מסדי נתונים – 236363 – תרגיל בית 2

שחר כץ 313574766 אילון קורנבוים 315677880

במסמך זה נסביר את מבנה מסד הנתונים (database) שלנו ואת אופן פעולתו

<u>מבנה מסד הנתונים</u>

בבסיס התוכנית אנו מתחזקים את הטבלאות הבאות:

הערה: נציין כי ברוב התיאורים הבאים, מהכרות עם התרגיל ברור למדי מה משמעות כל שדה ולכן חסכנו את ההסברים הטרוויאלים (לרבות המגבלות הנאכפות בעת כל הכנסה, לרבות IDs≥0 וכו') ובחרנו לפרט רק במקומות שנראה לנו חשוב להבהיר את רמת ההבנה שלנו. נציין שבסוף מסמך זה הוספנו את קטע הקוד SQL של בניית הטבלאות כדי להבהיר מה בדיוק רצינו לבדוק/לאכוף על השדות שלהן.

טבלת Tquery – Queries

size	purpose	queryID

טבלת TRAM – RAM

size	company	ramID

טבלת דיסקים – TDisk

size	free_space	speed	company	diskID

טבלת RAM המחוברים לדיסקים – DR, ממששת את היחס part of בין RAM לדיסקים.

ramID	diskID

הסבר: על מנת לייצג את יחסי השיוך בין RAM לבין דיסקים יצרנו טבלה בה משמעות קיום השורה (disk_id, ram_id) הינה שה-RAM ב-ram_id ID משויך (מחובר) לדיסק ב-disk_id ID. שני השדות בטבלה זו הינם foreign key מ-TRAM ו-TDisk (בהתאמה) וכן נאכפים ייחודיות קיום כל שורה רק פעם אחת בטבלה. טבלה זו מתעדכנת בעת הוספת/הסרת דיסקים ו-RAM (בנפרד) וכן על ידי הפעולות הייעודיות המוסיפות/מסירות RAM מדיסקים.

כלומר בזכות השימוש בתוך DR בforeign key מ- Tdisk,Tquery, מחיקה מהאחרונים תמחק גם מהראשון. כמו כן, בעת נסיון להכניס צמד, נרוויח בדיקה שאכן קיימת שאילתה כזו ודיסק כזה, על ידי אכיפת מפתח זר.

טבלת queries המורצים על דיסקים – DQ, ממששת את היחס on running בין שאילתות לדיסקים.

queryID	diskID

הסבר: על מנת לייצג queries המורצים על דיסקים יצרנו טבלה בה משמעות קיום השורה (disk_id, query_id) הינה שה-qyery ב-query _id ID מורץ לדיסק ב-disk_id ID. שני השדות בטבלה זו הינם foreign key מ-TRAM ו-TDisk (בהתאמה) וכן נאכפים ייחודיות קיום כל שורה רק פעם אחת בטבלה. טבלה זו מתעדכנת בעת הוספת/הסרת דיסקים ו-queries (בנפרד) וכן על ידי הפעולות הייעודיות המוסיפות/מסירות queries מדיסקים.

נציין במיוחד שבעת הוצאת/הכנסת שורה לטבלה זו אנו מעדכנים את השדה free_space בשורה המתאימה לדיסק ב-TDisk, למשל: בעת הוספת query עם ה-query_id ID לדיסק עם disk_ID ID אנו בודקים קיום של דיסק ו-query ב-TDisk, TQuery (בהתאמה), בודקים שבשורה המייצגת את disk_ID השדה free_sapce הינו גדול או שווה לגודל (size) ה-query הנתון, ואם אכן כן – מורידים את גודל זה מה-free_space של אותו הדיסק ומוסיפים את השורה (disk_ID, query_ID) ל-DQ.

באופן דומה, כאשר אנו מסירים את אותו ה-query מהדיסק או רק מוחקים את query מהמערכת, אנו רוצים להסיר כל שורה ב-DQ מהצורה (*, query_ID) (שבה אותו query_ID משתתף) לכן נעשה זאת יחד עם הוספת גודל אותו ה-query ("בחזרה") לכל דיסק הנמצא איתו ב-DQ.

בעזרת מנגנון זה, אנו מפקחים על גודלו הזמין של כל דיסק בכל רגע נתון.

Views

במסד הנתונים שלנו אנו משתמשים בשני views:

DCanRunQ : מייצרת את אוסף הצירופים האפשריים להרצת Query על כל דיסק (בצירוף גודלו של ה-query שניתן להוסיף)

size	queryID	diskID

שורה בטבלה אומרת שניתן להוסיף את queryID (שגודלו הוא size) לדיסק בעל המזהה diskID.

D_total_RAM : טבלה שמייצגת עבור כל דיסק במסד את סכום גדלי ה-RAM שמורכבים עליו (שנוספו לו).

totalRam	diskID

(עבור דיסק שלא נוסף לו אף RAM אז totalRam יהיה שווה 0).

API

כפי שציינו, החלטנו להתמקד במסמך בהסברים שאינם טריוויאלים, לפיכך חסכנו פונקציות מהצורה ...get/add ואנו מקווים ששאר ההסברים במסמך ישכנעו אתכם ברמת ההבנה שלנו.

deleteQuery/Disk/RAM

פונקציות המסירות מממסד הנתונים **כל** ייצוג של דיסק/query/RAM. אנו מוחקים את הייצוג מהטבלאות TQuery/TDiskqTRAM (בהתאמה לסוג הקומפוננט) וכן ייצוגים בטבלאות הנוספות DQ (עבור דיסקים ו-queries) ו-DR (עבור דיסקים ו-RAM) הגוררות הורדת שורה שלמה מאותן הטבלאות (לא משאירים שורה "חצי מלאה"). נזכיר כי תוך כדי מחיקה של query אנו מבצעים "ניקוי" מ-DQ של ייצוגו תוך הגדלת כל free_space של דיסק בגודלו כפי שהסברנו בחלק המתאר את DQ. המחיקה מDQ, DR נעשית אוטמית על ידי מסד הנתונים בזכות השימוש ב: (לדוגמא עבור DQ ובאופן דומה עבור DR)

FOREIGN KEY (diskID) REFERENCES TDisk(diskID) **ON DELETE CASCADE**

FOREIGN KEY (queryID) REFERENCES TQuery(queryID) **ON DELETE CASCADE**

averageSizeQueriesOnDisk

אנו מייצרים טבלה זמנית הכוללת את כל ה-queries הרצים על הדיסק המבוקש על ידי הצלבת הטבלאות DQ, TQuery ומחשבים את ממוצע הגדלים שלהם. אנו יודעים אילו queries רצים על אותו הדיסק על ידי DQ ואנו יודעים את גודל כל query על ידי TQuery, כל שנותר הוא לסכום את עמודת size בטבלה הזמנית.

diskTotalRAM

אנו מייצרים טבלה זמנית (תת טבלה של DR) עם עמודת המספר מזהה ורק השורות המתאימות לדיסק המבוקש. כלומר, מספרים מזהים של כל הRAMים שמחוברים לדיסק זה. כעת נותר לסכום את עמודת size בטבלת TRAM, כאשר סוכמים רק מהשורות ששדה הזיהוי שלהם נמצא בטלה הזמנית הנ"ל.

getCostForPurpose

הצלבנו את כל המידע בין שאילתות ודיסקים אשר מקיימים את היחס (רץ על) בעזרת הצלבה בין טבלת השאילתות לטבלה היחס, עם התנאי שהמספר המזהה שווה ואז עוד הצלבה עם טבלת הדיסקים, עם אותו תנאי. מטבלה זו סיננו את השורות שבהם לשאילתה יש את המטרה הרצויה, לבסוף בחרנו רק את העמודות של מחיר, גודל, ועוד עמודה שהערך בה מחושב לכל שורה בטלבת התוצאה – והיא המכפלה בין שני הנ"ל. מטבלה זו רק נותר לסכום את ערכי העמודה של המכפלות.

getQueriesCanBeAddedToDisk

ראשית ניצור טבלה זמנית (תת טבלה של הדיסקים) שהיא למעשה שורה אחת – שמתאימה למספר המזהה של הדיסק, והיא גם עמודה אחת – המקום הפנוי. נצליב את הטבלה הנ"ל עם טבלת כל השאילתות. מהטבלה שקיבלנו נסנן את השורות שבהן הגודל קטן/שווה למקום פנוי, ולבסוף ניקח רק את עמודת המספרים המזהים (ונמיין ולאחר מכן נחתוך את ה-5 עליונים).

getQueriesCanBeAddedToDiskAndRAM

ראשית ניצור טבלה זמנית שהיא הצלבה של DR (נמצא ב) עם TRAM ונסנן לקבלת השורות שמתאימות למזהה של הדיסק, כעת נסכום את עמודה הגודל וקיבלנו טבלה עם ערך יחיד שהוא סך הRAM על הדיסק המבוקש. כעת הפתרון כמעט זהה לסעיף הקודם אלא שבתנאי על סינון השורות, בדקנו שהגודל קטן/שווה למקום פנוי **וגם** קטן/שווה מהסכימה של ערכי הטבלה שהסברתי.

נשמע שVIEW היה יכול להתאים לחזרה בין שתי הפונ', אלא שVIEW נותן ערך קבוע, תת טבלה מסוימת, ואילו אנחנו מעוניינים בשורה ספציפית של דיסק ספציפי, ויותר יעיל לחשב את השורה שלו ולהצליב אותה בלבד, מאשר VIEW של כל המקומות הפנויים של כל הדיסקים

isCompanyExclusive

שואלים 2 שאילתות עם AND ביניהן: הראשונה מוודאת קיום של הדיסק המבוקש. בשניה בודקים שלא קיים אף RAM שמחובר לדיסק זה ובעל חברה אחרת – מצליבים את השורה של הדיסק עם היחס (מורכב ב) ועם טבלת הRAM ומחפשים קיום של אי השיוויון.

getConflictingDisks

נצליב את היחס (רץ על) DQ עם עצמו ונחפש שורות בהן יש את אותה שאילתה אבל דיסק שונה, אזי דיסקים המריצים את אותה שאילתה ולפיכך הם דיסקים מתנגשים (conflicting disk) לפי הגדרה, נוסיפם לרשימת מזהי הדיסקים המוחזרים.

mostAvailableDisks

ראשית ניצור תת שאילתה שתחושב בהמשך לכל שורה בטבלה שתחושב (יוסבר בהמשך). בתת שאילתה נצליב את המקום הפנוי בדיסק ספציפי עם כל השאילתות ונסנן את השורות בהן גודל השאילתה קטן/שווה לגודל המקום הפנוי, ועל טבלה זו נפעיל מניה לעמודת מזהי השאילתות השונות. כלומר זו תת שאילתה שמחשבת כמה שאילתות יכולות לרוץ על דיסק כלשהוא. נסמנה T. כעת מתוך כל הדיסקים נבנה טבלה מעמודות המזהה, המהירות ושדה נוסף שהוא חישוב T לכל מזהה דיסק. נמיין לפי הנדרש, נקצץ עליונים ולבסוף נחזיר רק את מזהי הדיסקים.

getCloseQueries

עבור על query במערכת השונה מה-query המבוקש, אנו בודקים מה מספר הדיסקים המשותפים להם ואוספים זאת לטבלה זמנית, זאת על ידי COUNT על DQ היכן שמזהה המבוקש נמצא גם בטבלה זמנית שבנינו ומייצגת את כל הדיסקים עליהם ה-query הנתון רץ. מתת השאילתה האחרונה אנו מקבלים טבלה זמנית, ממיינים את שורותיה לפי מספר הדיסקים המשותף הגבוהה ביותר ומחזירים את עשרת מזהי ה-queries עם הערך הגבוהה ביותר.

חומרים נוספים

תיאור הטבלאות על בסיס אופן יצירתם (ניתן לשים לב לקשרי המפתחות ואילוצי כל השדות)

```
CREATE TABLE TQuery(queryID INTEGER PRIMARY KEY NOT NULL UNIQUE CHECK(queryID > 0),\
                    purpose TEXT NOT NULL,\
                    size INTEGER NOT NULL CHECK(size >= 0));\
\
CREATE TABLE TRAM(ramID INTEGER PRIMARY KEY NOT NULL UNIQUE CHECK(ramID > 0),\
                  company TEXT NOT NULL,\
                  size INTEGER NOT NULL CHECK(size > 0));\
\
CREATE TABLE TDisk(diskID INTEGER PRIMARY KEY NOT NULL UNIQUE CHECK(diskID > 0),\
                   company TEXT NOT NULL,\
                   speed INTEGER NOT NULL CHECK(speed > 0),\
                   free_space INTEGER NOT NULL CHECK(free_space >= 0),\
                   cost INTEGER NOT NULL CHECK(cost > 0));\
\
CREATE TABLE DR(diskID INTEGER NOT NULL CHECK(diskID > 0),\
                ramID INTEGER NOT NULL CHECK(ramID > 0),\
                FOREIGN KEY (diskID) REFERENCES TDisk(diskID) ON DELETE CASCADE,\
                FOREIGN KEY (ramID) REFERENCES TRAM(ramID) ON DELETE CASCADE,\
                PRIMARY KEY (diskID, ramID), UNIQUE(diskID, ramID));\
\
CREATE TABLE DQ(diskID INTEGER NOT NULL CHECK(diskID > 0),\
                queryID INTEGER NOT NULL CHECK(queryID > 0),\
                FOREIGN KEY (diskID) REFERENCES TDisk(diskID) ON DELETE CASCADE,\
                FOREIGN KEY (queryID) REFERENCES TQuery(queryID) ON DELETE CASCADE,\
                PRIMARY KEY (diskID, queryID), UNIQUE(diskID, queryID));
```