M1 informatique - Optimisation des bases de données

TD 4 : évaluation de requêtes

1 Estimation de taille

Soit une instance de 10 000 enregistrements de la table *purchase* dont le schéma est :

```
CREATE TABLE purchase(  customer_id UNSIGNED SMALLINT
                          NOT NULL AUTO_INCREMENT,
                          store_name CHAR(15) NOT NULL,
                          purchased_item CHAR(10) NOT NULL,
                          purchase_date DATETIME NOT NULL,
                          PRIMARY KEY(customer_id));
```

Soient les statistiques suivantes pour cette table :

T(purchase)	10 000
V(purchase,customer_id)	10 000
V(purchase,store_name)	1 000
V(purchase,purchased_item)	10 000
V(purchase,purchase_date)	2 000

Donner l'arbre de requête et estimer la taille des relations résultats des requêtes suivantes :

1.

```
SELECT *
FROM purchase
WHERE store_name='amazon';
```
2.

```
SELECT purchased_item
FROM purchase
WHERE store_name='amazon' AND customer_id=100;
```
3.

```
SELECT P1.customer_id, P2.customer_id, P1.purchased_item, P2.purchased_item
FROM purchase P1, purchase P2
WHERE P1.purchased_date=P2.purchased_date
AND P1.store_name=amazon;
```

2 Nested loop et sort join

Soient des tables R et S pour lesquelles on a $B(R)=B(S)=10\,000$. Supposons que R et S soient rangées de manière contiguë et que le nombre de blocs disponibles en mémoire soit $M = 1000$. Quel est le coût de la jointure entre R et S avec un nested loop join? Avec un sort join?

3 Index join, sort join et hash join

Considérons une instance de 10 000 enregistrements de la table *purchase* définie comme suit :

```
CREATE TABLE purchase(  customer_id UNSIGNED SMALLINT
                          NOT NULL AUTO_INCREMENT,
                          store_name CHAR(15) NOT NULL,
                          purchased_item CHAR(10) NOT NULL,
                          purchase_date DATETIME NOT NULL,
                          PRIMARY KEY(customer_id));
```

Soit la requête :

```
SELECT  c1.customer_id
FROM    purchase c1, purchase c2
WHERE   c1.store_name = 'mystore'
AND    c1.purchase_date = c2.purchase_date;
```

Considérons le plan d'exécution qui propose une jointure entre *purchase* et $\sigma_{store_name='mystore'}(purchase)$. Supposons que *purchase* possède un index sur l'attribut *purchase_date*.

Quel est coût d'évaluation de cette requête avec un index join? Même question avec un sort join? Même question avec un hash join?

4 Sélectivité et sort join

Soient des tables R et S pour lesquelles on a $B(R)=1000$ et $B(S)=500$. Supposons que le nombre de blocs disponibles en mémoire soit $M = 101$. On cherche à estimer le coût de la jointure $R \bowtie S$ sur l'attribut Y , sachant que Y ne peut prendre que 2 valeurs et que celles-ci se trouvent également distribuées dans R comme dans S (c'est à dire que chacune de ces valeurs apparaît dans la moitié de R et la moitié de S).

Quel est le coût de cette jointure avec un sort-join?