition des index**

Comme la partition des tables, la partition des index améliore l'administration, la disponibilité et la performance d'une base de donnée [18].

Il existe deux façons de partitionner des index dans la base de données Oracles 10g.

- Index local: pour chaque partition de la table créée, il y aura une (et une seule) partition d'index. Les données dans chaque partition d'index pointent sur l'ensemble des données d'une et unique partition de la table.

Logiquement, si la table a N partitions, l'index aura également N partitions.

- Index global: un index global est un index sur une table, partitionnée ou non partitionnée, qui est partitionné selon une clé de partitionnement différente de celle de la table. Les index partitionnés globaux peuvent seulement être partitionnés selon un partitionnement par Range. Par exemple, une table pourrait être partitionnée par Range selon les mois et avoir ainsi douze partitions, alors qu'un index sur cette table pourrait également être partitionné par Range, mais en utilisant une clé de partitionnement différente et avoir ainsi un nombre de partitions différent.

1.3 Fonctionnalités de la base de données Oracle 10g

1.3.1 Support de tous les environnements (Portabilité)

Oracle 10g est disponible sur tous les systèmes d'exploitation supportés par Oracle tels que Windows, Linux et Unix. Ainsi, il est supporté sur toutes les configurations matérielles allant des petites machines monoprocesseurs jusqu'aux environnements 'SMP high-end' (symetrical multiprocessing).

Les environnements de cluster et de grille sont également supportés (option dans la version Oracle cluster pour application à temps réel 'Oracle cluster for real time application').

1.3.2 Géstion de toutes les informations (Extensibilité)

Oracle 10g accepte tous les types de données relationnelles standard, ainsi que le stockage « native » de données XML, texte, document, Image, audio et vidéo.

Grâce à l'option d'Oracle spatial, l'Oracle 10g supporte les données spatiales complexes.

L'accès aux données sauvegardées se fait via des interfaces standard telles que SQL, JDBC, SQLJ, ODBC, OLE DB, ODP.NET, SQL/XML, XQUERY et WebDav.

Les programmes de la logique d'affaire déployés dans la base de données Oracle peuvent être écrits en Java et/ou en PL/SQL.

Oracle 10g est capable de stocker jusqu'à 8 Exabytes (10^{18} bytes) des données dans une seule base de données. Les données peuvent être importées dans une base de données ou exportées d'une base de données existante en parallèle sans avoir à se soucier de la plateforme du système d'exploitation.

L'option de partitionnement d'Oracle permet de simplifier les opérations d'administration communes pour les environnements de base de données qui contiennent des tables volumineuses (dont le nombre de lignes dépasse la centaine de millions). Oracle supporte quatre types de partitionnement « Hachage, Plage, Liste et Composé » comme le montre la figure suivante [13].

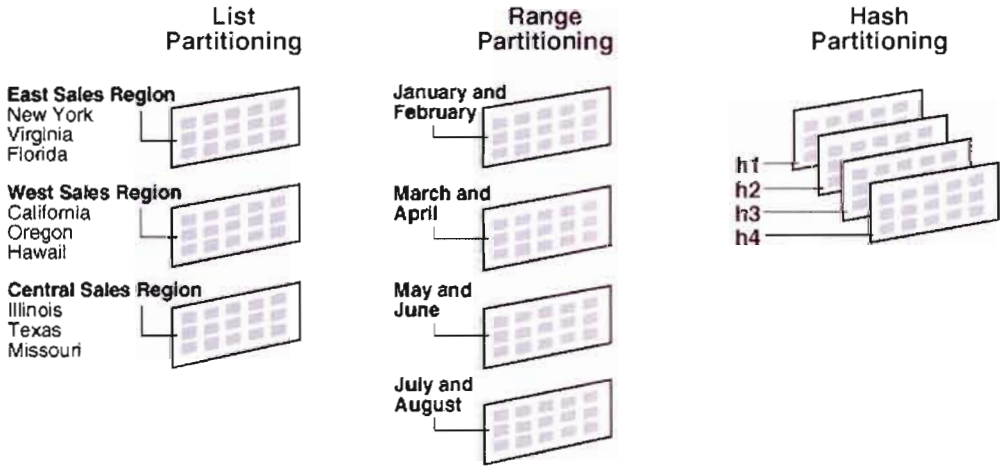

Figure 1.5 Partitionnements d'Oracle 10g

1.3.3 Intégration de toutes les informations

Oracle 10g supporte aussi les transactions et les requêtes distribuées entre deux ou plusieurs bases de données et fournit un support prédéfini pour la connexion à des bases de données tierces communes via ODBC. De plus, des passerelles transparentes aux bases de données tierces spécifiques sont disponibles offrant ainsi une solution d'intégration d'information hautement optimisée.

Oracle 10g fournit également un cadre de travail « framework » intégré pour la capture, le traitement et le stockage des événements dans la base de données. Les messages des applications peuvent être ainsi automatiquement propagés et appliqués par une ou plusieurs bases de données.

Oracle 10g peut être utilisé comme un coordinateur de stockage de données dans un environnement distribué. La réplication multi-master point à point est aussi supportée entre deux ou plusieurs bases de données Oracle.

1.3.4 Exécution de toutes les applications

Pour les environnements de traitement transactionnel en ligne, Oracle 10g supporte le déploiement d'un nombre important d'usagers en utilisant un verrouillage au niveau de l'enregistrement et une lecture multi-versions consistante, permettant ainsi de servir rapidement et facilement des dizaines de milliers d'usagers en ligne.

Pour l'intelligence d'affaire « Business Intelligence », Oracle 10g offre des capacités de modélisation statistique et analytique intégrées pouvant être accessibles directement d'un environnement basé sur SQL. Ces capacités intégrées peuvent être étendues davantage en utilisant les options 'Oracle OLAP' (Online Analytical Processing Database) et 'Oracle exploration de données' (Data Mining), qui offrent un moteur de calcul OLAP plus performant et la possibilité de prospecter les données plus rapidement.

1.3.5 Disponibilité en tout temps

Oracle 10g offre des capacités uniques afin d'assurer la disponibilité des données de l'application. 'Oracle Real Application cluster' (RAC) supporte le déploiement transparent d'une base de données sur un cluster de serveurs, offrant ainsi une tolérance aux erreurs matérielles ou pannes planifiées.

Oracle 10g offre une technologie de récupération après panne à redémarrage rapide (Fast start fault recovery), à quelques secondes, rendant la récupération plus rapide et prévisible et offrant la possibilité de rencontrer les objectifs en termes de niveau de disponibilité de service.

Oracle 10g offre des capacités de protection de données uniques. Les capacités de gestion des stockages automatisées et intégrées dans Oracle 10g répliquent les données à travers des unités de stockage disponibles pour la protection contre les pannes.

De nouveaux algorithmes de validation de données intégrées ont été implémentés en conjoncture avec les unités de stockage communes, éliminant ainsi une grande quantité de pannes causées par la corruption.

L'archivage automatique et la restauration à partir d'un disque de récupération sont offertes afin d'assurer la disponibilité en tout temps des archives, et éliminer ainsi les erreurs liées à l'opérateur et améliorer le temps de restauration.

L'intégration incrémentale et rapide d'archives dans une image d'archive existante est également offerte dans Oracle 10g, réduisant de manière significative le temps requis pour les archives en ligne et minimisant l'espace nécessaire pour le stockage quotidien.

Oracle 10g offre aussi une suite de capacités uniques de retour en arrière (flashback) qui aide les administrateurs à diagnostiquer facilement les erreurs humaines et les corriger par la suite:

- Les modifications apportées à une seule ligne (enregistrement),
- Les modifications causées par une transaction,
- Les modifications apportées à une ou plusieurs tables (y compris la suppression d'une table),
- Toutes les modifications apportées à la base de données en entier.

Oracle 10g introduit un démarrage rapide (Fast Start Fault Recovery) lors d'un transfert de traitement aux bases de données passives sans intervention humaine (manuelle).

Oracle 10g a été conçu également pour éviter que les routines de maintenance n'interfèrent avec les opérations à mission critique. De nouvelles unités mémoire et disques peuvent être rapportées aux systèmes sans avoir à le redémarrer. Dans la base de données, les tailles peuvent être re-localisées ou avoir leur type de stockage changé, de nouveaux index créés ou reconstruits et des colonnes ajoutées, supprimées ou renommées sans aucune interruption pour des usagers utilisant ces informations.

1.3.6 Assurance en sécurité prouvée

Aujourd'hui, la base de données Oracle offre le niveau de sécurité qui est parmi les plus élevés dans l'industrie. Dans la dernière décennie, Oracle a réussi 17 évaluations de sécurité effectuées par un organisme indépendant.

La consolidation, les exigences de confidentialité et les régulations gouvernementales nécessitent des capacités de sécurité accrue. Orale 10g offre des fonctionnalités de sécurité avant-gardistes telles que le chiffrage (*encryption*) de données, la gestion des clés, l'authentification via un Proxy, le contexte d'application et les rôles d'applications sécurisées. En plus, les fonctionnalités de sécurité communes telles

que l'audit, la vérification de complexité de mots de passe, les rôles de la base de données, les fonctions et procédures stockées, sont offertes.

Oracle Advanced Security (OAS) protège la confidentialité des données à travers le réseau en 'reniflant' l'adressage des données (data sniffing), la perte de données, les interceptions de réponses intermédiaires. Toutes les communications avec Oracle 10g peuvent être encryptées avec OAS, ce qui offre également des solutions d'authentification puissantes.

Oracle Label Security (OLS) offre une solution idéale pour les clients qui ont besoin de protéger des informations privées ou sensibles, basées sur une technologie de sécurité multi-niveaux. L'accès aux données est restreint en utilisant des étiquettes de sensibilité et de vérification de sécurité.

Pour les entreprises à gestion importante (à grand volume de gestion), la gestion des comptes usagers et des autorisations peuvent être centralisées en utilisant Oracle User Security (OUS) et Oracle Identity Management (OIM), éliminant le recours aux schémas d'utilisateurs de la base de données et en facilitant la gestion des autorisations d'utilisateur à travers l'organisation entière.

1.3.7 Installation rapide, gestion facile, développement facile

Oracle 10g offre une installation très rapide dans tous les environnements. La base de données est préconfigurée pour :

- Une utilisation en production,
- Une complexité avec un espace réservé,
- Une gestion de mémoire et de disques,
- Un archivage et une récupération automatique,
- Une gestion d'optimisateurs automatiques basée sur les statistiques.

La console de contrôle d'Oracle 10g offre une interface Web qui permet un affichage convivial du statut de la base de données et de l'environnement du cluster et permet des opérations d'administration de la base de données à partir de n'importe quel navigateur web.

CHAPITRE II

LA BASE DE DONNÉES GENBANK DE NCBI

2.1 Introduction

La bioinformatique a vu le jour dans les années 70, grâce notamment aux bases de données EMBL et GenBank. La bioinformatique fournit des méthodes et des logiciels permettant de gérer, d'organiser, d'analyser et d'exploiter des informations génétiques, génomiques et protéomiques en vue d'élaborer, de prédire et de produire de nouvelles connaissances.

En fournissant les données primaires à la bioinformatique, les bases de données participent directement à la construction de nouvelles connaissances.

Il existe deux types de bases de données :

1. Les bases de données généralistes : Elles offrent des informations hétérogènes suite à une collecte de données exhaustive.

2. Les bases de données spécialisées : Elles offrent des informations homogènes centrées sur un thème particulier.

En bioinformatique, on retrouve trois principales bases de données généralistes qui sont connues mondialement : la base de données GenBank du NCBI (http://www.ncbi.nlm.nih.gov/), la base de données d'ADN du Japon (DDBJ)

(http://www.ddbj.nig.ac.jp/) et la base du laboratoire de biologie moléculaire européen (EMBL) (http://www.ebi.ac.uk/embl/).

Le point commun entre ces bases de données est qu'elles font toutes partie de *'l'International Nucléotide Séquence Databases'* et qu'elles ont la même structure. Ces trois bases de données collaborent étroitement et régulièrement afin de s'enrichir mutuellement et d'offrir un ensemble de données cohérent et complet. Ceci fait en sorte que le contenu des trois bases est plus et moins identique.

Dans ce chapitre nous allons présenter l'évolution de la base de données GenBank. Ensuite, nous allons décrire les champs les plus importants de cette base de données. Finalement, nous présenterons les différents logiciels permettant d'y accéder et de soumettre de nouveaux enregistrements à cette base importante.

2.2 Évolution de GenBank (Genetic Sequence Bank)

Créée en 1982 par la société IntelliGenetics et diffusée par le NCBI (*National Centre for Biotechnology Information*, Los Alamos, USA), la base de données GenBank est une base publique de données de séquences d'acides nucléiques (ADN). Depuis 1982, GenBank connaît une croissance extraordinaire. En effet, son contenu double chaque 18 mois et une mise à jour est disponible chaque deux mois. Par exemple, elle a atteint le volume de 66 925 938 907 nucléotides en Octobre 2006 [1].

Les figures suivantes présentent l'évolution de la base de données GenBank entre Décembre 1982 et Octobre 2006.

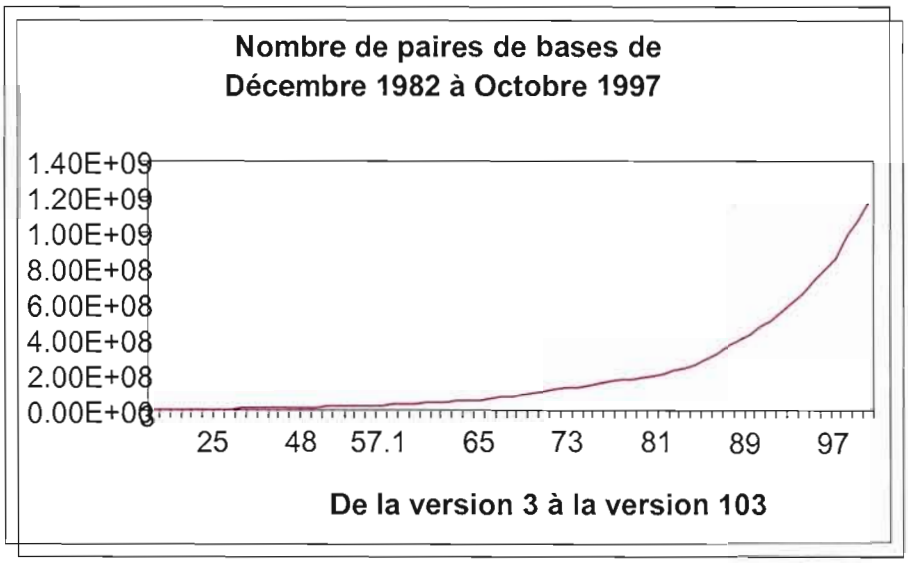

Figure 2.1 GenBank, données d'Octobre 2006 (de la version 3 à la Version 103) [8]

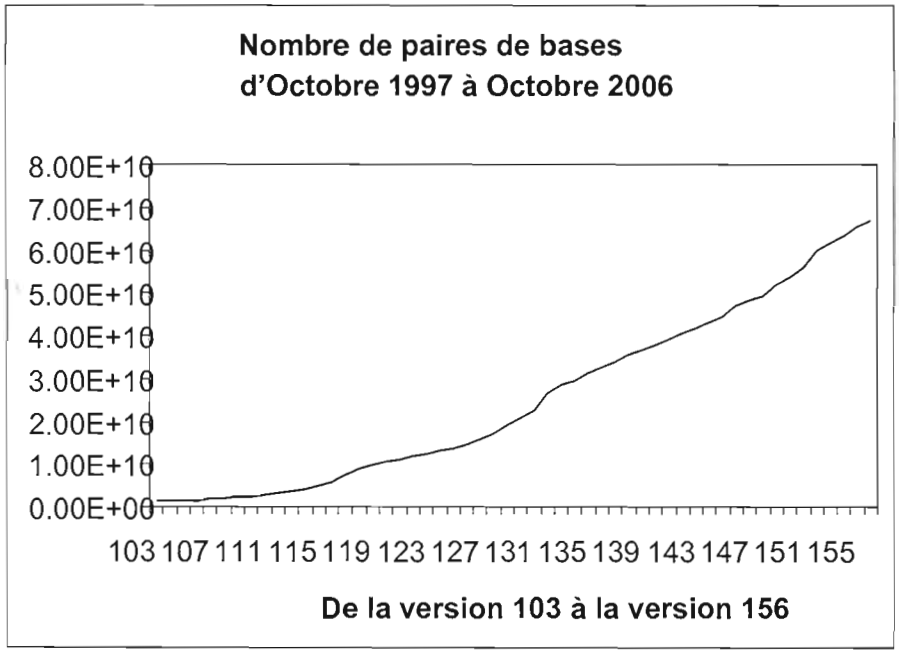

Figure 2.2 GenBank, données d'Octobre 2006 (de la version 103 à la Version 156) [8]

La figure suivante montre le nombre d'entrées et de bases d'ADN/ARN pour les 20 organismes les mieux séquencés dans la version 156.0 de GenBank.

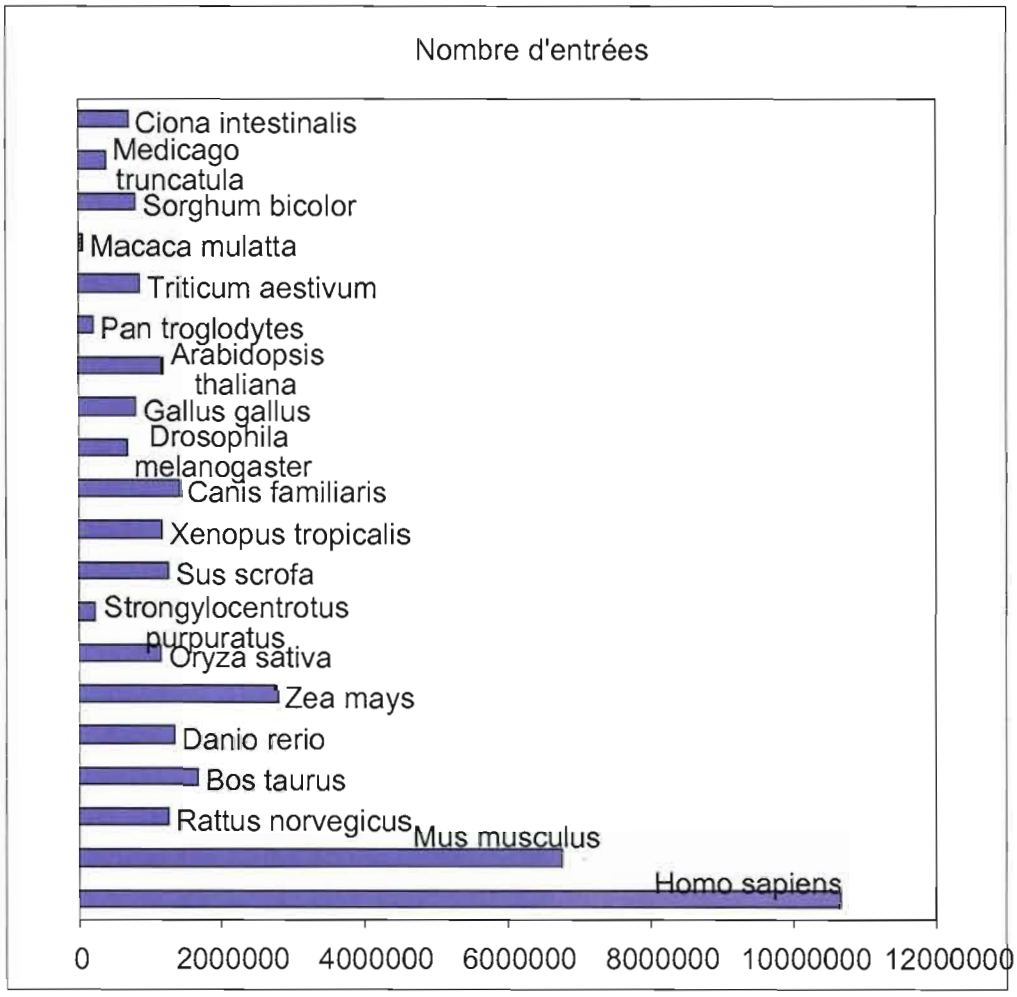

Figure 2.3 Nombre d'entrées dans GenBank, version 156.0, pour les 20 organismes les mieux séquencés

La figure ci-dessous montre le nombre de paires de bases de GenBank pour la même sélection de données que la figure précédente.

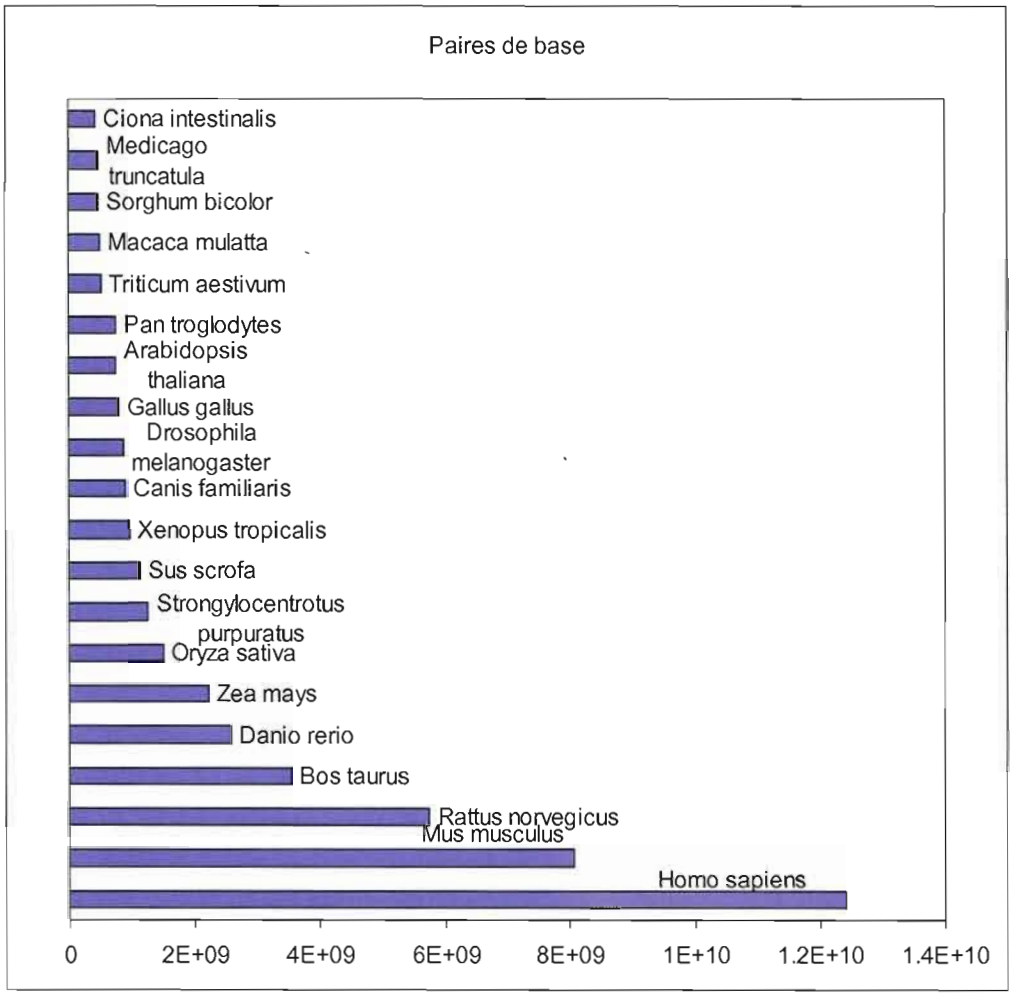

Figure 2.4 Nombre de paires de bases dans GenBank, version 156.0, pour les 20 organismes les mieux séquencés

2.3 Description des champs de GenBank

2.3.1 Format de stockage

La banque de données GenBank est composée d'un ensemble de fichiers texte, dans lesquels les séquences sont regroupées soit selon le critère taxonomique (e.g. virus, procaryotes, etc.), soit selon le critère d'origine (e.g. brevets, expressed sequence tags 'EST' et sequence tagged sites 'STS').

Une séquence est une entrée à laquelle on associe certains attributs tels que :

- Le nom : provenant du produit codé par la séquence et du nom de l'organisme d'origine;
- Le numéro d'accession;
- La structure;
- Le rôle biologique;
- La nature de la molécule qui a été séquencée;
- D'autres informations pertinentes.

Toutes ces informations sont structurées selon un format particulier.

Tableau 2.1 : Exemple d'entrée de GenBank (gène d'Influenza A virus) [3]

```
LOCUS       CY016511                847 bp ss-cRNA    linear   VRL 04-OCT-2006
DEFINITION  Influenza A virus (A/New South Wales/2/1999(H3N2)) segment 8,
            complete sequence.
ACCESSION   CY016511
VERSION     CY016511.1  GI:115521606
KEYWORDS    .
SOURCE      Influenza A virus (A/New South Wales/2/1999(H3N2))
  ORGANISM  Influenza A virus (A/New South Wales/2/1999(H3N2))
            Viruses; ssRNA negative-strand viruses; Orthomyxoviridae;
            Influenzavirus A.
REFERENCE   1  (bases 1 to 847)
  AUTHORS   Spiro,D., Sengamalay,N., Boyne,A., Halpin,R., Wang,S., Ghedin,E.,
            Zaborsky,J., Subbu,V., Sparenborg,J., Gallagher,T., Liu,X.,
            Salzberg,S.L., Sitz,J., Katzel,D., Neupane,R., Shumway,M., Koo,H.,
            Griesemer,S., St. George,K., Bennett,R., Taylor,J., Rawlinson,W.,
            Bao,Y., Bolotov,P., Dernovoy,D., Kiryutin,B., Lipman,D.J. and
            Tatusova,T.
  TITLE     The NIAID Influenza Genome Sequencing Project
  JOURNAL   Unpublished
REFERENCE   2  (bases 1 to 847)
  CONSRTM   The NIAID Influenza Genome Sequencing Consortium
  TITLE     Direct Submission
```

```
  JOURNAL   Submitted (04-OCT-2006) on behalf of TIGR/Wadsworth-NYSDOH and
            Prince of Wales Hospital/NCBI, National Center for Biotechnology
            Information, NIH, Bethesda, MD 20894, USA
FEATURES             Location/Qualifiers
     source          1..847
                     /organism="Influenza A virus (A/New South
                     Wales/2/1999(H3N2))"
                     /mol_type="viral cRNA"
                     /strain="A/New South Wales/2/1999"
                     /serotype="H3N2"
                     /specific_host="Human"
                     /db_xref="taxon:400325"
                     /segment="8"
                     /lab_host="MDCK Unknown passage(s)"
                     /country="Australia: Sydney, NSW"
                     /collection_date="1999"
     gene            7..844
                     /gene="NS2"
     CDS             join(7..36,509..844)
                     /gene="NS2"
                     /codon_start=1
                     /product="nonstructural protein 2"
                     /protein_id="ABJ09256.1"
                     /db_xref="GI:115521608"
                     /translation="MDSNTLSSFQDILLRMSKMQLGSSSEGLNGMITQFESLKIYRDS
                     LGEAVMRMGDLHLLQNRNGKWREQLGQKFEEIRWLIEEVRHRLKTTENSFEQITFMQA
                     LQLLFEVEQEIRTFSFQLI"
     gene            7..699
                     /gene="NS1"
     CDS             7..699
                     /gene="NS1"
                     /codon_start=1
                     /product="nonstructural protein 1"
                     /protein_id="ABJ09255.1"
                     /db_xref="GI:115521607"
                     /translation="MDSNTLSSFQVDCFLWHIRKQVVDQELSDAPFLDRLRRDQRSLR
                     GRGNTLGLDIKAATHVGKQIVEKILKEESDEALKMTMASTPASRYITDMTIEELSRNW
                     FMLMPKQKVEGPLCIRMDQAIMEKNIMLKANFSVIFDRLETIVLLRAFTEEGAIVGEI
                     SPLPSFPGHTIEDVKNAIGVLIGGLEWNDNTVRVSKNLQRFAWRSSNENGGPPLTPKQ
                     KRKMARTARSKV"
ORIGIN
        1 gacataatgg attccaacac tttgtcaagt ttccaggtag attgctttct ttggcatatc
       61 cggaaacaag ttgtagacca agaactgagt gatgccccat tccttgatcg gcttcgccga
      121 gatcagaggt ccctaagggg aagaggcaac actctcggtc tagacatcaa agcagccacc
      181 catgttggaa agcagattgt agaaaagatt ctgaaggaag aatctgatga ggcacttaaa
      241 atgaccatgg cctccacacc tgcttcgcga tacataactg acatgactat tgaggaattg
      301 tcaagaaact ggttcatgct aatgcccaag caaaaagtgg aaggacccct ttgcatcaga
      361 atggaccagg caatcatgga gaaaaacatc atgttgaaag cgaatttcag tgtgatcttt
      421 gaccgactag agaccatagt attactaagg gctttcaccg aagagggagc aattgttggc
      481 gaaatctcac cattgccttc ttttccagga catactattg aggatgtcaa aaatgcaatt
      541 ggggtcctca tcggagggct tgaatggaat gataacacag ttcgagtctc taaaaatcta
      601 cagagattcg cttggagaag cagtaatgag aatgggggac ctccacttac tccaaaacag
      661 aaacggaaaa tggcgagaac agctaggtca aaagtttgaa gagataagat ggctgattga
      721 agaagtgaga cacagactaa aaacaactga gaatagtttt gagcaaataa cattcatgca
      781 agcattacag ctgctgtttg aagtggaaca ggagataaga actttctcat ttcagcttat
      841 ttaataa
//
```

2.3.2 Description des champs

Ci-dessous, nous présentons une description sommaire des champs d'entrée d'une séquence.

1. LOCUS

Un nom court pour l'entrée en format mnémonique. Il est choisi de telle sorte qu'il décrit bien la définition de la séquence. C'est un mot clé obligatoire (il s'agit d'un seul enregistrement).

Le format détaillé de la ligne LOCUS est le suivant :

01-05	'LOCUS'
06-12	Espaces
13-28	Nom du Locus
29-29	Espace
30-40	Longueur de la séquence, justifiée à droite
41-41	Espace
42-43	bp
44-44	Espace
45-47	Espace, ss- (single-stranded), ds- (double-stranded), or ms- (mixed-stranded)
48-53	NA, DNA, RNA, tRNA (transfer RNA), rRNA (ribosomal RNA), mRNA (messenger RNA), uRNA (small nuclear RNA), snRNA, snoRNA. Justifié à gauche.
54-55	Espace
56-63	'linéaire' suivi de deux espaces, ou 'circulaire'
64-64	Espace
65-67	Code de la division
68-68	Espace
69-79	Date, avec le format dd-MMM-yyyy (e.g., 15-MAR-1991)

2. DEFINITION

Une description concise de la séquence. Elle commence du plus général vers le plus spécifique. C'est un mot clé obligatoire.

3. NUMÉRO ACCESSION

Le numéro d'accession primaire est unique et ne peut être modifié. Il comprend 6 caractères ou 8 caractères appelés 'accession numbers'. Le numéro d'accession à 6 caractères est composé d'une lettre en majuscule et de 5 chiffres (par exemple, A12345). Celui de 8 caractères est composé de 2 lettres en majuscule et de 6 chiffres (par exemple, AC123456). C'est un mot clé obligatoire (il peut s'agir d'un ou de plusieurs enregistrements).

4. VERSION

La version est un identifiant composé d'un numéro d'accession et d'un numéro de version associé à la version actuelle de la séquence dans l'enregistrement. Il est suivi par une clé entière (GI) affectée par le NCBI. C'est un mot clé obligatoire (il peut s'agir d'un ou de plusieurs enregistrements). Voici un exemple de version :

```
ACCESSION   AF181452
VERSION     AF181452.1  GI:6017929
            ^^^^^^^^^^  ^^^^^^^^^^
            Compound    NCBI GI
            Accession   Identifier
            Number
```

5. NID

Une méthode alternative pour représenter l'identifiant GI de NCBI mentionné ci-dessus.

6. KEYWORDS

Il s'agit de phrases courtes décrivant les produits génétiques et d'autres informations concernant l'enregistrement. C'est un mot clé obligatoire pour les entrées annotées et (il peut s'agir d'un ou de plusieurs enregistrements).

7. SOURCE

La source est le nom commun de l'organisme ou le nom le plus fréquemment utilisé dans la littérature. Il s'agit d'un mot clé obligatoire pour toutes les entrées annotées. Il peut être dans un ou plusieurs enregistrements. Il inclut un sous-mot clé.

a. ORGANISM

Nom scientifique et officiel de l'organisme (en première ligne de l'enregistrement) ainsi que les niveaux de classification taxonomique (deuxième et troisième ligne de l'enregistrement). C'est un sous-mot clé obligatoire dans toutes les entrées annotées (il peut s'agir de deux ou de plusieurs enregistrements).

8. REFERENCE

Ce descripteur comprend les références aux articles contenant les données pour cette entrée. Il inclut 7 sous-mots clés qui peuvent se répéter. Mots clés obligatoires (il peut s'agir d'un ou plusieurs enregistrements).

a. AUTHORS

Liste les auteurs de la citation. Sous-mot clé optionnel (il peut exister un ou plusieurs enregistrements).

b. CONSRTM

Liste les noms des consortiums associés à la citation (exemple : International Human Genome Sequencing Consortium). C'est un sous-mot clé optionnel (il peut s'agir d'un ou de plusieurs enregistrements).

c. TITLE

C'est le titre complet de la citation. Il est optionnel et peut comprendre un ou plusieurs enregistrements.

d. JOURNAL

Il comprend le nom du journal, le volume, l'année et les numéros des pages de la citation. C'est un sous-mot clé obligatoire (il peut s'agir d'un ou de plusieurs enregistrements).

e. MEDLINE

Il fournit l'identifiant unique Medline de la citation. Il est optionnel (il peut s'agir d'un ou de plusieurs enregistrements.

f. PUBMED

Il fournit l'identifiant unique Pubmed de la citation. Il est optionnel (il peut s'agir d'un ou de plusieurs enregistrements).

g. REMARK

Il montre l'importance de la citation par rapport à l'entrée. Il est optionnel (il peut s'agir d'un ou de plusieurs enregistrements).

Voici un exemple de REFERENCE et ses sous-mots clés :

```
REFERENCE   1  (bases 1 to 1440)
  AUTHORS   Gonzalez,J.M., Laiz,L. and Saiz-Jimenez,C.
  TITLE     Classifying bacterial isolates from hypogean environments:
            Application of a novel fluorimetric method dor the estimation of
            G+C mol% content in microorganisms by thermal denaturation
            temperature
  JOURNAL   (in) Saiz-Jimenez,C. (Ed.);
            MOLECULAR BIOLOGY AND CULTURAL HERITAGE: 47-54;
            A.A. Balkema, The Netherlands (2003)
```

9. COMMENT

Ce descripteur contient des références croisées avec d'autres entrées de séquences, des comparaisons avec d'autres collections, des commentaires sur les changements de nom LOCUS et d'autres remarques. C'est un mot clé optionnel (il peut s'agir d'un ou de plusieurs enregistrements).

10. FEATURES

Il s'agit d'une table contenant des informations sur des portions de séquence qui code les molécules d'ARN et d'autres informations utiles de type expérimental. Voici une partie de cette table. Pour de plus amples informations sur les FEATURES, référez à l'annexe I.

```
gene          Region that defines a functional gene, possibly
              including upstream (promotor, enhancer, etc)
              and downstream control elements, and for which
              a name has been assigned.
GC_signal     `GC box' in eukaryotic promoters
iDNA          Intervening DNA eliminated by recombination
intron        Transcribed region excised by mRNA splicing
J_region      Span of the J immunological feature
LTR           Long terminal repeat
```

Ci-dessous retrouve un exemple de la table FEATURES :

```
1    10    20    30    40    50    60    70    79
---------+---------+---------+---------+---------+---------+---------+---------
   CDS          5..1261
                /product="alpha-1-antitrypsin precursor"
                /map="14q32.1"
                /gene="PI"
   tRNA         1..87
                /note="Leu-tRNA-CAA (NAR: 1057)"
                /anticodon=(pos:35..37,aa:Leu)
   mRNA         1..>66
                /note="alpha-1-acid glycoprotein mRNA"
   transposon   <1..267
                /note="insertion element IS5"
   misc_recomb  105^106
```

```
                /note="B.subtilis DNA end/IS5 DNA start"
     conflict   258
                /replace="t"
                /citation=[2]
---------+---------+---------+---------+---------+---------+---------+---------
1    10    20    30    40    50    60    70    79
```

11. BASE COUNT

Ce descripteur comprend les sommes des nombres d'occurrence de chaque code de paires de bases (A, C, T ou G pour l'ADN) dans la séquence. C'est un mot clé optionnel et c'est un seul enregistrement.

Remarque : ce champ est maintenant obsolète, il a été enlevé de la base de donnée GenBank en Octobre 2003.

12. CONTIG

Ce type de ligne fournit de l'information sur la manière de combiner des séquences individuelles pour former des objets biologiques plus grands, comme des chromosomes ou des génomes complets. Un bon exemple d'utilisation de CONTIG serait celui d'Août 2005 à savoir le chromosome 2L de *Drosophila melanogaster*. Voici un exemple de l'utilisation de CONTIG.

```
CONTIG    join(AE003590.3:1..305900,AE003589.4:61..306076,
          AE003588.3:61..308447,AE003587.4:61..314549,AE003586.3:61..306696,
          AE003585.5:61..343161,AE003584.5:61..346734,AE003583.3:101..303641,

          [ lines removed for brevity ]

          AE003782.4:61..298116,AE003783.3:16..111706,AE002603.3:61..143856)
```

13. ORIGIN

Ce descripteur spécifie comment la première base de la séquence est située dans le génome. Autant que possible, ceci inclut l'emplacement dans la carte génétique. Ce mot clé est obligatoire et contient exactement un enregistrement.

- La ligne ORIGIN est suivie par les données de la séquence (plusieurs enregistrements).
- Le symbole de fin d'entrée est obligatoire.

2.3.3 Composition de la Base de données GenBank :

La base de données GenBank est partagée en 18 divisions que sont présentées dans le tableau 2.2:

Tableau 2.2 : Taxonomie de GenBank

1. PRI - primate sequences : (séquences des primates). 2. ROD - rodent sequences : (séquences des rongeurs). 3. MAM - other mammalian sequences : (séquences des autres collagènes de mammifères). 4. VRT - other vertebrate sequences : (séquences des autres vertébrés). 5. INV - invertebrate sequences : (séquences des invertébrés). 6. PLN - plant, fungal, and algal sequences : (séquences des plantes, des champignons et des algues). 7. BCT - bacterial sequences : (séquences des bactéries). 8. VRL - viral sequences : (séquences des virus). 9. PHG - bacteriophage sequences (séquences de bactériophage). 10. SYN - synthetic sequences (séquences des synthétiques). 11. UNA - unannotated sequences (séquences non-annotées). 12. EST - EST sequences (expressed sequence tags) : (séquences des EST). 13. PAT - patent sequences : (séquences brevetées). 14. STS - STS sequences (sequence tagged sites) : (séquences des STS). 15. GSS - GSS sequences (genome survey sequences) : (séquences d'enquête de génome). 16. HTG - HTG sequences (high-throughput genomic sequences): (séquences génomiques de haut débit). 17. HTC - unfinished high-throughput cDNA sequencing (séquences cDNA à haut débit). 18. ENV - Environmental sampling sequences : (séquences environnementales).

Parmi ces divisions, on retrouve des séquences provenant de groupes d'organisations différents, tandis que d'autres séquences (EST, GSS, HTG, etc.) sont générées par des technologies de séquençage provenant d'organismes spécifiques. Pour ce dernier type de séquences, il est recommandé d'utiliser le navigateur de taxonomie de NCBI.

La division CON, a été ajoutée dans la version 115.0 (décembre 1999) mais ne figure pas encore dans la liste ci-dessus puisque elle est encore dans un stade expérimental. Dans la division CON, les enregistrements ne contiennent aucune donnée de séquences, mais contiennent des instructions sur la manière de construire des CONTIGS à partir d'autres enregistrements de GenBank.

2.4 Communication avec GenBank

2.4.1 Soumission à GenBank [4]

Les principales sources de données pour GenBank sont les soumissions directes par les chercheurs scientifiques. NCBI fournit un traitement adéquat, précis et un examen biologique pour les nouvelles entrées et offre des mises à jour pour les entrées existantes.

Il y a deux moyens efficaces pour les soumissions :

- L'outil Internet des soumissions : appelé BankIt, pour une soumission simple et rapide des données des séquences.

- L'outil Sequin : logiciel autonome de soumission à NCBI pour MAC (Macintosh), PC et UNIX, qui est disponible par FTP. En utilisant Sequin, les dossiers résultats de la soumission directe sont envoyés à GenBank par courrier électronique.

Pour un bon contrôle des annotations des entrées, des enregistrements segmentés, ou des entrées très longues, Sequin, est l'outil le plus approprié.

De nombreuses revues exigent la soumission des informations sur les séquences à une base de données avant la publication de sorte qu'un numéro d'accession puisse apparaître dans le papier.

2.4.2 Réception de numéro d'accession

Il est coutumier que les séquences d'ADN et d'acides aminés apparaissant dans les articles scientifiques soient préalablement soumises à une base de données de séquences. Après soumission, l'auteur reçoit un numéro d'accession de la base de données qu'il peut indiquer dans son article comme référence à sa séquence. Les bases de données GenBank, EMBL et DDBJ s'échangent des données sur une base journalière. Des données de séquences soumises avant la publication peuvent être maintenues confidentiellement si l'auteur le désire.

Dans le cas d'une soumission par courrier électronique, GenBank fournit à l'auteur un numéro d'accession relatif à sa séquence, habituellement, dans les deux jours ouvrables qui suivent. Ce numéro d'accession servira, entre autre, comme accusé de réception pour les données soumises. Le numéro d'accession doit être inclus dans le manuscrit, de préférence dans une apostille à la première page de l'article.

2.4.3 BankIt

BankIt permet de gérer les informations relatives aux séquences selon un format prédéfini. Elle permet aussi l'ajout d'annotations biologiques (par exemple, codifications des régions, des dispositifs de mRNA). BankIt transforme les données selon le format de GenBank. Quand l'enregistrement est prêt, il peut alors être soumis directement à GenBank. Il existe aussi l'option d'ajouter des informations en utilisant des boîtes de texte afin de décrire d'une façon personnalisée la source de la séquence et ses dispositifs biologiques. Le personnel d'annotation de GenBank passe en revue l'information textuelle soumise, l'incorpore dans des champs structurés appropriés, et renvoie l'enregistrement par courriel à la revue.

2.4.4 Sequin

Le NCBI fournit aussi un programme autonome de soumission appelé Sequin qui est rendu actuellement à sa version 6.00. C'est un logiciel multi-plateforme (Mac, PC/Windows et UNIX). Il est interactif et graphique. Sequin est conçu pour simplifier le processus de soumission des séquences, fournir un visionnement graphique et des options d'édition. Il permet une vérification robuste des erreurs et est très bien adapté aux séquences longues et aux annotations complexes.

2.4.5 SequinMacroSend

Cet outil a été conçu principalement pour pallier au problème des gros fichiers attachés aux courriels. En effet, SequinMacroSend permet de télécharger de grands fichiers (.sqn) directement de GenBank. Il suffit d'introduire les informations moyennant l'interface graphique de SequinMacroSend, télécharger le dossier (.sqn) et de l'envoyer directement au personnel de soumission de GenBank.

2.4.6 TBL2ASN

Tbl2asn est un programme de ligne de commande qui sert à automatiser la création des enregistrements de séquence à soumettre pour GenBank. Il utilise plusieurs fonctions similaires à Sequin, mais utilise des fichiers de données. Tbl2asn génère des dossiers sqn pouvant être soumis à GenBank.

Exemple :
- Soumission simple : une séquence par dossier de fsa

tbl2asn -t template.sbt -p path_to_files –v

2.4.7 Soumissions spéciales : Génomes, séquences en lots, alignements

Sequin peut être utilisé pour une soumission simple ou multiple. Il est également conçu pour faciliter le traitement des types spéciaux de soumissions et devrait être

utilisé à la place de BankIt pour les types de soumissions suivants : génomes et d'autres séquences longues ; séquences multiples telles que des soumissions en lots et des ensembles segmentés, et finalement pour les études de population/phylogénétique/mutation. En préparant la soumission d'un génome dans Sequin on peut aussi bien importer une séquence complète d'un génome qu'un dossier contenant les traductions d'acides aminés dans le format FASTA. Le dossier final de soumission généré (*. sqn) sera trop grand. Dans ce cas, on peut l'envoyer au personnel de GenBank via ftp plutôt que par courriel.

2.4.8 Envoi des données à GenBank

En utilisant BankIt, les entrées préparées sont soumises directement à GenBank via Internet. En utilisant Sequin, les fichiers résultats sont envoyés à GenBank par courrier électronique à : gb-sub@ncbi.nlm.nih.gov ou en téléchargeant les dossiers avec SequinMacroSend.

2.4.9 Soumission de SNPs et d'autres données de polymorphisme

Des données portant sur la variation génétique des humains ou provenant d'autres organismes peuvent aussi êtres soumises à la base de données des polymorphismes simples de nucléotides (dbSNP) de NCBI. La base de données dbSNP est séparée de la base de données de GenBank. En plus, les soumissions qui y sont destinées ne reçoivent pas de numéros d'accession de GenBank. Cependant, ces entrées reçoivent des identifiants uniques de dbSNP et contiennent des liens aux enregistrements associés de GenBank.

CHAPITRE III

BASE DE DONNÉES GBANK UQAM

3.1 Introduction

La base de données GBank de l'UQAM a été développée afin d'adapter les données génétiques de NCBI pour leur utilisation en analyse phylogénétique. La base de données GenBank de NCBI contient seulement les informations générales qui exigent de nombreux traitements avant de les rendre utiles aux phylogénéticiens. En effet, nous énumérons les problèmes suivants qui ont mené au développement de GBank UQAM :

- Certaines requêtes complexes utilisées par les bioinformaticiens sont lents dû notamment à la taille énorme et toujours croissante de la base de données.
- En tant qu'utilisateur, nous n'avons aucun contrôle sur la base de données GenBank. En plus nous dépendons des mises à jours du NCBI.
- Les outils de GenBank pour le filtrage des données ne sont pas toujours adaptés à nos besoins. Ceci nous mène à nous plier au mode de fonctionnement de la base GenBank.

GBank UQAM se voit donc un sous ensemble de la base GenBank, qui résout en totalité ou partiellement les problèmes posés ci-dessus. La nouvelle base de l'UQAM permettrait donc de :

- **Améliorer le temps de réponse** : Étant traité localement, nous pouvons offrir un temps d'accès (le filtrage des données selon des critères spécifiées) nettement meilleur. Cette amélioration a été constatée par les utilisateurs de l'UQAM.
- **Mieux contrôler les données** : Nous pouvons organiser les données selon nos besoins et donc rendre la base de données optimale. En effet, maintenant on est capable de filtrer les données selon nos besoins spécifiques ce qui augmente nettement notre productivité. En particulier seulement les données nécessaires pour effectuer une analyse phylogénétique seront incluses dans la base de données.
- **Optimiser la base de données** : Maintenant, avec des temps de réponses améliorés et plus de maniabilité dans la gestion de la base de données de l'UQAM, il nous est possible d'optimiser continuellement notre base de données pour la rendre plus évolutive et plus adaptée à nos besoins futurs.

Dans les sections qui suivront nous allons présenter une vue globale du travail qui a été réalisé pour atteindre les objectifs cités ci-dessus. Ensuite, nous présenterons la nouvelle base de données de l'UQAM et les différents processus d'importation et de vérification des données. Finalement, nous montrerons les avantages de certaines techniques comme le partitionnement, sur les performances de la base de données GBank UQAM.

Notre projet contient trois parties principales comme l'indique la figure 3.1:

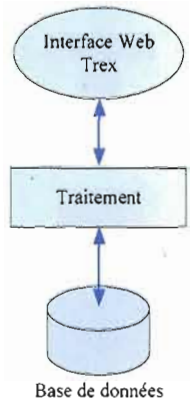

Figure 3.1 Vue global du projet GBank UQAM

- **Interface web** : Il s'agit de la couche présentation de l'application. C'est une interface Web qui fournit des outils de consultation et de filtrage visuels des données permettant d'interroger la base de données et d'afficher les résultats des requêtes et des traitements. Cette partie sera détaillée dans le chapitre 4.

- **Traitement** : C'est la couche '*Middleware*' de l'application. Elle contient les objets d'affaires et toute la logique de l'application. En d'autres termes, il s'agit des traitements et des fonctionnalités de l'application. Cette partie sera détaillée dans le chapitre 4.

- **Base de données** : Il s'agit de la base de données GBank développée à l'UQAM. Elle contient des données extraites de la base internationale et d'autres provenant des entrées spécifiques à l'UQAM. Cette base contient des tables allégées de la base internationale mais sa structure demeure quasiment la même pour des raisons de compatibilité. Cette partie sera détaillée dans la section suivante.

  Dans cette section, nous allons présenter la conception de la base de données GBank UQAM.

3.2 Conception de la base de données

Après l'analyse du projet, nous avons décomposé la base de données GBank en trois couches, la couche SCHEMA qui représente la structure de la base de données GBank UQAM, la couche physique qui représente l'architecture matérielle et finalement la couche qui représente la gestion des index et des partitions afin d'assurer une bonne performance.

3.2.1 Couche 1 : Structure de la base de données GBank UQAM

La base de données GBank UQAM est basée sur trois objets qui représentent le cœur de la base de données ainsi que d'autres tables secondaires (toutes les données recherchées et traitées par les utilisateurs sont dans ces trois tables). Ces objets sont les tables **SEQUENCES, FEATURES** et **ORGANISMS** qui se trouvent dans la première couche (voir Figure 3.2).

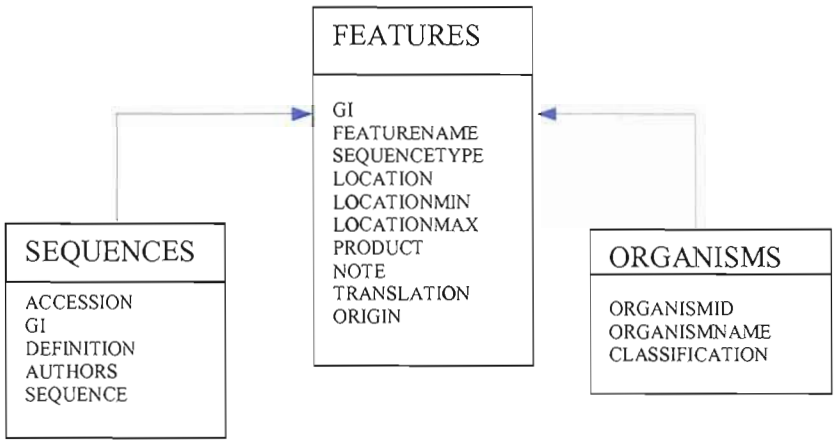

Figure 3.2 Les trois principales tables de GBank UQAM.

Description des tables

1. La table « SEQUENCES » : c'est une table qui regroupe toutes les informations concernant une séquence, telles que ACCESSION, GI, DEFINITION, AUTHORS, SEQUENCE, etc.

Figure 3.3 Écran de définition de la table SEQUENCES

Description des champs :

ACCESSION: Le champ Accession est une clé primaire de type alphanumérique de taille 20, qui représente le numéro d'accession d'une séquence.

GI: Est une clé secondaire de 20 caractères, qui correspond au GI (le numéro d'identification d'une séquence).

DEFINITION: Champ alphanumérique limité à 1000 caractères, qui représente la description concise de la séquence.

AUTHORS: Champ de 1000 caractères, qui correspond à la liste des auteurs de la citation.

SEQUENCE: Ensemble de données séquencées de tous les éléments de la table FEATURES de cette séquence de nucléotide.

2. **La table « FEATURES »** : c'est une table qui regroupe toutes les informations concernant un **Feature** (comme défini par NCBI), telles que GI, FEATURENAME, SEQUENCETYPE, LOCATION, LOCATIONMIN, LOCATIONMAX, PRODUCT, NOTE, TRANSLATION et ORIGIN.

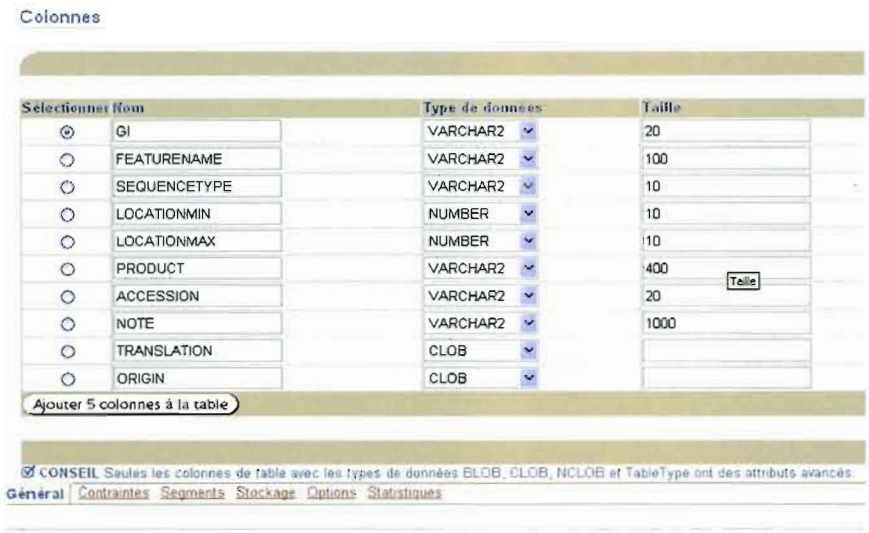

Figure 3.4 Écran de définitions de la table FEATURES

Description des champs :

GI (GenInfo Identifier): Numéro d'identification de la portion de la séquence correspondant à un **feature** traité.

FEATURENAME : Nom de gène du **feature** traité.

SEQUENCETYPE : Type de **feature** qui peut avoir les valeurs (CDS, mRNA, rRNA, tRNA).

LOCATIONMIN : Début de dispositif biologique de la base d'un **feature**.

LOCATIONMAX : Fin de dispositif de la base d'un **feature.**

PRODUCT : Nom d'un élément produit par un **feature** traité.

NOTE : Informations supplémentaires concernant un **feature**.

TRANSLATION : Synthèse d'acide aminé correspondant à la séquence de nucléotide programmée d'un **feature**.

ORIGIN : Sous ensemble de données séquencées qui correspondent à un **feature** traité (gène, RNA…).

3. **La table « ORGANISMS »** : c'est une table qui regroupe toutes les informations concernant un organisme, telles que ORGANISMID, ORGANISMNAME et CLASSIFICATION.

Figure 3.5 Écran de définitions de la table ORGANISMS

Description des champs :

ORGANISMID : Numéro d'identification taxonomique, il est unique et stable pour le taxon de l'organisme source.

ORGANISMNAME : Nom scientifique formel de l'organisme de source (genre et espèces).

CLASSIFICATION : Lignée de l'organisme de source, basée sur la classification phylogénétique utilisée dans la base de données de taxonomie de NCBI.

3.2.2 Couche 2 : Architecture de l'environnement

Vu l'énorme taille de la base de données GBank UQAM (environ 200 Giga présentement) et étant donné qu'elle peut doubler dans quelques années, nous avons essayé de concevoir une architecture adéquate pour la gestion de cette quantité d'information. En effet, nous avons choisi les plateformes logicielles et matérielles appropriées afin d'avoir la meilleure solution pour notre base de données (figure 3.6).

Voici donc notre choix de plateformes matériels et logiciels:

- Un serveur à deux microprocesseurs et à quatre disques durs. La taille de chaque disque est 200 Gig. Nous avons mis les fichiers d'index et les fichiers des données sur les disques différents pour avoir le meilleurs temps d'accès aux données (input/output).

- Un system d'exploitation Fedora (Linux), vu que le système Linux est stable et robuste et peut gérer efficacement les fichiers de grande taille comparé à d'autres systèmes d'explorations.

- Le langage de programmation Java qui offre la portabilité des fichiers binaires vers d'autres plateformes de système d'exploitation.

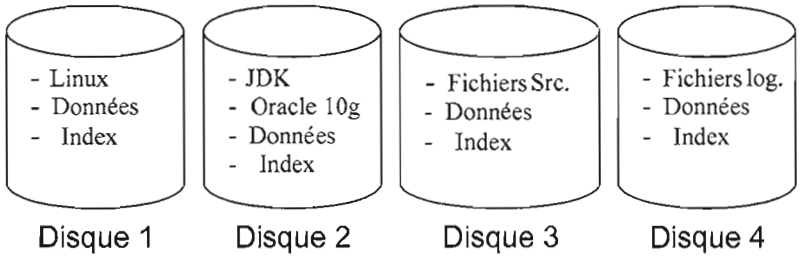

Figure 3.6 Plateformes matériels et logiciels

3.2.3 Couche 3 : Les index et les partitions

3.2.3.1 Les index

La base de données GBank UQAM étant volumineuse, on veut avoir de bonnes performances en réponse à des requêtes complexes. Ainsi, on a crée des index jumelés à une meilleure répartition des données grâce aux nouvelles technologies avancées d'Oracle 10g supportant les partitions de données [18].

Pour atteindre cet objectif, nous avons crée des espaces disques logiques pour les tables et les index, ainsi que pour les fichiers de données associés (voir Tableau 3.1). D'un point de vue pratique, la taille des fichiers de données doit être inférieur à 2 Giga afin d'avoir de bonnes performances [15]

Tableau 3.1 Organisation des fichiers physiques

Espace disque logique	Fichier de données
TBS_ORGANISMS	tbs_org.dbf
TBS_P1_FEATURES	tbs_p1_feat1.dbf tbs_p1_feat2.dbf
TBS_P2_FEATURES	tbs_p2_feat1.dbf tbs_p2_feat3.dbf tbs_p2_feat5.dbf tbs_p2_feat2.dbf tbs_p2_feat4.dbf tbs_p2_feat6.dbf
TBS_P3_FEATURES	tbs_p3_feat1.dbf tbs_p3_feat2.dbf
TBS_P_SEQUENCES	tbs_p_seq1.dbf tbs_p_seq2.dbf tbs_p_seq3.dbf tbs_p_seq4.dbf
TBS_IND_ORGANISMS	tbs_ind_org1.dbf
TBS_IND_SEQUENCES	tbs_ind_seq1.dbf tbs_ind_seq2.dbf tbs_ind_seq3.dbf tbs_ind_seq4.dbf
TBS_IND_P1_FEATURES	tbs_ind_p1_feat.dbf
TBS_IND_P2_FEATURES	tbs_ind_p2_feat1.dbf tbs_ind_p2_feat3.dbf tbs_ind_p2_feat2.dbf tbs_ind_p2_feat4.dbf
TBS_IND_P3_FEATURES	tbs_ind_p3_feat.dbf

La bonne répartition des fichiers physiques de données (datafiles) sur les différents disques, ainsi que le croisement avec les fichiers de données des index (voir Tableau 3.2), assurent une bonne performance (données et index associés dans des disques différents pour ne pas faire des lectures et écritures sur le même disque).

Tableau 3.2 Organisation des fichiers sur les 4 disques

Disque 1	Disque 2	Disque 3	Disque 4
tbs_p1_feat1.dbf tbs_p1_feat2.dbf tbs_p_seq1.dbf tbs_p_seq2.dbf	tbs_p3_feat1.dbf tbs_p3_feat2.dbf tbs_p_seq3.dbf tbs_p_seq4.dbf	tbs_org.dbf tbs_p2_feat1.dbf tbs_p2_feat3.dbf tbs_p2_feat5.dbf	tbs_p2_feat2.dbf tbs_p2_feat4.dbf tbs_p2_feat6.dbf
tbs_ind_p2_feat1.dbf tbs_ind_p2_feat3.dbf	tbs_ind_p2_feat2.dbf tbs_ind_p2_feat4.dbf	tbs_ind_org1.dbf tbs_ind_seq1.dbf tbs_ind_seq2.dbf tbs_ind_p1_feat.dbf	tbs_ind_seq3.dbf tbs_ind_seq4.dbf tbs_ind_p3_feat.dbf

3.2.3.2 Les partitions

I- Partition des tables

Dans notre base de données, la table FEATURES et la table SEQUENCES contiennent presque toutes les données comparées à la table ORGANISMS. Donc les deux tables ont fortement besoins d'être partitionnées.

1- Table FEATURES

Après l'analyse des données et les types des requêtes appliquées à la table FEATURES, nous avons décidé d'appliquer le type de partition composée sur cette table.

Nous avons partitionné la table en trois plages, suivant la valeur de la colonne FEATURENAME, parce que la plupart des accès aux données de cette table passent par la colonne FEATURENAME. Dans la table FEATURES on trouve

que 80% des valeurs de FEATURENAME commencent par la lettre 'n', et les 20% restant par les lettes situées entre 'a' et 'm' et entre 'o' et 'z'.

La taille de la table FEATURES étant grande, on va forcement appliquer la partition par hachage sur chaque partition par plage définie. Ensuite, diviser la première et la troisième partition en deux sous-partitions et la deuxième en 16 sous-partitions afin de mettre chaque sous-partition dans un fichier de données (datafile) de taille de 2 Giga au maximum (voir la figure 3.7).

Figure 3.7 Partitions de la table FEATURES

2- Table SEQUENCES

Puisque la taille de la table est très grande (plus de 25 Giga), la plupart des requêtes utilisées pour accéder aux données de la table SEQUENCES, passent par la colonne ACCESSION. Nous avons choisi de partitionner la table en 16 partitions par hachage en associant à chaque partition un fichier de données (datafile) de taille inférieur ou égale à 2 Giga.

Figure 3.8 Partitionnement de la table SEQUENCES

II- Partition des index

1- FEATURES

Selon les types de requêtes utilisées dans ce projet, nous avons choisi de créer les index suivants : Feat_FeaturesName, Feat_Accession et Feati_Gi_Location (figure 3.9, 3.10, 3.11):

Figure 3.9 Index Feat_FeaturesName

Figure 3.10 Index Feat_Accession

Figure 3.11 Index Feat_GI_Location

2- SEQUENCES

Généralement, les requêtes appliquées à des données de la table partitionnée SEQUENCES utilisent la colonne **OrganismId.** Donc, il est important de créer un index local pour cette colonne (figure 3.12).

Figure 3.12 Index Seq_OrganismsId

III- Avantages du partitionnement

Exemple 1 : Les Insertions.

En comparant les insertions des données sources (à partir des fichiers de données) dans une base de données ayant des tables partitionnées (le nouveau système) et une base de données ayant des tables non partitionnées (l'ancien système), nous constatons les écarts suivants (figure 3.13):

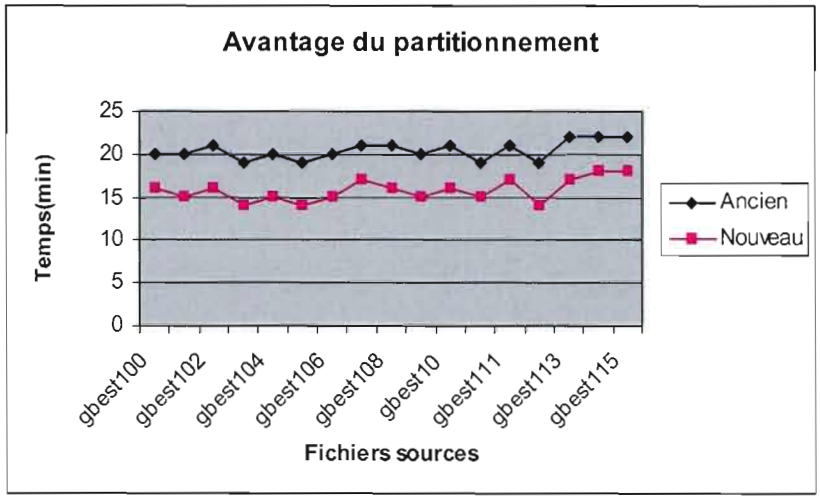

Figure 3.13 Avantage du partitionnement lors de l'insertion des données. L'axe des abscisses représente les fichiers sources, chaque fichier contient plusieurs séquences (i.e., enregistrements. L'axe des ordonnées représente le temps écoulé pour extraire et insérer les séquences d'un fichier source.

La courbe en haut (bleu foncée) montre les performances de l'ancien système dont les tables n'étaient pas partitionnées. La courbe en bas (rose) montre les performances du nouveau système dont les tables ont été partitionnées. L'amélioration en terme du temps de réponse à des requêtes est nette et l'apport du partitionnement est tangible.

Exemple 2 : La recherche.

En comparant les recherches dans une base de données ayant des tables partitionnées et dans une base de données ayant des tables non partitionnées, nous constatons les écarts suivants

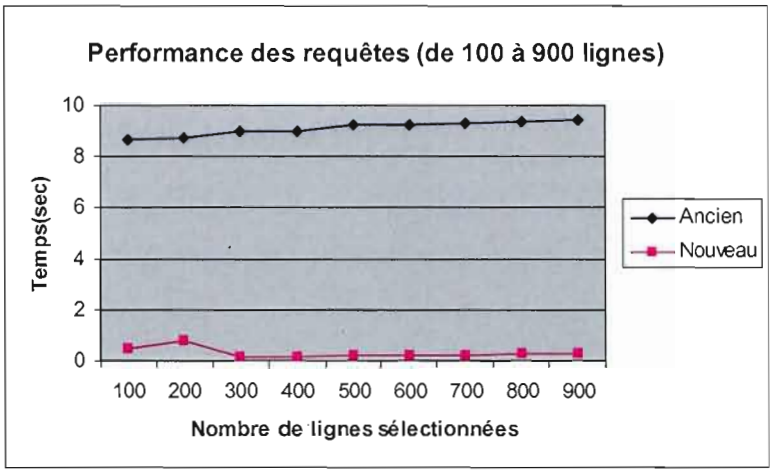

Figure 3.14 Performances du nouveau et de l'ancien système pour les requêtes de 100 à 900 lignes.

Pour des requêtes encore plus grandes, nous constatons de meilleures performances.

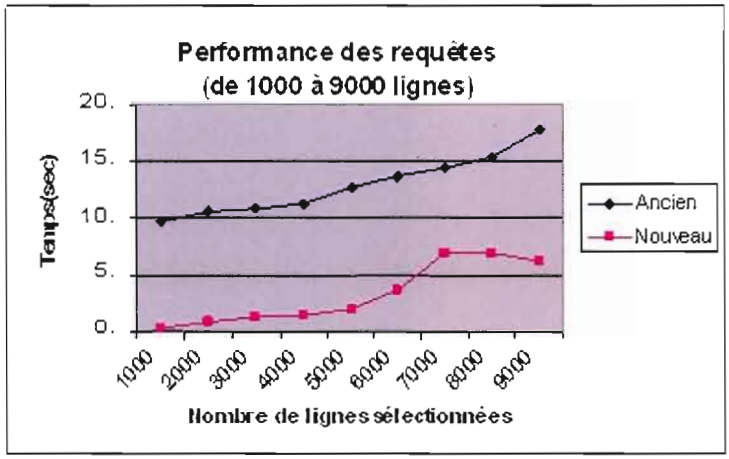

Figure 3.15 Performances du nouveau et de l'ancien système pour les requêtes de 1000 à 9000 lignes.

Ces figures montrent l'efficacité du partitionnement des tables pour les insertions et les sélections de données. En comparaison avec l'ancien système, la base de données est maintenant plus performante. Sur les figures 3.14 et 3.15, l'axe des abscisses

représente le nombre d'enregistrements retournés par une requête de recherche et l'axe des ordonnées représente le temps écoulé pour extraire les enregistrements.

3.3 Importation des données de GenBank

L'importation des données est un processus clé de notre projet. En effet, on a un bon nombre de fichiers sources à manipuler dans le premier chargement des données de GenBank (NCBI) (actuellement plus de 900 fichiers sources); la taille de chacun est d'environ 240 M (chaque fichier prend en moyenne 15 minutes à être téléchargé). Avec un simple calcul on arrive aux résultats suivants :

SomMinMoy = 900*15 = 13500 minutes,

NbrJoursMoy = 13500/ (60*24)= 10 jours.

Théoriquement, le chargement prend environ 10 jours (si le système marche 24 heures sur 24), mais en réalité, on effectue le chargement des fichiers par lot afin de s'assurer du bon déroulement du chargement, de la cohérence des données dans la base et de la vérification de la taille des fichiers de données (Datafiles). Donc, un chargement complet prend en moyenne entre 20 et 25 jours.

En plus, chaque deux mois, NCBI met à notre disposition trois fichiers : gbnew.txt, gbchg.txt et gbdel.txt qui contiennent respectivement la liste de numéros d'accession des nouvelles séquences insérées, des séquences modifiées et des séquences supprimées dans la base de données GenBank. Ainsi, nous pouvons effectuer les mises à jour de la base de données GBank UQAM. Ceci touche environ la moitié des fichiers sources. Ainsi, chaque deux mois nous consacrons en moyenne une dizaine de jours pour la mise à jour.

Contrairement à notre approche de mise à jour de la base de l'UQAM, le NCBI, dispose d'un outil puissant à savoir : FASTA.CMD, qui permet de faire la mise à jour de la base de données GenBank quotidiennement.

3.3.1 Processus d'importation des données de GenBank

Voici les principales étapes du processus d'importation que nous avons définies et adoptées pour importer les données de GenBank de NCBI vers GBank UQAM.

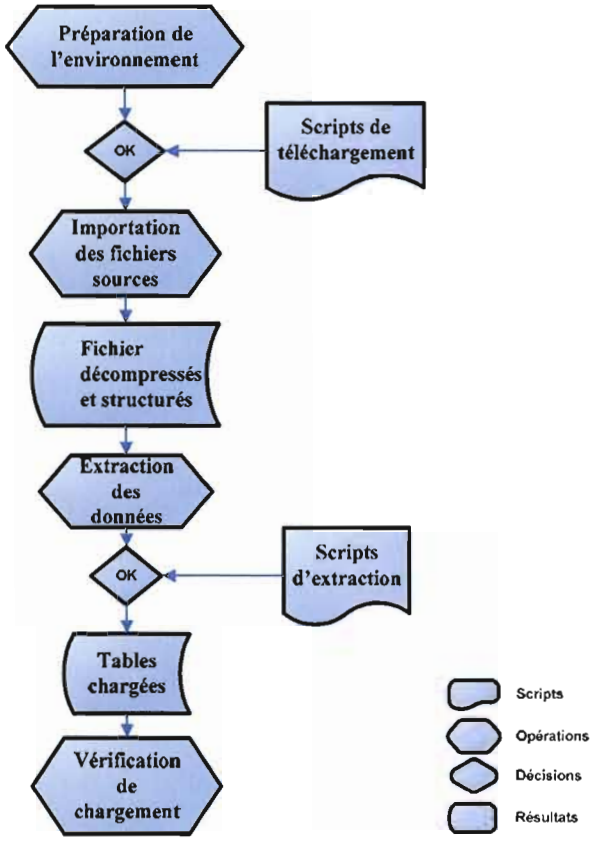

Figure 3.16 Processus utilisé pour l'importation des données de GenBank

Détails des étapes

Étape 1 : Préparation de l'environnement d'importation et d'extraction des données.

- Créer des répertoires (répertoires pour les fichiers sources 'compressés', les fichiers décompressés, les fichiers de traces 'logs' et les fichiers temporaires).
- Vérifier les ressources (espaces disque : dans notre cas on a besoin d'au moins 250 Gig).

Étape 2 : Préparation des scripts d'importation des fichiers sources de NCBI.

- Télécharger les fichiers source à partir du site de NCBI et les sauvegarder dans les répertoires spécifiés.
- Décompresser les fichiers sources dans les répertoires appropriés.

Étape 3 : Exécution des scripts d'extraction des données.

- Lire une séquence à la fois.
- Former un enregistrement qui correspond à la table **SEQUENCE** après la vérification des critères et des règles définies.
- Former les enregistrements des **TEATURES** associés à une séquence (suivant les règles définies).
- Former l'enregistrement de la table **ORGANISMS.**
- Insérer les enregistrements dans les tables correspondantes.
- S'assurer du bon déroulement du processus en terme de performance (la célérité, la consommation des ressources).

Dans cette étape et dans celle qui suivra, nous utilisons des scriptes Java que nous avons développés et qui nous permettrons d'extraire toutes les données sélectionnées grâces à des bibliothèques prédéfinies comme BIOJAVA. La

bibliothèque BIOJAVA contient toutes les fonctions nécessaires pour la manipulation des données qui se trouvent dans les fichiers sources de GenBank. [19]. N.B : le code du programme est fourni dans l'annexe B.

Étape 4 : Vérification de l'exportation et l'exécution des activités de formatage s'il y a lieu.

Dans cette étape, on effectue des tests de vérification afin de s'assurer du bon déroulement du processus d'import/export. Des requêtes SQL ont été définies pour connaître les volumes des enregistrements importés et les comparer avec ceux exportés. Des requêtes SQL ciblant la nouvelle base de l'UQAM ont été lancées afin de vérifier le contenu des enregistrements après la phase d'importation.

3.3.2 Processus de la mise à jour des données de GBank UQAM

La mise à jour des données de GBank UQAM est une tâche importante pour le projet. Elle doit être répétée chaque deux mois. On reçoit donc, la nouvelle version des fichiers sources, les nouvelles séquences, les séquences modifiées et les séquences supprimées. Les noms de fichiers sont respectivement *gbnew.txt*, *gbchg.txt* et *gbdel.txt*. Chaque ligne de ces fichiers est composée de deux champs séparés par une barre oblique. Le premier champ représente le nom du fichier où se trouve la séquence et le deuxième représente le numéro d'accession de la séquence (exemple : BCT1|AB001341)

Nos observations ont montré qu'à chaque mise à jour plus d'un tier de l'ensemble des données est affecté (en moyenne 300 fichiers sources).

Donc, pour avoir un bon déroulement et une meilleure performance de la mise à jour de la base de données, nous avons défini un processus adéquat tel qu'illustré dans la figure suivante (figure 3.17).

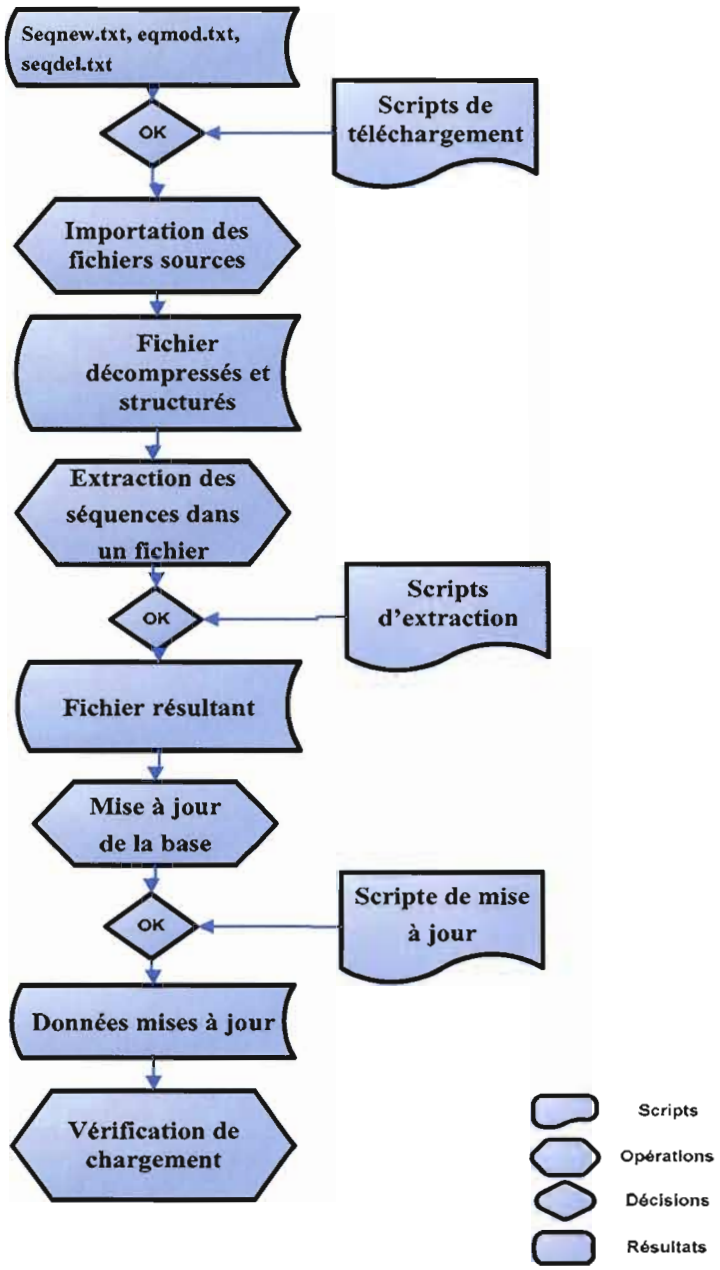

Figure 3.17 Processus de la mise à jour des données de GBank UQAM

Détails des étapes

Étape 1 : Préparation des scripts d'importation des fichiers sources de NCBI

- Télécharger à partir du site de NCBI les fichiers sources concernés par la mise à jour. Ils sont mentionnés dans les fichiers seqnew.txt, seqmod.txt et seqdel.txt. Sauvegarder ces fichiers dans les répertoires spécifiés (voir appendice E).

- Décompresser les fichiers sources dans les répertoires appropriés.

Étape 2 : Exécution des scripts d'extraction des séquences dans un fichier texte

Afin d'éviter de laisser la connexion avec la base de données ouverte lors de la mise à jour, nous avons stocké toutes les séquences concernées dans un fichier texte.

- Créer un nouveau fichier texte.
- Lire une séquence à la fois.
- Écrire la séquence dans un nouveau fichier.

Étape 3: Mise à jour des données

- Supprimer tous les enregistrements concernés par la mise à jour.
- Suivre l'étape 3 décrite dans la section 3.3.3.1.

Étape 4 : Vérification de l'exportation et l'exécution des activités de formatage s'il y a lieu

Idem l'étape 4 de la section 3.3.1.

CHAPITRE IV

INTERFACE WEB GBANK UQAM

4.1 Introduction

Cette partie décrit l'interface graphique 'GBank UQAM' conçue pour exploiter la base de données GBank UQAM et les exemples des traitements appliqués aux données. L'interface GBank UQAM est une partie importante de l'interface du serveur T-REX (Tree and Reticulogram Reconstruction).

Dans ce chapitre, nous présenterons en premier le serveur web T-REX. Ensuite, nous montrerons les principales composantes architecturales de l'interface web GBank UQAM et finalement, nous fourniront une description détaillée de chaque fonctionnalité que fournit l'interface Web GBank UQAM.

4.2 Présentation du serveur web de T-REX

L'interface web de T-Rex est très simple à utiliser. Elle comprend de nombreuses options et modes de calcul pour effectuer une analyse phylogénétique [28]. Elle a été développée par Vladimir Makarenkov, avec la collaboration d'un groupe de professeurs et d'étudiants. T-Rex est un logiciel supporté par plusieurs plateformes et systèmes d'exploitation. Il existe plusieurs versions de ce logiciel: pour DOS, Windows et Macintosh. La version Windows a été développée par Vladimir Makarenkov en langage C++. L'implémentation de la version Macintosh a été réalisée par Vladimir Makarenkov et Philippe Casgrain en C++ et Pascal.

Maintenant il existe une version web de T-Rex, développée par Alix Boc accessible à l'adresse suivante http://www.trex.uqam.ca à partir de laquelle les autres versions mentionnées peuvent être téléchargées.

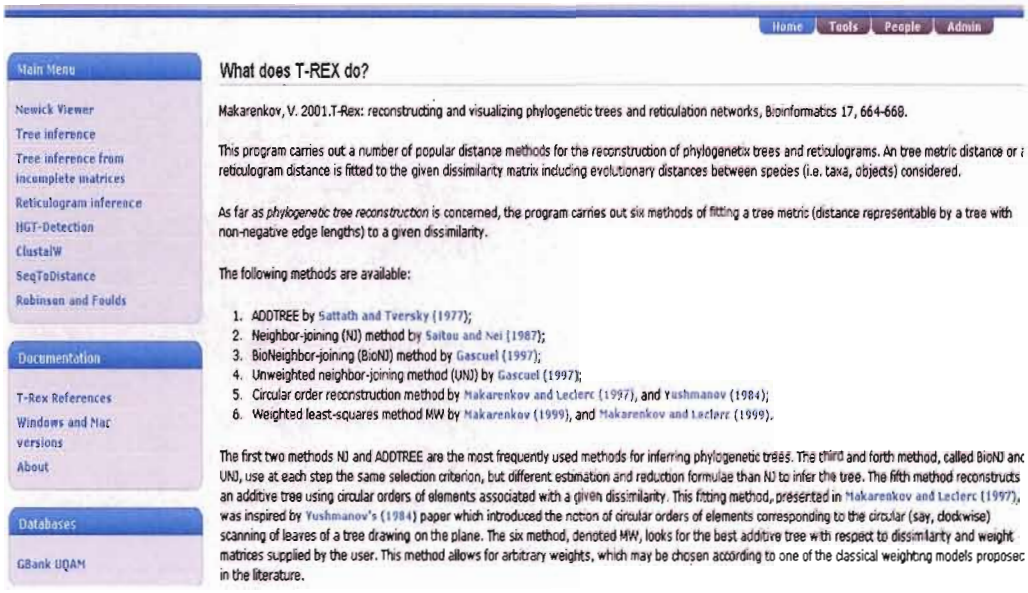

Figure 4.1 Le logiciel T-Rex

T-Rex permet la reconstruction des arbres phylogénétiques et des réseaux réticulés à partir d'une matrice de distance ou des séquences moléculaires. Il inclut un certain nombre de méthodes populaires d'inférence d'arbres phylogénétiques ainsi que de nouvelles méthodes d'analyse phylogénétique comme: la reconstruction d'arbres par les moindres carrés pondérés, l'inférence d'arbres à partir des matrices de distance incomplètes ou la modélisation d'un réseau réticulé pour un groupe d'objets ou d'espèces.

Dans T-Rex, il existe un lien vers l'interface web de GBank UQAM (Figure 4.1). Cette interface web coordonne la recherche des séquences de nucléotides des lignées des espèces dans la base de données.

4.3 Architecture de l'interface web de GBank UQAM

L'interface web de GBank UQAM est composée de 3 couches (Figure 4.2):

1. Couche Présentation : C'est le navigateur Web.
2. Couche Traitement : La couche métier comprend l'ensemble des traitements de GBank UQAM. Cette couche est divisée en deux parties :
    a. Le serveur d'application : Il s'agit des pages PHP qui contiennent des traitements liés notamment à la préparation des données à afficher dans la couche Présentation.
    b. La base de données : Les procédures stockées garantissent l'extraction des données des tables et vues de GBank UQAM.
3. Couche Données : Les données sont stockée dans quatre tables essentielles (Basket, SavedBasket, Usagers et Query).

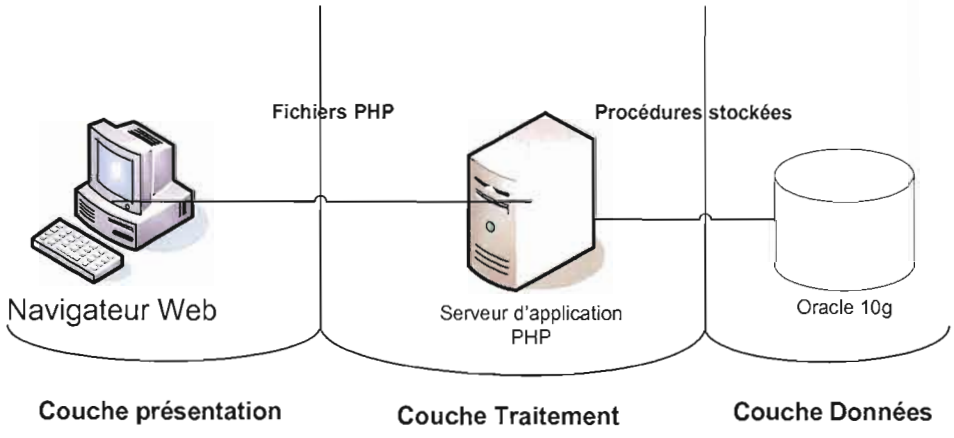

Figure 4.2 Architecture de l'interface web de GBank UQAM

4.3.1 Couche Présentation

La couche présentation n'est autre que les pages Web de GBank UQAM. Dans la section fonctionnalités du l'interface web GBank UQAM 4.4, nous avons présenté des exemples de ces pages web (par exemple, la figure 4.5 montre l'interface de recherche par nom du gène)

4.3.2 Couche Données

La base de données GBank UQAM comprend aussi les tables nécessaires au bon fonctionnement de système. Ces tables sont représentées sur la Figure 4.3. L'alimentation de ces tables intermédiaires se fait via les tables de GenBank (FEATURES, SEQUENCES et ORGANISMS).

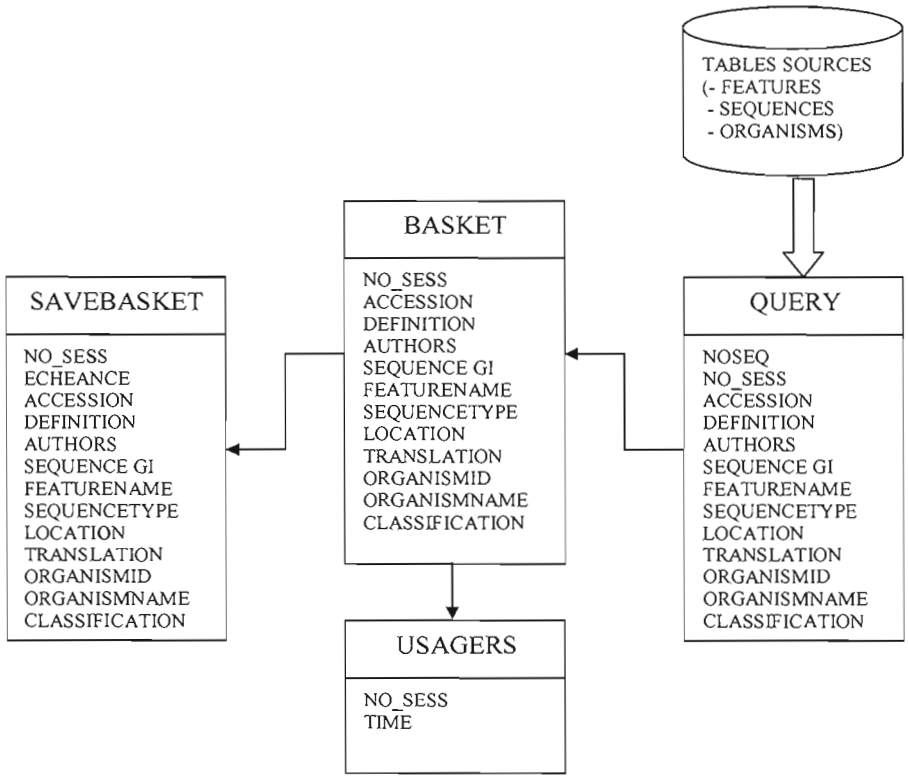

Figure 4.3 Couche Données Diagramme des tables de T-REX

4.3.3 Couche Traitement

Dans cette section, nous présenterons sous forme de tableau les fichiers sources (format PHP) de l'interface Web GBank UQAM et les procédures stockées d'Oracle utilisées dans la *couche Traitement.*

Tableau 4.1 Liste des fichiers PHP de la couche Traitement

Nom du fichier	Description
Connect_bd.php	Cette page assure la connexion à la BD
Index.php	Cette page sert d'accueil à l'usager tout en lui attribuant un numéro de session et un moment de départ. Les tables **Usagers**, **Query**, **Basket** et **Savedbasket** (Figure 4.3) sont nettoyées des enregistrements qui sont venue à l'échéance (7 jours pour **Savedbasket** et 6 heures pour les autres).
Basketgenes	Cette page est la page principale de la recherche. L'utilisateur y entre ses critères de recherche et soumet sa requête en cliquant sur "search" ou "advanced search" qui envoient à la page "list.php" traitent la requête. Les résultats sont ensuite affichés 25 par page, avec la possibilité de faire afficher les détails dans un pop-up en cliquant sur "view", ce qui envoie à la page "viewQ.php" et d'insérer l'élément dans le panier en cliquant sur "add to basket", ce qui envoie à la "page panier.php" qui effectuant les traitements. Le panier affiche également ses enregistrements, 25 par page, avec la possibilité pour chaque élément d'en voir les détails dans un pop-up en cliquant sur "view", ce qui envoie à la page

	"viewB.php"et de retirer l'élément du panier en cliquant sur "remove", ce qui envoie à la page "remove.php" effectuant les traitements. L'utilisateur peut sauvegarder son panier en cliquant sur "save basket", ce qui envoie à la page "save.php". Il peut également récupérer un panier sauvegardé en cliquant sur "open an existing basket", ce qui envoie à la page "saisiebasket.php". Finalement, en cliquant sur "Process data", on appelle la page "final.php", ceci est l'affichage final. L'usager peut sortir proprement du programme en cliquant sur "EXIT", ce qui l'amène à la page "close.php"
List.php	Cette page envoie la requête de recherche dans la base de données entière pour ensuite insérer les résultats dans la table "query". Les requêtes sont formulées selon que l'usager ait sélectionné *recherche par nom ou id d'espèce* et selon les mots-clés entrés dans les zones de textes. Le système recherche des enregistrements contenant ces mots.
Panier.php	Cette page permet d'ajouter un élément au panier suite à la demande de l'usager ayant cliqué sur "add to basket". L'élément est transféré de la table "Query" vers la table "Basket".
QueryB.php	Cette page permet de visualiser les détails d'un enregistrement obtenu d'une recherche dans la base de données. On appelle la table "Basket" pour obtenir le tuple ayant pour clé son numéro séquentiel et le numéro de session passés en

	paramètres.
QueryQ.php	Cette page permet de visusaliser les détails d'un enregistrement obtenu par recherche dans la base de données. On appelle la table "Query" pour obtenir le tuple ayant pour clé son numéro séquentiel et le numéro de session passés en paramètres.
Remove.php	Cette page permet de retirer un élément du panier suite à la demande de l'usager ayant cliqué sur "remove". L'élément est effacé de la table "Basket".
Restore.php	Cette page effectue le traitement sur le ID du panier sauvegardé saisi dans la page saisiebasket.php. Le contenu du panier sauvegardé est transféré dans la table Basket après que celle-ci fut nettoyée de son contenu. Un message avertit l'usager si le ID est erroné ou le délai a expiré.
Save.php	Cette page transfert les tuples contenus dans le panier dans une table prévue pour la sauvegarde du panier. L'usager est avisé de la durée de la sauvegarde et du numéro de panier sauvegardé.

Tableau 4.2 Liste des procédures stockées dans la couche Traitement

Nom de la fonction	Description
LOAD_RESULT (vGene, Specie, Mode, vNo_sess, number_rows)	Cette fonction envoie la requête de recherche dans la base de donnée pour ensuite insérer les résultats dans la table "Query". Cette fonction est appelée par le fichier "LIST.PHP".
INSERT_BASKET (no_seq, no_sess)	Cette fonction permet d'insérer les données d'un enregistrement au panier. L'enregistrement est transféré de la table "Query" vers la table "Basket". Cette fonction est appelée par le fichier "PANIER.PHP".
SAVE_BASKET (codeBasket, echeance, no_sess)	Cette fonction permet le transfert des tuples contenus dans le panier dans une table prévue pour la sauvegarde du panier "SAVEBASKET". Cette fonction est appelée par le fichier "SAVE.PHP".
RESTOREBASKET (codeBasket, no_sess)	Cette fonction effectue le transfert des enregistrements de la table "SAVEBASKET" vers la table "BASKET" après que celle-ci fut nettoyée de son contenu. Cette fonction est appelée par le fichier "RESTORE.PHP".

4.4 Fonctionnalités de GBank UQAM

Pour chaque requête à la base de données, les résultats sont affichés et l'information pertinente est sélectionnée et insérée dans le panier d'information. Quand toute information est sélectionnée dans le panier, l'utilisateur peut traiter les données pour obtenir les séquences de nucléotides et les lignées. Ces informations apparaîtront dans

une nouvelle page. Puis, l'utilisateur sera invité à lancer une nouvelle recherche ou à sortir du programme.

GBank UQAM contient des fonctionnalités de base qui sont résumées dans le processus suivant (Figure 4.4):

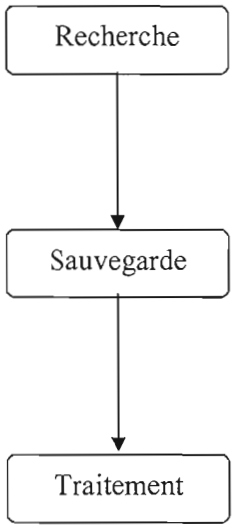

Figure 4.4 Processus de fonctionnement de l'interface de GBank UQAM

4.4.1 Recherche

Dans l'interface GBank UQAM, 4 modes de recherche sont disponibles :

- Recherche par nom du gène
- Recherche par nom de l'espèce
- Recherche par identifiant de l'espèce
- Recherche combinée par nom du gène et nom de l'espèce

4.4.1.1 Recherche par nom du gène

La recherche par nom du gène permet d'afficher les enregistrements de la base de données dont le nom du gène correspond à la valeur du champ de recherche saisie dans l'interface.

Il existe deux alternatives pour ce mode de recherche:

- Soit on spécifie le nombre d'enregistrements maximal à afficher (voir Figure 4.5);

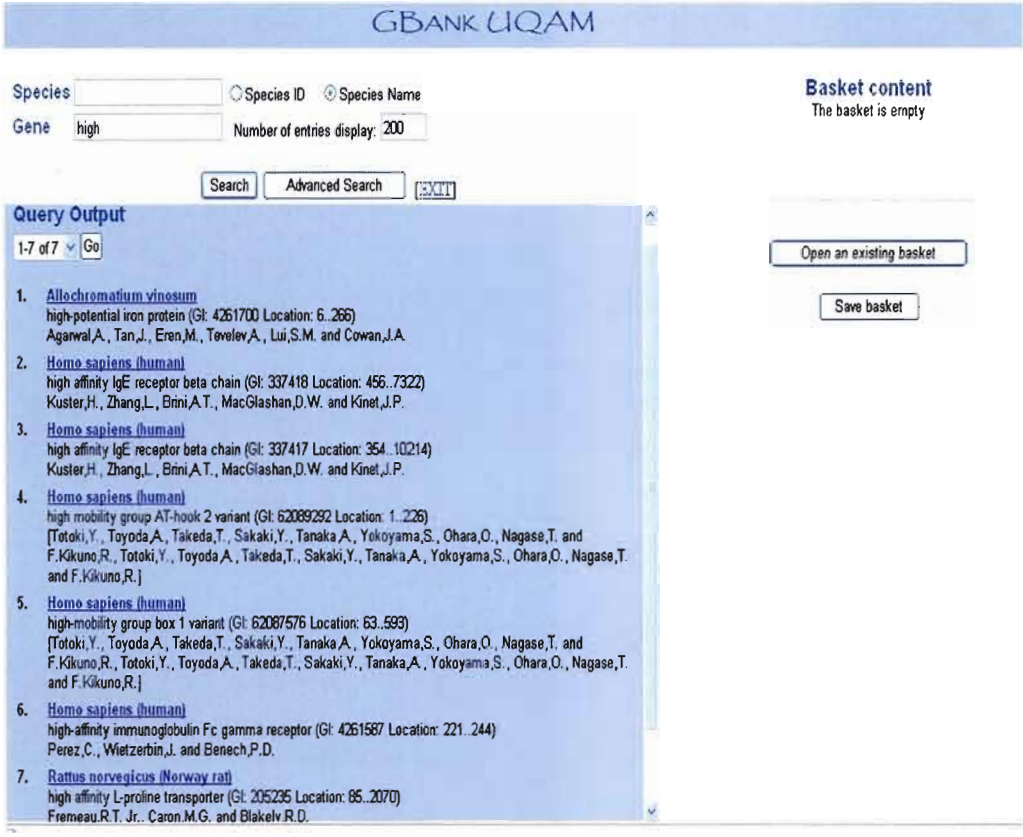

Figure 4.5 Exemple de recherche par nom du gène

- Soit on sélectionne tous les gènes qui contiennent le mot spécifié dans la case gène, en cliquant sur le bouton "*Advanced Search*", (voir Figure 4.6).

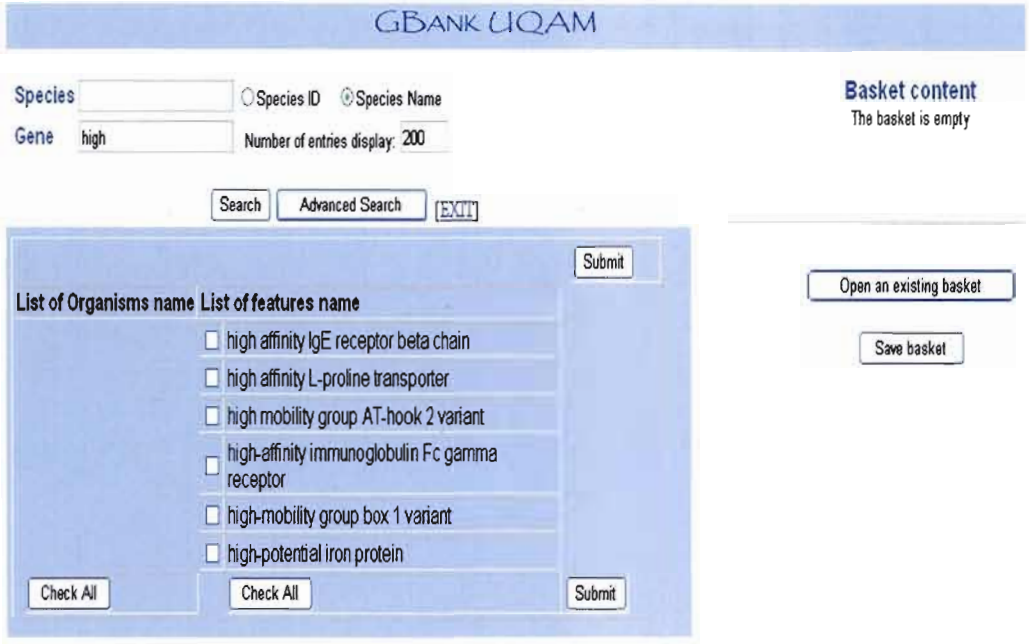

Figure 4.6 Exemple de recherche avancée par nom du gène

4.4.1.2 Recherche par nom ou identifiant de l'espèce

La recherche par nom ou identifiant de l'espèce, permet d'afficher les enregistrements de la base de données contenant le mot spécifié dans la case appropriée (Species), ou les enregistrements correspondants à l'identifiant spécifié.

Comme dans la section 4.3.1, pour ce mode de recherche nous avons deux façons de faire :

- Soit en spécifiant le nombre d'enregistrements à afficher (par défaut 200) et en cliquant sur le bouton Recherche "*Search*", (voir Figure 4.7).

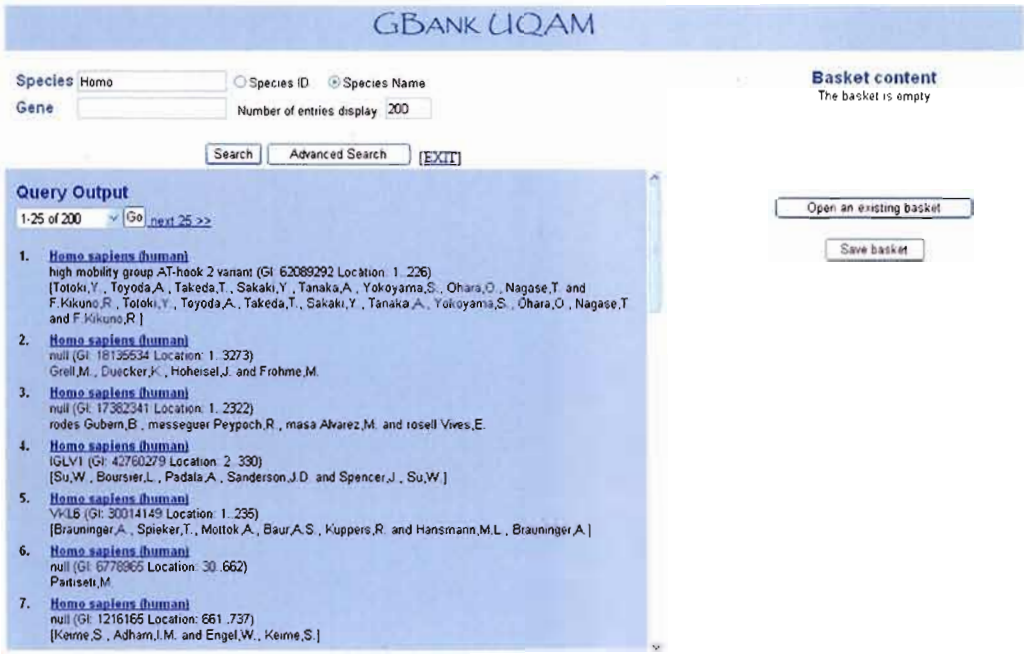

Figure 4.7 Exemple de recherche par le nom de l'espèce

- Soit en sélectionnant toutes les espèces qui contiennent le mot spécifié dans la case *Species*, en cliquant sur le bouton recherche avancée "*Advanced search*", (voir Figure 4.8).

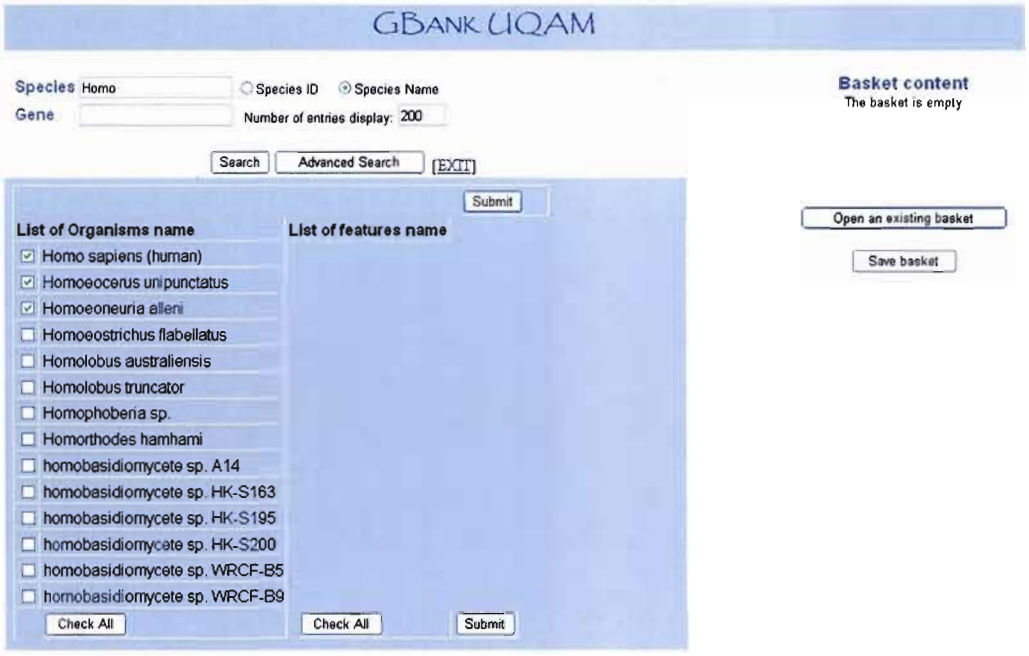

Figure 4.8 Exemple de recherche avancée par nom de l'espèce

4.4.1.3 Recherche combinée par nom du gène et nom de l'espèce

La recherche par la combinaison des gènes et des espèces passe par la recherche avancée, où l'utilisateur introduit le nom ou une partie du nom du gène et de l'espèce dans les cases correspondantes. Puis, il clique sur le bouton de recherche avancée 'Advanced Search'. Deux listes apparaissent : une liste des noms des espèces et une liste des noms des gènes.

L'utilisateur doit cocher les espèces et les gènes désirés et cliquer sur le bouton "*Submit*", (voir Figure 4.9).

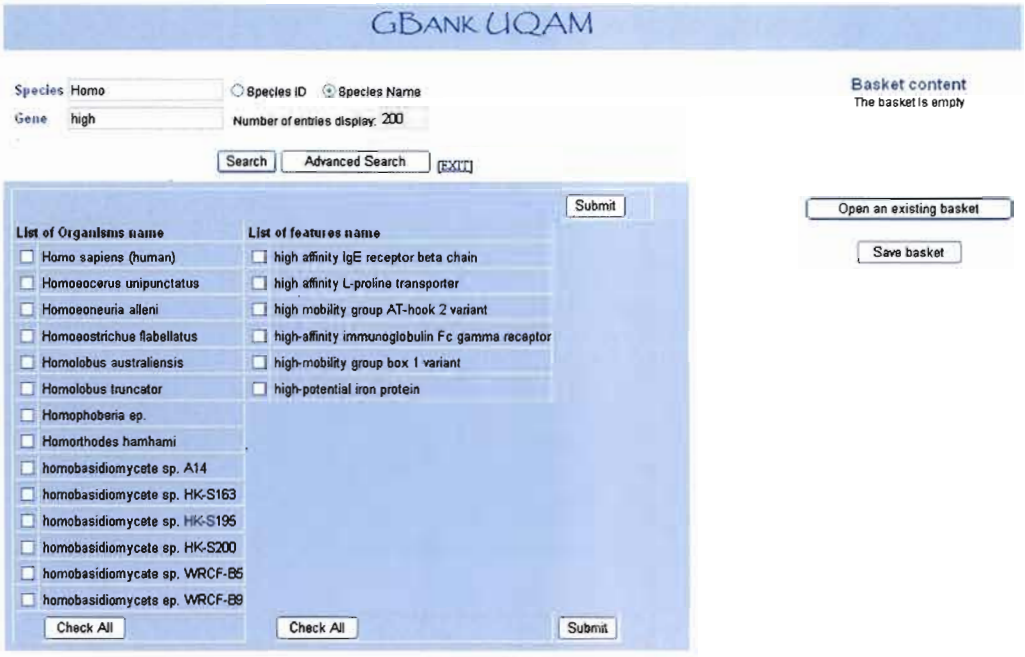

Figure 4.9 Exemple d'un choix combiné

4.4.2 Sauvegarde

Après chaque recherche, l'utilisateur peut voir les détails de l'enregistrement désiré en cliquant sur le lien du nom de l'espèce correspondant à cet enregistrement, puis une page affichant tous les détails apparaît, (voir Figure 4.10)

View details for Homo sapiens (human) / gene GI 4689142 Details

[Add to basket]

Close this window

Accession :	AF077047
Definition:	Homo sapiens galanin-related peptide mRNA, complete cds.
Authors :	[Song,H., Peng,Y., Zhou,J., Huang,Q., Dai,M., Mao,Y., Yu,Y., Xu,X., Luo,M., Hu,R. and Chen,J., Song,H.]
Sequence:	accgccctgcgaaggtctggaaaaggtggtgttcattagaaatctcaaaaccgagtcacc aagttccctctgttggagcccagtggagcctctgggagaaaagctggggtgacttttcct acaaggggcagagggactctgctagatttttgtttttcatttgtttttaattttgtaaca tggaaactctttccttaggatataccagctctcattactcagctaggaattatacctctt taaagcctgaatttaaaagtctgacagttttaaatgcttactaactgtgggagttaaatc attacgaagtgaggaatacagagtgttgtccctgattctgggtttaatctggtaagaatc tttacagaggacgaccacacgccgcttcctgtagcatgtgtcgtggttgtaattctctca tgtgcatattaagaagttgctgctcagatgtggctcttcccttgcagaggccggtgccc tcgaccgcctcctggatctccccgccgagcctcctcagaagacatcgagcggtcctgaga gcctcctgggcacgtttgctgtgtgctgtaacctgaagtcaaaccttaagataatggata atcttcggccaatttatgcagagtcagccattcctggtctctttgccttgatgttgggtt ggtatcatttaagatttttttttggtaataattttgagtggcaaaataaagaatagcaat tacttg
GI:	4689142
Feature Name:	null
Type:	
Location:	[396,716]
Min:	396
Max:	716
Product:	galanin-related peptide
Note:	null
Translation:	MCRGCNSLMCILRSCCSDVALPFAEAGALDRLLDLPAEPPQKTSSGPESLLGTFAVCCNL KSNLKIMDNLRPIYAESAIPGLFALMLGWYHLRFFFGNNFEWQNKE
OrganismID:	9606
Organism Name:	Homo sapiens (human)
Classification:	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo.

Figure 4.10 Exemple détaillé d'un enregistrement

La sauvegarde des gènes désirés, permet de mettre ces gènes dans un panier. Il suffit de cliquer sur le lien *'Add to Basket'* qui se trouve sur la page de Détails ou à coté de chaque enregistrement résultant de la recherche.

Dans le cadre droit de la page web principale, on trouve tous les gènes qui sont choisis (voir Figure 4.11).

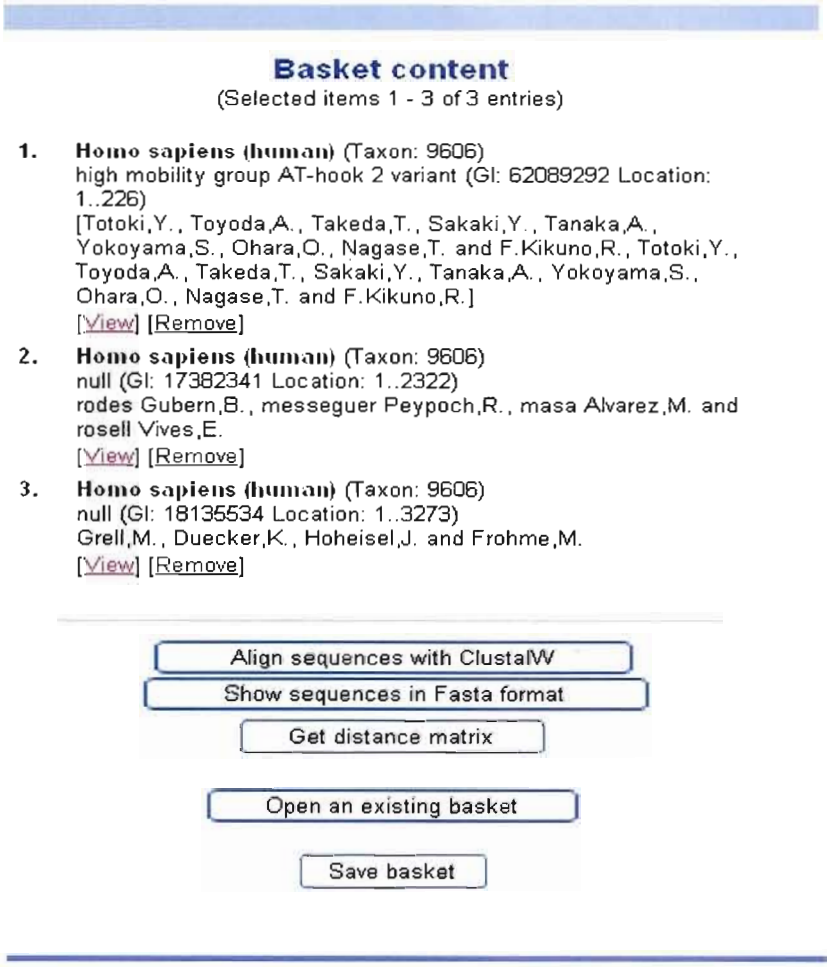

Figure 4.11 Exemple des items sélectionnés

A partir de cette liste, on peut visualiser de nouveau les détails de chaque gène en cliquant sur le lien *'View'*, comme on peut le retirer du panier en cliquant sur le lien *'Remove'*.

Afin de confirmer la sauvegarde des gènes, un simple clique sur le bouton "*Save Basket*" permettra de sauvegarder ces gènes dans un panier, et un numéro de session sera ensuite attribué à ce panier (voir Figure 4.12).

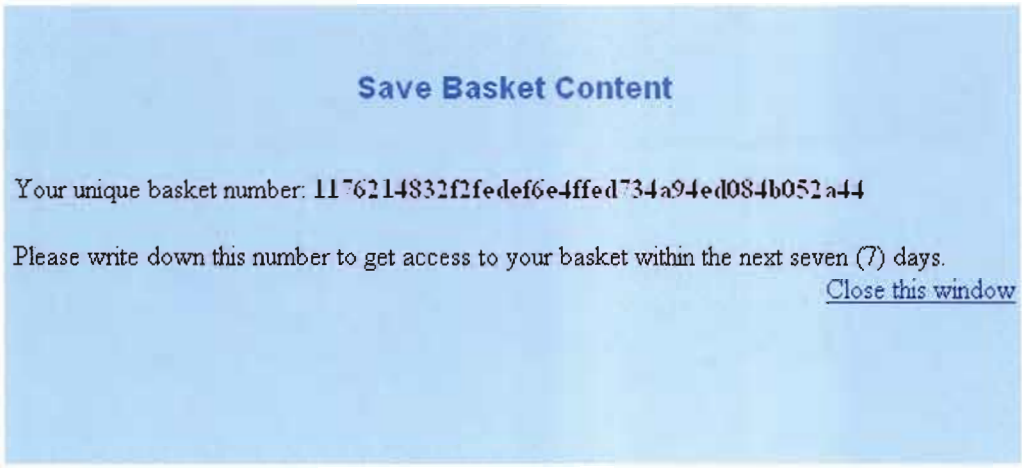

Figure 4.12 Exemple de sauvegarde d'un panier

Le panier reste sauvegardé pendant une semaine. À la fin de cette période, l'utilisateur devra sauvegarder le panier de nouveau.

Si l'utilisateur veut visualiser le panier sauvegardé, il suffit qu'il clique sur le bouton "*Open an existing basket*", et puis introduire le numéro du panier (voir Figure 4.13).

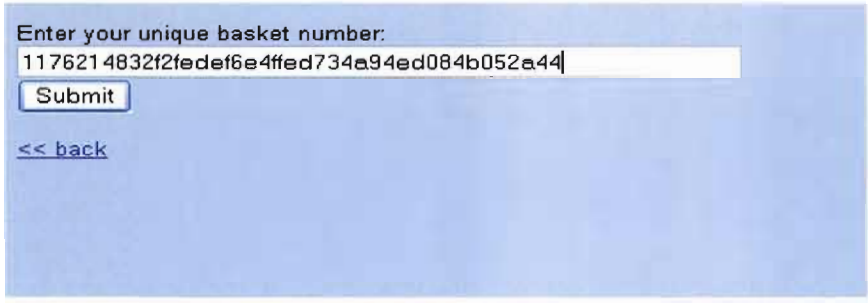

Figure 4.13 Exemple d'ouverture d'un panier existant

CONCLUSION

La base de données GenBank de NBCI est considérée comme étant la source d'information de tous les domaines de biologies.

Les outils de GenBank pour le filtrage des données ne sont pas toujours adaptés aux besoins de l'analyse phylogénétique, c'est la cause principale du développement de la nouvelle base de données bioinformatique « GBank UQAM » présentée dans ce mémoire de maîtrise. Les données de GBank UQAM serviront entre autre au développement et à la validation des algorithmes bioinformatiques pour la construction d'arbres phylogénétiques et de réseaux réticulés.

Dans le sens de l'analyse phylogénétique, la nouvelle base de données représente des améliorations par rapport à des bases de données connues (e.g., GenBank de NCBI) permettant de sélectionner et sauvegarder les données des séquences d'ADN dans le format qui permet de les traiter efficacement par les méthodes de reconstructions d'arbres et des méthodes d'alignement des séquences.

Dans ce travail, nous nous somme concentrés sur l'assurance d'une meilleure performance de la nouvelle base de données et le développement de bons processus de chargements et de mises à jour de la base. Le processus de mise à jour s'exécute chaque deux mois, touche la moitié des données de la base et prend un délai considérable (plusieurs jours). Ce processus vise à filtrer les données de mise à jour à partir de la base de données de GenBank de NCBI. Dans l'avenir, nous aurons deux choix. Soit on trouve une façon de demander aux responsables de GenBank de NCBI de nous fournir directement les donnes modifiées ou nouvelles dans des fichiers propres, soit nous devons trouver de bonnes stratégies afin d'améliorer ce processus.

APPENDICE A

DONNEES DU CHAMP FEATURE

http://www.ncbi.nlm.nih.gov/projects/collab/FT/index.html [9]

allele	Obsolete; see variation feature key
attenuator	Sequence related to transcription termination
C_region	Span of the C immunological feature
CAAT_signal	`CAAT box' in eukaryotic promoters
CDS	Sequence coding for amino acids in protein (includes stop codon)
conflict	Independent sequence determinations differ
D-loop	Displacement loop
D_segment	Span of the D immunological feature
enhancer	Cis-acting enhancer of promoter function
exon	Region that codes for part of spliced mRNA
gene	Region that defines a functional gene, possibly including upstream (promotor, enhancer, etc) and downstream control elements, and for which a name has been assigned.
GC_signal	`GC box' in eukaryotic promoters
iDNA	Intervening DNA eliminated by recombination
intron	Transcribed region excised by mRNA splicing
J_region	Span of the J immunological feature
LTR	Long terminal repeat
mat_peptide	Mature peptide coding region (does not include stop codon)
misc_binding	Miscellaneous binding site
misc_difference	Miscellaneous difference feature
misc_feature	Region of biological significance that cannot be described by any other feature
misc_recomb	Miscellaneous recombination feature
misc_RNA	Miscellaneous transcript feature not defined by other RNA keys
misc_signal	Miscellaneous signal
misc_structure	Miscellaneous DNA or RNA structure
modified_base	The indicated base is a modified nucleotide
mRNA	Messenger RNA
mutation	Obsolete: see variation feature key
N_region	Span of the N immunological feature
old_sequence	Presented sequence revises a previous version

polyA_signal	Signal for cleavage & polyadenylation
polyA_site	Site at which polyadenine is added to mRNA
precursor_RNA	Any RNA species that is not yet the mature RNA product
prim_transcript	Primary (unprocessed) transcript
primer	Primer binding region used with PCR
primer_bind	Non-covalent primer binding site
promoter	A region involved in transcription initiation
protein_bind	Non-covalent protein binding site on DNA or RNA
RBS	Ribosome binding site
rep_origin	Replication origin for duplex DNA
repeat_region	Sequence containing repeated subsequences
repeat_unit	One repeated unit of a repeat_region
rRNA	Ribosomal RNA
S_region	Span of the S immunological feature
satellite	Satellite repeated sequence
scRNA	Small cytoplasmic RNA
sig_peptide	Signal peptide coding region
snRNA	Small nuclear RNA
source	Biological source of the sequence data represented by a GenBank record. Mandatory feature, one or more per record. For organisms that have been incorporated within the NCBI taxonomy database, an associated /db_xref="taxon:NNNN" qualifier will be present (where NNNNN is the numeric identifier assigned to the organism within the NCBI taxonomy database).
stem_loop	Hair-pin loop structure in DNA or RNA
STS	Sequence Tagged Site; operationally unique sequence that identifies the combination of primer spans used in a PCR assay
TATA_signal	`TATA box' in eukaryotic promoters
terminator	Sequence causing transcription termination
transit_peptide	Transit peptide coding region
transposon	Transposable element (TN)
tRNA	Transfer RNA
unsure	Authors are unsure about the sequence in this region
V_region	Span of the V immunological feature
variation	A related population contains stable mutation
- (hyphen)	Placeholder
-10_signal	`Pribnow box' in prokaryotic promoters
-35_signal	`-35 box' in prokaryotic promoters
3'clip	3'-most region of a precursor transcript removed in processing
3'UTR	3' untranslated region (trailer)
5'clip	5'-most region of a precursor transcript removed in processing
5'UTR	5' untranslated region (leader)

APPENDICE B

PROGRAMMES JAVA

Dans le code source Java, nous avons utilisé des fonctions avancées d'Oracle 10g afin d'assurer une meilleure performance du point de vue de la rapidité du chargement. Par exemple, la fonction setStringForClob, nous a permis d'insérer des larges objets, comme dans le cas des séquences dont la taille atteint 10Mo, sans appeler d'autres fonctions du système.

```
import java.io.*;
import java.util.*;
import java.sql.*;

import org.biojava.bio.*;
import org.biojava.bio.seq.*;
import org.biojava.bio.seq.io.*;


import oracle.jdbc.*;


public class ParseGB41 {

// DECLARATION DES VARIABLES DE CHAMPS

String  ACCESSION, ORGANISM, LINEAGE, LOCATION, NOTE,
        TRANSLATION, SEQUENCE, GI, GI_LOCAL, AUTHORS, GENE,
        DEFINITION, SPECIE_ID, PRODUCT, SPECIESTYPE, ORIGIN, rep,featuresType;

 int LOCATION_MIN, LOCATION_MAX;

Manipfile errorsFeatures, errorsOrganisms, errorsSequences, outGenes,
            outSequences, outSpecies, outOthers, outReport, entree, requetes,
            requetesSequence, requetesOrganism, requetesFeature;

// DECLARATION DES VARIABLES DE CONNECTION A LA B.D.

   Connection conn = null;
   String query = null;
   String lastOrganism = "";
   OraclePreparedStatement pstmt = null;

   int compteurCommit=0;
   int compteur=0;
   String ligne;
   boolean debut = true;
   boolean hasSequence = true;

   public ParseGB41(){};


//////////////////////
  /**
   * loadData
   *
   * @param FileName String
   * @param exRep String
   * @return String
   */
```

```
public void LoadData(String FileName,String pathLog) {


   openfiles(pathLog);
   openConnection();
   BufferedReader br = null;
   entree = new Manipfile(FileName,"read");

   if(entree.open()==-1) System.out.println("Probleme"); //return Integer.toString(-1);

   while(hasSequence){

      loadSequence(pathLog);   // copy les sequences dans un fichier temporaire tmp

      try { br = new BufferedReader(new FileReader(pathLog+"tmp"));}
      catch (FileNotFoundException ex) {
         ex.printStackTrace();
         System.exit( -1);
      }

      SequenceIterator sequences = SeqIOTools.readGenBank(br);   //read the GenBank File

      processSequence(sequences);   //iterate through the sequences


    }
  }


  /**
   * loadSequence
   *
   * @param rep String
   * @return void
   */

void loadSequence(String rep){

         Manipfile sortie = new Manipfile(rep+"tmp", "write"); sortie.open();

      do {
         ligne = entree.lireL();
         if (debut) {
            do {
               if (ligne.startsWith("LOCUS"))
                     debut = false;

               else
                  ligne = entree.lireL();
```

```
            } while (debut);
          }
          if ( ligne == null) {
            outGenes.close(); outSequences.close();
            outSpecies.close();          outOthers.close();
            outReport.close();
                    requetesSequence.close();  requetesOrganism.close();
                    requetesFeature.close();    errorsFeatures.close();
                    errorsOrganisms.close();   errorsSequences.close();

             System.out.println("Number of record :"+compteur);
             hasSequence=false;
             return;

          }else{
            sortie.ecrireL(ligne);
               }
               if (ligne.length()==0) ligne = "  "; // pour faire passer la condition de while

      } while (!(ligne.charAt(0) == '/' && ligne.charAt(1) == '/'));
      sortie.close();
      compteur ++;
}

  /**
    * processSequence
    *
    * @param GI_LOCAL String
    * @param GI String
    * @return String
    */

void processSequence(SequenceIterator sequences){
      if (hasSequence==false) return;
          System.out.println("compteur is :  "+ compteur);
      while (sequences.hasNext()) {

              compteurCommit++;

ACCESSION = DEFINITION = AUTHORS = ORGANISM = SEQUENCE = null;
GENE = SPECIE_ID = PRODUCT = LOCATION = TRANSLATION = SPECIESTYPE = null;
LOCATION_MIN = LOCATION_MAX = 0;

  try {
       org.biojava.bio.seq.Sequence seq = sequences.nextSequence();
                ///// TRAITEMENT DES SEQUENCES, FEATURES
            makeSequenceRecord(seq);          // get sequence and organism data
            makeFeatureRecords(seq);          // get features data and insert them to table
                                                        FEATURES

            insertOrganismRecord();           // insert organism data to table ORGANISMS
```

```
            insertSequenceRecord();              // insert sequence data to table SEQUENCES


          } catch (BioException ex) {
            ex.printStackTrace();

          } catch (NoSuchElementException ex) {
            //request for more sequence when there isn't any
            ex.printStackTrace();
          }
              catch(SQLException e){e.printStackTrace();};
      }
}

   /**
    * getGI
    *
    * @param GI_LOCAL String
    * @param GI String
    * @return String
    */
   private String getGI(String GI_LOCAL, String GI) {
      String tmp=null;
      int compteur = 0,i=0;
      int val=0;
      boolean trouve = false;
      compteur = GI_LOCAL.indexOf("GI");

      if(compteur >= 0){
         compteur = compteur+3;
         while(compteur < GI_LOCAL.length()){
            if(GI_LOCAL.charAt(compteur)==',')
               break;
            else{
               val = val*10 + (int)GI_LOCAL.charAt(compteur)-48;
            }
            compteur++;
         }
         tmp = String.valueOf(val);
      }
      else
         return GI;

      return tmp;
   }
```

```
/**
 * getTaxon
 *
 * @param db_xref String
 * @return String
 */

private String getTaxon(String db_xref) {
   String tmp=null;
   int compteur = 0,i=0;
   int val=0;
   boolean trouve = false;
   compteur = db_xref.indexOf("taxon");

   if(compteur >= 0){
      compteur = compteur+6;
      while(compteur < db_xref.length()){
         if(db_xref.charAt(compteur)==',')
            break;
         else{
            val = val*10 + (int)db_xref.charAt(compteur)-48;
         }
         compteur++;
      }
      tmp = String.valueOf(val);
   }
   else
      return null;

   return tmp;
}



/**
 * openFiles
 *
 * @param rep String
 * @return void
 */

void makeFeatureRecords(org.biojava.bio.seq.Sequence seq)throws SQLException{

   for (Iterator fi = seq.features(); fi.hasNext(); ) {

      GENE = null;
      org.biojava.bio.seq.Feature f = (org.biojava.bio.seq.Feature) fi.next();
      String featuresType = f.getType();

      if (featuresType.equalsIgnoreCase("source")) {
         try {SPECIE_ID = f.getAnnotation().getProperty("db_xref").toString();
```

```
            SPECIE_ID = getTaxon(SPECIE_ID);
          } catch (NoSuchElementException e) { SPECIE_ID = null;}
             if(SPECIE_ID == null) SPECIE_ID = "0";
           }else if (featuresType.equalsIgnoreCase("CDS") ||
             featuresType.equalsIgnoreCase("mRNA") ||
             featuresType.equalsIgnoreCase("rRNA") ||
             featuresType.equalsIgnoreCase("tRNA")) {

                    LOCATION_MIN = f.getLocation().getMin();
              LOCATION_MAX = f.getLocation().getMax();

try {PRODUCT = getAnnotation().getProperty("product").toString().replace('\'','#');}
          catch (NoSuchElementException e) {PRODUCT = null;}

try {GENE = f.getAnnotation().getProperty("gene").toString().replace('\'','#');}
          catch (NoSuchElementException e) {GENE = null;}

try {GI_LOCAL = getGI(f.getAnnotation().getProperty("db_xref").toString(), GI);}
                    catch (NoSuchElementException e) { GI_LOCAL = GI;}

try {NOTE = f.getAnnotation().getProperty("note").toString().replace('\'','#');}
          catch (NoSuchElementException e) {NOTE = null;}

try {TRANSLATION = f.getAnnotation().getProperty("translation").toString();}
          catch (NoSuchElementException e) {TRANSLATION = null;}

          if (LOCATION_MIN== 2147483647 || LOCATION_MAX == 2147483647)
                    // si on ne peut avoit les LOCATIONS (cas gbpri26.seq)
                      ORIGIN = null;
                    else
          ORIGIN = SEQUENCE.substring(LOCATION_MIN,LOCATION_MAX);

            outGenes.ecrireL("----");
            outGenes.ecrireL("GENE       :" + GENE);
            outGenes.ecrireL("GI         :" + GI_LOCAL);
            outGenes.ecrireL("ACCESSION   :" + ACCESSION);
            outGenes.ecrireL("INTERVAL(min):" + LOCATION_MIN);
            outGenes.ecrireL("INTERVAL(max):" + LOCATION_MAX);
            outGenes.ecrireL("SEQUENCE TYPE:" + featuresType);
            outGenes.ecrireL("PRODUCT     :" + PRODUCT);
            outGenes.ecrireL("NOTE       :" + NOTE);
            outGenes.ecrireL("TRANSLATION  :" + TRANSLATION);

            try{

query = "insert into Features
             values('"+GI_LOCAL+"','"+GENE+"','"+featuresType+"',"+
                      LOCATION_MIN+","+LOCATION_MAX+",'"+PRODUCT
                        +"','"+ACCESSION+"','"+NOTE+"',?,?)";
                      pstmt = (OraclePreparedStatement)conn.prepareStatement(query);
                      pstmt.setStringForClob(1, TRANSLATION);
```

```
                    pstmt.setStringForClob(2, ORIGIN);

                pstmt.executeUpdate();


             }catch(SQLException e) {errorsFeatures.ecrireL(query);
                                        errorsFeatures.ecrireL(e.getMessage());
                                        }
                    finally {
            // Close the Statement and the connection objects.
              if (pstmt!=null) pstmt.close();
             }
                    }else if (featuresType.compareToIgnoreCase("gene") != 0) {
                    outOthers.ecrireL("ACCESSION   :" + ACCESSION);
               outOthers.ecrireL("=>SEQUENCE TYPE:" + featuresType);
            }
            }
}
  /**

   * openFiles
   * @param rep String
   * @return void
   */
void openfiles(String rep){
        boolean success = (new File(rep)).mkdir();
        errorsFeatures = new Manipfile(rep+"errorsFeatures.txt","write");
        errorsFeatures.open();
        errorsOrganisms = new Manipfile(rep+"errorsOrganisms.txt","write");
        errorsOrganisms.open();
        errorsSequences = new Manipfile(rep+"errorsSequences.txt","write");
        errorsSequences.open();
        requetes = new Manipfile(rep+"requetes.txt","write");  requetes.open();
        outGenes = new Manipfile(rep+"outGenes.txt","write"); outGenes.open();
        outSequences = new Manipfile(rep+"outSequences.txt","write"); outSequences.open();
        utSpecies = new Manipfile(rep+"outSpecies.txt","write");
        outSpecies.open();
        outOthers = new Manipfile(rep+"outOthers.txt","write");
        outOthers.open();
        outReport = new Manipfile(rep+"outReport.txt","write");
        outReport.open();
        requetesFeature = new Manipfile(rep+"requetesFeature.txt","write");  requetesFeature.open();
        requetesOrganism = new Manipfile(rep+"requetesOrganism.txt","write");
        requetesOrganism.open();
        requetesSequence = new Manipfile(rep+"requetesSequence.txt","write");
        requetesSequence.open();

        }
```

```
  /**
   * openConnection
   * @return void
   */
        void openConnection(){
            try{

                DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
                conn = DriverManager.getConnection ("jdbc:oracle:thin:@127.0.0.1:1521:GB",
"BIO","bioinfo");
            }
            catch(SQLException e){e.printStackTrace(); System.exit(0);}
        }
        ////////////////////////////////////////
  /**
   * makeSequenceRecord
   * @param seq Sequence
   * @return void
   */
void makeSequenceRecord(org.biojava.bio.seq.Sequence seq){
ACCESSION =
new StringTokenizer(seq.getAnnotation().getProperty("ACCESSION").toString(), " ").nextToken();

 DEFINITION = seq.getAnnotation().getProperty("DEFINITION").toString().replace('\'','#');

AUTHORS = seq.getAnnotation().getProperty("AUTHORS").toString().replace('\'','#');

if (AUTHORS.length() > 1000) AUTHORS = AUTHORS.substring(0,998);

ORGANISM = seq.getAnnotation().getProperty("SOURCE").toString().replace('\'','#');

 SEQUENCE = seq.seqString();

 LINEAGE = seq.getAnnotation().getProperty("ORGANISM").toString().replace('\'','#');



if (ORGANISM.length() < LINEAGE.length())
  LINEAGE = LINEAGE.substring(ORGANISM.length() + 1, LINEAGE.length());

GI = seq.getAnnotation().getProperty("GI").toString();

      }
//////////////////////////
        void insertOrganismRecord()throws SQLException{
//      OraclePreparedStatement pstmt = null;
          outSpecies.ecrireL("ORGANISM_ID  :" + SPECIE_ID);
          outSpecies.ecrireL("ORGANISM     :" + ORGANISM);
          outSpecies.ecrireL("LINEAGE      :" + LINEAGE);
          pstmt = null;
        if(SPECIE_ID != null){
```

```
        if(SPECIE_ID.compareTo(lastOrganism)!=0){
            lastOrganism = SPECIE_ID;
          try{

              timeStart = (new java.util.Date()).getTime();
              query ="insert into Organisms(OrganismID,OrganismName,Classification)
values("+SPECIE_ID+",'"+ORGANISM+"','"+LINEAGE+"')";
                        pstmt = (OraclePreparedStatement)conn.prepareStatement(query);

                  timeStart = (new java.util.Date()).getTime();
                pstmt.executeUpdate();
                        timeEnd = (new java.util.Date()).getTime();

            requetesOrganism.ecrireL("time is:  "+(timeEnd-timeStart));

          }catch(SQLException e){
                    //errorsOrganisms.ecrireL(query);
                    //errorsOrganisms.ecrireL(e.getMessage());
                  }
                  finally {
                // Close the Statement and the connection objects.
                if (pstmt!=null) pstmt.close();

                  }
        }

          }else{errorsOrganisms.ecrireL(ACCESSION +"=> PAS DE NUMERO D'ORGANISM");}
        }
////////////////////////////
        void insertSequenceRecord()throws SQLException{
          //if(SPECIE_ID == null) SPECIE_ID = "0";
//OraclePreparedStatement pstmt = null;

  try {
    // Create SQL query to insert data into the CLOB column in the database.
    query ="insert into Sequences
values('"+ACCESSION+"','"+GI+"','"+DEFINITION+"','"+AUTHORS+"',"+SPECIE_ID+",?)";

    // Create the OraclePreparedStatement object
    //pstmt = conn.prepareStatement(query);
    pstmt = (OraclePreparedStatement)conn.prepareStatement(query);
    pstmt.setStringForClob(1,SEQUENCE);

    // Execute the PreparedStatement
  timeStart = (new java.util.Date()).getTime();
    pstmt.executeUpdate();
  timeEnd = (new java.util.Date()).getTime();

  requetesSequence.ecrireL("time is:  "+(timeEnd-timeStart));

  } catch (SQLException sqlex) {
```

```
    errorsSequences.ecrireL(query); errorsSequences.ecrireL(sqlex.getMessage());

  } catch (Exception ex) {
    System.out.println("Exception while connecting and inserting into the" +
                " database table: " + ex.toString());
  } finally {

    if (pstmt!=null) pstmt.close();

  }
 }
//////////////////////////////
}
```

APPENDICE C

PAGES WEB DE L'INTERFACE GBANK UQAM (PHP)

Dans cet appendice, nous présentons les principales pages WEB de l'interface GBank UQAM que nous avons développées.

1. Se connecter à la base de données
2. Accueillir l'usager et nettoyer les tables d'utilisateur
3. Saisir les données de la recherche et choisir les critères appropriés
4. Récupérer les résultats de la recherche
5. Ajouter des données dans un panier d'utilisateur
6. Visualiser les détails d'un enregistrement
7. Retirer un élément du panier
8. Effectuer le traitement à partir d'ID du panier sauvegardé
9. Transfert les tuples contenus dans le panier

1. Se connecter à la base de données

Cette page assure la connexion à la BD

```
<?php
session_start();

if($_SESSION['DEBUG'] == "yes")
$conn = OCILogon("bioinfo", "b9159f15",'//trex.labunix.uqam.ca:1521/GBANK');
else
$conn = @OCILogon("bioinfo", "b9159f15",'//trex.labunix.uqam.ca:1521/GBANK');
?>
```

2. Accueillir l'usager et nettoyer les tables d'utilisateur

Cette page sert d'acceuil à l'usager tout en lui attribuant un numéro de session et le temps de départ.

```
<?php
session_start() ;
?>
<html><head>
<script language="Javascript">
function open_popup(page)
{
   top.window_handle =
open(page,'remote','height=550,width=700,resizable=yes,location=no,toolbar=no,menubar=no,scrollba
rs=yes');
   top.window_handle.focus();
   if (!top.window_handle.opener) top.window_handle.opener = self;
   return false;
}
</script>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<LINK HREF="style.css" REL= "stylesheet" TYPE="text/css">
<title>GBank UQAM</title></head>
<BODY bgcolor="#ACBFD5" topmargin="3">
<center><span style="font-variant: small-caps"><strong>
<font face="Papyrus" size="6" color="#0C2B74">GBank UQAM </font></strong></span></center>
<br><br><br>
<?php
$geneList = null;
$specieList = null;
include "connect_bd.php";
if($conn==""){
        echo "<center><h2>The database is temporarily unavailable due to maintenance, please try
later...</h2>";
        echo "<a href='http://www.trex.uqam.ca'>back to trex online</a></center><br><br>";
```

```
}
else{
        $Time = time();
        $delai = $Time-(6*3600);
        $_SESSION['no_sess'] = $no_sess =  session_id();
        $delete = 'delete from savedbasket where echeance < '.$Time;
        $stid = OCIParse($conn, $delete);
        OCIExecute($stid, OCI_DEFAULT);
        $num_columns = ociParse($conn,"select count(*) from query where no_sess =
'".$no_sess."'");
        $select = 'select no_sess from usagers'; // where time < '.$delai;
        $result = OCIParse($conn, $select);
        OCIExecute($result, OCI_DEFAULT);
  while (OCIFetch($result)) {
                for ($i = 1; $i < $num_columns+1; $i++) {
      $delete = "delete from usagers where no_sess = '".$i ."'";
      $stid = OCIParse($conn, $delete);
      OCIExecute($stid, OCI_DEFAULT);
      $delete = "delete from basket where no_sess = '".$i ."'";
      $stid = OCIParse($conn, $delete);
      OCIExecute($stid, OCI_DEFAULT);
      $delete = "delete from query where no_sess = '".$i ."'";
      $stid = OCIParse($conn, $delete);
      OCIExecute($stid, OCI_DEFAULT);
    }
  }
        echo "<form action=\"basketgenes.php?no_sess=$no_sess\" name=\"connexion\"
method=\"post\" >\n";
        OCILogoff($conn);
}
?>
<div align="center">
<A href="javaScript:;" onClick="open_popup('manual.php')">User manual</A>
<br><br>
<input type="submit" value="Start session"></div>
</form>
<br><br><br><br>
</body>
</html>
```

3. Saisir les données de la recherche et choisir les critères appropriés

Cette page est la page principale de la recherche. L'utilisateur y entre ses critères de recherche et soumet sa requête en cliquant sur "Search" ou "Advanced Search" qui envoient à la page "list.php" traita la requête.

```
<?php
$no_sess = $_GET['no_sess'];
?>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
function content()
{  if (document.getlist.gene.value == "" && document.getlist.specie.value == "" )
   { alert("You must enter a gene or species name");
    return false;
   }   return true;
}
function content2()
{
   if (!document.process.seq.checked && !document.process.line.checked)
   {  alert("You must at least check one display option");
   return false;
   }   return true;
}
function open_popup(page)
{   top.window_handle = open(page,'popupWindowName','height=700, width=620, scrollbars=yes');
   top.window_handle.focus();
   if (!top.window_handle.opener) top.window_handle.opener = self;
   return false;
}
function CloseWindow()
{     window.close();  }

</SCRIPT>
<HTML>
<frameset rows="50,*" border =0>
 <frame src="title.php?no_sess=<? echo $no_sess; ?>" name="title" SCROLLING="no"
NORESIZE>
 <frameset cols="60%,*" border =0>
  <frameset rows="120,*" border =0>
   <frame src="submitbasket.php?no_sess=<? echo $no_sess; ?>" name="submitbasket"
SCROLLING="no" NORESIZE>
    <frame src="resultbasket.php?no_sess=<? echo $no_sess; ?>" name="resultbasket"
SCROLLING="auto" NORESIZE>
  </frameset>
  <frame src="savebasket.php?no_sess=<? echo $no_sess; ?>" name="savebasket"
SCROLLING="auto" NORESIZE>
</frameset>
</html>
```

4. Récupérer les résultats de la recherche

Cette page envoie la requête de recherche dans la base de données entière pour ensuite insérer les résultats dans la table "Query".

```
<?
session_start();
include "connect_bd.php";

isset($HTTP_GET_VARS) or
die ("Le tableau n'existe pas");

$gene = $HTTP_POST_VARS['gene'];
$specie = $HTTP_POST_VARS['specie'];
$mode = $HTTP_POST_VARS['Mode'];
$number_rows = $HTTP_POST_VARS['number_rows']+1;
$no_sess = $_REQUEST['no_sess'];

$varSpecie = null;
$varGene = null;

$drop = "delete from query where no_sess = '".$no_sess."'";
no_session=$no_sess";

$stid = OCIParse($conn, $drop);
OCIExecute($stid, OCI_DEFAULT);
OCICommit($conn);

if($specie != ''){
    if($Mode != 'ID') $varSpecie = " and O.Organismname like '".$specie."%'";
    else $varSpecie = $specie;
}

if ($gene != ''){
   if($specie != '') $varGene = " featurename like '".$gene."%' and ";
   else $varGene = " F.featurename like '".$gene."%' and ";
}

$sql = "BEGIN Load_Result('".$varGene."'
,'".$varSpecie."','".$mode."','".$no_sess."','".$number_rows."'); END;";

$result = OCIParse($conn, $sql);
OCIExecute($result, OCI_DEFAULT);

OCICommit($conn);
OCILogoff($conn);
echo "<html><head><meta http-equiv=\"refresh\"
content=\"1;URL=resultbasket.php?no_sess=$no_sess\"></head></html>\n";
?>
```

5. Ajouter des données dans un panier d'utilisateur

Cette page permet d'ajouter un élément au panier suite à la demande de l'usager ayant cliqué sur "Add to Basket".

```
<?
include "connect_bd.php";                    //connexion a la BD
$NoSeq = $_REQUEST['NoSeq'];
$no_sess = $_REQUEST['no_sess'];
$query = "BEGIN insert_basket(".$NoSeq.",'".$no_sess."'); END;";
$Accession = $_GET['Gi'];
$LocationMin = $_GET['LocationMin'];

$result = OCIParse($conn, $query);        //lecture du tuple
OCIExecute($result, OCI_DEFAULT);  //lecture des details de l enregistrement

OCICommit($conn);
OCILogoff($conn);                  //deconnexion
echo "<meta http-equiv='refresh' content='0;URL=savebasket.php?no_sess=".$no_sess."'>";
?>
```

6. Visualiser les détails d'un enregistrement

Cette page permet de visualiser les détails d'un enregistrement obtenu d'une recherche dans la base de données.

```
<script language="Javascript">     //fermeture de la page sur click
function CloseWindow()
{
   window.close();
}
</script>
<HTML><HEAD><TITLE>Details</TITLE>
<META http-equiv=Content-Language content=fr-ca>
<META http-equiv=Content-Type content="text/html; charset=windows-1252">
<META content="MSHTML 6.00.2900.2523" name=GENERATOR></HEAD>
<BODY bgcolor="#ACBFD5" topmargin="3">
<P style="MARGIN-TOP: 7px; MARGIN-BOTTOM: 0px" align=center>
<FONT face="Arial" size=4 color="#0C2B74"><B>

<?
function seqOutput($Seq)
{
      if($Seq==null)
   {
      $Seq1= " ";
   }
   else
```

```
   {
      $Seq1= $Seq->load();
   }

   echo "<p><font face ='courier' size =2>\n";
   for($i=1;$i<=strlen($Seq1);$i++)
   {
    echo $Seq1[$i-1];
    if ($i % 60 ==0)
      echo "<br>";
   }
   echo "</font></p>\n";
}
include "connect_bd.php";      //connexion

$Accession = $_REQUEST['Accession'];
$NoSeq = $_REQUEST['NoSeq'];
$no_sess = $_REQUEST['no_sess'];
$Gi = $_REQUEST['Gi'];
$LocationMin = $_REQUEST['LocationMin'];

$sql = "BEGIN
P_UpdateQuery('".$Accession."','".$NoSeq."','".$no_sess."','".$Gi."','".$LocationMin."'); END;";

$result = OCIParse($conn, $sql);
OCIExecute($result, OCI_DEFAULT);
OCICommit($conn);

$query = "select * from query where NoSeq = ".$NoSeq." and no_sess = '".$no_sess."'";

$result = OCIParse($conn, $query);
OCIExecute($result, OCI_DEFAULT);   //lecture des details de l'enregistrement

if(OCIFetch($result)) {                                    //affichage des details

   echo "View details for ".OCIResult($result,"ORGANISMNAME")." / gene GI
".OCIResult($result,"GI")." Details</B></FONT></P>\n";

   echo "<TABLE border=1>";
   echo "<TR><TD>Accession : </TD><TD>";
   echo OCIResult($result,"ACCESSION")."</TD></TR>";
   echo "<TR><TD>Definition: </TD><TD>";
   echo OCIResult($result,"DEFINITION")."</TD></TR>";
   echo "<TR><TD>Authors : </TD><TD>";
   echo OCIResult($result,"AUTHORS")."</TD></TR>";
   echo "<TR><TD>Sequence: </TD><TD>";
   seqOutput(OCIResult($result,"SEQUENCE"));
   echo "</TD></TR>";
   echo "<TR><TD>GI: </TD><TD>";
   echo OCIResult($result,"GI")."</TD></TR>";
   echo "<TR><TD>Feature Name: </TD><TD>";
```

```
   echo OCIResult($result,"FEATURENAME")."</TD></TR>";
   echo "<TR><TD>Type: </TD><TD>";
   echo OCIResult($result,"SEQUENCETYPE")."</TD></TR>";

   echo "<TR><TD>Location: </TD><TD>";
   echo OCIResult($result,"LOCATION")."</TD></TR>";
   echo "<TR><TD>Min: </TD><TD>";
   echo OCIResult($result,"LOCATIONMIN")."</TD></TR>";
   echo "<TR><TD>Max: </TD><TD>";
   echo OCIResult($result,"LOCATIONMAX")."</TD></TR>";
   echo "<TR><TD>Product: </TD><TD>";
   echo OCIResult($result,"PRODUCT")."</TD></TR>";
   echo "<TR><TD>Note: </TD><TD>";
   echo OCIResult($result,"NOTE")."</TD></TR>";
   echo "<TR><TD>Translation: </TD><TD>";
   seqOutput(OCIResult($result,"TRANSLATION"));
   echo "</TD></TR>";
   echo "<TR><TD>OrganismID: </TD><TD>";
   echo OCIResult($result,"ORGANISMID")."</TD></TR>";
   echo "<TR><TD>Organism Name: </TD><TD>";
   echo OCIResult($result,"ORGANISMNAME")."</TD></TR>";
   echo "<TR><TD>Classification: </TD><TD>";
   echo OCIResult($result,"CLASSIFICATION")."</TD></TR>";
   echo "[<A
href=\"panier.php?Accession=$Accession&Gi=$Gi&LocationMin=$LocationMin&NoSeq=$NoSeq&
no_sess=$no_sess\" target=savebasket>Add to basket</A>]</FONT> \n";
}
OCILogoff($conn);              //deconnexion
?>

<a href="#" onClick="CloseWindow()"><div align="right">Close this window</div></a>
```

7. Retirer un élément du panier

Cette page permet de retirer un élément du panier suite à la demande de l'usager ayant cliqué sur "Remove"

```
<?
include "connect_bd.php";        //connexion
$acces = $_GET['acces'];         //effacer du basket
$gi = $_GET['gi'];
$oid = $_GET['oid'];
$no_sess = $_GET['no_sess'];
$locmin = $_GET['locmin'];
$locmax = $_GET['locmax'];
$min = $_GET['min'];
$Pmin = $_GET['Pmin'];
$query = "delete from basket where Accession= '".$acces."'and GI = '".$gi."' and LocationMin
=".$locmin." and LocationMax =".$locmax. " and OrganismID= ".$oid." and no_sess = '".$no_sess."'";
```

```
$result = OCIParse($conn, $query);
OCIExecute($result, OCI_DEFAULT);   //lecture des details de l enregistrement
$result = OCIParse($conn, "commit");
OCIExecute($result, OCI_DEFAULT);
OCILogoff($conn);                   //deconnexion
echo "<html><head><meta http-equiv=\"refresh\"
content=\"1;URL=savebasket.php?min=$min&Pmin=$Pmin&no_sess=$no_sess\"></head></html>\n"
;
?>
```

8. Effectuer le traitement à partir d'ID du panier sauvegardé

Cette page effectue le traitement à partir d'ID du panier sauvegardé

```
<?php

include "connect_bd.php";                   //connexion
$basket = $_POST['saisie'];                 //lecture du no de basket sauvé
$no_sess = $_GET['no_sess'];
$min = $_POST['min'];
$Pmin = $_POST['Pmin'];

$count = ociParse($conn,"select count(*) from savedbasket where nobasket = '".$basket."'");
//compter les resultats
OCIExecute ($count);
OCIFetchInto($count, $numrows) ;
$numrows = $numrows[0];

if($numrows==0)                     //si 0 result, afficher message
{
   OCILogoff($conn);      //deconnexion
   echo "The basket number is not valid or the delay expired";
   Echo "<BR>    <A  \n";
   Echo "href=\"savebasket.php?no_sess=$no_sess&min=$min&Pmin=$Pmin\"><< back</A>";
}

else                                //sinon effacer basket courant et inserer new contenu
{
$sql = "BEGIN RESTOREBASKET('".$basket."' ,'".$no_sess."'); END;";
$result = OCIParse($conn, $sql);
OCIExecute($result, OCI_DEFAULT);
OCICommit($conn);
OCILogoff($conn);      //deconnexion
  echo "<html><head><meta http-equiv=\"refresh\"
content=\"1;URL=savebasket.php?no_sess=$no_sess\"></head></html>\n";
}
?>
```

9. Transfert les tuples contenus dans le panier

Cette page transfert les tuples contenus dans le panier dans une table prévue pour la sauvegarde du panier.

```
<HTML><HEAD><TITLE>Save basket</TITLE>
<META http-equiv=Content-Language content=fr-ca>
<META http-equiv=Content-Type content="text/html; charset=windows-1252">
<META content="MSHTML 6.00.2900.2523" name=GENERATOR></HEAD>
<BODY bgcolor="#ACBFD5" topmargin="3">
<center><span style="font-variant: small-caps"><strong>
<font face="Papyrus" size="6" color="#0C2B74">Genebase Uqam </font></strong></span></center>
<br><br>
 <P style="MARGIN-TOP: 7px; MARGIN-BOTTOM: 0px" align=center>
<FONT face="Arial" size=4 color="#0C2B74"><B>Save Basket Content</B></FONT></P>
<BR><BR>
<?php
include "connect_bd.php";            //connexion
$Time = time();                  //creation du code de basket
$codeBasket = $Time.$no_sess;            //avec id session et time
$echeance = $Time+(168*3600);             //echeance 7 jours
$no_sess = $_GET['no_sess'];
$min    = $_GET['min'];
$Pmin   = $_GET['Pmin'];
$count = ociParse($conn,"select count(*) from basket where no_sess = '".$no_sess."'");
OCIExecute ($count);
OCIFetchInto($count, $numrows) ;
$numrows = $numrows[0];
if($numrows==0)   //affichage de l entê "no entries"
{  Echo "<P style=\"MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px\" align=center><FONT \n";
   Echo "face=\"Courier New\" size=2>\n The basket is empty! \n";
   Echo "</FONT></P> \n";
} else                    //sinon insertion dans basket sauvegardé
{ $sql = "BEGIN SAVEBASKET('".$codeBasket."' ,'".$echeance."','".$no_sess."'); END;";
$result = OCIParse($conn, $sql);
OCIExecute($result, OCI_DEFAULT);
OCICommit($conn);
OCILogoff($conn);      //deconnexion
echo "Your unique basket number: <STRONG>";    //infos a l usager
echo $codeBasket;
echo "</STRONG><br><br>";
echo "Please write down this number to get access to your basket within the next seven (7)
days.<BR>";
}
?>
</BODY></HTML>
<script language="Javascript">            //fermeture de la page sur click
function CloseWindow()
{   window.close(); }
</script>
<a href="#" onClick="CloseWindow()"><div align="right">Close this window</div></a>
```

APPENDICE D

PROCEDURES ET FONCTIONS (PL/SQL)

Dans cet appendice, nous présentons les principales fonctions et procédures (PL/SQL) que nous avons développées dans le cadre du projet de la base de données GBank UQAM.

1. Extraire et insérer les résultats de la recherche
2. Insérer l'enregistrement choisi
3. Sauvegarder l'enregistrement désiré
4. Restaurer l'enregistrement désiré

1. Extraire et insérer les résultats de la recherche

Cette fonction permet l'extraire des données recherchées de la base de données, ensuite, elle est insert dans la table "Query".

```
  LOAD_RESULT  (vGene varchar2, vSpecie varchar2,  vMode varchar2, vNo_sess varchar2,
                    number_rows varchar2) is

i number;

query varchar2(1000);
query1 varchar2(1000);


TYPE TYP_REF_CUR IS REF CURSOR ;

cur  TYP_REF_CUR ;

  -- variables d'accueil
Accession SEQUENCES.accession%Type ; Authors  SEQUENCES.Authors%Type ;

Gi FEATURES.gi%Type ; FeatureName FEATURES.featurename%Type ; LocationMin
FEATURES.LocationMin%Type ;
LocationMax FEATURES.LocationMax%Type ;

OrganismId ORGANISMS.organismId%Type ;  OrganismName
ORGANISMS.organismName%Type ;

begin


if(vMode = 'name') then

  if (vSpecie is not null and vGene is not null) then

     query := 'select S.Accession,Authors,F.GI, FeatureName, LocationMin,LocationMax,
O.OrganismID, OrganismName
          from  Features F, Sequences S, Organisms O
          where F.accession=S.accession  and  S.organismid=O.organismid ' || vSpecie || ' and ' ||
vGene || ' and rownum <' || number_rows ;


  end if;

  if (vSpecie is not null and vGene is null) then

     query := 'select S.Accession,Authors,F.GI, FeatureName, LocationMin,LocationMax,
O.OrganismID, OrganismName
          from  Features F, Sequences S, Organisms O
```

```
            where  F.accession=S.accession  and  S.organismid=O.organismid' || vSpecie || ' and
rownum <' || number_rows ;

  end if;

  if (vSpecie is null and vGene is not null) then

     query := 'select S.Accession,Authors,F.GI, FeatureName, LocationMin,LocationMax,
O.OrganismID, OrganismName
            from  Features F, Sequences S, Organisms O
            where  F.accession=S.accession  and  S.organismid=O.organismid and ' || vGene || ' rownum
<' || number_rows ;

  End if;

else   -- vMode = 'Id'

  if (vSpecie is not null and vGene is not null) then

     query := 'select S.Accession,Authors,F.GI, FeatureName, LocationMin,LocationMax,
O.OrganismID, OrganismName
            from  Features F, Sequences S, Organisms O
            where  F.accession=S.accession  and  S.organismid=O.organismid' || vSpecie || ' and ' ||
vGene || ' and rownum <' || number_rows ;

  end if;

  if (vSpecie is not null and vGene is null) then

     query := 'select S.Accession,Authors,F.GI, FeatureName, LocationMin,LocationMax,
O.OrganismID, OrganismName
            from  Features F, Sequences S, Organisms O
            where  F.accession=S.accession  and  S.organismid=O.organismid' || vSpecie || ' and
rownum <' || number_rows ;

  end if;

End if;
--//////////////////////////////////////////
if (vSpecie is not null and vGene is not null) then
 OUTPUT_LINE(query) ;
 ----------------------
 Open cur For query;
     i:=0;
 Loop
  i:=i+1;
  Fetch cur Into  Accession, Authors, GI, FeatureName, LocationMin, LocationMax, OrganismID,
OrganismName;
  Exit When cur%NOTFOUND ;
```

```
   INSERT INTO query (no_sess, Noseq, Accession, Authors, GI, FeatureName, LocationMin,
LocationMax, OrganismID, OrganismName)
   VALUES   (vNo_sess,to_char(i),Accession, Authors, GI, FeatureName, LocationMin,
LocationMax, OrganismID, OrganismName);
  End loop;
  Close cur;
end if;

if (vSpecie is not null and vGene is null) then

  Open cur For query;
      i:=0;
  Loop
   i:=i+1;
   Fetch cur Into  Accession, Authors, GI, FeatureName, LocationMin, LocationMax, OrganismID,
OrganismName;
   Exit When cur%NOTFOUND ;

   INSERT INTO query (no_sess, Noseq, Accession, Authors, GI, FeatureName, LocationMin,
LocationMax, OrganismID, OrganismName)
   VALUES   (vNo_sess,to_char(i),Accession, Authors, GI, FeatureName, LocationMin,
LocationMax, OrganismID, OrganismName);
  End loop;
  Close cur;

end if;


if (vSpecie is null and vGene is not null) then

  Open cur For query;
      i:=0;
  Loop
   i:=i+1;
   Fetch cur Into  Accession, Authors, GI, FeatureName, LocationMin, LocationMax, OrganismID,
OrganismName;
   Exit When cur%NOTFOUND ;

   INSERT INTO query (no_sess, Noseq, Accession, Authors, GI, FeatureName, LocationMin,
LocationMax, OrganismID, OrganismName)
   VALUES   (vNo_sess,to_char(i),Accession, Authors, GI, FeatureName, LocationMin,
LocationMax, OrganismID, OrganismName);
  End loop;
  Close cur;

End if;
--*******************
delete from query_tmp;

end;
```

2. Insérer l'enregistrement choisi

Cette fonction permet d'insérer les données d'un enregistrement au panier dans la table "Basket".

```
INSERT_BASKET   (no_seq number, no_sess varchar2) is

TYPE TYP_REF_CUR IS REF CURSOR ;
cur  TYP_REF_CUR ;
-- variables d'accueil
  q varchar2(1000);
  NoSeq1 QUERY.NOSEQ%Type ;
  Accession1 QUERY.accession%Type ;
  Definition1 QUERY.Definition%Type ;
  Authors1  QUERY.Authors%Type ;
  Sequence1  QUERY.Sequence%Type ;
  GI1 QUERY.gi%Type ;
  FeatureName1 QUERY.featurename%Type ;
  SequenceType1 QUERY.SequenceType%Type ;
  Location1 QUERY.Location%Type ;
  LocationMin1 QUERY.LocationMin%Type ;
  LocationMax1 QUERY.LocationMax%Type ;
  Product1 QUERY.Product%Type ;
  Note1 QUERY.Note%Type ;
  Translation1 QUERY.Translation%Type;
  OrganismId1 QUERY.organismId%Type ;
  OrganismName1 QUERY.organismName%Type ;
  Classification1 QUERY.Classification%Type ;
BEGIN
q :='SELECT NoSeq, Accession, Definition, Authors, Sequence, GI, FeatureName, SequenceType,
Location, LocationMin,
      LocationMax, Product, Note, Translation, OrganismID, OrganismName, Classification
      FROM query WHERE NoSeq =' || no_seq || ' AND no_sess= ' || "" || no_sess || "";
Open cur For q;
Loop
   Fetch cur Into  NoSeq1, Accession1, Definition1, Authors1, Sequence1, GI1, FeatureName1,
SequenceType1,  Location1, LocationMin1, LocationMax1, Product1, Note1, Translation1,
OrganismId1, OrganismName1, Classification1;

    Exit When cur%NOTFOUND ;

   INSERT INTO basket VALUES(no_sess,Accession1, Definition1, Authors1,Sequence1,GI1,
FeatureName1,   SequenceType1, Location1,LocationMin1, LocationMax1,Product1, Note1,
Translation1,OrganismID1,OrganismName1, Classification1);

End loop;
Close cur;
OUTPUT_LINE(q);
END;
```

3. Sauvegarder l'enregistrement désiré

Cette fonction permet la sauvegarde des tuples contenus dans le panier dans une table prévue "SAVEBASKET".

```
SAVEBASKET (codeBasket varchar2, echeance varchar2, no_sess varchar2) is

TYPE TYP_REF_CUR IS REF CURSOR ;
cur  TYP_REF_CUR ;

-- variables d'accueil
  q varchar2(1000);
  Accession1 SAVEDBASKET.accession%Type ;
  Definition1 SAVEDBASKET.Definition%Type ;
  Authors1  SAVEDBASKET.Authors%Type ;
  Sequence1  SAVEDBASKET.Sequence%Type ;
  GI1 SAVEDBASKET.gi%Type ;
  FeatureName1 SAVEDBASKET.featurename%Type ;
  SequenceType1 SAVEDBASKET.SequenceType%Type ;
  Location1 SAVEDBASKET.Location%Type ;
  LocationMin1 SAVEDBASKET.LocationMin%Type ;
  LocationMax1 SAVEDBASKET.LocationMax%Type ;
  Product1 SAVEDBASKET.Product%Type ;
  Note1 SAVEDBASKET.Note%Type ;
  Translation1 SAVEDBASKET.Translation%Type;
  OrganismId1 SAVEDBASKET.organismId%Type ;
  OrganismName1 SAVEDBASKET.organismName%Type ;
  Classification1 SAVEDBASKET.Classification%Type ;

BEGIN
q :='SELECT Accession, Definition, Authors, Sequence, GI, FeatureName, SequenceType, Location,
LocationMin,
      LocationMax, Product, Note, Translation, OrganismID, OrganismName, Classification
      FROM BASKET WHERE no_sess= ' || "" || no_sess || "";
Open cur For q;
Loop
   Fetch cur Into  Accession1, Definition1, Authors1, Sequence1, GI1, FeatureName1,
SequenceType1,  Location1, LocationMin1, LocationMax1, Product1, Note1, Translation1,
Organismld1, OrganismName1, Classification1;

 Exit When cur%NOTFOUND ;
 INSERT INTO SAVEDBASKET VALUES(codeBasket,echeance,Accession1, Definition1,
Authors1,Sequence1,GI1, FeatureName1,  SequenceType1, Location1,LocationMin1,
LocationMax1,Product1, Note1,  Translation1,OrganismID1,OrganismName1, Classification1);

End loop;
Close cur;

END;
```

4. Restaurer l'enregistrement désiré.

Cette fonction effectue le transfert des enregistrements de la table "SAVEBASKET" vers la table "BASKET".

```
RESTOREBASKET (codeBasket varchar2, no_sess varchar2) is

TYPE TYP_REF_CUR IS REF CURSOR ;
cur  TYP_REF_CUR ;

-- variables d'accueil
  q varchar2(1000);
  Accession1 SAVEDBASKET.accession%Type ;
  Definition1 SAVEDBASKET.Definition%Type ;
  Authors1  SAVEDBASKET.Authors%Type ;
  Sequence1  SAVEDBASKET.Sequence%Type ;
  GI1 SAVEDBASKET.gi%Type ;
  FeatureName1 SAVEDBASKET.featurename%Type ;
  SequenceType1 SAVEDBASKET.SequenceType%Type ;
  Location1 SAVEDBASKET.Location%Type ;
  LocationMin1 SAVEDBASKET.LocationMin%Type ;
  LocationMax1 SAVEDBASKET.LocationMax%Type ;
  Product1 SAVEDBASKET.Product%Type ;
  Note1 SAVEDBASKET.Note%Type ;
  Translation1 SAVEDBASKET.Translation%Type;
  OrganismId1 SAVEDBASKET.organismId%Type ;
  OrganismName1 SAVEDBASKET.organismName%Type ;
  Classification1 SAVEDBASKET.Classification%Type ;

BEGIN
q :='SELECT Accession, Definition, Authors, Sequence, GI, FeatureName, SequenceType, Location,
LocationMin, LocationMax, Product, Note, Translation, OrganismID, OrganismName, Classification
     FROM SAVEDBASKET WHERE nobasket= ' || "" || codeBasket || "";

Open cur For q;
Loop
   Fetch cur Into  Accession1, Definition1, Authors1, Sequence1, GI1, FeatureName1,
SequenceType1,  Location1, LocationMin1, LocationMax1, Product1, Note1, Translation1,
OrganismId1, OrganismName1, Classification1;

   Exit When cur%NOTFOUND ;

   INSERT INTO BASKET VALUES(no_sess,Accession1, Definition1, Authors1,Sequence1,GI1,
FeatureName1, SequenceType1, Location1,LocationMin1, LocationMax1,Product1, Note1,
                  Translation1,OrganismID1,OrganismName1, Classification1);
End loop;
Close cur;

END;
```

APPENDICE E

SCRIPT DE MISE À JOUR DE GBANK UQAM (C SHELL)

Dans cet appendice, nous présentons le script (C Shell) que nous avons développé dans le but de mettre à jour la base de données GBank UQAM. Ce script génère le fichier 'ftp.get' qui permet de télécharger et décompresser automatiquement les fichiers sources de NCBI.

```
#!/bin/sh
USER=anonymous
PASSWD=
ftp -n ftp.ncbi.nih.gov <<SCRIPT
user $USER $PASSWD
cd genbank
lcd /home/oradba/GB/ftp
binary
get gbchg.txt.gz
get gbnew.txt.gz
get gbdel.txt.gz
quit
SCRIPT
cd /data/oradata/GB/ftp
gzip -d gbchg.txt.gz
gzip -d gbnew.txt.gz
gzip -d gbdel.txt.gz
\cp -f header ftp.get && cat gbchg.txt gbnew.txt | tr "[A-Z]"  "[a-
z]" | awk -F"|" '{print "get gb"$1".seq.gz"}'  | sort -u >> ftp.get
&& cat footer >> ftp.get
echo "file: ftp.get.......done"
cat gbdel.txt gbchg.txt | awk -F"|" '{print $2}' | sort -u >
accession.delete
echo "file: accession.delete.......done"
cat gbnew.txt gbchg.txt | sort -u > accession.insert
echo "file: accession.insert.......done"
chmod +x ftp.get
echo "done"
```

Exemple de fichier ftp.get

```
#!/bin/sh
USER=anonymous
PASSWD=
ftp -n ftp.ncbi.nih.gov <<SCRIPT
user $USER $PASSWD
cd genbank
lcd /data/oradata/GB/FilesIn
binary
get gbpat15.seq.gz
get gbpat16.seq.gz
get gbpat17.seq.gz
get gbpat18.seq.gz
get gbpat19.seq.gz
get gbpat20.seq.gz
get gbpri29.seq.gz
get gbpri30.seq.gz
get gbpri3.seq.gz
get gbpri4.seq.gz
get gbpri5.seq.gz
get gbpri6.seq.gz
.
.
.
get gbpri8.seq.gz
get gbpri9.seq.gz
get gbrod10.seq.gz
get gbrod11.seq.gz
get gbrod12.seq.gz
get gbrod13.seq.gz
get gbrod14.seq.gz
get gbrod15.seq.gz
get gbrod16.seq.gz
get gbrod17.seq.gz
get gbrod18.seq.gz
get gbrod19.seq.gz
get gbrod1.seq.gz
get gbrod20.seq.gz
get gbrod21.seq.gz
get gbrod22.seq.gz
get gbrod23.seq.gz
get gbrod24.seq.gz
get gbrod2.seq.gz
get gbrod3.seq.gz
get gbrod4.seq.gz
get gbrod5.seq.gz
quit
SCRIPT
cd /data/oradata/GB/FilesIn
gzip -d *
```

BIBLIOGRAPHIE

[1] http://www.ncbi.nlm.nih.gov/GenBank/ page web du logiciel T-REX.

[2] http://www.ncbi.nlm.nih.gov page web de la base de données de NCBI

[3] http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html page web de la base de données de NCBI. Exemple d'un enregistrement de données dans GenBank.

[4] http://www.ncbi.nlm.nih.gov/GenBank/submit.html page web de la base de données de NCBI. Transfert des données dans GenBank.

[5] http://www.ncbi.nlm.nih.gov/GenBank/update.html page web de la base de données de NCBI. Mise à jour des donnees dans GenBank.

[6] http://fr.wikipedia.org/wiki/R%C3%A8gne page web du wikipedia. Encyclopédie.

[9] http://www.ncbi.nlm.nih.gov/projects/collab/FT/index.html page web de la banque de données de NCBI.

[10] http://profils.monster.fr/oraclefr/ page web du profils de l'entreprises d'Oracle.

[11] http://www.commentcamarche.net/oracle/oracarchi.php3 page web du site commentcamarche (Architecture du SGBD Oracle).

[12] http://www.cxotoday.com/cxo/jsp/article.jsp?article_id=71523&cat_id=910 page web de cxotoday (l'architecture de la grille d'Oracle 10g).

[14] http://www.Oracle.com/enterprise_manager/docs/Managing-Applications-with-Oracle-Enterprise-Manager-10g.pdf page web d'Oracle.

[15] Oracle Database Concepts 10g Release 2 (10.2) edition Oracle Inc. October 2005.

[16] Razvan Bizoï, Oracle 10g administration, Paris : Tsoft éditeur : Eyrolles, c2006.

[17] Ian Abramson, Michael Abbey et Michael Corey, Oracle 10g notions fondamentales [traduit de l'anglais par Freenet]. Paris : CampusPress, 2004

[18] Gaja Krishna Vaidyanatha, Kirtikumar Deshpande, John A. Kostelac. Optimisation des performances sous Oracle, Paris : CampusPress, 2001

[19] Bales, Donald, Java programming with Oracle JDBC. Beijing : O'Reilly, 2002.

[20] Andrew Odewahn, Oracle et Web, Paris : O'Reilly , 2001.

[21] Boc, A. et Makarenkov, V. New Efficient Algorithm for Detection of Horizontal Gene Transfer Events. Algorithms in Bioinformatics, G. Benson and R. Page (Eds.), 3rd Annual WABI'03, Springer-Verlag, 190-201. 2003.

[23] Makarenkov, V. Boc, A. et Diallo, A. B. Representing lateral gene transfer in species classification. Unique scenario. Accepté pour publication à IFCS, Chicago. 2004.

[24] Makarenkov, V., Lapointe, F.-J. A weighted least-squares approach for inferring phylogenies from incomplete distance matrices, accepted for publication in Bioinformatics. 2004.

[25] Makarenkov, V., Leclerc, B. An algorithm for the fitting of a tree metric according to a weighted least-squares criterion, Journal of Classification 16 3-26. 1999.

[26] Makarenkov, V., Leclerc, B. The fitting of a tree metric to a given dissimilarity with the weighted least squares criterion. Journal of Classification 16 : 3-26. 1999.

[27] Makarenkov, V., Leclerc, B. Tree metrics and their circular orders: some uses for the reconstruction and fitting of phylogenetic trees, in Mathematical

hierarchies and Biology (B. Mirkin, F.R. McMorris, F. Roberts, A. Rzhetsky, eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Amer. Math. Soc., Providence, RI, 183-208. 1997.

[28] Makarenkov, V. T-Rex: reconstructing and visualizing phylogenetic trees and reticulation networks. Bioinformatics, 17 : 664-668. 2001.

[29] Cavalli-Sforza, L.L. & Edwards, A.W.F. Phylogenetic analysis. Model and estimation procédures. Am. J. Hum. Genet. 19 : 223. 1967.

[30] Darwin, C. On the origin of species, John Murray, 1859.

[31] De Soete, G. Additive-Tree Representations of Incomplete Dissimilarity Data, Quality and Quantity, 18, 387-393. 1984.

[32] Efron, B. Bootstrap methods: Another look at the jackknife. The Annals of Statistics, 7 : 1-26. 1979.

[33] Felsenstein, J. Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. 17 : 368-376. 1981.