בס"ד

תמר ברכר: [tamar.brecher5775@gmail.com](mailto:tamar.brecher5775@gmail.com)

צפורה שינדלהיים: [tzipora.shindelhaim@gmail.com](mailto:tzipora.shindelhaim@gmail.com)

**Design --- ---**

**תיאור המערכת:**

הדטה ביס לא רלציוני – בעל סכמה משתנה, ומונחה לפי שורות.

הדטה בייס יהיה שמור בקבצי shelve בצורה של .key-value כל collection ישמר בקובץ shelve נפרד, כאשר המפתח בקובץ shelve יהיה ה primary key של הcollection והערך של המפתח יהיה דיקט שיבטא את הרשומה ב collection, כל דיקט של רשומה יכיל כמפתח את שם העמודה וכערך את הערך המתאים בהתאם לאותה רשומה.

מהגדרת הטבלה נקבל שלכל טבלה קיים אינדקס שהוא המפתח בטבלה.

העמודה בטבלה תוכל להכיל את כל סוגי הטיפוסים או טיפוס אחד בלבד - נתון לבחירתו של המשתמש, במקרה שהמשתמש יחליט על טיפוס מסוים לעמודה המערכת תטפל בזה.

**הפיצ'רים של המערכת:**

השימוש במערכת הוא ע"י יצירת אובייקט דטה בייס שגורם לטעינת collections קיימים בזיכרון על מנת למנוע אבוד מידע לאחר סגירת התוכנית.

* קבלת מידע על הדטה ביס – מס' טבלאות קיימות, שמות הטבלאות הקיימות.
* יצירה/מחיקה של טבלה.
* קבלת מידע על טבלה – מס' שורות בטבלה, שמות העמודות בטבלה.
* הכנסה/מחיקה/עדכון של רשומה מטבלה.
* מחיקה של מס' רשומות ע"פ תנאים מסוימים (שימוש באינדקס אם קיים)ץ
* ביצוע שאילתות על טבלה (שימוש באינדקס אם קיים).
* יצירת אינדקס על עמודה מסוימת עבור טבלה.

בס"ד

**יתרונות:**

* מידע חסוי – קובץ shelve הוא קובץ בינארי שמצפין את המידע ששמור בו.
* נפח המידע קטן – קובץ בינארי תופס מעט מקום בזיכרון.
* גישה מהירה לנתונים – השימוש בקובץ shelve הוא כמילון (hash- table)
* גמישות הcollection - הdatabase לא רלציוני ולכן נותן גמישות לשדות ולטיפוסים שלהן בכל שורה.
* שימוש לטווח ארוך – בגלל הגמישות והנפח הקטן.

**חסרונות:**

* טעינת הקובץ איטית.

**4 שיקולים עיקריים:**

* **Performance - ביצועים**

שליפת הנתונים פר שורה היא מהירה וכן הוספת שורה, עדכון, מחיקת ערכים בשורה.

* **Space Utilization – ניצול שטח**

קובץ shelve הוא קובץ בינארי וממילא טופס פחות מקום בזיכרון.

* **Complexity of DB - סיבוכיות**

עיין בעלות הפעולות.

* **Flexibility - גמישות**

כיוון שהדטה ביס לא רלציוני הוא נותן גמישות לשדות ולטיפוסים שלהן בכל שורה.

**עלות הפעולות CRUD על טבלה** (כאשר n מציין מס' שורות בטבלה וm מס' העמודות בטבלה)**:**

* **Create –** יצירת שורה עולה O(m)(עדכון כל האינדקסים הקיימים לטבלה)
* **Read –** קריאת שורה ע"פ מפתח / ערך שיש לו אינדקס עולה O(1)

ולכל ערך אחר עולה O(n)

* **Update –** עדכון שורה ע"פ מפתח/ ערך שיש לו אינדקס עולה O(m)

ולפי ערך אחר עולה O(n + m)

(m עבור עדכון כל האינדקסים הקיימים לטבלה)

עדכון ערך ע"פ מפתח/ ערך שיש לו אינדקס עולה O(1)

ולפי ערך אחר עולה O(n)

* **Delete –** מחיקת שורה מטבלה ע"פ מפתח עולה O(m))(עדכון כל האינדקסים הקיימים לטבלה)

בס"ד

**תיאור המימוש:**

* **הגדרת טבלה**

על מנת ליצור טבלה ניצור אוביקט של טבלה ונכנסו לרשימת הטבלאות בדטה בייס

וכן ניצור קובץ shelve חדש (בשם הטבלה)

* **מחיקת טבלה**

מחיקת קובץ הטבלה מהזיכרון וקבצי האינדקסים מהזיכרון

ושחרור אובייקט הטבלה מהזיכרון ומחיקתו מרשימת הטבלאות.

* **הכנסת רשומה לטבלה (ברשומה יכולים להיות מחרוזות, מספרים ותאריכים)**

הוספת הרשומה לקובץ כאשר הkey יהיה המפתח של הטבלה והvalue יהיה הרשומה בצורה של מילון כפי שנשלחה וכן עדכון האינדקסים.

* **מחיקת רשומה מטבלה**
* לפי מפתח: גישה ישירה לרשומה באמצעות המפתח ומחיקתה מהטבלה ומכל האינדקסים.
* לפי ערך שיש לו אינדקס: גישה לאינדקס לקבלת המפתח לרשומה המתאימה, גישה לטבלה באמצעות המפתח למחיקת הרשומה מהטבלה ומחיקת המפתח מכל האינדקסים.
* לפי ערך (ללא אינדקס) : חיפוש הערך ע"י מעבר על כל הרשומות (המפתחות) בקובץ ומחיקת הרשומה אליו הוא שייך, וכן מחיקת המפתח מכל האינדקסים.
* **מחיקת מספר רשומות לפי תנאי/ים מסוימים מטבלה**
* במידה ואחד התנאים הוא שהמפתח שווה לערך כלשהו גישה לערך בטבלה ובדיקת התנאים על שאר העמודות ובמידה וכל התנאים מתקיימים נמחק את הרשומה אליה הוא שייך.
* במידה ואחד התנאים הוא שהשדה שיש לו אינדקס שווה לערך כלשהו גישה לערך באינדקס לקבלת המפתח בטבלה, גישה לטבלה באמצעותו ובדיקת התנאים על שאר העמודות ובמידה וכל התנאים מתקיימים נמחק אותה.
* אחרת נעבור על כל הרשומות בטבלה וכל רשומה שעומדת בתנאי או שהערכים שלה עומדים בתנאי נמחק אותה – בהתאם למחיקת רשומה מהטבלה.
* **עדכון רשומה בטבלה**
* לפי מפתח: גישה ישירה לרשומה באמצעות המפתח ועדכון הרשומה וכל האינדקסים.
* לפי ערך שיש לו אינדקס: גישה לאינדקס לקבלת המפתח לרשומה המתאימה, גישה לטבלה באמצעות המפתח לעדכון הרשומה בטבלה ועדכון כל האינדקסים.
* לפי ערך (ללא אינדקס) : חיפוש הערך ע"י מעבר על כל הרשומות (המפתחות) בקובץ , עדכון הרשומה וכן עדכון האינדקסים.
* **חיפוש רשומות העונות על תנאי/ים מסוימים בטבלה**
* במידה ואחד התנאים הוא שהמפתח שווה לערך כלשהו גישה לערך בטבלה ובדיקת התנאים על שאר העמודות ובמידה וכל התנאים מתקיימים נחזיר את הרשומה אליה הוא שייך.
* במידה ואחד התנאים הוא שהשדה שיש לו אינדקס שווה לערך כלשהו גישה לערך באינדקס לקבלת המפתח בטבלה ,גישה לטבלה באמצעותו ובדיקת התנאים על שאר העמודות ובמידה וכל התנאים מתקיימים נחזיר אותה.
* אחרת נעבור על כל הרשומות בטבלה וכל רשומה שהערכים (העמודות) שלה עומדים בתנאים נחזיר אותה.

בס"ד

**אובייקטים קיימים:**

* DataBase
* DBTable
* DBField
* SelectionCriteria

**אתגרים במהלך הפרויקט:**

* מימוש דטה ביס לא רלציוני כאשר ההכרות היתה רק עם דטה ביס רלציוני**.**
* יצירת אינדקס לאחר מימוש ללא האינדקס גרמה לשינויים רבים בקוד.

**הצעדים להמשך:**

* מימוש join בין טבלאות.
* שמירת כל טבלה במס' קבצים על מנת להקל על טעינת הקבצים.
* יעילות תחזוקת האינדקס.