Appel à projet PostgreSQL Francophone

De PostgreSQL à PostgreSQL_f

Édition : 2011
Acronyme du projet : PostgreSQL_f

Table des matières

I. Introduction

I.1) L'équipe de recherche PILGRIM

Le projet Pilgrim est une équipe de recherche au sein de l'IRISA - Institut de Recherche en Informatique et Systèmes Aléatoires - (unité mixte de recherche regroupant le CNRS, l'Université Rennes 1, l'INSA à Rennes et l'ENS Cachan (antenne de Bretagne)). Pilgrim est un acronyme pour la PrIse en compte de la GRadualité, de l'Imprécision et de la Médiation dans les systèmes de gestion de bases de données.

L'équipe PILGRIM est composée de 2 professeurs des universités (Patrick Bosc et Olivier Pivert), 5 maîtres de conférences (Daniel Rocacher titulaire d'une habilitation à diriger des recherches, Ludovic Lietard, Allel Hadjali, Hélène Jaudoin et Grégory Smits), 3 doctorants (Amine Mokthari, Nouredine Tamani, Katia Abbaci).

Les travaux de l'équipe PILGRIM se situent à l'intersection de différents domaines de recherche : base de données, extraction et gestion de connaissances, recherche d'information, systèmes incertains, etc. Ces travaux ont donné lieu à de nombreuses publications scientifiques (plusieurs centaines d'articles dans des livres, journaux et conférences internationales) et pédagogiques (ouvrage « Gradualité et imprécision dans les bases de données », édition ellipses). Les axes de recherche de l'équipe s'orientent autour de :

- l'interrogation flexible de bases de données (requêtes graduelles),

- la modélisation et la manipulation de données mal connues,
- la médiation flexible de données.

I.2) SQLf

Le langage SQL s'est imposé comme un standard d'interrogation dans la plupart des Systèmes de Gestion de Base de Données Relationnelles (SGBDR), PostgreSQL notamment. Bien que son efficacité ne puisse être remise en cause, ce langage souffre tout de même d'un manque de flexibilité, tant sur le plan syntaxique que sémantique. Sur ce dernier point, l'équipe PILGRIM et plus particulièrement Patrick Bosc et Olivier Pivert ont proposé une extension de SQL, baptisée SQLf, offrant un langage flexible d'interrogation d'une base de données relationnelles « classique ». Cette flexibilité concerne notamment la possibilité d'introduire des conditions vagues dans les requêtes qui apparaissent indispensables lorsque l'utilisateur n'est pas en mesure de traduire de manière booléenne (en « tout ou rien ») ses attentes et contraintes, celles-ci étant elles-mêmes de nature imprécise. La flexibilité introduite dans ce langage d'interrogation traduit également une notion de *préférence* permettant une distinction qualitative des éléments sélectionnés et ainsi d'ordonner les réponses par leur degré de satisfaction vis-à-vis des préférences de l'utilisateur. En fonction d'un nombre attendu de résultats, ces préférences sont également exploitées pour suggérer des réponses indirectes.

La théorie des ensembles flous est utilisée comme cadre formel pour étendre les opérateurs ensemblistes et relationnels utilisés dans SQL. L'article [BOSC1995] ainsi que l'ouvrage [BOSC2004] présentent la façon dont les opérateurs usuels ensemblistes (intersection, union, différence) et relationnels (sélection, projection, jointure, division) ont été étendus aux relations graduelles, i.e. floues. Pour une description précise de ces extensions, veuillez vous référer aux différents articles [BOSC1995, BOSC1996, BOSC1997, BOSC1998, BOSC1999, BOSC2011].

D'un point de vue plus pratique et illustré, SQLf s'appuie sur le fait que les ensembles flous modélisent des qualificatifs du langage naturel qui sont par nature imprécis et subjectifs. Ainsi l'âge d'une personne qui est précisément décrite par un attribut numérique *âge* dans une base de données peut servir à caractériser des propriétés telles que *jeune* ou *âgé*. Ces deux qualificatifs correspondent à des notions vagues et subjectives modélisées dans SQLf par des ensembles flous , chacun étant caractérisé par une fonction d'appartenance. Cette fonction d'appartenance qualifie par un score normalisé dans l'intervalle [0,1] l'appartenance d'un tuple à ce prédicat. Ainsi, contrairement à SQL qui repose sur une logique booléenne, un tuple appartiendra de manière graduelle à l'ensemble des réponses. Ce score d'appartenance offre directement une information utile pour le classement qualitatif des réponses vis-à-vis de la requête flexible. Outre, la prise en compte de prédicats flous, SQLf introduit des modificateurs qui expriment des adverbes de la langue naturelle, tels que *très*, *assez*, *peu*, etc..

SQLf reprend la structure de son bloc d'interrogation de SQL tout en introduisant de la gradualité partout où elle présente un intérêt. Ainsi le bloc d'interrogation de SQLf est le suivant :

select [**distinct**] [n | t | n, t] attributs **from** relations **where** cond-floue;

La clause **where** peut ainsi contenir des conditions booléennes (*département = 'finance'*) et des conditions floues (âge *is 'jeune'*). La clause *select* est également complétée par des paramètres indiquant la quantité de résultats attendus (n) et/ou la qualité, c'est-à-dire le degré de satisfaction minimal attendu (t).

Voici un exemple d'un telle requête :

select 0.6 emp#, e-nom **from** emp E, dept D **where** E.dep# = D.dep# **and** E.âge **is** 'jeune' **and** D.budget **is** 'important';

Le résultat d'une telle requête n'est plus une relation classique, mais une relation floue où chaque tuple est associé à un degré de satisfaction des conditions spécifiées dans la clause **where**.

SQLf n'est pas l'unique extension flexible de SQL [BOSC1993, KACPRZYK1995, TAHANI1977], mais certainement la plus aboutie théoriquement et d'un point de vue méthodologique.

L'objectif de ce projet est d'aboutir à la première implémentation d'une telle extension en s'appuyant sur le SGBD PostgreSQL. Ce projet vise également à promouvoir la dynamique de la communauté francophone de développement de PostgreSQL en proposant un démonstrateur d'interrogation flexible d'une base de données relationnelle à l'aide de motifs linguistiques en français.

II. Description du projet

II.1) Objectifs et caractère ambitieux du projet

À travers les publications dans le domaine des bases de données ou de la recherche d'information, ainsi que différents partenariats universitaires ou industriels, nous avons pu constater que l'absence d'implémentation de SQLf était un frein important à son essor et sa reconnaissance. Les contextes applicatifs pouvant exploiter une implémentation publiquement accessible, stable et efficace sont nombreux et variés, allant des systèmes de recommandation de contenu aux systèmes d'analyse statistique et décisionnelle de données métier.

Outre les difficultés liées à l'implémentation qui seront décrits dans la section II.3), l'impact et la visibilité de ce développement seront conditionnés par l'efficacité du traitement de requêtes flexibles. Il sera ainsi nécessaire d'évaluer à travers des expérimentations les temps d'exécution de requêtes flexibles par un SGBD intégrant dans son plan d'exécution une prise en compte de conditions graduelles et de les comparer avec un traitement *a posteriori* de la gradualité utilisant un SGBDR classique (cette dernière méthode étant pour le moment la seule disponible).

II.2) Retombées scientifiques et intérêt pour la communauté francophone de PostgreSQL

Dans un cadre universitaire de recherche, la promotion de nos travaux repose essentiellement sur la publication d'articles dans des conférences et journaux internationaux et nationaux. Cette démarche s'inscrit parfaitement dans la philosophie du logiciel du monde libre où l'objectif est de faire bénéficier à un public le plus ouvert possible les travaux réalisés. Cette volonté explique notamment notre choix pour PostgreSQL qui bénéficie d'une indépendance commerciale désormais rare. Depuis plusieurs années, l'ensemble des prototypes de recherche développés dans l'équipe PILGRIM utilise ce SGBD [BOSC2010].

Nous proposons donc d'associer la communauté francophone de développement de PostgreSQL à ces travaux ainsi qu'à leur publication et de fournir en résultat une version sous licence BSD de PostgreSQL prenant en compte des requêtes flexibles telles qu'elles sont définies dans le framework SQLf. Ce résultat sera également associé à un démonstrateur en français relié à une interface utilisateur d'interrogation (Section II.3).

II.3) Difficultés et acquis

Des premières expérimentations réalisées par Amine Mokthari doctorant dans l'équipe PILGRIM, ont été conduites sur l'implémentation de SQL_f à partir de PostgreSQL version 8.4. L'objectif de ces travaux étaient d'évaluer la faisabilité de cette implémentation et également de

découvrir la structuration du code de PostgreSQL.

La prise en compte de conditions floues dans la clause **where** entraîne des modifications du fonctionnement de la version de base dans PostgreSQL. Ainsi, pour qu'une requête telle que :

***select * from** employe **where** salaire **is** 'élevé' **and** âge is 'jeune';*

soit exécuté, il faut à la fois que PostgreSQL dispose de la définition des prédicats *'élevé'* et *'jeune'*, mais également qu'il puisse combiner les degrés de satisfaction attachés aux différentes contraintes graduelles pour calculer un score global pour chaque tuple. Ainsi, le degré de satisfaction pour la conjonction *salaire is 'élevé'* ***and*** *âge is 'jeune'* repose sur l'utilisation d'une norme triangulaire, le minimum entre le degré de satisfaction *salaire is 'élevé'* et celui de â*ge is 'jeune'*. La co-norme triangulaire maximum est utilisée dans le cas d'une disjonction.

D'un point de vue plus technique, nous avons expérimenté une version préfixée des prédicats flous pour obtenir des requêtes telles que :

***select * from** employe **where** élevé(salaire) **and** jeune(âge);*

La définition des termes flous *'élevé'* et *'jeune'* peut ensuite être donnée comme une fonction implémentée en PL/pgsql. La prise en compte de la notation « post-fixée » définie dans SQL_f pourrait reposer une étape de transformation syntaxique définie dans le parseur PostgreSQL. Ceci n'a pas encore été étudié. Le code suivant illustre l'implémentation du terme flou en tant que fonction PL/pgsql :

Code PL/pgsql :

```
CREATE OR REPLACE FUNCTION public.jeune(age integer)
 RETURNS real
 LANGUAGE plpgsql
AS $function$
DECLARE
    support1 real := 0;
    core1   real := 0;
    core2   real := 20;
    support2 real := 30;
BEGIN
        if ((support1 < cost) AND (cost < support2 )) then
        if ((core1 <= cost) and (cost <= core2)) then
            return 1;
        ELSE
            return ((cost - support2)/(core2 - support2));
        END IF;

    ELSE
        RETURN 0;
    END IF;
END;
```

Une telle fonction implémentant un terme flou retourne un degré de satisfaction normalisé dans l'intervalle [0,1]. Ce degré est ensuite attaché à chaque tuple par l'intermédiaire d'un attribut score qui est ajouté si celui-ci n'existe pas encore.

Les scores retournés par chaque condition floue ou booléenne sont ensuite combinés suivant l'expression fournie dans la clause **where**. Notons que d'un point de vue théorique, la sémantique des opérateurs **and** et **or** sont définis dans la logique floue comme des normes et co-normes. L'association du degré calculé à chaque tuple est réalisée directement dans le module executor (fonction *ExecQual*). Par défaut, l'attribut score (de type *real*) est utilisé pour stocker le score. La

définition des opérateurs **and**, **or** et **not** sont également implémentés dans le module executor (par les fonctions *ExecEvalAnd*, *ExecEvalOr* et *ExecEvalNot*). SQL_f introduit également de nouveaux connecteurs flous apportant une sémantique enrichie par rapport aux connecteurs existants **AND** et **OR**. Il est ainsi possible dans SQL_f de combiner des prédicats en introduisant un compromis matérialisé par des moyennes, pondérées ou non, des propositions quantifiées, etc.

Pour illustrer le calcul de ce score, le tableau 1 propose une extension possible de la table *Employé* et la figure 1 donne la définition des ensembles flous '*jeune'* et *'élevé'* pour respectivement les attributs *âge* et *salaire*.

Emp#	Âge	Salaire	Departement
1	25	30000	Administration
2	45	39000	Développement
3	28	29000	Technique

Tableau 1: extension de la table Employé

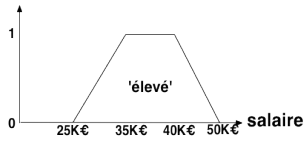

Figure 1 : prédicat flou 'élevé' pour l'attribut salaire

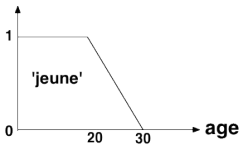

Figure 2 : prédicat floue 'jeune' pour l'attribut âge

À partir de l'extension du Tableau 1 et de la définition des prédicats 'jeune' et 'élevé' des Figures 1 et 2, nous pouvons calculer le résultat (Tableau 2) de la requête introduite précédemment : ***select*** * ***from*** *employe* ***where*** *élevé(salaire)* ***and*** *jeune(age);*

Emp#	Age	Salaire	Département	**Score** min(salaire is 'élevé' , age is 'jeune')
1	25	30000	Administration	min(0.5, 0.5) = 0.5
2	45	39000	Développement	min(1, 0) = 0.0
3	28	29000	Technique	min(0.6, 0.2) = 0.2

Tableau 2: Relation floue constituant le résultat d'une requête flexible

Ces premiers travaux sur la prise en compte de prédicats flous ont été très encourageants et nous ont permis de constater la clarté et la qualité du code de PostgreSQL. Les premiers résultats ne sont cependant restés qu'au statut de prototype et beaucoup de tests de performance et de stabilité sont encore à effectuer. De même, aucune documentation n'a encore été rédigée sur ces aspects.

De plus, SQL_f introduit une version flou de la clause **group by** qui a été notamment décrite dans

[BOSC2010]. L'implémentation de cette extension au cœur de PostgreSQL n'a pas encore été étudiée. La principale différence avec la clause **group by** de SQL est l'introduction dans le corps de la requête d'une partition floue du domaine de définition d'un attribut. Cette partition est utilisée pour effectuer le regroupement des tuples, où un tuple peut désormais appartenir à plusieurs groupes avec des degrés différents. Cette extension de la clause **group by** est notamment très utile pour analyser des données et générer des statistiques. Ainsi, si nous souhaitons savoir pour quelles tranches d'âge les salaires d'une entreprise sont élevés, nous pouvons utiliser la requête suivante :

```
select label(âge), count from Emp
where salaire is 'élevé' group by label(âge)
using part(age) = {'jeune', 'moyen', 'âgé'}.
```

où *jeune*, *moyen* et *âgé* correspondent aux étiquettes linguistiques des éléments d'une partition floue définie sur l'attribut *age*. La Figure 3 illustre un exemple de partition floue sur l'attribut â*ge*.

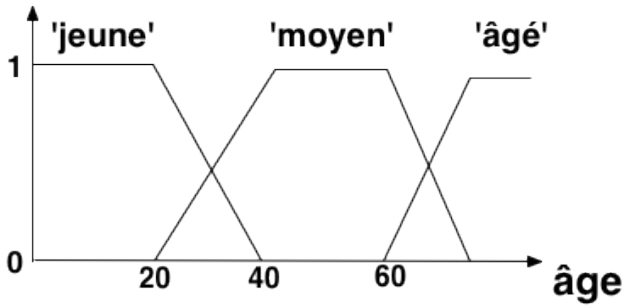

Figure 3 : exemple de partition floue du domaine de définition de l'attribut âge

Le résultat d'une requête faisant intervenir groupement flou est constitué des différents étiquettes de la partition utilisée et des résultats des fonctions d'agrégat appliquées aux éléments qui satisfont ces prédicats.

Parmi les autres éléments définis dans SQL_f qui sont encore à étudier et à implémenter, on peut citer la prise en compte de quantificateurs flous. Ces quantificateurs peuvent intervenir à différents niveaux de la requête pour étendre sa sémantique :

- dans la clause where pour étendre les connecteurs usuels de conjonction et de disjonction. Ceci permet notamment de spécifier des requêtes telles que :

***select * from** employe **where** la_plupart(salaire **is** 'élevé' **and** age **is** 'jeune' **and** département **is** 'hautement_qualifié' **and** ...);*

L'usage de quantificateurs flous est par exemple très utile pour éviter de retourner un ensemble vide de résultats.

- pour l'imbrication de requêtes. Ces quantificateurs flous s'appliquent d'une manière similaire à ceux existants, à savoir ***any*** et ***all***, mais permettent d'exploiter la relation floue résultant d'une sous-requête dans la requête principale.

***select * from** employe **where** salaire >la_plupart (**select** salaire **from** employe);*

- pour la restriction de groupes. En complément d'une clause de groupement étendue à des conditions d'appartenances graduelles et non plus booléennes, la restriction sur les groupes peut comporter une condition floue éventuellement quantifiée.

***select** 5, dep# **from** emp **group by** dep#*

***having** la_plupart **are** (salaire **is** 'bien_payé');*

Bien que l'opération de division ne soit pas directement implémentée dans la plupart des SGBD commerciaux, différentes méthodes de division étendues aux relations floues ont été proposées, dont l'implémentation pourra être étudiée. Cet opérateur de division offrirait la possibilité de formuler des requêtes visant par exemple à « trouver les magasins auxquels on a commandé en quantité *importante* tous les produits *chers* ». SQL_f utilise un opérateur d'inclusion graduelle, noté ***contains***$_i$ dans la requête ci-dessous, pour exprimer la division :

select *m#* ***from*** *cmg* ***where*** *quantité* ***is*** *'importante'* ***group by*** *m#*

having set*(p#)* ***contains***$_i$ ***(select*** *p#* ***from*** *prd* ***where*** *prix* ***is*** *'élevé');*

Ce projet a donc pour ambition de poursuivre l'implémentation de SQL_f au cœur du SGBD PostgreSQL en s'appuyant notamment sur nos premières expérimentations concernant les conditions floues. Une fois l'implémentation de PostgreSQL_f achevée, documentée et mise à disposition du public, nous proposerons un démonstrateur qui associera ce projet et un développement concernant une interface intuitive et personnalisée d'interrogation flexible de bases de données. L'interface graphique d'interrogation flexible permettra ainsi à un utilisateur de définir son propre jeu de prédicats et modificateurs flous pour étendre notamment un ensemble d'éléments par défaut prédéfinis. L'implémentation de cette interface sera réalisée par deux étudiant stagiaire que nous accueillerons de juillet à août 2011.

II.4) Échéancier prévisionnel

Les membres de l'équipe PILGRIM ont un emploi du temps entièrement occupé par leurs activités de recherche et d'enseignement. Le pilotage de ce projet sera effectué par Olivier Pivert et Grégory SMITS (voir Section IV) et le développement confié à un étudiant ingénieur dans le cadre d'un stage de fin d'étude. À partir des premières implémentations effectuées et d'une estimation de la difficulté des tâches qu'il reste à accomplir, nous avons établi un échéancier prévisionnel pour ce stage d'une durée de 4 mois qui repose sur les tâches suivantes :

- T1 : étude du langage SQL_f
  - Au cours de cette tâche l'étudiant stagiaire s'assurera d'avoir parfaitement compris les différents composants de SQL_f et les différences avec SQL.
- T2 : étude de la structure du code de PostgreSQL 9.0
  - À partir des différentes documentations de PostgreSQL que nous avons déjà recensées et en s'appuyant sur l'expérience de la communauté francophone de développement de PostgreSQL, cette tâche permettra au stagiaire de comprendre le fonctionnement du code PostgreSQL et les connexions entre les différents éléments. Au cours de cette tâche, l'étudiant mettra également en place et se familiarisera avec un environnement graphique de développement (*Eclipse*) ainsi que le débogueur *gdb*.
- T3 : étude et test des premières implémentations effectuées sur PostgreSQL 8.4
  - Sur une machine virtuelle sera installée la version 8.4 de PostgreSQL dans laquelle les prédicats flous sont actuellement gérés. Une analyse critique de cette première implémentation sera effectuée avant de suggérer des éventuelles modifications et d'effectuer des tests de stabilité.
- T4 : analyse et proposition de schémas d'intégration de SQL_f dans PostgreSQL 9.0
  - À partir des tâches précédentes, une analyse et un plan d'implémentation seront suggérés pour porter les développements existants vers la nouvelle version de PostgreSQL et

intégrer les autres éléments de SQL_f.

- T5 : implémentation
  - L'analyse de T4 a déterminé la façon dont les différents éléments de SQL_f devaient être implémentés et les emplacements du code de PostgreSQL affectés par ces développements. L'implémentation sera progressive, élément de langage par élément de langage : prédicats flous, conjonction/disjonction de conditions booléennes/floues, quantificateurs, groupement flou, division, etc. Chaque implémentation donnera lieu à des tests de stabilité, de non-régression et de performance (complexité temporelle et spatiale).
- T6 : documentation et publication
  - En utilisant les standards de documentation de PostgreSQL, une documentation technique sera rédigée. Un paquet contenant sources et documentations sera ensuite construit pour publication sous licence BSD.
- T7 : démonstrateur
  - L'interface utilisateur d'interrogation flexible sera ensuite associée à une instance de PostgreSQL pour réaliser un démonstrateur accessible publiquement. Ce démonstrateur sera hébergé sur notre plateforme interne de démonstration des prototypes de recherche.
- T8 : publications scientifiques
  - Cette tâche sera réalisée par les membres de l'équipe PILGRIM et aura pour objectif de rédiger des articles à destination de la communauté scientifique spécialisée dans les bases de données.

À partir de cet échéancier, nous pouvons énumérer un ensemble de livrables que nous nous engageons à produire :

- L1 : site Internet sur le projet qui sera mis-à-jour à la fin de chaque tâche,
- L2 : documentation de synthèse technique de SQL_f décrivant les différentes extensions proposées par rapport à SQL,
- L3 : code source de PostgreSQL_f,
- L4 : documentation technique de PostgreSQL_f a destination des développeurs,
- L5 : manuel d'utilisation/d'interrogation d'une base à l'aide de SQL_f,
- L6 : paquet complet contenant le code source et les documentations,
- L7 : démonstrateur en ligne,
- L8 : présentation des travaux et résultats devant la communauté Francophone PostgreSQL lors d'un séminaire,
- L9 : article pour la conférence francophone de Bases de données avancées (BDA),
- L10 : article pour une conférence internationale reprenant la description de SQL_f, son implémentation dans PostgreSQL et les tests de performance.

Comme l'illustre le diagramme de la Figure 4, ce projet se déroulera sur une durée d'environ 6 mois. Un rapport d'état d'avancement sera rédigé et envoyé aux responsables de l'appel à projet chaque mois ou à la fin de chacune des tâches citées ci-dessus.

En cas d'acceptation de ce projet par l'association Francophone PostgreSQL, dont la décision sera communiquée fin mars, ce projet débuterait en mai 2011 par une étude préliminaire effectuée par les membres de l'équipe PILGRIM avant d'accueillir deux stagiaires ingénieurs pour un stage de 2 mois de juillet à août. Le projet s'achèverait en octobre 2011.

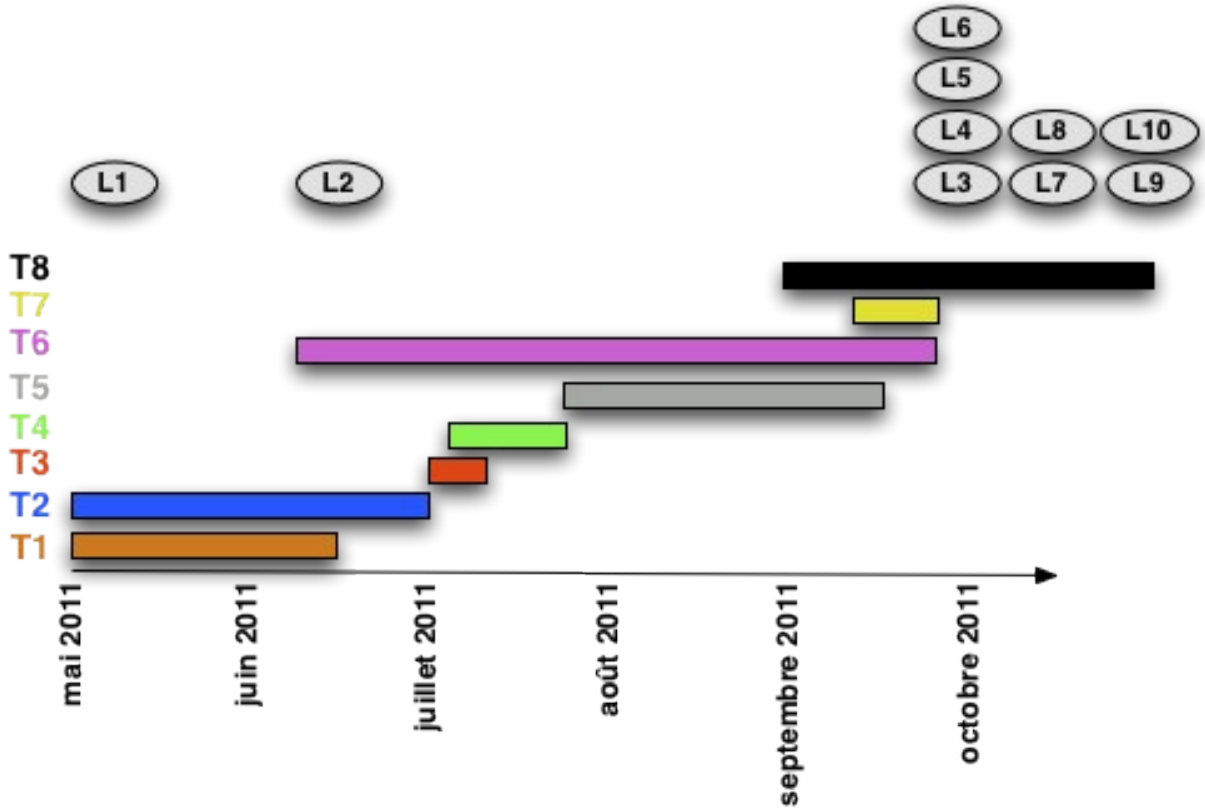

Figure 4 : échéancier du projet PostgreSQL_f

Les tâches T1 et T2 seront initiées collectivement par l'équipe PILGRIM et reprises par le stagiaire à son arrivée. Les tâches T3, T4, T5, T6 et T7 seront principalement réalisées par les étudiants stagiaires sous la supervision des membres de l'équipe PILGRIM. La tâche T8 sera remplie par les membres de l'équipe PILGRIM.

III. Budget Prévisionnel

III.1) Personnel

Pour mener à bien ce projet, nous sollicitons un financement pour l'accueil de deux étudiants ingénieurs pour un stage d'été de 2 mois. Nous ne disposons malheureusement pas de budget pour financer l'accueil de stagiaires. La réalisation de ce projet est donc entièrement conditionnée par l'obtention de financements annexes tels que celui proposé par l'association Francophone PostgreSQL. Le financement de ces stages s'élèverait à 650€ x 4 = 2600€. Le coût mensuel d'accueil correspond à celui pratiqué en général à l'école d'ingénieur dans laquelle nous effectuons nos enseignements (l'ENSSAT de Lannion). Il comprend la gratification de l'étudiant ainsi que les frais de gestion.

III.2) Communication

Afin de publier et de communiquer sur les résultats obtenus au cours de ce projet, nous

sollicitons également un financement pour couvrir ces déplacements. Ce projet s'inscrivant dans une démarche de promotion du développement francophone autour de PostgreSQL, nous incluons dans le budget du projet la prise en charge d'une publication pour une conférence francophone sur les bases de données, en l'occurrence Bases de Données Avancées 2012. Le financement d'une telle mission inclut l'inscription 500€, le transport et l'hébergement 600€, soit un total de 1100€.

Afin d'effectuer une première présentation du projet auprès de la communauté Francophone PostgreSQL puis une seconde présentation des résultats obtenus, nous sollicitons également le financement de deux déplacements de Lannion à Paris, pour un coût approximatif de 300 euros.

Nous proposons de prendre sur notre budget de recherche le financement d'une publication dans une conférence internationale.

Nous disposons déjà du matériel nécessaire pour réaliser ce projet : serveur de base de données, postes de développement et bases de données de test. Nous ne sollicitons donc pas d'autres types de financement.

Le Tableau 3 résume l'aide financière sollicitée :

Motif du financement	***Nature***	***Montant***
Financement étudiant stagiaire	Avance sur frais	2600€
Mission de publication dans une conférence francophone	Sur facture	1100€
Séminaires de présentation du projet et des résultats pour l'association Francophone PostgreSQL	Sur facture	300€
TOTAL		**4000€**

Tableau 3: synthèse du budget sollicité

IV. Référents du projet

IV.1) Identités et coordonnées

Grégory SMITS
IRISA / PILGRIM
ENSSAT de Lannion
6, rue de Kerampont - BP 80518
22305 Lannion cedex
tél. : +33(0)2.96.46.91.94
Web : http://www.irisa.fr/pilgrim/compositionpilgrim/gsmits

Né le 7 décembre 1980 (30 ans).
Situation actuelle : Maître de conférences depuis le 1/09/2009 affecté à l'IUT de Lannion département R&T pour les enseignements et pour la recherche à l'équipe PILGRIM/IRISA/Université de Rennes 1.

Cursus :

- 2009 : Maître de conférences à l'IUT de Lannion / équipe PILGRIM/IRISA
- 2008 : ATER IUT de Lannion
- 2007 : ATER ENSSAT de Lannion
- 2004-2007 : doctorat Orange Labs de Lannion / Université de Caen (laboratoire du GREYC)
- 2003 : DEA Intelligence Artificielle à l'Université de Caen
- 2001-2002 : licence, maîtrise d'informatique à l'Université de Caen
- 1999-2001 : DUT informatique à l'IUT d'Ifs (Université de Caen)
- 1998 : baccalauréat littéraire

Liste des cinq publications les plus significatives :

- Bosc, P., Hadjali, A., Pivert, O. and Smits, G. An approach based on predicate correlation to the reduction of plethoric answer sets. (2011 à paraître). In Advances in Knowledge Discovery and Management AKDM2, Series *Studies in Computational Intelligence* *S*pringer, Editors: F. Guillet, G. Ritschard and D. Zighed.
- Bosc, P., Hadjali, A., Pivert, O. and Smits, G. On the Use of Fuzzy Cardinalities for Reducing Plethoric Answers to Fuzzy Queries. (2010). In: Proc. of the 4th IEEE International Conference on Scalable Uncertainty Management (SUM'10), Toulouse, France.
- Bosc, P., Pivert, O. and Smits, G. A database preference query model based on a fuzzy outranking relation. (2010). In: Proc. of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'10), Barcelona, Spain.
- Heinecke, J., Smits, G., Chardenon, C., Guimier de Neef, E., Maillebuau, E. et Boualem, M. TiLT : une Plateforme pour le Traitement Automatique des Langues. (2008). Revue Traitement Automatique des Langues, vol. 49, p.17-41.
- Smits, G. and Tardif, O. Resolving Coreference using an Outranking Approach. In: Proc. of Recent Advances in Natural Language Processing, Borovetz Bulgaria, p. 571-575.

Nombre de publications :

- chapitre de livre : 1
- article de journal à comité de lecture : 1
- conférences internationales à comité de lecture : 8
- conférences nationales à comité de lecture : 7

Olivier PIVERT
IRISA / PILGRIM
ENSSAT de Lannion
6, rue de Kerampont - BP 80518
22305 Lannion cedex
tél. : +33(0)2.96.46.90.31.
Web : http://www.irisa.fr/pilgrim/compositionpilgrim/opivert
45 ans

Situation actuelle : Olivier Pivert est professeur en informatique à l'ENSSAT de Lannion (Université de Rennes 1). Il est également responsable de l'équipe de recherche Irisa/Pilgrim.

Cursus :

- 2000 : Habilitation à Diriger des recherches, Université de Rennes 1 « Applications de la théorie des ensembles flous dans le domaine des bases de données »,
- 1991 : doctorat en informatique, Université de Rennes 1 « Contribution à l'interrogation

flexible de bases de données : expression et évaluation de requêtes floues »,
- 1987 : diplôme d'ingénieur en informatique, INSA Rennes.

Liste des cinq publications les plus significatives :

- P. Bosc, O. Pivert, About approximate inclusion and its axiomatization, *Fuzzy Sets and Systems*, 157 (11), pp. 1438-1454, 2006.
- P. Bosc, O. Pivert, About possibilistic queries and their evaluation, IEEE Transactions on Fuzzy Systems, vol. 15, pp. 439-452, 2007.
- P. Bosc, A. Hadjali, O. Pivert, On the versatility of fuzzy sets for modeling flexible queries, in: *Handbook of Research on Fuzzy Information Processing in Databases*, J. Galindo (Ed.), Information Science Reference, Hershey, PA, USA, 2008, pp. 143-166.
- P. Bosc, A. Hadjali, O. Pivert, Incremental controlled relaxation of failing flexible queries, *Journal of Intelligent Information Systems*, vol. 33(3), 261-283, 2009.
- P. Bosc, O. Pivert, Modeling and querying uncertain relational databases: A survey of approaches based on the possible worlds semantics, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 18(5), pp. 565-603, 2010.

Nombre de publications :

- chapitre de livre : 19, dont 5 lors des 5 dernières années,
- articles de journaux internationaux avec comité de lecture : 30, dont 19 lors des 5 dernières années,
- conférences internationales à comité de lecture : 172, dont 84 lors des 5 dernières années.

IV.2) Compétences pour mener à bien ce projet

Ce projet est évidemment ambitieux et les difficultés sont à la hauteur des enjeux et retombées escomptés. Nous avons bon espoir que ce projet puisse aboutir à l'obtention de résultats probants pour différentes raisons. Tout d'abord, Olivier Pivert, responsable de l'équipe PILGRIM est à l'origine avec Patrick Bosc du développement de SQL_f et bénéficie de plus de 20 ans d'expérience dans la recherche en bases de données. Olivier Pivert a également une grande expérience d'encadrement et de gestion de projets qu'il s'agisse de projets financés par l'Agence Nationale pour la Recherche ou d'encadrement de doctorants.

Grégory SMITS, référent de ce projet, a également encadré plusieurs stagiaires pour le développement d'applications informatiques. Ces encadrements ont abouti à un logiciel utilisé en interne dans les laboratoires de recherche d'Orange Labs à Lannion et à un logiciel également utilisé en interne à l'IUT de Lannion.

De plus, Olivier Pivert et Grégory Smits disposent tous les deux d'une parfaite maîtrise des langages C et SQL qu'ils enseignent respectivement à l'ENSSAT et à l'IUT de Lannion.

V. Références

- [BOSC1993]Bosc, P.; Pivert, O., On the evaluation of simple fuzzy relational queries ~ principles and measures, in Fuzzy Logic – State of the Art (R. Lowen editor), Kluwer Ac. Pub., 1993.
- [BOSC1995] Bosc, P.; Pivert, O.; , SQLf: a relational database language for fuzzy querying, Fuzzy Systems, IEEE Transactions on , vol.3, no.1, pp.1-17, Feb 1995
- [BOSC1996] Bosc, P.; Some approaches for processing SQLf nested queries. International Journal of Intelligent Systems, Vol.11, 1996.
- [BOSC1997] Bosc, P.; Pivert, O.; Requêtes flexibles et division relationnelle, Ingénierie des systèmes d'information, vol. 5, pp. 661-690, 1997.
- [BOSC1998] Bosc, P.; On the primitivity of the division of fuzzy relations, Journal of Soft Computing, vol. 2, pp. 35-47, 1998.
- [BOSC1999] Bosc, P.; Legrand, C.; Pivert, O.; , "About fuzzy query processing-the example of the division," Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE '99. 1999 IEEE International , vol.2, no., pp.592-597 vol.2, 1999.
- [BOSC2004] Bosc, P.; Liétard, L.; Pivert, P.; Rocacher, D.; Gradualité et imprécision dans les bases de données, ed. Ellipses, 2004.
- [BOSC2010] P. Bosc, A. Hadjali, O. Pivert, and G. Smits. CORTEX -- CORrelaTion-based Query EXpansion. In Actes des 25èmes Journées Bases de Données Avancées (BDA'10), Session démonstration, Toulouse, France, 2010.
- [BOSC2011] Bosc, P.; Pivert, O.; Smits G. On a Fuzzy Group-By and Its Use for Fuzzy Association Rule Mining. In proc. of Advances in Databases and Information Systems, Lecture Notes in Computer Science, 2011.
- [KACPRZYK1995] J. Kacprzyk, S. Zadrosny, FQUERY for Access: fuzzy querying for a Windows-based DBMS, in: P. Bosc, J. Kacprzyk (Eds.), Fuzziness in Database Management Systems, Physica-Verlag, Heidelberg, 1995, pp. 415–433.
- [TAHANI1977] Tahani, V., A conceptual framework for fuzzy query processing--A step toward very intelligent database systems, Information Processing & Management, Volume 13, Issue 5, 1977, Pages 289-303.