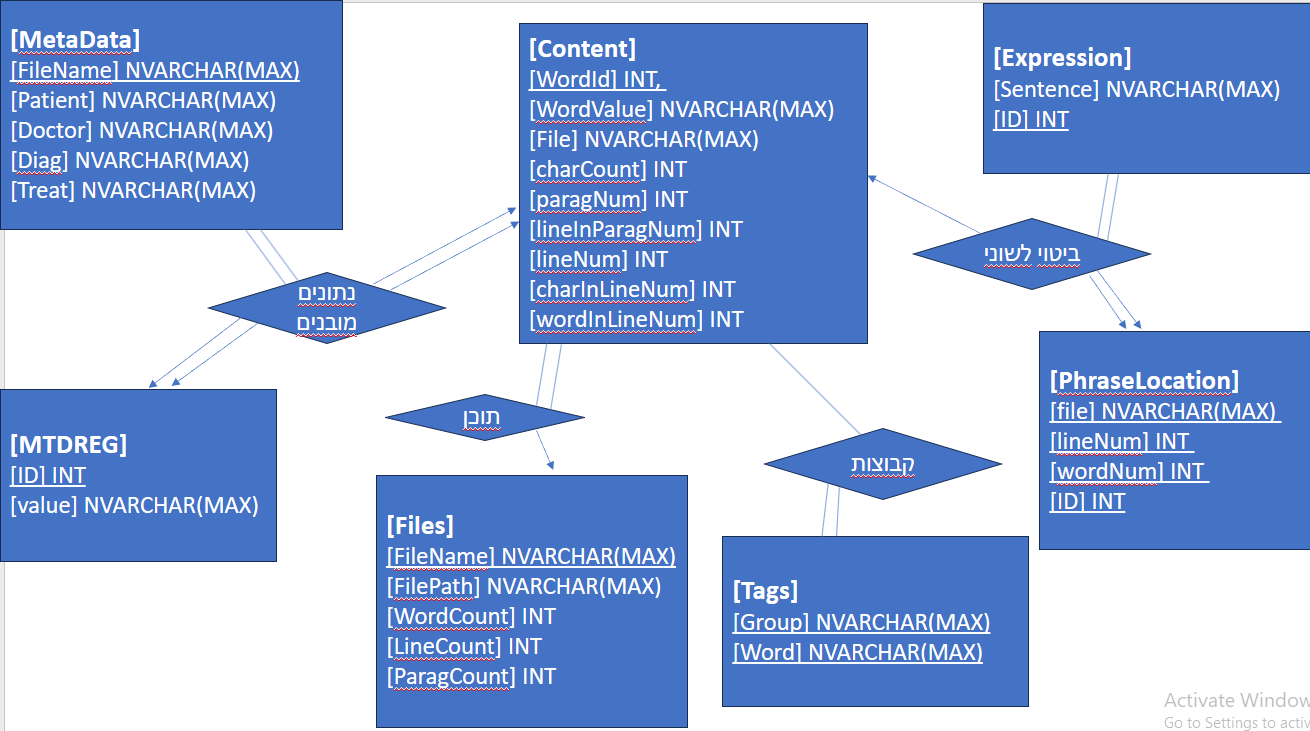
סדנא בבסיסי נתונים - 20563 סמסטר 2023ב   
מגישה : מיכל נפתולינה   
 מנחה: הילה קליף

**איפיון פרוייקט**

הפרוייקט מדמה בסיס נתונים של מרפאה, ומחזיק מסמכי תיקים רפואיים של מטופלים, כל קובץ טקסט יחזיק סיכום ביקור של מטופל ויכיל 5 חלקים : 1 פרטי הביקור , 2 פרטי הרופא , 3 פרטי החולה, 4 פרטי האבחנה , 5 פיסקת סיכום חופשית של הרופא ו 6 פרטי הטיפול שהרופא נותן

כל פסקה מתחילה במילות פתיחה שיזהו אותה , כל פסקה למעט זאת עם הסיכום מכילה נתונים מובנים על התוכן, כמו שם הרופא וכדומה, שישמשו לסינון בהמשך



הטבלאות :

1. טבלת קבצים [**FILES**] במכילה רשומה לכל קובץ ובה כל הנתונים המובנים על הקובץ: שמו, מיקומו וסטיסטיקות של הקובץ: מספר מילים , מספר שורות ומספר פסקאות
2. טבלה נתונים מובנים [**METADATA**] שמכילה רשומה לכל קובץ ממנו ניקרא המידע ובה יש את שם הקובץ המשמש כמפתח ומיספרי זיהוי של 4 נתונים מובנים: הרופא, המטופל , האבחנה והטיפול (INT) שאפשר לסנן איתן ,בהצבה עם טבלת הרישום של הנתונים המובנים
   1. טבלת רישום הנתונים המובנים [**MTDREG**] שמכילה ערך שניקרא מהקובת וזוהה כנתון מובנה עם מזהה יחודי שנוצר עבורו

כך אותו נתון מובנה יכול להופיע בטבלת הרישום פעם אחת בלבד והמזהה שלו ישובץ בטבלת הנתונים בכל מקום בו הוא מופיע וכמספר זוגים שונים (למשל בקובץ אחד שם יופיע כרופא ובקובץ האחר אותו שם יופיע כמטופל אך לשם זה מזהה אחיד)

1. טבלת מילים [**CONTENT**] המכילה את כל המילים מהקבצים כרשומה והנתונים עליה : המזהה היחודי של המילה כי יש חזרות, המילה עצמה , הקובץ שהיא מופיעה בו, המיקומים שלה לפי שני האינדקסים , סטיסטיקות על המילה כמו מספר תווים, מספר המילה בשורה
2. טבלת קבוצות [**TAGS**] שמכילה זוגות של מילה והתג שלה שהוא מזהה יחודי של קבוצה בה נימצא המילה, יכולים להיות מספר מילים באותה קבוצה אז יש כמה רשומות בהם מופיע אותו תג , ויכולים להיות מילים שנימצאים באותה קבוצה אז יש כמה רשומות בהם מופיע אותה מילה עם תגים שונים
3. טבלת ביטויים [**EXPRESSION**] שמכילה ביטוי לשוני ומספר ID לזיהוי
   1. טבלת מיקומי ביטויים [**PhraseLocation**] שמכילה את המזהה היחודי של הביטוי עם שם קובץ והאינדקס של המילה הראשונה בביטוי כנקות ההתחלה של הביטוי   
      יכולים להיות מספר חזרות של אותו מזהה ביטוי שמופיע עם הרבה מיקומים שונים , לרבות באותו קובץ

קשרי היישויות:

* קבוצות : לכל קבוצה יכולים להיות הרבה מילים , ומילה יכולה להיות בכמה קבוצות   
  לא כל המילים חייבות להימצא בקבוצה, לקבוצה חייב להיות לפחות מילה אחת
* ביטוים - לא כל מילה בטקסט היא תחילת ביטוי, ואם כן אז היא תחילתו של ביטוי יחיד שמתאים לה   
  יכולים להיות הרבה מופעים של הביטוי בטקסטים, ביטוי במאגר חייב להופיע בטקסטים,
* תוכן - לכל מילה חייב להיות קובץ מקור והוא יחיד , יכולים להיות הרבה מילים מאותו קובץ
* נתונים מובנים - לכל נתון מובנה יש בהכרח מזהה יחודי והוא יחיד,   
  לכל קובץ בהכרח יש רשומת מיפוי נתונים מבנים והיא יחידה,   
  אותו מזהה יחודי של נתון מובנה יכול להופיע בהרבה מקומות שונים במיפוי , וחייב להופיע

**התוכנית :**

* התוכנית רצה על שרת MICROSOFT SQL מקומי
* לקוד יש 2 משתנים גלובליים
  + string serverName - שמשמש לזיהוי השרת המקומי עליו התוכנית תתחבר לביצוע השאילתות בSQL
  + string connectionString - שמשמש להתחברות לשרת ומכיל את שם השרת , הפורט 1433 , ופרמטרים לאבטחה שבעבודה זו מצומצמים

**הפונקציות**

**static void Main(string[] args)**

פונקצייה ראשית שמשמשת כממשק משתמש command line לקבלת קלט , תפעול של כל פונקציות הביצוע ונותנת את הפלט בחזרה

הפונקצייה מאפשרת בדיקת תקינות קלט מינימלי , ותדפיס הודעה מתאימה בהינתן קלט ריק או קלט פקודה שגוי , כמוכן פרמטרים ריקים אך שגיאות בפרמטרים יתקבלו ויבדקו מול השרת

הפקודות שהפונקצייה מקבלת :

יודפס המילה welcom ומה השרת אליו מחוברים, ולאחר מכן ניכנסים ללולאה שבה המתשמש יכול להכניס פקודות ככל שירצה   
בשלב ראשון יתבקש המשתמש להכניס: [start, load, stats, search, view, data mine, exit ]

Start, load, stats, view, data mine, exit הם פקודות עצמאיות:

**Start** - המשתמש מבקש לפתוח בסיס נתונים חדש ,

1. נבקש ממנו שם חדש לDB
2. אם הקלט אינו ריק נבדוק שהשם אינו תפוס על ידי בסיס נתונים קיים על ידי פונקצייה DatabaseExists(DBNAME)
3. אם כן נדפיס בהתאם ואם לא אז נפנה לפונקצייה Setup(DBNAME)

**Load** - המשתמש מבקש לעלות קבצים לבסיס הנתונים

1. נשאל אותו איזה DB ומה שם התקייה בה נימצאים הקבצים
2. אם הקלט אינו ריק ובסיס הנתונים קיים לפי פונקצייה DatabaseExists(DBNAME)
3. נפנה לפוקצייה loadFiles(DBNAME, DirPath)

**Stats** - המשתמש מבקש לראות סטטיסטיקות של המילים בקבצים השונים , נפנה לפונקצייה stats() ובו יש ממשק נוסף

**View** - המשתמש מבקש לראות את ההקשר של מילה מסויימת, ויקבל כפלט את 3 השורות הקודומות למילה ואת 3 השורות הבאות אחרי המילה ובכך יראה את הטקסט בה מוכנסת המילה

1. נישאל את המשתמש מה המילה שהוא רוצה לראות
2. אם הקלט אינו ריק נפנה לפוקציה view(word)

**Data mine** - המשתמש מבקש לראות מסקנות מהנתונים המובנים על ידי שימוש בכרית מידע , בפרוייקט זה בחרתי להציג assosiation rules על ידי אלגוריתם אפריורי , נפנה לפונקצייה ExecuteAprioriOnMetaData() שמבצעת זאת

**Exit** זה פקודה שתסיים את הלולאה ותוציא את המשתמש מהממשק

Search, view הם פקודות שיתשאלו עוד את המשמתמש עד להצגת התוצאה הרצוייה לו

**search** ישאל המשתמש מה ירצה לחפש [file, word, group, expression]

**File** - חיפוש קובץ , ישאל המשתמש לפי מה ירצה לחפש [word, metadata]

עבור החיפוש אנחנו מגדירים רשימת מחרוזות printlist, פונקציית החיפוש המתאימה תחזיר רשימת ערכים שהמשתמש ביקש ואותם נציב במשתנה זה , ואז משתנה זה נישלח לפונקציית הדפסה print(printlist) משותפת

**Word** - חיפוש לפי ערך המילה , נבקש מהמשתמש את המילה שהוא רוצה ואם הקלט לא ריק נפנה לפונקצייה FindFileByWord(word) , שיחזיר רשימת מחרוזות של הקבצים שבהם מילה זו מופיעה , ואז נדפיס על ידי print(printlist)

**Metadata**- חיפוש לפי ערך נתון מובנה,

נבקש מהמשתמש את הנתון שהוא רוצה ואת סוג הנתון שהוא מסנן ,

[any=0,Patient=1,Doctor=2,Diagnosis=3,Treatment=4] , הנתון יתקבל כמספר ואם המשתמש לא מעוניין לסנן אז יתקבל 0 לחיפוש בכל הסוגים

אם הקלט לא ריק נפנה לפונקצייה FindFileByMTD(MTDNum, word) , שיחזיר רשימת מחרוזות של הקבצים שבהם מילה זו מופיעה כנתון מובנה המבוקש , ואז נדפיס על ידי print(printlist)

**Word** - חיפוש מילים , נפנה לפונקצייה ListWords() , שהיא פונקצייה שמדפיסה את כל המילים הנימצאות בטבלת המילים ובכך נותנת למשתמש יכולת הצגה של כל המילים (פונקציונליות 1), בפונקצייה זו יש ממשק נוסף שמאפשר למשתמש לסנן את כל המילים לפי קובץ ולפי אינדקס לבחירתו מהשניים (פונקציונליות 2),   
פונקצייה זו מחזירה רשימת כל המילים (כולם או לפי הסינון) ואותם נדפיס עם print(printlist)

**Group** - ישאל המשתמש מה ירצה לעשות בנוגע לקבוצות [add, list ]

**Add** - ליצור קבוצה חדשה ולהוסיף מילים קיימות לתוכה

* + 1. מבקשים שיכניס כקלט את שם הקבוצה החדשה
    2. יכניס כקלט מילים מופרדות ברווח ומקבלים מחרוזת ארוכה שאם לא ריקה מומרת לרשימת מחרוזות על ידי userInput.Split(new[] { ' ' })
    3. מפנים לפונקצייה GroupCreate(wordList, groupName) ליצירת הקבוצה

**List** - להדפיס את כל המילים הקיימות בקבוצה שהמשתמש נותן

נבקש מהמשתמש את שם הקבוצה שהוא רוצה ואם הקלט לא ריק נפנה לפונקצייה GroupSearch(groupName) , שיחזיר רשימת מחרוזות של מילים בקבוצה המבוקשת, ואז נדפיס על ידי print(printlist)

**Expression** - נבקש מהמשתמש את הביטוי שהוא מעוניין למצוא , אם הקלט לא ריק נפנה לפונקצייה returnExpression(exprs) שתחזיר רשימה של אינדקסים המסמנים את תחילתו של המצאות הביטוי המבוקש בטקסט כולל הקובץ ,

הפונקצייה הזו תחזיר אינדקסים מרישום מיקומים קיים של איחסון נתונים אלו וגם תטפל בהוספת מיקומים מקבצים חדשים או מביטוי חדש שמהמשתמש רוצה לשמור

**static void print(List<string> printlist)**

פונקציית עזר המשמשת להדפסת רשימה של מחרוזות

אם הרשימה שהתקבלה לא ריקה ולא null אז עבור על כל איבר ברשימה והדפס אותו

**static bool DatabaseExists(string dbName)**

פונקציית עזר המשמשת לבדיקת קיומו של בסיס נתונים בשם הנתון

הפונקצייה מתחברת לשרת בעזרת הconnection string ומוציאה שאילתא מטבלת ה sys.databases לספור כמה יש רשומות של בסיסי נתונים עם השם המבוקש ואם חוזר 0 אז משמע לא קיים

**static void Setup(string databaseName)**

פונקצייה שמאתחלת בסיס נתונים חדש   
הפונקצייה מתחברת לשרת ומוציאה שאילתא של יצירת בסיס נתונים ואז עוד 7 שאילתות ליצירת כל אחד מהטבלאות שבשימוש בקוד זה, הטבלאות הן כפי המפורט למעלה .  
בכל יצירה של טבלה יוצאת הודעה בהתאם

**static void loadFiles(string databaseName, string directoryPath)**

פונקציית טעינת מסמכי טקטס והכנסת נתונים מובנים

הקלט הוא directory path ממנו ילקחו מסמכי ה\*.txt ושם בסיס הנתונים לשם מזינים את המידע

הפונקצייה סורקת את הקבצים בתקייה הנתונה, ומכניסה אותם לרשימת קבצים , לכל קובץ ברשימה זו מתקיים הדברים הבאים:

* בודקים אם קיים עבורו רשומה בטבלת הקבצים על ידי שאילתא מטבלת Files לספור כמה יש רשומות ואם חוזר 0 אז משמע לא קיים וניתן לטעון או שחוזר 1 ואז לדלג
* במידה וטוענים את הקובץ , מחלצים ממנו את פרטי הקובץ על ידי פנייה לפונקצייה  
   CountFileStats(filePath, out paragraphCount, out lineCount, out wordCount)   
  לקבלת סטיסטיקות של כמויות מילים/ משפטים/ פסקאות   
  נתונים אלו מוכנסים לטבלה על ידי שאילתא שיוצרת רשומה חדשה לקובץ
* יוצאת שאילתא ליצירת רשומה לקובץ בטבלת METADATA הנתונים המובנים המכילה את שם הקובץ ומזהה 0 לכל אחד מ4 העמודות של סוגי הנתונים והם יושלמו אחר כך
* מפנים לפונקצייה wordInsert(filePath) שתפקידה לקרוא את הקובץ ולהוסיף את המילים עצמם ולעדכן את הנתונים המובנים

לאחר המעבר על כל הקבצים החדשים ועדכון כל פרטי המילים החדשים , יש לטפל במקרה מסויים של ביטויים לשוניים קיימים בטבלת הביטויים שעבורם ישנם מיקומים של ביטוי זה בטבלת המילים לפני העדכון ויש להוסיף את מיקומי הביטויים בטבלת המילים מתול המילים החדשות שהוספו כרגע ולכן נפנה לפונקצייה updateExpression() כדי לשמור על מאגר הביטויים מעודכן, שכן המאגר מתעדכן לפי בקשת המשתמש רק כאשר הוא מכניס ביטוי חדש, לכן ביטוי קיים יטופל ללא ידיעת המשתמש .

**static void CountFileStats(string filePath, out int paragraphCount, out int lineCount, out int wordCount)**

פונקציית עזר שמקבלת קובץ , קוראת אותו ומפרקת אותו 3 פעמים עם פונקציית Split על ידי רווח למילים, שורה חדשה למשפטים ורווח גדול לפסקאות , ומחזירה את מספר האיברים בכל פיצול כמספר המילים/ שורות / פסקאות בקובץ

**public static void wordInsert(string filePath)**

הפונקצייה שקוראת ומכניסה את המילים עצמם לבסיס הנתונים

מתחילים מלאתחל משתנים מונים המאותחלים ל0 שתפקידם לספור את המילה/ משפט/ פיסקה הנוכחי של המילה שקוראים:

ParagNum- מספר הפסקה , lineNum - מספר השורה בקובץ , lineInParagNum - מספר השורה בפסקה , charInLineNum - מספר התו בשורה , wordInLineNum - מספר המילה בשורה , charInWordNum - מספר התו במילה

מתחילים לקרוא שורה שורה, וסופרים בהתאם את הפסקה והשורה , כל שורה מפרקים לרשימת מילים המופרדות ברווח , וכל מילה סופרים בהתאם את התווים ומקומה בשורה, ואז מכניסים את כל הפרטים של המשתנים הצוברים עד כה על ידי פנייה לפונקצייה InsertWordData(word, filePath, charInWordNum, paragNum, lineInParagNum, lineNum, charInLineNum, wordInLineNum);

ואם המילה שהתקבלה היא מילת מפתח שמבציעה על נתונים מובנים : patient, doctor, diagnosis, treatment אז המילה הבאה היא ערך הנתון אותו נירצה לשמור ,   
ישנו דגל mtd ומשתנה שומר mtd\_type , אם קוראים מילת מפתח אז מדליקים את הדגל ושומרים את מילת המפתח

באיטרצייה הבא של הקריאה , כאשר רואים שהדגל נידלק אז שולחים את המילה הנוכחית והמילה השומרה ב mtd\_type לפונקציה MTDHandler(word, mtd\_type, filePath) לרישום ברשומת הקובץ בטבלת הMETADATA והMTDREG ,

בשביל שזה יקרה באיטרצייה הבאה של הלולאה בדיקת הדגל נעשת לפני הדלקתו , הפונקצייה מניחה שהקובץ נכון ואין נתון מובנה על המילה הראשונה ללא התראת מילת המפתח מילה לפני

**private static void InsertWordData(string word, string filePath, int charInWordNum, int paragNum, int lineInParagNum, int lineNum, int charInLineNum, int wordInLineNum)**

פונקציית עזר שמקבלת מילה וכל הנתונים שלה , יוצרת עבורה מזהה יחודי, מתחברת לשרת ומוציאה שאילתא ליצור רשומה חדשה עבור מילה זו בטבלת הcontent

**public static void MTDHandler(string word, string type, string fileName)**

פונקציית עזר שמקבלת מילה שהיא ערך של נתון מובנה , הסוג של הנתון המובנה והקובץ ממנו ניקרא הנתון המובנה ,

מחפשת את מספר הזיהוי של הנתון המובנה על ידי הערך בעזרת פניה לפונקצייה GetOrCreateEntryId(word)

ומכניסה לטבלת הMETADATA בעזרת שאילתא את המזהה של הנתון המובנה לרשומה של הקובץ הנתון תחת העמודה של הסוג הנתון

**public static int GetOrCreateEntryId(string word)**

פונקציית עזר שמקבלת ערך של נתון מובנה ומחפשת אותו בטבלת רישום הנתנוים המובנים MTDREG  
אם נימצא מחזיר את המזהה היחודי שלו ואם לא , יוצר מזהה יחודי חדש , מכניס ערך זה לטבלה עם המזהה שנוצר ומחזיר את המזהה החדש

ואז להתחיל לקרוא את התוכן למילוי שאר עמודות הרשומה בטבלת הנתונים וטבלת המילים:

לכל מילה ומילה (תוים בהפרדה של רווח) הניקראת מהקובץ נוצרת רשומה ומוסיפים את את ערכי העמודות למילה, כמו מאיזה קובץ נוספה המילה, האינדקסים שלה וסטיסטיקות כמו מספר התווים בה

אם המילה הנקראת עונה לתנאי מסויים שמראה על פתיחה של פסקה: "ביקור:" "המוטפל:" "הרופא:" "אבחנה:" "טיפול:" או כדומה, להדליק דגל שהמילה הבאה יש לאחסן כנתון מובנה המתאים לדגל , למשל בקריאה של "הרופא:" יודלק דגל למילוי עמודת "הרופא" לשורה זו

**static void stats()**

פונקצייה שפותחת ממשק קטן מול המשתמש להצגת נתונים סטיסטיים של הקבצים הנמצאים במאגר בסיס הנתונים

המשתמש ניכנס ללולאה אין סופית בה יכול להמשיך לבקש לראות פרטים עד שיוצא מהמשק הפנימי לממשק החיצוני של הפקודות

פונקצייה ראשית: לבחור קובץ או לצאת

במידה ובוחר לראות קובץ , ישאל לשמו של הקובץ , יוצאת שאילתא שמחזירה את מספר הרשומות בטבלת הקבצים בהם יש את שם הקובץ :

אם חוזר 0 משמע לא קיים ויוצאת הודעה בהתאם

אם חוזר לא 0 משמע יש קובץ ויוצאת שאילתא לטבלת הקבצים להוציא את פרטי הקובץ שנימצאים ברשומה של שם הקובץ, מחזירים את מספר הפסקאות / שורות / מילים ומדפיסים אותם

לאחר מכן נישאר אם המשתמש רוצה להמשיך עם הקובץ [Y/N] אם כן יופנה לפונקצייה drill(fileName) ואם לא חוזר ללולאה לבחור עוד קובץ או לצאת

**static void drill(string fileName)**

פונקצייה שנייה: לבחור לראות פסקה מסויימת או שורה מסוימת או לחזור אחורה

* במידה ובוחר לראות פסקה, ישאל מה מספר הפסקה הרצוייה ויופנה לפונקצייה statsParagraphs(fileName, parag)
* במידה ובוחר לראות שורה, ישאל מה מספר השורה הרצוייה ויפנה לפונקצייה statsline(fileName, line)   
  ואז ישאל אם ירצה לבחור מילה מסויימת מתוך שורה זו שבחר [Y/N] אם כן ישאל מה מספר המילה הרצוייה בשורה , ויוצאת שאילתא לטבלת המילים להוציא את פרטי המילה שנימצאים ברשומה של שם הקובץ, מספר השורה ומספר המילה , מחזירים את ערך המילה וכמה תווים יש בה ומדפיסים אותם

**static void statsParagraphs(string fileName, int num)**

יוצאת שאילתא לטבלת המילים להוציא את פרטי המילה שנימצאים ברשומה של שם הקובץ, ומספר הפיסקה, מחזירים את מספר השורות שבה ומדפיסים אותו

SELECT MAX(lineInParagNum) AS lineCount FROM Content WHERE paragNum = @num AND File = @fileName

ואז יוצאת שאילתא שסוכמת את ערכי מספר המילה הגדול ביותר של כל שורה שהיא רשומה כשורה בפסיקה הנבחרת כדי לקבל את מספר המילים בשורה ומדפיסים אותו

SELECT SUM(MaxWordInLineNum) AS wordCount" +

" FROM (SELECT MAX(wordInLineNum) AS MaxWordInLineNum" +

" FROM Content WHERE paragNum = @num AND File = @fileName GROUP BY lineNum" +

" ) AS Subquery

**static void statsline(string fileName, int num)**

יוצאת שאילתא לטבלת המילים שמוציאה את ערך מספר המילה בשורה הגדול ביותר בשורה זו בקובץ זה כדי לקבל את מספר המילים בשורה ומדפיסים אותו

SELECT MAX(wordInLineNum) AS LineCount FROM Content WHERE lineNum = @num AND File = @fileName

**static List<string> FindFileByWord(string word)**

פונקצייה שמקבלת מילה ומחזירה רשימה של כל הקבצים שבהם יש מילה זו

על ידי מעבר על כל רשומה בה ערך המילה מתאים לקלט, ערך שם הקובץ אם לא מופיע כבר ברשימה יתווסף ואז הרשימה תוחזר

על ידי שאילתא SELECT File FROM Content WHERE WordValue = @word

**static List<string> FindFileByMTD(int number, string word)**

פונקצייה מקבלת מילה וסוג הנתון המובנה הרצוי , ומחזירה רשימה של כל הקבצים בהם יש מילה זו כנתון מובנה מבוקש

שלב ראשון: לבדוק את קיומו של נתון מובנה זה בטבלת הרישום של הנתונים המובנים MTDREG , ושליפת מספר הזיהוי שלו, אם לא נימצא יוצאת הודעה בהתאם

שלב שני: מעבר על טבלת הנתונים המובנים , בכל רשומה בה בעמודה המתאימה לסוג הנתון בקלט (או במקרה שבחרו any metadata type אז בכל 4 העמודות) קיים ערך המזהה המתאים למילה בלט , ערך שם הקובץ אם לא מופיע כבר ברשימה יתווסף ואז הרשימה תוחזר

SELECT FileName FROM MetaData WHERE {"Patient"} = @entryId OR {"Doctor"} = @entryId OR {"Diag"} = @entryId OR {"Treat"} = @entryId

**static List<string> ListWords()**

פונקצייה שמחזירה את כל המילים המתאימיות לסינון מסויים כרשימת מחרוזות

האינדקסים הם עמודות בטבלת המילים: אינדקס ראשון - שורה ועמודה(תו בשורה) בכלל הקובץ fileLine & char ואידקס שני - פסקה ושורה בפסקה part & partLine.

המילים נימצאות על ידי שאילתא שניבנת בהדרגה במהלך הפונקצייה על ידי תשאול של המשתמש   
נגדיר רשימת מחרוזות filters, שמתווספים אלייה חלקי השאילתא הנוספים לפי פילטרים שהמשתמש יבחר אם להוסיף או לא , ולבסוף נוסיף חלקים אלו לשאילתא

* ישנה שאילתת בסיס   
  SELECT WordValue FROM Content  
  אם אף פרמטר לא מתווסף , מפעילים אותה ומוציאים את כל המילים בטבלה
* שואלים אם ירצה לסנן לפי קובץ , במידה וכן נשאל לשם הקובץ ואת הקלט נוסיף ל filters את התוספת : File = {file}
* שואלים אם ירצה להוסיך אינדקס אחד מהשניים   
  המשתמש יחזיר כקלט 1 או 2 לבחירת סוג האינדקס ולפי זה ישאל לפרמטרים המתאימים אם אינם ריקים:  
   אם יבחר לפי אינדקס שורה נוסיף ל filters חלק lineNum = {x} AND charInLineNum = {y}  
  ואם יבחר לפי אינדקס פסקה נוסיף ל filters את paragNum = {a} AND lineInParagNum = {b}
* נבנה מחרוזת תוספת string addon = string.Join(" AND ", filters); ואם אינה ריקה אז נוסיף אותה לשאילתת הבסיס baseQuery += $"WHERE " + addon;
* נבצע את השאילתא ונחזיר את רשימת המחרוזות המתקבלת

**static void view(string givenWord)**

פונקצייה שמקבלת מהמשתמש מילה ומדפיסה את ההקשר שלה על ידי הדפסת 3 המשפטים שלפני המילה ו3 המשפטים שאחרי המילה

ראשית הפונקצייה צריכה לזהות איזה מילה להציג הקשר מכלל החזרות, לכן ישאל את המשתמש לשם הקובץ ואם זה אינו ריק יפנה לפונקצייה getLine(givenWord, file) לזהות את מספר השורה מתוך הקובץ לחזרות של המילה באותו קובץ, אם חוזר -1 משמע המילה לא נימצאת בקובץ זה

ואם כל הפרמטרים תקינים תופעל השאילתא למציאת כל המילים הנימצאות בטווח ויוכנסו לרשימת מחרוזות

SELECT WordValue FROM Content WHERE File = @file AND lineNum BETWEEN @minSentence AND @maxSentence ORDER BY lineNum, wordInLineNum

כאשר , יהי x מספר השורה הניבחרת :

int sentenceRange = 3;

int minSentence = Math.Max(1, x - sentenceRange);

int maxSentence = x + sentenceRange;

רשימת המחרוזות תודפס למשתמש כפלט

**static int getLine(string givenWord, string file)**

פונקציית עזר שמקבלת את המילה והקובץ בה היא נמצאת ומוציאה שאילתא לטבלת המילים שלכל רשומה בה ערך המילה ושם הקובץ מתאימים להוסיך לרשימה את מספר השורה של מילה זו

לאחר מכן מודפסת רשימת השורות הרלוונטיות בעזרת print(printlist) ומתוכן המשתמש בוחר שורה לראות את ההקשר שלה

המספר שמתקבל מהמשתמש בקלט מוחזר לפונקציית הview

**static void GroupCreate(List<string> wordList, string groupName)**

ניהול הקבוצות משתמש ב2 הטבלאות , הקבוצות TAGS והמילים COMTENT

הפונקצייה מקבלת רשימת מילים ושם קבוצה, יהיה לולאה לכל מילה ברשימת המילים יתבצע

* ניבדוק שקומבינציית המילה ותג הקבוצה לא קיים כבר יל ידי שאילתת ספירה של כמה רשומות עם ערכים אלו קיימות , אם יוחזר 0 משמע לא קיים וניתן להוסיף , אם קיים נדלג
* הוספת הרשומה לטבלת הקבוצות מתבצעת על ידי שאילתא לשרת עם הערכים מהקלט

כך בכל הכנסה ניתן ליצור קבוצה חדשה או לעדכן קבוצה קיימת

**static List<string> GroupSearch(string group)**

הפונקצייה מקבלת את שם הקבוצה עבורה רוצים לראות את כל המילים בה

הפונקצייה מגדירה שני רשימות wordsInGroup, wordslocations

הרשימה wordsInGroup תחזיק את רשימת כל המילים המתוייגות בתג הקבוצה הנתונה ויוחזרו משאילתת חיפוש על טבלת TAGS

רשימה זו תשלח לפונקציית עזר GetWordslocations(wordsInGroup) שתחזיר רשימה של כל המיקומי אינדקסים של כל המילים המופיעים ב wordsInGroup ותוצב ברשימה wordslocations אשר תוחזר למשתמש

**static List<string> GetWordslocations(List<string> words)**

פונקציית עזר שמקבלת רשימת מילים ומחזירה רשימה של מחרוזות המציינות מיקומים של מילים אלו בפורמט :   
{wordValue} | File: {File}, File Line: {lineNum}, Char Index: {charInLineNum}, Paragraph: {paragNum}, Part Line: {lineInParagNum}

לכל מילה ברשימת הקלט תופעל שאילתא על טבלת המילים CONTENT ומכל רשומה בה ערך המילה מתאים יוחזרו שני האינדקסים לפי השורה ולפי עמודה , יכנסו לרשימת הפלט לפי הפורמט ורשימת הפלט תוחזר

**static List<string> returnExpression(string expression)**

פונקצייה זו משמשת להחזרת ביטויים קיימים מהמאגר

מטבלת EXPRESSION יוצאת שאילתא שעבור ביטוי הקלט יחזיר את המספר המזהה היחודי שלו

אם יוחזר 0 משמע לא קיים ביטוי כזה ולכן לא יהיו מיקומים שלו במאגר ועל כן נפנה לפונקצייה newExpression(expression) למציאת הביטוי החדש

אם יוחזר מזהה , נחפש במאגר, כלומר טבלת PhraseLocation שיחיזר את האינדקסים בכל רשומה בה ערך מזהה הביטוי מתאים לזה שמצאנו   
כל ביטוי שימצא , האינדקסים שלו יווספו לרשימה שתוחזר למשתמש

**static List<string> newExpression(string expression)**

פונקצייה זו מחפשת בטבלת המילים, מילים עוקבות שיוצרות את הביטוי שהתקבל

ראשית נפצל את הביטוי שהתקבל כמחרוזת לרשימת מילים , ונישמור בצד את המילה הראשונה שתהי נקודת פתיחה לביטוי

נוציא שאילתא ולכל רשומה של מילים שערכן מתאים למילה הראשונה של הביטוי נחזיר את האינדקס של השורה והתו בשורה ואת שם הקובץ ונשלח מידע זה עם הביטוי לפונקצייה matchCheck(words, file, firstlineNum, firstWordNum) שתבדוק אם מילה זו והעוקבות לה אכן התפתחו לביטוי המבוקש

אם כן ישלחו אינדקסים אלו לפונקצייה addExpression(expression, file, firstlineNum, firstWordNum) שיתווסף המיקום שנימצא למאגר וגם יתווספו לרשימת אינדקסים שיוחזרו למשתמש

**static public void addExpression(string expression, string file, int firstlineNum, int firstWordNum)**

פונקצייה זו מפנה לפונקציית עזר exprsID(expression) לקבל מספר מזהה יחודי לביטוי , ובעזרת זה והמידע מהקלט על מיקום הביטוי בטבלת המילים מוציאה שאילתא לטבלת המיקומים PhraseLocation להוסיף רשומה חדשה של ביטוי למאגר

**static int exprsID(string expression)**

פונקצייה זו בודקת בטבלת הביטויים אם הביטוי שהתקבל קיים , אם השאילתא מחזירה 0 משמע לא קיים והפונקצייה יוצרת עבורו מספר מזהה חדש ומוציא שאילתא להוספת הביטוי עם המזהה שלו לטבלת הביטויים expression   
אם הביטוי קיים אז יוחזר המזהה הקיים שלו

**static public bool matchCheck(string[] expression, string file, int firstlineNum, int firstWordNum)**

פונקציית עזר שמקבלת ביטוי ואינדקסים להתחלה ועוקבת אחרי המילים הבאות אחרי האידקס מהקלט לראות שהם יוצרות את הביטוי שהתקבל , ומחזירה true/false

ניכנסים ללולאה כאורך מספר המילים ברשימת מילות הביטוי

לוקחים ומשווים את ערך המילה הi בביטוי לערך המילה שנישלפה מטבלת המילים content לפי האינדקסים , אם היא שווה ממשיכים הלאה , מעלים את I ב1 לקבלת המילה הבאה בביטוי ומעלים את האינדקסים למילה העוקבת בטבלה , ישנם שני אפשרויות להשיג את האינדקסים הבאים :  
יוצאת שאילתא לבדוק את המילה האחרונה בשורה הנוכחית ואם מספר המילה בנוכחית זהה למילה הנוכחית יודעים לעדכן לפי הערכים הנכונים

1. אם זה מילה באותה השורה (כלומר מספר השורה לא משתנה) ורק מספר המילה בשורה עולה ב1
2. אם זו הייתה מילה אחרונה בשורה וצריך את המילה הראשונה בשורה הבאה (כלומר רק מספר השורה עולה ב1 ומספר המילה בשורה הופך ל1)

לפי זה בודקים את המילה בטבלה ואם היא לא זהה למילה המצופה לפי הביטוי אז הביטוי נישבר וזהו אינדקס התחלת ביטוי שאינו מתאים ולכן יוחזר false

אם הגענו לסוף הלולאה ויצאנו, כלומר באף שלב לאורך מילות הביטוי לא נישבר הרצף אזי מצאנו מיקום של ביטוי ונחזיר true

**static void updateExpression()**

במידה ונטענים מילים חדשות מטקסטים חדשים על ידי הפונקצייה load , יש לעדכן מיקומי ביטויים קיימים בטקסט החדש

כדי לעדכן את המאגר , ניכנס ללולאה על טבלת הexpression לכל ביטוי הנימצא שם נפנה לפונקצייה newExpression(expr)

**נושא נוסף :**

כנושא נוסף בחתי להוסיף כריית מידע על הטקטסטים , יודפסו חוקי הקשר שנבעו מתוך טבלת הנתונים המובנים על ידי מימוש של אלגוריתם אפריורי על הטקסטים שניטענו , והתוצאות יתרעננו בכל הוספה של טקסטים נוספים

**static void ExecuteAprioriOnMetaData()**

נגידר רשימת רשימות transactions לשמירת הנתונים עליהם נבצע את האלגוריתם

ניפתח חיבור לשרת ונוציא שאילתא כדי לאחזר את כל הרשומות מטבלת MetaData מהם נחלץ את כל התכונות הרלוונטיות ( מטופל, רופא, דיאג, טיפול, סיכום) מכל רשומה.

ניבנה רשימה transaction פרטנית לרשומה הניקראת עם כל תכונותייה ונוסיף rשימה זו לרשימת הרשימות transactions

ניקרא לפונקצייה FindAssociationRules(transactions) לכרות כללי הקשר בהתבסס על רשימה זו ויחזיר רשימת מחרוזות שהם כללי הקשר , ואז לכל כלל ברשימת התוצאה נידפיס למשתמש

**static List<string> FindAssociationRules(List<List<string>> transactions)**

פונקצייה שמקבלת רשימה של תכונות ומחזירה כללי הקשר מתוכם

נגדיר ספי מינימום תמיכה (minSupport = 0.2) ומינימום ביטחון (minConfidence=0.5).

על כל רשומה נבדוק אם היא עומדת בתנאי הסף על ידי קריאה לפונקצייה MeetMinSupport(transaction, transactions, minSupport)

אם רשומה עומדת בסף התמיכה, ניצור כללים מועמדים על ידי קריאה לפונקצייה GenerateCandidateRules(transaction)

לכל כלל מועמד נחשב את סף הביטחון על די קריאה לפונקצייה CalculateConfidence(candidateRule, transaction, transactions)

אם עומד בסף המינימלי, הוסף אותו לרשימת חוקי ההקשר שיוחזרו.

**static bool MeetMinSupport(List<string> itemset, List<List<string>> transactions, double minSupport)**

מקבל את הרשומה, ורשימת כלל הרשומות וסף תמיכה מינימלי.

סופר את מספר הרשומות לפי הנוסחה הבאה :

Support(X)= Transactions containing X / Total transactions

מחזירה true אם התמיכה גדולה או שווה לסף התמיכה המינימלי שקיבל.

**static List<string> GenerateCandidateRules(List<string> transaction)**

מקבל רשומה כקלט. יוצר רשימה של כל הכללים המועמדים האפשריים על ידי התאמה של כל פריט בה עם כל פריט אחר בלולאת for מקוננת , ומחזיר אותה

**static double CalculateConfidence(string rule, List<string> transaction, List<List<string>> transactions)**

מקבל כלל, רשומה ורשימת הרשומות הכללית כקלט.

מחשבת את הconfidence של הכלל על ידי פיצול הכלל לאגפים מימין ושמאל ל =>

סופר את מספר הרשומות לפי הנוסחה הבאה :

Confidence(A⇒B)= Transactions containing A / Transactionscontaining both A and B

ומחזיר את התוצאה של החישוב