בס"ד

מיני פרויקט בבסיסי נתונים

שלום תעיזי ודוד דגן

תוכן עניינים

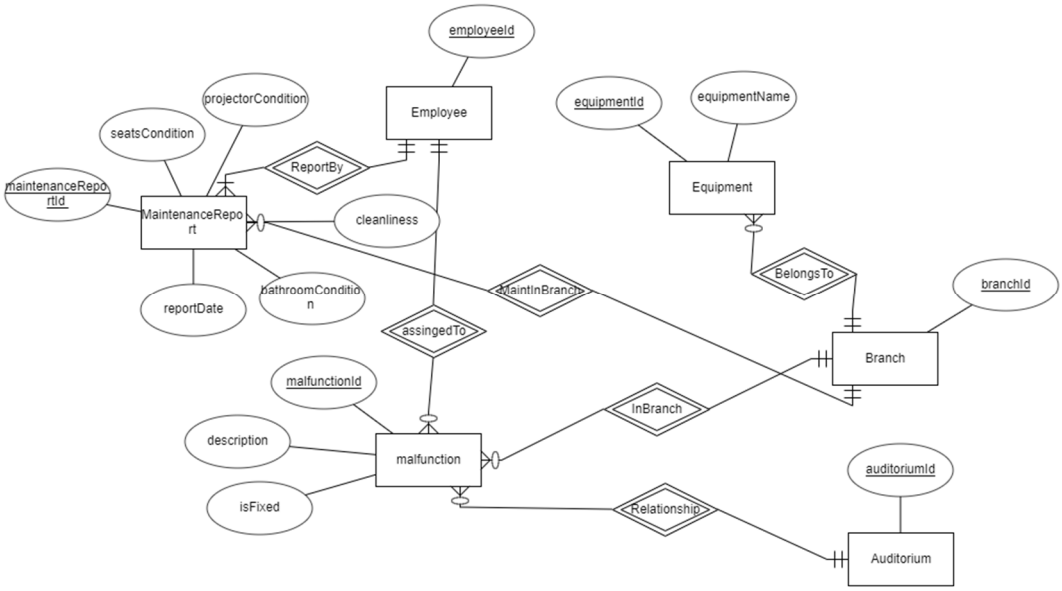
הפרויקט שלנו

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.

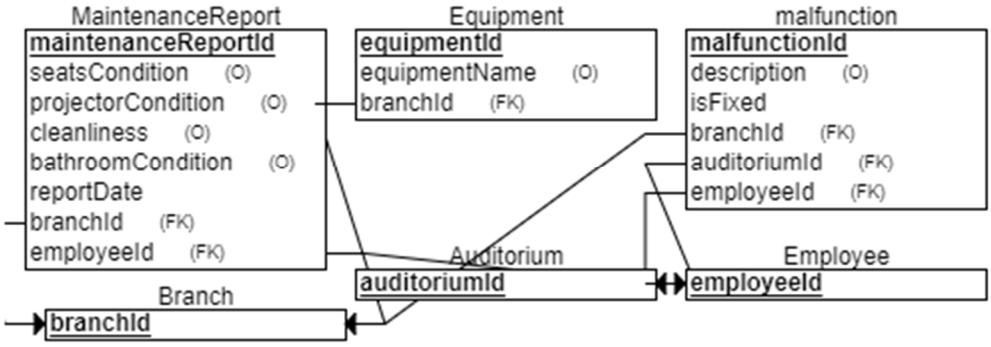
- MaintInBranch – הקשר בין MaintenanceReport לבין Branch. הקשר הינו חלש, כיוון שמקשר בין ישות חלשה לחזקה. הקשר הוא M:1 משום שיכול להיות כמה דוחות תחזוקה לסניף אך לא כמה סניפים בדוח תחזוקה אחד

- 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:

Owner	Table	Number of records
STAIZI	EQUIPMENT	100000

Name	Type	Size	Data
EQUIPMENTNAME	VARCHAR2	20	Elements.Name
EQUIPMENTID	NUMBER		random(10000000,99999999)
BRANCHID	NUMBER		List(select branchId from Branch)

שאילתות SQL

1.return how many types of equipment are in branches that have more than 5 malfunctions
מחזיר כמה סוגים של ציוד יש בסניפים שיש בהם יותר מחמש תקלות

```
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
מחזיר מספרי זיהוי של עובדים שיש להם יותר מתקלה אחת שלא טיפלו בה עדיין

```
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
מחזיר כמה תקלות תיקן העובד שהמספר זיהוי שלו הוא 57725998

```
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
מחזיר מספרי זיהוי של עובדים שמילאו דוחות תחזוקה וכמה הם מילאו

```
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
מחזיר כמה דוחות תחזוקה מילא העובד שמספר הזיהוי שלו הוא 76452032

```
select count(*)
from maintenancereport
where employeeId=76452032
```

6.return the id and the count of malfunctios in branches that have at least one malfunction
מחזיר את מספרי הזיהוי ואת כמות התקלות של סניפים שיש בהם לפחות תקלה אחת

```
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

מחזיר את מספרי הזיהוי ואת כמות התקלות של סניפים שיש להם יותר מחמש יחידות של ציוד וגם שיש בהם לפחות תקלה אחת

```
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

מחזיר את מספר התקלות ואת מספר דוחות התחזוקה של כל סניף

```
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

מחזיר את מספר הזיהוי של עובדים שטיפלו ביותר מחמש תקלות ודוחות תחזוקה ביחד, ומחזיר בכמה תקלות ובכמה דוחות תחזוקה הם טיפלו

```
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

מחזיר את מספרי הזיהוי של סניפים שיש להם יותר מעשר יחידות ציוד או שהיו להם יותר מחמש תקלות

```
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)
```

אינדקסים

1. יצרנו אינדקס על העמודה branchId בטבלה Equipment
יצירת האינדקס: create index branch_equipment_index on equipment(branchId)
דוגמא לשאילתה:

מחזיר את מספרי הזיהוי של סניפים שיש להם יותר מעשר יחידות ציוד או שהיו להם יותר מחמש תקלות, לבקשת הנהלת הרשת לדעת באילו סניפים יש סיכויים גדולים לתקלות.

```
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)
```

זמן ריצה לפני הוספת האינדקס: 127.097 שניות
זמן ריצה אחרי הוספת האינדקס: 21.614 שניות

הסבר - יש שיפור משמעותי בזמן ריצה מכיוון בשאילתה אנחנו עוברים בשביל כל מספר סניף על כל הטבלה של הציוד ומחפשים ציוד עם אותו מספר סניף לכן זה עזר שעשינו אינדקס לטבלה של הציוד לפי המספר סניף.
קיימים 110000 פריטים של ציוד ו-20000 סניפים, בכל סניף יש בממוצע 5.524 פריטים של ציוד, ולכן יצירת אינדקס בטבלה של הציוד על המספר סניף משפר את הזמן ריצה של השאילתה.

2. יצרנו אינדקס על העמודה branchId בטבלה Malfunction
יצירת האינדקס: create index branch_malfunction_index on malfunction(branchId)
דוגמא לשאילתה:

מחזיר את מספרי הזיהוי של סניפים שיש להם יותר מעשר יחידות ציוד או שהיו להם יותר מחמש תקלות, לבקשת הנהלת הרשת לדעת באילו סניפים יש סיכויים גדולים לתקלות.

```
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)
```

זמן ריצה לפני הוספת האינדקס: 21.614 שניות
זמן ריצה אחרי הוספת האינדקס: 2 שניות

הסבר - יש שיפור משמעותי בזמן ריצה מכיוון בשאילתה אנחנו עוברים בשביל כל מספר סניף על כל הטבלה של התקלות ומחפשים תקלה עם אותו מספר סניף לכן זה עזר שעשינו אינדקס לטבלה של התקלות לפי המספר סניף.
קיימים 20000 תקלות ו-20000 סניפים, בכל סניף יש בממוצע 1.58 תקלות, ולכן יצירת אינדקס בטבלה של התקלות על המספר סניף משפר את הזמן ריצה של השאילתה.

3. יצרנו אינדקס על העמודה employeeId בטבלה Malfunction
יצירת האינדקס:
create index employee_malfunction_index on malfunction(employeeId)
דוגמא לשאילתה:

השאילתה מחזירה את מספר הזיהוי של עובדים שטיפלו ביותר מחמש תקלות ודוחות תחזוקה ביחד, ומחזיר בכמה תקלות ובכמה דוחות תחזוקה הם טיפלו, לבקשת הנהלת הרשת במטרה לצ'פר עובדים חרוצים.

```
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
```

זמן ריצה לפני הוספת האינדקס: 55.077 שניות
זמן ריצה אחרי הוספת האינדקס: 27.868 שניות

הסבר - יש שיפור משמעותי בזמן ריצה מכיוון בשאילתה אנחנו עוברים בשביל כל מספר עובד על כל הטבלה של התקלות ומחפשים תקלה עם אותו מספר עובד לכן זה עזר שעשינו אינדקס לטבלה של התקלות לפי המספר עובד.
קיימים 20000 תקלות ו-20000 עובדים, כל עובד טיפל בממוצע ב 1.45 תקלות, ולכן יצירת אינדקס בטבלה של התקלות על המספר עובד משפר את הזמן ריצה של השאילתה.

פונקציות

1. יצרנו פונקציה שמקבלת כפרמטר שם של עיר ומחזירה כמה פריטים של ציוד קיימים בכל הסניפים בעיר ביחד.

יצירת הפונקציה:

```
create or replace function equipment_in_city(city_name in string)
return number is
num_of_equipment number;
begin
select count(*)
into num_of_equipment
from equipment e natural join branch b natural join city c
where c.cityname=city_name;

return(num_of_equipment);
end equipment_in_city;
```

2. יצרנו פונקציה שמקבלת כפרמטר שם של עיר ומחזירה כמה עובדים בסך הכל טיפלו בתקלות בסניפים שנמצאים באותה העיר.

יצירת הפונקציה:

```
create or replace function employees_in_city(city_name in string)
return number is
num_of_employees number;
begin
select count(*)
into num_of_employees
from (select distinct m.employeeid
from malfunction m natural join branch b natural join city c
where c.cityname=city_name) t;

return(num_of_employees);
end employees_in_city;
```

פרוצדורות

1. יצרנו פרוצדורה שמקבלת מספר סניף ומעדכנת בבסיס נתונים שכל התקלות באותו סניף טופלו (isFixed='YES').

יצירת הפרוצדורה:

```
create or replace procedure fixed_branch_malfunctions(branch_id number)
is
begin
update malfunction set
isfixed='YES'
where branchid=branch_id;
end fixed_branch_malfunctions;
```

2. יצרנו פרוצדורה שמקבלת מספר תקלה ומספר עובד ומעדכנת בבסיס הנתונים שהתקלה עברה לטיפולו של אותו עובד

יצירת הפרוצדורה:

```
create or replace procedure reassign_malfunction(maulfunction_id number,
employee_id number)
is
begin
update malfunction
set employeeid=employee_id
where malfunctioid=malfunction_id

end reassign_malfunction;
```

עוד שאילתות SQL

אחרי שאיחדנו את שני הפרויקטים כתבנו עוד שאילתות שמשתמשות בטבלאות משני הפרויקטים

1. לבקשת הנהלת הרשת שמעוניינים לדעת באיזה ערים יש יותר תקלות מדוחות תחזוקה בשביל להבין מה מצב התחזוקה באותם, ערים כתבנו שאילתה שמחזירה את כל הערים שבהן יש יותר תקלות מדוחות תחזוקה

```
select distinct cityname
from malfunction m natural join city c
where (select count(*) from malfunction m1 natural join city c1 where
c1.cityid=c.cityid)
    > (select count(*) from maintenancereport m2 natural join city c2 where
c2.cityid=c.cityid)
```

2. לבקשת הנהלת הרשת שמעוניינים לדעת באיזה ערים יש יותר תקלות שלא טופלו עדיין כתבנו שאילתה שמחזירה רשימה של כל הערים שקיימים בהן סניפים ומה הסכום של כל התקלות שלא טופלו עדיין בסניפים באותה העיר, והרשימה ממוינת לפי מספר התקלות באותה העיר (העיר שמופיעה ראשונה זאת העיר עם הכי הרבה תקלות שלא טופלו עדיין)

```
select cityname, count(malfunctionid) as malfunctions_to_fix
from malfunction natural join branch natural join city
where isfixed='NO'
group by cityname
order by count(malfunctionid) desc
```

3. לבקשת הנהלת הרשת שמעוניינים לדעת באיזה ערים יש יותר מ-50 פריטי ציוד, כתבנו שאילה שמשתמשת בפונקציה equipment_in_city ומחזירה רשימה של כל הערים שיש בהם יותר מ-50 פריטים של ציוד

```
select distinct cityname
from equipment natural join branch natural join city
where 50<staizi.equipment_in_city(cityname)
```

Views

1. יצרנו view עם המידע של כל דוחות התחזוקה בעיר Sean זה מיועד למנהלי חברת התחזוקה באותה עיר שלא אמורים לראות דוחות תחזוקה של סניפים שלא נמצאים בעיר שלהם

```
create view maintenancereport_sean_view as
select *
from maintenancereport m natural join branch b natural join city c
where c.cityname='Sean'
```

2. יצרנו view עם המידע של כל דוחות התחזוקה בעיר Rod זה מיועד למנהלי חברת התחזוקה באותה עיר שלא אמורים לראות דוחות תחזוקה של סניפים שלא נמצאים בעיר שלהם

```
create view maintenancereport_rod_view as
select *
from maintenancereport m natural join branch b natural join city c
where c.cityname='Rod'
```

3. יצרנו view עם מידע שרלוונטי למי שצריך את הכתובות של הסניפים שנמצאים ביפן בלי מידע מיותר כמו מספר סניף וכדומה

```
create view branchaddressview as
select b.branchname, b.branchaddress, c.cityname, a.areacountry
from branch b natural join city c natural join area a
where a.areacountry='Japan'
```

4. יצרנו view עם מידע שרלוונטי למי שצריך את הכתובות של הסניפים שנמצאים בארצות הברית בלי מידע מיותר כמו מספר סניף וכדומה

```
create view branchaddressview as
select b.branchname, b.branchaddress, c.cityname, a.areacountry
from branch b natural join city c natural join area a
where a.areacountry='USA'
```

5. יצרנו view עם המידע של כל התקלות בעיר Sean זה מיועד למנהלי חברת התחזוקה באותה עיר שלא אמורים לראות תקלות שלא נמצאות בעיר שלהם

```
create view malfunction_sean_view as
select *
from malfunction m natural join branch b natural join city c
where c.cityname='Sean'
```

6. יצרנו view עם המידע של כל התקלות בעיר Rod זה מיועד למנהלי חברת התחזוקה באותה עיר שלא אמורים לראות תקלות שלא נמצאות בעיר שלהם

```
create view malfunction_Rod_view as
select *
from malfunction m natural join branch b natural join city c
where c.cityname='Rod'
```