Indexes and their Effects on SQL Queries

Big Data Infrastructure, SA 2016

Alberto Tonon - *alberto@exascale.info*

Connecting to

Powered by ASQ

~ Today's Menu ~

1. What do you remember about *indexes*?
2. Indexes in action!
3. Nested SQL statements (if time permits)

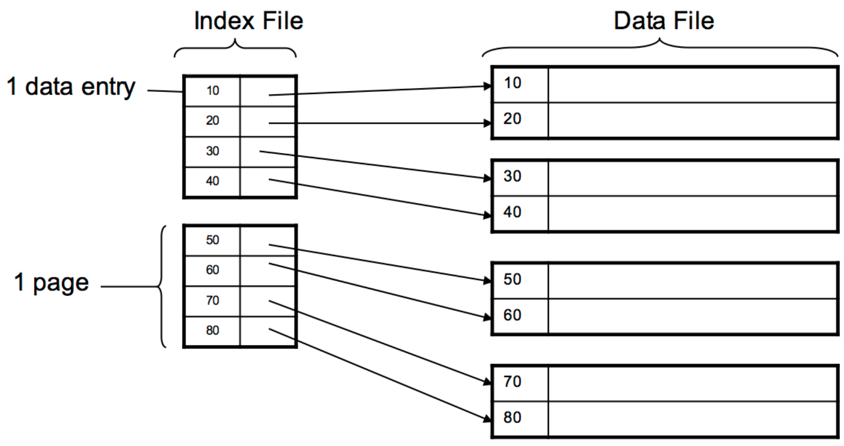
Primary Indexes

How many primary indexes can be defined on a table?

- ○ One for each field
- ○ One for each primary key field
- ○ Only one
- ○ An arbitrary amount

How many primary indexes can we define on a table?

The records in the data files are sorted by the key on which the index is defined. You cannot sort the records with more than one criterion, so, only one primary index per table can be defined.



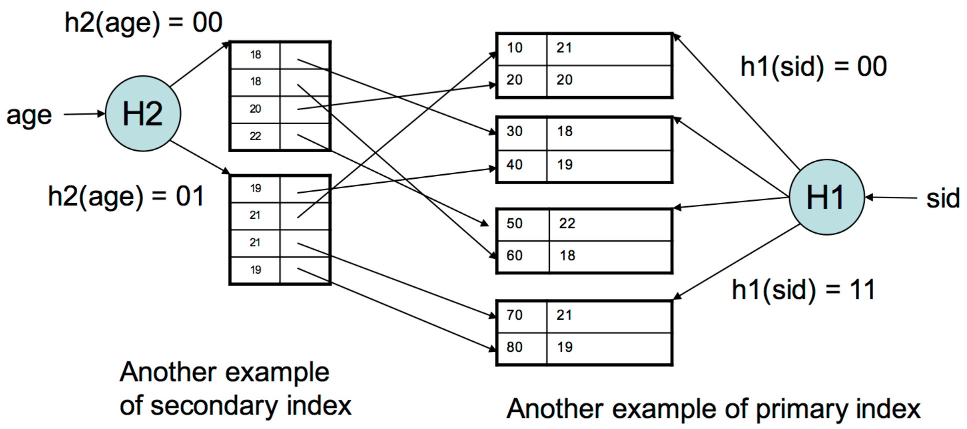
Hash Indexes

Which operation types are hash indexes good for?

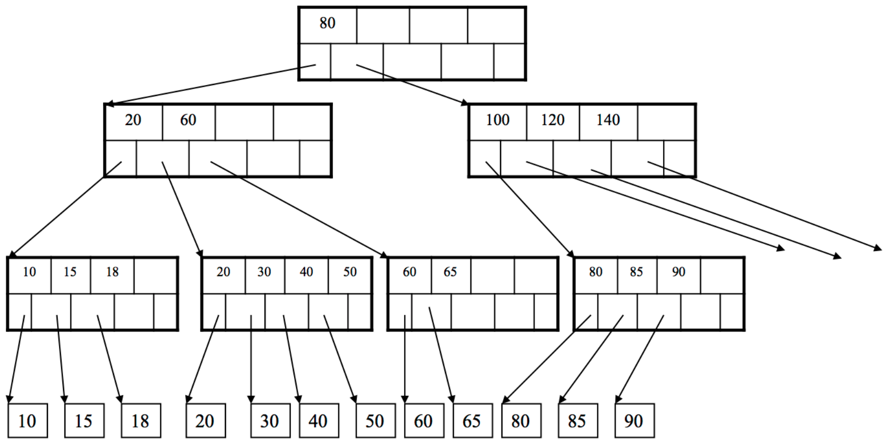
- ☐ Range queries
- ☐ Spatial queries
- ☐ Keyword queries
- ☐ Order-By queries
- ☐ Point queries

Which operation types are hash indexes good for?

Only point queries: hash indexes are really fast when it comes to retrieving a record given a specific value of the indexed attribute.



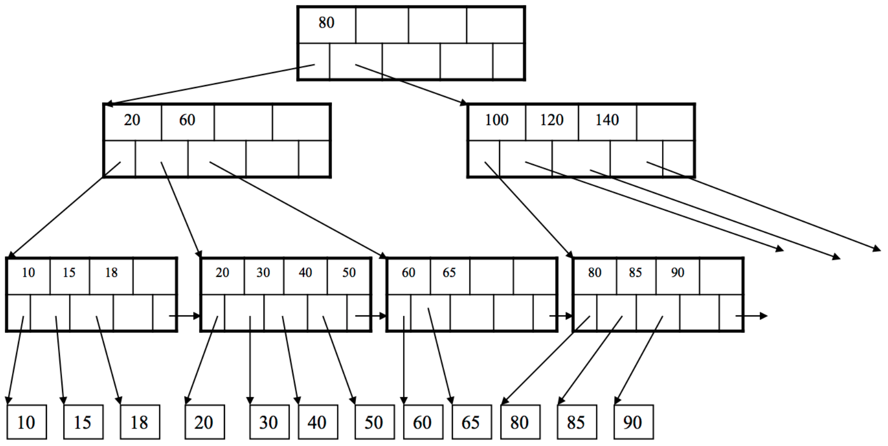
What's missing in this B$^+$-tree?



(hint: it's not a number)

How would you use the B$^+$-tree depicted below to answer the following query?

```
SELECT * FROM Person WHERE age BETWEEN 20 AND 63;
```



Answer:

Powered by ASQ

Indexes in Action!

Back to SQL!

Powered by ASQ

```
supplier(suppID, name)
product(prodID, name)
component(compID, name)
composed_by(prodID, compID, quantity)
sells(suppID, compID, price)
```

Select the name of the suppliers who sell the most expensive components.

```
SELECT ...
    FROM ...
    WHERE ... >=ALL (
      ...
    );

```

Powered by ASQ

supplier(<u>suppID</u>, name)
product(<u>prodID</u>, name)
component(<u>compID</u>, name)
composed_by(<u>prodID, compID</u>, quantity)
sells(<u>suppID, compID</u>, price)

```
SELECT DISTINCT name
FROM supplier NATURAL JOIN sells
WHERE price >=ALL (
  SELECT price
  FROM sells
);
```

```
supplier(suppID, name)
product(prodID, name)
component(compID, name)
composed_by(prodID, compID, quantity)
sells(suppID, compID, price)
```

Select the ID of all suppliers selling exactly one component.

`sells(suppID, compID, price)`

Given a certain component sold by SupplierX, there shouldn't exist another component sold by SupplierX.

```
SELECT suppID
FROM sells AS s0
WHERE NOT EXISTS (
  SELECT *
  FROM sells AS s1
  WHERE s1.suppID = s0.suppID
  AND s1.compID <> s0.compID
);
```

```
supplier(suppID, name)
product(prodID, name)
component(compID, name)
composed_by(prodID, compID, quantity)
sells(suppID, compID, price)
```

Select all the pairs of suppliers who sell exactly the same components.

Powered by ASQ

`sells(suppID, compID, price)`

Fixed a pair of suppliers (s1, s2), there shouldn't exist a component sold by s1 and not sold by s2, and a component sold by s2 but not sold by s1.

```
SELECT s1."suppID", s2."suppID"
FROM sells AS s1, sells AS s2
WHERE s1.suppID < s2.suppID
AND NOT EXISTS ( -- a component sold by s1 but not sold by
  SELECT *  FROM sells WHERE suppID = s1.suppID
  AND compID NOT IN ( SELECT compID FROM sells
    WHERE suppID = s2.suppID) )

AND NOT EXISTS ( -- a component sold by s2 but not sold by
  SELECT *  FROM sells WHERE suppID = s2.suppID
  AND compID NOT IN ( SELECT compID FROM sells
    WHERE suppID = s1.suppID) )
```

That's all, folks!

- *Constructive* feedback on this interactive presentation is much appreciated.
- If you have problems with the homeworks or you don't understand something contact me.

alberto@exascale.info