Institut de recherche pour le développement

université de bretagne occidentale

Développement d'une base de données relationnelle

Exploitation des statistiques de pêche au thon tropical

© IFREMER, Olivier Barbaroux

Stage réalisé au centre IRD de Brest
Responsable : M. Jean-Jacques Lechauve

Rapport de stage – Masson Laurent
Etudiant en IUP GMI 2ème année
Année 2001 / 2002

Sommaire

Remerciements

Je tiens à remercier tout particulièrement mon maître de stage, Jean-Jacques Lechauve, qui m'a très bien encadré. Il a su répondre à mes questions d'ordre scientifique et technique, j'ai beaucoup appris durant ces trois mois de stage grâce à lui.

Je remercie aussi Daniel Corre, le responsable des ressources informatiques du centre IRD de Brest, qui m'a fourni les logiciels nécessaires au bon déroulement du stage, et n'a pas hésité à prendre sur son temps pour répondre à mes questions.

Merci à Michèle Fichaut, du centre Ifremer qui nous a reçu, mon maître de stage et moi, pour répondre à nos questions techniques à propos du logiciel Oracle.

Introduction

« L'importance du secteur thonier de l'Union Européenne est considérable, avec des captures annuelles de 500.000 tonnes d'espèces qui sont en majorité de bonne valeur économique. Ces captures placent l'Union Européenne en deuxième position mondiale, derrière le Japon, pour le volume des captures.

Dominée quantitativement par l'Espagne et la France, cette pêche thonière reste un secteur important, au moins dans divers pays de l'Union : Italie, Grèce, Portugal. On note que ces prises de thons par les flottilles de l'Union Européenne ont manifesté de 1960 à 1995 un remarquable accroissement. La pression halieutique sur ces ressources et sur les écosystèmes hauturiers qu'elles habitent est par ailleurs en fort accroissement dans tous les océans, entraînant des risques accrus de conservation des ressources thonières et des écosystèmes. Ceci nécessite la mise en œuvre de recherches scientifiques actives par l'Union Européenne, recherches qui reposent d'abord sur de bonnes statistiques de pêche (respectant les normes internationales de toutes les pêcheries thonières). »

Dr Alain FONTENEAU, IRD, Rapport de faisabilité : Observatoire thonier ORDET, Rapport final du projet 96-041

Les statistiques de la flottille visant les thons tropicaux, font l'objet d'un suivi détaillé depuis le début des années 60 pour la France et 1979 pour l'Espagne. Dans le cadre du programme TESS (98-062) (« Thons, Echantillonnages, Systèmes Statistiques »), étape intérimaire du vaste projet ORDET (Observatoire de Recherches sur la Dynamique de l'Exploitation des Thonidés), la définition et la mise en place des nouveaux outils informatiques sont au cœur de la phase de transition entre l'ancien système et le nouveau.

Les outils informatiques évoluant, un gros travail de traitement des données historiques a dû être fourni, pour pouvoir utiliser au sein d'un même système, à la fois les données des années 60, et les données actuelles.

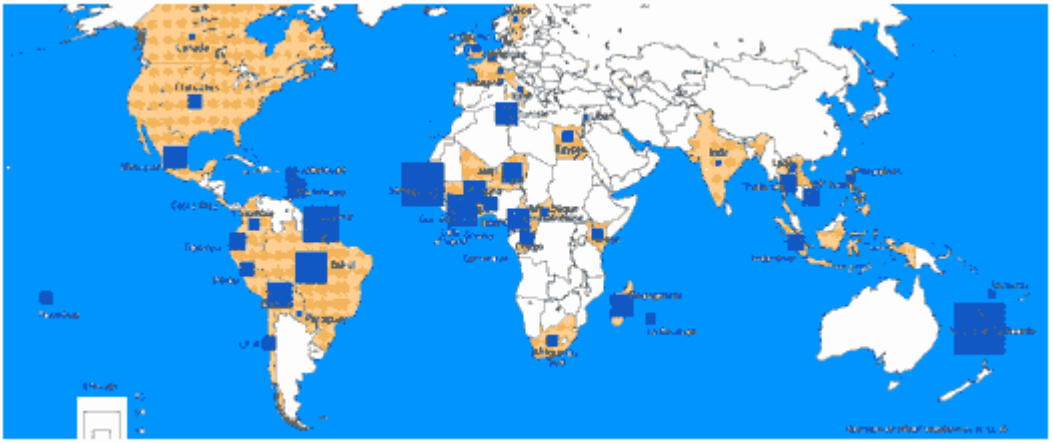
Dans la réalisation de cette migration, l'une des tâches cardinales consiste à concevoir et implémenter les bases de données nécessaires à l'archivage et à l'exploitation de la considérable masse de données acquises depuis trois décennies. L'objet de la proposition de stage concerne la construction et la mise en place d'une base de données relationnelle Oracle sous Unix.

Présentation de l'IRD

Présentation générale

L'IRD (Institut de Recherche pour le Développement, anciennement ORSTOM), est un établissement public à caractère scientifique et technologique, placé sous la double tutelle des ministères chargés de la Recherche et de la Coopération. Depuis cinquante ans, L'institut conduit des recherches sur les milieux intertropicaux, lesquelles sont devenues des références internationales.

L'IRD mène des recherches dans le monde entier, la carte ci-dessous illustre son implantation (35 en tout)



Ses trois missions fondamentales sont la recherche, l'expertise et la formation. Pour ce faire, l'IRD conduit des programmes scientifiques centrés sur les relations entre l'homme et son environnement dans les pays du sud, dont l'objectif est de contribuer à leur développement durable.

Il propose à ses partenaires du sud et aux acteurs du développement, des recherches dans les grands domaines suivants :

- Milieux et environnement
- **Ressources vivantes** (domaine relatif au stage)
- Sociétés et santé

Budget 2000 : 175 millions d'Euros (soit 1,148 milliard de francs) de budget total dont 76 % de dépenses de personnel .

Son effectif comprend 2165 agents dont 770 chercheurs, 754 ingénieurs, techniciens et administratifs, ainsi que 641 personnels de statut divers (personnels locaux, marins…).

Le centre IRD de Brest

Considéré comme pôle de référence de l'IRD en matière de recherche océanologique, le centre de Bretagne est une base privilégiée des Départements Milieux et Environnement, et Ressources Vivantes.

Situé dans un environnement scientifique particulièrement favorable, il est au cœur du campus scientifique le plus important d'Europe en ce qui concerne les Sciences de la Mer (près de mille chercheurs et ingénieurs) et il est intégré au Technopôle brestois (une cinquantaine d'entreprises privées ou publiques).

La recherche

L'IRD est structuré en Unités de Recherche et Unités de Service, dont certaines sont implantées au Centre de Brest.

- Moyens scientifiques à la mer (instrumentation scientifique et embarquée).
- Observatoire de l'océan (acquisition, validation et archivage des données pour la communauté scientifique).
- Acoustique marine et applications en recherche halieutique (recherches méthodologiques afin d'utiliser au mieux les technologies avancées et recherches spécifiques sur les populations de ressources halieutiques).
- Sclérochronologie (étude des pièces squelettiques des animaux marins, afin de déterminer la croissance, l'âge, conditions de l'environnement…).
- Gestion des bases de données relationnelles (développée notamment dans le cadre des recherches thonières).
- Modélisation du climat de l'Atlantique tropical.

La formation

Outre les enseignements délivrés dans les différents DEA de l'École doctorale de l'Université de Bretagne Occidentale et les accueils de stagiaires et doctorants, le centre propose des formations spécialisées (Sclérochronologie, Acoustique) aux scientifiques nationaux ou étrangers.

La capitalisation scientifique

Le Centre de Documentation de l'IRD existant à Brest est renforcé par son intégration au Centre Européen de Documentation de la Mer (CEDM), pôle associé de la Bibliothèque de France. Le CEDM centralisera les fonds de l'IFREMER, de l'UBO et de l'IRD dont la fusion est déjà abordée.

Parmi les nombreuses banques de données accessibles ici, certaines sont de la responsabilité de l'Institut au plan international comme le Centre TOGA/WOCE de données de subsurface.

Les développements technologiques

Les développements technologiques, notamment en acoustique sous-marine ou en acquisition de paramètres environnementaux, sont privilégiés sur une base contractualisée avec les partenaires du Campus ou du Technopôle.

D'autre part, la gestion des moyens navigants de l'IRD, constitués de deux navires de recherches hauturières, l'Antéa, basé à Abidjan (Côte d'Ivoire), et l'Alis, basé à Nouméa (Nouvelle Calédonie), est assurée au centre de Brest.

Département Ressources vivantes

Le département ressources vivantes se consacre à l'étude de la biodiversité, des écosystèmes aquatiques (marins, littoraux, continentaux) et des agrosystèmes tropicaux (amélioration génétique, défense des plantes cultivées, biotechnologies...) afin d'assurer la viabilité de leur exploitation par une gestion appropriée.

Contexte scientifique

C'est dans le cadre de deux projets européens, ORDET puis TESS, que se situe le travail qui m'a été confié.

Le projet ORDET

Les objectifs du projet ORDET (Observatoire de Recherches sur la Dynamique de l'Exploitation des Thonidés) sont les suivants :

- L'objectif primaire du projet ORDET est de créer un outil de recherche thonières efficace à l'échelle européenne.
- Sur le plan technique, le but est la mise en place d'un réseau informatique doté d'un laboratoire central qui gère les données, les outils d'analyse, et la documentation sur les thonidés mondiaux et sur leur exploitation.
- La vocation d'ORDET est pluridisciplinaire, elle vise tous les domaines nécessaires à une recherche halieutique efficace sur les thonidés tels que la biologie, la dynamique des populations, l'environnement, l'économie, etc…
- ORDET s'adresse à des groupes cibles très divers : principalement les scientifiques, mais aussi le secteur de la pêche, les administrations des pêches (nationales et européennes) et le grand public.
- ORDET respectera la totale confidentialité des données fines nationales, et visera essentiellement à permettre une pleine utilisation des données scientifiques qui sont dans le domaine public. Les données confidentielles resteront dans les laboratoires nationaux.
- etc…

Phase de transition entre l'étude de faisabilité du projet ORDET et sa mise en œuvre effective, le projet TESS (Thons, Echantillonnages, Systèmes Statistiques) a pour but de réaliser une partie des tâches d'ORDET. Lors du déroulement de ce projet, les autorités compétentes ont décidé de surseoir à la réalisation du projet ORDET.

Le projet TESS

L'objectif principal de ce projet est d'améliorer le système de collecte des statistiques de la pêche thonière tropicale européenne dans les océans Atlantique et Indien. Pour ce faire, il doit remplir différentes tâches :

- Assurer la transition entre l'ancien système et le nouveau (en conservant les données acquises depuis les années 50).
- Définir et mettre en place les nouveaux outils.
- Assurer la continuité de la collecte des données thonières.
- etc…

Réseau d'observation

Les données thonières sont saisies à partir de livres de bord et de mensurations, remplis par les pêcheurs et des enquêteurs. Depuis plus de 25 ans, l'IRD et l'IEO (Institut Espagnol d'Océanographie), en partenariat avec la Côte d'Ivoire et le Sénégal pour l'Atlantique, et Madagascar et Les Seychelles pour l'océan Indien, ont collecté un très grand nombre de ces livres.

Tous ces livres, aussi appelés *logbook*, ont été photographiés puis stockés dans une grande base d'images (des dizaines de milliers) permettant ainsi de conserver les données de base.

Validation des données

Afin de reconstituer des collections de données homogènes à partir des données historiques et des données récentes, l'application AVDTH (Acquisition et Validation des Données de pêche au Thon Tropical) a été utilisée.

Ce logiciel trilingue (français, espagnol, anglais), développé à l'IRD, s'appuie sur une base de données relationnelle bureautique. Il est exploité au quotidien dans les océans Atlantique et Indien. Toutes les collections de données de base sont ainsi au même standard.

Bases de données centrales

Différentes bases de données centrales doivent accueillir les données de l'observatoire :

- **Données de pêche**
- Données observateur
- Caractéristiques techniques des senneurs
- Données sur l'environnement

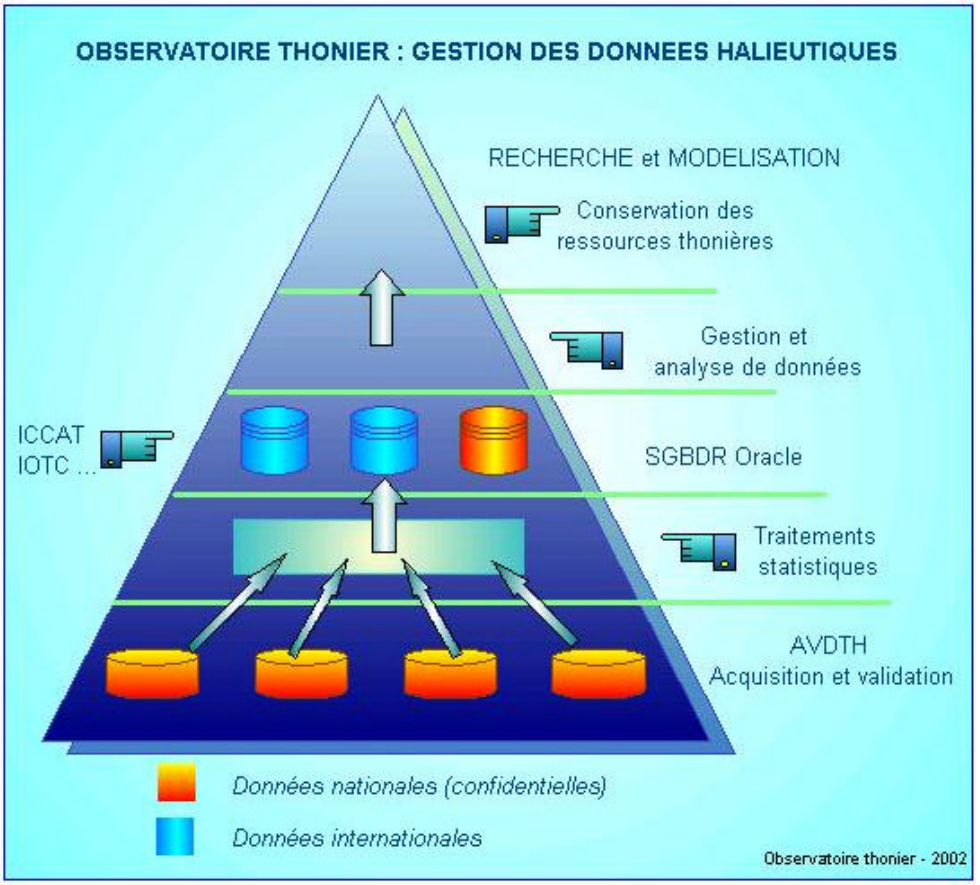
L'objet de mon stage consiste à créer la base de données de pêche qui accueillera les données extrapolées et standardisées. Cette base se nomme **ALIPA** (**A**lbacore, **Li**stao et **Pa**tudo sont les trois espèces de thons tropicaux les plus fréquentes dans la base).

Contenu de la base ALIPA

La base de données ALIPA contiendra à terme des données de pêche à différentes échelles :

- Données nationales de captures et mensurations à grande échelle (très sensibles) pour les océans Atlantique et Indien. Ces données resteront dans les pays d'origine. Données au coup de filet, positions géographiques au centième de degré.
- Données internationales, d'étude des stocks pour les océans Atlantique et indien. Echelle plus petite, captures : strates de 1°/mois, mensurations : strates de 5°/mois.
- Données internationales, captures annuelles par espèce et par grandes régions océaniques pour les océans Atlantique, Indien et Pacifique.
- Eventuellement d'autres données ajoutées plus tard…

La figure ci-dessous illustre les objectifs d'ORDET, qui ont ensuite été repris dans le projet TESS. Les deux pyramides l'une sur l'autre représentent deux systèmes de gestion de données équivalents, dans deux pays du réseau (à savoir la France et l'Espagne).



Le but de ce stage concerne exclusivement les données nationales. Le modèle conceptuel de données retenu est tout naturellement une déclinaison du modèle AVDTH. La construction de la base ALIPA sera réalisée sous Oracle 8 version Entreprise. Ce SGBDR est parfaitement adapté à la gestion de différentes classes de données au sein d'une même base.

Cahier des charges

Le début de mon projet consiste à reprendre le modèle conceptuel de données de l'application AVDTH, et de l'adapter aux besoins de la nouvelle base. Ensuite, il me faut générer le modèle physique de données, et créer la base de données Oracle 8 sous Unix.

Les données contenues dans cette base sont confidentielles. Une gestion minutieuse des accès sera effectuée, grâce à différents comptes utilisateurs Oracle. Chaque compte disposera de droits spécifiques.

Une interface PHP sera développée pour assurer la gestion (insertion, suppression) des jeux de données. L'entité « jeu de données » sera introduite au chapitre conception. Elle sera particulièrement bien commentée afin de servir d'exemple pour les applications futures.

Présentation d'Oracle

Durant ce stage il m'a été demandé de concevoir et d'implémenter une base de données relationnelle Oracle sous Unix. Ayant déjà découvert le langage PL/SQL en première année d'IUP, et la modélisation du SGBDR Oracle, la nouveauté pour moi aura été de d'appréhender la partie administration du serveur Oracle.

C'est pourquoi je ne présenterai ici que les différents aspects à maîtriser avant de pouvoir créer une base de données sous Oracle.

Historique

La création d'Oracle Corporation eut lieu en 1977, la première version fut créée en 1979, faisant d'Oracle le premier SGBDR commercialisé, apparition du langage SQL (Première version d'Oracle écrite intégralement en assembleur).

Entre 1979 et 1984, plusieurs outils ont été créés pour faciliter l'utilisation d'Oracle (notamment en C, Fortran, Cobol…), jusqu'en 1983 ou il y'eut une réécriture en C offrant ainsi une plus grande portabilité.

En 1984, une première version sort sur PC, ce qui fait d'Oracle le premier SGBD authentiquement relationnel disponible et portable sur PC.

En 1986, l'architecture client/serveur est introduite, et permet ainsi la distribution des accès et la répartition des données (limitée). En 1988, la version 6 voit le jour, offrant de meilleures performances, c'est aussi a cette date qu'apparaît le PL/SQL.

En 1992, la version 7 pour Unix permet à Oracle de rattraper le retard par rapport à ses concurrents, notamment au niveau des procédures stockées et de bases de données réparties. Beaucoup d'autres changements interviennent dans cette version (gestion mémoire, CPU, E/S). En 1995, la version 7 pour Windows sort.

En 1996, Oracle 8 est la première version orientée objet du système, puis amélioration en 1997, supportant plus d'utilisateurs et plus de données. Deux ans plus tard la version 8i (ou version 8.1.5) avec l'intégration de Java. En 2000, sortie de la version Oracle 8i version 2, puis de la 9i en 2001 (version 9.0.1).

La version d'Oracle installée à l'IRD est la version 8 (Release 8.0.5)

Un serveur de données Oracle est constitué d'une base de données (ensemble des fichiers utiles à la base), et d'une instance (ensemble des process Oracle et de la mémoire). Voici la présentation en détail de ce qu'est une base de données, puis de ce qu'est une instance Oracle.

La base de données

Une base de données Oracle est constituée d'une structure physique et d'une structure logique. Les deux sont indépendantes ; la structure physique peut être modifiée sans affecter l'accès aux structures de stockage logiques.

La structure physique

Elle est constituée d'un ensemble de fichiers qui peuvent être de quatre types différents :

- Fichiers de données (*datafiles*)
- Fichiers de journalisation *(redo log files*)
- Fichiers de contrôle (*control files*)
- Autres fichiers

Les fichiers de données

Ils contiennent, comme leur nom l'indique, toutes les données de la base (tables, index, vues…) ainsi que celles nécessaires au fonctionnement d'Oracle (dictionnaire de données). Voici leurs principales caractéristiques :

- Il faut au moins un fichier de données dans la base, mais il est fortement conseillé d'en utiliser plusieurs afin d'organiser les données sur le(s) disque(s). Par exemple, pour les tables contenant beaucoup de données, on sépare généralement les index des tables en les plaçant dans différents *tablespaces*, et si possible sur différents disques. On peut ainsi accéder aux données et aux index simultanément.
- Un fichier de données ne peut être associé qu'à une et une seule base de données.
- La taille d'un *datafile* est spécifiée lors de sa création mais il se peut que la taille de ce fichier augmente automatiquement si la base de données arrive à saturation.
- Un ou plusieurs fichiers de données constituent une unité de stockage logique d'Oracle appelée *Tablespace* (notion définie plus loin dans ce rapport).

Les fichiers de journalisation

Toute base de données Oracle contient au moins deux fichiers de journalisation. Leur fonction est d'enregistrer toutes les modifications effectuées sur les données afin de pouvoir garantir la cohérence des données après un incident logiciel ou matériel.

Oracle offre la possibilité de dupliquer ces fichiers pour une plus grande sécurité. Ainsi les informations sont écrites simultanément sur plusieurs fichiers, dans le cas d'un crash de disque dur, on pourra donc retrouver un fichier en bon état sur un autre disque par exemple.

Les fichiers de contrôle

Ces fichiers contiennent des informations indispensables au démarrage de la base de données telles que le nom de la base, le nom et l'emplacement des fichiers de données et de journalisation, la date de création de la base…

De même que pour les fichiers de journalisation, il est intéressant de créer des copies sur différents disques afin de prévoir des pannes matérielles éventuelles.

Autres fichiers

Fichier de configuration du système généralement appelé *initNomBase.ora*, il contient diverses informations nécessaires au démarrage de la base (emplacement des fichiers de contrôles, nom de la base de données…)

La structure logique

Sous Oracle, la structure logique est articulée autour de :

- Schémas ou collections d'objets (*Schema objects*)
- Unités de stockage logiques (*Tablespaces*, segments, extensions, blocs)

Les schémas

Les schémas constituent la structure relationnelle de la base de données, ils sont constitués d'une collection d'objets (tables, vues, index, procédures, triggers…).

Il n'y a aucune relation de dépendance entre un *tablespace* et un schéma. Les objets d'un schéma peuvent être contenus dans différents *tablespaces*, et un *tablespace* peut contenir des objets de différents schémas.

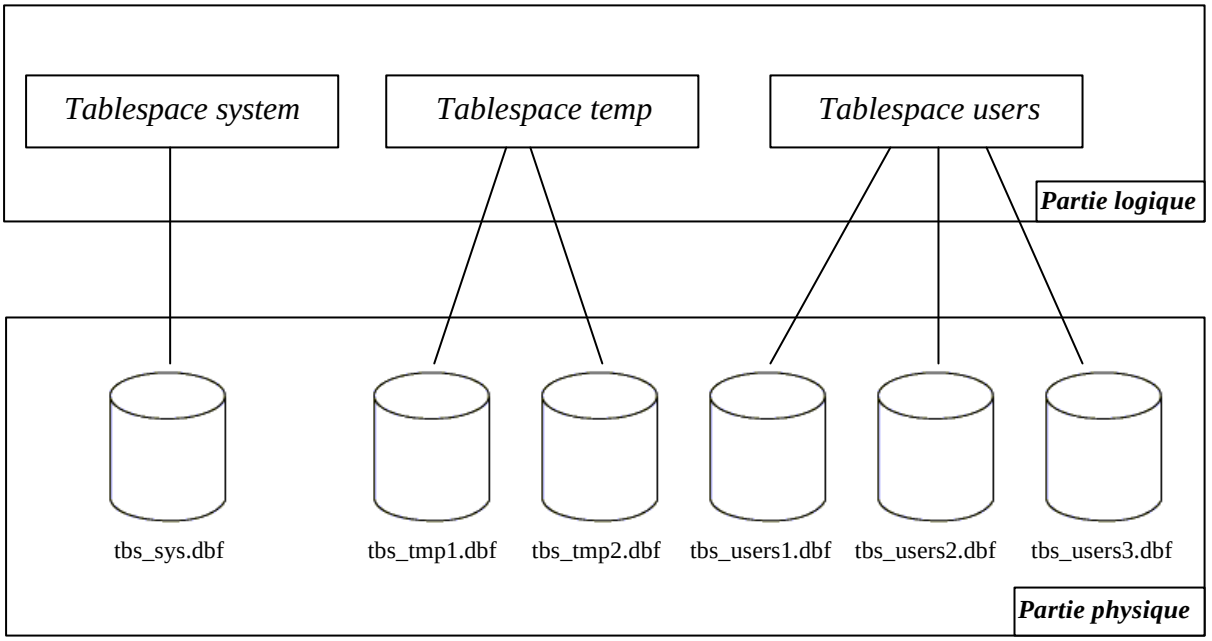
Les tablespaces

Une base de données Oracle est divisée en unités de stockage logiques appelées *tablespaces*. Le rôle d'un *tablespace* est de regrouper logiquement des structures de données, il permet ainsi d'avoir de « l'ordre » dans la base, et permet parfois d'accélérer certaines transactions.

Lors de la création d'une nouvelle base de données, un *tablespace* appelé SYSTEM est créé par défaut, c'est lui qui contient le dictionnaire de données, il est préférable d'en créer au moins un deuxième afin d'y stocker les données.

Comme on le voit sur le schéma suivant, les données d'un même *tablespace* peuvent se trouver dans un seul fichier de données physiques (*datafiles*) ou dans plusieurs. C'est lui qui fait le lien entre la partie logique et la partie physique de la base de données.

Sur la figure ci-dessous, deux *tablespaces* supplémentaires sont présentés, un pour les données temporaires (utilisé lors des opérations de tris surtout), et un pour le stockage des données des utilisateurs.



Note : Chaque objet logique doit être contenu dans **un et un** seul *tablespace*.

La capacité de stockage d'un *tablespace* est la somme des capacités de stockage de chacun des *datafiles* qui le constitue. La capacité de stockage de la base de données est la somme des capacités de chacun des *tablespaces* qui la constitue.

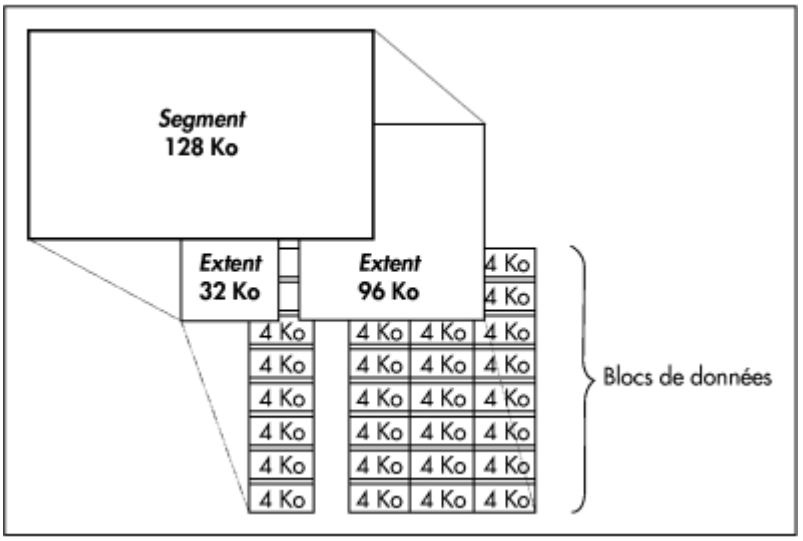
Segments, extensions et blocs

Les segments sont la contrepartie physique des objets logiques de la base. Leur rôle est de stocker les données. Lors de la création d'un fichier, Oracle alloue immédiatement tout l'espace qui lui est associé. A l'intérieur de ces fichiers, l'espace disque est géré dynamiquement. Cette gestion s'appuie sur 3 niveaux de structures logiques :

- Sous Oracle, la plus petite unité logique d'entrée/sortie est le bloc de données (*data block*), il a généralement une taille de 1024, 2048, 4096 ou 8192 octets (il est possible d'aller au-delà). La taille de ce bloc est déterminée à la création de la base et ne peut plus être changée ensuite.
- Le niveau suivant est constitué par les extensions (*extents*). Un *extent* est composé d'un certain nombre de blocs contigus alloués pour stocker un certain type de données à l'intérieur d'un segment.
- Le dernier niveau est appelé *segment*. C'est une suite d'*extents* alloués à un *tablespace*.

Exemple : Les données contenues dans les tables et les index sont stockées dans des segments différents. Quand tous les *extents* d'un segment sont pleins, le serveur Oracle lui alloue un nouvel *extent*. Ces opérations peuvent provoquer la création d'*extents* non contigus sur les disques.

Le schéma ci-dessous représente un segment de données, composé de 2 *extents*, chacun composé de blocs de données (qui sont de 4 Kilo octets ici).



Il existe cinq types de segments :

- segments de données,
- segments d'index,
- segment d'amorçage,
- segments d'annulation (rollback segments),
- segments temporaires

Les rollback segments

Les segments d'annulation permettent d'enregistrer les actions effectuées par les transactions. Ils contiennent les données avant modification. Les effets d'une transaction peuvent ainsi être annulés au besoin.

Ces segments sont indispensables dans toute base Oracle. Lors de la création d'une base, un rollback segment est créé dans le tablespace SYSTEM. Il faut ensuite en créer un nouveau pour créer d'autres tablespaces.

Les segments temporaires

Ces segments sont utilisés lors de traitement de commandes SQL nécessitant un espace disque temporaire (généralement des opérations de tri). Si aucun tablespace temporaire n'est défini dans une base, Oracle utilise le tablespace SYSTEM par défaut.

Le segment d'amorçage

Le segment d'amorçage (bootstrap segment) est utilisé une seule fois, lors de la création de la base. Ce segment ne nécessite aucune attention de la part de m'administrateur de la base.

L'instance

A chaque démarrage d'une base Oracle, une zone mémoire est allouée par le serveur (cette zone s'appelle la **SGA** pour System Global Area), et plusieurs process sont lancés. L'ensemble de ces process et de la SGA est appelé une **instance**.

Plusieurs instances Oracle peuvent s'exécuter simultanément sur un système, dans ce cas, chacune d'elle aura sa propre base de données physique, son propre espace mémoire, et ses propres process.

La SGA

Le serveur Oracle utilise cette zone mémoire pour stocker diverses informations :

- Code des programmes exécutés
- Informations sur les sessions connectées
- Données nécessaires à l'exécution du programme
- Informations partagées et communiquées aux process Oracle
- Blocs de données de la base présents dans le cache mémoire (*Database buffer cache*)

Les process

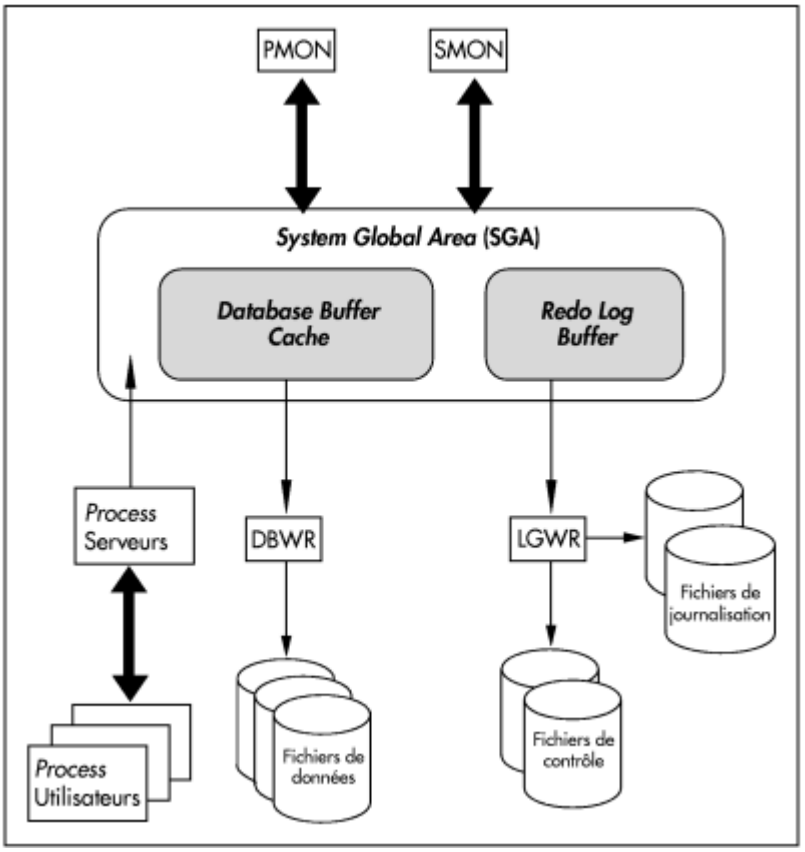
En règle générale, une instance Oracle utilise plusieurs process, qui ont chacun un rôle bien défini au sein du serveur Oracle (Exception pour systèmes mono-utilisateurs). Les process Oracle peuvent être de deux types : arrière-plan (*background*) ou serveur.

Voici une liste des principaux process d'arrière-plan qui tournent sur le serveur Oracle d'un système multi-utilisateurs :

- Process utilisateurs : Programmes d'application lancés par les utilisateurs
- Database Writer (DBWR) : Ecrit les modifications apportées au cache dans les fichiers de données.
- Log Writer (LGWR) : Ecrit le contenu des buffers de la SGA dans les fichiers de journalisation.
- System Monitor (SMON) : Chargé du démarrage des instances Oracle, et de la libération des segments temporaires qui ne sont plus utilisés.
- Process Monitor (PMON) : Libère les ressources des processus utilisateurs en cas de problème. Assure aussi la libération des segments temporaires inutilisés.

Les process serveur ont pour but de compiler et d'exécuter les ordres SQL transmis par les applications, de charger les blocs de données depuis les *datafiles* dans la SGA, puis de retourner les résultats des requêtes aux applications.

Le schéma suivant montre l'architecture générale d'une instance Oracle, on voit bien ici que les différents process Oracle communiquent avec la SGA.


PMON
SMON
System Global Area (SGA)
Database Buffer Cache
Redo Log Buffer
Process Serveurs
DBWR
LGWR
Fichiers de journalisation
Process Utilisateurs
Fichiers de données
Fichiers de contrôle

Conception

Avant de modifier le Modèle Conceptuel de Données de l'application AVDTH, j'ai dû me familiariser avec certains termes de la pêche thonière, comprendre et assimiler le rôle des différentes tables du modèle.

Notions fondamentales

Activité

Une activité est un intervalle de temps (au plus 24 heures), pendant lequel l'unité de pêche accomplit une ou plusieurs opérations de même nature. Pour mieux saisir la notion d'activité, examinons la liste des tuples de la table AVDTH.OPERA :

C_OPERA	L_OP8L	L_OPERA
0	coup -	coup(s) nul(s)
1	coup +	coup(s) positif(s)
2	coup ?	détail inconnu
3	rech	Recherche
4	route	route sans veille
5	dcp +	pose de DCP
6	dcp -	relève de DCP
7	avarie	Avarie
8	cape	à la cape
9	appât	à l'appât
10	attente	en attente
11	transb	transbordement en mer

Le détail des activités d'une marée est communiqué par le commandant du bateau. Chaque activité est consignée sur un livre de bord. Ce document sera récupéré en général par l'enquêteur au terme de la marée.

Dans ALIPA, la notion d'activité prend en compte les seules opérations de pêche et de recherche. Les captures, les temps de mer et de pêche sont extrapolés. De nouvelles variables sont introduites : le temps de pêche standardisé et la durée du coup de senne.

Les informations principales relevées lors d'une activité sont : la date, la position (en degrés minutes), le poids de poisson capturé pour chaque espèce, comment le banc de poisson a été repéré…

Note : Un exemple de logbook vierge est fourni en annexe de ce rapport.

Echantillon

Lorsque du poisson est pêché, il est réparti dans les cuves du bateau par le frigoriste, qui lui aussi remplit un cahier. Les plans de cuve seront également remis à l'enquêteur.

Les échantillons sont réalisés dans des strates spatio-temporelles conformément aux exigences du plan d'échantillonnage. Suivant les informations recueillies dans ces cahiers, l'enquêteur choisit la ou les cuve(s) dans laquelle ou lesquelles il va réaliser des comptages et des mesures (*).

(*) Il existe différents types de mesure des poissons. Dans AVDTH, deux sont utilisées :

- Mesure *LF* (Longueur Fourche): Taille totale du poisson, utilisée pour les petits poissons.
- Mesure *LD1* : Taille de la tête à la première dorsale, utilisée pour les gros poissons.

Dans ALIPA, les mesures de poisson sont exclusivement en *LF* (conversion des données de *LD1* vers *LF*), et les effectifs sont extrapolés.

Note : Un exemple de plan de cuve vierge est fourni en annexe de ce rapport.

Jeu de données

Un jeu de données est une collection de données scientifiquement validée, concernant des mesures de captures/efforts, ou de mensurations, sur une période de temps donnée (au plus un an). Une fois introduit dans ALIPA, un jeu de données peut être retiré et remplacé par un autre jugé meilleur.

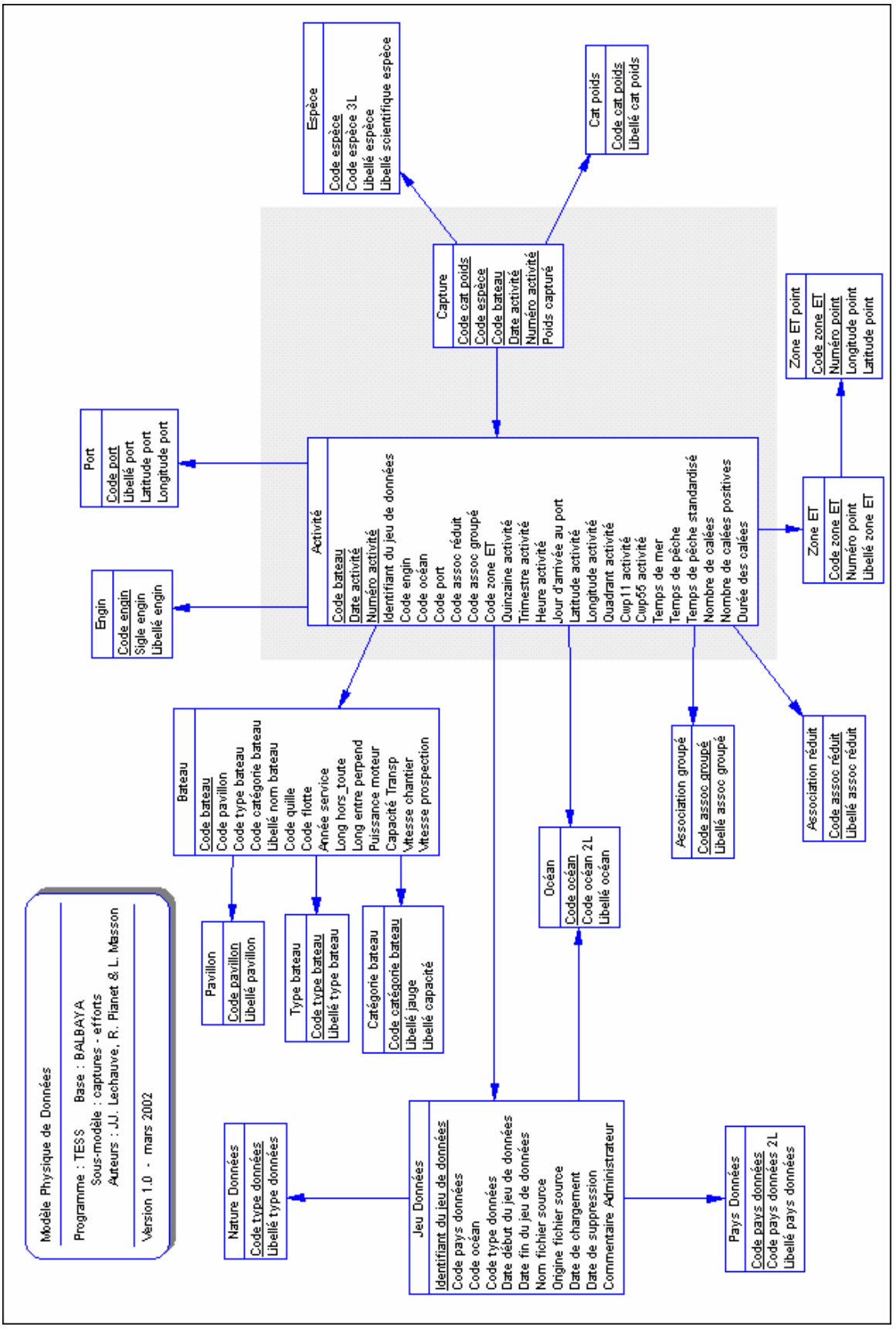
Leur rôle est d'archiver les étapes successives d'insertion et de suppression des données. En outre, on souhaite conserver les coordonnées des différents fichiers sources.

Les trois notions définies ci-dessus traduisent la présence des trois entités majeures (activité, échantillon et jeu de données) que l'on trouve dans le Modèle Conceptuel de Données.

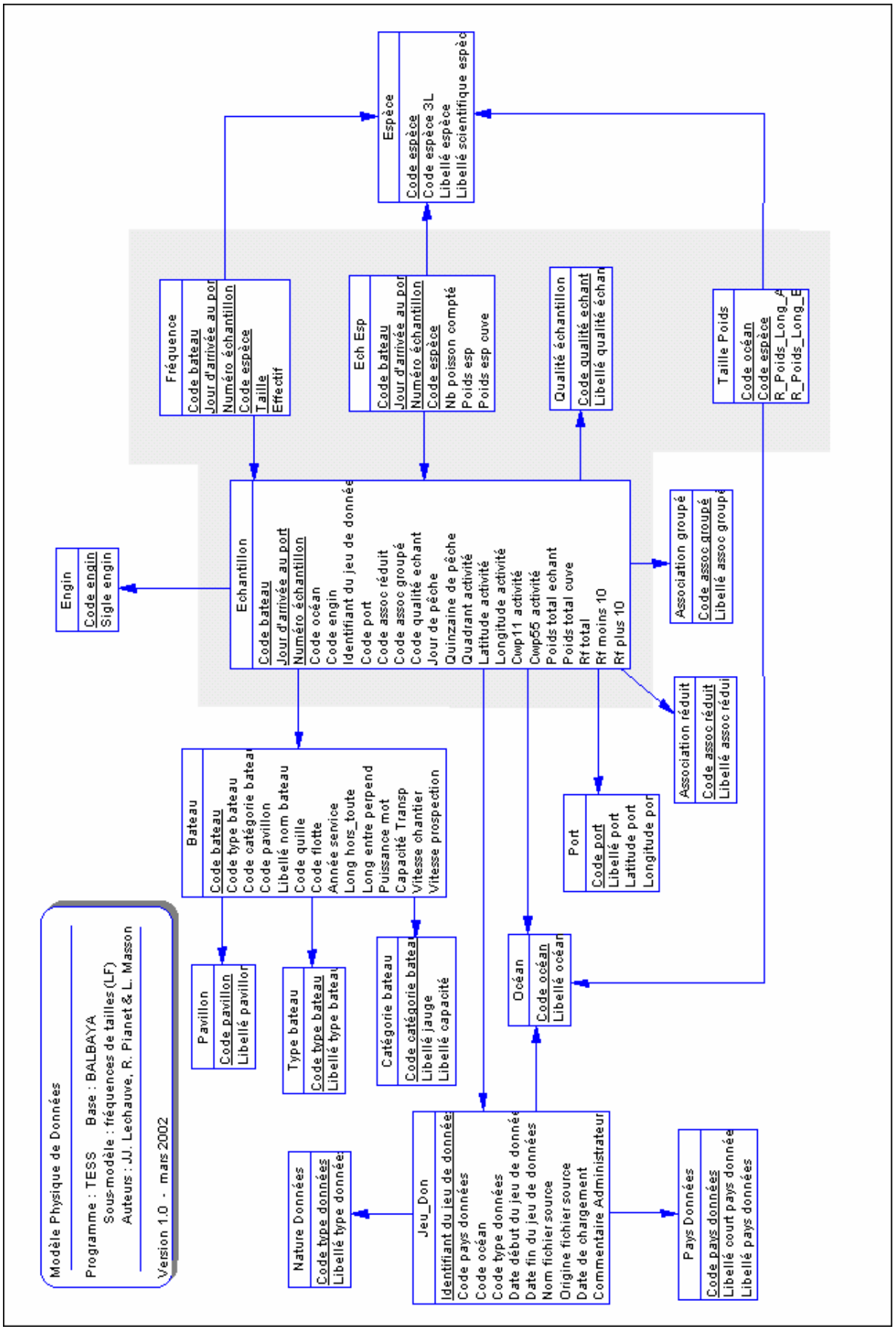
Les modèles logiques (captures et mensurations) qui suivent ont été réalisés grâce au logiciel **Power AMC** de **Sybase**.

Modèles Logiques de Données

Partie captures



Partie Mensurations :



Rôle des différentes tables

Voyons le rôle des différentes tables du modèle logique :

- BATEAU : Contient des informations sur les bateaux de pêche en activité ou ayant été en activité. Cette table sera modifiée pour tenir compte de l'évolution de la flottille.
- PAVILLON : Pays dont dépendent les bateaux. (*)
- TYPE_BATEAU : Types d'unités de pêche.
- CAT_BATEAU : Capacités de transport des bateaux.
- ENGIN : Types d'engin (canneur ou senneur).
- CAT_POIDS : Catégories de poids de l'espèce composant une capture (trois possibilités, p < 10kg, 10kg < p < 30kg, ou p > 30kg).
- ESPECE : Différentes espèces de thons tropicaux.
- CAPTURE : Poids extrapolé de poisson d'une activité par espèce.
- LONPOI : Relation longueur-poids (tient compte de l'espèce et de l'océan).
- ECH_ESP : Composition spécifique de l'échantillon.
- FREQT : Fréquences de taille extrapolées.
- A_PAYS_D : Pays producteurs des jeux de données. (*)
- A_TYP_D : Types de jeux de données (captures – efforts, ou fréquences de tailles).
- ASSOC_G : Types de banc (banc libre ou banc sous objet).
- ASSOC_R : Codes indiquant comment le banc de thons a été repéré.
- PORT : Ports à partir desquels les bateaux partent ou arrivent.
- QUAL_ECH : Codes de qualité des échantillons.
- QUAL_ACT : Codes de qualité des activités.
- ZONE_ET : Zones géographiques déterminées par des polygones. Les lettres ET proviennent du nom d'un groupe de travail : "Analyse de la stratégie d'**E**chantillonnage multi-spécifique des **T**hons tropicaux"
- ZONE_ET_POINT : Coordonnées des points délimitant les différentes zones ET.

(*) On voit apparaître deux tables concernant les pays dans ce modèle, ce qui pourrait laisser à penser qu'il y'a une redondance d'information et qu'une seule table aurait suffit. Cependant, les pays contenus dans la table A_PAYS_D sont parfois une association de plusieurs pays.

Noms des colonnes

Voici quelques éléments concernant les noms des différentes colonnes du modèle logique, notamment au niveau des préfixes :

- C : indique un code.
- L : Indique un libellé.
- D : Indique une date.
- V : Indique une valeur, un nombre saisi, qui peut varier contrairement aux codes qui ne peuvent prendre qu'un certain nombre de valeurs.
- ID : Indique un identifiant, il s'agit d'un numéro généré par Oracle (séquence)

Organisation de la base

Comme on l'a précisé dans la présentation du contexte scientifique, cette base de données est prévue pour accueillir des données à différentes échelles. Nous avons décidé de créer un schéma pour chacune de ces classes de données.

Le sujet de ma contribution concerne uniquement les données de pêche à grande échelle. Le schéma nommé ***balbaya*** leur est consacré. Les autres schémas seront introduits dans un proche avenir.

Avant de créer la base de données et le schéma ***balbaya***, plusieurs paramètres doivent être évalués :

- Taille à allouer pour chaque table et chaque index,
- Taille des *rollback segments*,
- Répartition logique des tables et index dans les tablespaces,
- Répartition physique des tablespaces sur les disques.

Pré-requis

Pour calculer les tailles, des formules sont données dans le manuel d'Oracle 8, « Administrator's guide ». Avant d'exposer les méthodes de calculs, présentons quelques notions indispensables.

Taille d'un bloc de données

Un des premiers éléments requis est la taille d'un bloc de données Oracle (*data block*). Par défaut celle-ci est de 2 Ko, on peut l'augmenter lorsqu'on bénéficie d'un bon serveur. Nous avons décidé de prendre des blocs de 8 Ko.

Clause STORAGE

Lors de la création d'un tablespace, d'une table ou d'un index, on spécifie une clause **STORAGE** qui indique à Oracle comment gérer les *segments* et les *extents* qui vont contenir les données.

Les différents paramètres définis dans cette clause sont :

- INITIAL : Taille de la première extension allouée (*extent*) lorsqu'un segment est créé (défaut = 5 blocs). Exprimée en octets.
- NEXT : Taille du deuxième *extent* alloué au segment (défaut =5 blocs). Exprimée en octets.
- MINEXTENTS : Nombre d'*extents* alloués à la création du segment (défaut =1).
- MAXEXTENTS : Nombre total d'*extents* pouvant être alloués au segment (défaut : dépend de la taille du bloc de données).
- PCTINCREASE : Pourcentage d'accroissement de l'*extent* i+1 par rapport à l'*extent* i (i > 2).

Les calculs des tailles nous permettent de savoir quelles valeurs nous allons ensuite donner aux valeurs INITIAL et NEXT. INITIAL est prévu pour stocker 25 ans de données, et NEXT fera un dixième de la valeur d'INITIAL. Les valeurs de MINEXTENTS et de MAXEXTENTS seront à 1, PCTINCREASE sera à 0.

Paramètre INITRANS

Ce paramètre est à renseigner lors de la création d'index et de table. Chaque opération de mise à jour d'un bloc de données (transaction), nécessite une entrée de transaction dans le bloc. Ce paramètre assure un minimum de transactions concurrentes sur le même bloc, et évite l'allocation dynamique de nouvelles entrées de transaction.

On garde généralement la valeur par défaut de ce paramètre, qui est de 1 pour les tables et de 2 pour les index.

Paramètres PCTFREE et PCTUSED

Les deux paramètres PCTFREE et PCTUSED permettent de gérer l'espace libre dans les blocs de données. Lors de la création de tables, on spécifie ces deux paramètres. Pour les index, seul PCTFREE est utilisé.

- PCTFREE

Ce paramètre fixe le pourcentage d'espace gardé libre pour les **mises à jour** des lignes contenues dans un bloc. La valeur par défaut est 10, elle peut varier de 1 à 99. Cette valeur dépend essentiellement de la nature des données contenues dans la table Certaines données (établies par un traitement statistique) seront fixes, d'autres subiront dans le temps quelques modifications. Si la mise à jour d'une ligne entraîne une augmentation de sa taille, on prendra une valeur supérieure à 10. Sinon, une valeur inférieure à 10 est conseillée.

- PCTUSED

Ce paramètre est utilisé simultanément avec PCTFREE. Une fois que le pourcentage d'espace libre d'un bloc de données atteint la limite fixée par PCTFREE, le bloc est disponible uniquement pour des **mises à jour** des lignes existantes. Si ensuite le pourcentage d'espace utilisé descend en dessous de PCTUSED, alors des nouvelles lignes pourront y être **insérées**. Par défaut sa valeur est de 40.

Dans le cas de la base **ALIPA**, les mises à jour qui entraînent une augmentation de la taille de la ligne seront rares. Seules les tables BATEAU et A_JEU_D sont concernées.

La table BATEAU sera modifiée afin de tenir compte de l'évolution de la flottille. Cependant, ces modifications n'entraîneront pas une grande augmentation de la taille de la ligne, c'est pourquoi nous avons choisi un PCTFREE de 10 et un PCTUSED de 40.

La table A_JEU_D peut avoir des libellés qui augmentent considérablement la taille d'un enregistrement. Cette table comprend quatre « varchar2 » dont les tailles sommées font 2432 octets. Il est important de garder de l'espace pour les mises à jour. Son PCTFREE sera de 30 et son PCTUSED de 10.

Toutes les autres tables ont pour valeur : PCTFREE = 5 et PCTUSED = 60.

Tailles des tables et index

Les tables

Afin de mieux présenter les calculs de tailles, trois exemples numériques sont proposés en annexe (tables pavillon, a_jeu_d et freqt). Les différentes étapes sont :

- Calcul de la taille du bloc sans l'en-tête
  hsize = db_block_size – 86
  db_block_size : Taille d'un bloc de données Oracle

- Calcul de l'espace disponible par bloc
  availSpace = arrondi_supp (hsize * (1 – (PCTFREE / 100))) – 4

- Calcul de l'espace utilisé par ligne de données
  On applique l'algorithme suivant :
  pour chaque colonne de la table **faire**
  si (tailleColonne > 250)
  tailleColonne = tailleColonne + 3
  sinon
  tailleColonne = tailleColonne + 1
  fin si
  tailleColonne = tailleColonne + 3
  fin pour
  rowSpace = valeur_max (tailleColonne, 9) + 2

- Calculs du nombre de lignes pouvant être stockées dans un bloc de données
  nbLignes = arrondi_inf (availSpace / rowSpace)

Les index

De la même façon que pour les tables, les calculs de taille des index se font en plusieurs étapes qui sont :

- Calcul de la taille de l'en-tête du bloc
  header = fixedHeader + variableHeader
  FixedHeader = 113 octets
  variableHeader = 24 * INITRANS (2 par défaut pour les index)

- Calcul de l'espace disponible par bloc
  availSpace = (db_block_size – header) – ((db_block_size – header) * …
  … (PCTFREE / 100))

- Calcul de la somme des colonnes participant à l'index
  somCol = sizeCol1 + sizeCol2 + … + sizeColN

- Calcul de la taille de l'index
  indexSize = 8 + pCol + (2 * gCol) + somCol
  pCol : Nombre de colonnes <= 127 octets composant l'index.
  gCol : Nombre de colonnes > 127 octets composant l'index.

- Calcul du nombre de lignes stockées dans un bloc
  nbLignes = arrondi_inf (availSpace / indexSize)

On voit que le calcul des tailles peut être long, surtout lorsque les tables contiennent beaucoup de colonnes. Dans le but d'automatiser le processus de calcul, j'ai écrit un programme en C dont l'algorithme est fourni en annexe.

Les tableaux suivants montrent un extrait des résultats des calculs concernant les tables et les index :

Table	Initial	Next	Total (Mo)
capture	117190	11720	
activite	137695	13770	
freqt	380860	38086	
echant	6592	660	
espece			
zone_et	8	8	
zone_et_points	8	8	
Total	**655026**	**65642**	**703.78**

Table	Index	Initial	Next	Total (Mo)
capture	capture_pk	101594	10000	
	activite_r_capture_fk	80096	8000	
	espece_r_capture_fk	36958	3700	
	cat_poids_r_capture_fk	33865	3400	
activite	activite_pk	26699	2700	
	assoc_r_r_activite_fk	11289	1130	
	assoc_g_r_activite_fk	11289	1130	
	zone_et_r_activite_fk	11289	1130	
	ocean_r_activite_fk	11289	1130	
	a_jeu_d_r_activite_fk	15400	1540	
	port_r_activite_fk	12320	1240	
	engin_r_activite_fk	11289	1130	
	bateau_r_activite_fk	13350	1340	
zone_et_points	zone_et_points_pk	8	8	
	zone_et_r_zone_et_points_fk	8	8	
Total		**1220806**	**124500**	**1313.78**

Soit un total de 2017.56 Mo (1.97 Go)

Taille des *rollback segments*

Les segments d'annulation sont obligatoires dans toute base Oracle. Nous savons que les insertions de données dans notre base se feront sur une durée d'un an maximum. Nous pouvons calculer la taille maximale requise lors d'une insertion de données afin de déterminer la taille des *rollback segments* à utiliser.

D'après nos calculs on sait que :

- Echantillons : 1317 Mo pour 20 ans => 65.85 Mo pour un an.
- Activités : 679 Mo pour 20 ans => 33.95 Mo pour un an.

Nous avons décidé de prendre un gros rollback segment de 100 Mo qui sera activé (*online*) lors des insertions de données. Dans les autres cas, il sera désactivé (*offline*).

En plus de ce segment, quatre autres de 5 Mo chacun seront ajoutés à la base. On les nommera rbs1, rbs2, rbs3 et rbs4. Le dernier, celui de 100 Mo sera rbs5. Soit un total de 120 Mo pour les *rollback segments.*

Répartition logique

Nous avons décidé de créer trois tablespaces pour les tables, et trois autres pour les index. Ainsi, on sépare les données de captures/effort, de mensurations et les autres.

Voici une liste des tablespaces et des tables ou index qu'ils contiendront :

- tbs_balbaya_capt_d : activite, capture.
- tbs_balbaya_capt_i : activite_pk, capture_pk, + index clés étrangères
- tbs_balbaya_mens_d : freqt, echant, ech_esp.
- tbs_balbaya_mens_i : freqt_pk, echant_pk, ech_esp_pk, + index clés étrangères
- tbs_balbaya_thes_d : toutes les autres tables.
- tbs_balbaya_thes_i : tous les autres index.
- tbs_rbs : rollback segments rbs1, rbs2, rbs3, rbs4 et rbs5.

Répartition physique

Afin de savoir comment organiser nos tablespaces, nous avons rencontré Michèle Fichaut, une spécialiste des bases de données Oracle travaillant à Ifremer. D'après ses conseils, il est utile de placer les tables accédées simultanément sur des supports physiques différents. Ainsi les accès disques concurrents sont possibles, et on gagne du temps lors des opérations d'entrée/sortie.

D'autre part, d'après la documentation Oracle, il est conseillé de placer les fichiers de journalisation sur un autre disque que ceux qui contiennent les données. Les *redo-log files* étant accédés régulièrement, cela permettra de toujours accéder rapidement aux données contenues dans la base.

Les disques disponibles pour la base ALIPA sont :

- ora_wk1, capacité 1 Go,
- ora_wk4, capacité 4 Go,
- ora_wk20, capacité 20 Go.

Le tableau suivant illustre la répartition physique du schéma ***balbaya*** sur ces trois disques :

Disque	Rôle	Nom	Chemin complet
Ora_wk1	Fichier de contrôle	ctrl1	/alipa/ctrl1.dbf
	Fichiers redo logs	log1	/alipa/log1.dbf
		log2	/alipa/log2.dbf
		log3	/alipa/log3.dbf
		log4	/alipa/log4.dbf
Ora_wk4	Fichier de contrôle	ctrl2	/alipa/ctrl2.dbf
	Tablespace de données de mensurations	tbs_balbaya_mens_d	/alipa/balbaya/tbs_mens_d.dbf
	Tablespace de données de captures	tbs_balbaya_capt_d	/alipa/balbaya/tbs_capt_d.dbf
	Tablespace d'index du thesaurus	tbs_balbaya_thes_i	/alipa/balbaya/tbs_thes_i.dbf
Ora_wk20	Fichier de contrôle	ctrl3	/alipa/ctrl3.dbf
	Tablespace d'index de mensurations	tbs_balbaya_mens_i	/alipa/balbaya/tbs_mens_i.dbf
	Tablespace d'index de captures	tbs_balbaya_capt_i	/alipa/balbaya/tbs_capt_i.dbf
	Tablespace de données du thesaurus	tbs_balbaya_thes_d	/alipa/balbaya/tbs_thes_d.dbf
	Tablespace temporaire	tbs_balbaya_temp	/alipa/balbaya/tbs_temp.dbf
	Tablespace des rollback segments	tbs_rbs	/alipa/tbs_rbs.dbf

Note : On voit dans ce tableau que certains fichiers se trouvent dans le répertoire alipa (fichiers de contrôle, fichiers de journalisation, et *datafile* du *tablespace* des *rollback segments*). Ces fichiers n'ont pas été placés dans le répertoire balbaya, car ils seront aussi utilisés par les schémas futurs de la base **ALIPA**.

Utilisateurs

Avant de créer une base de données, il est important de savoir qui va y accéder, et dans quel but. Certaines données sont confidentielles, en conséquence il est important de définir les droits dont les utilisateurs bénéficieront.

Authentification

Oracle permet de se connecter à une base de données grâce à un login et un mot de passe. Lors de la création d'une base de données, l'administrateur crée des utilisateurs et leur donne certains droits (insertion, mise à jour, consultation).

Il est aussi possible de définir des rôles. Un rôle est une liste d'autorisations (accès aux données et privilèges systèmes) qui pourra être attribué à des utilisateurs et/ou à d'autres rôles.

Utilisateurs existants

Lors de la création d'une base de données, plusieurs utilisateurs sont créés, parmi lesquels :

- SYS

Il dispose de tous les privilèges système. Toutes les tables et les vues du dictionnaire sont créées dans son schéma, elles jouent un rôle très important dans le fonctionnement d'Oracle. Afin de maintenir l'intégrité du dictionnaire de données, seul Oracle en manipule les données.

- SYSTEM

Lui aussi bénéficie de tous les privilèges système. C'est l'administrateur par défaut de chaque nouvelle base.

Les utilisateurs SYS et SYSTEM bénéficient du rôle prédéfini DBA, qui leur donne tous les privilèges système. Ce rôle permet de faire toutes les modifications possibles sur la base de données. Cependant, il ne permet ni de démarrer, ni d'arrêter la base, pour cela il existe d'autres rôles prédéfinis qui sont SYSDBA et SYSOPER.

SYSOPER permet principalement le démarrage et l'arrêt d'une base de données. SYSDBA a les mêmes privilèges que SYSOPER, ainsi que tous les privilèges systèmes. Lorsque l'on se connecte en tant que SYSDBA, on se trouve dans le schéma de SYS.

Une autre alternative est d'utiliser le compte INTERNAL. Ce compte permet de se connecter en tant qu'administrateur sans avoir à donner de mot de passe si on fait partie du groupe d'utilisateurs Unix irddba (défini lors de l'installation d'Oracle). Cependant, il est préférable de créer un autre compte administrateur après la création de la base.

Lorsque l'on crée une nouvelle base de données, il est impératif de modifier les mots de passe des utilisateurs SYS et SYSTEM, et il est fortement conseillé de créer un nouvel administrateur.

Fichier de mots de passe

Le fait d'utiliser le rôle **SYSDBA** pour l'utilisateur ***balbaya*** requiert un système d'authentification particulier. Il est possible d'utiliser un fichier de mots de passe, ou de sous-traiter la phase d'identification au système d'exploitation. Nous avons choisi d'utiliser un fichier de mots de passe.

Ce fichier a pour but de déclarer les utilisateurs qui ont été autorisés à de connecter en tant que **SYSOPER** ou **SYSDBA**. Pour créer ce fichier de mot de passe, il faut utiliser la commande *orapwd*.

La syntaxe est la suivante :
orapwd file=<filename> password=<password> entries=<users>

On remplace filename par le nom du fichier de mots de passe à créer, password par le mot de passe administrateur de la base, et entries par le nombre d'utilisateurs autorisés à se connecter en tant que SYSDBA ou SYSOPER. L'utilisateur internal se voit affecté de <password> comme nouveau mot de passe.

Cette commande écrit dans des répertoires où seul l'administrateur possède le droit d'écriture, en conséquence elle ne peut être exécutée que dans une session de cet utilisateur privilégié. Pour mémoire, l'administrateur est celui qui a installé le logiciel Oracle et créé la base de données.

Administrateur

Sous Oracle, l'utilisateur qui crée un schéma en est le propriétaire, il est l'administrateur de son propre schéma. C'est pourquoi nous avons décidé de créer un utilisateur ***balbaya***, c'est lui qui créera le schéma de la base de données **ALIPA**. Il en sera donc l'administrateur. Afin d'avoir tous les droits, il héritera des rôle **SYSDBA** et **DBA**.

Administrateur de données

L'administrateur de données, chargé de l'alimentation de la base ainsi que de sa mise à jour, doit avoir le droit d'insérer et de modifier des données dans le schéma ***balbaya***, c'est lui qui assure le suivi des collections de données au sein de la base. On créera un rôle balbaya_admin, et un utilisateur balbaya_adm qui héritera du rôle dont on vient de parler.

Lecteurs

Les lecteurs sont les utilisateurs qui n'ont aucun droit de modification des données de la base. Seuls les ordres SQL *select* leur seront autorisés sur certaines tables. Deux types d'utilisateurs seront des « lecteurs ».

Dans un premier cas, une application chargée de faire un traitement sur les données présentes dans la base effectuera une connexion en tant que balbaya_app. D'autre part, un utilisateur pourra se connecter à la base de données afin d'effectuer des commandes **select** uniquement, il s'identifiera en tant que balbaya_lect.
Le rôle balbaya_lecteur sera attribué à ces deux utilisateurs.

Réalisation

Une fois les calculs de tailles et les scripts de création des objets de la base terminés, j'ai procédé à la création de la base proprement dite. L'étape suivante a tout naturellement consisté à charger un jeu d'essai. Enfin, je me suis consacré à la conception et à la réalisation d'une application cliente.

Création de la base ALIPA

Fichier d'initialisation

Avant de créer la base, certaines variables d'environnement doivent être définies. Entre autres, la variable ORACLE_SID doit obligatoirement être positionnée. Elle indique le nom de la base (aussi appelé SID de la base). Dans notre cas il s'agira de **ALIPA**, la commande Unix (C-Shell) est : *setenv ORACLE_SID ALIPA*

Pour démarrer une instance Oracle et monter une base de données, il faut spécifier un fichier d'initialisation. Celui-ci contient divers renseignements sans lesquels la base ne pourrait être montée. Généralement le fichier s'appelle initSID.ora, dans notre cas ce sera initALIPA.ora.

Voici une liste de paramètres présents dans ce fichier, avec les valeurs que j'ai positionnées :

- db_name : Nom de la base de données (doit être identique à ORACLE_SID), valeur : ALIPA.
- db_block_size : Taille d'un bloc de données Oracle, valeur : 8192 (8 Ko).
- control_files : Nom et emplacement du ou des fichier(s) de contrôle, valeur :
- (/export/ora_wk1/alipa/ctrl1.dbf,
- /export/ora_wk4/alipa/ctrl2.dbf,
- /export/ora_wk20/alipa/ctrl3.dbf).
- db_files : Nombre de fichiers ouverts simultanément sur cette base de données, valeur : 80.
- nls_language : Langue par défaut de la base de données, utilisé pour les messages, valeur : french.
- nls_date_language : Langue par défaut pour les noms de jours, de mois, abréviations de date (AM, PM…), valeur : french.
- nls_date_format : Format des dates par défaut, valeur : "DD/MM/YYYY".
- db_file_multiblock_read_count : Nombre maximum de blocs de données lus en une opération d'entrée/sortie, valeur : 8.
- db_block_buffers : Nombre de blocs de données disponibles dans la mémoire cache, valeur : 100.
- rollback_segments : Nom des segments d'annulation à utiliser avec cette base de données. Valeur : (rbs1, rbs2, rbs3, rbs4).
- remote_login_passwordfile : Indique à Oracle le type d'authentification utilisé. Nous utilisons un fichier de mots de passe unique par base de données. Valeur : exclusive.
- etc…

Script de création d' ALIPA

La création de la base est réalisée avec Server Manager, un utilitaire d'Oracle qui permet d'effectuer toutes les tâches d'administration grâce à des lignes de commandes SQL.

Une fois les variables d'environnement renseignées, et le fichier d'initialisation constitué, on crée la base grâce aux commandes suivantes (ce qui a été introduit au clavier apparaît en gras) :

```
galaad> svrmgrl (1)

Oracle Server Manager Release 3.0.5.0.0 - Production

(c) Copyright 1997, Oracle Corporation.  All Rights Reserved.

Oracle8 Enterprise Edition Release 8.0.5.0.0 - Production
PL/SQL Release 8.0.5.0.0 - Production

SVRMGR> spool '/export/ora_wk20/creation_base_alipa.log' (2)
SVRMGR> connect internal (3)
Connected.
SVRMGR> startup nomount pfile=./initALIPA.ora (4)
ORACLE instance started.
Total System Global Area       5185040 bytes
Fixed Size                       48656 bytes
Variable Size                  4235264 bytes
Database Buffers                819200 bytes
Redo Buffers                     81920 bytes
SVRMGR> create database ALIPA
      datafile '/export/ora_wk20/alipa/balbaya/tbs_system.dbf' size 60M reuse
      logfile '/export/ora_wk1/alipa/log1.dbf' size 4M,
            '/export/ora_wk1/alipa/log2.dbf' size 4M,
            '/export/ora_wk1/alipa/log3.dbf' size 4M,
            '/export/ora_wk1/alipa/log4.dbf' size 4M; (5)
Statement processed.
SVRMGR> @$ORACLE_HOME/rdbms/admin/catalog (6)
Statement processed.
...
SVRMGR> @$ORACLE_HOME/rdbms/admin/cataudit (6)
Statement processed.
...
SVRMGR> @$ORACLE_HOME/rdbms/admin/catproc (6)
Statement processed.
...
SVRMGR> spool off (7)
SVRMGR> shutdown (8)
Database closed.
Database dismounted.
ORACLE instance shut down.
SVRMGR>
```

svrmgrl (1)
Cette commande lance Server Manager

spool '/export/ora_wk20/creation_base_alipa.log' (2)
Cette commande crée un fichier appelé 'creation_base_alipa.log' dans lequel seront écrits tous les messages de sortie d'Oracle.

connect internal (3)
Connexion à Oracle sous le nom 'internal'. Ce nom permet de se connecter en tant qu'administrateur sans avoir à donner de mot de passe, si on fait partie du groupe d'utilisateurs irddba (définit lors de l'installation d'oracle). Il est utile pendant la création de la base, mais ensuite, il est préférable de créer un autre compte administrateur.

startup nomount pfile=./initALIPA.ora (4)
Démarrage d'une instance Oracle. L'option 'nomount' indique que l'on ne va pas monter de base de données. On spécifie ici le fichier d'initialisation qui va fixer les différents paramètres que l'on a vus précédemment. On voit d'après le message d'Oracle, que des zones mémoires ont été allouées (notamment la SGA, qui est ici de 5 185 040 octets, soit environ 5 Mo).

create database ALIPA... (5)
Création de la base de données ALIPA proprement dite. Le premier fichier spécifié (datafile), est le fichier correspondant au tablespace SYSTEM. Ensuite on indique le nom des différents fichiers de journalisation qui font chacun 4 Mo.

@$ORACLE_HOME/rdbms/admin/cat* (6)
Les trois lignes suivantes appellent des scripts qui créent les tables et les vues du dictionnaire de données. Les scripts contiennent beaucoup de lignes, c'est pourquoi tous les messages de sortie ne sont pas affichés ici.

spool off (7)
Fin de l'enregistrement des messages Oracle dans le fichier de sortie.

shutdown (8)
On ferme la base, on la démonte, puis on arrête l'instance d'Oracle.

A présent la base est créée, le schéma balbaya peut être introduit.

Script de création de *balbaya*

Ici aussi on utilise le server manager, ainsi que le compte internal. Voici le script utilisé :

galaad> **svrmgrl**

Oracle Server Manager Release 3.0.5.0.0 - Production

Oracle8 Enterprise Edition Release 8.0.5.0.0 - Production
PL/SQL Release 8.0.5.0.0 - Production

SVRMGR> **spool '/export/ora_wk20/alipa/creation_schema_balbaya.log'**
SVRMGR> **connect internal**
Connected.
SVRMGR> **startup open pfile=./initALIPA.ora** (1)
ORACLE instance started.
Total System Global Area 5185040 bytes
Fixed Size 48656 bytes
Variable Size 4235264 bytes
Database Buffers 819200 bytes
Redo Buffers 81920 bytes
Database mounted.
Database opened.
SVRMGR> **create user balbaya identified by XXXX;**
SVRMGR> **grant dba to balbaya with admin option;**
SVRMGR> **grant sysdba to balbaya;** (2)
Statement processed.
SVRMGR> **@./creer_rbs.sql** (3)
Statement processed.
...
SVRMGR> **@./creer_tbs.sql** (4)
Statement processed.
...
SVRMGR> **alter user balbaya temporary tablespace tbs_balbaya_temp;**
SVRMGR> **alter user balbaya quota unlimited on tbs_balbaya_temp;** (5)
Statement processed.
SVRMGR> **alter user balbaya quota unlimited on tbs_balbaya_temp quota unlimited on tbs_balbaya_mens_d quota unlimited on tbs_balbaya_mens_i quota unlimited on tbs_balbaya_capt_d quota unlimited on tbs_balbaya_capt_i quota unlimited on tbs_balbaya_thes_d quota unlimited on tbs_balbaya_thes_i;** (6)
Statement processed.
SVRMGR> **@./creer_tables_index_balbaya.sql** (7)
Statement processed.
...
SVRMGR> **create sequence s_id_jeu_d minvalue 1 nomaxvalue;** (8)
Statement processed.

SVRMGR> **@./creer_users_balbaya.sql** (9)
Statement processed.
...
SVRMGR> **connect balbaya/XXXX as sysdba**
Connected.
SVRMGR> **shutdown** (10)
Database closed.
Database dismounted.
ORACLE instance shut down.

Commentaires

startup open pfile=./initALIPA.ora (1)

Contrairement à la commande startup utilisée lors de la création d'**ALIPA**, ici on demande à ce que la base de données soit montée (mot clé **open**). Le message d'Oracle nous dit bien que la base est montée et ouverte.

create user balbaya identified by XXXX;
grant dba to balbaya with admin option;
grant sysdba to balbaya; (2)

Création du user balbaya, et affectation d'un mot de passe. On lui attribue ensuite les rôles DBA et SYSDBA.

@./creer_rbs.sql (3)

Script qui crée les rollback segments. (Extrait en annexe)

@./creer_tbs.sql (4)

Appel d'un script qui crée les tablespaces de la base. (Extrait en annexe)

alter user balbaya temporary tablespace tbs_balbaya_temp;
alter user balbaya quota unlimited on tbs_balbaya_temp; (5)

Attribution du tablespace tbs_balbaya_temp en tant que tablespace temporaire de balbaya. Il n'aura pas de limite de taille sur ce tablespace (quota unlimited).

alter user balbaya quota unlimited on tbs_balbaya_temp quota unlimited on ... (6)

De la même façon que pour le tablespace temporaire, balbaya n'aura aucune restriction de taille sur les tablespaces de la base.

@./creer_tables_index_balbaya.sql (7)

Appel du script de création des tables et des index du schéma balbaya. En exemple, cet extrait présente la définition de la table a_jeu_d et des index associés à ses clés étrangères.

```
/* ================================================================ */
/* Création de la table : A_JEU_D                                   */
/* ================================================================ */
create table A_JEU_D  (
   ID_JEU_D          NUMBER(6)          not null,
   C_PAYS_D          NUMBER(3)          not null,
   C_OCEA            NUMBER(2)          not null,
   C_TYP_D           NUMBER(2)          not null,
   D_DEB_D           DATE               not null,
   D_FIN_D           DATE               not null,
   L_FIC_D           VARCHAR2(128)      not null,
   L_ORG_FIC_D       VARCHAR2(256)      not null,
   D_DISPO           DATE               not null,
   L_COM_INS         VARCHAR2(1024)     not null,
   L_COM_SUPPR       VARCHAR2(1024),
   D_SUPPR           DATE,
   constraint PK_A_JEU_D primary key (ID_JEU_D),
   constraint FK_A_PAYS_D_R_A_JEU_D foreign key (C_PAYS_D)
        references A_PAYS_D (C_PAYS_D),
   constraint FK_OCEAN_R_A_JEU_D foreign key (C_OCEA)
        references OCEAN (C_OCEA),
   constraint FK_A_TYP_D_R_A_JEU_D foreign key (C_TYP_D)
        references A_TYP_D (C_TYP_D)
)
pctfree 30
pctused 10
storage
(
    initial 2438K
    next 244K
    pctincrease 0
)
tablespace TBS_BALBAYA_THES_D
/
* ================================================================ */
/* Création de l'index A_PAYS_D_R_A_JEU_D_FK                        */
/* ================================================================ */
create  index A_PAYS_D_R_A_JEU_D_FK on A_JEU_D (
   C_PAYS_D ASC
)
pctfree 0
storage
(
    initial 16K
    next 8K
    pctincrease 0
)
tablespace TBS_BALBAYA_THES_I
/
```

```
/*============================================================*/
/* Création de l'index : OCEAN_R_A_JEU_D_FK                   */
/*============================================================*/
create  index OCEAN_R_A_JEU_D_FK on A_JEU_D (
  C_OCEA ASC
)
pctfree 0
storage
(
   initial 16K
   next 8K
   pctincrease 0
)
tablespace TBS_BALBAYA_THES_I
/
/* ============================================================ */
/* Création de l'index : A_TYP_D_R_A_JEU_D_FK                   */
/* ============================================================ */
create  index A_TYP_D_R_A_JEU_D_FK on A_JEU_D (
  C_TYP_D ASC
)
pctfree 0
storage
(
   initial 16K
   next 8K
   pctincrease 0
)
tablespace TBS_BALBAYA_THES_I
/
```

create sequence s_id_jeu_d minvalue 1 nomaxvalue; (8)

La clé de la table a_jeu_d est générée grâce à une séquence Oracle. Il faut donc la créer au moment de la création de la base. Elle démarrera avec une valeur de 1, et s'incrémentera de 1 lorsqu'on appellera **s_id_jeu_d.nextvalue**.

@./creer_users_balbaya.sql (9)

Script de création des rôles balbaya_admin, et balbaya_lecteurs. Le script crée aussi les utilisateurs balbaya_adm, balbaya_lect et balbaya_app. (Extrait en annexe)

connect balbaya/XXXX as sysdba
shutdown (10)

On se connecte en tant que SYSDBA afin de pouvoir faire le shutdown, la base ALIPA et le schéma BALBAYA sont maintenant créés.

Chargement des données

J'ai utilisé trois méthodes différentes pour charger des données dans la base. La première consiste à écrire simplement les ordres SQL **insert** dans un fichier, puis à exécuter le script ainsi constitué. Cela est uniquement applicable au chargement de petites tables. Ensuite, pour les tables similaires à celles utilisées dans AVDTH, j'ai utilisé l'utilitaire Oracle **SQL Loader**. Enfin, pour l'introduction des jeux de données proprement dit (activités et captures), un traitement préliminaire s'imposait, j'ai écrit un programme en C.

Petites tables

Afin d'insérer les données dans les petites tables telles que la table engin ou la table mois, j'ai saisi les commandes insert a la main, ce qui donne pour ces deux exemples :

- Fichier engin.sql:

```
insert into engin values (1,'PS','Senneur');
insert into engin values (2,'BB','Canneur');
```

- Fichier mois.sql

```
insert into mois values (1,'Janv','Janvier');
insert into mois values (2,'Fevr','Février');
insert into mois values (3,'Mars','Mars');
insert into mois values (4,'Avri','Avril');
insert into mois values (5,'Mai','Mai');
etc...
```

Ensuite, le chargement est fait en lançant Server Manager, puis en exécutant les scripts :

```
svrmgrl
@./engin.sql
@./mois.sql
```

J'ai procédé ainsi pour les tables ocean, cat_poids, assoc_g, mois, zone_et, engin, a_typ_d, qual_ech, a_pays_d, et assoc_r.

SQL Loader

Certaines tables du schéma ***balbaya*** sont identiques à celles de AVDTH, nous avons récupéré des fichiers ascii contenant les valeurs des colonnes séparées par des tabulations. L'utilitaire SQL Loader accepte des fichiers de ce genre en entrée, accompagné d'un fichier de contrôle.

Le fichier de contrôle spécifie à SQL Loader comment traiter le fichier de données qu'il reçoit en entrée. Il définit le nom du fichier de données, la description de la table qui va recevoir des données, quel est le caractère qui sépare deux valeurs, etc…

Voici le fichier type_bateau.ctl, utilisé pour charger la table type_bateau.

```
LOAD DATA
INFILE 'type_bateau.txt'
INTO table TYPE_BATEAU FIELDS TERMINATED BY X'09' (C_TYP_B, L_TYP_B)
```

L'instruction LOAD DATA, indique le début du fichier de contrôle.
Ensuite on spécifie le nom du fichier de données.
Enfin, on donne le nom de la table, des différentes colonnes, et on indique que les caractères sont séparés par le caractère X'09' (tabulation).

Fichier type_bateaux.txt correspondant :

```
1	Canne / glacier
2	Canne / congélateur
3	Mixte
4	Senneur avec appât
5	Senneur sans appât
6	Grand senneur
7	Palangrier
8	Bateau Mère
9	Bateau Usine
10	Supply
```

Afin de charger ces données, il faut qu'une instance Oracle soit active, et que la base de données soit montée. On frappe au clavier les commandes suivantes :

```
galaad> svrmgrl
SVRMGR> startup open pfile=./initALIPA.ora
SVRMGR> exit
galaad> sqlldr userid=balbaya/XXXX control=type_bateau.ctl

SQL*Loader: Release 8.0.5.0.0 - Production on Thu Mar 14 11:6:2 2002

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Commit point reached - logical record count 10
galaad>
```

On voit que les dix valeurs ont été insérées dans la table, grâce au message « logical record count 10 ».

SQL Loader génère le fichier type_bateau.log décrivant comment s'est passée l'opération. Ainsi on sait si les données ont été correctement insérées, et si ce n'est pas le cas, les enregistrements écartés sont archivés dans un fichier nommé type_bateau.bad.

J'ai procédé ainsi pour les tables type_bateau, cat_bateau, bateau, espece, lonpoi, pavillon, port.

Note: Il faut s'assurer que la table type_bateau est vide avant d'y insérer des données avec SQL Loader, sinon un message d'erreur est retourné.

Programme C

Afin de charger les tables activités et captures, on récupère des fichiers de données ASCII. Ces fichiers contiennent les résultats des traitements statistiques effectués sur les collections de données élaborées sous AVDTH.

Le programme que j'ai réalisé lit un fichier séquentiel. A partir de celui-ci, il crée en sortie trois fichiers contenant chacun des commandes SQL insert. Ces trois fichiers concernent les tables activite, capture et a_jeu_d.

Chaque ligne du fichier d'entrée contient des informations concernant une activité, et éventuellement des captures. On doit garder en mémoire qu'un jeu de données concerne un an de données maximum.

Voici l'algorithme du programme :

Début
Déclaration des variables
Vérification des paramètres d'entrée
Ouverture du fichier de données en lecture
Ouverture des fichiers de sortie (création ou remplacement)
Tant que l'on n'est pas à la fin du fichier, **faire**
Lecture d'une ligne de données, stockage dans des variables
Mise en forme des dates de débarquement et d'activité vers un format SQL
Conversion des latitudes et longitudes (degrés, minutes vers degrés, centièmes)
Ecriture d'un insert de la table activite **(*)**
Si on vient de passer à une nouvelle année
Ecriture d'un insert de la table a_jeu_d **(*)**
Fin si
Si le nombre de calée(s) positive(s) est supérieur à 0
Pour chaque espèce **faire**
Si prise supérieure à 0
Ecriture d'un insert de la table capture **(*)**
Fin si
Fin pour
Fin si
Fin tant que
Ecriture d'un insert de la table a_jeu_d
Fermeture des fichiers
Fin

(*) Lors de la création des lignes d'insert, les valeurs nulles du fichier de données sont devenues la valeur SQL « null ».

Une fois le programme compilé, on le lance avec plusieurs paramètres qui sont :

- Nom du fichier de données,
- Premier numéro de séquence à utiliser pour les jeux de données,
- Premier numéro à utiliser en tant que numéro d'activité,
- Nom du fichier de sortie activités,
- Nom du fichier de sortie a_jeu_d,
- Nom du fichier de sortie captures.

J'ai pu faire un test avec un fichier d'entrée contenant les activités de pêche du 1er janvier 1991 au 31 décembre 1999. Voici les différentes commandes à saisir :

```
galaad > cc activites.c –o activites
galaad > activites data.txt 1 1 act.sql jeu.sql capt.sql
```

Le fichier d'entrée comporte 55440 lignes. Sachant qu'à partir d'une ligne, on génère un enregistrement dans la table activite, le fichier de sortie act.sql contient 55440 lignes lui aussi.

Afin de vérifier combien de lignes ont été générées, on utilise la commande wc d'Unix, en tapant :

```
galaad > wc –l act.sql
  55440 capt.sql
galaad > wc –l capt.sql
  46019 capt.sql
galaad > wc –l jeu.sql
      9 jeu.sql
```

Avant de tenter l'insertion de ces données dans la base, il faut prendre en compte la taille du rollback segment qui va être utilisé lors de cette transaction.

Taille d'un enregistrement de la table activite : 141 octets
Taille des index générés : 121 octets
Taille des 55440 lignes : 14525280 octets, soit environ 13.9Mo.

Taille d'un enregistrement de la table capture : 40 octets
Taille des index générés : 82 octets
Taille des 46019 lignes : 5614318 octets, soit environ 5.4 Mo.

Les rollback segments rbs1, rbs2, rbs3 et rbs4 font chacun 5 Mo, ils ne peuvent donc pas assurer la transaction des activites , il faut donc utiliser rbs5 qui fait 100 Mo. Voici les commandes à exécuter :

```
galaad > svrmgrl
SVRMGR> connect balbaya/XXXX
SVRMGR> alter rollback segment rbs5 online;
SVRMGR> alter rollback segment rbs1 offline;
SVRMGR> alter rollback segment rbs2 offline;
SVRMGR> alter rollback segment rbs3 offline;
SVRMGR> alter rollback segment rbs4 offline;
SVRMGR> @./jeu.sql
SVRMGR> @./act.sql
SVRMGR> @./capt.sql
```

Vérification du bon déroulement des insertions :

```
SVRMGR> select count(*) from a_jeu_d;
COUNT(*)
----------
         9
1 row selected.
```

```
SVRMGR> select count(*) from activite;
COUNT(*)
----------
  55440
1 row selected.
SVRMGR> select count(*) from capture;
COUNT(*)
----------
  46019
1 row selected.
```

Les insertions se sont bien déroulées, la base est créée, et « remplie », on peut maintenant l'utiliser.

Remarque

Par souci d'efficacité, l'introduction du jeu d'essai (9 années) a été réalisée en une seule opération. Les enregistrements concernant les jeux de données ont été insérés manuellement (clause insert). Il va de soit qu'en phase d'exploitation, l'administrateur de données traitera des collections d'au plus une année. A l'issue de la phase de mise à jour de la table A_JEU_D, soit manuelle, soit au travers de l'application cliente PHP, l'administrateur de données recueillera la valeur de l'identifiant du jeu à ajouter à la base et le communiquera au programme de préparation des données.

Interface PHP

Nous avons décidé de créer une interface en PHP afin de gérer la table a_jeu_d. L'interface doit être capable d'afficher les jeux de données disponibles dans la base, de les modifier et d'en ajouter. Cette application permettra à l'administrateur de données (balbaya_adm) de gérer simplement les collections de données. Il sera en effet plus aisé de saisir les paramètres dans des zones de textes, plutôt que de taper les commandes *insert* à la main.

Présentation de PHP

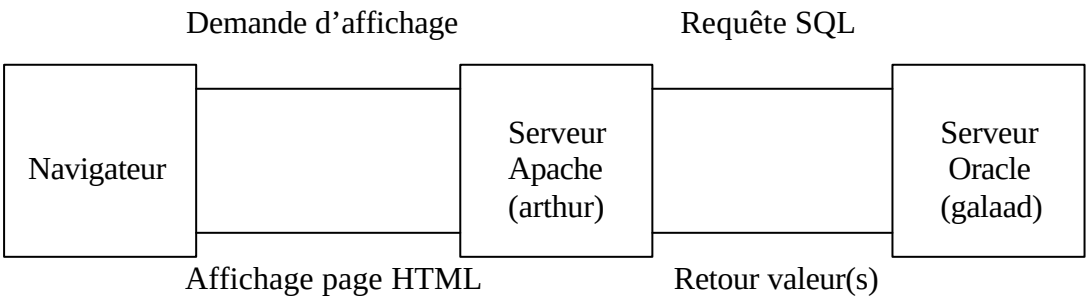
Le langage PHP est un langage interprété (langage de script), exécuté du côté serveur. Il permet de réaliser des pages web dynamiques. Sa syntaxe provient des langages C, Perl, et Java. Ses principaux atouts sont :

- Sa gratuité,
- La simplicité d'écriture des scripts,
- La possibilité d'inclure le script PHP au sein d'une page HTML,
- La simplicité d'implémentation avec une base de données,
- Son intégration au sein de nombreux serveurs web.

Ce langage a été inventé par Rasmus Lerdorf. Son premier script fut créé afin de savoir combien de personnes consultaient son CV sur son site. De nombreux internautes furent intéressés par le programme, ce qui l'amena à créer un nouveau langage baptisé PHP v1.0 (Personal Home Page) en 1995. Après le succès de la deuxième version sortie en été 1995, Zeev Suraski et Andi Gurmans se sont joints à lui pour créer PHP 3 en 1997. Aujourd'hui, PHP (PHP : HyperText Preprocessor) en est à sa quatrième version et la combinaison PHP – Apache est très largement répandue.

Architecture matérielle

Voici l'architecture mise en œuvre (dite architecture 3 tiers) :



Chacune des trois machines représente un « tiers », d'où l'appellation. Avec ce type d'architecture, seules les ressources des serveurs sont sollicitées. Le client se contente uniquement de l'affichage. Grâce à cette architecture, l'utilisation d'une application ne nécessite aucune connaissance en SQL.

Module Oracle

Pour mener à bien nos objectifs, il est impératif que l'application PHP accède aux données de la base Oracle. Plusieurs modules peuvent être ajoutés au noyau PHP lors de son installation, dont un module Oracle. Afin de vérifier que ce module est bien installé, on peut écrire ce script :

```
<?
        phpinfo() ;
?>
```

Ce code affiche toutes les données relatives au module PHP installé. Lors de l'exécution de ce script dans un navigateur, j'ai pu voir ces lignes :

oracle

Oracle Support	Enabled
Oracle Version	8.0
Compile-time ORACLE_HOME	/export/logiciels/oracle/product/8.0.5
Libraries Used	

Ce qui signifie que l'on va pouvoir utiliser les fonctions PHP destinées à Oracle.

Connexion

Voici le premier script, permettant de tester si la connexion avec la base de données se passe correctement (des détails concernant la connexion à Oracle sont fournis en annexe).

```
<?
// On positionne la variable d'environnement TNS_ADMIN qui indique où se trouve
// le fichier tnsnames.ora. Ce fichier doit être accessible par la partie cliente (navigateur)
// pour permettre une connexion au serveur de base de données (galaad)
putenv ("TNS_ADMIN=/home/galaad/irdlmdba/http_doc/tnsadmin_ird");

// Connexion à la base (toutes les fonctions Oracle sont préfixées par "ora_")
// On ajoute "@ALIPA" a la fin du nom de login pour préciser à quelle base on veut se
// connecter
$cnx = Ora_Logon ("balbaya@ALIPA","XXXX");

// Si Ora_Logon retourne une valeur positive, la connexion est établie
if (! $cnx) {
        echo "Echec de connexion <BR> Ora_Error($cnx)<BR>";
        die () ;            // Fonction qui termine le script
}

// Si la connexion s'est bien passée, on affiche un message puis on ferme la connexion
else {
        echo "Base Balbaya - Connexion établie";
        Ora_logoff($cnx);
}
?>
```

Consultation de données

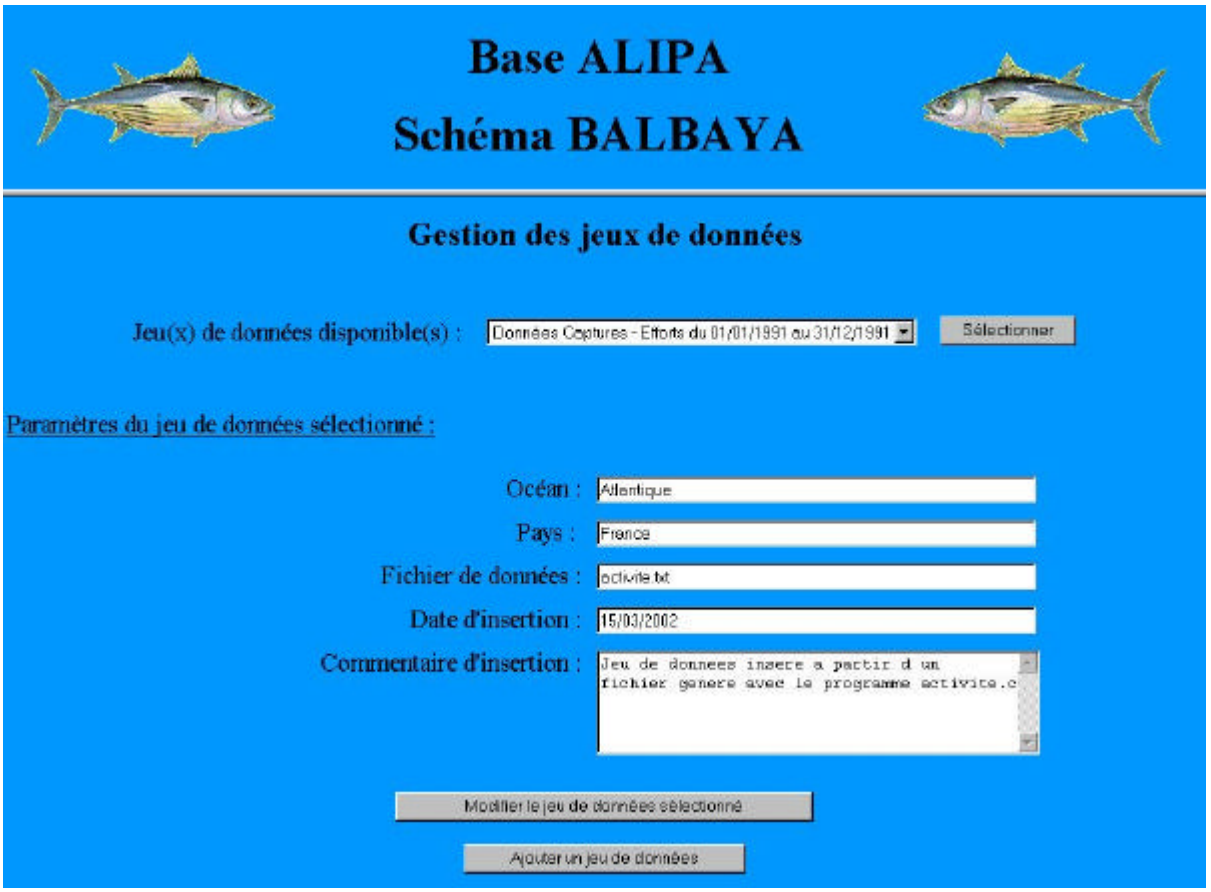
Le code que nous venons de voir a été placé dans une fonction nommée connecter. Cette fonction reçoit un login et un mot de passe en entrée. Si la connexion a lieu, la fonction retourne le handle de connexion au programme appelant, sinon, un message d'erreur est affiché et l'exécution du script s'arrête.

Le script ci-dessous affiche les données relatives aux jeux de données présents dans la base. Une liste déroulante permet de sélectionner facilement un jeu de données particulier.

```
<?
// Fichier affiche_jeu_d.php
// Affiche les jeux de données disponibles dans les zones de texte prévues à cet effet

// On inclut le fichier qui contient la fonction connecter
include ("fonctions.inc");

// Connexion à la base ALIPA en tant qu'administrateur de données
$cnx = connecter ("balbaya_adm@ALIPA", "XXXX");

 // La fonction connecter gère les erreurs de connexion, il n'est pas nécessaire de tester sa
// valeur de retour
$query = "SELECT * FROM a_jeu_d ORDER BY id_jeu_d ASC";
$cpt = 0;
// Lecture des enregistrements de la table a_jeu_d et stockage des colonnes dans des tableaux
if ($cursor = Ora_Do ($cnx, "SELECT * FROM a_jeu_d ORDER BY id_jeu_d ASC")) {
        do {
                // Stockage de toutes les colonnes dans des tableaux
                $id_jeu_d[$cpt] = ora_getcolumn ($cursor,0);
                $c_pays_d[$cpt] = ora_getcolumn ($cursor,1);
                $c_ocea[$cpt] = ora_getcolumn ($cursor,2);
                $c_typ_d[$cpt] = ora_getcolumn ($cursor,3);
                $d_deb_d[$cpt] = ora_getcolumn ($cursor,4);
                $d_fin_d[$cpt] = ora_getcolumn ($cursor,5);
                $l_fic_d[$cpt] = ora_getcolumn ($cursor,6);
                $l_org_fic_d[$cpt] = ora_getcolumn ($cursor,7);
                $d_dispo[$cpt] = ora_getcolumn ($cursor,8);
                $l_com_ins[$cpt] = ora_getcolumn ($cursor,9);
                $l_com_suppr[$cpt] = ora_getcolumn ($cursor,10);
                $d_suppr[$cpt] = ora_getcolumn ($cursor,11);

                // On recupere le libellé du pays, de l'océan et du type de données
                $pays[$cpt] = ora_getcolumn(Ora_Do ($cnx,"SELECT l_pays_d FROM
a_pays_d WHERE a_pays_d.c_pays_d = $c_pays_d[$cpt]"),0);
                $ocean[$cpt] = ora_getcolumn(Ora_Do ($cnx,"SELECT l_ocea FROM ocean
WHERE ocean.c_ocea = $c_ocea[$cpt]"),0);
                $typeD[$cpt] = ora_getcolumn(Ora_Do ($cnx,"SELECT l_typ_d FROM
a_typ_d WHERE a_typ_d.c_typ_d = $c_typ_d[$cpt]"),0);
                $cpt++;
        } while (ora_fetch ($cursor));
```

```
        // Fermeture du curseur
        ora_close ($cursor);
}
// Déconnexion
Ora_logoff ($cnx);
?>
…
```

La suite du code (HTML) est chargé d'afficher les colonnes lues dans les zones de texte et la liste déroulante, présentes dans l'interface :



Cette page HTML a été conçue avec **DreamWeaver v4** de **MacroMedia**.

Insertion de données

L'administrateur de données peut modifier les paramètres du jeu de données sélectionné, et/ou en ajouter un nouveau. Deux boutons sont prévus à cet effet.

Le code PHP nécessaire à une insertion ou à une mise à jour est plus simple que celui d'une requête *select*. Il suffit d'envoyer la requête avec la fonction ora_do ($cnx,$requete), puis d'appeler la fonction ora_commit ($cnx) qui valide la transaction.

Remarque

Ce point du sujet n'avait pas pour objectif de développer une application définitive, mais de réaliser une étude à des fins méthodologiques. Il est prévu de réaliser ultérieurement d'autres applications en s'appuyant sur celle-ci. C'est pourquoi un grand soin a été porté aux commentaires et à la lisibilité du code.

Conclusion

Ce stage effectué au centre IRD de Brest aura été pour moi très intéressant et très instructif.

Au niveau informatique, j'ai appris énormément concernant le SGBDR Oracle. Mes connaissances en ce domaine se sont enrichies tant au niveau de la création d'une base de données, que pour l'alimentation d'une base, ou encore son administration. Ce logiciel est largement répandu dans le domaine de l'industrie, c'est pourquoi cette expérience ne peut être que bénéfique pour ma poursuite d'étude, ainsi que pour mon entrée dans le monde de l'emploi, où les bases de données sont omniprésentes.

Le développement de l'application m'a permis de me perfectionner dans les langages PHP et HTML que je connaissais déjà.

Le contexte scientifique, la pêche thonière, était très intéressant. De plus, le fait que ce stage entre dans le cadre d'un projet européen était un facteur de motivation supplémentaire.

ANNEXES

Exemples de calculs de taille

Les tables

Quelques rappels :

- hsize : l'espace total d'un bloc sans l'en-tête.
- availSpace : Espace disponible par bloc.
- rowSpace : Taille d'un enregistrement.
- nbLignes : Nombre de lignes par blocs.

Voici les résultats des calculs pour les tables pavillon, a_jeu_d et freqt

	hsize	availSpace	rowSpace	nbLignes
pavillon	8106	7697	74	104
a_jeu_d	8106	5671	2493	2
freqt	8106	7697	39	197

Connaissant le nombre d'enregistrement approximatif de chacune des tables pour 25 ans de données, on peut savoir quel espace ces trois tables occuperont :

- pavillon, 100 enregistrements : cette table occupera un seul bloc de données (8 Ko), car on a vu qu'on pouvait en mettre 104 dans un bloc.
- a_jeu_d, 1000 enregistrements : sachant qu'on peut placer 2 enregistrements de cette table dans un bloc, on utilisera 500 blocs pour stocker les 1000 lignes, soit 4 Mo.
- freqt, 10 000 000 enregistrements : arrondi_sup (10 000 000 / 197) = 50762 blocs, soit plus de 396 Mo.

Les index

Quelques rappels :

- header : taille de l'en-tête.
- availSpace : Espace disponible par bloc.
- somCol : Somme des colonnes participant à l'index.
- indexSize : Taille de l'index.
- nbLignes : Nombre d'index pouvant être stockés dans un bloc.

Résultats des calculs :

	index	header	availSpace	somCol	indexSize	nbLignes
pavillon	**pavillon_pk**	161	8031	3	12	669
a_jeu_d	**a_jeu_d_pk**	161	8031	6	15	535
	a_pays_d_r_a_jeu_d_fk	161	8031	3	12	669
	ocean_r_a_jeu_d_fk	161	8031	2	11	730
	a_typ_d_r_a_jeu_d_fk	161	8031	2	11	730
freqt	**freqt_pk**	161	8031	21	34	236
	espece_r_freqt_fk	161	8031	3	12	669
	echant_r_freqt_fk	161	8031	15	26	308

De la même façon que pour les tables, on détermine l'emplacement nécessaire au stockage des différents index :

- Table pavillon, 100 enregistrements
  - pavillon_pk : 1 bloc
- Table a_jeu_d, 1000 enregistrements
  - a_jeu_d_pk : 2 blocs
  - a_pays_d_r_a_jeu_d_fk : 2 blocs
  - ocean_r_a_jeu_d_fk : 2 blocs
  - a_typ_d_r_a_jeu_d_fk : 2 blocs
- Table freqt, 10 000 000 enregistrements
  - freqt_pk : 42373 blocs
  - espece_r_freqt_fk : 14948 blocs
  - echant_r_freqt_fk : 32468 blocs

Extraits de scripts

Les trois parties qui suivent contiennent des extraits des scripts utilisés lors de la création du schéma balbaya. Pour chacun des scripts qui suivent on se connecte en tant qu'administrateur, c'est-à-dire sous le nom ***balbaya***.

Création des rollback segments

```
/* Création d'un deuxième rollback segment dans le tablespace SYSTEM nécessaire */
/* pour pouvoir ensuite creer de nouveaux tablespaces */
create rollback segment system2 tablespace system;
alter rollback segment system2 online;

/* Création d'un tablespace réservé aux rollback segments */
create tablespace TBS_ALIPA_RBS
datafile
        '/export/ora_wk20/alipa/tbs_rbs.dbf'
        size 120M reuse
/

/* Création de 4 rollback segments de 5M Maximum chacun */
create rollback segment rbs1 tablespace tbs_alipa_rbs
storage (
        initial 1M
        next 1M
        maxextents 5
);
…

alter rollback segment rbs1 online;
…

/* Suppression du rollback segment system2 */
alter rollback segment system2 offline;
drop rollback segment system2;

/* Création d'un gros rollback segment qui sera utilise lorsque l'on ajoutera */
/* ou que l'on supprimera un an de données d'un coup */
create public rollback segment rbs5 tablespace tbs_alipa_rbs
storage (
        initial 20M
        next 20M
        maxextents 5
);

/* Par défaut ce rollback segment est offline, avant de procéder a une opération  d'ajout ou */
/* de suppression d'un an de données il faudra désactiver les 4 autres segments (offline) */
/* et activer rbs5 (online) grâce a la commande: alter rollback segment rbs5 online */
```

Création des tablespaces

```
/* Création d'un tablespace temporaire pour les opérations de tris */
create tablespace TBS_BALBAYA_TEMP
datafile
        '/export/ora_wk20/alipa/balbaya/tbs_temp.dbf'
        size 300M reuse
/

/* Création des tablespaces utilisés pour stocker les données et index des tables capture */
/* et activite */
create tablespace TBS_BALBAYA_CAPT_D
datafile
    '/export/ora_wk4/alipa/balbaya/tbs_capt_d.dbf'
   size 274M reuse
/
create tablespace TBS_BALBAYA_CAPT_I
datafile
    '/export/ora_wk20/alipa/balbaya/tbs_capt_i.dbf'
   size 405M reuse
/

/* Tablespaces pour les données et index des tables echant, freqt et ech_esp */
create tablespace TBS_BALBAYA_MENS_D
datafile
    '/export/ora_wk4/alipa/balbaya/tbs_mens_d.dbf'
   size 468M reuse
/
create tablespace TBS_BALBAYA_MENS_I
datafile
    '/export/ora_wk20/alipa/balbaya/tbs_mens_i.dbf'
   size 909M reuse
/
```

Note : L'option *reuse* est utilisée pour chaque tablespace. Elle permet, lors de la création de la base, de réutiliser les *datafiles* s'ils existent déjà. Cela permet de ne pas avoir de message d'erreur lorsque l'on recrée la base.

Création des utilisateurs

```
/* Création des rôles */
create role balbaya_admin identified by XXXX;
create role balbaya_lecteur identified by XXXX;
create role balbaya_appli identified by XXXX;

/* Permet aux utilisateurs de se connecter à Oracle et de créer des synonymes */
grant create session, create synonym to balbaya_admin, balbaya_lecteur, balbaya_appli;

/* Attribution des privilèges du role balbaya_admin (administrateur de données) */
grant insert on capture to balbaya_admin;
grant insert on activite to balbaya_admin;
…
grant update on a_jeu_d to balbaya_admin;
grant update on bateau to balbaya_admin;
grant select any table to balbaya_admin;
grant select any sequence to balbaya_admin;

/* Création de l'utilisateur balbaya_adm et affectation du rôle balbaya_admin */
create user balbaya_adm identified by XXXX;
grant balbaya_admin to balbaya_adm;

/* Maintenant que l'utilisateur est créé, et qu'il a des privilèges sur le schéma balbaya */
/* il va pouvoir exécuter des commandes SQL. Cependant, s'il veut par exemple consulter */
/* les  activités présentes dans la base, il devra taper select * from balbaya.activite, car les */
/* tables se trouvent dans le schéma de balbaya. */
/* Pour éviter des lignes de commandes de ce type, il faut utiliser des synonymes */
/* Pour chacun des utilisateurs créés, ce script se connecte sous leur nom et crée les */
/* les synonymes dont ils ont besoin. */

connect balbaya_adm/XXXX

/* Création des synonymes */
create synonym activite for balbaya.activite;
create synonym assoc_g for balbaya.assoc_g;
create synonym assoc_r for balbaya.assoc_r;
…

connect balbaya/XXXX

/* Attribution des privilèges du role balbaya_lecteur, création des utilisateurs balbaya_app */
/* et balbaya_lect. Création des synonymes dans leurs propres schémas */
…
```

Connexion à Oracle

La connexion à Oracle via un réseau nécessite un minimum de configuration. Côté serveur, un process appelé *listener* doit être lancé au préalable. Il est un indépendant des autres process Oracle. Avant de le lancer, il est nécessaire de renseigner le fichier *listener.ora*. Côté client, un fichier tnsnames.ora doit être défini.

Listener.ora

Le fichier listener.ora que nous avons utilisé est présenté ci-dessous :

```
# Installation Generated Net8 Configuration
# Version Date: Jun-17-97
# Filename: Listener.ora
#
LISTENER =
  (ADDRESS_LIST =
    (ADDRESS=
          # Description du protocole utilisé, du nom du serveur de base de données et du
          # numéro de port.
          (PROTOCOL= TCP)
          (Host= galaad)
          (Port= 1521)
    )
  )
# Cette partie peut contenir plusieurs noms (SID) différents pour lesquels un listener est
# installé. Dans notre cas, le serveur galaad n'héberge qu'une seule base : ALIPA
SID_LIST_LISTENER =
  (SID_LIST =
          (SID_DESC =
                  (ORACLE_HOME= /export/logiciels/oracle/product/8.0.5)
                  (SID_NAME = ALIPA)
          )
  )
STARTUP_WAIT_TIME_LISTENER = 0
CONNECT_TIMEOUT_LISTENER = 10
TRACE_LEVEL_LISTENER = OFF
```

Le lancement du *listener* se fait grace à la commande :

```
galaad> lsnrctl start
Services Summary...
  ALIPA        has 1 service handler(s)
The command completed successfully
galaad>
```

Note : La commande *lsnrctl start* peut être suivie du SID de la base, cependant, le fichier listener.ora contient un seul *SID_DESC*. Le *listener* sera bien positionné sur la base **ALIPA**.

tnsnames.ora

Ce fichier doit être accessible côté client. Il contient, comme le fichier listener.ora, des informations sur le serveur de base de données, ainsi que le SID de la base.

Contenu du fichier *tnsnames.ora* utilisé :

```
# Installation Generated Net8 Configuration
# Version Date: Oct-27-97
# Filename: Tnsnames.ora
#

# On retrouve ici les mêmes informations que dans le fichier listener.ora présent sur galaad
ALIPA =
	(DESCRIPTION =
		(ADDRESS =
			(COMMUNITY=ird.fr)
			(PROTOCOL= TCP)
			(Host= galaad)
			(Port= 1521)
		)
		(CONNECT_DATA =
			(SID = ALIPA)
		)
	)
```

Connexion

Mécanisme de connexion d'une l'application PHP à la base **ALIPA**, à l'aide d'un navigateur sous Windows (Cas d'une connexion avec un login et un mot de passe corrects) :

- Appel de la fonction PHP ora_logon ($login,$password). Le login est consitué ainsi : login@ALIPA, indiquant le nom de la base à laquelle on désire se connecter.
- Lecture des informations contenues dans le fichier tnsnames.ora, le paramètre CONNECT_DATA est positionné sur ALIPA => envoi de la demande de connexion au serveur de base de données galaad (via le procole TCP, en utilisant le numéro de port 1521).
- Le listener reçoit la demande de connexion et l'envoie à Oracle
- Le SGBDR vérifie le login et le mot de passe transmis, et accepte la connexion

Une fois la connexion établie, les échanges pourront se faire entre les deux machines. Les requêtes et les réponses transiteront via le *listener*.

Script PHP de saisie

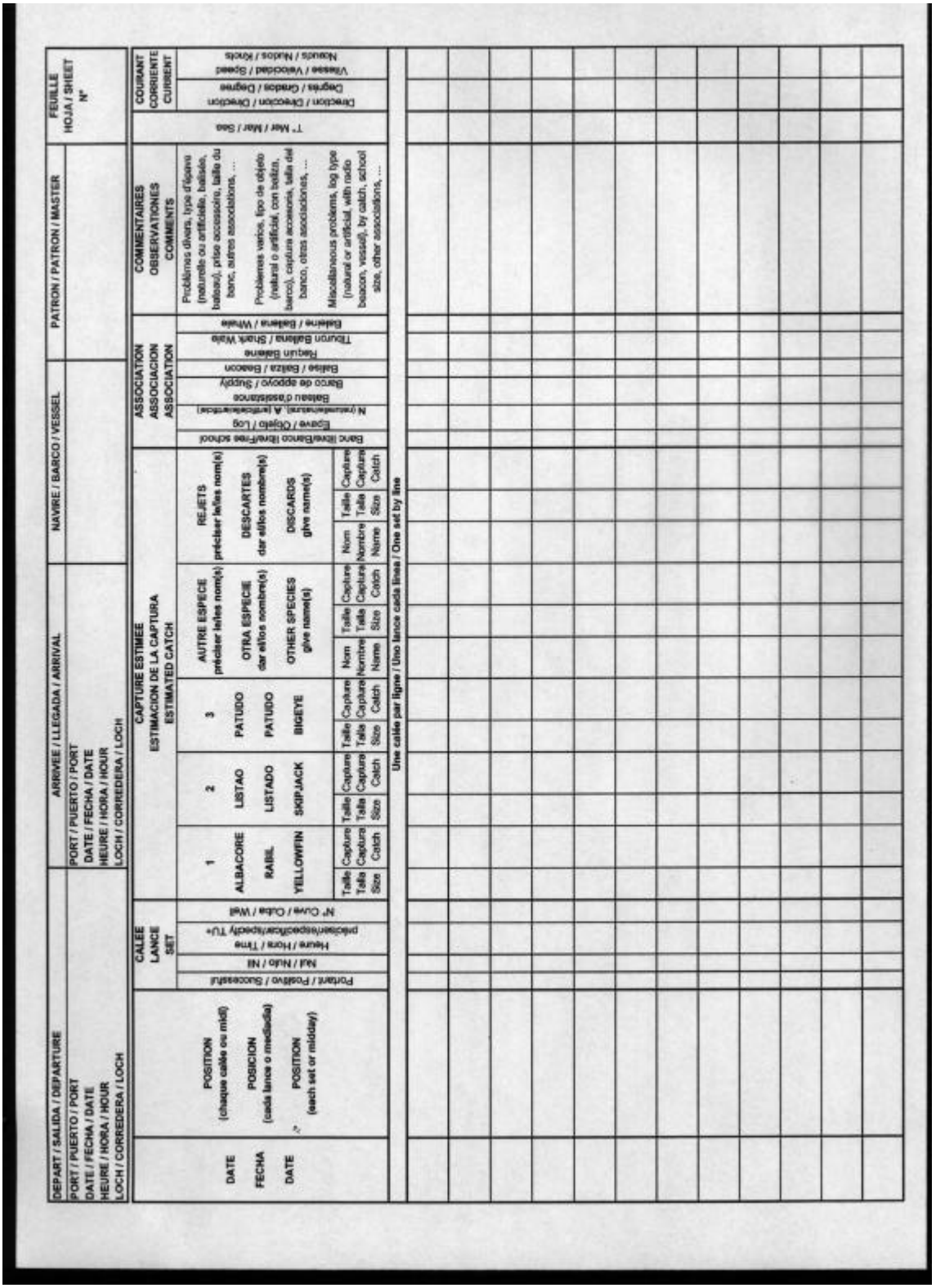
```
<?
// Fichier saisie_jeu_d.php

// Ce script reçoit en parametres les valeurs correspondants au jeu de données sélectionné dans la liste déroulante. Ces
// parameters sont passes par l'intermédiaire de champs caches (type hidden)
// Ils portent les mêmes noms que les colonnes de la table a_jeu_d, avec un p_ devant indiquant qu'il s'agit de paramètres
include ("fonctions.inc");

// Connexion sous le login balbaya_adm
$cnx = connecter ("balbaya_adm","XXXX");

// Echec de connexion affichage d'un message d'erreur et arrêt du script (appel de la fonction erreur définie dans functions.inc
if (! $cnx) {
	erreur ("Echec de connexion à Oracle sous le nom balbaya_adm<br>ora_error($cnx)");
}

// Connexion réussie
else {
	// Date d'aujourd'hui au format JJ/MM/AAAA
	$jour = date("j/m/Y");

	// On récupère maintenant les données que l'on aura besoin plus tard pour remplir les listes et les champs
	// de texte HTML

	// Valeur de la sequence S_ID_JEU_D
	$query = 'SELECT s_id_jeu_d.nextval FROM DUAL';
	if ($cursor = ora_do ($cnx,$query)) {
		$seq = ora_getcolumn ($cursor,0);
	}
	else {
		erreur ("Problème lors de la lecture de la valeur de la sequence s_id_jeu_d<br>");
	}

	// Noms des pays
	$query = "SELECT c_pays_d,l_pays_d FROM a_pays_d ORDER BY c_pays_d ASC";

	if ($cursor = ora_do ($cnx,$query)) {
		$cpt=0;
		do {
			// Stockage du code et du libellé de tous les pays de données
			$c_pays_d[$cpt] = ora_getcolumn ($cursor,0);
			$l_pays_d[$cpt] = ora_getcolumn ($cursor,1);

			$cpt++;
		}while (ora_fetch ($cursor));

		// Fermeture du curseur
		ora_close ($cursor);
	}

	// Erreur lors de l'exécution de la requete
	else {
		erreur ("Problème lors de l\'exécution de la requête $query<br>");
	}

	// Noms des océans
	$query = "SELECT c_ocea,l_ocea FROM ocean ORDER BY c_ocea ASC";

	if ($cursor = ora_do ($cnx,$query)) {
		$cpt=0;
		do {
			// Stockage du code et du libellé des océans
			$c_ocea[$cpt] = ora_getcolumn ($cursor,0);
			$l_ocea[$cpt] = ora_getcolumn ($cursor,1);

			$cpt++;
		}while (ora_fetch ($cursor));

		// Fermeture du curseur
		ora_close ($cursor);
	}
```

```
		// Erreur lors de l'exécution de la requete
		else {
			erreur ("Problème lors de l\'exécution de la requête $query<br>");
		}

		// Libellés des type de données
		$query = "SELECT c_typ_d,l_typ_d FROM a_typ_d ORDER BY c_typ_d ASC";

		if ($cursor = ora_do ($cnx,$query)) {
			$cpt=0;
			do {
				// Stockage du code et du libellé des types de données
				$c_typ_d[$cpt] = ora_getcolumn ($cursor,0);
				$l_typ_d[$cpt] = ora_getcolumn ($cursor,1);

				$cpt++;
			}while (ora_fetch ($cursor));

			// Fermeture du curseur
			ora_close ($cursor);
		}

		// Erreur lors de l'exécution de la requete
		else {
			erreur ("Problème lors de l\'exécution de la requête $query<br>");
		}

?>


<html>

<body bgcolor="#0099FF" text="#000000">
<p> </p><form name="form1" method="post" action="inserer_jeu_d.php">
  <div align="center">
    <table width="84%" border="0" cellspacing="10">
      <tr>
        <td width="52%">
          <div align="right"><font size="+2">Pays</font></div>
        </td>
        <td width="48%">
          <div align="left">
            <select name="pays">
<?
// Creation de la liste des pays
for ($i=0;$i<count($l_pays_d);$i++) {
		// $p_c_pays_d est le code du pays sélectionné dans la fenêtre HTML précédente
		// C'est donc cette valeur qui sera sélectionnée dans la liste
		if ($c_pays_d[$i] == $p_c_pays_d) {
			print ("<option selected value=$c_pays_d[$i]>$l_pays_d[$i]\n");
		}
		else {
			print ("<option value=$c_pays_d[$i]>$l_pays_d[$i]\n");
		}
}
 ?>
            </select>
          </div>
        </td>
      </tr>
      <tr>
        <td width="52%">
          <div align="right"><font size="+2">Oc&eacute;an</font></div>
        </td>
        <td width="48%">
          <div align="left">
            <select name="ocean">
<?
// Creation de la liste des océans
for ($i=0;$i<count($l_ocea);$i++) {
		if ($c_ocea[$i] == $p_c_ocea) {
			print ("<option selected value=$c_ocea[$i]>$l_ocea[$i]\n");
		}
		else {
			print ("<option value=$c_ocea[$i]>$l_ocea[$i]\n");
		}
```

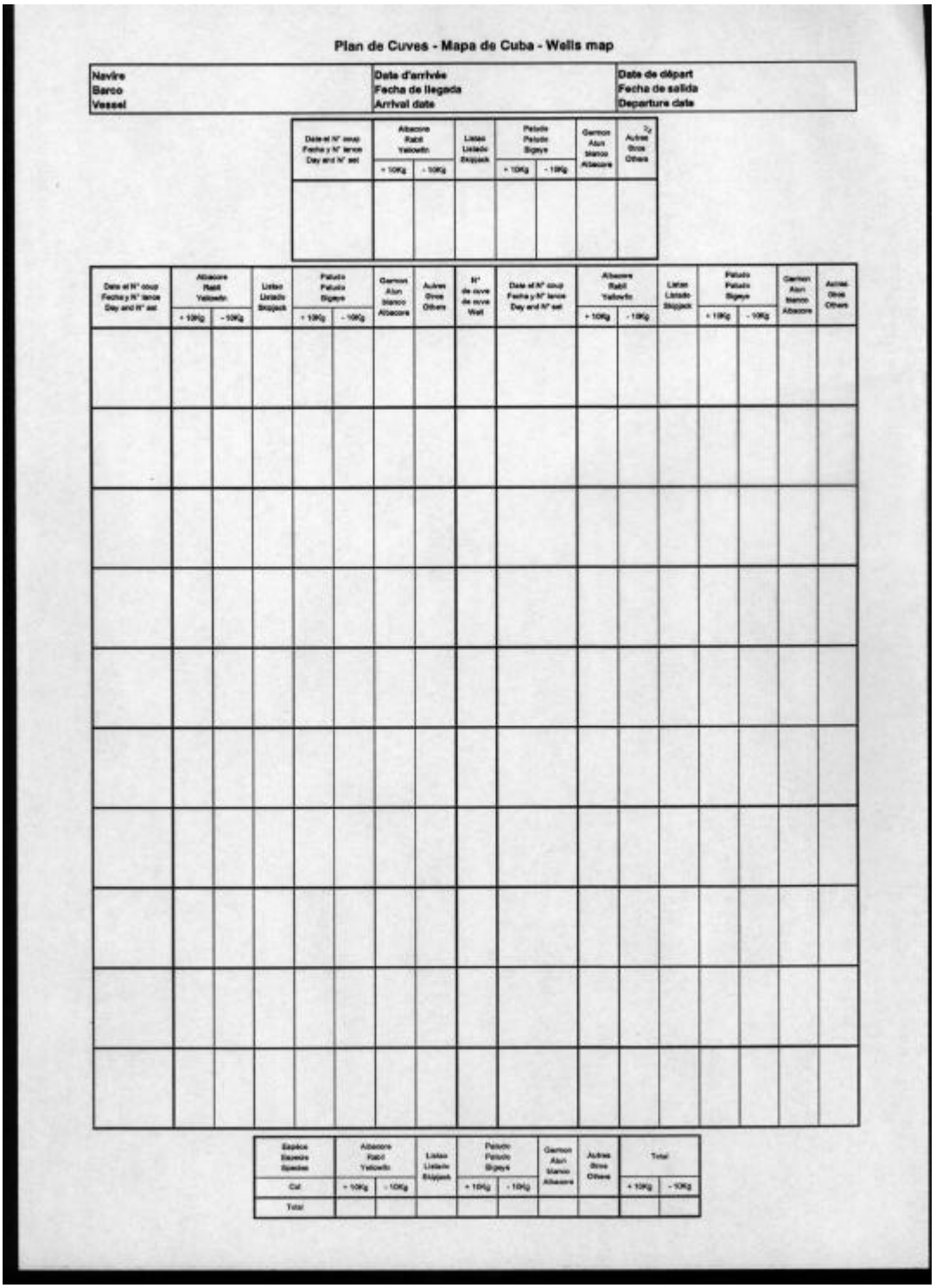
```
    }
?>
          </select>
        </div>
       </td>
     </tr>
     <tr>
       <td width="52%">
         <div align="right"><font size="+2">Type de donn&eacute;es</font></div>
       </td>
       <td width="48%">
         <div align="left">
           <select name="typ_d">
<?
// Creation de la liste des types de données
for ($i=0;$i<count($l_typ_d);$i++) {
              if ($c_typ_d[$i] == $p_c_typ_d) {
                        print ("<option selected value=$c_typ_d[$i]>$l_typ_d[$i]\n");
              }
              else {
                        print ("<option value=$c_typ_d[$i]>$l_typ_d[$i]\n");
              }
}
?>
    </select>
         </div>
       </td>
     </tr>
     <tr>
       <td width="52%">
         <div align="right"><font size="+2">Date de d&eacute;but</font></div>
       </td>
       <td width="48%">
         <div align="left">
           <input type="text" size="10" name="d_deb_d"
<?
print ("value=\"$p_d_deb_d\">");
?>
                  <font size="+1">  (jj/mm/aaaa)</font>
         </div>
       </td>
     </tr>
     <tr>
       <td width="52%">
         <div align="right"><font size="+2">Date de fin</font></div>
       </td>
       <td width="48%">
         <div align="left">
           <input type="text" size="10" name="d_fin_d"
<?
print ("value=\"$p_d_fin_d\">");
?>
                  <font size="+1">  (jj/mm/aaaa)</font>
         </div>
       </td>
     </tr>
     <tr>
       <td width="52%">
         <div align="right"><font size="+2">Nom du fichier de donn&eacute;es</font></div>
       </td>
       <td width="48%">
         <div align="left">
           <input type="text" name="l_fic_d"
<?
print ("value=\"$p_l_fic_d\">");
?>
         </div>
       </td>
     </tr>
     <tr>
       <td width="52%">
         <div align="right"><font size="+2">Origine du fichier</font></div>
       </td>
       <td width="48%">
         <div align="left">
           <input type="text" name="l_org_fic_d"
```

```
<?
print ("value=\"$p_l_org_fic_d\">");
?>
        </div>
       </td>
     </tr>
     <tr>
       <td width="52%">
         <div align="right"><font size="+2">Date d'insertion</font></div>
       </td>
       <td width="48%">
         <div align="left">
           <input type="text" size="10" name="d_dispo"
<?
print (" value=$jour>");
?>
                 <font size="+1">  (jj/mm/aaaa)</font>
         </div>
       </td>
     </tr>
     <tr>
       <td width="52%">
         <div align="right"><font size="+2">Commentaire d'insertion</font></div>
       </td>
       <td width="48%">
         <div align="left">
           <input type="text" name="l_com_ins"
<?
print ("value=\"$p_l_com_ins\">");
?>
         </div>
       </td>
     </tr>
   </table>
  </div>
  <p align="center">
    <input type="submit" name="Submit" value="  Ajouter  ">
  </p>
</form>
<p> </p>
</body>
</html>

<?
// Deconnexion de la base
Ora_logoff($cnx);

// Fin du else du début
}
?>
```

Exemple de livre de bord



Exemple de plan de cuve



Plan de Cuves - Mapa de Cuba - Wells map

Navire Barco Vessel	Date d'arrivée Fecha de llegada Arrival date	Date de départ Fecha de salida Departure date

Date et N° coup Fecha y N° lance Day and N° set	Albacore Rabil Yellowfin		Listao Listado Skipjack	Patudo Patudo Bigeye		Germon Atun blanco Albacore	Autres Otros Others	N° de cuve de cuva Well	Date et N° coup Fecha y N° lance Day and N° set	Albacore Rabil Yellowfin		Listao Listado Skipjack	Patudo Patudo Bigeye		Germon Atun blanco Albacore	Autres Otros Others
	+ 10Kg	- 10Kg		+ 10Kg	- 10Kg					+ 10Kg	- 10Kg		+ 10Kg	- 10Kg		

Espèce Especie Species	Albacore Rabil Yellowfin		Listao Listado Skipjack	Patudo Patudo Bigeye		Germon Atun blanco Albacore	Autres Otros Others	Total	
Cul	+ 10Kg	- 10Kg		+ 10Kg	- 10Kg			+ 10Kg	- 10Kg
Total									

Bibliographie

Documentation sur Oracle 8 fournie avec le logiciel *(en anglais)* **:**

Oracle 8 Administrator's guide
Oracle 8 concepts
Oracle 8 Tuning
Oracle 8 Utilities
Oracle 8 Reference
Oracle 8 Error Messages
SQL Reference
PL/SQL Reference

AVDTH98 – Acquisition et validation des données de pêche au thon tropical, *Jean Jacques Lechauve*, 1999.

Oracle et le Web – *Didier Deléglise* – Eyrolles, 1998

Oracle 7 – Langages – Architecture – Administration, *A. Abdellatif, M. Limame, A Zeroual* – Eyrolles 1998.

Oracle 8 DBA Handbook – *Kevin Loney* **–** Oracle Press, 1998.

Pratique d'Oracle – Universal Data Server – *Editions Weka*, 2001.

Programmation en PHP – *Leon Atkinson* – Campus Presse Référence, 1999.

Programmation Web avec PHP – *L. Lacroix, N. Leprince, C. boggero, C. Lauer* – Eyrolles, 2000.

Power AMC – Mode d'emploi

Sites consultés

www.ird.fr

www.oracle.com/fr/
didier.deleglise.free.fr/dba/sqlnet/sqlnet.htm (Informations sur listener.ora et tnsnames.ora)

www.phpindex.com/
dev.nexen.net/
www.php.net/

www.bd.enst.fr/~dombd/Cours/Applications/ (Architectures 2 tiers et 3 tiers)