Σχεδίαση Βάσεων Δεδομένων και Κατανεμημένες ΒΔ

1 η Εργασία στη Σχεδίαση ΒΔ 2021-2022

Σιγάλας Σπυρίδων 21991
Παππάς Στέφανος 218131
Μαυροπουλος Ανδρεας 217129

Μέρος 1:

1) Το μέγεθος του αρχείου αν είναι αρχείο σωρού:

Απο την εκφωνιση εχουμε τα εξεις στοιχεια :

r_A = 20000000 εγγραφές
S_A = 250 bytes
S_B = 2048 bytes

Αρα

Σε καθε block χωράνε floor(S_B/S_A)=(2048/250)= 8 εγγραφές
Μέγεθος αρχείου δεδομένων S_F =20000000/8 = 2500000 blocks

2) Το μέγεθος του αρχείου αν είναι αρχείο κατακερματισμού:
Εφοσον ειναι το ιδιο αρχειο αλλα ταξινομημενο θα πιασει τον ίδιο χώρο.

3)Πόσα επίπεδα έχει το B+ δέντρο συμπεριλαμβανομένου και του τελευταίου επιπέδου:

Πεδίο κλειδιού S_K= 8 bytes
Μέγεθος δείκτη block P= 8 bytes

Σε κάθε block ευρετηρίου εχουμε
floor(S_B/(S_K/P))=floor(2048/(8+8))=floor(2048/16)=128 εγγραφές

128+1=129
$\log_{129}(2500000)$= 3.031~4

Το B+ Δέντρο θα εχει 4 (επίπεδα μαζι με την ρίζα 5).

4)Πόσους κόμβους θα περιεχει το κάθε επίπεδο του, ποιο το μέγεθος του ευρετηρίου συνολικά:

Το πρωτο επιπεδο θα εχει 129 κομβους
Το δευτερο επιπεδο θα εχει 16.641 κομβους
Το τριτο επιπεδο θα εχει 2.146.689 κομβους
Το τεταρτο επιπεδο θα εχει 336.541
Για συνολικα 2.500.000 κομβους

Μέγεθος αρχείου ευρετηρίου S_I=ceiling(2500000/128)=19532 blocks

Μέγεθος του ευρετηρίου ειναι 19532 blocks

5)Αν το δέντρο σας γίνει B* ποια η απάντηση στο 4;
Εξηγήστε τη διαδικασία επίλυσης σε κάθε βήμα:

6)Ποιο θα είναι το κόστος αναζήτησης ισότητας για μια συγκεκριμένη τιμή που γνωρίζετε ότι εμφανίζεται 5 φορές σε όλο το αρχείο:

Επειδη ειναι αρχειο κατακερματισμου εχει ιδιο χρονο αναζητησης για οποιαδηποτε αναζητηση στο αρχειο

Μέρος 2:
Βρείτε τη σύνταξη της εντολής CREATE INDEX σε ORACLE, MySQL και PostgreSQL. Εξηγείστε τις παραμέτρους κάθε εντολής σε σχέση με τους διαθέσιμους τύπους ευρετηρίου που έχουμε δει στο μάθημα. Παρουσιάστε συγκριτικά (σε ένα πίνακα) τις δυνατότητες των τριών ΣΔΒΔ, ποια ευρετήρια υποστηρίζουν και ποια όχι η κάθε μια

Oracle
CREATE INDEX index_name

ON table_name(column1[column2,....])

MySql
CREATE INDEX index_name
ON table_name(column1,column2)

PostgreseSQL
CREATE INDEX index_name [method]
(
column_name1[],
column_name2[]
)

Βλέπουμε ότι και τα 3 έχουν παρόμοια αρχή για την δημιουργία του index. Και οι 3 γλώσσες όμως αλλάζουν δραστικά στον τρόπο με τον οποιον προσθέτουν columns. Επίσης και τα 3 έχουν Secondary indexes. Πιο συγκεκριμένα η postegresesql στην δημιουργία του index μπορεί να προσθέσει τον τρόπο με τον οποίον θα φτιαχτεί αυτό το index (btree, hash, gist, gin). Στην mysql μπορείς να καθορίσεις εύκολα το key block size, στην δημιουργία των index.

Μέρος 3:

Για αρχή δημιουργήσαμε με βάση το xsales τους πίνακες μας. Στην συνεχεία για το υποερωτημα α προσθέσαμε το agegroup σαν varchar σαν καινούργια στήλη. Για να πάρουμε λοιπόν το έτος από κάθε άτομο βγάλαμε από την ημερομηνία γέννησης το έτος το αφαιρέσαμε από το 2021 το και τέλος το συγκρίναμε. Για να γίνει αποδεκτό όμως και να μην έχουμε σφάλματα έπρεπε να μετατρέψουμε το birth_data σε χαρακτήρα. Το β υποερωτημα το αντιμετωπίσαμε χρησιμοποιωντας την απλή λογική ότι τα income_levels είναι σχεδόν ήδη ταξινομημένα. Άρα παίρνοντας με το subset το αρχικό γράμμα καταφέραμε να βγάλουμε της κατάλληλες κατηγορίες. Το γ υποερωτημα ακολούθησε την ίδια λογική με την β. Τέλος αφαιρέσαμε και μεταονομάσαμε της στήλες που έπρεπε για να φτιάξουμε την τελική εικόνα.

```
CREATE TABLE CUSTOMERS AS SELECT * FROM XSALES.CUSTOMERS;
CREATE TABLE PRODUCTS AS SELECT * FROM XSALES.PRODUCTS;
CREATE TABLE ORDERS AS SELECT * FROM XSALES.ORDERS;
CREATE TABLE ORDER_ITEMS AS SELECT * FROM XSALES.ORDER_ITEMS;
```

```
--- A
ALTER TABLE CUSTOMERS
ADD (AGEGROUP VARCHAR2(20));

UPDATE CUSTOMERS
SET
  CUSTOMERS.AGEGROUP = 'under 30'
WHERE 2021- TO_CHAR(BIRTH_DATE,'YYYY') < 30;

UPDATE CUSTOMERS
SET
  CUSTOMERS.AGEGROUP = '30-40'
WHERE 2021- TO_CHAR(BIRTH_DATE,'YYYY') >= 30 and 2021- TO_CHAR(BIRTH_DATE,'YYYY') < 40;

UPDATE CUSTOMERS
SET
  CUSTOMERS.AGEGROUP = '40-50'
WHERE 2021- TO_CHAR(BIRTH_DATE,'YYYY') >= 40 and 2021- TO_CHAR(BIRTH_DATE,'YYYY') < 50;

UPDATE CUSTOMERS
SET
  CUSTOMERS.AGEGROUP = '50-60'
WHERE 2021- TO_CHAR(BIRTH_DATE,'YYYY') >= 50 and 2021- TO_CHAR(BIRTH_DATE,'YYYY') < 60;

UPDATE CUSTOMERS
SET
  CUSTOMERS.AGEGROUP = '60-70'
WHERE 2021- TO_CHAR(BIRTH_DATE,'YYYY') >= 60 and 2021- TO_CHAR(BIRTH_DATE,'YYYY') < 70;

UPDATE CUSTOMERS
SET
  CUSTOMERS.AGEGROUP = 'over 70'
WHERE 2021- TO_CHAR(BIRTH_DATE,'YYYY') >= 70 ;
```

```
--- B
select substr(INCOME_LEVEL,1,1) from CUSTOMERS;
ALTER TABLE CUSTOMERS
ADD (INCOME_LEVEL VARCHAR2(20));

UPDATE CUSTOMERS
SET
    CUSTOMERS.INCOME_LEVEL = 'low income'
WHERE substr(INCOME_LEVEL,1,1) = 'A' or substr(INCOME_LEVEL,1,1) = 'B' or substr(INCOME_LEVEL,1,1) = 'C'
or substr(INCOME_LEVEL,1,1) = 'D' OR substr(INCOME_LEVEL,1,1) = 'E';

UPDATE CUSTOMERS
SET
    CUSTOMERS.INCOME_LEVEL = 'medium income'
WHERE substr(INCOME_LEVEL,1,1) = 'F' or substr(INCOME_LEVEL,1,1) = 'G' or substr(INCOME_LEVEL,1,1) = 'H'
or substr(INCOME_LEVEL,1,1) = 'I' OR substr(INCOME_LEVEL,1,1) = 'E';

UPDATE CUSTOMERS
SET
    CUSTOMERS.INCOME_LEVEL = 'high income'
WHERE substr(INCOME_LEVEL,1,1) = 'J' or substr(INCOME_LEVEL,1,1) = 'K' or substr(INCOME_LEVEL,1,1) = 'L';
```

```
--- G
UPDATE CUSTOMERS
SET
    MARITAL_STATUS = 'unknown'
WHERE MARITAL_STATUS is null ;

UPDATE CUSTOMERS
SET
    MARITAL_STATUS = 'married'
WHERE MARITAL_STATUS = 'Married' or MARITAL_STATUS = 'Mabsent' or MARITAL_STATUS = 'married' or
      MARITAL_STATUS = 'Married' or MARITAL_STATUS = 'Mar-AF' ;

UPDATE CUSTOMERS
SET
    MARITAL_STATUS = 'Single'
where MARITAL_STATUS = 'NeverM' or MARITAL_STATUS = 'single' or MARITAL_STATUS = 'Separ.' or
      MARITAL_STATUS ='Divorc.' or MARITAL_STATUS = 'Widowed' or MARITAL_STATUS='widow';
```

```
--Table alter
ALTER TABLE CUSTOMERS
RENAME COLUMN ID TO Customer_ID;
ALTER TABLE CUSTOMERS
DROP (NAME,BIRTH_DATE,CREDIT_LIMIT,USER_CREATED,DATE_CREATED,USER_MODIFIED,DATE_MODIFIED,EMAIL);
```

Μέρος 4: