CSCI317 – Database Performance Tuning

Tutorial - To Index or Not to Index?

Sionggo Japit
sjapit@uow.edu.au

17 August 2022

Question

To index or not to index, that is the question !

Consider the following relational schemas:

AGENT(emp#, fname, lname, phone#)
primary key = (emp#)
candidate key = (phone#)

BUYER(phone#, fname, lname)
primary key = (phone#)

SELLER(phone#, fname, lname)
primary key = (phone#)

HOUSE(city,street,house#,category)
primary key = (city,street,house#)

TRANSACTION(city, street, house#, sphone#, bphone#, emp#, price)

primary key = (city, street, house#, sphone#)

foreign key = (sphone#) references SELLER(phone#)

foreign key = (bphone#) references BUYER(phone#)

foreign key = (emp#) references AGENT(emp#)

foreign key = (city, street, house#) references HOUSE(city, street, house#)

AGENT(emp#, fname, lname, phone#)
primary key = (emp#)
candidate key = (phone#)

BUYER(phone#, fname, lname)
primary key = (phone#)

SELLER(phone#, fname, lname)
primary key = (phone#)

HOUSE(city,street,house#,category)
primary key = (city,street,house#)

TRANSACTION(city, street, house#, sphone#, bphone#, emp#, price)
primary key = (city, street, house#, sphone#)
foreign key = (sphone#) references SELLER(phone#)
foreign key = (bphone#) references BUYER(phone#)
foreign key = (emp#) references AGENT(emp#)
foreign key = (city, street, house#) references HOUSE(city, street, house#)

Consider the following SELECT statements:

```
SELECT *
FROM   HOUSE
WHERE  CITY = 'SF'
AND    STREET = 'Sunshine Blvd.';
```

Solution

There is **NO** need to create and index as the index on primary key of a relational table HOUSE can be used to compute a query. The string constants 'SF' and 'Sunshine Blvd.' will be concatenated into 'SFSunshine Blvd.' and index on primary-key will be used to find key 'SFSunshine Blvd.<smallest house#)>' at leaf level of the index. When found, the leaf level of the index will be traversed for all prefixes of index key equal to 'SFSunshine Blvd.'. The row identifiers associated with the keys found will be used to access the rows in a relational table HOUSE.

AGENT(emp#, fname, lname, phone#)
primary key = (emp#)
candidate key = (phone#)

BUYER(phone#, fname, lname)
primary key = (phone#)

SELLER(phone#, fname, lname)
primary key = (phone#)

HOUSE(city,street,house#,category)
primary key = (city,street,house#)

TRANSACTION(city, street, house#, sphone#, bphone#, emp#, price)
primary key = (city, street, house#, sphone#)
foreign key = (sphone#) references SELLER(phone#)
foreign key = (bphone#) references BUYER(phone#)
foreign key = (emp#) references AGENT(emp#)
foreign key = (city, street, house#) references HOUSE(city, street, house#)

```
SELECT   BPHONE#, PRICE
FROM     TRANSACTION JOIN HOUSE
         ON TRANSACTION.CITY = HOUSE.CITY
AND      TRANSACTION.STREET = HOUSE.STREET
AND      TRANSACTION.HOUSE# = HOUSE.HOUSE#;
```

Solution

There is **NO** need to create a new index as the index on primary key of relational table TRANSACTION will be used to compute a join operation on TRANSACTION and HOUSE. A relational table HOUSE will be traversed in a row-by-row mode and with each row taken from HOUSE an index on primary key of TRANSACTION will be used to find all rows in a relational table TRANSACTION that can be joined with the rows from HOUSE. The index on primary key of relational table TRANSACTION can be used because the attributes used in join condition form a prefix of an index key.

AGENT(emp#, fname, lname, phone#)
primary key = (emp#)
candidate key = (phone#)

BUYER(phone#, fname, lname)
primary key = (phone#)

SELLER(phone#, fname, lname)
primary key = (phone#)

HOUSE(city,street,house#,category)
primary key = (city,street,house#)

TRANSACTION(city, street, house#, sphone#, bphone#, emp#, price)
primary key = (city, street, house#, sphone#)
foreign key = (sphone#) references SELLER(phone#)
foreign key = (bphone#) references BUYER(phone#)
foreign key = (emp#) references AGENT(emp#)
foreign key = (city, street, house#) references HOUSE(city, street, house#)

```
SELECT    FNAME, LNAME
  FROM    AGENT
  WHERE   PHONE# = 1234567;
```

Solution

- Since PHONE# is a candidate key, an index on the attribute PHONE# may be created to help speed up the access.

  CREATE INDEX T3 ON AGENT(PHONE#);

- An index T3 will be traversed vertically to find a value of index key equal to 1234567. Then, a row identifier associated with a key value found will be used to access a relational table AGENT.

- Note: In a database system where candidate keys are automatically indexed there is no need to create and index.

AGENT(emp#, fname, lname, phone#)
primary key = (emp#)
candidate key = (phone#)

BUYER(phone#, fname, lname)
primary key = (phone#)

SELLER(phone#, fname, lname)
primary key = (phone#)

HOUSE(city,street,house#,category)
primary key = (city,street,house#)

TRANSACTION(city, street, house#, sphone#, bphone#, emp#, price)
primary key = (city, street, house#, sphone#)
foreign key = (sphone#) references SELLER(phone#)
foreign key = (bphone#) references BUYER(phone#)
foreign key = (emp#) references AGENT(emp#)
foreign key = (city, street, house#) references HOUSE(city, street, house#)

```
SELECT    FNAME, LNAME, PRICE
  FROM    AGENT JOIN TRANSACTION
          ON AGENT.EMP# = TRANSACTION.EMP#
  WHERE   PHONE# = 7654321
  AND     PRICE > 1000000;
```

Solution

CREATE INDEX T41 ON AGENT(PHONE#);
CREATE INDEX T42 ON TRANSACTION(E#)

An index T41 will be vertically traversed to find a value of index key equal to 7654321. Next a row identifier associated with an index key will be used to access a relational table AGENT and a value of attribute EMP# will be extracted from a row. Next, the value of EMP# will be used to vertically traverse an index T42 to find all transactions handled by a given employee. When an index key with a value of EMP# found in the previous step is located at leaf level of an index T42 the row identifiers associated with the index key will be used to access a relational table TRANSACTION and to compute a condition PRICE>1000000.

Solution

An index on PRICE i.e.

CREATE INDEX T43 ON TRANSACTION(PRICE);

can also be used to reduce the total number of rows accessed in a relational table TRANSACTION. In such a case an index T43 has to be vertically traversed with a key 1000000 and later on its leaf level should be horizontally traversed to find all identifiers of all rows that satisfy a condition PRICE>1000000. Next, a set of row identifiers found should be intersected with a set of row identifiers found from vertical traversal of index T42.

AGENT(emp#, fname, lname, phone#)
primary key = (emp#)
candidate key = (phone#)

BUYER(phone#, fname, lname)
primary key = (phone#)

SELLER(phone#, fname, lname)
primary key = (phone#)

HOUSE(city,street,house#,category)
primary key = (city,street,house#)

TRANSACTION(city, street, house#, sphone#, bphone#, emp#, price)
primary key = (city, street, house#, sphone#)
foreign key = (sphone#) references SELLER(phone#)
foreign key = (bphone#) references BUYER(phone#)
foreign key = (emp#) references AGENT(emp#)
foreign key = (city, street, house#) references HOUSE(city, street, house#)

```
SELECT    COUNT(DISTINCT PRICE)
  FROM    TRANSACTION
  WHERE   PRICE > 50000;
```

Solution

- **CREATE INDEX T5 ON TRANSACTION(PRICE);**
- An index T5 will be vertically traversed to find a key 50000 and later on leaf level of the index will be horizontally traversed to count all prices whose value is greater than 50000. No access to relational table transaction is needed.

(Note if the index T43 was created in the previous question, there is no need to create index T5.)

Solution

Finally, the smallest number of indexes that improve the performance of all queries listed above is as follows:

- **CREATE INDEX T3 ON AGENT(PHONE#);**
- **CREATE INDEX T41 ON AGENT(PHONE#);**
- **CREATE INDEX T42 ON TRANSACTION(E#);**
- **CREATE INDEX T43 ON TRANSACTION(PRICE);**
- **CREATE INDEX T5 ON TRANSACTION(PRICE); (This is the same of index T43. If Index T43 is created, then index T5 is not required.)**