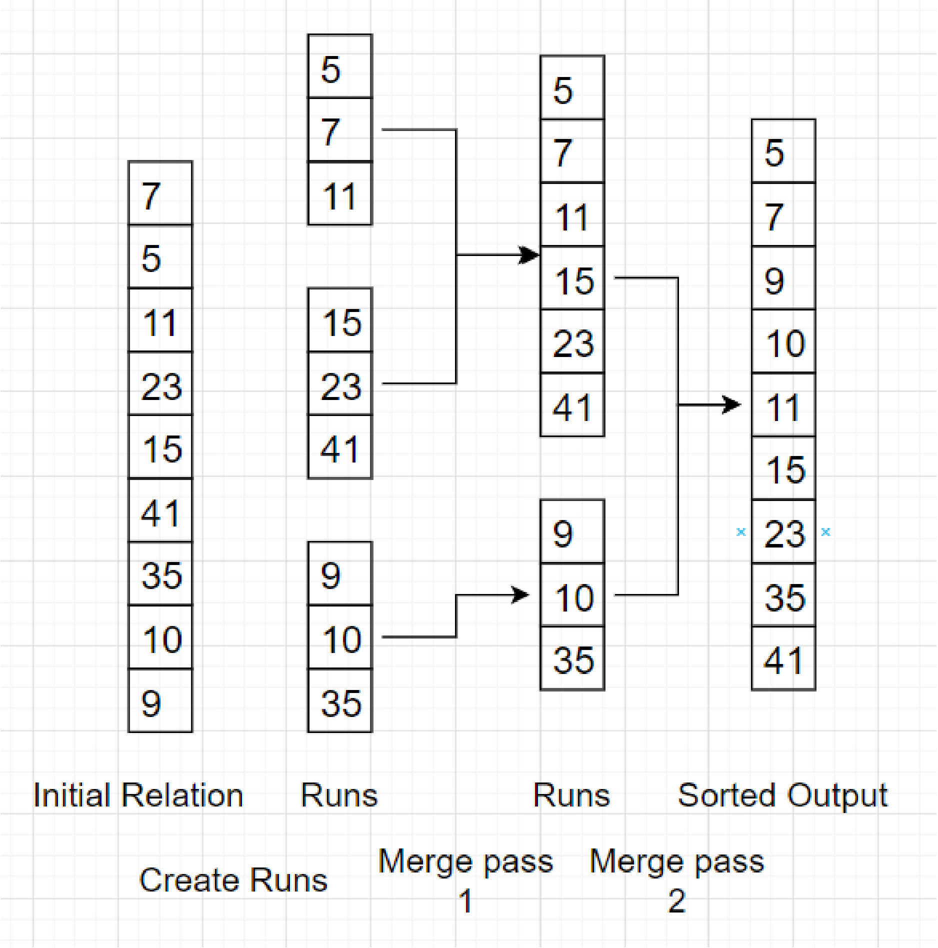
SOFTENG 351: Assignment #3

June 11, 2020

Aiden Burgess

`abur970 - 600280511`

1.



2.

a)

$n_w \times \frac{n_x}{V(w\,B)} \times \frac{n_y}{V(x\,C)} \times \frac{n_z}{V(y\,D)} = \frac{1000\times2000\times3000}{60\times100\times50} = 80,000,000\,tuples$

b)

$estimate(|\sigma_{A=10}(w)|) = \frac{n_w}{V(w,A)} = \frac{1000}{20} = 50$

c)

$estimate(|\sigma_{C=20}(y)|) = \frac{n_y}{V(y,C)} = \frac{3000}{50} = 60$

d)

$estimate(|\sigma_{C=20}(y) \bowtie z|) = 60 \times \frac{n_z}{V(y,D)} = 60 \times \frac{4000}{50} = 4800$

e)

$w \bowtie y = 0$

As there are no common fields, the result of a natural join will be empty.

f)

$estimate(|\sigma_{D>10}(z)|) = \frac{n_z}{3} = 1333$

Assuming queries involving inequality tend to retrieve a small fraction of the possible tuples.

g)

$estimate(|\sigma_{A=1\, and\, B=2}(w)|) = \frac{n_w}{V(w,A)} \times \frac{1}{V(w,B)} = \frac{1000}{20} \times \frac{1}{60} = 1$

3.

a)

- $((w \bowtie x) \bowtie y) \bowtie z$
- $z \bowtie ((w \bowtie x) \bowtie y)$
- $((x \bowtie w) \bowtie y) \bowtie z$
- $z \bowtie ((x \bowtie w) \bowtie y)$
- $w \bowtie ((x \bowtie y) \bowtie z)$
- $((x \bowtie y) \bowtie z) \bowtie w$
- $w \bowtie ((y \bowtie x) \bowtie z)$
- $(w \bowtie (x \bowtie y)) \bowtie z$
- $z \bowtie (w \bowtie (x \bowtie y))$
- $(w \bowtie (y \bowtie x)) \bowtie z$
- $z \bowtie (w \bowtie (y \bowtie x))$
- $w \bowtie (x \bowtie (y \bowtie z))$

- $(x \bowtie (y \bowtie z)) \bowtie w$
- $w \bowtie (x \bowtie (z \bowtie y))$
- $(x \bowtie (z \bowtie y)) \bowtie w$
- $(w \bowtie x) \bowtie (y \bowtie z)$
- $(y \bowtie z) \bowtie (w \bowtie x)$
- $(y \bowtie z) \bowtie (x \bowtie w)$
- $(x \bowtie w) \bowtie (y \bowtie z)$
- $(x \bowtie w) \bowtie (z \bowtie y)$
- $(z \bowtie y) \bowtie (x \bowtie w)$
- $(w \bowtie x) \bowtie (z \bowtie y)$
- $(z \bowtie y) \bowtie (w \bowtie x)$

Theres another 16, but this is left as an exercise for the reader. Total possible is 40

b)

$estimate(|w \bowtie x|) = n_w \times \frac{n_x}{V(w, B)} = 1000 \times \frac{2000}{60} = 33,333$

$estimate(|y \bowtie z|) = n_y \times \frac{n_z}{V(y, D)} = 3000 \times \frac{4000}{50} = 240,000$

$estimate(|(w \bowtie x) \bowtie (y \bowtie z)|) = n_{w \bowtie x} \times \frac{n_{y \bowtie z}}{V(y \bowtie z, C)}$

Therefore, $estimate(|(w \bowtie x) \bowtie (y \bowtie z)|) = 33,333 \times \frac{240{,}000}{100} = 80,000,000\, tuples$

Number of blocks: $4,000,000$

c)

$cost(w \bowtie x) = n_w + n_x = 200\, I/Os$

$cost(y \bowtie z) = n_y + n_z = 200\, I/Os$

Blocking factor of $w \bowtie x = \frac{1}{\frac{1}{10} + \frac{1}{20}} = 6.66 \approx 6$

Blocking factor of $y \bowtie z = \frac{1}{\frac{1}{30} + \frac{1}{40}} = 17.14 \approx 17$

Sorting cost of $w \bowtie x = 4 \times \frac{33{,}333}{6} = 22222\, I/Os$

Sorting cost of $y \bowtie z = 4 \times \frac{240{,}000}{17} = 56471\, I/Os$

Merge join cost: $\frac{33{,}333}{6} + \frac{240{,}000}{17} = 19,673\, I/Os$

Total cost = 98,766 I/Os

4.

a)

1. Table scan entire relation and filter by those which match the condition: $1000\, I/Os$
2. Index-scan for a=1 then filter based on other conditions: $50\, I/Os$
3. Index scan by b=2 then filter based on other conditions: $5\, I/Os$
4. Index scan by d=3 then filter based on other conditions: $10\, I/Os$

Choose plan 3, which has lowest cost at 5 I/Os

b)

1. Table scan entire relation and filter by conditions: $1000\, I/Os$
2. Index scan for a=1 then filter based on other conditions: $50\, I/Os$
3. Index scan for b=2 then filter based on other conditions: $5\, I/Os$
4. Index scan for c>=3 then filter based on other conditions: $1667\, I/Os$

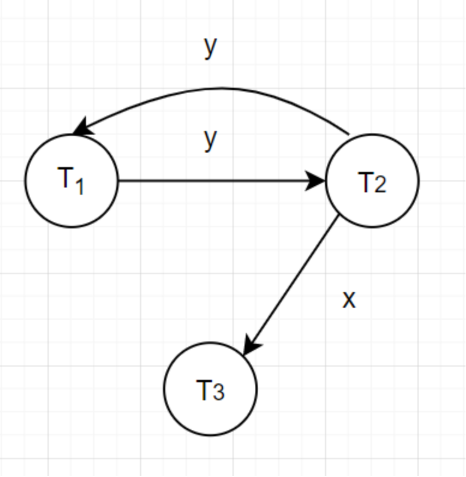
Choose plan 3, which has lowest cost at 5 I/Os

c)

1. Table scan entire relation and filter by conditions: $1000\, I/Os$
2. Index scan for a=1 then filter based on other conditions: $50\, I/Os$
3. Index scan for b<=2 then filter based on other conditions: $1667\, I/Os$
4. Index scan for c>=3 then filter based on other conditions: $1667\, I/Os$

Choose plan 2, which has lowest cost at 50 I/Os

5.



The schedule is not conflict-serializable, as there is a cycle in the precedence graph between T_1 and T_2.

6.

i) refers to T_1 and ii) refers to T_2

a)

i) 1

ii) 12

b)

i) 1

ii) 13

c)

i) 4

ii) 15

d)

i) 24

ii) 5120

7.

active -> partially committed -> committed -> terminated

This occurs if the transaction ends successfully and the commit conditions are met. The changes are committed to the database state then the transaction is terminated.

active -> partially committed -> failed -> terminated

This occurs if the transaction ends successfully but the commit conditions are not met. The transaction is then aborted and goes to the failed state, which is then terminated. No changes are made to the database state.

active -> failed -> terminated

This occurs if the transaction is not successful, it goes to a failed state, and then is terminated. No changes are made to the database.

8.

A deadlock occurs when there is a cycle in the wait-for graph. This means that the transactions in this cycle can not be completed as they block each other from completing.

One solution is to detect when a deadlock occurs and to choose a transaction as a victim to be undone and restart.

Another solution is to setup a time limit so that a transaction that exceeds this limit waiting is undone and restarted.

9.

After the system crash, all the operations after the checkpoint are rolled back.

We will use active transactions (L_1) and commited transactions since last checkpoint (L_2).

Undo all write_item operations in L_1, using the UNDO procedure. This should be done in reverse order.

1. [write_item, T_2, D, 15, 25]
2. [write_item, T_3, C, 30, 40]
3. [write_item, T_2, B, 12, 18]

Redo all write_item operations of committed transactions in L_2, in the order they are written in the log using REDO procedure.

1. [write_item, T_4, D, 25, 15]
2. [write_item, T_4, A, 30, 20]

No dirty reads take place, so no cascading rollback takes place.