

# Cost Hash Join

July 2023

## 1 Notation

$B$  : Nombre de partitions.

$F$  : Fudge factor

$s$  : Facteur de selectivité

$|M|$  : Nombre de pages disponibles en mémoire.

$|R|$  : Nombre de pages occupées par R.

$|S| \geq |R|$  : Nombre de pages occupées par S.

$\{R\}$  : Nombre de tuples dans R.

$|H|$  : Taille de la table de hachage.

$A_r = \left\lceil \frac{|R|}{B} \right\rceil$  : Nombre de pages par partition de R.

*comp* : Temps pour comparer des clé dans la mémoire.

*hash* : Temps pour hacher une clé dans la mémoire.

*move* : Temps pour déplacer un tuple dans la mémoire.

*swap* : Temps pour échanger deux tuples dans la mémoire.

*IO* : Temps pour lire ou écrire une page entre le disque et la mémoire.

## 2 Simple Hash Join

### 2.1 Cas 1 : La table de hachage rentre dans la mémoire

Condition :  $F * |R| = |H| \leq |M| - 2$

#### 2.1.1 I/O

Lecture :  $|R| + |S|$

Ecriture :  $\lceil |R| * s \rceil$

### 2.1.2 Coût

$$temps = \{R\} * (hash + move) \quad (1)$$

$$+ \{S\} * (hash + F * comp) \quad (2)$$

$$+ [Lecture + Ecriture] * IO + [\{R\} * s] \quad (3)$$

- (1) Tuples de R que l'on hache et déplace pour créer la table H en mémoire.
- (2) Tuples de S que l'on hache et compare pour chercher dans la table H en mémoire.
- (3) Entrées/Sorties et le déplacement des tuples satisfaisant la jointure.

## 2.2 Cas 2 : La table de hachage ne rentre pas dans la mémoire

$$\text{Condition : } F * |R| = |H| > |M| - 2$$

### 2.2.1 Partitionnement

$$\text{Condition : } F * |R| = |H| \leq |M| - 3$$

$$B = \left\lceil \frac{|R| * F}{|M| - 3} \right\rceil$$

$$|H| = \left\lceil \frac{|R| * F}{B} \right\rceil$$

### 2.2.2 I/O

$$\text{Lecture : } |R| + |S| + \frac{B * (B - 1)}{2} * (A_r + A_s)$$

$$\text{Ecriture : } \lceil |R| * s \rceil + \frac{B * (B - 1)}{2} * (A_r + A_s)$$

### 2.2.3 Coût

$$temps = \left\lceil \frac{B * (B + 1)}{2} \right\rceil * [\{A_r\} + \{A_s\}] * (hash + move) \quad (4)$$

$$- \{S\} * move \quad (5)$$

$$+ \{R\} * hash \quad (6)$$

$$+ \{S\} * (hash + F * comp) \quad (7)$$

$$+ [Lecture + Ecriture] * IO + [\{R\} * s] \quad (8)$$

- (4) Tuples de R et S qu'on hache en partition et réécrit sur le disque ou déplace dans la table  $H_i$ .
- (5) Tuples de S qui appartiennent à la partition  $B_i$  et que l'ont ne déplacent

pas et qui on été comptabilisées dans (4).

(6) Tuples de R que l'on hache et avons déplacé en (4) pour créer la table  $H_i$  en mémoire.

(7) Tuples de S que l'on hache et compare pour chercher dans la table  $H_i$  en mémoire.

(8) Entrées/Sorties et le déplacement des tuples satisfaisant la jointure.

### 3 Grace Hash Join

#### 3.1 Cas 2 : La table de hachage ne rentre pas dans la mémoire

##### 3.1.1 Partitionnage

$$B = |M| - 1$$

$$\text{Condition : } \left\lceil \frac{|R| * F}{B} \right\rceil = |H| \leq |M| - 2$$

##### 3.1.2 I/O

$$\text{Lecture : } |R| + |S| + B * (A_r + A_s)$$

$$\text{Ecriture : } \lceil |R| * s \rceil + B * (A_r + A_s)$$

##### 3.1.3 Coût

$$temps = \left[ \{R\} + \{S\} \right] * (hash + move) \quad (9)$$

$$+ \{R\} * (hash + move) \quad (10)$$

$$+ \{S\} * (hash + F * comp) \quad (11)$$

$$+ \left[ Lecture + Ecriture \right] * IO + \lceil \{R\} * s \rceil \quad (12)$$

(9) Tuples de R et S que l'on hache en partition et qu'on écrit sur la partition correspondante en mémoire.

(10) Tuples de R que l'on hache et déplace pour créer la table  $H_b$  en mémoire.

(11) Tuples de S que l'on hache et compare pour chercher dans la table  $H_b$  en mémoire.

(12) Entrées/Sorties et le déplacement des tuples satisfaisant la jointure.

## 4 Hybrid Hash Join

### 4.1 Cas 2 : La table de hachage ne rentre pas dans la mémoire

#### 4.1.1 Partitionnage

$$\text{Condition : } \left\lceil \frac{|R| * F}{B} \right\rceil = |H| \leq |M| - 2$$

$$B = 1 + \left\lceil \frac{|R| * F - |M| + 2}{|M| - 3} \right\rceil$$

#### 4.1.2 I/O

$$\text{Lecture : } |R| + |S| + (B - 1) * (A_r + A_s)$$

$$\text{Ecriture : } \lceil |R| * s \rceil + (B - 1) * (A_r + A_s)$$

#### 4.1.3 Coût

$$temps = \left[ \{R\} + \{S\} \right] * (hash + move) \quad (13)$$

$$- \{A_s\} * move \quad (14)$$

$$+ \{R\} * hash \quad (15)$$

$$+ \{S\} * (hash + F * comp) \quad (16)$$

$$+ [Lecture + Ecriture] * IO + \lceil \{R\} * s \rceil \quad (17)$$

(13) Tuples de R et S que l'on hache en partition et qu'on écrit sur la partition correspondante en mémoire ou déplace dans la table  $H_0$ .

(14) Tuples de S de la partition 0 pour qui n'ont pas à être déplacés et qui ont été comptabilisés en (13).

(15) Tuples de R que l'on hache pour créer la table  $H_b$ .

(16) Tuples de S que l'on hache et compare pour chercher dans la table  $H_b$  en mémoire.

(17) Entrées/Sorties et le déplacement des tuples satisfaisant la jointure.

## 5 Sort-Merge Join

### 5.1 Tri

#### 5.1.1 Passe

$$B_R = 1 + \left\lceil \log_{M-1} \left\lceil \frac{|R|}{|M|} \right\rceil \right\rceil$$

$$B_S = 1 + \left\lceil \log_{M-1} \left\lceil \frac{|S|}{|M|} \right\rceil \right\rceil$$

### 5.1.2 I/O

Lecture :  $|R| * B_R + |S| * B_S$

Ecriture :  $|R| * B_R + |S| * B_S$

### 5.1.3 Coût

$$temps = \left[ \{R\} * \log_2 \{R\} + \{S\} * \log_2 \{S\} \right] * (swap + move) \quad (18)$$

$$+ \left[ Lecture + Ecriture \right] * IO \quad (19)$$

(18) Tri fusion des tuples R et S.

(19) Entrées/Sorties.

## 5.2 Fusion

### 5.2.1 I/O

Lecture :  $|R| + |S|$

Ecriture :  $|R| + |S|$

### 5.2.2 Coût

$$temps = Tri_R + Tri_S \quad (20)$$

$$+ \left[ \{R\} + \{S\} \right] * comp \quad (21)$$

$$+ \left[ Lecture + Ecriture \right] * IO + \lceil \{R\} * s \rceil \quad (22)$$

(20) Cout des tris de R et S.

(21) Comparaisons des tuples R et S lors de la fusion.

(22) Entrées/Sorties et le déplacement des tuples satisfaisant la jointure.

## 6 Index nested loop join

### 6.1 Build

#### 6.1.1 Niveaux

$lg$  = nombre de pointeurs/tuples par page

$n = 1 + \lceil \log_{lg} |R| \rceil$  : nombre de niveaux de l'index I

$|I_i| = \frac{|R|}{lg^i}$  : nombre de pages de par niveaux de I

$|I| = \sum_{i=0}^{i=N} |I_i|$  : nombre de pages de I

$n_M$  : nombre de niveaux de l'index rentrant dans la mémoire.

### 6.1.2 I/O

Lecture :  $IO_{Tri_R} + |R|$

Ecriture :  $IO_{Tri_R} + |I|$

### 6.1.3 Coût

$$temps = Tri_R \quad (23)$$

$$+ \{R\} * (move + hash) \quad (24)$$

(23) Tri fusion des tuples R .

(24) Détermination de la clé d'indexation et déplacement des tuples de R.

## 6.2 Probe

### 6.2.1 I/O

Lecture :  $IO_{Build_I} + |S| + \{S\} * \left[ (n+1) - n_M \right]$

Ecriture :  $IO_{Build_I} + \lceil |R| * s \rceil$

### 6.2.2 Coût

$$temps = Build_I \quad (25)$$

$$+ \{S\} * comp * \left[ (n+1) \right] \quad (26)$$

$$+ \left[ Lecture + Ecriture \right] * IO + \lceil \{R\} * s \rceil \quad (27)$$

(25) Cout du build de l'index.

(26) Comparaisons des tuples S et des plusieurs niveaux de l'index.

(27) Entrées/Sorties et le déplacement des tuples satisfaisant la jointure.