CSCI3170 Introduction to Database Systems

Tutorial 10 – Final Exam Revision

DECOMPOSITION

Decomposition

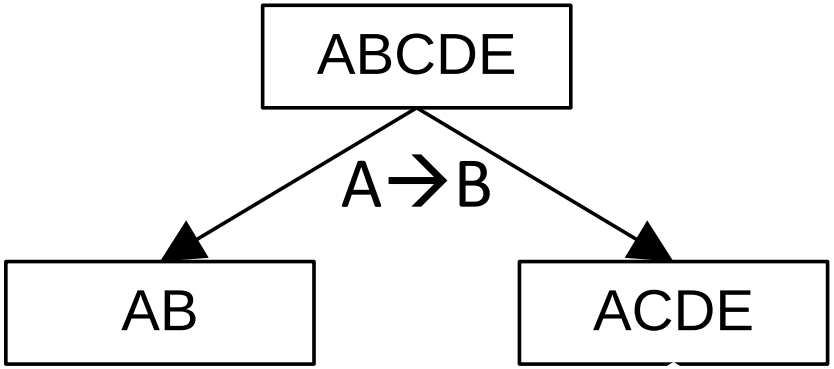
- $\{R_1, \ldots, R_n\}$ – a set of relation schemas
- $\{R_1, \ldots, R_n\}$ is a *decomposition* of R
  - if $R_1 \cup R_2 \cup \ldots \cup R_n = R$
- e.g.
  - R = (A, B, C)
  - R_1 = (A, B)
  - R_2 = (A, C)
- R_1 and R_2 is a decomposition of R

BCNF decomposition algorithm

Suppose R is not in BCNF, A is an attribute, and X → A is a FD that violates the BCNF condition.

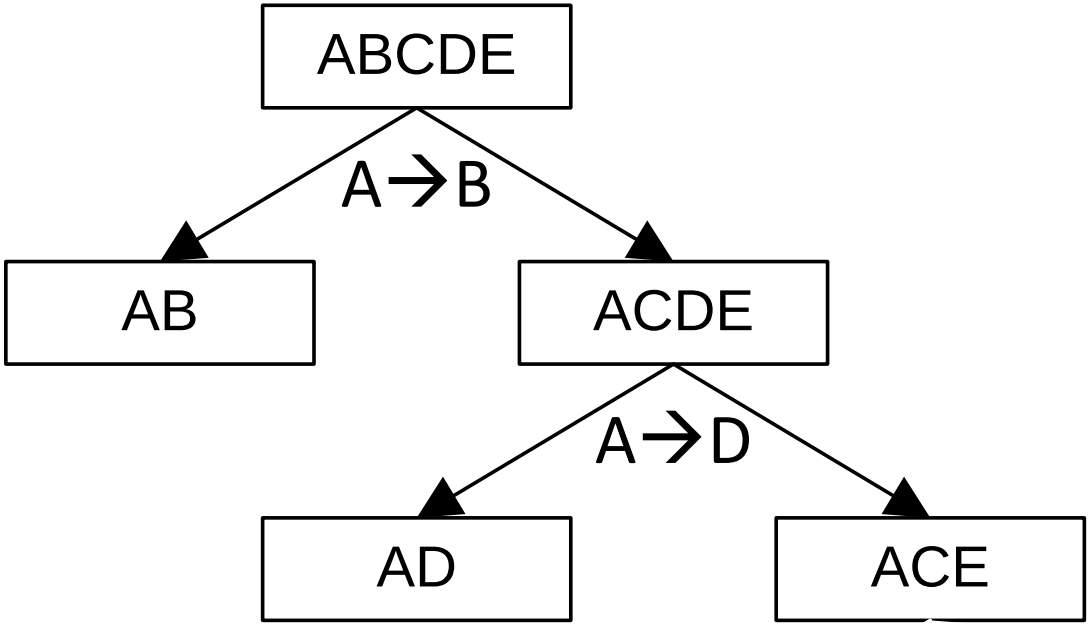
1. Remove A from R
2. Decompose R into XA and R-A
3. Repeat this process until all the relations become BCNF

Example



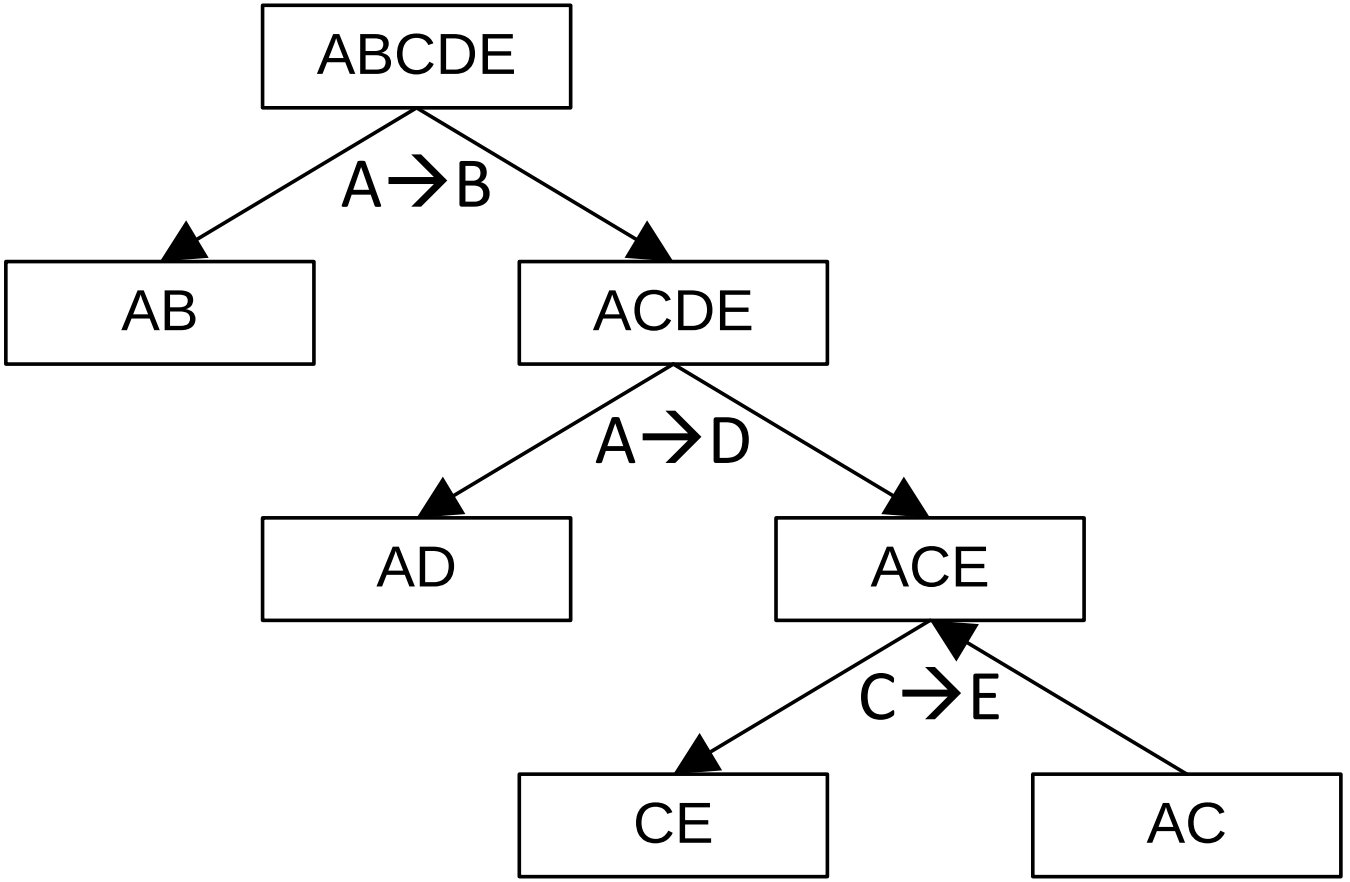
R = (A,B,C,D,E)
Key = AC
A → B

Example



R = (A,B,C,D,E)
Key = AC
$A \rightarrow B$
$A \rightarrow D$

Example



R = (A,B,C,D,E)
Key = AC
A → B
A → D
C → E

Canonical Cover

repeat
    Replace any $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ by $\alpha_1 \rightarrow \beta_1\beta_2$
    Delete any extraneous attribute from any $\alpha \rightarrow \beta$
until F does not change

Example

F = {E→A, E→B, A→BC, B→C}

- F_c = {**E→A, E→B**, A→BC, B→C}

  Combining **E→ A, E → B** into **E → AB**

Example

F = {E→A, E→B, A→BC, B→C}

- F_c = {E→A, E→B, A→BC, B→C}
- F_c = {E→AB, **A→BC, B→C**}

A→BC is extraneous
From **A→B** and **B→C**, we deduce **A→C** (transitivity)
From **A→B** and **A→C**, we deduce **A→BC** (union)

Example

F = {E→A, E→B, A→BC, B→C}

- F_c = {E→A, E→B, A→BC, B→C}
- F_c = {E→AB, A→BC, B→C}
- F_c = {**E→AB, A→B**, B→C}

E→AB is extraneous
From **E→A** and **A→B**, we deduce **E→A** (transitivity)
From **E→A** and **E→B**, we deduce **E→AB** (union)

Example

$F = \{E \rightarrow A, E \rightarrow B, A \rightarrow BC, B \rightarrow C\}$

- $F_c = \{E \rightarrow A, E \rightarrow B, A \rightarrow BC, B \rightarrow C\}$
- $F_c = \{E \rightarrow AB, A \rightarrow BC, B \rightarrow C\}$
- $F_c = \{E \rightarrow AB, A \rightarrow B, B \rightarrow C\}$
- $F_c = \{E \rightarrow A, A \rightarrow B, B \rightarrow C\}$

Final Result: $F_c = \{E \rightarrow A, A \rightarrow B, B \rightarrow C\}$

3NF decomposition algorithm

find a canonical cover F_c for F
result = { }
for each $\alpha \rightarrow \beta$ in F_c do
    if no schema in result contains $\alpha\beta$ then
        add schema $\alpha\beta$ to result
end for
if no schema in result contains a candidate key for R
   choose any candidate key α for R
   add schema α to result
end if

Example

- R = (A, B, C, D, E, F, G)
- F = { A → B, A → C, D → E, B → A, F → BG }
- F_c = { A → BC, D → E, B → A, F → BG }
- Candidate key = DF

Example

- R = (A, B, C, D, E, F, G)
- F_c = { A → BC, D → E, B → A, F → BG }
- Candidate key = DF

F_c	result
A → BC	
D → E	
B → A	
F → BG	

Example

- R = (A, B, C, D, E, F, G)
- F_c = { A → BC, D → E, B → A, F → BG }
- Candidate key = DF

F_c	result
A → BC	ABC
D → E	
B → A	
F → BG	

Example

- R = (A, B, C, D, E, F, G)
- F_c = { A → BC, D → E, B → A, F → BG }
- Candidate key = DF

F_c	result
A → BC	ABC
D → E	DE
B → A	
F → BG	

Example

- R = (A, B, C, D, E, F, G)
- F_c = { A → BC, D → E, B → A, F → BG }
- Candidate key = DF

F_c	result
A → BC	ABC
D → E	DE
B → A	
F → BG	

Example

- R = (A, B, C, D, E, F, G)
- F_c = { A → BC, D → E, B → A, F → BG }
- Candidate key = DF

F_c	result
A → BC	**AB**C
D → E	DE
B → A	
F → BG	

Example

- R = (A, B, C, D, E, F, G)
- F_c = { A → BC, D → E, B → A, F → BG }
- Candidate key = DF

F_c	result
A → BC	ABC
D → E	DE
B → A	
F → BG	FBG

Example

- R = (A, B, C, D, E, F, G)
- F_c = { A → BC, D → E, B → A, F → BG }
- Candidate key = **DF**

F_c	result
A → BC	ABC
D → E	DE
B → A	
F → BG	FBG

Result = {ABC, DE, FBG}

Example

- R = (A, B, C, D, E, F, G)
- F_c = { A → BC, D → E, B → A, F → BG }
- Candidate key = DF

F_c	result
A → BC	ABC
D → E	DE
B → A	
F → BG	FBG

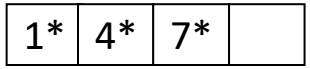
Result = {ABC, DE, FBG, DF}

B+ TREE

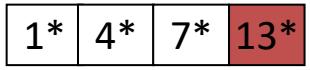
Insertion of B+ Tree

Order 2 B+ Tree
Root node: $1 \leq$ # elements ≤ 4
Non-root nodes: $2 \leq$ # elements ≤ 4

Original Tree

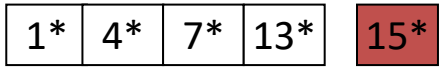


Insert 13



(No overflow)

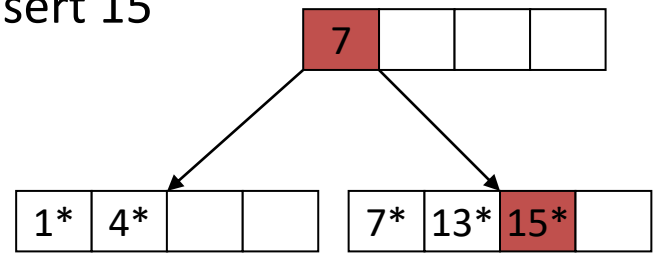
Insert 15



(Overflow)

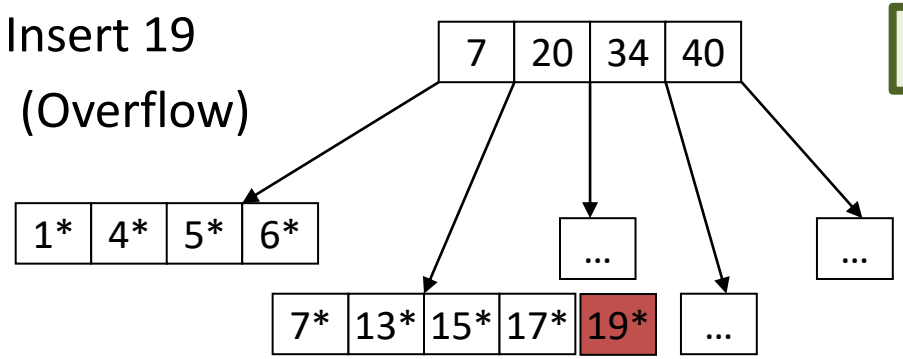
Leaf node overflow: split leaf node, **copy** middle **key** to the parent.

After Insert 15

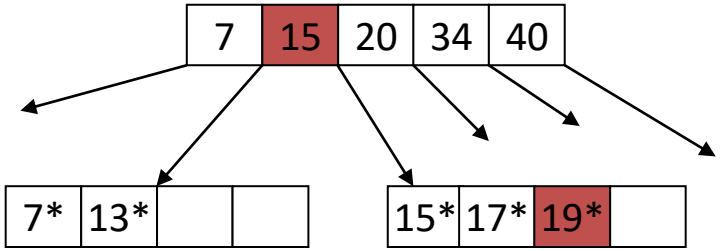


Tree grows

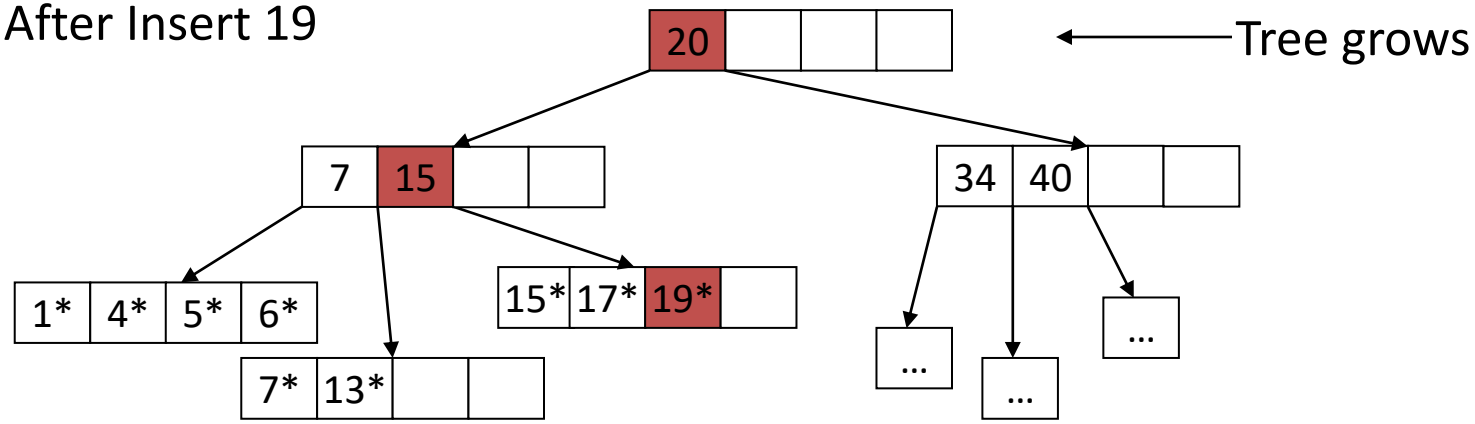
Insertion of B+ Tree



1. Leaf node overflow: split leaf node.

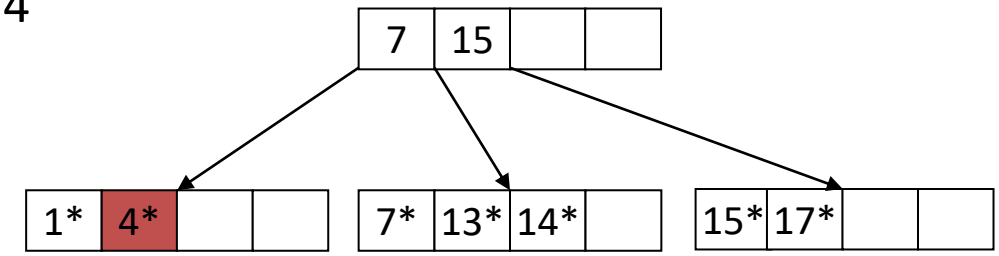


2. Non-leaf node overflow: split non-leaf node, **push** middle key up.



Deletion of B+ Tree

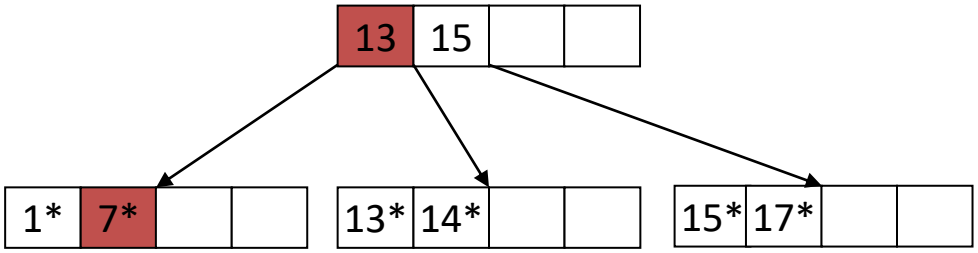
Delete 4



(Underflow)

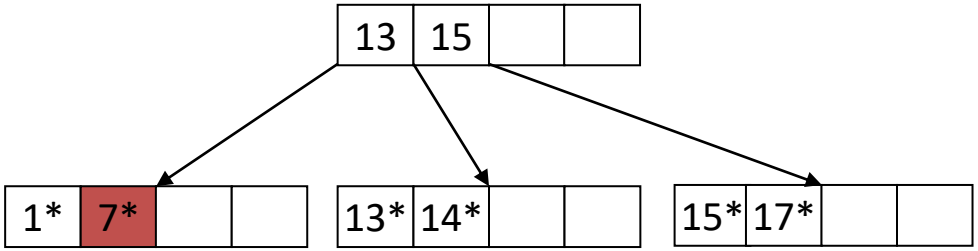
After delete 4, the page will underflow, need to borrow from sibling, update the parent.

After Delete 4



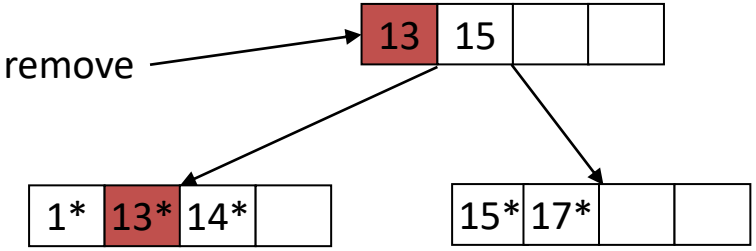
Deletion of B+ Tree

Delete 7

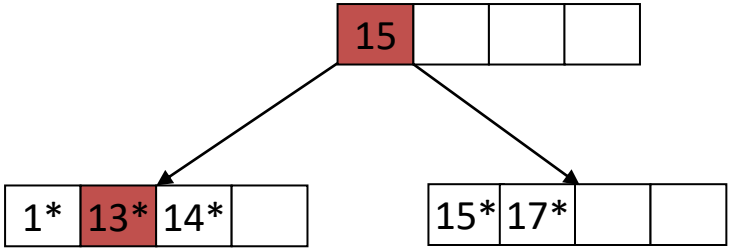


(Underflow)

After delete 7, the page will underflow, can't borrow from sibling, need to merge with sibling and update the parent.



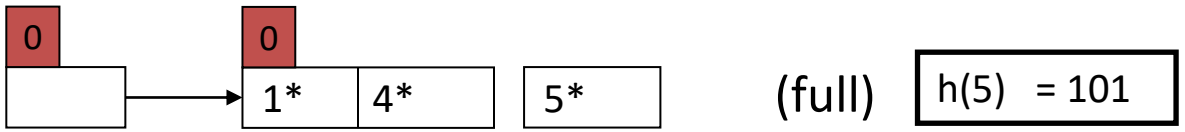
After Delete 7



HASHING

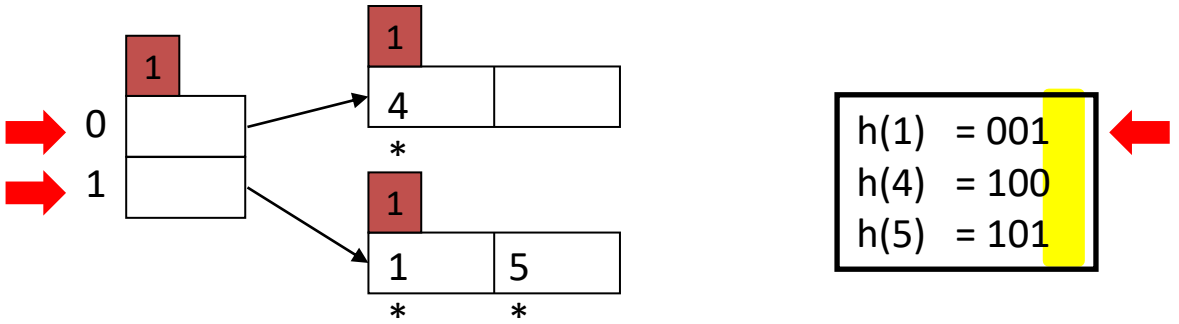
Insertion of Extensible Hashing

Insert 5:



If bucket with local depth = global depth, then double the directory and split the bucket.

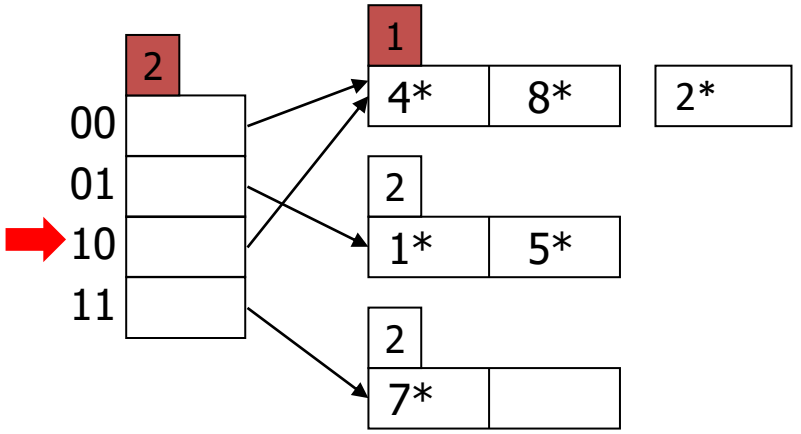
After insert 5:



Increase the global depth and local depth of the new buckets by 1.

Insertion of Extensible Hashing

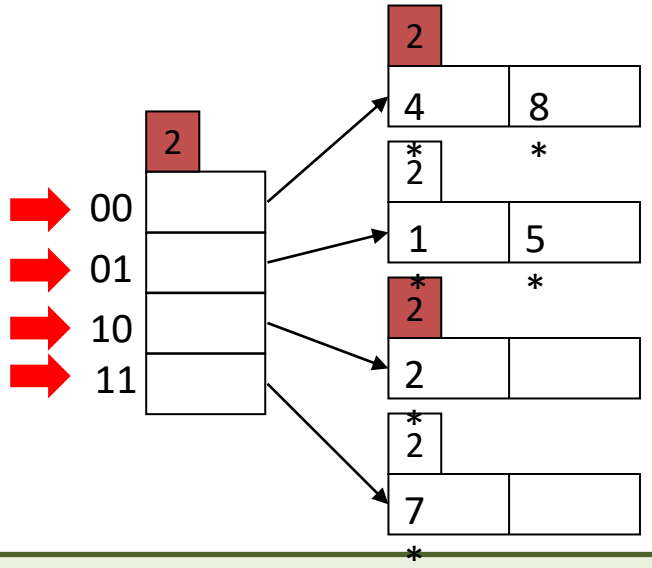
Insert 2:



h(2) = 010

If bucket with local depth < global depth, then split the bucket and update the pointers.

After insert 2:



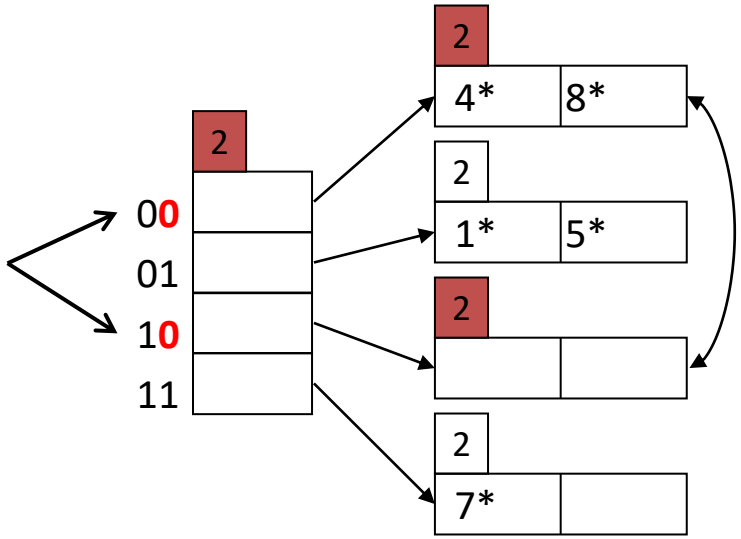
h(1) = 0001
h(2) = 0010
h(4) = 0100
h(5) = 0101
h(7) = 0111
h(8) = 1000

Increase the local depth of the new buckets by 1.

Deletion of Extensible Hashing

Delete 2:

Look at the last
(global depth-1) bits

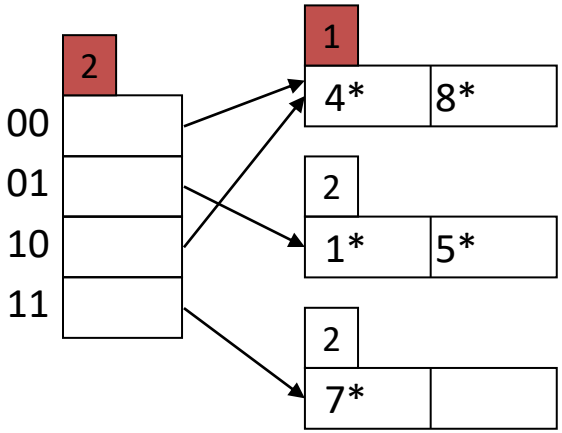


Merge them

(empty)

If the removal makes the bucket empty, then remove the bucket and update the pointer.

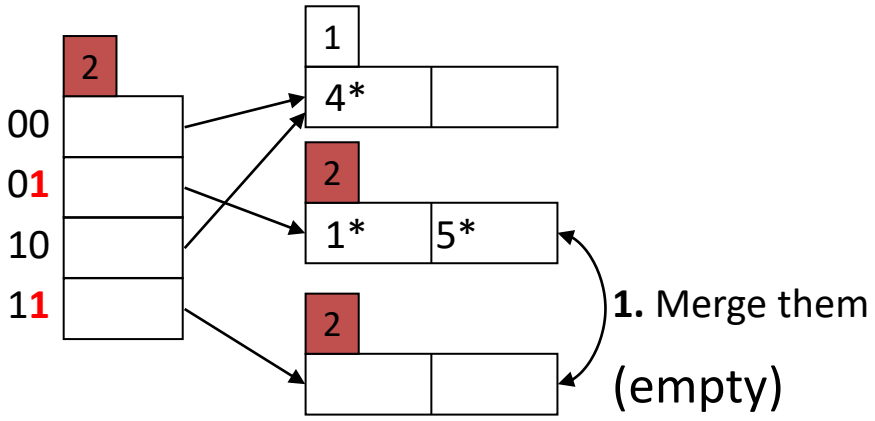
After delete 2:



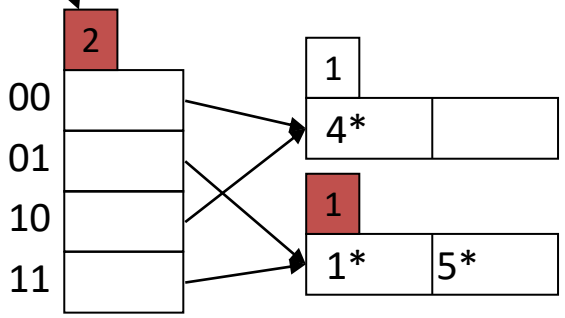
Decrease the local depth of the new bucket by 1.

Deletion of Extensible Hashing

Delete 7:

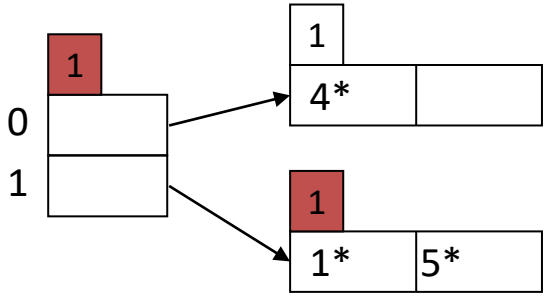


2. Halve the directory



If the mergence makes max(local depth of all buckets) < global depth, then halve the size of directory.

After delete 7:



Decrease the global depth of the new bucket by 1.

CSC3170A 2007-2008 1st Term – Q5

QUERY PROCESSING

CSC3170A 2007-2008 1st Term – Q5

Suppose there are two relations for a car rental company:

Customers (cid: integer, cname: string, age: real)

Reserves (cid: integer, carNO: integer, day: date)

Assumptions:

Reserves: 100 tuples per page, 1000 pages. Totally 100 cars, uniform distribution.

Customers: 80 tuples per page, 500 pages.

The buffer size is 20 pages.

a) Assume ***cid* in Reserves is a foreign key** on **Customers**. Please estimate the size of **Customers ⋈ Reserves** in terms of number of tuples.

100 x 1000 = 100000

CSC3170A 2007-2008 1st Term – Q5

```
SELECT Distinct R.cid, R.carNO
FROM Reserves R
```

Suppose the query is executed in the following steps:

1. **Scan *R* and write *cid* and *carNO* of each tuple to a temporary file *T*.**
2. Sort ***T*** based on both *cid* and *carNO.*
3. Scan the sorted file, compare the adjacent tuples and discard duplicate tuples

b) Assume the size of ***T*** is 250 pages. Estimate the number of page accesses to process the above query. *Hint*: the number of page accesses to sort a file is $2 \times M \times (\lceil \log_{B-1} M/B \rceil + 1)$.

1000 + 250 = 1250 I/Os

CSC3170A 2007-2008 1st Term – Q5

```
SELECT Distinct R.cid, R.carNO
FROM Reserves R
```

Suppose the query is executed in the following steps:

1. Scan ***R*** and write *cid* and *carNO* of each tuple to a temporary file ***T***.
2. **Sort *T* based on both *cid* and *carNO.***
3. Scan the sorted file, compare the adjacent tuples and discard duplicate tuples

b) Assume the size of ***T*** is 250 pages. Estimate the number of page accesses to process the above query. *Hint*: the number of page accesses to sort a file is $2 \times M \times (\lceil \log_{B-1} M/B \rceil + 1)$.

$$2 \times 250 \times (\lceil \log_{20-1} 250/20 \rceil + 1) = 1000 \text{ I/Os}$$

CSC3170A 2007-2008 1st Term – Q5

```
SELECT Distinct R.cid, R.carNO
FROM Reserves R
```

Suppose the query is executed in the following steps:

1. Scan ***R*** and write *cid* and *carNO* of each tuple to a temporary file ***T***.
2. Sort ***T*** based on both *cid* and *carNO.*
3. **Scan the sorted file, compare the adjacent tuples and discard duplicate tuples**

b) Assume the size of ***T*** is 250 pages. Estimate the number of page accesses to process the above query. *Hint*: the number of page accesses to sort a file is $2 \times M \times (\lceil \log_{B-1} M/B \rceil + 1)$.

250 I/Os

CSC3170A 2007-2008 1st Term – Q5

```
SELECT Distinct R.cid, R.carNO
FROM Reserves R
```

Suppose the query is executed in the following steps:

1. Scan ***R*** and write *cid* and *carNO* of each tuple to a temporary file ***T***.
2. Sort ***T*** based on both *cid* and *carNO.*
3. Scan the sorted file, compare the adjacent tuples and discard duplicate tuples

b) Assume the size of ***T*** is 250 pages. Estimate the number of page accesses to process the above query. *Hint*: the number of page accesses to sort a file is $2 \times M \times (\lceil \log_{B-1} M/B \rceil + 1)$.

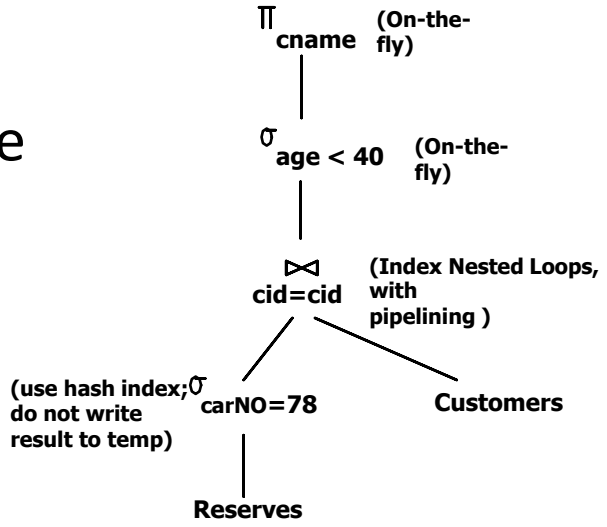
1250 + 1000 + 250 = 2500 I/Os

CSC3170A 2007-2008 1st Term – Q5

Assume there is clustered hash index on *carNO* of ***Reserves,*** and hash index on *cid* of ***Customers***. Consider the following query:

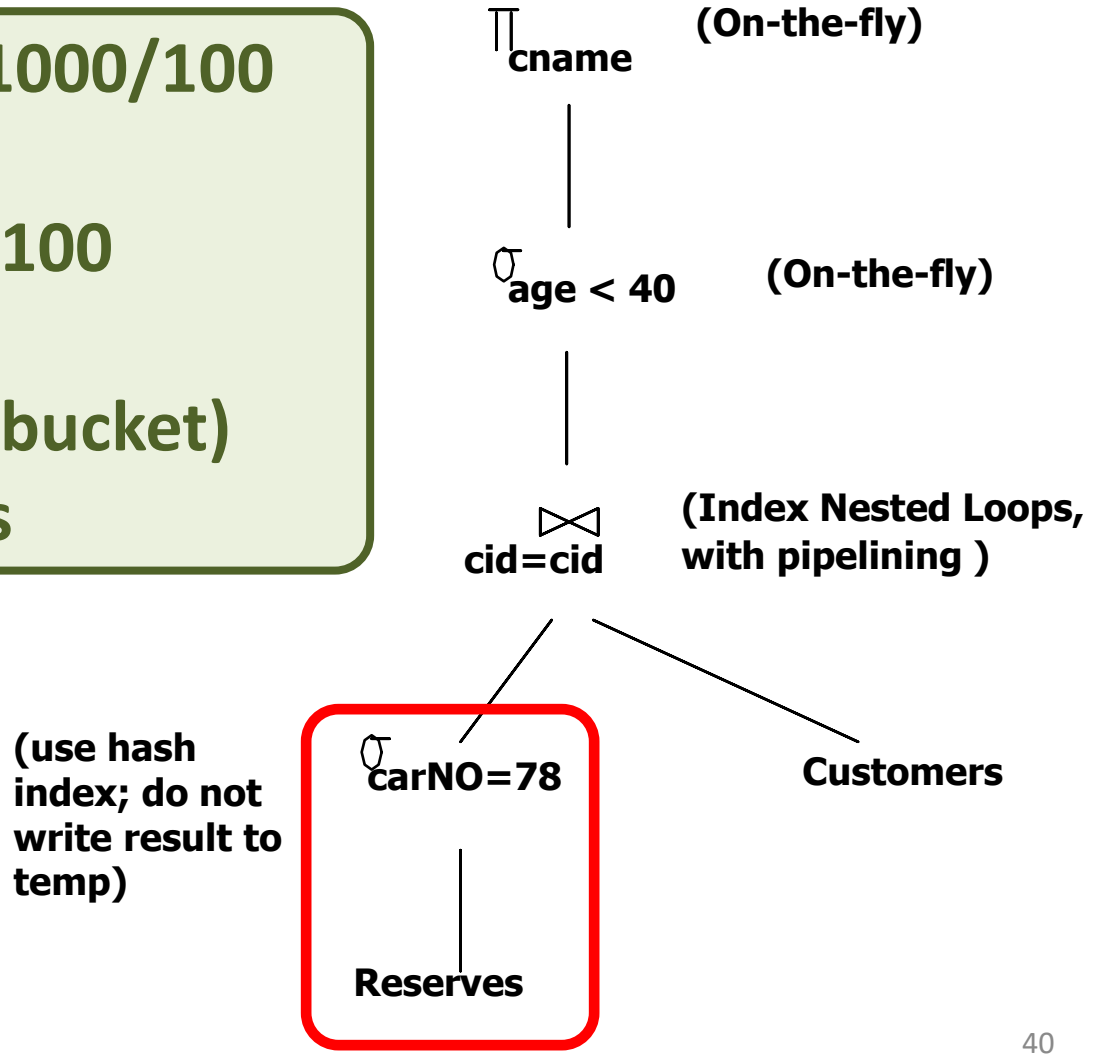
```
SELECT  C.cname
FROM  Reserves R, Customers C
WHERE  R.cid=C.cid AND R.carNO=78 AND C.age<40
```

c) Please estimate the number of page accesses for the query. Assume that there is no overflow bucket and the directories of both the hash structures are in the main memory.



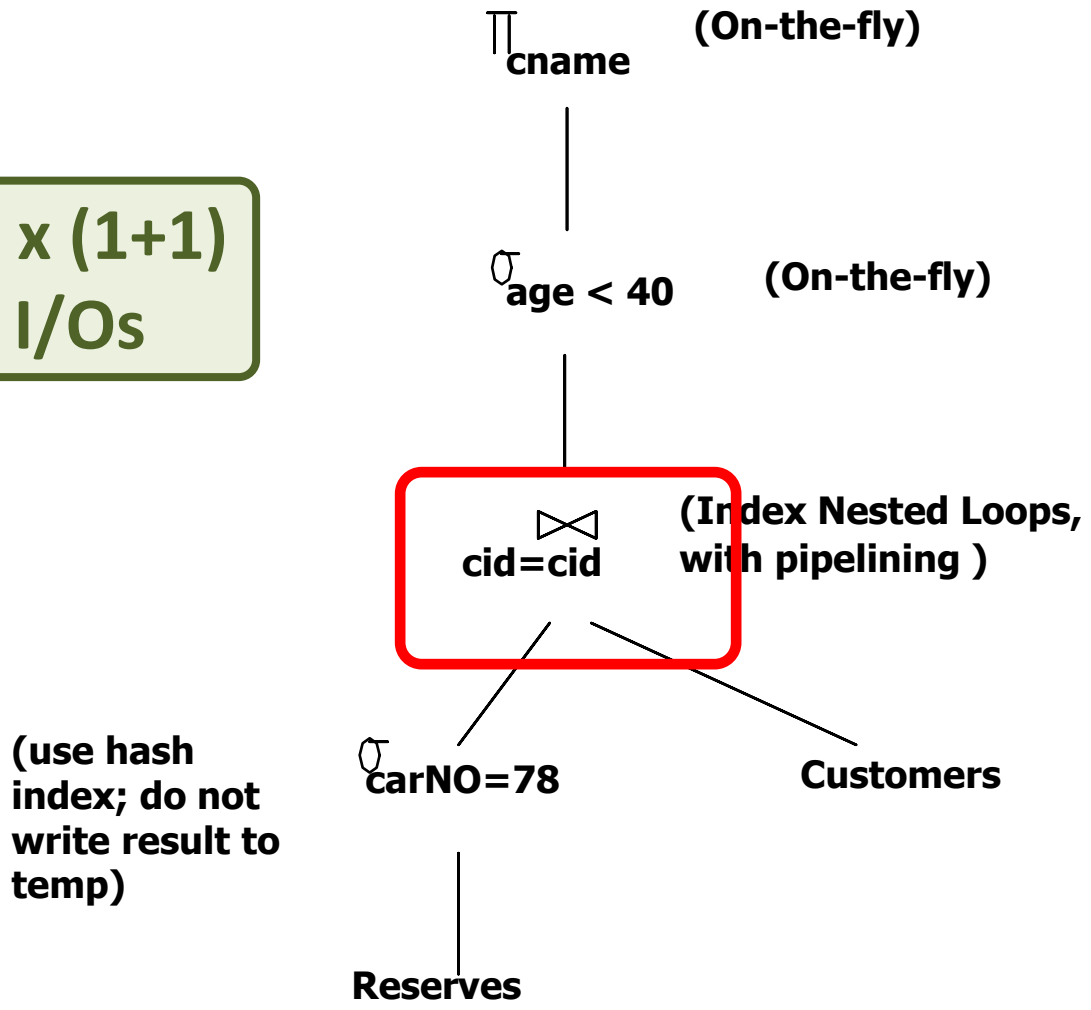
CSC3170A 2007-2008 1st Term – Q5

No. of tuples = 100x1000/100
= 1000
No. of pages = 1000/100
= 10
Page access = 10 + 1(bucket)
= 11 I/Os



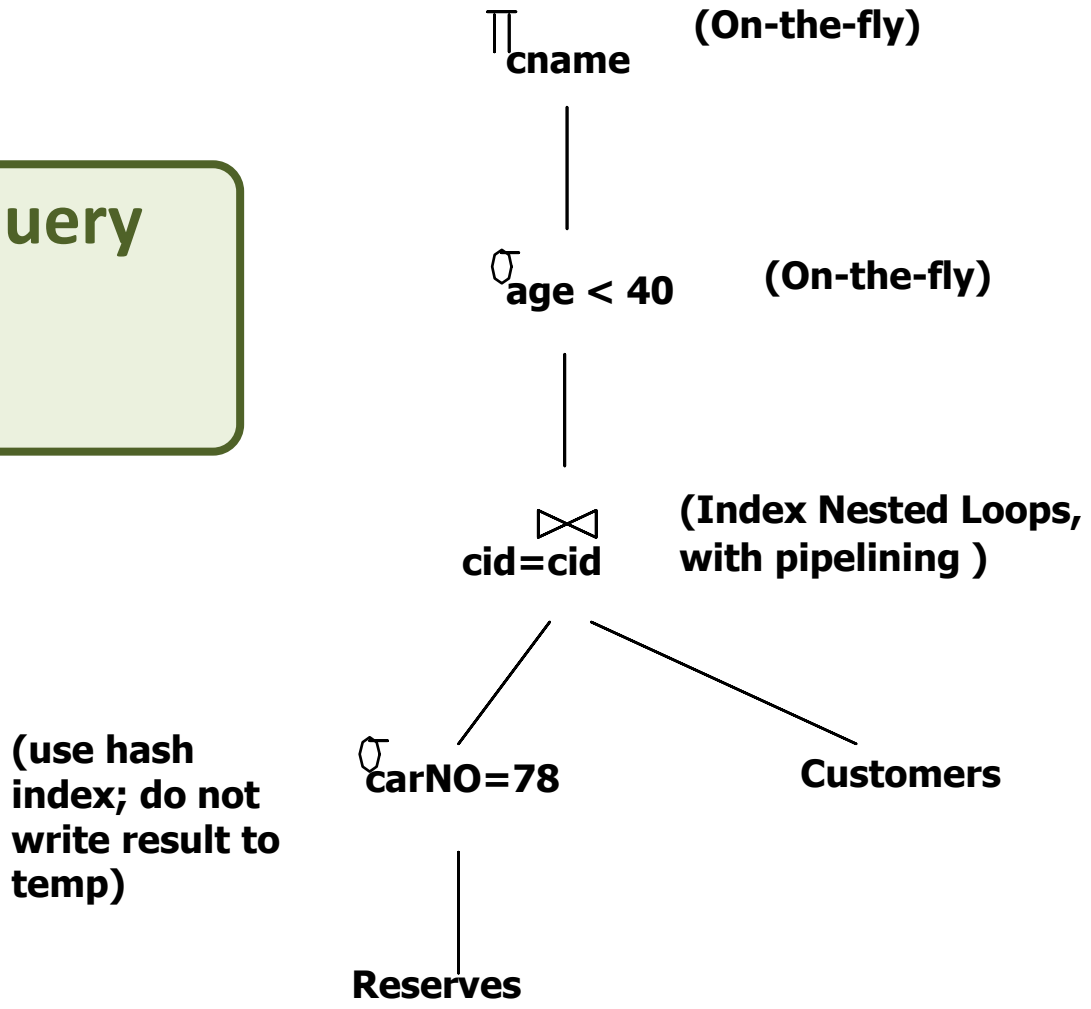
CSC3170A 2007-2008 1st Term – Q5

Page access = 1000 x (1+1)
= 2000 I/Os



CSC3170A 2007-2008 1st Term – Q5

Total cost of the query
= 11 + 2000
= 2011



CONCURRENCY CONTROL

CSC3170 2009-2010 1st Term – Q7

$w_3[a]$ $r_2[a]$ $w_1[a]$ $w_2[a]$

(a) Please state all the ***read-from*** relations.

T_2 reads a from T_3

CSC3170 2009-2010 1st Term – Q7

$w_3[a]\ r_2[a]\ w_1[a]\ \mathbf{w_2[a]}$

(b) Please state which transaction executes the ***final write*** operation on ***a***.

T_2 issues the final write on ***a***

CSC3170 2009-2010 1st Term – Q7

$w_3[a]\ r_2[a]\ w_1[a]\ w_2[a]$

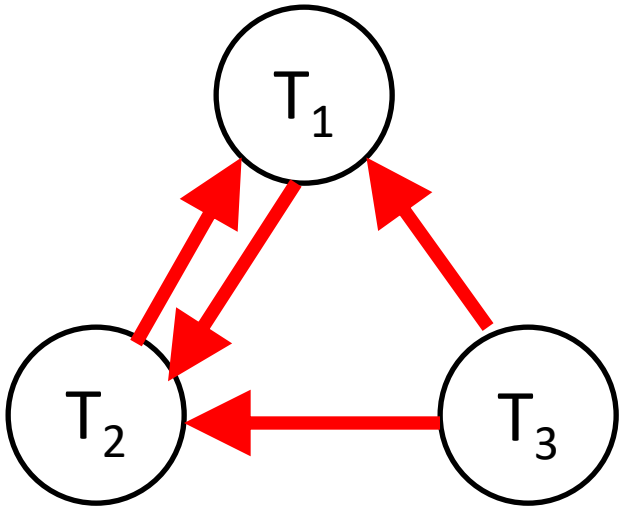
(b) Is the history **view serializable**? Why? Please state the **serialization order** if it is view serializable.

Yes. The history has the same read from relation and same final write operation as $\mathbf{T_1\ T_3\ T_2}$

CSC3170 2009-2010 1st Term – Q7

$w_3[a]\ r_2[a]\ \mathbf{w_1[a]\ w_2[a]}$

(d) Draw the serialization graph of the above history.



CSC3170 2009-2010 1st Term – Q7

$w_3[a]\ r_2[a]\ w_1[a]\ w_2[a]$

(e) Is the history **conflict serializable**? Why? Please state the serialization order if it is conflict serializable.

No, because there is a cycle in the serialization graph.