**המחלקה להנדסת תוכנה**

**פרויקט גמר – תשע"ו**

**יצירת תבניות שאילתות**

**Metaquery Generator**

**מאת**

**מירי יעקבי**

**מנחה אקדמי:** פרופ' רחל בן אליהו זוהרי **אישור: תאריך:**

**רכז הפרויקטים:** פרופ'/דר' ...... **אישור: תאריך:**

מערכות ניהול הפרויקט:

|  |  |  |
| --- | --- | --- |
| # | מערכת | מיקום |
| 1 | מאגר קוד | <https://github.com/miriyak/MetaqueryGenerator> |
| 2 | יומן | <https://trello.com/b/37ZLusj8/metaquery-generator> |
| 3 | ניהול פרויקט (אם בשימוש) |  |
| 4 | הפצה |  |

**תוכן עניינים**

[מילון מונחים (לפי סדר א'-ב') 5](#_Toc498592374)

[1 מבוא 6](#_Toc498592375)

[1.1 רקע כללי 6](#_Toc498592376)

[1.2 מה זהData Mining **?** 6](#_Toc498592377)

[1.3 מתי משתמשים בו? 7](#_Toc498592378)

[1.4 טכניקות כריית נתונים 7](#_Toc498592379)

[1.5 שימוש ב - Metaqueries ככלי לכריית מידע 7](#_Toc498592380)

[1.6 מסגרת הפרויקט 8](#_Toc498592381)

[2 תיאור הבעיה 9](#_Toc498592382)

[2.1 מה זהMetaquery **?** 9](#_Toc498592383)

[2.2 מדדים ל-Metaquery 10](#_Toc498592384)

[2.3 המטרה 10](#_Toc498592385)

[2.4 המצב כיום 10](#_Toc498592386)

[2.5 הפתרון שיוגש בפרויקט 11](#_Toc498592387)

[2.6 דרישות ואפיון הבעיה 12](#_Toc498592388)

[2.6.1 הדרישות העיקריות מהמערכת 12](#_Toc498592389)

[2.6.2 רשימת הדרישות מהפרויקט 12](#_Toc498592390)

[2.7 הבעיה מבחינת הנדסת תוכנה 12](#_Toc498592391)

[3 תיאור הפתרון 14](#_Toc498592392)

[3.1 מהי המערכת 14](#_Toc498592393)

[3.1.1 יעדים עיקריים 14](#_Toc498592394)

[3.1.2 ממשקים חיצוניים 14](#_Toc498592395)

[3.2 ארכיטקטורת המערכת 15](#_Toc498592396)

[3.2.1 תרשים ארכיטקטורת המערכת 15](#_Toc498592397)

[3.2.2 תיאור טבלאי של התהליכים העיקריים בתרשים 16](#_Toc498592398)

[3.3 אפיון פתרון האלגוריתם 17](#_Toc498592399)

[3.3.1 יצירת השאילתות 17](#_Toc498592400)

[3.3.2 שימוש בחישובים קודמים כתוצאות ביניים 17](#_Toc498592401)

[3.3.3 אלגוריתם חיפוש בעץ 18](#_Toc498592402)

[3.3.4 האלגוריתם Metaquery Generator 19](#_Toc498592403)

[3.3.5 אפשרויות הרחבה ושיפור ביצועים 21](#_Toc498592404)

[4 סקירת עבודות דומות בספרות והשוואה 22](#_Toc498592405)

[4.1 סקירה והשוואה 22](#_Toc498592406)

[4.1.1 מהו Association rules 22](#_Toc498592407)

[4.1.2 מבנה Association rules 22](#_Toc498592408)

[4.1.3 המשותף ל-2 הטכניקות 22](#_Toc498592409)

[4.1.4 מה חסר ב- Association rules 23](#_Toc498592410)

[4.2 סיכום ומסקנות 23](#_Toc498592411)

[5 נספחים 24](#_Toc498592412)

[5.1 רשימת ספרות / בביליוגרפיה 24](#_Toc498592413)

[5.2 תכנון הפרויקט – תאריכים משוערים 25](#_Toc498592414)

[5.3 טבלת סיכונים 26](#_Toc498592415)

[5.4 טבלת דרישות (User Requirement Document) 27](#_Toc498592416)

[5.4.1 דרישות מממשק המשתמש 27](#_Toc498592417)

[5.4.2 דרישות מהאלגוריתם Metaquery Generator 27](#_Toc498592418)

[5.4.3 דרישות מבחינת הנדסת תוכנה 28](#_Toc498592419)

## מילון מונחים (לפי סדר א'-ב')

להלן מוצגים מונחים, אשר יאפשרו להבין בצורה מיטבית את המערכת:

| # | מונח | הגדרה |
| --- | --- | --- |
|  | **Metaquery**  או בקיצור **MQ** | תבנית שאנו מנסים למצוא לה התאמה בבסיס הנתונים, במטרה לגלות קשרים נסתרים חשובים בין טבלאות. לדוגמא: |
|  | **ראש ה-Metaquery** | החלק השמאלי של התבנית |
|  | **גוף ה-Metaquery** | החלק ימני של התבנית |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. מבוא
   1. רקע כללי

"סמנכ"לית הבנק לא האמינה...הממצאים שהתקבלו עלו על כל דמיון. מסתבר שלקוח שבמהלך שלושת החודשים האחרונים ביצע סדרה מוגדרת של פעולות פיננסיות, יחד עם עלייה של X% במספר הפניות היזומות שלו לבנק וירידה של Y% ביתרות העו"ש הוא פוטנציאלי לנטישה! הצלילה שלנו לנתונים הקטינה משמעותית את אחוז הלקוחות הנוטשים"

מתוך ממצאי פרויקט Data Mining אצל אחד מלקוחות נורמטיב

במשך מאות שנים בני אדם זיהו "ידנית" תבניות ודפוסים במידע, אולם הצמיחה בנפח המידע בזמן המודרני הגבירה את הצורך בגישות אוטומטיות יותר.

בשנים האחרונות, יחד עם מהפכת המחשבים והתקשורת, כמות המידע שהוצפה לאדם הולכת וגדלה. היקפי המידע שאנו מקבלים לצורך קבלת החלטות אינם ניתנים לעיבוד וניתוח על – ידי שימוש ביכולתנו הטבעיות בלבד. שימוש במאגרי מידע ובמסדי נתונים מהווה היום חלק אינטגראלי של כל תהליך ניהולי ועסקי.

ההנחה היא כי בידי המשתמש אין מידע מוקדם על קשר בין נתונים שונים המצויים במאגר המידע. עקב כך גובשה תפיסה חדשה בעיבוד וניתוח מידע, והיא  - כריית נתונים, שמטרתה חשיפת קשרים בין נתונים שונים במאגר המידע המנותח.

* 1. מה זהData Mining **?**

Data Mining (כריית מידע), הוא יישום אנליטי ומורכב העוסק בחקירה מעמיקה של נתונים באמצעות אלגוריתמים מתקדמים, במטרה לזהות דפוסים קבועים ומערכות יחסים בין המשתנים, ובכך להפוך את המידע הרב ליעיל ושמיש.  
תהליך כריית המידע מאתר מגמות ודפוסים רבי חשיבות ו"אוצרות של מידע" אחרים מתוך אוקיינוס עצום של נתונים נטולי משמעות – כל זאת מבלי להעלות השערות מוקדמות המתבססות על דפוסי חשיבה מוכרים ומקובעים.

הכינוי לתחום זה מקורו בדמיון בין שיטות אלו לבין הפעולה של כריית יהלומים. הרעיון הוא לחפור עמוק בתוך מסדי נתונים גדולים בעלי מידע גולמי באמצעות כלים חישוביים עוצמתיים כדי לגלות את היהלומים שאינם נראים לעין .

* 1. מתי משתמשים בו?

לרוב המטרה היא מסחרית, ויעדו של התהליך האנליטי היא ליצור מודל חיזוי, אשר בעזרתו מנהלי הארגון מקבלים החלטות. ארגונים בעידן הנוכחי מוצפים בכמות עצומה של נתונים מורכבים ולא מנוצלים ולכן היכולת להפיק מהם מידע עסקי משמעותי, הינה קריטית ומכאן החשיבות הרבה המיוחסת לפתרונות מתחום ה-Data Mining.

הדפוסים והחוקיות המתגלים בתהליך כריית המידע משמשים בד"כ את הלקוחות לצורך הערכת סיכויים וניבוי התנהגויות, ואלו מסייעים להם בקבלת החלטות אסטרטגיות כגון: אפיון לקוחות והחלטה להתמקד בפלח לקוחות מסוים, שיפור תהליכים קיימים ועוד.

כך למשל, בנק יכול לפתח מודל היכול לקבוע את מידת ההסתברות שלקוח מסוים יחזיר הלוואה שתינתן לו, ועל-פי כך לשלול או לעודד מתן הלוואות ללקוחות מסוימים.

* 1. טכניקות כריית נתונים

קיימים סוגי אלגוריתמים שונים עבור כריית מידע, לדומא:

|  |  |  |
| --- | --- | --- |
|  | טכניקה | תחום |
| 1 | Decision Trees | בינה מלאכותית |
| 2 | Neural Networks | רשתות נוירונים |
| 3 | K-Nearest Neighbor | גאומטריה חישובית |
| **4** | **Association Rules** | **מסדי נתונים** |

בפרויקט זה נתמקד באלגוריתם מהסוג הרביעי.

* 1. שימוש ב - Metaqueries ככלי לכריית מידע

פרויקט זה הינו מחקרי העוסק בנושא Metaqueries – שאילתות על. נושא זה שייך למחלקה של כריית מידע מתוך בסיסי נתונים ויוסבר במפורט בהמשך.

הפרויקט מתבסס על מאמר שנכתב על ידי פרופ' רחל בן-אליהו זהרי ושותפיה למחקר [1]. מאמר זה עוסק בMetaqueries ככלי לכריית מידע, באמצעותו ניתן לחפש קשרים בין טבלאות ועמודות בבסיס הנתונים.

* 1. מסגרת הפרויקט

בניית מערכת אשר תפעיל אלגוריתם ליצירת כל ה-Metaqueries העשויים להיות בעלי פתרון ותשלח אותם למערכת Metaquery solver אשר תפתור אותם בפועל בהתאם ל-DB נתון.

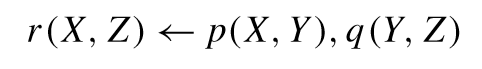
מתוך סיכום המאמר:

"Several research topics remain open in the field of data mining using metaqueries. One interesting direction is to develop a system that will generate the metaqueries automatically. A simple way to do this is to go over all the possible combinations. A more sophisticated approach would be to learn in which direction the interesting information can be found from answers to preliminary metaqueries…" *(Metaqueries: Semantics, complexity, and efficient algorithms,* *[Rachel Ben-Eliyahu-Zohary](http://www.sciencedirect.com/science/article/pii/S0004370203000730" \l "!)**,* [*[Ehud Gudes](http://www.sciencedirect.com/science/article/pii/S0004370203000730" \l "!), Giovambattista Iann*](http://www.sciencedirect.com/science/article/pii/S0004370203000730#!)*i).*

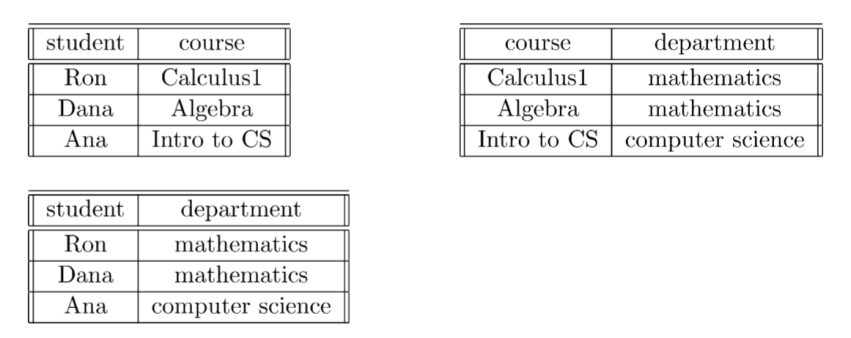
1. תיאור הבעיה
   1. מה זהMetaquery **?**

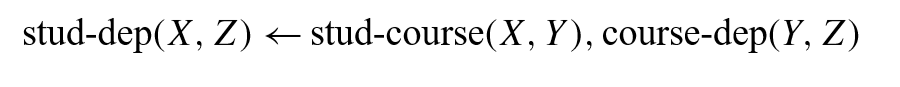
Metaquery זה תבנית שאנו מנסים למצוא לה התאמה בבסיס הנתונים, במטרה לגלות קשרים נסתרים חשובים בין טבלאות.

דוגמא לתבנית Metaquery:



דוגמא להתאמה שנמצאה על DB נתון:





החלק השמאלי של התבנית מכונה **ראש ה-Metaquery**

החלק ימני של התבנית מכונה **גוף ה-Metaquery**

בהתאמה לעיל התגלה כלל חדש: אם סטודנט לוקח קורס והקורס שייך למחלקה מסוימת, סביר להניח שהסטודנט שייך לאותה מחלקה.

* 1. מדדים ל-Metaquery

הסכנה בניתוח מורכב זה היא שתמיד יש סיכוי שהתוצאה שתתקבל מהרצת Metaquery תהיה מקרית ולא מייצגת. לכן נקבע ערכי וודאות מ-2 סוגים, שמטרתם למנוע הצגה של מידע זניח למשתמש:

* תמיכה (Support value): התמיכה בכלל היא אחוז הרשומות ביחס שמכילות את כל הפריטים מ-2 צדי התבנית.  
  בצורה אינטואיטיבית נאמר שאם לכלל יש תמיכה נמוכה יתכן שהוא התגלה במקרה ואין מספיק עדויות כדי לבסס אותו.
* וודאות (Cnfidence): הוודאות של כלל היא אחוז הרשומות שמספקות את ראש השאילתה מבין הרשימות שמספקות את גוף השאילתה.  
  בצורה אינטואיטיבית נאמר שאם לכלל יש וודאות נמוכה סביר להניח שאין קשר בין גוף השאילתה לראש השאילתה.
  1. המטרה

בהנחה שיש מכונה שיכולה לענות על Metaqueries (להלן Metaqueries Solver), נרצה לספק לה את התבניות שאותן היא תפתור על פי סדר מחושב מראש, במטרה למצוא תבניות בעלי תמיכה גבוהה וודאות גבוהה (בהתייחס לערכי סף של תמיכה וודאות הניתנים על ידי המשתמש)

* 1. המצב כיום

בעקבות המאמר של פרופ' רחל בן אליהו זוהרי, תמר באש, סטודנטית לתואר שני כתבה אלגוריתם שייתן מענה למטרה זו. האלגוריתם כתוב כ- pseudo code, ונדרש ליישם אותו ולשכלל אותו בהתאם.

**הערה: במסמך להלן מפורטים העקרונות ומתווה ראשוני לבניית האלגוריתם שפותח על ידי תמר.   
לאחר בחינת האלגוריתם ויישומו יבוצע עידון ועדכון שלו כדי לשפר את יעילותו.**

* 1. הפתרון שיוגש בפרויקט

מטרת הפרויקט הינה בניית מערכת אשר תייצר את כל התבניות (ה-(Metaqueries הקיימות, ושליחתם ל- Metaqueries Solver לפתרון הטכני של התבנית.

דבר זה ייעשה באמצעות מימוש אלגוריתם יעיל אשר ידאג לתיעדוף סדר שליחה של תבניות, וסינון תבניות לא רלוונטיות במטרה לצמצם עלויות חישוב.

במערכת שלנו לא נתעכב על מציאת פתרון לmetaquery נתון אלא נרצה לייצר Metaqueries שאותן יש לבדוק.

Metaquery

Generator

Metaquery

Solver

* 1. דרישות ואפיון הבעיה
     1. הדרישות העיקריות מהמערכת
* המערכת תאפשר למשתמש להכניס כקלט DB וערכי סף.
* המערכת נדרשת לייצר עבור הקלט הנתון את כל הmetaquries העשויים להיות בעלי פתרון.

(נגדיר MQ כבעל פתרון –אם נמצא עבורו יחסים מתאימים לתבנית, שערכי support ו-confidence של ההתאמה שלהן עומדים בערכי הסף שקבענו)

* פתרון של Metaquery (ע"י מערכת MQ Solver) הוא תהליך מורכב, והוגדר כבעיית NP קשה (הוכח במאמר של פרופ' בן אליהו)

מכיוון שכך, יש חשיבות גדולה להמעיט כמה שניתן בהפעלת ה-Solver, ולכן:

המערכת תיתן דגש על יעילות של האלגוריתם במישורים הבאים:

1. הסתמכות על תוצאות של פתרונות MQ קודמים. במקרים בהם MQ הוא תת תבנית של MQ אחר.
2. מתן אפשרות למשתמש להצביע על תבניות או טבלאות שמעניינות אותו.

* המערכת תידרש כחלק מהתהליך לשלוח דוחות של תוצאות MQ בעלות פתרון בזמן בו הם התגלו.
  + 1. רשימת הדרישות מהפרויקט

ראה להלן בנספח, סעיף 5.5.

* 1. הבעיה מבחינת הנדסת תוכנה

אנו מעוניינים שהמערכת תמצא את כל הmetaqueries הקיימים לכל DB נתון.

ככל שה-DB מורכב יותר, כך מספר התבניות יהיה גדול יותר. כדי לצמצם עלויות חישוב, יש לחשוב היטב על אלגוריתם שייבנה ביעילות מירבית.

דוגמאות להחלטות שהתבצעו בעקבות בעיות עליהם האלגוריתם אמור לתת מענה:

* שמירת ה-metaqueries יהיה במבנה של עץ
  + לכל MQ יהיו בנים שמהווים את אותה שאילתה בתוספת יחס אחד נוסף
  + סדר יצירת ופתרון השאילתות יהיה ע"פ העומק בעץ
* העץ ייבנה תוך כדי ריצה (ולא יישמר כולו בזיכרון)
  + נקצץ מהעץ את הענפים שגילינו שאין להם פתרון.
  + יש לוודא שתבנית לא נוספה בעץ יותר מפעם אחת
  + בהוספת תבנית יש לוודא שאנו לא מוסיפים תבנית שכבר נמחקה כי אין לה פתרון.
* שימוש בתוצאות חישוב של הדרגות הגבוהות בעץ עבור הדרגות הנמוכות בעץ.
* המערכת מחייבת התממשקות למערכת נוספת עבור שליחה תבניות וקבלת תוצאות.
* המערכת תעבוד בצורה א-סינכרונית כדי לפתור את הבעיות הבאות:
  + פתרון MQ היא בעית NP קשה שעלולה לקחת זמן. ולא נרצה להגיע למצב של נעילת משאבים.
  + המערכת תלויה במערכת נוספת ששולחת אליה את הפתרונות, ואנו לא רוצים ליצור מצב של מערכת לא יציבה בעקבות מערכת אחרת.
* המערכת תעבוד על מספר תבניות בו זמנית
  + במטרה לזרז את התהליך ניתן לבצע את האלגוריתם בצורה יותר מקבילית, ולא לרוץ על תבנית אחת ולחכות לתוצאות.
  + יש צורך להגביל את העבודה המקבילית, כדי למנוע עבודה מיותרת. (לדוגמא אם התחלנו לעבוד במקביל על 2 תבניות שאחת היא תת תבנית של השנייה, לא נוכל להסתמך על פתרון של תבנית האב.) לכן נעבוד במקביל רק על תבניות שנמצאות באותו עומק בעץ.

**במסגרת הזו יש לחקור את צורת הפיתוח לנושאים הבאים:**

1. רכיב ניהול תורים.
   * בשלב זה נשקלות החלופות הבאות:
     1. שימוש ברכיב לניהול תורים (כדוגמת RabitMQ – רכיב קוד פתוח(
     2. פיתוח רכיב לניהול תורים המותאם לדרישות הפרויקט.
2. עבודה ב- Multi-threading
3. עבודה א-סינכרונית.
4. שימוש בשירותי ענן (או דרך אחרת) לשמירת המידע המועבר בין המערכות, והחלטה על פורמט מתאים.
5. תיאור הפתרון
   1. מהי המערכת

המערכת היא מערכת עצמאית המורכבת מממשק למשתמש המפעיל אלגוריתם ליצירת כל ה-Metaqueries האפשריים עבור מסד נתונים וערכי סף נתונים.

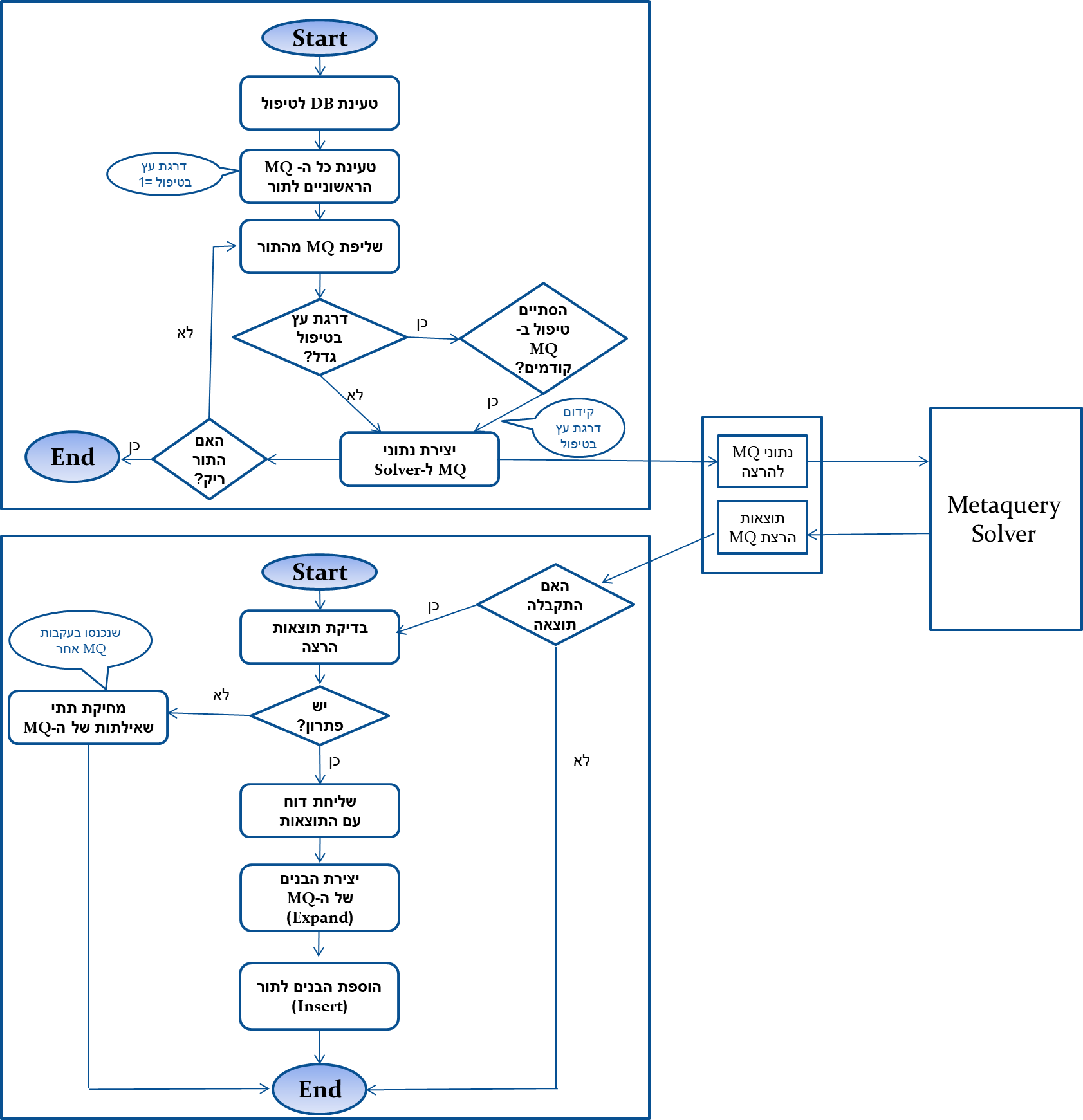
* + 1. יעדים עיקריים
* בניית אלגוריתם שייצר את כל הMetaqueries שעשויים להיות בעלי פתרון.
* שימת דגש על יעילות האלגוריתם כדי למזער את הפעלת ה Metaqueries Solver (במיוחד לאור העובדה שפעולת חישוב ה Metaquery שנעשה על ידי ה-Solver היא יקרה).
  + 1. ממשקים חיצוניים

המערכת תתממשק עם מערכת Metaquery solverבצורה דו כיוונית:

* + העברת מידע: המערכת תיצור תבנית Metaquery בתוספת פונקציונליות מיוחדת ותשלח את התבנית לחישוב.
  + קבלת מידע: המערכת תקבל תוצאות הרצה של תבנית, ותמשיך את פעולת האלגוריתם בהתאם.

* 1. ארכיטקטורת המערכת
     1. תרשים ארכיטקטורת המערכת

התרשים מתאר את ארכיטקטורת המערכת ומורכב מהשלבים העיקריים שיכיל הפתרון:



* + 1. תיאור טבלאי של התהליכים העיקריים בתרשים

|  |  |  |
| --- | --- | --- |
| **מס'** | **שלב** | **תיאור** |
| **1.** | **טעינת MQ ראשוניים לתור** | הכנסה של שאילתות ראשוניות הכוללות מספר מינימלי של יחסים ומשתנים לתור. |
| **2.** | **בדיקת דרגת העץ** | כדי לייעל את האלגוריתם, העבודה תהיה מקבילית. אך נגביל זאת לעבודה על MQ באותה דרגה בעץ. מכיוון שכל דרגה מסתמכת על תוצאות הדרגה שמעליה. |
| **3.** | **יצירת נתוני**  **MQ ל-Solver** | רכיב MQ Solver מבצע את הפתרון בפועל. המערכת תיצור את הנתונים הנדרשים לו ותשמור את המידע במאגר משותף. (לדוגמא בענן) |
| **5.** | **שליחת דוח**  **עם התוצאות** | במקרה של-MQ יש פתרון – זה אומר שנמצא מידע בעל חשיבות על קשר בין טבלאות. נרצה לשלוח את המידע לגורמים הרלוונטיים. (בצורת דוח בענן, אימייל, או כל פיתרון אחר שיימצא לנכון) |
| **6.** | **יצירת הבנים**  **של ה-MQ**  **(Expand)** | הפונקציה מקבלת MQ ומייצרת את כל הבנים, באמצעות הוספת משתנה או יחס אחד נוסף בכל פעם.  דוגמא בנספח סעיף 5.2.3 |
| **7.** | **הוספת הבנים לתור**  **(Insert)** | הפונקציה מכניסה לתור את השאילתות החדשות שנוצרו.  לפני הכנסת MQ לתור נבדוק שאינה קיימת, כיוון שאותה שאילתה יכולה להיווצר על ידי שני אבות שונים. |
| **8.** | **Metaquery Solver** | כפי שניתן לראות בתרשים – תהיה הפרדה מלאה בין מערכת זו לבין המערכת שפותרת את ה-MQ. ניתן להתייחס אליה כאל קופסא שחורה שמקבלת קלט, ומחזירה פלט בפורמט שהוגדר מראש בין המערכות. |
| **9.** | **נתוני MQ להרצה,**  **תוצאות הרצת MQ** | נתונים אלו הם הנתונים שעוברים בין 2 המערכות. יישמרו בפורמט שייקבע בין המערכות, במקום משותף לשניהם. |
| כפי שניתן לראות מהתרשים, המערכת תעבוד בצורה אסינכרונית, ולא תמתין לתוצאות ההרצה שמחושבים ב-MQ | | |

* 1. אפיון פתרון האלגוריתם

כפי שכתבנו לעיל, תמר באש כתבה אלגוריתם לצורך מציאת תבניות.

להלן פירוט האלגוריתם והתהליך שהוביל לבנייתו:

* + 1. יצירת השאילתות

שמירת אוסף ה- Metaqueries במבנה של עץ, כאשר לכל metaquery יהיו בנים שמהווים את אותה שאילתה בתוספת יחס אחד נוסף.

סדר יצירת ופתרון השאילתות יהיה על פי העומק בעץ.

המטרה: שימוש בתוצאות חישוב של הדרגות הגבוהות בעץ עבור הדרגות הנמוכות בעץ.

העץ ייבנה תוך כדי ריצה (ולא יישמר כולו בזיכרון)

* + 1. שימוש בחישובים קודמים כתוצאות ביניים

בהינתן metaquery ופתרון עבורה, ניתן להשתמש בפתרון זה כתוצאות ביניים לפתרון מהיר ויעיל עבור שאילתה המערבת משתנה/יחס אחד נוסף באופן הבא:

נניח יש לנו פתרון ל:

r(X, Z) ← p(X, Y ), q(Y, Z)

ואנחנו רוצים למצוא פתרון ל:

r(X, Z) ← p(X, Y ), q(Y, Z), s(X,Z)

**בהינתן פתרון של שאילתה, שאילתה זהה עם הוספת יחס נוסף תהיה קלה יותר לפתרון:**

נעבור על מרחב הפתרון הקודם ונבדוק אילו מהיחסים עדיין מתקיימים. כיוון שהוספנו אילוץ, מרחב הפתרון יהיה בהכרח קטן/שווה לפתרון הקודם. לכל היותר נעבור על כל הDB, אך ברוב המקרים נצמצם את מרחב החיפוש.

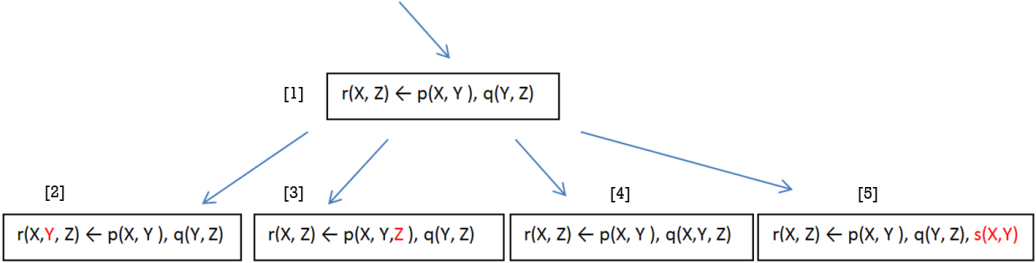
עם כל הוספת יחס ל-metaquery נקטין את מרחב החיפוש והדרך לפתרון תהיה קצרה ופשוטה יותר. ניתן להתייחס ל metaquery המקורי כאל "אב" ול-Metaqueries החדשים כאל "בנים". אם נחשב את הבנים לאחר חישוב האב נוכל להשתמש בתוצאת החישוב של שאילתת האב ולהפוך את חישוב הבנים לזול ומהיר יותר. כמובן שאם לשאילתה אין פתרון גם לבנים אין פתרון ואין טעם לבדוק אותם.

**נציע דרך לפתרון אוסף השאילתות באופן יעיל:**

נשמור את כל השאילתות במבנה נתונים של עץ כאשר לכל metaquery יהיו בנים שמהווים את אותה שאילתה בתוספת יחס אחד נוסף. סדר פתרון השאילתות יהיה על פי העומק בעץ, כאשר העלים יחושבו אחרונים. ככל שמתקרבים לעלים כך הפתרון קל יותר. נרצה לעבור על כל העץ: בכל פעם לחשב metaquery ולייצר את הבנים, לחשב את הבנים וכו' עד לסיום.

לדוגמא:

שאילתה [1] היא ה"אב", שאילתות [5]-[2] הן דוגמאות לבנים. כאשר נייצר את התבניות, נייצר קודם את [1] ונשלח לפתרון. לאחר מכן נייצר את הבנים ונספק לsolver- את פתרון האב לצורך חישוב מהיר יותר.



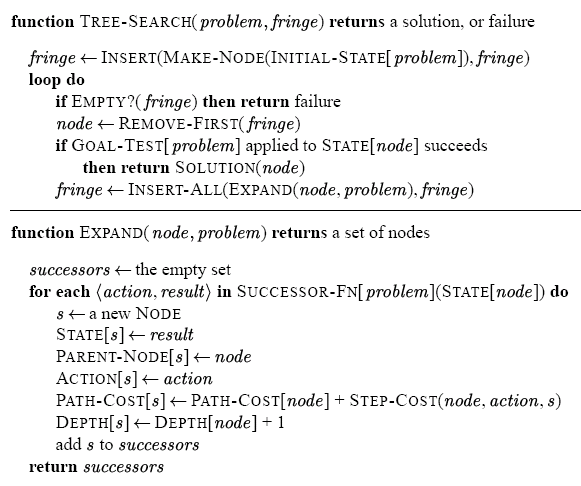
* + 1. אלגוריתם חיפוש בעץ

אלגוריתם חיפוש בעץ משתמש במבנה של עץ כדי למצוא פתרון לבעיות חיפוש. בבעיות חיפוש יש מרחב מצבים, מצב התחלה ומצב מטרה המסודרים כקדקודים בעץ, כאשר ניתן לבצע פעולות שונות כדי לעבור ממצב למצב. האלגוריתם מחפש שביל בעץ (כלומר סדרת פעולות) שיוביל ממצב התחלה למצב מטרה.

**האתגר:** הבעיה יכולה להיות אקספוננציאלית. מבנה העץ לא ידוע מראש והוא נבנה תוך כדי ריצה. לכן אי אפשר להשתמש באלגוריתמים פשוטים של גרפים.

אלגוריתם חיפוש בעץ בונה את עץ החיפוש תוך כדי ריצה ומשתמש בתור כדי לשמור קדקודים. בתחילה מכניסים לתור את מצב ההתחלה, בכל איטרציה שולפים קדקוד מהתור ובודקים אם זה מצב מטרה. אם כן – סיימנו. אם לא – מייצרים את הבנים ומכניסים אותם לתור בסדר מסוים.

להלן פסאודו קוד לאלגוריתם לחיפוש בעץ



* + 1. האלגוריתם Metaquery Generator

**האלגוריתם שלנו יעבוד באופן דומה, כאשר:**

* אצלנו אין "מצב מטרה". נרצה לחשב את כל השאילתות האפשריות.
* אין מצב התחלה אחד. בתחילה יוכנסו לתור כל השאילתות הכוללות מספר מינימלי של יחסים (2 טבלאות).
* כאשר לשאילתה אין פתרון אין צורך לייצר את הבנים.
* הסדר בתוך התור ייקבע על פי עומק הקדקוד בעץ.
* האלגוריתם אינו מסתיים בהגעה למצב מטרה אלא לאחר שבדקנו את כל השאילתות האפשריות, כלומר כאשר התור ריק.

**האלגוריתם יעבוד באופן הבא:**

נכניס לתור שאילתות ראשוניות הכוללות מספר מינימלי של יחסים ומשתנים. בכל פעם נשלוף קדקוד (metaquery) וננסה לפתור. אם מצאנו פתרון – נייצר את הבנים ונכניס אותם לתור, תוך שמירה על תוצאת החישוב של שאילתת האב. אם אין פתרון לא נייצר בנים אלא נמשיך ונוציא מהתור את ה- metaquery הבא. כאשר התור ריק - סיימנו.

הסדר בתוך התור ייקבע על פי עומק הקדקוד בעץ (אפשר לחשב בקלות בעזרת מספר המשתנים המעורבים בשאילתה) – נחשב תחילה שאילתות גבוהות יותר בעץ ונתקדם לכיוון העלים.

להלן פסאודו קוד של האלגוריתם:

MQs\_Generator(DB\_schema,

support\_threshold, confidence\_threshold , fringe)

{

fringe 🡨 initial\_MQs (schema)

loop do

if Empty (fringe) then return

MQ 🡨 remove\_first (fringe)

sol 🡨 solver (MQ, support\_threshold, confidence\_threshold)

if (sol)

send\_report(MQ, sol)

MQ\_collection🡨 Expand (MQ, schema, fringe, sol)

Insert(MQ\_collection, fringe)

}

* + 1. אפשרויות הרחבה ושיפור ביצועים
  + **fixed Metaqueries**

במאמר נכתב כי קיים מקרה בו מענה על metaquery הוא פולינומיאלי, והוא כאשר מדובר בסכמה וב- metaquery קבועים:

"We identify several tractable classes: answering a fixed query on a database with a fixed schema (but varying relation size) can be done in polynomial time"

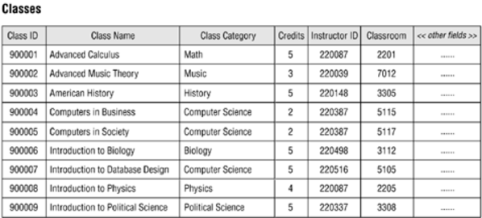
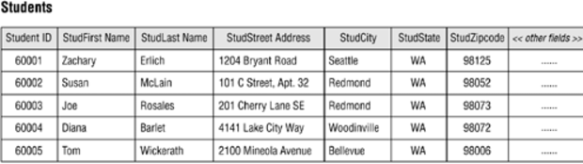
ניתן להשתמש במקרה זה ולבחור מראש מספר metaqueries קבועים (לדוגמה, metaquery המייצגת יחס טרנזיטיביות) וסכמה קבועה של בסיס הנתונים. ישנן סכמות שהן קבועות כמו בסיס נתונים רלציוני, את האחרות נצטרך להתאים. באופן הזה יהיו לנו מספר metaquery קבועות שניתן לחשבן בזמן פולינומיאלי והן יהיו הראשונות שנבחן.

* + **Significant Metaqueries**

בחלק זה נרצה שיהיה ברשותנו ידע מקדים על שאילתות משמעותיות. כמובן שמידע וודאי ומדויק לא נוכל לקבל מראש. אך ננסה למצוא דרכים להעריך האם metaquery תהיה משמעותית.

מידע בעל ערך חייב להיות מידע שכיח. אחרת, אם קיים קשר כלשהו אך הוא נדיר, אין לו משמעות אמתית עבורנו. זהו בדיוק התפקיד של ערכי ה-support והconfidence-, לוודא שקשרים שמצאנו קיימים בשכיחות גבוהה מספיק בבסיס הנתונים.

נבצע מדגם על בסיס נתונים כדי לחזות ערכי סף גבוהים. המדגם עשוי להיות על השורות או על העמודות.



ככל שנמצא יותר התאמות כך נשער ערכי support וconfidence גבוהים יותר.

אפשר לבחור את החלק מבסיס הנתונים באופן רנדומלי או לבקש מהמשתמש שיצביע לנו על טבלאות ועמודות חשובות בבסיס הנתונים וזה החלק שנדגום.

לדוגמה, בבסיס נתונים של בנק, המשתמש יתעדף את שדה המשכורת כחשוב יותר משדה השם.

ניתן ציון ל metaquery לפי תוצאות המדגם וזה ידרג את החשיבות שלה.

נוכל להשתמש בזיהוי metaqueries משמעותיות לצורך סידור הקדקודים בתור,שייקבע על ידי שני מדדים:

1. עומק הקדקוד בעץ
2. בין הקדקודים השייכים לאותה רמה בעץ נבחר לחשב את המשמעותיים ביותר תחילה.
3. סקירת עבודות דומות בספרות והשוואה
   1. סקירה והשוואה

כדי להבין יותר מה זה Metaquery ולמה הוא משמעותי, נשווה את זה לטכניקה הידועה: "mining of association rules".

* + 1. מהו Association rules

Association rules הינם כלי משמעותי בתחום כריית המידע. בעזרת Association rules ניתן לבצע כריית מידע מעל בסיסי נתונים . Association rules מאפשרים לנו למצוא ולאפיין קשרים בין שדות בבסיס הנתונים. על סמך הקשרים הנ"ל ניתן יהיה להסיק מידע חדש שלא היה ידוע קודם לכן מתוך הנתונים שלפנינו

שיטת ה- Association Rules מאפשרת למצוא קשרים מעניינים בין משתנים ב- database, שעוזרים לשיווק, לקבלת החלטות, ולניהול עסקים. אפליקציות רבות קיימות בתחום זה, הכוללות גילוי קשרים לניתוח שוק הכלכלה, תכנון קטלוג, ניתוח מכירות מוצרים, סידור חנויות, וסיווג לקוחות על-בסיס דפוסי קניה.

Association rules נחקרו באופן רחב בספרות מבחינת הערכת יעילות ותחזוקה.

* + 1. מבנה Association rules

בהינתן סט של טרנזקציות, כאשר כל טרנזקציה הינה סט של פריטים, ה- association rule הינו ביטוי מהצורה *X* → *Y*, כאשר X ו- Y הינם סטים של פריטים.

* + 1. המשותף ל-2 הטכניקות
  + שתיהן עוסקות בכריית מידע על בסיס נתונים – כאשר המטרה מציאת קשרים בין משתנים ב-DB.
  + שתיהן נמדדות בערכי סף: Support Value ו- Confidence Value
  + שתי הטכניקות משקפות אחת את השנייה.

**ניתן להגיד ש- Metaqueriesמשמש כהרחבה של Association rules**

* + 1. מה חסר ב- Association rules
  + עובד על טרנזקציות בלבד.
  + סוג של תבנית קבועה בלבד.
  + הטכניקה דורשת שהמידע יאוחסן בטבלה אחת.
  + כדי לעבוד על DB מ"העולם האמתי", נדרש לעיתים קרובות עיבוד מוקדם כדי להפוך את הנתונים המנורמלים לנתונים בטבלה בודדת.
  1. סיכום ומסקנות

Metaqueries משמש כהרחבה של Association rules ומאפשר:

* + מגוון רחב של תבניות שנבנות בהתאם ליחסים ב-DB.
  + התבססות על סכמה רלציונית.
  + עבודה על מספר יחסים באותה תבנית של MQ.
  + היחסים יכולים להיות יחסים מקומיים או Views.
  + מינימום עיבוד מוקדם.

באמצעות שימוש ב-Metaquery נוכל לקבל מידע נסתר על נתונים שלא היינו יכולים לזהות זאת בשיטה הקודמת.

באמצעות שימוש באלגוריתם שנפתח בפרויקט זה ניצור את כל התבניות האפשריות, וכך נוכל לקבל את מקסימום המידע שניתן לנצל מה-DB.

1. נספחים
   1. רשימת ספרות / בביליוגרפיה

הערה: זוהי רשימה זמנית, לשלב זה.

[1] R. Ben-Eliyahu-Zohary, E. Gudes, G. Ianni, **Metaqueries: Semantics, complexity, and efficient algorithms,** Artificial Intelligence, 149(1):61–87 (2003)

<http://www.sciencedirect.com/science/article/pii/S0004370203000730>

[2] Hahsler, Michael, **Introduction to arules – A computational environment for mining association rules and frequent item sets**, *Journal of Statistical Software*(2005)

<https://mran.revolutionanalytics.com/web/packages/arules/vignettes/arules.pdf>

[5] Wei-Min Shen, KayLiang Ong, Bharat Mitbander, Carlo Zaniolo, **Metaqueries for Data Mining**. Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, MA ,pp. 375-397 (1996).

<http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=9217B6E3598E0D2D564A921A117392A0?doi=10.1.1.28.4694&rep=rep1&type=pdf>

[6] F. Angiulli, R. Ben-Eliyahu-Zohary, G. Ianni, and L. Palopoli. **Computational properties of metaquerying problems**. ACM Trans. on Comput. Logic, 4(2):149–180 (2003). <https://arxiv.org/pdf/cs/0106012.pdf>

* 1. תכנון הפרויקט – תאריכים משוערים

|  |  |
| --- | --- |
| 25/09/2017 | פגישת התנעה עם המנחה |
| **22/10/2017** | **סיום שלב ההתנעה** |
| 03/11/2017 | פגישת הנחייה לקראת הגשת הצעת פרויקט |
| 11/11/2017 | הגשת טיוטת הצעה למנחה |
| 16/11/2017 | מחקר וסקירת ספרות |
| **19/11/2017** | **סיום הגשת הצעת הפרויקט** |
| 01/12/2017 | בניית ארכיטקטורה |
| 20/12/2017 | מימוש אב טיפוס |
| 30/12/2017 | בדיקות |
| 31/12/2017 | פגישה עם המנחה לבדיקת המענה שניתן באב טיפוס |
| 10/01/2018 | תיקונים בהתאם לממצאי בדיקות והערות מנחה |
| 20/01/2018 | הכנת דו"ח אמצע וסרטון |
| **21/01/2018** | **סיום אב טיפוס (אלפא)** |
| 01/02/2018 | תיכון מפורט |
| 01/03/2018 | מימוש |
| 09/03/2018 | בדיקות |
| 10/03/2018 | פגישה עם המנחה לבדיקת המימוש |
| 31/03/2018 | תיקונים בהתאם לממצאי בדיקות והערות מנחה |
| **01/04/2018** | **סיום שלב בנייה (בטא)** |
| 01/05/2018 | הכנת דוח סופי וסרטון |
| **22/06/2018** | **מסירת הפרויקט. בחינה סופית** |
| 25/06/2018 | אריזה והפצה |
| 29/06/2018 | השלמת תיעוד |
| **30/06/2018** | **העברה** |

* 1. טבלת סיכונים

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **הסיכון** | **חומרה** | **מענה אפשרי** |
| 1 | כחלק מתהליך של המערכת, האלגוריתם אמור להפעיל את ה- Metaquery solver ולחכות ממנו לתשובה כדי להתקדם בתהליך. אך זה עלול להיות איטי מאד.  (כפי שהוזכר, זוהי בעיית NP קשה) | גבוה | המערכת תיבנה בצורה עצמאית, וההתקשרות שלה עם מערכת ה-Metaquery solver תהיה בצורה אסינכרונית, כך שלא ניצור מצב בו המערכת ממתינה לתגובה. |
| 2 | כמות המידע, וצורת שמירת המידע במסדי נתונים מורכבת וגדולה, ויהיה קשה לנתח את יכולות האלגוריתם על בסיס מסד נתונים נתון. | בינוני | נבחר מסד נתונים ראשוני ונעבוד עליו קודם באב טיפוס, ויוקדש זמן לבדיקות מקיפות על מספר מסדי נתונים שונים מורכבים יותר. |
| 3 | כריית נתונים מאופיינת בניתוח מידע על סמך קבוצות נתונים גדולות מאוד, ולפיכך יש חשש שהאלגוריתם יעבוד בצורה איטית ויזלול משאבי מחשב רבים. | בינוני | יש לבדוק ולטפל במשאבי החומרה בהתאם.  אם בבדיקות נגלה שהמערכת כבדה מדי, נגדיל את המשאבים של מחשב ההרצה.  בנוסף, כדי לנצל משאבים: נעבוד במודל Multi threading, וכך נוכל להריץ בו זמנית תבניות שונות. |
| 4 | האלגוריתם עלול להשתנות משמעותית בהתאם להרצות שונות של מסדי נתונים שמתגלה רק במהלך הפרויקט.  מצב זה עלול לחבל בעמידה בלוחות הזמנים של הפרויקט. | נמוך | הקצאת זמן להרצת ה-DB לבדיקה |
| 5 | שגיאות ורעש: לרוב מסדי נתונים מכילים נתונים שאינם שלמים, אינם מדויקים או חסרים  מצב זה עלול לפגוע בהסקת מסקנות האלגוריתם. | בינוני | תהליך מיפוי שגיאות חייב להיות ע"י המשתמש המכיר את הלוגיקה של הנתונים. ולכן באחריות המשתמש יהיה לטייב את ה-DB לפני הכנסת ה-DB למערכת. |

* 1. טבלת דרישות (User Requirement Document)
     1. דרישות מממשק המשתמש

ממשק משתמש מאפשר להפעיל את האלגוריתם על DB מסויים. המשתמש יכניס את הקלט הנדרש כמפורט להלן, בו ישתמשו בשלב הבא – יצירה אוטומטית של ה-MQ

| **מס'** | **דרישה** |
| --- | --- |
|  | המערכת תאפשר הכנסת DB מסוג Sql Server בלבד . |
|  | המערכת תאפשר הכנסת ערכי סף. |
|  | המערכת תפעיל את האלגוריתם בצורה אסינכרונית – על ידי רכיב תורים  המטרה בדרישה זו היא ליצור מערכת שיודעת לעבוד במקביל על יותר מ-DB אחד. |
|  | כחלק משמירת נתוני הקלט בקוד, ננהל meta-data על האובייקט כדוגמת תאריך התחלת התהליך, תאריך עדכון אחרון, סטטוס התהליך, וכדו'. |
|  | המערכת תאפשר למשתמש לצפות במצב הנתון של כל הרצה בזמן אמת. המערכת תאפשר בדיקת סטטוס התהליך, סיום יזום של התהליך (למקרה שהמשתמש רואה שקיבל את המידע המספק וכו') |

* + 1. דרישות מהאלגוריתם Metaquery Generator

מרכיב זה הוא המרכיב המרכזי בפתרון הנדרש. המרכיב ישלב את האלגוריתם המפורט [בסעיף 5.5 בנספח](#_מילון_מונחים_(לפי_1) ותפקידו ליצור Metaqueries ולהעבירם לרכיב ה-solver.

| **מס'** | **דרישה** |
| --- | --- |
|  | האלגוריתם יזהה את כל התבניות האפשריות עבור DB נתון . |
|  | האלגוריתם יפעיל את מערכת Metaquery Solver כדי לחשב פתרון ל-MQ. |
|  | האלגוריתם יפעיל רק תבניות שידוע שלאבות שלהם היה פתרון. |
|  | האלגוריתם יפעיל כל תבנית פעם אחת לכל היותר. |
|  | האלגוריתם ישלח מידע על תוצאות של MQ בעלי פתרון. |

## 

* + 1. דרישות מבחינת הנדסת תוכנה

| **מס'** | **דרישה** |
| --- | --- |
|  | המערכת תתמוך בעבודה בצורה א-סינכרונית |
|  | האלגוריתם יתמוך בעבודה מרובת תהליכים Multi-threading |
|  | המערכת תשלב/תיצור כלי לניהול תורים (שיודע לעבוד בצורה אסינכרונית) |
|  | המערכת תשתמש בשירותי ענן לשמירת מידע המועבר בין מערכות.  (או בדרך אחרת למטרה זו) |
|  | המערכת תשלח מידע למשתמש באמצעות מיילים |