**בס"ד**

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

תוכן עניינים

[הפרויקט שלנו 1](#_Toc517367492)

[תרשים ERD 1](#_Toc517367493)

[תיאור הישויות והקשרים 2](#_Toc517367494)

[ישויות 2](#_Toc517367495)

[קשרים 3](#_Toc517367496)

[תרשים DSD 4](#_Toc517367498)

[יצירת הטבלאות 4](#_Toc517367499)

[הכנסת נתונים 6](#_Toc517367500)

[שאילתות SQL 7](#_Toc517367501)

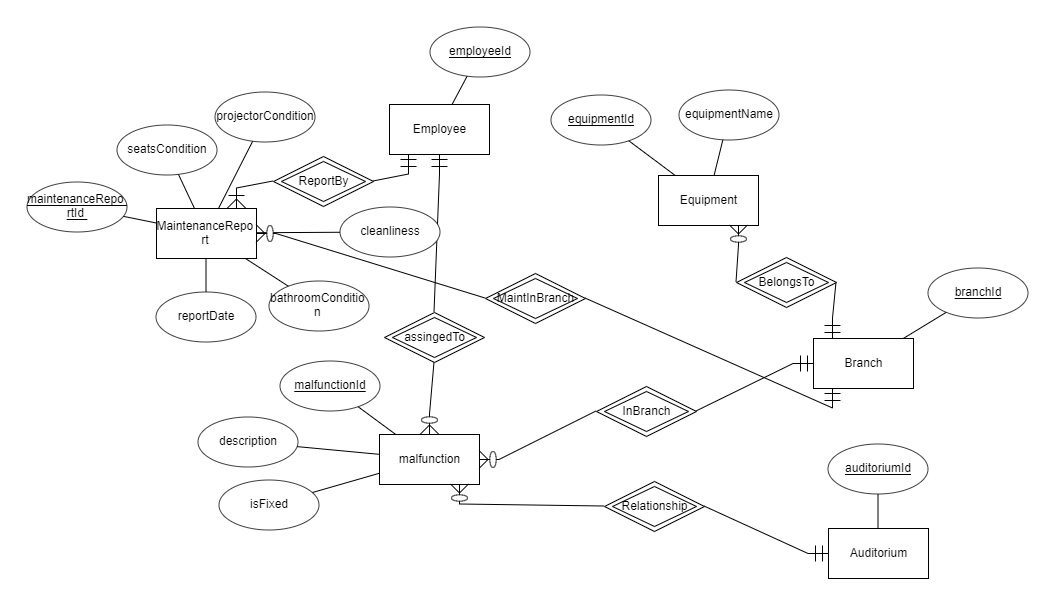
# הפרויקט שלנו

## תרשים ERD

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)

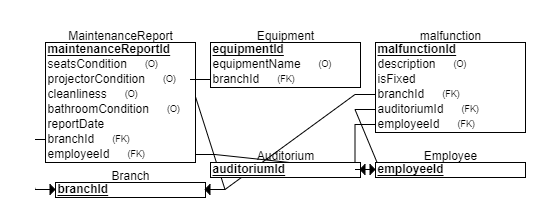
### קשרים

* 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.

## 

## 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),

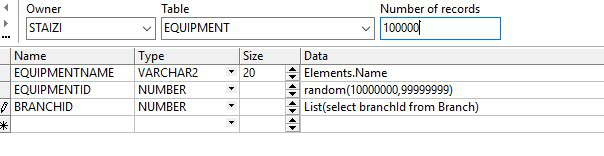
FOREIGN KEY (employeeId) REFERENCES Employee(employeeId)

);

## הכנסת נתונים

על מנת לאכלס את הטבלאות שיצרנו בנתונים, השתמשנו ב Data Generator של PL/SQL.

לדוגמה כך הכנסנו נתונים לטבלה :Equipment



## שאילתות SQL

--1.return how many types of equipment are in branches that have more than 5 malfunctions

select branchId, count(equipmentName)

from malfunction natural join equipment

group by branchId

having count(\*)>5

-------------------------------------------------------------

--2.return id of employees that have more than 1 malfuntions that they didn't fix yet

select distinct employeeId

from (select \* from malfunction natural join employee) l

where 1< (select count(\*)

from malfunction m

where m.employeeid=l.employeeId and isFixed='NO')

-------------------------------------------------------------

--3.return how many malfunctions the employee with the id 57725998 has fixed

select count(\*)

from malfunction m

where m.employeeid=57725998 and isFixed='YES'

-------------------------------------------------------------

--4.return id of all employees that filled a miantenance report and how many maintenance reports they filled

select employeeId, count (\*)

from maintenancereport natural join employee

group by employeeId

-------------------------------------------------------------

--5.return how many maintenance reports the employee with the id 76452032 has filled

select count(\*)

from maintenancereport

where employeeId=76452032

-------------------------------------------------------------

--6.return the id and the count of malfunctios in branches that have at least one malfunction

select branchId, count(\*)

from malfunction natural join branch

group by branchId

-------------------------------------------------------------

--7.return the id and the count of malfunctios in branches that have more than 5 pieces of equipment and at least one malfunction

select branchId, count(\*)

from (select \* from malfunction natural join branch) m

group by branchId

having 5 < (select count(\*)

from equipment e

where m.branchid=e.branchid)

-------------------------------------------------------------

--

8.return the number of malfunctions and the number of maintenance reports of every branch

select b.branchId,

(select count(\*) from malfunction m where m.branchid=b.branchid) as nunOfMalfunctions,

(select count(\*) from maintenancereport m where m.branchid=b.branchid) as numOfMaintenanceReports

from branch b

--------------------------------------------------------------

--9. return the id od employees that handled more than 5 maintenance reports and malfunctions in total, and the number of reports and malfunctions they have handled

select distinct employeeId,

(select count(\*) from Maintenancereport m1 where e.employeeId=m1.employeeId) as numOfReports,

(select count(\*) from malfunction m2 where e.employeeId=m2.employeeId) as numOfMalfunctions

from employee e

where ((select count(\*) from Maintenancereport m1 where e.employeeId=m1.employeeId)+

(select count(\*) from malfunction m2 where e.employeeId=m2.employeeId)) > 5

--------------------------------------------------------------

--10.return the id of branches that have more 10 pieces of equipment or that had more than 5 malfunctions

select branchId

from Branch b1

where 10 < (select count(\*) from equipment e2 where b1.branchId=e2.branchId)

union

select branchId

from Branch b2

where 5 < (select count(\*) from malfunction e2 where b2.branchId=e2.branchId)