- אם ישום מסי חואים המחוררים ב- Conjunctive selection שלבים: 1. Close .3 SQL commands.2 Open connection AND ואת אחד מהם ניתן למצוא לפי אחת השיטות לעיל, אזי עבור connection כל רשומה שעונה על תנאי זה, נבדוק את יתר התנאים. **גישות**: Embedded commands .1- הפקודות הן חלק מהשפה. □N- Conjunctive selection using a composite index למשל:LIbrary of DB functions.2 JAVA ב-SQLJ למשל:JAVA ב-JIDRA ב-JAVA , ו-

קיימים יותר משדה אחד בתנאי שוויון, וקיים אינדקס על כל אחד מהשדות. ADO.NET ב-#C. מכילים את כל השיטות לעבודה עם בסיסי Conjunctive selection by intersection of record driver טוען –Driver manger – טוען **pointers** -היימים אינדקסים משניים על השדות בתנאי השוויוו. שיוצר connection מול ה-DB session) DB- יש לציין את שם ה-נשלוף את הכתובות לרשומות המתאימות לפי האינדקס. נבצע

ספה OR-אם התנאים מחוברים ב-Disjunctive selection לבצע אופטימיזציה. יש לבצע חיפוש בנפרד לכל תנאי ואחר כך מעבר -FFETCH . באמצעותו עוברים על התוצאות: Cursor לרשומה הבאה.Result set הוא cursor לתוצאות השאילתה. לאחד את כל הרשומות. אם אחד מחייב מעבר סידרתית/לינארי אזי מבצעים רק מעבר לינארי ובודקים את כל התנאים. אלגורי<u>תמים ליישום JOIN</u>

שתי לולאות, אחת בתוך השנייה, רצות על Nested loop $B_{\scriptscriptstyle R} + t_{\scriptscriptstyle R} B_{\scriptscriptstyle R} B_{\scriptscriptstyle S}$: רשומות הטבלאות. ${\it <u>ניתן לביצוע תמיד</u> זמן חישוב$

(tR) מספר רשומות בבלוק של הרלציה הגדולה) שתי לולאות, אחת בתוך Block nested loop (single loop)

השנייה, רצות על בלוקי הטבלאות.זמן חישוב: אם קיים מספיק Block – אחרת $B_{\!\scriptscriptstyle R}+B_{\!\scriptscriptstyle S}$ היחסים משני לאחד מאחר בזיכרון מקום

> , בייא משני היחסים גדול מהזיכרון הפנימי – nest loop $B_R + B_S \lceil B_R \div (B-2) \rceil$

> BI הוא מספר הבלוקים בזיכרון, Br – הרלציה הקטנה].

כאשר קיים אינדקס על - Index nested loop (single loop) אחד השדות בjoin. לולאה אחת שעוברת על הרלציה שאין לה אינדקס, ועבור כל רשומה נחפש רשומה מתאימה לפי האינדקס בטבלה השנייה. זמן חישוב: $B_R + t_r(x+k)$ - בטבלה השנייה. זמן בx, R. הזמן הנדרש לאיתור רשומה באינדקס-k,מסי בלוקים מתאימים ב-2 אינדקס $x \approx 2 - 4$ וו hash אינדקס אינדקס ב-2 (S-מתאימים ב-2 אם האינדקס מקובץ, אם דליל-לחשב כמה k=1) B+ TREE רשומות עבור ערך מסויים ב-S לפי התפלגות)

m JOINאם באות ממוינות על שדות - Sort merge join אזי זמו חישוב הוא סכום אורכי 2 הטבלאות. במקרה שהטבלאות לא ממוינות אך קיימים אינדקסים על שדות ה-JOIN נעזרים באינדקסים למציאת הרשומות ובדיקת תנאי ה-JOIN. אם הטבלאות לא ממיונות ואין אנדקסים מתאימים

 $B_R \log(B_R) + B_S \log(B_S) + B_R + B_S$

ובמקרה שיש רק M בלוקים בזיכרון, הטבלאות לא נכנסות במלואן בזיכרון:

 $2B_R \log_{M-1} B_R + 2B_S \log_{M-1} B_S + (B_R + B_S)$ שיטה משופרת : זמן חישוב כמו Hash JOIN

על HASH אייי הפעלת פונקציות B-1 עייי חלוקה של – Hash join ערכי התכונות שלפיה נעשה JOIN. ב. חילוק כל קבוצה של R לתתי קבוצות באמצעות פונקציית hash <u>חדשה</u> ושומרים בזיכרון הראשי. קבוצות המתאימות של S קוראים את הקבוצות המתאימות של ${
m S}$ R ומנסים למצוא התאמות.

כמות הזיכרון הנדרשת: א. מספר החלוקות הוא לכל היותר B-1. , אכן בשלב לכן אכן ארול פוא: אוא הוא חלוקה לכן העני הגודל של הוא הגודל אל הוא הגודל אל הוא הוא הוא הוא הוא הוא ה מספר מחיצות: $\frac{fB_R}{B-1}$ איזון (f>1) איזון (f>1) איזון מספר מחיצות: איזון איזון $.3(B_R+B_S)$: זמן חישוב סהייכ. f=1 למוחלט. f=1

בודל תפסתי - ERD

* ניתן להמיר קשר טרינארי לקשרים בינארים ע"י שימוש בישויות ולשוח.

ישויות חלשה ניתן להחליף בתכונה מורכבת מרובת ערכים. (עובד פרויקט וכלים) – Aggregation – מאפשר לחבר קשרים מעבר מ-ERD לרלציות מנורמלות

בקשר טרינארי – נבנה טבלה על שם הקשר (תצוגה בגלריה)

דוגמא - החזירו את שם המפלגה שחבריה בכנסת ה-12 היו בעלי הגיל הממוצע הגבוה ביותר.

ממוצע ת.ל מינימלי = גיל ממוצע גדול SELECT M.PartyName

FROM Members AS M JOIN KnessetMembers AS KM ON M.MemberID=KM.MembersID WHERE KM.KnessetNum=12

GROUP BY M.PartyName

HAVING AVG(M.BrithYear) <= ALL (SELECT AVG(M.BrithYear)

FROM Members AS M1 JOIN KnessetMembers AS KM1 ON M1.MemberID=KM1.MembersID

WHERE KM1.KnessetNum=14 GROUP BY M1.PartyName)

JOIN על עצמו – בשאילתות של 2 לפחות דוגמא - הציגו את כל המשתמשים שיש להם סיסמא הזהה לסיסמא של משתמש אחר.

SELECT u1.uid FROM Users as u1, Users as u2 WHERE u1.usernme<>u2.username AND u1.pass<>u2.pass

דוגמא אלגברה רלציונית –מצאו מרשמים שיש להם לפחות 2 תרופות שונות של חברת טבע

 $C \leftarrow (\Pi_{dname}(\sigma_{company="TEVA"}(Drugs)))) \bowtie drug-prescription$

 $\Pi_{\text{T1.per-no}}(\sigma_{\text{T1.per-no}=\text{T2.per-no}\land\text{T1.dname}\neq\text{T2.dname}}(T\times\rho_{\text{T1}}(T)))$

* ALTER – לבצע שינוי בטבלה לאחר שנוצרה כבר ALTER TABLE

השינוי שרוצים לבצע :שינוי אפשריים

* הוספת עמודה – ADD column_name datatype DROP COLUMN column_name - הורדת עמודה *

authentication-יDB). ה-connection מנהל את הטרנזקציות, חיתוך כתובות בזיכרון וניגש לנתונים. הstatement מבצע שאילתה שהוא מקבל כמחרוזת וזה מחזיר .result set

> -Stored procedures פרוצדורות/פונקציות שמאוחסנות ומבוצעות .DBMS-עייי ה מושגים

> > .תרשים מחלקות כמודל תפיסתי. – UML

עדכון עדכון – Constraints אילוצים על ערכי רשומה. Assertions – טענות על הנתונים שחייבות להתקיים בכל מצב. Triggers – הגדרת פעולות לביצוע בהכנסה/עדכון/מחיקה. Prepared Statement – פרוצדורה

אופטימיזציה SQL שמתבצעת הרבה פעמים ולכן נשמרת לצורכי אופטימיזציה SQL היא שיטת כתיבה של אילתות - Stored Procedure המאוחסנות בשרת של בסיס הנתונים. כמו בפרוצדורה רגילה, היא מאוחסנת עם קוד של התחלה, קוד של סיום, פרמטרים וגוף הפרוצדורה שבו נכתבות הפעולות המתבצעות בפרוצדורה.

CREATE TRIGGER [trigger name] [Before/After/Instead of] [Insert/Update/Delete] ON [Table name/View name]

FOR EACH [Row/Statement]

[Condition] [SQL Query]

הרשומה החדשה –

שנמחקה רשומה אחרי העדכון - New טבלה לפני העדכוו – Old טבלה לפני – Old :Views

להתייחס לפלט של שאילתה כאל טבלה נפרדת (בלי ליצור אותה פיזית). בפועל: השאילתה שמגדירה את המבט נשמרת במסד הנתונים ומופעלת כל פעם מחדש.

 $\left| \frac{P_S * P_r}{P_r + P_r} \right| : R \bowtie S$ בנוסחא לחישוב מספר רשומות בבלוק

.i מספר רשומות בבלוק -Pi

את שמקיימות אסי הרשומות מסי $-\sigma_{A=V}(R)$.1 את שמקיימות שמקיימות הערכה של פעולת את המספר המוערך של - $\sigma_{A \leq V}(R)$.2 התנאי המספר המוערך של - $\sigma_{A \leq V}(R)$.2 $\max(A,R)$ ו ו- $\min(A,R)$ שומות על התנאי במידה

c=0 או Min(A,R)>V מוחזקים בקטלוג: א. אם , בהעדר מידע סטטיסטי , בהעדר $p_R * \frac{v-\min(A,K)}{\max(A,R)-\min(A,R)}$

יהה ל- R אז $R \cap S = \emptyset$ אז 1. והה ל- 1. זהה ל- 1. אם מעולת אז 2 אז 1.

ומכיל P_R*P_S רשומות כשכל רשומה דורשת $S_R^*+S_S$ בתים R imes S מסי הרשומות S מפתח אר ב-S מפתח מסי הרשומות R או S = $\{A\}$ אם 2. אינו R א S = {A} אינו ב-S. 3. אם R א S אינו R אינו מפתח זר ב-S או ב-R, בהנחה שכל רשומה t ב-R מייצרת רשומות ובמקרה ב- $\frac{P_R*P_S}{V(A,S)}$ ב-S א R ב-R נעריך את מסי הרשומות ב-R א S ב-R נעריך את מסי

 $\frac{V(A,S)}{V(A,R)}$. ההפוך - החערכה הנמוכה מבין השניים היא המדוייקת. FULL OUTER |R| + |R ⋈ S|: OUTER JOIN הערכה ל-

 $|S| + |R| + |R \bowtie S| : IOIN$ סוגי אינדקסים

מניח שקובץ הנתונים ממויין לפי שדה שהוא מפתח -Primary .1 מועמד. האינדקס כולל כניסה 1 לכל בלוק רשומות (אינדקס דלילי).2. Clustering מניח שקובץ הנתונים ממויין אך לא לפי שדה מפתח.יכיל כניסה לכל ערך ייחודי של השדה.ה-ptr צביע לבלוק שממנו הערך הייחודי מתחיל, אם יש המשך הבלוק יצביע לבלוק שאחריו, כשאר כל בלוק מכיל ערך ייחודי אחר.**3**. Secondary- האינדקס ממויין, הנתונים לא ממויינים. אם נגדיר ע[ّ]ל שדה שהוא מפתח מועמד :כ[']ל כניסה מכילה מצביע לבלוק ומסי

רשומה בבלוק (אינדקס צפוף). אם נגדיר על שדה שהוא לא מפתח:כל כניסה תצביע לבלוק והבלוק יחזיק את המצביעים לרשומות עם הערך. (אינדקס צפוף/דליל).4. Multi-level-אינדוקס האינדקס. הערך באינדקס הראשון הוא הערך הראשון .B+Tree או BTree-בבלוק האינדקס עליו הוא מצביע. נשתמש ב-ההבדל ביניהם: ב-B+tree בצמתים שהם לא עלים נצביע לצמתים אחרים ולא למידע עצמו.**-Bitmap.5** כאשר יש מסי קטן של ערכים. מחזיק טבלה בינארית עם עמודה לכל ערך ושורה לכל רשומה. הביט יהיה דלוק אם הרשומה מכילה את הערך.Join.6-אינדקס לפי טבלה א,כאשר המצביעים הם לרשומות בטבלה ב, ניתן לוותר על שלב ה-join. אינדקס דליל – שומר רק חלק מהערכים של מפתח החיפוש (לעומת אינדקס צפוף)

אלגוריתמים ליישום SELECTION רחיפוש ליניארי על כל הטבלה, בזבזני, תמיד ניתן Linear search

.מחייב טבלה ממוינת כאשר התנאי שויון Binary search

חיפוש – Primary index to retrieve a single record Hash בסיוע אינדקס ראשי או נוסחת

השוואה – Primary index to retrieve multiple records (גדול/קטן וכוי) ביחס למפתח. שימוש באינדקס וחיפוש דינמי. Secondary Index (B+tree) to retrieve multiple

שדה ערך על שיויון ערך על שדה – records שאינו מפחח.

ALTER COLUMN column_name datatype

* ניתן באותו אופן עבור אילוצים עייי שינוי ה-Column ל-Constraint רק עבור ADD ו-DROP.

: FK עבור אילוץ של ALTER דוגמא לשימוש ב ALTER TABLE your_table
ADD CONSTRAINT <name> FOREIGN KEY (<key>)

REFERENCES PROGRAM (<key name>) ON DELETE cascade ON UPDATE no action

של המפתח המקורי – Delete\Update .your_table-עבור הרשומות cascade\no action ומה נעשה

INSERT INTO table_name

VALUES (value1, value2, value3,...)

* כאשר מכניסים את כל הערכים בטבלה דוגמא

INSERT INTO Customer VALUES (78901, 'Roy Peled',

*כאשר מכניסים חלק מהערכים בטבלה INSERT INTO table_name (column1,column2,column3,...) VALUES (value1, value2, value3,...)

DELETE FROM table_name WHERE some_column=some_value; DELETE FROM Ordered

דוגמא WHERE $Cust_Id = 12345$;

UPDATE table name SET column1=value1,column2=value2,... WHERE some_column=some_value;

UPDATE DrugPrescriptions SET DrugPrescriptions.days = 5 WHERE drugName = 'drug5';

תבנית של יצירת טבלה

CREATE TABLE table name (

column_name1 data_type(size) <constraint>, column_name2 data_type(size) <constraint>,

ADD constraint_name Check(...)

:(אופציונלי) Constraint - דוגמאות

check (birthdate >= 1/01/1900'), primary key, Unique CREATE TABLE Persons

(PersonID int primary key, LastName varchar,(255)

FirstName varchar.(255) Address varchar.(255)

Birthdate datetime, Add constraintName Check(birthdate >= 1/01/1900'))

 \mathbf{n} תרגיל – הוח שבמערכת יש 25,000 תלמידים. ידוע שכ-4,800 מהח

, גרים בבאר שבע. טבלת התלמידים מאורגנת כ-heap וקיים אינדקס B-tree על השדה Students.city. גודל רשומת אינדקס (כולל שדה ה-Bytes 64 (ptr וגודל index-page הוא Bytes 4. הנח שבממוצע מצויות בכל 15% index page מכמות הרשומות המקסימאלית. כמו-כן הנח שבזמן החיפוש באינדקס שורש האינדקס כבר מצוי בזיכרון. הסבר את אופן ביצוע הפעולה הבאה שבשאילתה: Where ... students.city = "Beer-Sheva"

חשב והסבר כמה פעולות קריאה ֹ (בלבד) מבסיס הנתונים תבוצענה : סדי לבצע אותה. **תשובה** – נחשב כמה רשומות יש ב-page אחד $\frac{index\ page\ size}{record\ size} = \frac{4KB}{64B} = \frac{4,096}{64} = 64.$

נחשב כמה רשומות נכנסות בפועל ב-Page אחד: 64 * 0.75 = 48

נחשב כמה דפים בממוצע שתופסת טבלה – רמת העלים: $\frac{Table\ records}{records\ in\ page} = \frac{25,000}{48} = 521.$

חישוב גודל ה-B+Tree: pages in leaves $=\frac{521}{1}=11-$ העלים רמת מעל אחת מעל. records in page

 $\frac{48}{pages\ in\ leaves}}{records\ in\ page} = \frac{11}{48} = 1 - n$ השורש רמת

כמות הדפים עבור רשומות של תלמידי בייש: $\frac{pages in leaves}{records in page} = \frac{4800}{48} = 100.$

יתפרס על 100 עלים מלאים, או 99 עלים ועוד 2 של חצי וחצי. לכן לכל היום 101 דפים עבור רשומות בייש.

מספר הקריאות הכולל: השורש כבר בזכרון, קריאה לדף יחיד מרמה 2 בעץ (אבא של העלה) + של הדף הראשוו בעלה שהוא בייש + קריאה של 100 דפים נוספים מרמת העלים (סהייכ 101 דפים). 1+100+1 = 102: סהייכ

שאלה -

U(a,d) - 1000 records R(a,b) - 1000 records T(c,d) - 100 records S(b,c) - 100 records

תכונות a,b,d כוללות 100 ערכים שונים כל אחד. תכונה c כוללת עשרה ערכים שונים.

כל הערכים מתפלגים אחיד.

ב. איזה מהסידורים הבאים סביר שיכיל את מספר הפעולות הקטן ביותר? הניחו כי פעולת ה-Join מבוצעת באמצעות אלגוריתם Hash-Join והזיכרון גדול דיו כך שמחיר כל פעולת Hash-Join פעמים גודל הטבלאות.

 $(((S \bowtie T) \bowtie R) \bowtie U)$ $J_1(100+100)*3=600$ $J_1(100 + 100) * 3 = 600$ $J_2(1000 + 1000) * 3 = 6000$ $J_3(10,000 + 1000) * 3 = 33,000$ 600 + 6000 + 33,000 = 39,000

דוגמא - החזירו את שמות המפלגות עבור המפלגות אשר קיבלו בסך הכול פחות מ-100000₪ בתרומות (<u>כולל המפלגות שלא קיבלו</u> <u>אף תרומה, כלומר לא מופיעות בטבלת donations).</u> אך רק עבור

מפלגות שהוקמו אחרי שנת 1960. **תשובה-**

SECLECT P.Pname FROM parties WHERE Fyear>1960

```
SELECT Pname FROM parties AS P, donations AS D
WHERE P.pname=D.pname GROUP BY P.Pname
HAVING SUM(D.Amount)>100000
דוגמא- מצאו את המתווך שקיבל את הסכום הגבוה ביותר עבור
מכירת נכס שבאחריותו
T \leftarrow \rho_{x(rId,amount)}(\Pi_{rid,percentage \times sAmount/100}(houses) \times sales)
Not \max \leftarrow \prod_{T: Td, T: amount} (\sigma_{t1: amount} \triangleright_{T: amount} (\rho_{t1}(T) \times T))
T-not max
דוגמא - הציגו מזהה ושם מתווך למתווכים שהראו את כל הנכסים
  \Pi_{rid,name}((\Pi_{rid,address}(visits) \div \Pi_{address}(houses))) \nearrow realtors)
דוגמא – מצאו שמות לקוחות שמעולם לא קיבלו תרופה שמייצרת
\Pi_{\text{name.}} (patients \bigcirc (\Pi_{\text{pid}}(patients) –
\Pi_{pid}(prescriptions) drug-prescriptions \Pi_{dname}(\sigma_{con}
             B_r + \left\lceil \frac{B_r}{B-2} \right
ceil * B_s \le X: פתרון של אייש עם ערך עליון
            B גולה כך את \frac{B_r}{B-2} = \left[\frac{X-B_r}{B_S}\right] נגלה כך את
(לא מינימלי שמזהים רשומה) – Super key
                    כמו הקודם רק מינימלי – Candidate Key
               Candidate-אחד מהמפתחות – Primary Key
                    האחרים Candidate-ה – Alternate Key
    שדה או שדות המתייחסים למפתח ראשי של – Foreign Key
 רלציה אחרת. שם המפתח הזר אינו חייב להיות זהה לשם המפתח
    הראשי המתאים, אך חייב להיות מאותו טיפוס. מפתח זר יכול
    להיות חלק ממפתח של טבלת האב. אם מפתח זר זהה למפתח
              הראשי של הטבלה יש קשר של 1 ל-1 בין הטלבאות.
     מפתח ערך של מפתח – Referential Integrity בעיית
                ראשי בטבלת האב נוצרת בעיה עם מפתחות זרים.
    כל הפונקציות הסיכמויות (SUM,AVG,COUNT,MAX,MIN)
                    מתעלמות מערכי NULL, מלבד (*) COUNT.
       הצג את השם וה-ID של המוצר שנמכר הכי הרבה פעמים:
SELECT [Products].[ProductID], [ProductName]
FROM [Order Details], [Products]
WHERE [Order Details].[ProductID] =
[Products].[ProductID]
GROUP BY [Products].[ProductID],[ProductName]
HAVING COUNT(*) >= ALL (SELECT
COUNT([OrderID] FROM [Order Details]
GROUP BY [ProductID])
                  שליפת כל המוצרים (שם ו-IDשהוזמנו בעבר:
SELECT [ProductID], [ProductName]
FROM [Products] P
WHERE EXISTS (SELECT * FROM [Order Details]
WHERE [ProductID] = P.[ProductID])
או SELECT [ProductID], [ProductName]
FROM [Products]
WHERE [ProductID] IN (SELECT DISTINCT
[ProductID] FROM [Order Details])
דוגמה לחילוק: מי הם הלקוחות שקנו אותם מוצרים כמו רן
Select C.customerName
From Customers C
Where Not Exists(
Select CBP.productsName
from CustomersBuyProducts CBP
Where CBP.customerName = C.customerName
EXCEPT
Select CBP.productsName
from CustomersBuyProducts CBP
                                             2 כורלה
Where CBP.customerName = 'RAN')
פרטי המטופל שקיבל הכי מעט טיפולים (שונים).
    R_0 \leftarrow \pi_{PatientId, TreatmentId}(PatientsTreatments)
    R1_{PatientId,NumOfTreatments}
    \leftarrow PatientId\Im COUNT_{TreatmentId}\left(R0\right)
     R2_{MinOfTreatments} \leftarrow \Im MIN_{NumOfTreatments}(R1)
 \leftarrow R_1 \bowtie_{NumOfTreatments = MinOfTreatments} (R_2)
R \leftarrow \pi_{PatientId,fName} \Big( Patients \bowtie_{PatientId=PatientId} (R_3) \Big)
```