Requêtes en SQL et leur graphe des dépendences

Giovanni Bernardi

September 18, 2017

1 Motivation

Nos activités quotidiennes dépendent souvent de programmes fournis par entreprises comme Facebook, Amazon, etc etc. Ces programmes dépendent eux-même de bases de données (DB) qui sont distribués en pays differents. Par exemple, considérons Facebook: chaque opération de lecture (*read*) invoqué par un utilisateur est executé sur le DB le plus géographiquement proche au utilisateur même (dans notre cas Dublin), tandis que chauqe opération d'ecriture (*write*) est executé sur un DB central en Californie.

Cettes bases de données distribués sont modifiés par de code qui nous represéntons comme un ensemble fini $\mathcal{Q}$ de rêquetes écrites en SQL:

$$\mathcal{Q} = \{Q_1, Q_2, Q_3, \ldots, Q_n\}.$$

Example 1 *Considérons un DB pour une application de vente en-ligne (comme e-Bay), qui consiste en deux tables décrites respectivement par les schemas*

`ITEMS`	(`iId`, `nbids`)
`BIDS`	(`bId`, `iId`)

oú

- `ITEMS` *contient une ligne avec clé primaire* `iId` *pour chaque objet (item) en vente;*
- `BIDS` *contient une ligne avec clé primaire* `bId` *pour chaque offre (bid) sur un object* `iId`*.*

Les suivants sont deux rêquetes SQL, une pour lire l'êtat des items dans le DB, et l'autre pour ajouter une offre,

```
StoreBid(iid, val) {
insert into BIDS values (_, iid, val)
select nbids from ITEMS as n where iId = iid
update ITEMS set nbids = n + 1 where iId = iid
}
```

```
ViewItem(iid) {
select * from ITEMS as ret_item where iId = iid
}
```

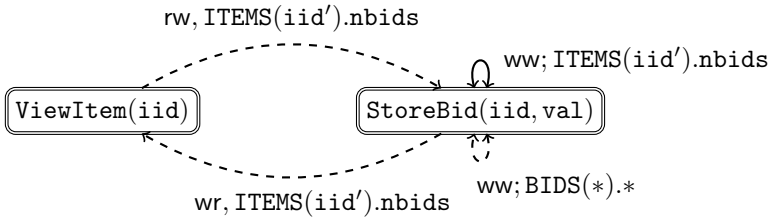

Figure 1: Graph statique de dependences de deux rêquetes en Exemple (1).

□

À cause de la distribution géographique du DB, la semantique des rêquetes à priori n'est pas claire, et encore moins leur correction. Pour surmonter cette difficulté, les développeurs employent outiles qui analysent et prouvent propriétés du logiciel qu'ils écrivent.

2 But du projet

Une technique d'analyse d'un ensemble des rêquetes SQL $\mathcal{Q}$ consiste à calculer un *graph de dépendances* et en vérifier dans le graphe l'absence de certains cycles dits *critiques*.

Dans ce contexte un graphe est une couple (N, E), oú N est un ensemble des *sommets* (nodes), L est un ensemble des etiquettes, et $E \subseteq N \times L \times N$ est un ensemble des *arcs* qui connectent les sommets. Par exemple, le graphe de dépendances des rêquetes en Exemple (1) est dessiné en Figure (1).

Étant donné un ensemble des rêquetes $\mathcal{Q}$, le graphe de dépendances de $\mathcal{Q}$, denoté $ddg(\mathcal{Q})$, est par definition la couple $(\mathcal{Q}, E)$, oú l'ensemble E contient les arcs qui representent comme les operations read/write dans le code peuvent creé de conflicts à run-time:

- si une rêquete Q_i lit un objet x et un rêquete Q_j écrit x alors $ddg(\mathcal{Q})$ doit contenir lesarcs $Q_i \xrightarrow{\mathsf{rw},x} Q_j$ et $Q_j \xrightarrow{\mathsf{wr},x} Q_i$;
- si deux rêquetes Q_i and Q_j écrivent un object x alors $ddg(\mathcal{Q})$ doit contenir un les arcs $Q_i \xrightarrow{\mathsf{ww},x} Q_j$ et $Q_i \xrightarrow{\mathsf{ww},x} Q_j$.

Le but du projet est developper un programme qui (a) calcule le graphe de dependences d'un ensemble des queries donné, et qui (b) verifie l'absence des cycles critiques dans le graphe. Les rêquetes sont écrites avec le language dont la grammaire est en Figure (2).

Q	::=	$name(x_1, x_2, \ldots x_n)\{C\}$	Query
C	::=		
		$C; C$	Sequence
		$if\ B\ then\ C\ else\ C$	Conditional
		$INSERT\ INTO\ T\ colId$ $VALUES\ v$	Insertion
		$SELECT\ (colId_1, \ldots, colId_n)$ $FROM\ T$ $WHERE\ B$	Selection
		$UPDATE\ T$ $SET\ colId = v$ $WHERE\ B$	Update
		$DELETE\ FROM\ T\ WHERE\ B$	Deletion
B	::=	$V = V$	Equality check
		$V > V$	Greater then
		$B \wedge B$	Conjunction
		$\neg B$	Negation
V	=	$\mathbb{N} \cup Vars \cup Vals$	

Figure 2: Grammaire d'un language de rêquetes proche à SQL.

3 Prérequis:

- IO2 + BD3: bases de données
- POOIG: programmation oriente objets

4 Jalons

Fonction f_1: Un parseur f qui converte une rêquete Q en une 4-uplet des ensembles des objets ou variables,

$$(R^{may}, W^{may}, R^{mst}, W^{mst}). \tag{1}$$

Les ensembles en (1) contient respectivement

- R^{may}: les objets ou variables qui Q peut lire
- W^{may}: les objets ou variables qui Q peut écrire
- R^{mst}: les objets ou variables qui Q doit lire
- W^{mst}: les objets ou variables qui Q doit écrire

La fonction f_1 est définie comme le lift de f à les ensembles des rêquetes:

$$\begin{aligned} f_1(\{\, Q_1, \ldots, Q_n \,\}) &= \{\, (f(Q_1), \ldots, f(Q_2)) \,\} \\ &= \{\, (R^{may}, W^{may}, R^{mst}, W^{mst})_1, \\ &\quad \ldots, \\ &\quad (R^{may}, W^{may}, R^{mst}, W^{mst})_n \,\} \end{aligned}$$

Fonction f_2: Un programme qui prenne comme input un ensemble fini N des n-uplets comme dans (1) ci-dessus, et qui calcule l'ensemble des dépendences E, c'est à dir les arcs du graphe de dépendences. La sortie du programme est le graphe (N, E).

Fonction f_3: Un programme qui prenne comme input un graphe des dépendences $\mathcal{G} = (N, E)$ et qui vérifie que $\mathcal{G}$ ne contienne pas des cycles critiques. Le programme doit réturner un cycle critique s'il trouve au moins un, en cas contraire il doit returner 0.