**בס"ד**

**מיני פרויקט בבסיסי נתונים שלום תעיזי ודוד דגן**

תוכן עניינים

[הפרויקט שלנו 7](#_Toc517367492)

[תרשים ERD 7](#_Toc517367493)

[תיאור הישויות והקשרים 7](#_Toc517367494)

[ישויות 7](#_Toc517367495)

[קשרים 8](#_Toc517367496)

[נרמול הטבלאות 8](#_Toc517367497)

[תרשים DSD 9](#_Toc517367498)

[יצירת הטבלאות 9](#_Toc517367499)

[הכנסת נתונים 10](#_Toc517367500)

[שאילתות SQL 11](#_Toc517367501)

[בחירה - SELECT 11](#_Toc517367502)

[עדכון – UPDATE 12](#_Toc517367503)

[מחיקה – DELETE 14](#_Toc517367504)

[הכנסה – INSERT 14](#_Toc517367505)

[אינדקסים 15](#_Toc517367506)

[הרשאות 18](#_Toc517367507)

[Views 19](#_Toc517367508)

[פונקציות 20](#_Toc517367509)

[20](#_Toc517367510)

[נספחים 21](#_Toc517367511)

[נספח ראשון: שאילתות ואינדקסים 21](#_Toc517367512)

[נספח שני: עדכון,מחיקה 24](#_Toc517367513)

[נספח שלישי: views 25](#_Toc517367514)

[נספח רביעי: פונקציות 25](#_Toc517367515)

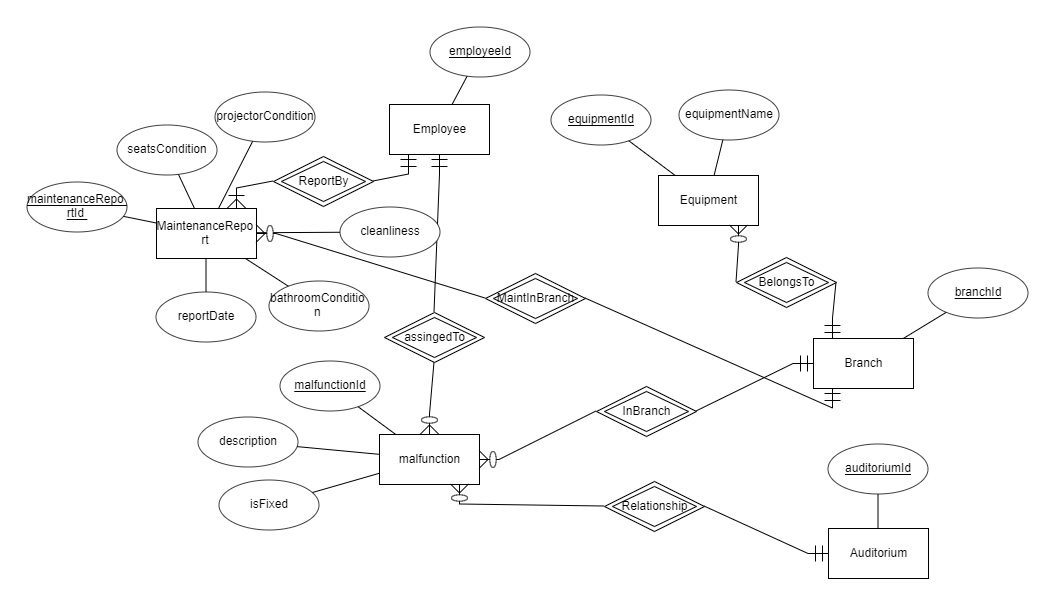
# הפרויקט שלנו

## תרשים ERD

כפי שהזכרנו, במחלקה שלנו, ישנן 3 ישויות עיקריות ו3 משניות המייצגות מפתחות זרים בישויות העיקריות :

עיקריות - דוח תחזוקה, ציוד ותקלה. משניות – עובד, אודיטוריום וסניף.

בשלב הראשון יצרנו תרשים ERD שיתאר את הקשרים בין הישויות הללו ואת התכונות שלהן.



## תיאור הישויות והקשרים

### ישויות

* Employee, Branch, Auditorium – ישויות אלה אחראיות על כל האולמות הקיימים בסניף, כל העובדים הקיימים והסניפים הקיימים בהתאמה.

ישויות אלה הינם חזקות, כיוון שהן יכולות להתקיים ללא תלות בישות אחרת. (שוב ישויות אלה לא באחריותנו אלה רק בשביל תמיכה בישויות שלנו)

* employeeId -מספר מזה של עובד.
* branchId – מספר מזהה של סניף.
* auditoriumId – מספר מזהה של אולם.
* MaintenanceReport – ישות זאת אחראית על דוחות התחזוקה.

ישות זאת הינה חלשה, כיוון שהיא תלויה בסניף ובעובד.

* reportDate – תאריך יצירת הדו"ח
* bathroomCondition – מצב השירותים באולם
* cleanliness– הנקיון הכללי
* ReportBy – שם יוצר הדוח
* projectorCondition – מצב המקרן
* seatsCondition – מצב הכיסאות באולם
* maintenanceReportId – מצב התחזוקה הכללי
* employeeId -מספר מזה של עובד (FK).
* branchId – מספר מזהה של סניף (FK).
* malfunction – ישות זאת אחראית על כל התקלות הנמצאות במערכת.

ישות זאת הינה חלשה, כיוון שהיא בכל הישויות החזקות.

* malfunctionid – מספר מזהה של החשבון (FK)
* description – תאור התקלה
* isFixed – האם התקלה תוקנה
* employeeId -מספר מזה של עובד (FK).
* branchId – מספר מזהה של סניף (FK).
* auditoriumId – מספר מזהה של אולם (FK).
* Equipment – ישות זאת אחראית על כל הציוד הנמצא בסניף.

ישות זאת הינה חלשה, כיוון שהיא תלויה בין היתר בסניף.

* equipmentName – שם הציוד
* equipmentId – מספר מזהה של הציוד
* branchId – מספר מזהה של סניף (FK)

### קשרים

* BelongsTo – הקשר בין Equipment לבין Branch. הקשר הינו חלש, כיוון שמקשר בין ישות חלשה לחזקה. הקשר הוא M:1 משום שיכול הרבה ציוד לסניף אבל לא הרבה סניפים לסוג מסויים של ציוד
* InBranch – הקשר בין malfunction לבין Branch. הקשר הינו חלש, כיוון שמקשר בין ישות חלשה לחזקה. הקשר הוא M:1 משום שיכול להיות לסניף אחד הרבה תקלות אך לא הרבה סניפים לתקלה אחת.
* Relationship – הקשר בין Auditorium לבין malfunction. הקשר הינו חלש, כיוון שמקשר בין ישות חלשה לחזקה. הקשר הוא M:1 משום שיכול להיות לאולם אחד הרבה תקלות אבל לא יכול להיות הרבה אולמות עבור תקלה אחת.
* MaintInBranch – הקשר בין MaintenanceReport לבין Branch. הקשר הינו חלש, כיוון שמקשר בין ישות חלשה לחזקה. הקשר הוא M:1 משום שיכול להיות כמה דוחות תחזוקה לסניף אך לא כמה סניפים בדוח תחזוקה אחד
* ReportBy – הקשר בין MaintenanceReport לבין Employee. הקשר הינו חלש, כיוון שמקשר בין ישות חלשה לחזקה. הקשר הוא M:1 משום שיכול להיות כמה דוחות שמשויכים לאדם אחד אך דוח אחד שמיוחס לכמה אנשים
* assingedTo – הקשר בין Employee לבין malfunction. הקשר הינו חלש, כיוון שמקשר בין ישות חלשה לחזקה. הקשר הוא M:1 משום שיכול להיות אדם אחד שמשיוך אליו כמה תקלות אך לא תקלה שמשויכת אליה כמה אנשים.

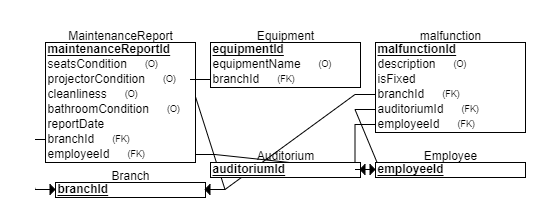
## 

## נרמול הטבלאות

היחסים עומדים ב- NF3 וב- BCNF : מכיוון שבכל טבלה, התלויות הפונקציונאליות הלא-טריוויאליות הן מהמפתח אל תכונות נוספות לכן מתקיים שלכל X Y , X הוא מפתח ולכן הם עומד בתנאים.

## תרשים DSD

על פי תרשים ה- ERD ועל ידי הבנת הקשרים בין הישויות, יצרנו תרשים DSD עבור החלק שלנו במערכת: מחלקת תחזוקת הקולנוע.



## יצירת הטבלאות

אחרי שהבנו כיצד בסיס הנתונים צריך להראות בצורה מדויקת, מה תכיל כל טבלה ומהם הקשרים בין כל הטבלאות, ניגשנו ליצירת הטבלאות בפועל בעזרת פקודות הcreate table.

יצרנו קוד לייצור הטבלאות באמצעות export SQL של האתר erdPlus, יצרנו קובץ SQL ואז העתקנו את קוד ה- SQL של כל טבלה אל תוכנת ה-plsql לשם יצירת הטבלאות בפועל:

CREATE TABLE Branch

(

branchId INT NOT NULL,

PRIMARY KEY (branchId)

);

CREATE TABLE Auditorium

(

auditoriumId INT NOT NULL,

PRIMARY KEY (auditoriumId)

);

CREATE TABLE Employee

(

employeeId INT NOT NULL,

PRIMARY KEY (employeeId)

);

CREATE TABLE MaintenanceReport

(

seatsCondition VARCHAR(50),

projectorCondition VARCHAR(50),

cleanliness VARCHAR(50),

bathroomCondition VARCHAR(50),

reportDate DATE NOT NULL,

maintenanceReportId INT NOT NULL,

branchId INT NOT NULL,

employeeId INT NOT NULL,

PRIMARY KEY (maintenanceReportId),

FOREIGN KEY (branchId) REFERENCES Branch(branchId),

FOREIGN KEY (employeeId) REFERENCES Employee(employeeId)

);

CREATE TABLE Equipment

(

equipmentName VARCHAR(20),

equipmentId INT NOT NULL,

branchId INT NOT NULL,

PRIMARY KEY (equipmentId),

FOREIGN KEY (branchId) REFERENCES Branch(branchId)

);

CREATE TABLE malfunction

(

malfunctionId INT NOT NULL,

description VARCHAR(200),

isFixed INT NOT NULL,

branchId INT NOT NULL,

auditoriumId INT NOT NULL,

employeeId INT NOT NULL,

PRIMARY KEY (malfunctionId),

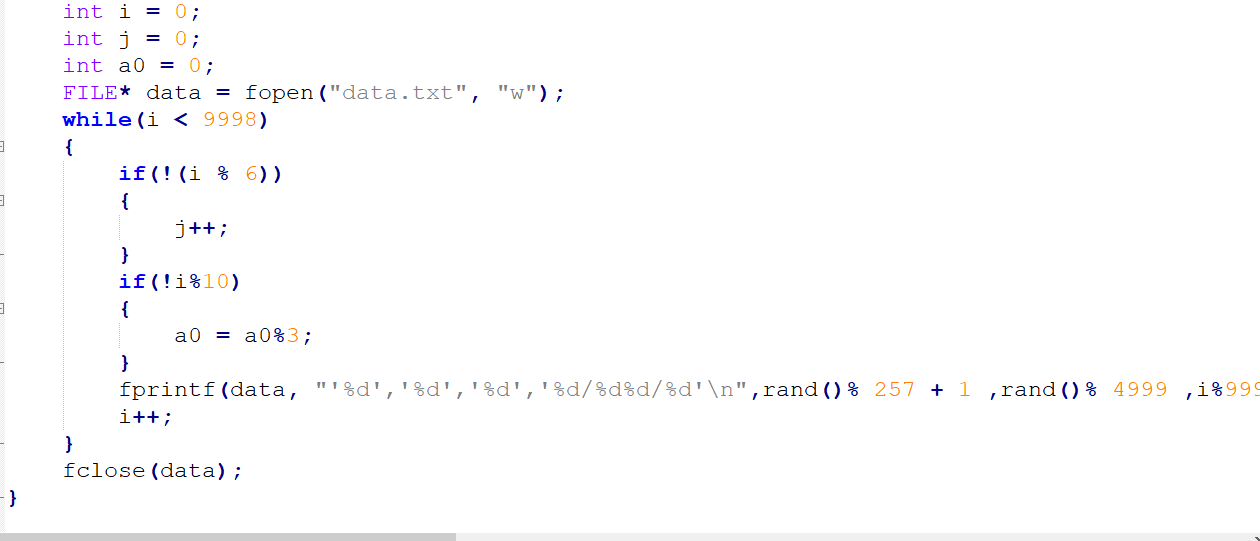
FOREIGN KEY (branchId) REFERENCES Branch(branchId),

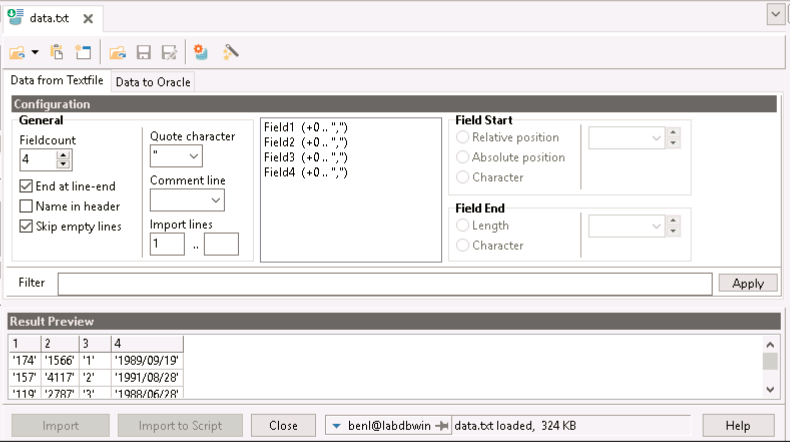
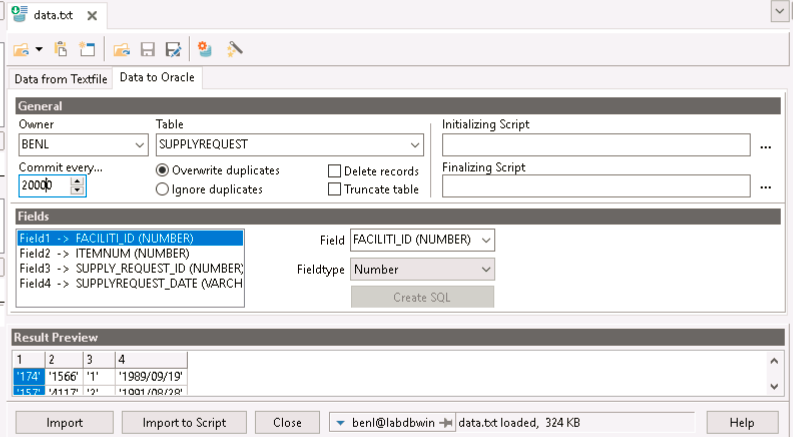
FOREIGN KEY (auditoriumId) REFERENCES Auditorium(auditoriumId),

FOREIGN KEY (employeeId) REFERENCES Employee(employeeId)

);

## הכנסת נתונים

על מנת לאכלס את הטבלאות שיצרנו בנתונים, השתמשנו ביבוא מידע מקובץ טקסט לתוך הטבלאות בבסיס הנתונים. יצרנו קובץ טקסט ובעזרת קוד בשפת C (הקוד מצורף ב- zip)אכלסנו אותו בנתונים.

אחר כך השתמשנו בtext importer בתוכנת ה- PL/SQL כדי להכניס את הנתונים לטבלאות. כך לדוגמא, הכנסנו נתונים לתוך הטבלה supplyrequest:

## שאילתות SQL

### בחירה - SELECT

*עבור כל שאילתה צרפו צילום מסך של תוצאות ריצה וזמן ריצה. לעדכון- תראו לפני ואחרי. למחיקה- תראו צילום מסך שהתוכנה כתבה שנמחקו מספר שורות ולהוספה תראו שהתווספו השורות שהוספתם.*

*עבור כל שאילתה צרפו צילום מסך של תוצאות ריצה וזמן ריצה. לעדכון- תראו לפני ואחרי. למחיקה- תראו צילום מסך שהתוכנה כתבה שנמחקו מספר שורות ולהוספה תראו שהתווספו השורות שהוספתם.*

לאחר שיצרנו את בסיס הנתונים והכנסנו לתוכו מידע, כתבנו כמה שאילתות מעניינות על מנת לתשאל אותו.

1. כחלק מהרצון של ההנהלה לדעת מה הם המתקנים ה"כבדים" של המלון, היא ביקשה לדעת מהם המתקנים אשר חויבו בהם לקוחות מעל 1000 ₪.
   * לכן כתבנו שאילתה המחזירה טבלה של כל שמות המתקנים וסכומי החיוב בחשבונות כאשר סכום החיוב גדול מ- 1000 ₪ .
2. ההנהלה החליטה לפנק במתנה את הלקוחות שבידם קבלת תשלום חשבון במתקן מעל 1000 ₪.

* לכן ההנהלה ביקשה רשימה של כל הלקוחות שהוצאה להם קבלה על סכום הגדול מ- 1000 ₪. כתבנו שאילתה המבצעת את הבקשה הנ"ל.

1. כחלק מבחינת הכדאיות של החזקת המתקנים במלונות, ההנהלה ביקשה לדעת מי הם המתקנים שסכום החיוב בכולל בהם הוא מעל 70,000 ₪.

* לכן כתבנו שאילתה המחזירה את מזהה המתקן וסכום החיוב הכולל של המתקן בהתאם לבקשת ההנהלה.

1. לאחרונה הובא לידי הנהלת הרשת כי בחלק מחדרי הכושר במלונות מוזמנים מוצרים שלא בהכרח נצרכים. כחלק מייעול הזמנות המוצרים לחדרי הכושר, ההנהלה ביקשה לדעת מהם המוצרים שהוזמנו לחדרי הכושר.

* לכן כתבנו שאילתה המחזירה טבלה של שמות המוצרים שהוזמנו לחדרי הכושר.

1. ההנהלה חוששת שעלויות אחזקת הסאונה גבוהות מדי ביחס למספר האנשים שפוקדים אותה, לכן היא מבקשת לדעת כמה אנשים השתמשו בסאונה על מנת לבחון את המשך קיומה במלון.

* לכן כתבנו שאילתה הסופרת את מספר המשתמשים בסאונה.

1. חבורת סטודנטים רצתה לערוך מחקר על השפעת הרימון על רמת ההנאה בגוף ולכן ביקשה מהנהלת המלון לבדוק באילו מתקנים במלון משתמשים ברימונים.

* לכן כתבנו שאילתה המחזירה את המתקנים בהם הזמינו רימונים מהאספקה.

1. להנהלת המלון הגיעה שמועה שהמתקנים מבצעים המון הזמנות שלא בהכרח נצרכות ולכן ההנהלה ביקשה לראות את כמות ההזמנות שבוצעו עבור כל מוצר שהוזמן על ידי המתקנים.

* לכן כתבנו שאילתה המחזירה עבור כל מתקן שהזמין מוצר מסויים את מזהה המוצר ומספר ההזמנות שבוצעו לאותו מוצר. מיינו את הרשימה לפי סוג המוצר, על מנת להקל בחיפוש המוצרים.

1. לידי מס הכנסה הגיעו שמועות על כך שהמלון העלים מס בשנת 2000 כאשר דיווח על הוצאות רבות. לכן הם דרשו מהנהלת המלון לספק להם דו"ח על הזמנת המוצרים בשנה זו והשימוש בהם במתקני המלון על מנת לבדוק אם אכן הייתה התנהלות שאינה חוקית.

* לשם כך כתבנו שאילתה המאגדת את כל הזמנות המוצרים שנעשו עבור המתקנים השונים בשנת 2000, ממוינים על פי מזהה המוצר.

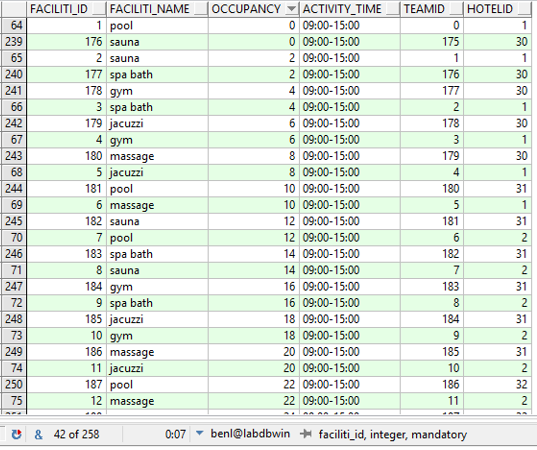
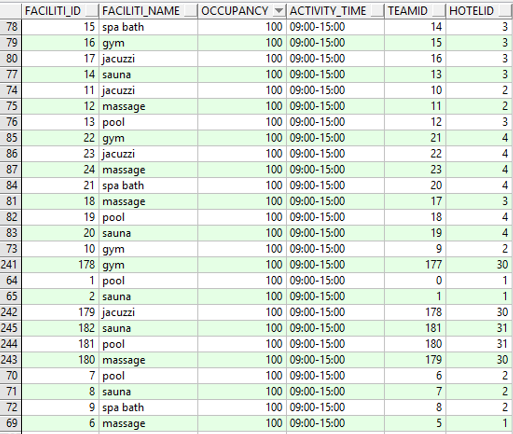
1. הנהלת המלונות הראשית ביקשה לבחון את כדאיות אחזקת חמשת המלונות הראשונים שנפתחו ברשת. לשם כך היא ביקשה לדעת את סכום ההכנסות מכל המתקנים בכל מלון בנפרד. ההנהלה מאד מסודרת ולכן לכל מלון ברשת ניתן מספר מזהה חדש בצורה סדרתית, לכן יש לבחון את המלונות שהמזהה שלהם קטן מ-6.

* לכן כתבנו שאילתה המחזירה רשימה שבה נמצאים חמשת המלונות הראשונים בלבד עם מזהה המלון וסכום ההכנסות הכולל מכל המתקנים שבו.

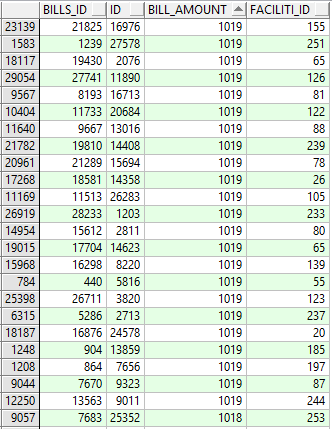
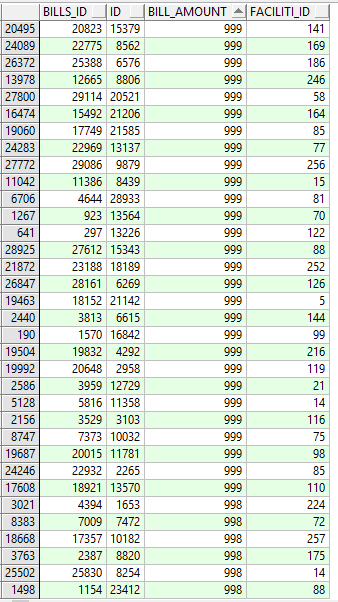
### עדכון – UPDATE

כתבנו כמה שאילתות על מנת לעדכן את בסיס הנתונים.

1. הנהלת המלון החליטה לערוך שדרוג כללי במתקני המלון ולאפשר לכל מתקן להכיל לפחות 100 אנשים בו זמנית. לשם כך כתבנו שאילתה המעדכנת את תפוסת המתקנים ל- 100, במתקנים בהם הייתה פחות מכך.



1. לאחר שהתלבטו רבות מה הפינוק אותו יעניקו ללקוחות שבידם קבלת תשלום חשבון במתקן מעל 1000 ₪, הוחלט שיוחזרו להם 10% מסכום הרכישה. לכן עלינו לעדכן את בסיס הנתונים ולהוריד לכל קבלה כזו את הסכום מהחשבון.



### מחיקה – DELETE

כתבנו שאילת מחיקה על מנת למחוק רשומות מבסיס הנתונים.

1. עקב מחסור כללי גדול בשוק ב-Cake - Pancake (שמספר המזהה שלו הוא 888) בוטלו כל ההזמנות הכוללות אותו. לכן מחקנו מבסיס הנתונים את כל ההזמנות שלו.



1. בתאריך 10/APR/00 הייתה תקלה גדולה בכל המחשבים ולכן התווספו לרשימת ההזמנות הזמנות שכלל לא בוצעו. לכן הוחלט למחוק מבסיס הנתונים את כל ההזמנות מאותו היום.



### הכנסה – INSERT

השתמשנו בשאילתות insert על מנת להכניס נתונים לטבלאות

1. הוחלט לפתוח במלון 1 מסעדה בנוסף לחדר האוכל שכבר קיים . הוספנו את המסעדה לבסיס הנתונים.
2. במתקן 17 אחראי המתקן שם לב שעומדים להיגמר ה- Duck - Legs ולכן פתח בקשה חדשה להזמנת הפריט. הוספנו לבסיס הנתונים את בקשת ההזמנה הזו.

## אינדקסים

*פה תסבירו על האינדקסים. לכל שאילתה תראו הבדל בזמן ריצה עם ובלי ותסבירו איזה אינדקס בחרתם ולמה.*

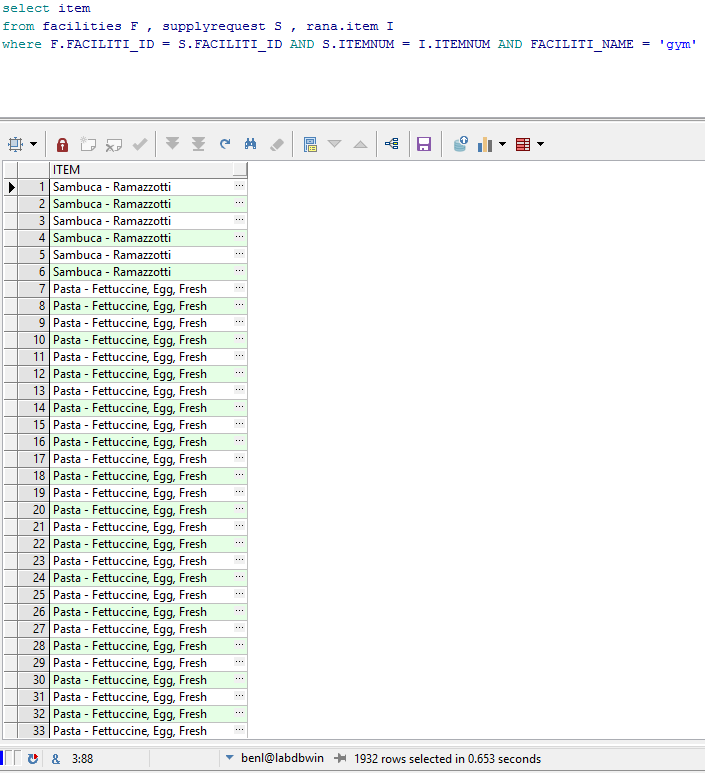
אינדקסים עוזרים למצוא במהירות גדולה יותר נתונים שנשמרו בטבלאות בבסיס הנתונים. אפשר לדמות את האינדקסים כמו מראה מקום בספר. במקום שנקרא את כל הספר כדי למצוא את מה שאנחנו מחפשים נלך למראה מקום שיראה לנו את כל המקומות שבהם מוזכר הנושא שאנחנו מחפשים. השימוש באינדקסים יחסוך לנו זמן ויהפוך את תהליך החיפוש ליעיל יותר. מהבחינה הזו האינדקסים בטבלאות של ה- SQL זהים לאינדקס בספר.

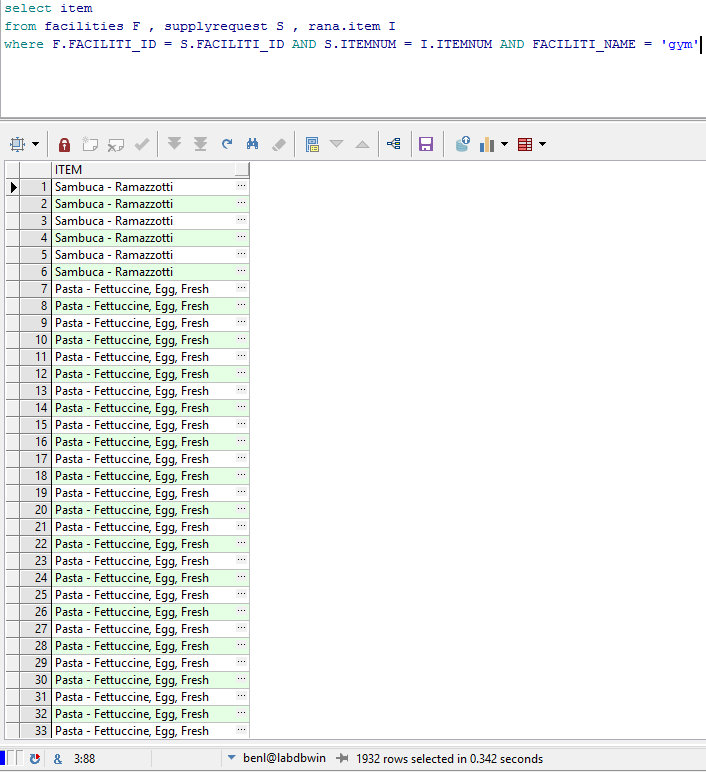
במידה ולא נגדיר אינדקס לטבלה אז בכל שאילתה על הטבלה השאילתה תגרום למעבר על כל הרשומות בטבלה עד שתמצא את כל הרשומות העונות למה שחיפשנו. כשנגדיר אינדקס מתאים אז החיפוש יהיה מהיר יותר כי הפניה לבסיס הנתונים תגרום לזה שמנוע החיפוש בבסיס הנתונים יפנה קודם לאינדקס וילך לרשומות המתאימות על פי מה שרשום באינדקס.

לכן יצרנו אינדקסים (הקוד בנספח הראשון) שמקצרים את תהליך ביצוע השאילתות.

1. שאלנו שאילתה המחזירה טבלה של שמות המוצרים שהוזמנו לחדרי הכושר בעקבות בקשה של ההנהלה לייעל את הזמנת המוצרים לחדר הכושר.

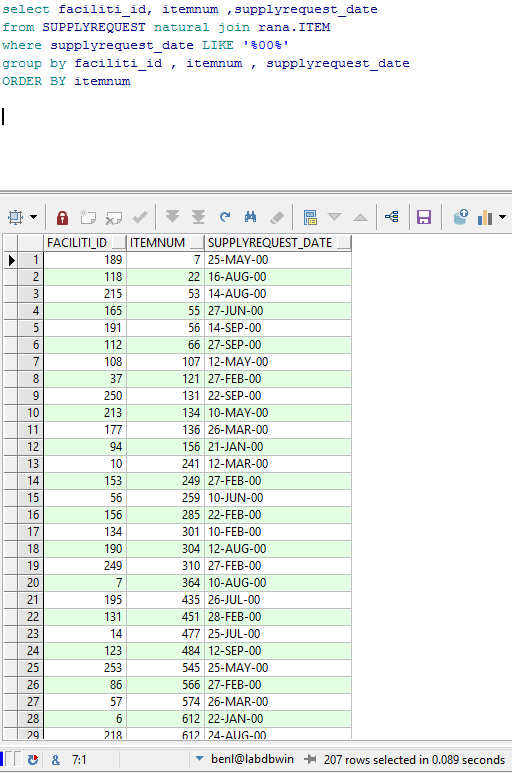
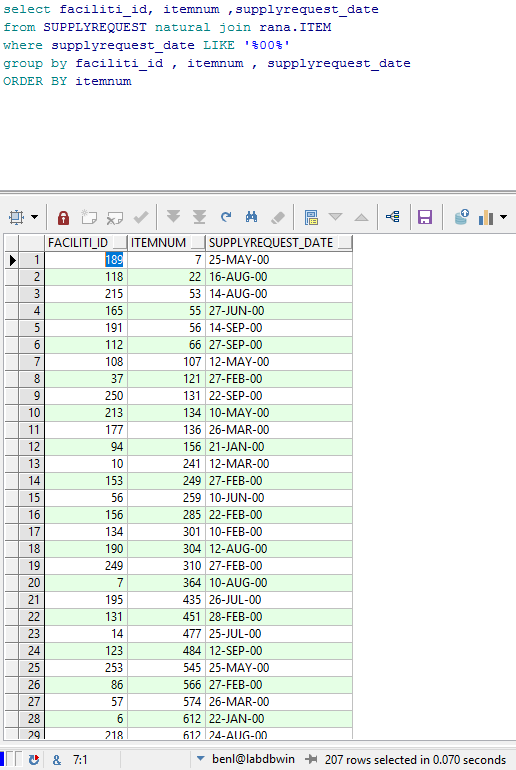
האינדקס שיצרנו היה על המתקן 'gym' ולכן כשחיפשנו בשאילתה את שמות המוצרים שהוזמנו בחדר הכושר היה מהיר יותר לקבל את התוצאה כי ההסתברות לקבל שם של מתקן היא שישית.



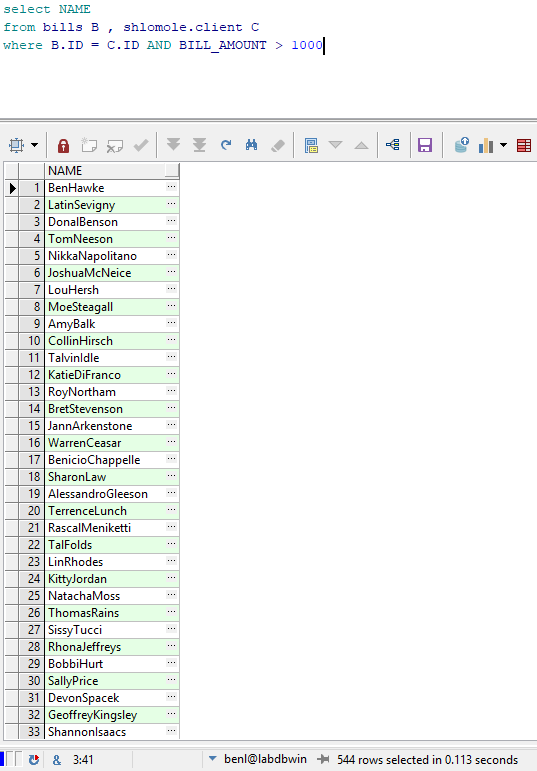


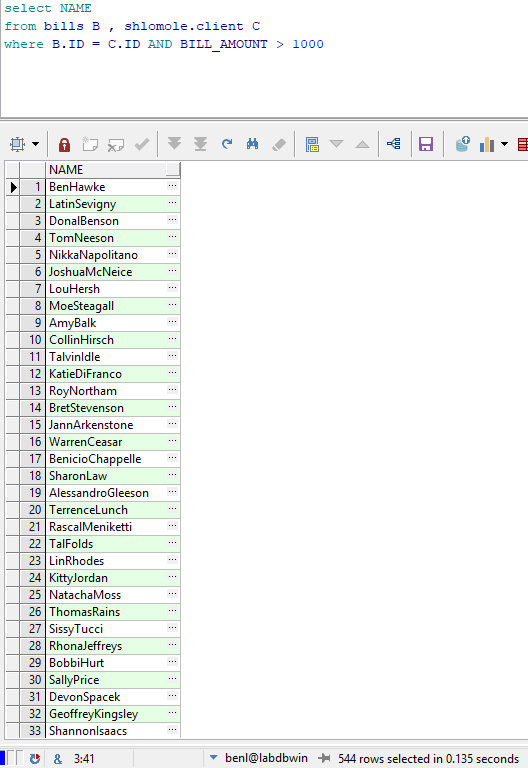
1. שאלנו שאילתה המחזירה טבלה של כל הזמנות המוצרים שנעשו עבור המתקנים השונים בשנת 2000, ממוינים על פי מזהה המוצר.

האינדקס שיצרנו היה על תאריך בקשת ההזמנה שנוצר על ידי הגרלת תאריכים, לכן מבחינה הסתברותית יש לנו פחות רשומות שחוזרות על עצמן מאשר מזהה שהוא רציף ושונה מרשומה אחת לשנייה ולכן זמן החיפוש היה מהיר יותר.



1. שאלנו שאילתה המחזירה טבלה של כל הלקוחות שהוצאה להם קבלה על סכום הגדול מ- 1000 ₪. האינדקס שיצרנו היה על העמודה bill\_amount שהוא מספר רנדומלי , לכן מבחינה הסתברותית יש לנו פחות רשומות זהות מאשר עמודות אחרות בטבלה ולכן זמן החיפוש היה מעט קצר יותר.



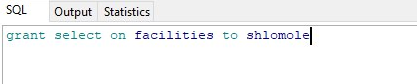


## הרשאות

*פה תכתבו על ההרשאות כולל צילומי מסך אם יש.*

כדי שנוכל לעבוד ככיתה שלמה המפתחת בסיס נתונים כאשר כל קבוצה בונה טבלאות שונות לבסיס הנתונים ואנו צריכים לגשת לטבלאות שלהם כדי לקחת מידע אנו צריכים לקבל מיוצרי הטבלאות הרשאה כדי שנוכל להשתמש בטבלאות שלהם.

במקרה שלנו כמעט ולא היינו צריכים לתת הרשאות לאף אחד מהסיבה הפשוטה שאנחנו היינו די תלויים באחרים מכיוון שבמתקנים השייכים למלונות מסוימים יש התייחסות גם לעובדים גם ללקוחות וגם למוצרים.

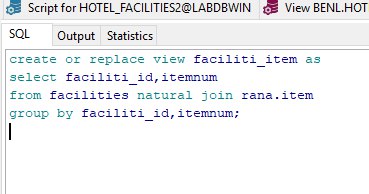


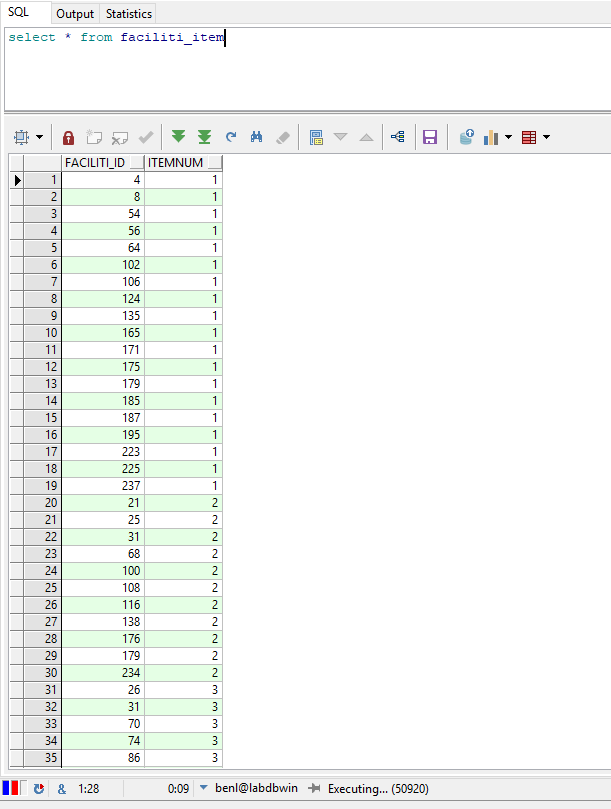
## Views

*תסבירו על הוויו. מה זה. מה עשיתם וצילומי מסך.*

VIEWS הם טבלאות וירטואליות. VIEWS מכילים הגדרות של עמודות וסוגי מידע שאותן עמודות יכולות להכיל. ההבדל בין הטבלאות לבין ה- VIEWS הוא שבטבלאות נשמרים נתונים באופן פיזי ואילו ב- VIEWS הנתונים לא נשמרים באופן פיזי בתוכם אלא הם רק מציגים נתונים הנשמרים בטבלאות. לכן לא ניתן לעדכן או להוסיף נתונים ל- VIEWS כפי שעושים לטבלאות.

יצרנו view עם עמודות facility\_id וitemnum כיוון שרבות מהשאילתות שלנו מתייחסות לעמודות אלו ולכן יעיל יותר לעבוד עם שתי עמודות אלו באופן נפרד, בלי התייחסות לטבלאות המלאות. (הקוד עצמו נמצא בנספח השלישי)





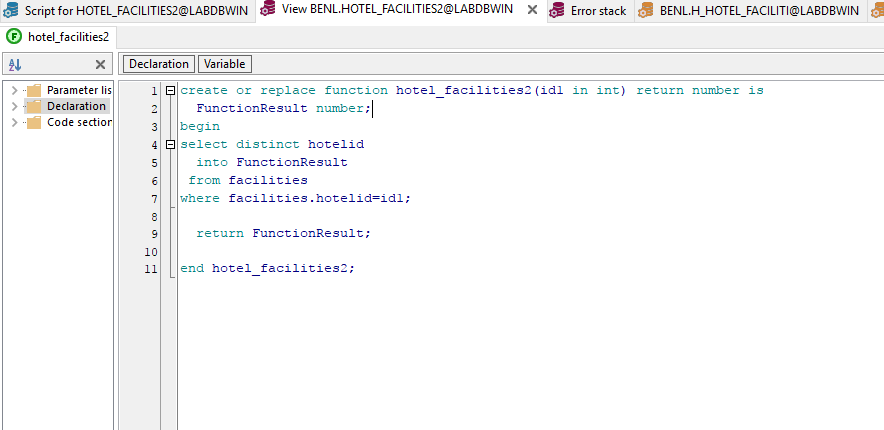
## פונקציות

*פה תסבירו על הפונצקיות. מה כל אחת עושה וצילומי מסך של תוצאות.*

פונקציה בשפת SQL  היא צורה מיוחדת של פקודה אשר מבצעת פעולות שונות על הנתונים בבסיס הנתונים.

בשאילתה 9 ההנהלה המסודרת של המלון החליטה לתת לכל מלון מספר מזהה חדש, בעקבות כך, נוצרו מספרים מזהים למלונות שזהים למספרים המזהים של מתקנים מסוימים, בשם הסדר של ההנהלה הוחלט ליצור פונקציה שבה על פי מספר שיוכנס על ידי המנהל אם הוא קיים גם במלונות וגם במתקנים יוחזר אותו id ואם לא תצא הודעת שגיאה.

(הקוד נמצא בנספח הרביעי)



# 



# נספחים

## נספח ראשון: שאילתות ואינדקסים

(האינדקסים מודגשים)

1. select faciliti\_name , BILL\_AMOUNT

from facilities F , bills C

where F.FACILITI\_ID = C.FACILITI\_ID AND BILL\_AMOUNT>1000

**create index IDX\_BILL\_AMOUNT ON BILLS(BILL\_AMOUNT)**

1. select NAME

from bills B , shlomole.client C

where B.ID = C.ID AND BILL\_AMOUNT > 1000

1. select FACILITI\_ID ,SUM( BILL\_AMOUNT )

from bills B

GROUP BY FACILITI\_ID

having SUM ( BILL\_AMOUNT )> 70000

1. select item

from facilities F , supplyrequest S , rana.item I

where F.FACILITI\_ID = S.FACILITI\_ID AND S.ITEMNUM = I.ITEMNUM AND FACILITI\_NAME = 'gym'

**create index IDX\_FACILITI\_NAME\_3 ON FACILITIES( 'gym' )**

1. select count (\*)

from shlomole.client C , bills B , facilities f

where C.ID = B.ID AND F.FACILITI\_ID = B.FACILITI\_ID AND FACILITI\_NAME = 'sauna'

1. select F.faciliti\_id , faciliti\_name

from facilities F , supplyrequest S , rana.item I

where F.FACILITI\_ID = S.FACILITI\_ID AND I.ITEMNUM = S.ITEMNUM AND ITEM = 'Pomegranates'

1. select faciliti\_id,itemnum,count(\*) "NUMOFREQUEST"

from (supplyrequest natural join rana.item ) natural join facilities

group by itemnum , faciliti\_id

order by itemnum

1. select faciliti\_id, itemnum ,supplyrequest\_date

from SUPPLYREQUEST natural join rana.ITEM

where supplyrequest\_date LIKE '%00%'

ORDER BY Decnding supplyrequest\_date

**create index IDX\_SUPPLYREQUEST\_DATE ON SUPPLYREQUEST( SUPPLYREQUEST\_DATE )**

1. select total\_bills\_for\_faciliti.hotelid, sum(total\_bills\_for\_faciliti.total\_sum\_for\_faciliti) total\_sum\_for\_hotel

from (

select sum(bill\_amount) total\_sum\_for\_faciliti, faciliti\_name, hotelid

from Facilities

inner join bills on Facilities.faciliti\_id = bills.faciliti\_id

group by faciliti\_name, hotelid

having hotelid < 6

) total\_bills\_for\_faciliti

group by total\_bills\_for\_faciliti.hotelid

## נספח שני: עדכון,מחיקה

================= update ===============

---- 1 ----

update Facilities

SET Facilities.occupancy = 100

WHERE Facilities.occupancy < 100;

---- 2 ----

update bills

SET bills.bill\_amount = bills.bill\_amount \* 0.9

WHERE bills.bill\_amount > 999;

============ delete ============

--- 1 -----

delete from SupplyRequst

WHERE SupplyRequst.itemnum = 125;

---- 2 ----

delete from SupplyRequst

WHERE SupplyRequst.supplyrequest\_date = "10-APR-00";

## נספח שלישי: views

create or replace view faciliti\_item as

select faciliti\_id,itemnum

from facilities natural join rana.item

group by faciliti\_id,itemnum;

## נספח רביעי: פונקציות

create or replace function hotel\_facilities2(id1 in int) return number is

FunctionResult number;

begin

select distinct hotelid

into FunctionResult

from facilities

where facilities.hotelid=id1;

return FunctionResult;

end hotel\_facilities2;