דו"ח פרויקט בבסיס נתונים

מגישים: מאיר קלמפנר ונתנאל בשן

תאריך: 09/06/2022

גרסה: 2.0

1. תוכן עניינים

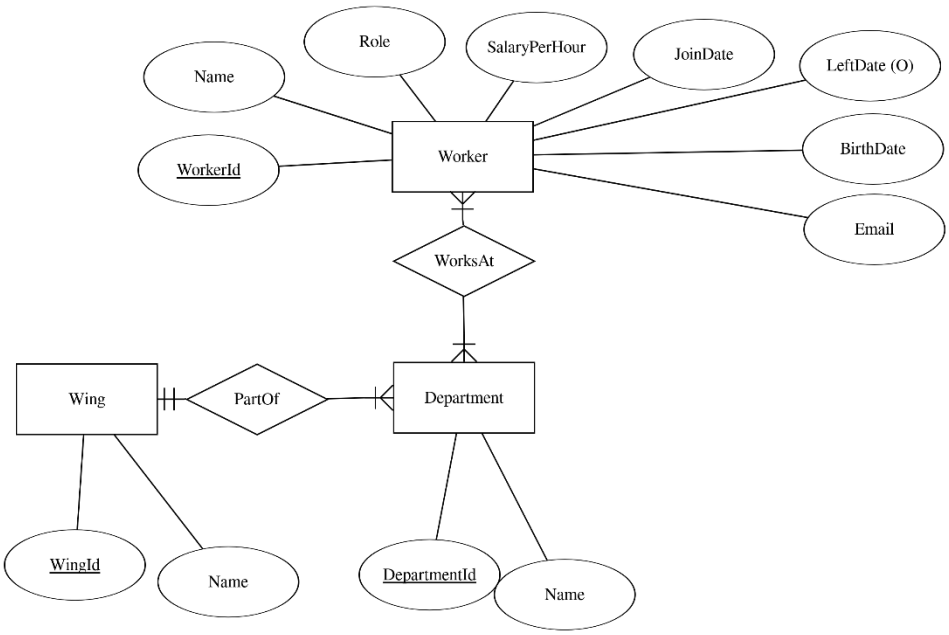
2. היכרות עם הכלים

על מנת להכיר את הכלים שבהם נשתמש בשביל ליצור את הטבלאות, השתמשנו בכלים האלו עם ישויות בסיסיות.

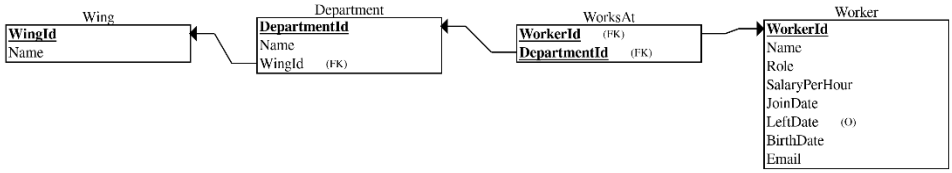
הישויות הן:

1. סוכן – Agent:
   - מספר סוכן (מספר רץ)
   - שם סוכן
   - שכר
   - מספר בוס
   - דרוג
   - שנת שכירה
2. לוח זמנים – Schedule:
   - מספר סוכן
   - מספר לקוח
   - זמן פגישה
3. לקוח – Client:
   - מספר לקוח (מספר רץ)
   - מספר פלאפון
   - כתובת
   - מספר עיר
   - מספר סוכן
4. עיר – City:
   - מספר עיר (מספר רץ)
   - שם עיר
   - מספר אזור
5. אזור – Area:
   - מספר אזור (מספר רץ)
   - שם אזור

להלן גרף הERD שהכנו:



להלן הגרף הרציונלי שנוצר מהגרף הקודם, כאן רואים את הטבלאות והמפתחות:



את הקוד שנוצר בשביל ליצור את הטבלאות אפשר לראות בנספח.

3. אפיון

הנושא שנבחר בביתה הינו ניהול בסיס נתונים של מוזיאון. והישויות שבחרנו הן: אגף, מחלקה ועובד.

החלטנו שיש היררכיה בין הישות מחלקה לבין אגף, כך שלכל אגף יש כמה מחלקות, בעצם קשר יחיד לרבים.

בנוסף החלטנו שלכל עובד יש כמה מחלקות בהן הוא עובד ולכן יש קשר רבים לרבים בין עובד לבין מחלקה.

להלן התכונות של הישויות:

אגף - Wing:

- מספר אגף (מספר רץ)
- שם האגף

מחלקה - Department:

- מספר מחלקה (מספר רץ)
- שם המחלקה
- מספר האגף (מפתח של המחלקה אגף)

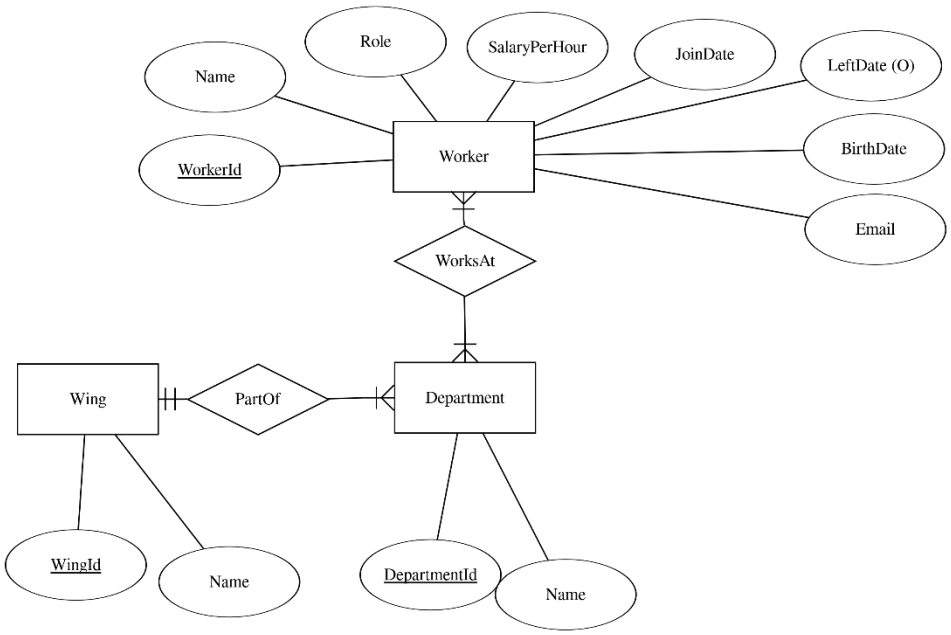
עובד - Worker:

- מספר עובד (מספר רץ)
- שם העובד
- תפקיד
- שכר לשעה
- תאריך הצטרפות
- תאריך עזיבה (יכול להיות NULL במקרה ועדיין עובד)
- תאריך לידה
- אימייל

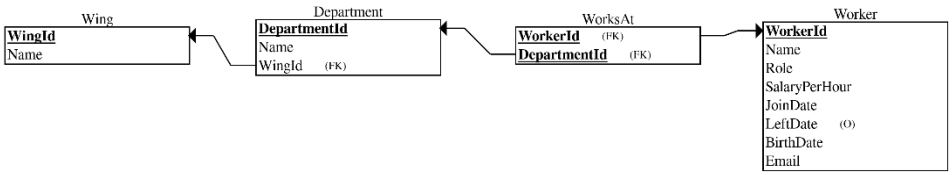
עובד במחלקה - WorksAt:

- מספר עובד (מפתח של המחלקה עובד)
- מספר מחלקה (מפתח של המחלקה אגף)

להלן גרף הERD שהכנו:



להלן הגרף הרציונלי שנוצר מהגרף הקודם, כאן רואים את הטבלאות והמפתחות:



את הקוד שנוצר בשביל ליצור את הטבלאות אפשר לראות בנספח.

4. אכלוס טבלאות

השתמשנו בpython בשביל להכין את הscripts וגם השתמשנו בספריות random וfaker.

הסקריפטים שלנו יצרו קובץ csv כיוון שרצינו להכניס הרבה ישויות והדרך המהירה והפשוטה ביותר שמצאנו זה באמצעות csv.

מספר הנתונים שנוצרו:

- עובדים: 20,911
- מחלקות: 1,002
- אגפים: 400
- עובדים במחלקות: 31,291

בעיות:

- היינו צריכים לגרום לכך שהעובדים גדולים מגיל 18.
- היינו צריכים לבדוק שהעובד התחיל לעבוד אחרי גיל 21 וסיים לעבוד לפני גיל 70.
- היינו צריכים לגרום שהעובד עזב אחרי שהצטרף לעבודה.
- היינו צריכים להזין את הנתונים של המחלקות לאחר שהכנסנו את הנתונים של האגפים, כיוון שיש בטבלה של המחלקות מפתח של האגפים.
- היינו צריכים להזין את הנתונים של העובדים במחלקות לאחר שהכנסנו את הנתונים של העובדים ושל המחלקות, כיוון שיש בטבלה של העובדים במחלקות מפתחות של העובדים ושל המחלקות.

את הקוד שנוצר בשביל ליצור את הטבלאות אפשר לראות בנספח.

5. שאילתות

5.1 שאילתות טריוויאליות

1. כל העובדים שעובדים הרבה זמן (30 שנה ומעלה) ומקבלים שכר נמוך (פחות מ100 לשעה).

שימוש: משאבי אנוש החליטו שהעובדים שעובדים כבר מעל 30 שנה הם נכס חשוב למוזיאון, לכן הם התחילו תהליך בעזרת הנהלת החשבונות להעלות את השכר של כל העובדים האלה שיהיה לפחות 100 לשעה.

התכונות החוזרות: מספר העובד, שם העובד, שכר לשעה, תאריך הצטרפות ואימייל.

מספר השורות: 113.

זמן הריצה: 0.036 שניות.

2. ממוצע ההוצאות החודשיות על שכר העובדים.

שימוש: הנהלת חשבונות רוצים לדעת את ההוצאות החודשיות של שכר העובדים בחודש ממוצע (20 ימים בחודש עם 8 שעות בכל יום). על מנת לעשות את זה, הם צריכים את השאילתה הזאת.

התכונות החוזרות: מספר העובד, שם העובד, שכר לשעה, תאריך הצטרפות ואימייל.

מספר השורות: 1.

זמן הריצה: 0.019 שניות.

3. תפקידים של עובדים ומספר העובדים שעובדים בהם (גם ברגע הנתון וגם לכל אורך השנים)

שימוש: משאבי אנוש יוכלו כך לראות את התפלגות התפקידים של העובדים כרגע ולכל אורך השנים, וכך לדעת באיזה תפקיד יש יותר מדי עובדים ובאיזה תפקיד אין מספיק.

התכונות החוזרות: תפקיד, מספר העובדים לכל אורך השנים, מספר העובדים כרגע.

מספר השורות: 9.

זמן ריצה: 0.171 שניות.

4. מספר העובדים שמקבלים שכר (מעוגל ל10 הקרוב)

שימוש: הנהלת החשבונות יוכלו להשתמש בשאילתה הזאת על מנת לבדוק מה ההתפלגות של שכר העובדים במוזיאון.

התכונות החוזרות: שכר, מספר העובדים.

מספר השורות: 10.

זמן ריצה: 0.023

5.2 שאילתות לא טריוויאליות

5. מספר העובדים שעובדים כרגע בכל מחלקה

שימוש: נתון למשאבי אנוש שיש מחלקות בהן יש יותר מדי עובדים ומחלקות שבהן חסר

עובדים. לכן על מנת שמשאבי אנוש יוכלו כך לבדוק באיזה מחלקות יש יותר מדי עובדים ובאיזה מחלקות חסר עובדים, הם יוכלו להשתמש בשאילתה הזאת.
התכונות החוזרות: מספר מחלקה, שם מחלקה ומספר העובדים במחלקה.
מספר השורות: 953.
זמן ריצה: 0.36 שניות.

6. מספר המחלקות שכל עובד עובד בהן
שימוש: על מנת שמשאבי אנוש יוכלו לחלק את העובדים בין המחלקות, משאבי אנוש צריכים לדעת מי העובדים שעובדים קשה ומי העובדים שכמעט ואין להם עבודה. בעזרת השאילתה הזאת הם יכולים לדעת את הנתונים האלו.
התכונות החוזרות: מספר עובד, שם עובד, שכר לשעה, תאריך הצטרפות, אימייל ומספר המחלקות שהעובד עובד בהן.
מספר השורות: 1893.
זמן ריצה: 0.884 שניות.

7. עובדים שיש להם את אותו יום ההולדת ממוין לפי יום ההולדת.
שימוש: ההנהלה החליטה במקום לחגוג לכל עובד יום הולדת בפני עצמו, לחגוג לכל העובדים באותו שחוגגים באותו היום מסיבת יום הולדת אחת וכך לחסוך כספים.
התכונות החוזרות: מספר עובד, שם עובד, אימייל, תאריך לידה, תאריך הצטרפות ותאריך עזיבה.
מספר השורות: 20911.
זמן ריצה: 9.448 שניות.

8. כל המחלקות שאין להן מנהל.
שימוש: משאבי אנוש רוצים שלכל מחלקה יהיה לפחות מנהל אחד, ועל מנת לעשות זאת הם צריכים שאילתה שתציג להם את המחלקות שלהן אין מנהל.
התכונות החוזרות: מספר עובד, שם עובד, אימייל, תאריך לידה, תאריך הצטרפות ותאריך עזיבה.
מספר השורות: 34.
זמן ריצה: 0.073 שניות.

הקודים של כל השאילתות נמצאים בנספח.

6. אינדקסים

1. ימי ההולדת של העובדים.

שאילתה: האינדקס משפר את השאילתה מספר 7.

התפלגות: מספר התאריכים המיוחדים הם 12361 מתוך 20911.

שיפור: חל שיפור כבשניה וחצי.

אחוז השינוי: ירד בכ14 אחוזים מהזמן המקורי.

הסבר למה התקבלו התוצאות: השאילתה עושה הרבה מאוד שימושים בתאריך הלידה וכיוון שהנדקסנו את התכונה הזאת אז זמן החיפוש קטן.

ריצה לפני האינדקס:

20911 rows selected in 9.448 seconds

ריצה אחרי האינדקס:

20911 rows selected in 8.101 seconds

2. זמן סיום העבודה של העובד.

שאילתה: האינדקס משפר את השאילתה מספר 6.

התפלגות: יש 1894 תאריכים שהם Null (עובדים בפועל) לעומת 20,911 שורות בטבלה של העובדים.

	NULLELEMENTS	ALLELEMENTS
1	1894	20911

שיפור: חל שיפור ב0.072 שניות.

אחוז השינוי: ירד בכ8 אחוזים.

הסבר למה התקבלו התוצאות: השאילתה בודקת האם העובד עדיין עובד לפי האם תאריך העזיבה הוא NULL, לכן האינדקס היה צריך לייעל את זמן השאילתה.

ריצה לפני האינדקס:

1893 rows selected in 0.884 seconds

ריצה אחרי האינדקס:

1893 rows selected in 0.812 seconds

3. מספר עובד בטבלה של עובד במחלקה.

שאילתה: האינדקס צריך לשפר את השאילתה מספר 5.

התפלגות: יש 20910 מספרי עובדים שונים בתוך הטבלה עם 31291 שורות.

שיפור: חל שיפור ב0.016 שניות.

אחוז השינוי: ירד בכ5 אחוזים.

הסבר למה התקבלו התוצאות: השאילתה עושה natural join בין הטבלאות של עובד ועובד במחלקה.

ריצה לפני האינדקס:

953 rows selected in 0.327 seconds

ריצה אחרי האינדקס:

953 rows selected in 0.311 seconds

את הקוד של האינדקסים אפשר לראות בנספח.

7. אינטגרציה

נתנו הרשאות select וreference לטבלה Worker לשתי קבוצות.
את הקוד שנותן את הגישה, אפשר לראות בנספח.

קיבלנו הרשאה מהקבוצה של חננאל זגורי ואביחי סוריה לגשת אל הטבלאות שלהם.

7.1 שאילתות האינטגרציה

1. הצגת מחיר לכל מחלקה של כל מוצגי התערוכות שפועלות כרגע.

שימוש: על מנת שמשאבי אנוש יוכלו לחלק בצורה הוגנת את כל מאבטחי המוזיאון בין התערוכות, הם צריכים לדעת כמה יקר כל תערוכה, שאת זה השאילתה נותנת.
התכונות החוזרות: מספר מחלקה, שם מחלקה ומחיר.
מספר השורות: 505.
זמן ריצה: 0.226 שניות.

2. כל המחלקות שאין אצלן תערוכה כרגע.

שימוש: על מנת שהמעצבים יוכלו לפתוח תערוכות חדשות, הם צריכים לדעת באיזה מחלקות אין כרגע תערוכה בשביל לפתוח שם, שאת זה השאילתה נותנת.
התכונות החוזרות: מספר מחלקה ושם מחלקה.
מספר השורות: 991.
זמן ריצה: 0.281 שניות.

3. כל המחלקות שמתקיים אצלן תערוכה כרגע אבל אין שם שומרים.

שימוש: חשוב מאוד שבכל מחלקה שמתקיימת אצלה תערוכה כרגע שיהיה שם לפחות שומר אחד, לכן צריך שאילתה שנותנת את כל המחלקות שבהן אין שומר שעובד כרגע.
התכונות החוזרות: מספר מחלקה ושם מחלקה.
מספר השורות: 365.
זמן ריצה: 0.158 שניות.

הקודים של כל השאילתות נמצאים בנספח.

8. Views

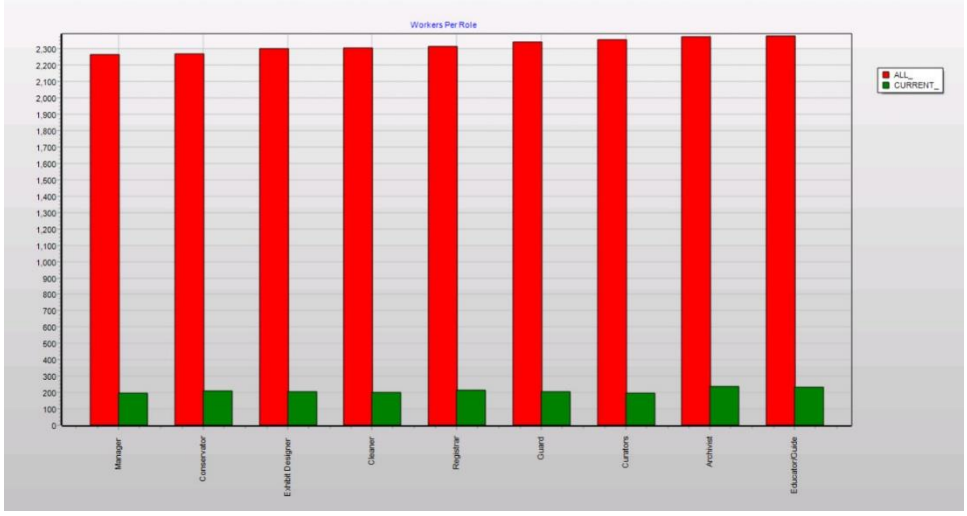
יצרנו 4 תצוגות שונות, שניים בשביל הנהלת חשבונות ושניים בשביל משאבי אנוש.

1. הנהלת חשבונות
   - העובדים מסודרים בסדר יורד לפי השכר לשעה
   הנהלת החשבונות ישתמשו בזה על מנת לבדוק אצל אילו עובדים יורד הכי הרבה כסף בשכר החודשי.
   - בממוצע סך כל ההוצאות על שכר העובדים בחודש
   הנהלת החשבונות ישתמשו בזה על מנת לבדוק את ההוצאה החודשית בממוצע של המוזיאון על העובדים.
2. משאבי אנוש
   - המנהלים שעובדים כרגע
   משאבי אנוש יוכלו להשתמש בזה על מנת לקבל מידע אודות המנהלים שעובדים במוזיאון.
   - העובדים שעובדים לצד המחלקות בהם עובדים
   משאבי אנוש יוכלו להשתמש בזה על מנת לבדוק כל עובד באיזו מחלקה הוא עובד.

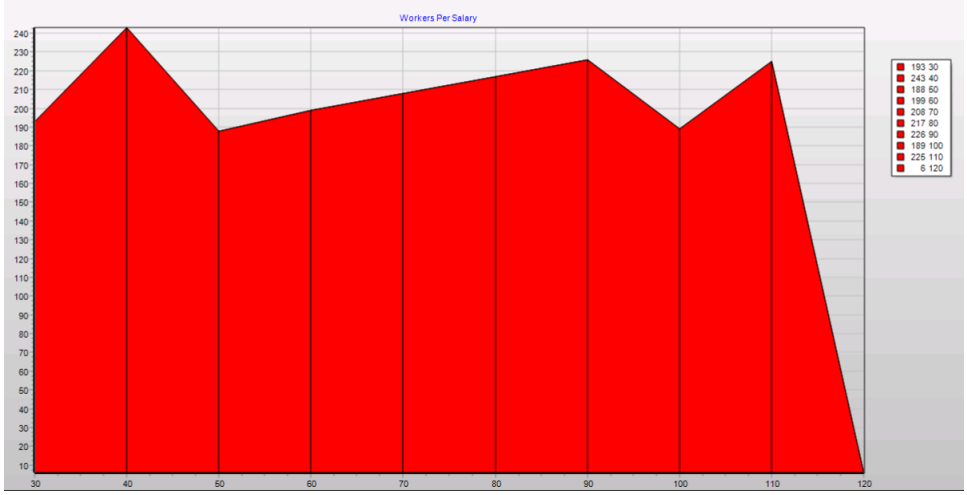
את הקוד של התצוגות אפשר לראות בנספח.

9. גרפים

בגרף הבא רואים בציר X את כל התפקידים שיש לעובדים ובציר Y מספר העובדים בכל תפקיד. בנוסף אפשר לראות מספר העובדים שכרגע עובדים (בצבע הירוק) ביחס למספר העובדים הכללי שעבדו עד עכשיו (בצבע האדום).



בגרף הבא רואים בציר X את המשכורות של העובדים (מעוגל) ובציר Y את מספר העובדים שמקבלים משכורת בתחום הזה.



10. תכנות PL/SQL

10.1 פרוצדורות

1. שינוי שכר לעובדים שעבדו הרבה זמן ומקבלים שכר לשעה נמוך

מטרה: כאמור לפני כן, משאבי אנוש רצו להעלות את השכר לשעה של העובדים שעבדו מעל 30 שנה ומקבלים פחות מ100 לשעה. על מנת לעדכן את השכר, יצרנו את הפרוצדורה הבאה:

```
CREATE OR REPLACE PROCEDURE SetSalaryToOldWorkers(yearsWorking INT, salaryPH FLOAT, salaryPHLessThan FLOAT, updatedRows OUT INT)
AS
BEGIN
    UPDATE Worker SET SalaryPerHour=salaryPH
    WHERE LeftDate IS NULL
        AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM JoinDate) >= yearsWorking)
        AND (SalaryPerHour < salaryPHLessThan);
    updatedRows := SQL%ROWCOUNT;
END;
```

כאן בדקנו שהפרוצדורה באמת עובדת:

Test script | DBMS Output | Statistics | Profiler | Trace

```
declare
  updatedRows integer;
begin
  SetSalaryToOldWorkers(30, 100, 100, :updatedRows);
end;
```

Variable	Type	Value
updatedRows	String	113

2. פיטור עובד.

מטרה: משאבי אנוש צריכים את הפרוצדורה הזאת על מנת לפטר עובד (או שהמוזיאון מפטר אותו או שהוא עוזב), לכן קיימת הפרוצדורה הזאת.

```
CREATE OR REPLACE PROCEDURE FireWorker(workerId_ INT)
AS
BEGIN
    UPDATE Worker SET LeftDate = CURRENT_DATE
    WHERE workerId = workerId_ AND LeftDate IS NULL;
END;
```

כאן בדקנו שהפרוצדורה באמת עובדת:

Test script | DBMS Output | Statistics | Profiler | Trace

```
declare
  id INT;
  d Worker.LeftDate%TYPE;
begin
  id := 19786;
  FireWorker(id);
  SELECT LeftDate INTO d FROM Worker WHERE WorkerId = id;
end;
```

10.2 פונקציות

1. האם עובד עדיין עובד.

מטרה: במהלך כתיבת השאילתות והתצוגות השתמשנו הרבה בתנאי LeftDate IS NULL בשביל לבדוק שהעובד עדיין עובד. על מנת לשפר את הקריאות של השאילתות הוספנו את הפונקציה IsWorking(LeftDate) וכך יהיה מובן מה הכוונה.

```
CREATE OR REPLACE FUNCTION IsWorking(leftDate DATE)
RETURN INT
AS
BEGIN
    IF leftDate IS NULL THEN
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
END;
```

להלן אפשר לראות שהפונקציה עובדת:

```
SELECT * FROM Worker WHERE IsWorking(leftDate) = 1;
```

	WORKERID	NAME	ROLE	SALARYPERHOUR	JOINDATE	LEFTDATE	BIRTHDATE	EMAIL
1	19787	Courtney Arroyo	Cleaner	44.52	30/06/2018		28/04/1988	Courtney_Arroyo577@gmail.com
2	19788	Luis Raymond	Manager	118.26	23/08/2008		04/01/1976	LuisRaymond644@gmail.com
3	19789	Richard Bowman	Cleaner	120	21/07/1996		24/01/1966	RichardBowman991@gmail.com
4	19790	Christopher Moore PhD	Conservator	73.64	19/09/2011		10/08/1971	ChristopherMoorePhD634@walla.com
5	19791	Kelsey Pennington	Conservator	80.67	23/09/1997		15/12/1960	Kelsey-Pennington542@hotmail.com
6	19792	Rachael Beasley	Exhibit Designer	62.15	30/10/2019		18/11/1994	RachaelBeasley773@hotmail.com
7	19793	Sarah Johnson	Exhibit Designer	102.4	20/12/2007		20/04/1972	Sarah-Johnson316@hotmail.com
8	19794	Darren Flowers	Manager	87.45	17/11/2019		14/03/1982	Darren-Flowers231@outlook.com
9	19795	Scott Cox	Curators	121.79	13/06/2018		03/04/1963	Scott_Cox383@outlook.com
10	19796	Kenneth Frank	Guard	81.41	17/09/2004		08/01/1973	KennethFrank543@hotmail.com

2. מספר העובדים במחלקה.

מטרה: הפונקציה נותנת את מספר העובדים במחלקה עם מספר המחלקה בקלט.

```
CREATE OR REPLACE FUNCTION NumberWorkersOfDepartment(departmentId_ INT)
RETURN INT
AS
    result INT;
BEGIN
    SELECT COUNT(*) INTO result
    FROM Department NATURAL JOIN WorksAt wa
         INNER JOIN Worker w ON w.WorkerId = wa.WorkerId
    WHERE IsWorking(leftDate) = 1 AND DepartmentId = departmentId_
    GROUP BY DepartmentId;

    RETURN result;
END;
```

להלן אפשר לראות שהפונקציה עובדת:

Test script | DBMS Output | Statistics | Profiler | Trace

```
1
2 declare
3   res integer;
4 begin
5   res := NumberWorkersOfDepartment(0);
6 end;
```

Variable	Value
res	6

Call stack

[Line 6] end;

11. נספח

11.1 הקוד ליצירת טבלאות הדוגמה

```sql
CREATE TABLE area
(
  areaID NUMERIC(3) NOT NULL,
  areaName VARCHAR(20) NOT NULL,
  PRIMARY KEY (areaID)
);

CREATE TABLE city
(
  cityID INT NOT NULL,
  cityName VARCHAR(20) NOT NULL,
  areaID NUMERIC(3) NOT NULL,
  PRIMARY KEY (cityID),
  FOREIGN KEY (areaID) REFERENCES area(areaID)
);

CREATE TABLE agent
(
  agentID NUMERIC(9) NOT NULL,
  agentName VARCHAR(20) NOT NULL,
  rating NUMERIC(2) NOT NULL,
  hireYear NUMERIC(4) NOT NULL,
  bossID NUMERIC(9) NOT NULL,
  salary FLOAT NOT NULL,
  areaID NUMERIC(3) NOT NULL,
  PRIMARY KEY (agentID),
  FOREIGN KEY (areaID) REFERENCES area(areaID)
);

CREATE TABLE client
(
  clientID NUMERIC(9) NOT NULL,
  clientName VARCHAR(20) NOT NULL,
  phoneNr VARCHAR(10) NOT NULL,
  address VARCHAR(25) NOT NULL,
  cityID INT NOT NULL,
  agentID NUMERIC(9) NOT NULL,
  PRIMARY KEY (clientID),
  FOREIGN KEY (cityID) REFERENCES city(cityID),
  FOREIGN KEY (agentID) REFERENCES agent(agentID)
);

CREATE TABLE Schedule
(
  meetingTime DATE NOT NULL,
  clientID NUMERIC(9) NOT NULL,
  agentID NUMERIC(9) NOT NULL,
  PRIMARY KEY (meetingTime, clientID, agentID),
  FOREIGN KEY (clientID) REFERENCES client(clientID),
  FOREIGN KEY (agentID) REFERENCES agent(agentID)
);
```

11.2 הקוד ליצירת הטבלאות

```sql
CREATE TABLE Worker
(
  WorkerId INT NOT NULL,
  Name VARCHAR(50) NOT NULL,
  Role VARCHAR(50) NOT NULL,
  SalaryPerHour FLOAT NOT NULL,
  JoinDate DATE NOT NULL,
  LeftDate DATE,
  BirthDate DATE NOT NULL,
  Email VARCHAR(50) NOT NULL,
  PRIMARY KEY (WorkerId)
);

CREATE TABLE Wing
(
  WingId INT NOT NULL,
  Name VARCHAR(50) NOT NULL,
  PRIMARY KEY (WingId)
);

CREATE TABLE Department
(
  DepartmentId INT NOT NULL,
  Name VARCHAR(50) NOT NULL,
  WingId INT NOT NULL,
  PRIMARY KEY (DepartmentId),
  FOREIGN KEY (WingId) REFERENCES Wing(WingId)
);

CREATE TABLE WorksAt
(
  WorkerId INT NOT NULL,
  DepartmentId INT NOT NULL,
  PRIMARY KEY (WorkerId, DepartmentId),
  FOREIGN KEY (WorkerId) REFERENCES Worker(WorkerId),
  FOREIGN KEY (DepartmentId) REFERENCES Department(DepartmentId)
);
```

11.3 הקוד לאכלוס הטבלאות וקבצי CSV

https://github.com/nbashan/MiniProjectInDataBases/tree/master/scripts

11.4 הקוד של השאילתות

.1

```sql
-- Find all working workers that works for a long time (30 years and above) and earns a low salary (below 100 per hour)
SELECT WorkerId, Name, SalaryPerHour, JoinDate, Email, BirthDate
FROM Worker WHERE LeftDate IS NULL
    AND (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM JoinDate) >= 30)
    AND (SalaryPerHour < 100.0);
```

.2

```sql
-- Calculate the salary to pay for workers per an average month (20 days with 8 hours per day)
SELECT SUM(SalaryPerHour) * 8 * 20 AS Salary
FROM Worker WHERE LeftDate IS NULL;
```

.3

```sql
-- Show all roles with their workers and working workers
SELECT Role, Count(*) AS All_, Sum(IsWorking) AS Current_
FROM (SELECT Role, IsWorking(LeftDate) AS IsWorking FROM Worker)
GROUP BY Role
ORDER BY All_;
```

.4

```sql
-- Shows the distribution of salary between the workers (rounds to the closest ten)
SELECT (ROUND(SalaryPerHour) - MOD(ROUND(SalaryPerHour),10)) AS Salary, COUNT(*) AS Workers
FROM Worker
WHERE LeftDate IS NULL
GROUP BY (ROUND(SalaryPerHour) - MOD(ROUND(SalaryPerHour),10))
ORDER BY salary
```

.5

```sql
-- Number of working workers per department
SELECT DepartmentId, Name, COUNT(*) AS Workers
FROM WorksAt NATURAL JOIN Department
WHERE WorkerId IN (SELECT WorkerId FROM Worker WHERE LeftDate IS NULL)
GROUP BY DepartmentId, Name;
```

.6

```sql
-- Number of departments each working worker works on
SELECT WorkerId, Name, SalaryPerHour, JoinDate, Email, COUNT(*) AS Departments
FROM WorksAt NATURAL JOIN Worker
WHERE LeftDate IS NULL
GROUP BY WorkerId, SalaryPerHour, Name, JoinDate, Email;
```

.7

```sql
-- Workers with the same birth date
SELECT w1.WorkerId, w1.Name, w1.Email, w1.BirthDate, w1.JoinDate, w1.LeftDate
FROM Worker w1 JOIN Worker w2
ON TO_CHAR(w1.BirthDate, 'MM-dd') = TO_CHAR(w2.BirthDate, 'MM-dd')
GROUP BY w1.WorkerId, w1.Name, w1.Email, w1.BirthDate, w1.JoinDate, w1.LeftDate
ORDER BY TO_CHAR(w1.BirthDate, 'MM-dd');
```

.8

```sql
-- Find all departments that don't have a manager
SELECT DepartmentId, Name FROM Department
WHERE DepartmentId IN (SELECT DepartmentId FROM Department
                       MINUS
                       SELECT d.DepartmentId
                       FROM Worker NATURAL JOIN WorksAt wa INNER JOIN Department d ON d.DepartmentId = wa.DepartmentId
                       WHERE Role = 'Manager');
```

11.5 הקוד של האינדקסים

.1

```sql
CREATE INDEX i_WorkerBirthdate
ON Worker(BirthDate)
```

.2

```sql
CREATE INDEX i_WorkerLeftdate
ON Worker(LeftDate)
```

.3

```sql
CREATE INDEX i_WorksAtWorkerId
ON WorksAt(WorkerId);
```

11.6 הקוד של שאילתות האינטגרציה

הרשאות גישה:

```sql
GRANT SELECT, REFERENCES ON Department
TO aheller;

GRANT SELECT, REFERENCES ON Wing
TO aheller;

GRANT SELECT, REFERENCES ON Department
TO silon;

GRANT SELECT, REFERENCES ON Wing
TO silon;

GRANT SELECT, REFERENCES ON Worker
TO yyohanan;
```

.1

```sql
-- Select all departments and their price of current exhibitions
SELECT d.DepartmentId, d.Name, SUM(Price)
FROM Department d
     INNER JOIN Soria.ExhibitionAtDepartment ead ON d.DepartmentId = ead.Department_Id
     NATURAL JOIN Soria.Exhibition
     NATURAL JOIN Soria.ExhibitAtExhibition ee
     INNER JOIN Soria.Exhibit e ON e.ExhibitId = ee.ExhibitId
WHERE StartTime <= CURRENT_DATE AND CURRENT_DATE <= EndTime
GROUP BY d.DepartmentId, d.Name
```

.2

```sql
-- Find all departments that currently don't contains any exhibitions.
SELECT DepartmentId, Name FROM Department
WHERE DepartmentId NOT IN (
    SELECT Department_Id
    FROM Soria.ExhibitionAtDepartment NATURAL JOIN Soria.Exhibition
    WHERE StartTime <= CURRENT_DATE AND CURRENT_DATE <= EndTime
);
```

.3

```sql
-- Finds departments that currently contains an exhibition but without any guards.
SELECT DepartmentId, Name
FROM Department
    INNER JOIN Soria.ExhibitionAtDepartment ON DepartmentId = Department_Id
    NATURAL JOIN Soria.Exhibition
WHERE StartTime <= CURRENT_DATE AND CURRENT_DATE <= EndTime AND DepartmentId NOT IN (
    SELECT DepartmentId
    FROM Department NATURAL JOIN WorksAt wa INNER JOIN Worker w ON wa.WorkerId = w.WorkerId
    WHERE LeftDate IS NULL AND Role = 'Guard'
)
GROUP BY DepartmentId, Name
ORDER BY DepartmentId;
```

11.7 הקוד של הViews

.1

```sql
CREATE VIEW workersOrderedBySalary
AS
SELECT WorkerId, Name, Role, SalaryPerHour, Email, JoinDate, BirthDate
FROM Worker
WHERE IsWorking(LeftDate)
ORDER BY SalaryPerHour desc
```

.2

```sql
CREATE VIEW totalSalaryPerMonth
AS
SELECT sum(SalaryPerHour)*160
FROM Worker
```

.3

```sql
CREATE VIEW v_WorkingManagers
AS
SELECT WorkerId, Name, Role, SalaryPerHour, Email, JoinDate, BirthDate
FROM Worker WHERE LeftDate IS NULL AND Role == 'Manager';
```

.4

```sql
CREATE VIEW v_WorkingWorkersWithDepartment
AS
SELECT DepartmentId, DepartmentName, WorkerId, Name AS WorkerName, Role, SalaryPerHour, Email, JoinDate, BirthDate
FROM (SELECT DepartmentId, Name AS DepartmentName, WorkerId
      FROM WorksAt NATURAL JOIN Department) NATURAL JOIN
 (SELECT * FROM Worker WHERE LeftDate IS NULL)
ORDER BY DepartmentId;
```