Brian Erichsen Fagundes
Lab 9: Indexes
CS6016 - MSD
UofU
Summer Semester 2024

Part 1 - Selecting Indexes

Find all employees that started after a certain date → Index(Start Date).

Find all employees that started on a certain date, and worked until at least another certain date.

a - Index(Start Date, End Date)

Minimal set of additional secondary indexes I should add to each table start date and end date because searching for former-employee records using start date we can find all employees that started after a certain date. Similarly, End Date where we have a start date already established helps us find which employees started in a specific time period from x to y time period.

Database that contains student grades

All Students with a grade better than 'B'

b - Index(Grade)

No need for extra indexes since we are only searching for specific grades.

All Classes where any student earned a grade worse than 'D'

c - Index(Grade)

No need for extra indexes since we are only searching for specific grades.

All classes ordered by class name

d - Index('className')

No need for extra indexes since we are only searching by class name.

All students who earned an 'A' in certain class

e - Index('className', Grade)

Since we need to know in which class a student received a specific grade then we must use className and grade where that grade is associated with the specific class.

Queries on the chess database

Select Name from Players where Elo >= 2050

f - CREATE INDEX idx_elo ON Players (Elo);

Select Name, gID from Players join Games where pID=WhitePlayer

g - Existing Index: Players.pID

CREATE INDEX idx_white_player ON Games (WhitePlayer);

Queries on the Library database

Select * from Inventory natural join CheckedOut;

h - Inventory (Serial) is already the primary key for Inventory and same for CheckOut table. Hence None.

More Library queries

Select * from Inventory natural join CheckedOut where CardNum=2;

i - CREATE INDEX idx_checkedOut_cardnum ON CheckedOut (CardNum);

Select * from Patrons natural join CheckedOut;

j - Patrons (CardNum) however Patrons.CardNum is already the primary key for the Patrons table; hence None.

Still more library queries

k - Inventory(ISBN)

The table Titles has PK in ISBN, Inventory has PK on Serial; having an index over Inventory.ISBN then we can get a proper match for the same title and serial.

Part 2 - B+ Tree Index Structures

Students table:

- The number of rows of the table that we can place into the first leaf node before we split is related to how many bytes each row needs. Each row is - studentID 4 bytes, className 10 bytes, Grade 1 byte; which in total is 15 bytes. In this, we use 4096 byte pages hence the answer is 4096 / 15 → ~= 272.
- The maximum number of keys stored in an internal node of the primary index is related to the size of each key - studentID 4 bytes and className 10 bytes hence the answer is 4096 / 14 → ~= 293.
- The maximum number of rows in the table if the primary index has a height of 1 is related to how many keys the internal node can have and each key points to a leaf node can hold; in this case that would be 272 + 1 * 293 → 79989
- Since the minimum number of children of a root node is 2 then we have a minimum number of leaf nodes being 2. The minimum number of rows per leaf node is 273 / 2 → ~= 136. Hence the minimum number of rows in the table where primary index has height 1 is 2 * 136 → 272
- If there is a secondary index on Grade, the maximum number of entries a leaf node can hold in the secondary index is related to the size of the secondary index entry – in this case it would 1 byte for grade char + the size of pointer, assuming a size of 4 bytes then each entry would be total of 5 bytes. Dividing total leaf node capacity of 4096 by 5 → ~= 819 entries approximately.

Another table

- If we assume that rows occupy 128 bytes; the maximum number of leaf nodes in the primary index if the table contains 48 rows is related to rows per leaf node can where each can hold 4096 / 128 → 32 rows and a leaf node should hold at least 32 / 2 → 16 rows. The maximum number of leaf nodes would be 48 / 32 → 2 when rounded up.

- Minimum number of leaf nodes for 48 rows would be the minimum number of leaf nodes = 48 divided by how many each can hold. Since each leaf node must be at least 50% full so each can hold 16 rows hence 48 / 16 → 3 leaf nodes.