

ifremer

Olivier CATRY

DUT Informatique

IUT Informatique de Nantes
3, rue du Maréchal Joffre
B.P. 34103
44041 Nantes Cedex 1



**Outil de volumétrie pour
Quadrigé²**
Développement web et base de données
Rapport de stage

Enseignant responsable : Didier Kuéviakoé
Maître de stage : Jean-Claude MASSON
Dates : du 14 avril au 19 juin 2009

Remerciements

Je remercie Jean-Claude Masson pour m'avoir proposé ce stage à l'Ifremer, pour toutes les explications qu'il m'a données et le soutien qu'il m'a apporté.

Je remercie également tous les membres du service VIGIES, dont Antoine Huguet pour m'avoir accueilli dans son service, Dominique Soudant, Alain Le Magueresse, Bernard Raffin, Gaétane Durand, Emilie Gauthier, Caroline Hourcade, Elodie Fleury, Christine Maisonneuve et Laure Remande pour leur gentillesse et leur disponibilité.

Enfin, je remercie grandement l'équipe de professeurs de l'IUT informatique de Nantes pour l'enseignement de qualité qu'ils m'ont apporté. Je remercie particulièrement dans le cadre de ce stage :

- Didier Kuéviakoé, mon responsable de stage, qui nous a enseigné les différents aspects de la vie dans le monde de l'entreprise et du travail au travers des cours de projet professionnel et d'« info et société », ce qui m'a permis de trouver ce stage et de m'intégrer au service VIGIES de l'Ifremer.
- Jean-François Hüe pour son enseignement sur la stratégie à mettre en œuvre dans la conception d'algorithmes, le traitement et l'optimisation des structures de données, au travers des cours de complément algorithmique, d'architecture et de structure de donnée, cela m'a permis de réaliser de nombreuses fonctions de bas niveau d'opérations sur les tableaux.
- Abdelghani Hadj-Rabia, Gilles Nachouki et Pascal Sotin pour tous leurs cours de base de données, qui m'ont permis de traiter efficacement la base de données Quadrige² et de mieux comprendre ses enjeux.
- Christine Jacquin et Jean François Rémml pour leurs cours et leurs conseils sur le développement web, avec l'aide desquels j'ai réalisé efficacement un site ajouté au réseau Ifremer.
- Sébastien Cazalas pour ses nombreux conseils que j'ai suivis dans la rédaction de ce rapport de stage, et dans la préparation à la soutenance.
- Et tous les autres professeurs dont les précieux conseils m'auront permis d'avoir un œil plus analytique sur le travail qui m'a été demandé.

Résumé

Ce rapport présente le développement de l'outil de volumétrie pour la base de données Quadrige². Il présente tout d'abord le rôle de Ifremer et les enjeux de la base de données Quadrige². L'outil de volumétrie permet de récupérer le nombre de résultats de mesures en fonction de critères.

Le développement de l'outil est alors présenté.

Pour réaliser cet outil, j'ai dû étudier la demande des utilisateurs et la base de données Quadrige². J'ai alors écrit un cahier des charges fonctionnel et réalisé les spécifications et développé un gestionnaire de données en PHP ainsi qu'un programme de communication par requêtes avec la base de données. Pour finir, j'ai réalisé une interface graphique pour ce logiciel.

Enfin, un point est fait sur la présentation du logiciel aux futurs utilisateurs. Leurs remarques permettront au service de réaliser des spécifications précises en vue d'un développement définitif ultérieurement.

Sum up

This report explains the development of the volumetric tool for the Quadrige² database. Its aim is, firstly, to point out the role of Ifremer and those of the Quadrige² database. The volumetric tool allows the user to get the amount of measurements relative to restrictive criterias and grouped by chosen data.

Then, the development of the tool is explained.

In order to develop this tool, I studied the requests of the users and the Quadrige² database. After that, I wrote the book of specifications which leads me to the development of a data management program in php, and a SQL requests-based communication program in order to communicate with the database. At last, I realized a graphical user interface for the software.

Finally, the presentation of the program to the users is summarized. The comments will allow to make the final specifications, in order to develop a final version of the software.

Sommaire

<i>Remerciements</i>	2
<i>Résumé</i>	4
<i>Sum up</i>	4
<i>Sommaire</i>	6
<i>Présentation de l'entreprise</i>	6
<i>Introduction</i>	9
<i>I) - Présentation du projet</i>	10
<i>II) – Etudes</i>	11
<i>III) – Développement</i>	18
<i>Bilan</i>	30
<i>Conclusion</i>	30
<i>Bibliographies</i>	31
<i>Glossaire</i>	31

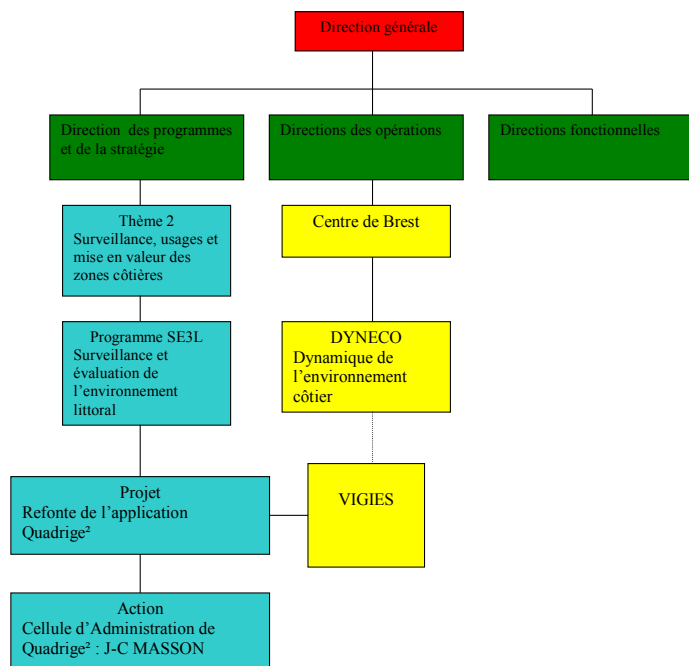
Présentation de l'entreprise

L'Ifremer (Institut français de recherche pour l'exploitation de la mer) est un organisme public à caractère industriel et commercial (EPIC) créé en 1984 et placé sous la tutelle conjointe des ministères de l'Écologie, de l'Énergie, du Développement durable et de l'Aménagement du Territoire, de l'Enseignement supérieur et de la Recherche et de l'Agriculture et de la Pêche.

Il compte cinq centres de recherches (Manche et mer du Nord, Brest, Nantes, Méditerranée et Tahiti) et 26 implantations sur tout le littoral métropolitain et dans les DOM-TOM.

Son organisation s'articule autour de 3 structures :

- les directions fonctionnelles (Affaires financières et juridiques, communication, opérations et moyens navals, ressources humaines et valorisation),
- la direction des programmes et de la stratégie qui anime et coordonne les activités scientifiques et technologiques,
- la direction des opérations à laquelle sont rattachées les unités. Le département DYNECO (Dynamique de l'Environnement Côtier) contribue, par ses activités de recherche finalisée et par une offre de services adaptés, aux missions dévolues à l'Ifremer pour l'observation et la modélisation des dynamiques couplées de l'environnement côtier. Au sein de ce département, le service VIGIES (Valorisation de l'information pour la Gestion Intégrée Et la Surveillance) assume la responsabilité nationale de la valorisation des réseaux de la surveillance littorale via la base Quadrige².



positionnement de l'action « Administration de Quadrige² » dans l'organigramme Ifremer pour la période 2007 – 2008.

Organigramme du service VIGIES (centre de Nantes) :

Antoine HUGUET (chef de service)	Ingénieur Cahier des Charges / Bases de données
Jean-Claude MASSON (maître de stage)	Ingénieur Cahier des charges / Intégration de données
Dominique SOUDANT	Ingénieur Statistiques / Indicateurs
Emilie GAUTHIER	Ingénieur gestion de données biologiques
Anne GROUHEL	Ingénieur Réseaux de surveillance / Coordination
Alain LE MAGUERESSE	Ingénieur Technologies WEB / Cahier des charges
Bernard RAFFIN	Technicien Bases de données / Cartographie
Gaétane DURAND	Technicienne Statistiques / Bases de données
Christine MAISONNEUVE	Assistante Suivi budgétaire / Organisation
Élodie FLEURY	CDD 18 mois – Cartographie
Caroline HOURCADE	CDD 18 mois – Programmation sous R
Laure REMANDE	Stage - programmation sous R
Olivier CATRY	Stage - développement base de données

Introduction

L'Ifremer contribue, par ses travaux et expertises, à la connaissance des océans et de leurs ressources, à la surveillance du milieu marin et littoral et au développement durable des activités maritimes. Au sein de l'Ifremer le service VIGIES a pour mission la mise en place d'un système d'information permettant de stocker les données de la surveillance littorale : Quadrige².

Le MEDDAT a désigné Quadrige² comme étant le Système d'information national de référence dédié au stockage et à l'exploitation des données collectées par les réseaux de surveillance du littoral. Il fait suite au système Quadrige et comporte une base de données sous ORACLE. Quadrige² ne possède pas d'outil de volumétrie, contrairement à Quadrige. Cet outil permettra aux responsables des programmes de la surveillance de posséder des indicateurs pour alimenter leur rapport d'activité.

En 2008, au cours d'une réunion, les responsables des différents réseaux ont définis les grandes lignes de cet outil. L'objectif de ce stage est de réaliser un prototype qui sera testé par les futurs utilisateurs. Leurs remarques permettront de réaliser les spécifications détaillées de l'outil définitif qui sera réalisé ultérieurement.

Ce dossier débute par l'étude des besoins exprimés et de la base de données Quadrige². Ce travail a permis l'écriture d'un cahier des charges fonctionnel, de spécifications et la réalisation de l'architecture globale du logiciel.

Le développement se déroule en 3 étapes :

- la création d'un gestionnaire de données en PHP,
- le développement d'un programme de communication alliant code PHP et requêtes SQL entre le gestionnaire et la base de données Quadrige²,
- la mise en place de l'interface graphique du logiciel.

I) - Présentation du projet

1) La demande

L'outil de volumétrie de Quadrige² doit permettre de récupérer le nombre de résultats de mesures en fonction de critères, présentées groupées suivant un certain nombre d'entités.

Il doit proposer de nouveaux critères de restriction par rapport à l'outil de volumétrie de Quadrige, comme les entités de regroupement.

Les utilisateurs ont exprimé le besoin d'avoir un écran intermédiaire permettant de mettre en forme les résultats de volumétrie. On y choisit les données de groupement et de tri.

Les données de volumétrie s'affichent dans l'interface du logiciel. Mais les utilisateurs souhaitent pouvoir les extraire pour les lire depuis un tableur tel que Microsoft Excel.

Dans la base de données Quadrige², les résultats de mesure peuvent être rattachées à plusieurs réseaux, donc l'outil doit pouvoir gérer le multi-programmes. Ce fonctionnement est détaillé par la suite.

2) Les raisons de mon choix

Dans le cadre du stage de seconde année d'IUT informatique, je cherchais à accroître mes compétences en base de données, mais aussi à travailler dans un milieu scientifique. C'est pourquoi le stage proposé par l'Ifremer, dont le sujet était le développement autour de données volumétriques Quadrige², m'a attiré.

Les ingénieurs en base de données sont très recherchés de nos jours, et le thème des données autour de la surveillance maritime est un sujet original et passionnant. C'est pour ces raisons, que j'ai choisi ce stage à l'Ifremer.

II) – Etudes

1) L'organisation

Après avoir étudié le compte rendu de réunion dans lequel les utilisateurs ont décrit leurs besoins, j'ai pu définir :

- les tâches à réaliser,
- le planning de réalisation, avec des marges pour ne pas manquer de temps.

J'ai réparti mes tâches selon le planning suivant:

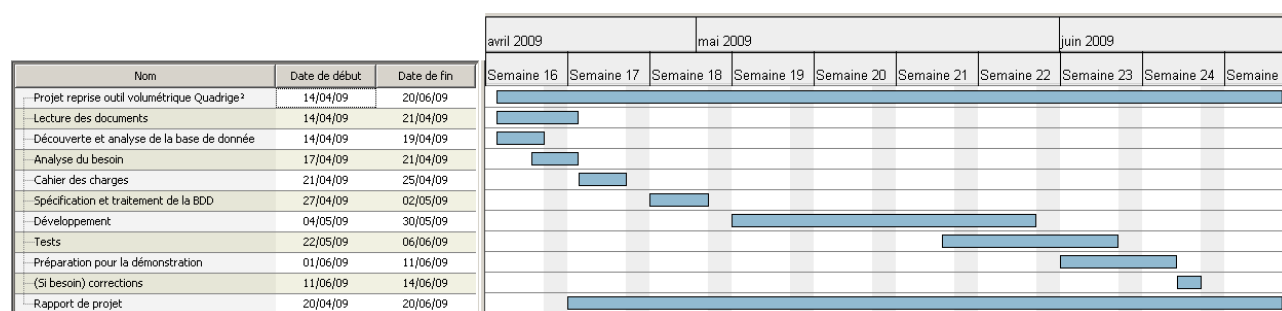


Figure 1 : Répartition des tâches sous un diagramme de Gantt

Avant de découvrir la base de données Quadrigé², j'ai manipulé l'outil d'extraction du logiciel Quadrigé². C'est un logiciel réalisé en java avec l'outil « Eclipse RCP » permettant de récupérer les résultats de mesures en fonction de critères de sélection. On sélectionne plusieurs choix parmi plusieurs critères de sélection, et, quand on le désire, on obtient la liste des mesures relatives à tous ces choix.

Ensuite, j'ai étudié la base de données, qui est composée de 247 tables. Mon premier travail, absolument nécessaire pour cerner les données dont j'ai besoin, et leurs relations dans la base de données, était de réaliser le modèle logique des données intervenant dans l'outil de volumétrie. Pour cela, j'ai réalisé une vue du modèle logique global en y faisant figurer uniquement les tables et les attributs dont j'ai besoin.

--> voir annexe modèle logique de données

2) Le choix des outils

J'ai eu à choisir le langage que j'allais utiliser afin de réaliser l'application. Il y avait deux solutions :

- développer en java pour intégrer l'outil au logiciel Quadrigé²,
- créer un site pour développer un outil en PHP.

Développer le logiciel en java m'aurait demandé d'analyser le code et l'architecture applicative du logiciel Quadrigé². Le stage ne durant que 10 semaines, je n'avais pas le temps pour une telle analyse.

C'est donc pourquoi je me suis orienté vers le choix du site en PHP. Le langage PHP est un langage serveur dans le cadre d'un modèle client-serveur, c'est à dire que le client envoie de simples

requêtes au serveur qui lui restitue des pages générées automatiquement par un moteur PHP. Le langage PHP est un langage de plus en plus utilisé de nos jours.

L'Ifremer m'autorisant à télécharger et installer des logiciels gratuits, j'ai choisi Oracle SQL Developer pour analyser la base de données, car le Système de Gestion de Base de Données (SGBD) est un serveur Oracle, et, de plus, nous avons été formé à l'IUT à l'utilisation de ce logiciel.

Un programme en PHP nécessitant un serveur pour fonctionner, j'avais le choix entre développer mon programme et le charger sur le serveur intranet de l'Ifremer pour chaque test, ou installer un serveur virtuel sur ma machine, et tout développer localement. J'ai opté pour la seconde solution pour des raisons pratiques, mais il a alors fallu que je configure mon serveur virtuel pour qu'il soit paramétré comme le serveur PHP de l'Ifremer.

Ceci étant fait, j'ai pu déterminer les tâches à réaliser :

- réaliser un jeu de requêtes sur la base de données, simulant une utilisation précise de l'outil de volumétrie,
- spécifier le fonctionnement du logiciel,
- développer un première étape du logiciel consistant à proposer et enregistrer des choix, automatiquement, parmi une liste de critères de restriction,
- créer la liste de critères de restriction et leur requête associée,
- développer une seconde étape du logiciel consistant à retourner la liste des mesures, restreintes aux choix réalisés. Ceci permettrait de tester l'étape suivante,
- développer une troisième étape du logiciel consistant à retourner le nombre de mesures, également restreintes aux choix réalisés,
- tester et corriger les bugs par couche de développement (voir le modèle de développement plus bas),
- réaliser l'interface du logiciel,
- mettre en place de la sécurité du logiciel (accès par mot de passe, restriction selon les utilisateurs),
- présenter l'outil aux futurs utilisateurs.

Le stage devant durer 10 semaines, j'ai partagé les tâches en 10 parties, chacune ne devant durer qu'une semaine au maximum.

3) Le cahier des charges fonctionnel

a. Présentation générale du problème

Objectif :

Réaliser un prototype afin d'affiner les demandes des utilisateurs par rapport à l'outil de volumétrie de Quadrige².

Finalités :

Le logiciel est réalisé en PHP, ce qui lui permet d'être interprété par un navigateur web. Il réalise les

différentes opérations sur la base de données, sur le serveur d'Ifremer, en SQL. (Oracle SQL).

Contexte :

Quadrigé² est le système d'information environnemental de l'Ifremer, dont l'objectif est la gestion des données d'environnement des réseaux de surveillance opérés par l'Ifremer et par des partenaires extérieurs.

Il existait un outil de volumétrie sur l'ancienne base de données Quadrigé. Une réunion (22 janvier 2008) réunissant les responsables de programme de Quadrigé² ont mis en évidence plusieurs points d'évolution de l'outil de volumétrie de Quadrigé.

Études déjà effectuées :

- étude de la base de données Quadrigé² et de son modèle logique de donnée,
- étude de l'outil de volumétrie du logiciel Quadrigé².

Enoncé du besoin :

L'utilisateur a besoin d'évolutions à l'outil de volumétrie du logiciel Quadrigé.. Pour cela, les développeurs de cette nouvelle version ont besoin d'un prototype affinant ces demandes d'évolution.

Le périmètre innovant se décompose en quatre sous-ensembles :

- ajouter de nouveaux critères d'extraction des mesures,
- obtenir le nombre de mesures selon des critères choisis et les trier selon un critère choisi,
- afficher les résultats selon un ordonnancement des données choisi.

Environnement du produit recherché :

- l'utilisateur est un humain capable et ayant la possibilité d'utiliser un ordinateur,
- l'utilisateur n'a pas obligatoirement une machine puissante,
- l'utilisateur possède une connexion intranet ou extranet,
- l'utilisateur n'a pas obligatoirement une version du navigateur mise à jour,
- l'utilisateur n'a pas obligatoirement la plateforme java d'installée,
- le temps est une contrainte pour l'utilisateur,
- la complexité d'utilisation est une contrainte,
- le serveur n'est pas capable de réaliser une infinité de tâches. Il ne doit pas être surchargé.

b. Expression fonctionnelle du besoin

Fonctions de service principales :

- le projet propose à l'utilisateur et au développeur un prototype de l'outil de volumétrie de Quadrigé² avec les évolutions demandées,
- l'utilisateur peut faire un ou plusieurs choix parmi une liste de critères,
- ces choix sont interdépendants des autres choix réalisés. (Par exemple, il ne peut choisir un lieu n'ayant été visité qu'en 1999 s'il a choisi un événement qui s'est réalisé en 2006),
- l'utilisateur peut supprimer un ou plusieurs choix parmi la liste des choix pour chaque critère,
- l'utilisateur peut choisir de terminer ses choix et de préparer ses résultats,
- il peut regrouper les résultats selon un ou plusieurs critères,
- il peut trier les résultats selon un critère,
- il peut ordonnancer les résultats,

- il peut obtenir le nombre de mesures selon un ou plusieurs critères voulus,
- il peut trier ces nombres de mesures selon un critère au choix,
- l'utilisateur doit pouvoir connaître le nombre de mesures pour chaque programme, y compris chaque groupe de programmes,
- l'utilisateur doit pouvoir exporter ses résultats au format « csv »,
- l'utilisateur doit pouvoir naviguer parmi toutes les pages sans contrainte,
- le programme doit être un site web que l'on peut intégrer au site de l'Ifremer.

Fonctions de service complémentaires :

De nouvelles fonctions de service pourront être envisagées au cours du développement du projet.

Contraintes :

- la base de données Quadrige² est fortement sollicitée, et donc parfois lente d'accès,
- elle supporte mal les requêtes retournant un grand nombre de résultats,
- le projet doit être réalisé en 10 semaines,
- l'interface doit obéir à la charte graphique d'Ifremer,
- le programme en PHP ne doit pas surcharger la machine serveur.

Critères d'appréciation :

Rapidité, efficacité, ergonomie,

Besoins fonctionnels :

- besoin d'un prototype affinant les demandes d'évolution de l'outil de volumétrie de Quadrige²,
- besoin que l'application tourne sur un ordinateur munis d'une connexion Internet ou intranet,
- besoin que peu de ressources soient demandées,
- besoin que l'application puisse tourner sur un système vierge,
- besoin d'une ergonomie familière.

c. Cadre de réponse

Pour chaque fonction

Solution proposée

Critère	Solution proposée
Rapidité	Requêtes optimisées
	Programme optimisé
Efficacité	Listes de choix
	Choix multiples
	Aider l'utilisateur à extraire des résultats
	Ecran intermédiaire récapitulatif
	Choix de critère de regroupement des résultats
Ergonomie	Ergonomie semblable à celle de Quadrige ²
	Style graphique Ifremer

Rapidité	Requêtes optimisées
----------	---------------------

Réaliser un jeu de requêtes sollicitant le moins possible la base de données pour des résultats les plus précis possibles.

	Requêtes optimisées
--	---------------------

Ne pas réaliser d'opérations sur la base de données autrement qu'avec des requêtes SQL. Le programme PHP ne doit que traiter les retours des requêtes, il ne doit pas réaliser d'autres opérations dessus.

Efficacité	Listes de choix
------------	-----------------

Ne pas demander à l'utilisateur d'entrer ses choix au clavier : tout doit être cliquable

	Choix multiples
--	-----------------

En plus d'être cliquables, pour chaque critère, il doit être possible de sélectionner plusieurs choix à la fois.

	Aider l'utilisateur à extraire des résultats
--	--

Restreindre les listes de choix pour ne pas faire apparaître les choix qui n'ont aucun rapport avec les autres choix précédemment réalisés. Ainsi, il est peu probable de n'extraire aucune mesure.

	Ecran intermédiaire récapitulatif
--	-----------------------------------

Réaliser un écran intermédiaire qui indique les critères pris en compte, les choix parmi ces critères, et les choix de regroupement des résultats.

	Choix de critère de regroupement des résultats
--	--

Les résultats doivent pouvoir être regroupés selon tous les critères choisis, dont des critères géographiques.

Ergonomie	Ergonomie semblable à celle de Quadrige ²
-----------	--

L'ergonomie du logiciel doit s'appuyer sur l'ergonomie du logiciel quadrige². Notamment, respecter le système d'onglets afin de trier les critères.

	Ergonomie semblable à celle de Quadrige ²
--	--

Par l'intermédiaire d'une feuille de style CSS, le programme doit obéir à la charte graphique des sites Ifremer.

Autres fonctions notables :

Les choix multiples se feront avec l'aide des listes de choix multiples des formulaires HTML :

- le maintien du clic gauche permet de sélectionner les choix traversés par le curseur,
- l'appui sur la touche SHIFT permet d'ajouter des choix en cliquant dessus,
- l'appui sur la touche CONTROL permet le retrait de choix en cliquant dessus.

Le programme en PHP doit construire des requêtes SQL selon différents critères. Ces requêtes SQL doivent être valides. La conception de l'automatisation de la construction de ces requêtes doit donc être longuement réfléchie, à l'aide d'un jeu regroupant tous les types de requête avec la totalité des critères.

Pour cela, une grammaire doit être mise en place.

Le programme en PHP doit se souvenir des listes de choix de l'utilisateur par l'intermédiaire de tableaux contenus dans des variables de session. La suppression se fera alors sur des copies de ces tableaux, et les tableaux seront alors remplacés par leur copie.

Les variables de session permettent de faire en sorte que la navigation parmi les pages se fasse sans

aucune contrainte.

En supplément de ces fonctions, une partie sécurité doit être mise en place avec une authentification à l'entrée du programme.

III) – Développement

Le fonctionnement du programme se fait en 3 étapes :

- l'utilisateur choisi quel type de requête il souhaite réaliser,
- le programme en PHP génère la requête à la base de données en fonction du contexte,
- l'utilisateur reçoit le résultat de la requête et dispose des outils pour l'exploiter.

C'est un déroulement qui s'applique à toutes les parties du programmes que ce soit dans la sélection de critères de restriction ou dans l'extraction de mesures final.

Avant d'automatiser un programme de génération de requêtes, il fallait au moins vérifier que toutes les requêtes pouvaient fonctionner. C'est ce qui m'a amené à réaliser une série de requêtes à envoyer à la base de données afin de couvrir un maximum de cas d'utilisation.

1) Le jeu de requêtes

Pour réaliser ce jeu de requêtes, j'ai commencé par le cas le plus simple , la sélection de programme, car elle retourne simplement la liste des programmes sans aucune restriction (on verra plus tard qu'il y aura néanmoins une restriction sur cette liste).

J'ai alors choisi deux programmes en fonction du nombre de résultats rattachés à ces programmes : ERIKAC et REMIC. J'ai choisi aussi une période pertinente, sélectionnant suffisamment mais pas trop de données, avec l'aide du logiciel Quadrige².

A partir de ces données, et du Modèle Logique de Données que j'ai réalisé, j'ai créé toutes les requêtes me permettant d'obtenir la liste des éléments de chaque tables relatifs aux deux programmes et à la période choisie. Je peux donc choisir des critères de restriction qui permettront la récupération de résultats. En effet, si j'avais pu choisir un service qui n'a été créé qu'en 2004, alors que la période que j'avais choisi était l'année 2000, alors, forcément, je n'aurai eu aucun résultat en retour, car aucune mesure n'a pu être saisie en 2000 par un service qui n'existait pas.

Les restrictions entre les requêtes ne se font pas qu'en fonction du programme et de la période. En effet, si j'ai choisi un service saisisseur qui n'a saisi que 25% des mesures de la période pour ces deux programmes, alors je ne pourrais choisir un support que parmi les supports de ces 25% de mesures.

C'est sur cette logique que j'ai réalisé toutes les requêtes.

Ces restrictions obéissent à une hiérarchie qui est générée en temps réel en fonction des critères choisis au fur et à mesure. Ce phénomène peut donc être représenté par un arbre, dans lequel chaque fils retourne une liste restreinte par les données du père.

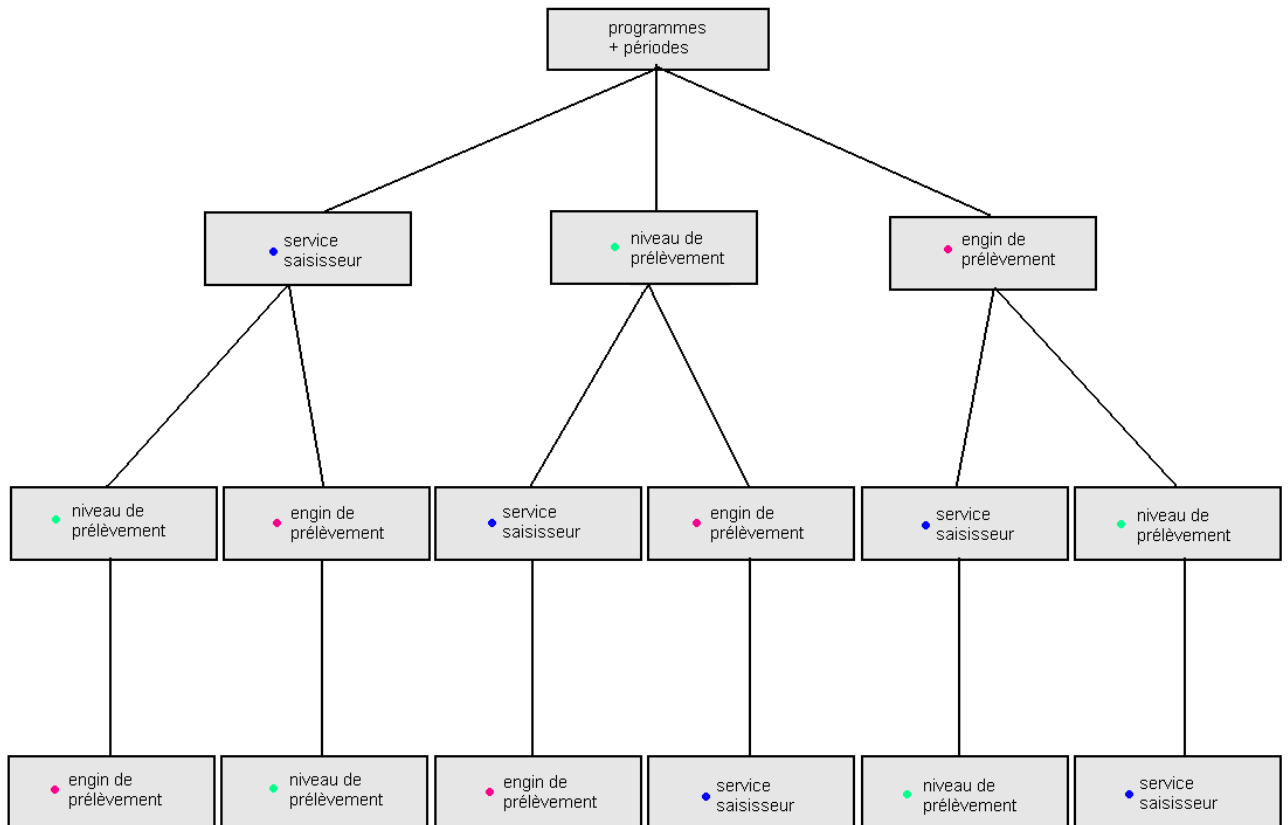


Figure 2 : Un exemple de l'arbre de restriction, avec 3 critères uniquement.

J'ai créé une norme d'écriture pour le jeu de requêtes, avec un en-tête, un nom pour le résultat obtenu, et des options de restriction basées sur l'arbre des restriction.

```

/*selection du niveau de prélèvement*/
/*prerequis : programme_choisi, date_fin_choisie, date_debut_choisie*/
/*prerequis optionnels : ensemble restreint des prélèvements(service_saisisseur_prelevement_choisi, engin_de_prelevement_choisi)*/
select depth_level_id, depth_level_nm from q2reprise.depth_level where
depth_level_id in (select depth_level_id from q2reprise.sampling_operation where
sampling_oper_id in (select sampling_oper_id from q2reprise.sampling_oper_prog where prog_cd = programme_choisi)
and survey_id in (select survey_id from q2reprise.survey where
TO_DATE(survey_dt) < TO_DATE(date_fin_choisie)
and TO_DATE(survey_dt) > TO_DATE(date_debut_choisie))
/*si on a choisi un service_saisisseur_prelevement_choisi*/
and rec_dep_id=service_saisisseur_prelevement_choisi
/*si on a choisi un engin_de_prelevement_choisi*/
and sampling_equipment_id=engin_de_prelevement_choisi
group by depth_level_id
)
order by depth_level_nm
/*ex : */
select depth_level_id, depth_level_nm from q2reprise.depth_level where
depth_level_id in (select depth_level_id from q2reprise.sampling_operation where
sampling_oper_id in (select sampling_oper_id from q2reprise.sampling_oper_prog where prog_cd = 'ERIKAC')
and survey_id in (select survey_id from q2reprise.survey where
TO_DATE(survey_dt) < TO_DATE('10/05/00')
and TO_DATE(survey_dt) > TO_DATE('01/04/00'))
/*si on a choisi un service_saisisseur_prelevement_choisi*/
and rec_dep_id=129
/*si on a choisi un engin_de_prelevement_choisi*/
and sampling_equipment_id=170
group by depth_level_id
)
order by depth_level_nm
/*-> resultat : niveau_de_prelevement_choisi*/
/*-> jeu d'essai : 1, Emergé */

```

En-tête du critère avec les éléments restrictifs qui interviennent s'ils le chemin dans l'arbre est passé par eux.

Selection de la liste des choix pour ce critère, déjà restreinte aux programmes et aux périodes, car c'est obligatoire.

S'il y a lieu, restriction aux éléments cités dans l'en-tête.

Jeu d'essai avec le programme ERIKAC, et une période d'un mois environ. J'ai utilisé un service saisisseur de prelevement et un engin de prelevement que j'ai déjà obtenus avec l'outil volumétrique de Quadrigé². "Emergé" était l'un des choix possibles parmi deux autres, qui sont les mêmes donnés par Quadrigé².

Figure 3 : Un extrait du jeu de requêtes commenté.

Ces requêtes allaient donc permettre de récupérer les choix pour les critères que l'on veut stocker.

2) Le stockage des choix pour les critères

Le programme doit être capable de récupérer les listes de choix pour chaque critère et de les stocker de façon à ce qu'elles soient utilisées par la requête d'extraction des mesures.

Pour gérer une telle liste, il faut :

- des champs d'affichage qui affichent, pour chaque critère, la liste des choix réalisés,
- un stockage des clés ou numéros d'identifiant des choix réalisés pour chaque critère, afin de ne pas accepter de doublon,
- une procédure d'ajout,
- une procédure de suppression.

Pour automatiser la gestion des critères, j'ai mis en place un tableau contenant lui-même plusieurs tableaux, identifiés par le nom de code du critère. Ces sous-tableaux sont alors structurés de façon à renseigner plusieurs informations :

- le code, qui est le nom code du tableau et aussi le code utilisé pour la création des formulaires et pour le choix de la requête à effectuer,
- les deux libellés servent à l'affichage du critère. L'utilisateur doit savoir pour quel type de critère il est en train de réaliser ses choix. Et le code est formaté pour ne pas contenir de caractères d'espace.
- lors de la récupération des choix possibles, on récupère un tableau contenant une valeur pour chaque attribut extrait des choix, puisque ceux-ci sont stockés dans des tables avec plusieurs attributs. Dans les requêtes, on peut voir qu'on extrait plusieurs attributs, comme la clef ou encore le libellé. On précise alors dans le champ affichage le numéro des attributs dont on souhaite afficher le contenu à l'utilisateur. Dans la plupart des cas, il s'agit simplement du libellé,
- de la même manière, on précise quel est le contenu à conserver dans le tableau de stockage des choix, de façon à pouvoir réafficher ultérieurement les choix réalisés,
- on précise également quel est le numéro de l'attribut qui contient la clef du choix, que l'on garde également dans un autre tableau de stockage. Ce procédé me permet d'éviter les doublons et l'identification lors de la suppression. C'est le tableau de stockage des clefs qui permettra de réaliser toutes les restrictions lors de l'extraction des mesures,
- enfin, le type indique dans quelle catégorie ce critère est rangé. Ainsi, c'est en allant dans l'onglet correspondant que l'on verra ce critère et que l'on pourra réaliser toutes les opérations dessus. L'onglet est transmis par l'URL dans la page principale, il apparaît donc dans le header, relativement au protocole HTTP. C'est une des rares utilisations de passage à l'URL que j'ai effectuées.

```
/*niveau_de_prelevement*/
$critere['niveau_de_prelevement']['code']='niveau_de_prelevement'; nom de code
$critere['niveau_de_prelevement']['libelle']='Sélection de niveaux de prélèvement'; libellé pour le bouton
$critere['niveau_de_prelevement']['libelle2']='Niveaux de prélèvement'; libellé
$critere['niveau_de_prelevement']['affichage']=array(1); numéro des cases dont le contenu est à afficher
$critere['niveau_de_prelevement']['retour']=array(0,1); numéro des cases dont le contenu est à conserver
$critere['niveau_de_prelevement']['clef']=array(0); numéro des cases dont le contenu est à conserver en tant qu'identifiant unique
$critere['niveau_de_prelevement']['type']='prelevement'; onglet dans lequel afficher ce critère
```

Figure 4 : Le tableau de paramétrage des critères dans le cas du critère « niveau de prélèvement ».

Le programme doit alors générer, sur le modèle du jeu de requêtes, des requêtes SQL avec les restrictions relatives aux choix de critères déjà réalisés.

a. la procédure d'ajout

Pour tout critère défini correctement dans le tableau des critères, un bouton est créé, et il

offre l'accès à une page qui fait appel à une fonction qui contient la liste des requêtes SQL associées à chaque critère. Donc il faut aussi créer la requête associée dans cette liste.

Si la requête est trouvée, alors on récupère une liste de choix possibles pour ce critère. Cette liste permet de faire des choix multiples avec les touches de raccourcis standard comme :

- CTRL pour sélectionner ou désélectionner plusieurs choix,
- MAJ pour sélectionner tous les choix compris entre ceux déjà sélectionnés et celui qui est cliqué,
- garder enfoncé le bouton gauche de la souris et glisser pour faire plusieurs choix. Pour se faire, j'ai utilisé un formulaire html doté d'une table d'options dont la valeur est un tableau, et qui permet alors de faire plusieurs choix.

Le programme récupère alors, grâce au tableau de critères, la liste des clés de la liste de choix et la confronte à la liste de clef des choix du critère déjà existante. Il réalise l'ajout à chaque fois que la clef n'a pas été trouvée dans la liste. Puis il ajoute le choix et toutes ses données au tableau des choix du critère. Ce second tableau ne sert qu'à l'affichage pour rappeler à l'utilisateur tous les choix qu'il a fait.

b. la procédure de suppression

La liste des choix affichés citée ci-dessus fait partie intégrante d'un formulaire qui est généré pour chaque choix dans les onglets de la page principale. Ces listes sont également des listes d'option à sélection multiple. Un bouton est créé offrant l'accès à une page de suppression. Le formulaire envoie alors la liste des choix sélectionnés et la procédure de suppression supprime simplement ces choix de la liste de choix du critères, ainsi que les clés correspondantes de la liste des clés des choix du critère. Elle réordonne finalement les deux tableaux, car les choix supprimés sont des « trous » dans les tableaux, et un tableau de données avec des zones vides est une source de lenteur pour toute opération de recherche et de traitement.

3) Génération des requêtes

Lors de la génération des requêtes il faut pouvoir restreindre à toute la liste des choix réalisés pour chaque critère, et non pas à un seul choix pour chaque critères comme on a pu le voir dans le jeu de requêtes.

Il faut pouvoir réaliser :

```
Je veux les Y de la table U
tels que
    (
        Y est un Y de la table T telle que
            X de la table T soit égal à choix_1
            Ou
            X de la table T soit égal à choix_2
            Ou
            ... (autant de fois qu'il y a de choix_i pour le critère X)
            X de la table T soit égal à choix_n
    )
Et (...)(autant de fois que l'on a de critères ou l'on a fait un choix)
```

Afin de réaliser l'algorithme générant de telles requêtes, j'ai défini une grammaire à laquelle doivent obéir les requêtes :

```
Requete <- selection + restriction
Restriction <- critere
Restriction <- Restriction + 'et' + Restriction
Critere <- type_de_critere + '=' + valeur
Critere <- Critere + 'ou' + Critere
Sélection <- (début de la sélection comme indiqué dans le jeu de requête)
```

A partir de cette grammaire découle un algorithme dont voici le fonctionnement :

```
Debut
Requete <- (début de la sélection comme indiqué dans le jeu de requête) ;
Liaison1 <- " ;
Pour chaque critère appelé critère_i
    Si il existe un choix pour ce critère
        Requete <- Liaison1 + '('
    Fin si
    Liaison2 <- " ;
    Pour chaque choix fait pour le critère_i appelé choix_i
        Requete <- Requete + Liaison2 + critere_i + '=' + choix_i
        Liaison2 <- ' ou ' ;
    Fin pour chaque
    Requete <- ')' ;
    Liaison1 <- ' et ' ;
Fin pour chaque
```

La première liaison est toujours « et » car des restrictions obligatoires existent déjà avant.

```
/*si l'on a choisi un service_analystes_mesure*/
if ($_SESSION['service_analyste_mesure_clef'] != null)
{
    $requete.=' and (';
    $liaison='';
    foreach($_SESSION['service_analyste_mesure_clef'] as $element)
    {
        $requete.=$liaison."measurement.dep_id='". $element[0]. "'";
        $liaison=" or ";
    }
    $requete.=')';
}
```

Figure 5 : Extrait du code PHP d'un bloc inspiré de l'algorithme de génération de requête

4) Retourner les résultats

A ce stade, mon programme est capable de gérer une liste de tableaux relatifs à chaque critère de restriction, et contenant des choix parmi ces critères. Il est capable de supprimer des choix et d'en ajouter.

Cette série de choix consiste en une série de critères de restriction pour la requête finale.

Pour réaliser la requête finale, j'ai procédé en cinq étapes :

- une requête finale manuelle en fonction des résultats retenus du jeu de requête,
- le développement, en m'inspirant des algorithmes de génération de requête, d'une requête qui retourne toutes les mesures,
- l'adaptation de cette requête à une autre requête qui compte ces mesures,
- la prise en compte de paramètres pour que le compte de mesures se fasse en fonction de certains critères. (ex : 'je veux le nombre de mesures par service et par semaine').

a. La requête finale manuelle

J'ai été confronté à un choix difficile qui m'a obligé à faire de lourdes répétitions pour la requête finale, mais ces répétitions sont nécessaires :

En effet, il existe deux types de résultats :

- les résultats de mesures,
- les résultats de mesures sur taxon,

ces deux types de résultats peuvent être saisis au niveau in-situ :

- passage,
- prélèvement,
- échantillon.

Cela m'a obligé à réaliser six requêtes différentes, séparées par des 'union', qui permettent la fusion simple des résultats, comme on peut le voir sur cet extrait :

```
/*----- 5 -----mesure de type passage*/ le cas d'une mesure simple, de type passage.
select meas_id, prog_cd from q2reprise.measurement, q2reprise.programme
  where object_type_cd='PREL'
  and (meas_id, prog_cd) in (select meas_id, prog_cd from q2reprise.prog_meas)
and object_id in (select survey_id from q2reprise.survey
  where
    /*criteres sur le passage*/
    /*monitoring_location*/
    (
      mon_loc_id = $_SESSION['monitoring_location']
    )
    /*fin monitoring_location*/
  and
    /*service_saisisseur_passage*/
    (
      rec_dep_id = $_SESSION['service_saisisseur_pe
    )
    /*fin service saisisseur passage*/
  /*fin criteres sur le passage*/
)
/*critere sur la mesure*/
/*service_saisisseur_mesure*/
(
  rec_dep_id = $_SESSION['service_saisisseur_mesure']
)
/*fin service_saisisseur_mesure*/
and
/*PSFM*/
```

Les restrictions
pour un passage
(extrait)

les restrictions pour
une mesure simple
(extrait)

Figure 6 : Extrait de la requête finale tirée du jeu de requêtes.

b.

Développement de la requête d'extraction de mesures.

Vu la complexité de la requête finale, le programme procède en plusieurs étapes

- Il extrait les mesures, il extrait aussi les données de groupement choisies par l'utilisateur (ex : « je veux la liste des mesures, leur service saisisseur, leur paramètre, leur support et leur date »),
- Il sélectionne ces données parmi toutes les tables qui contiennent leurs libellés,
- Il réalise les jointures entre ces données,
- Il retourne toutes les mesures, et pour chaque mesure, toutes les données qui y sont relatives.
- Il restreint cette liste de mesures et, pour cela, il fait appel à six fonctions de restriction qui sont les six cas de mesures différentes,
- Il groupe les résultats comme il était prévu lors du jeu de requêtes.
- Il trie accessoirement les résultats selon le choix de l'utilisateur

```
$requete="/*mesure_echantillon*/";
$requete.="select meas_id, object_id, object_type_cd";
$requete.=" , programme.prog_cd";
$requete.="selections($astrategie, $aparametre, $asupport, $aservice_analyste);
$requete.="tables_pas_taxon($astrategie, $aparametre, $asupport, $afraction, $amethode, $alieu, $adate);
$requete.="une autre fonction permet de récupérer les tables où sont rangées ces données.
where object_type_cd='ECHANT'";
    if ($astrategie)
        $requete.=" and strategy.prog_cd=programme.prog_cd
                    and strategy.strat_id in (select strat_id from
in (select survey_id from q2reprise.sampling_operation where sa
= object_id)))";
    if ($aparametre)
        $requete.=" and measurement.par_cd=parameter.par_cd";
    if ($asupport)
        $requete.=" and measurement.matrix_id=matrix.matrix_id";
    if ($afraction)
        $requete.=" and measurement.fraction_id=fraction.fraction_id";
    if ($amethode)
        $requete.=" and measurement.method_id=method.method_id";
    if ($alieu)
        $requete.=" and monitoring_location.mon_loc_id in (select m
q2reprise.sampling_operation where sampling_oper_id in (select
    if ($adate)
        $requete.=" and survey.survey_id in (select survey_id from
from q2reprise.sample where sample_id = object_id)";
    if ($aniveau)
        $requete.=" and depth_level_id in (select depth_level_id fr
sampling_oper_id from q2reprise.sample where sample_id = object
    if ($aservice_saisisseur)
        $requete.=" and department.dep_id = measurement.rec_dep_id";
    if ($aunite)
        $requete.=" and unit_id in (select unit_id from q2reprise.P
measurement.matrix_id and PMFM.fraction_id = measurement.fracti
        $requete.="
        and (meas_id, programme.prog_cd) in (select meas_id, prog c
$requete.="criteres_mesure_echantillon() ;
$requete.="

UNION

```

une mesure simple de type échantillon

partie sélection : on sélectionne des données obligatoires, comme le numéro de la mesure ou le nom du programme. Puis on fait appel à une fonction pour sélectionner les données voulues.

Partie jointure : On fait une jointure entre toutes les données voulues et les mesures à extraire.

enfin, on fait appel à la fonction de restriction pour une mesure de type échantillon.

Puis, on passe à l'extraction de mesure de type taxon, et de type échantillon, avec un union.

Figure 7 : Extrait de la requête finale en PHP.

c. Le compte des mesures.

Avec la possibilité de retourner la liste complète des mesures restreinte à tous les critères de restriction, je pouvais finalement les compter. Mais pour que le compte se fasse en fonction de certains critères, il a fallu procéder ainsi :

De la liste des mesures restreintes, je récupère les numéros d'identifiant des mesures, ainsi que toutes les entités faisant office de critère de groupement pour les compter. Puis, je groupe ces résultats afin de ne pas avoir de doublon. Il ne me reste alors qu'à compter ces lignes de résultats en sélectionnant simplement toutes les données autres que le numéro de la mesure, et en faisant un compte (count(*) en SQL).

etape 1 : extraire les mesures				
Parametre	Methode	numero mesure	type de mesure	programme
parametre 1	methode 1	1	echantillon	remic
parametre 1	methode 1	2	prelevement	remic
parametre 1	methode 2	3	prelevement	remic

etape 2 : ne garder que les numeros d'identifiant des mesures, et les parametre pour		
Parametre	Methode	numero mesure
parametre 1	methode 1	1
parametre 1	methode 1	2
parametre 1	methode 2	3

etape 3 : compter le nombre de lignes identiques		
Parametre	Methode	compter
parametre 1	methode 1	2 (il y a 2 lignes comme celle-ci)
parametre 1	methode 1	2 (il y a 2 lignes comme celle-ci)
parametre 1	methode 2	1 (il y a 1 lignes comme celle-ci)

etape 4 : grouper, de façon à supprimer toute redondance.		
Parametre	Methode	compter
parametre 1	methode 1	2 (lorsque l'on groupe ces résultats)
parametre 1	methode 2	1

Figure 8 : Exemple du comptage des mesures en fonction deux critères.

d. L'automatisation des paramètres de groupement

Lorsque l'utilisateur prépare ses résultats il dispose d'un formulaire qui lui permet de choisir les critères de groupement pour le compte des résultats. Il peut également choisir un critère pour trier. Cette liste de critères est envoyée à la page qui génère les résultats.

Pour que les fonctions qui génèrent les requêtes prennent en compte ces critères, j'ai décidé de les passer en paramètre. C'est donc dans la fonction même que la requête se génère en fonction de tous les critères voulus.

Il y a eu cependant un critère de groupement plus compliqué à mettre en œuvre, celui du programme. Dans Quadrigé² un résultat peut être rattaché à un ou plusieurs programmes. Le rapport de réunion disait :

« Lorsque des résultats sont 'multi-programmes' l'état de restitution doit faire apparaître tous les programmes sous la forme d'une ligne par programme. Le nombre de résultats exprimés par programme est exclusif. »

Par exemple, si REPHY a 17 résultats exclusifs, et RNOHYD en a 8 exclusifs, mais que 5 résultats sont communs à REPHY et RNOHYD, alors il faut afficher ceci :

REPHY	17
REPHY+RNOHYD	5
RNOHYD	8

Il s'est avéré qu'il n'était pas possible d'obtenir un tel résultat avec une simple requête SQL. En effet, la fusion de deux attributs en un, tout en conservant des données issues d'un seul attribut, le tout dans une même colonne, est une opération ambiguë pour un traitement de base de données.

J'ai donc dû créer un algorithme en PHP qui, à partir de la liste des mesures extraites avec leur programme associé, fait le compte des mesures exclusives à chaque programme, puis fait le compte des mesures relatives à chaque n-uplet de programmes.

--> voir annexe -_ algorithme_multiprogramme

Pour trouver un jeu d'essai qui me permettrait de valider cet algorithme, j'ai créé moi-même de nombreuses mesures fictives en PHP afin de tester des cas complexes, avec des triplets de programmes et d'autres situations plus compliquées.

Pour ensuite tester l'algorithme dans une situation réelle, j'ai dû chercher des cas d'utilisation.

A l'aide d'une requête :

```
select meas_id, somme from (select meas_id, count(*) as somme from q2reprise.prog_meas group by (meas_id)) where somme >=2
```

(sélection d'un numéro de mesure et d'une somme égaux à un numéro de mesure et un compte de ces numéros dans la table qui joint les mesures avec les programmes, ou le compte de ces mesures est égal à au moins 2.)

J'ai trouvé une mesure, la 60000709, qui appartient à deux programmes. De là, j'ai enquêté afin de trouver les programmes correspondant, et les périodes. Je n'avais plus qu'à entrer ces programmes et ces périodes dans la sélection de critères pour obtenir le jeu d'essai voulu. Voici le résultat :

Nombre total de mesures : 36	
Nombre de mesures par ...	
programme	nombre de mesures
OMEGA	27
REMIC	6
OMEGA + REMIC	3

Il y avait 39 mesures au total, dont 6 étaient communes à deux programmes, soit 36 mesures différentes au total. 27 exclusives à OMEGA, 6 à REMIC, et 3 communes aux deux.

5) L'interface

J'ai étudié la charte graphique des sites de l'Ifremer. J'ai alors créé l'interface du programme sur ce modèle. Pour cela, j'ai utilisé une feuille de style CSS et tout le code « html » généré par le programme en PHP a été balisé pour qu'il puisse être interfacé aisément.

Sur le principe de l'arbre des widgets, j'ai construit un arbre des éléments graphiques de l'interface du site.

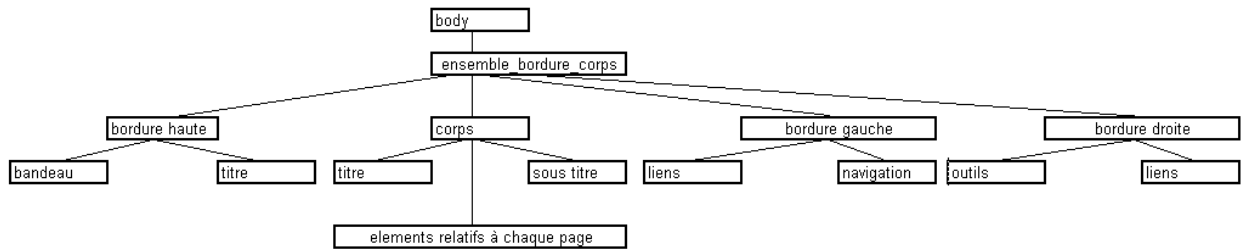
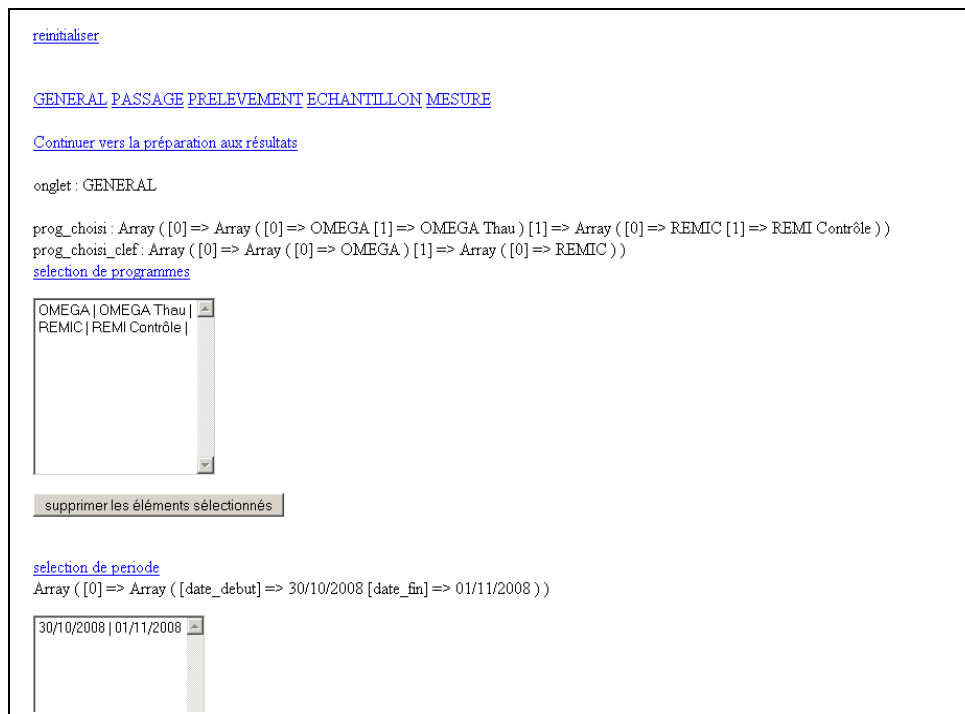


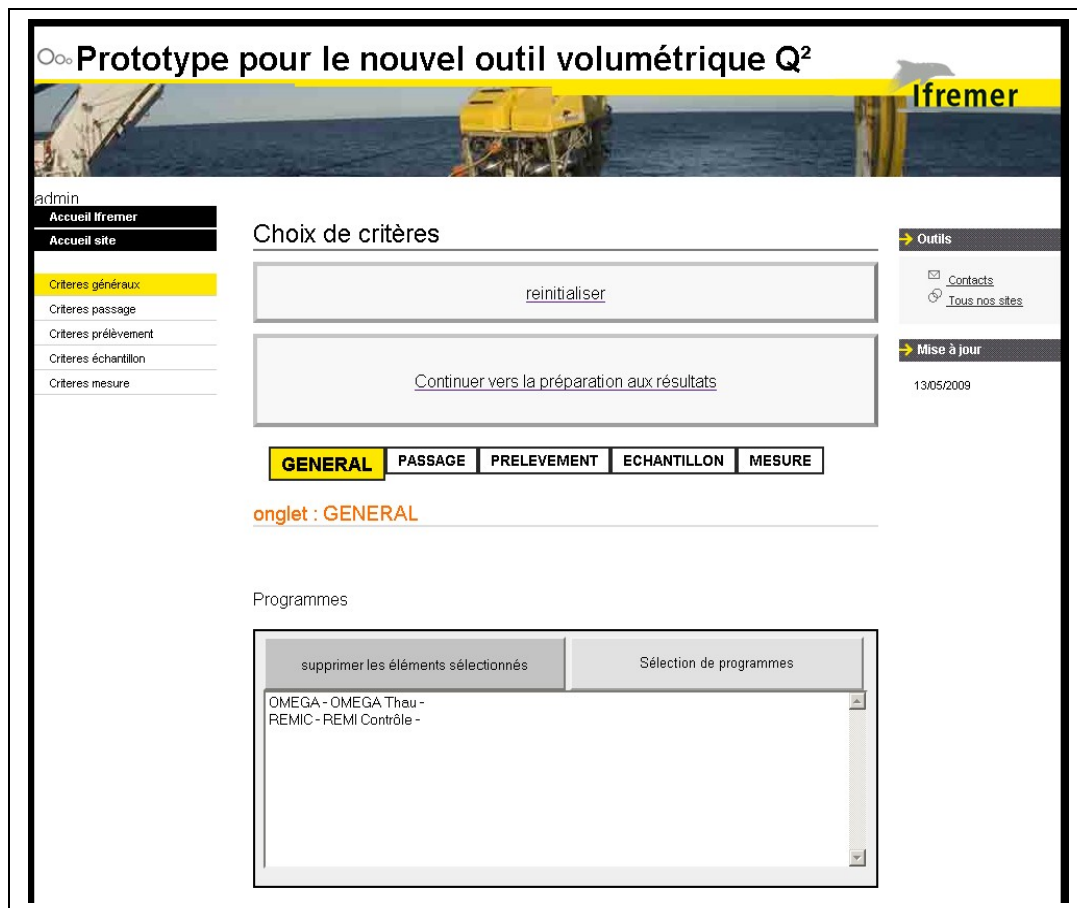
Figure 9 : Arbre des éléments graphiques de l'interface.

A partir de cet arbre, et en m'inspirant du site de référence du look des sites Ifremer, j'ai détaillé ma feuille de style.

Je suis passé de cette interface sommaire de développement ...



... à cette interface ergonomique, et accessible par l'utilisateur.



6) Les finitions

a. L'extraction au format « csv »

Pour afficher les résultats de nombre de mesures à l'utilisateur, j'ai opté pour un tableau HTML encapsulé dans une balise « div » dont le style d'overflow est automatique. Ce principe permet d'afficher concrètement un tableau contenu dans une sous-fenêtre de taille fixe et pourvue d'ascenseurs, au sein même de la page web. Dans ce cas, peu importe la taille en longueur ou en largeur du tableau de résultats, la page ne sera pas déformée.

Ces tableaux sont construits de façon classique, où chaque colonne est un attribut (la liste des critères de groupement, puis le nombre de mesures) et chaque ligne correspond à un groupement de critères et son nombre de mesures correspondant.

Dans le format « csv », on construit le tableau en séparant chaque colonne par un point-virgule, et chaque ligne par un retour chariot. La construction d'un tableau « html » se faisant de façon similaire, j'ai pu adapter le code pour proposer une extraction des résultats au format « csv ». Ce format est très intéressant puisqu'il permet de lire et traiter le tableau avec des tableurs comme Excel.

b. L'identification à l'entrée du programme

Chaque utilisateur de Quadrigé² n'a accès aux données que d'un nombre restreint de

programmes. Une fonction permet de vérifier l'appartenance de la personne qui se connecte à l'annuaire LDAP. Le système bloque l'accès aux résultats à tout utilisateur externe au réseau Ifremer.

Une fois la page d'identification passée, le numéro d'identifiant est récupéré par la première requête, qui est chargée de retourner une liste de programmes restreinte à ce numéro d'identifiant.

L'utilisateur identifié a alors la possibilité de compter les résultats rattachés aux programmes auxquels il a droit.

Bilan

Au cours du projet, j'ai rencontré de nombreux obstacles. La principale difficulté a été de comprendre et de me représenter la base de données Quadrige². Le modèle logique de données établi en début de stage et le jeu de requête réalisé m'ont permis de surmonter cette difficulté.

La modélisation d'un planning en début de projet m'a permis de respecter la contrainte temps afin d'arriver au terme du projet avant la présentation aux futurs utilisateurs, le 11 Juin 2009. Lors des phases de tests, en fin de planning, j'ai dû corriger plusieurs problèmes mineurs, mais j'avais prévu des marges afin d'avoir le temps de les résoudre.

La réalisation de ce projet m'a fait étudier chaque étape dans le processus de développement d'une couche supérieur à un jeu de requête dans une base de données. De plus, les aspects communication entre le programme en PHP et la base de données Oracle, l'interface web et la structure de données m'ont permis d'établir un lien entre les différents enseignements reçus à l'IUT.

Dans l'avenir, il pourrait être envisageable de développer une application java en s'inspirant du prototype que constitue mon programme, et l'extraction du nombre de mesures pourrait être une évolution du logiciel Quadrige².

Conclusion

L'objectif de départ, construire un prototype pour un outil de volumétrie sur la base de données Quadrige², a été atteint. Cet outil a été présenté aux futurs utilisateurs. Cela leur a permis de faire déjà quelques remarques mineures que j'ai pu prendre en compte avant la fin du stage.

Il a été très agréable de travailler au sein de ce service à l'Ifremer. J'ai reçu un soutien important de la part des agents du service, de Jean-Claude Masson et Emilie Gauthier particulièrement, si bien qu'il a été un plaisir de développer ce projet. Par ailleurs, l'environnement de travail, tout à fait respectueux des critères idéaux que l'on a vus en cours d'Info et société, a été bénéfique pour le bon déroulement de mon stage. Je me suis renseigné sur la vie en entreprise auprès des agents, et en écoutant les conversations, j'en ai appris un peu plus sur le fonctionnement d'un service, ce qui devrait faciliter ma future insertion professionnelle, quand j'aurai fini mes études.

Bibliographies

The PHP Group. Pages de manuel consultées du 28/04/09 au 02/06/09. *PHP : Hypertext Processor*, (En ligne). <http://www.php.net>

JACQUIN, Christine. *Cours de Technologie Web (1 et 2)*, notes rédigées en 2008.

HADJ RABIA, Abdelghani. *Cours de Base de Données 2*, notes rédigées en 2009.

HÛE, Jean-François. *Cours de Complément Algorithmique*, notes rédigées en 2009.

Glossaire

Algorithme :

Un algorithme est un processus systématique de résolution, par le calcul, d'un problème permettant de présenter les étapes vers le résultat à une autre personne physique (un autre humain) ou virtuelle (un calculateur). En d'autres termes, un algorithme est un énoncé d'une suite d'opérations permettant de donner la réponse à un problème.

Bug :

Un bug ou bogue informatique (francisation de l'anglais bug, " insecte ") ou est une déficience dans un programme l'empêchant de fonctionner correctement.

Client – Serveur :

L'architecture client/serveur désigne un mode de communication entre plusieurs ordinateurs d'un réseau qui distingue un ou plusieurs postes clients du serveur : chaque logiciel client peut envoyer des requêtes à un serveur. Un serveur peut être spécialisé en serveur d'applications, de fichiers, de terminaux, ou encore de messagerie électronique.

CSV :

Le format de fichier CSV, de l'anglais Coma Separated Values (valeurs séparées par une virgule), contient des tableaux dont chaque colonne est séparée par une virgule ou un point-virgule, et chaque ligne est séparée par un retour chariot.

Header :

En technologie de l'information, le header renvoie aux données contenues au début d'un bloc de contenu à stocker ou transmettre. En transmission de données, les données qui suivent le header sont souvent appelées charge utile ou body.

HTML :

Ou Hyper Text Markup Language. C'est le langage de paramétrage des pages Internet. Un site web est constitué de documents HTML, que l'on appelle des pages.

Interface :

Une interface définit la frontière de communication entre deux entités, comme des éléments de logiciel, des composants de matériel informatique, ou un utilisateur. Elle se réfère généralement à une image abstraite qu'une entité fournit d'elle-même à l'extérieur.

Protocole :

Dans les réseaux informatiques et les télécommunications, un protocole de communication est une spécification de plusieurs règles pour un type de communication particulier. Initialement, on nommait protocole ce qui est utilisé pour communiquer sur une même couche d'abstraction entre deux machines différentes.

Serveur :

ordinateur central d'un réseau informatique où sont stockées les données et applications accessibles par les autres postes connectés.

Spécifications :

Les spécifications sont un ensemble de documents qui – par des textes et des diagrammes - décrit de manière formelle et exhaustive le produit informatique à réaliser. La rédaction de la spécification est la première étape du développement d'un logiciel.