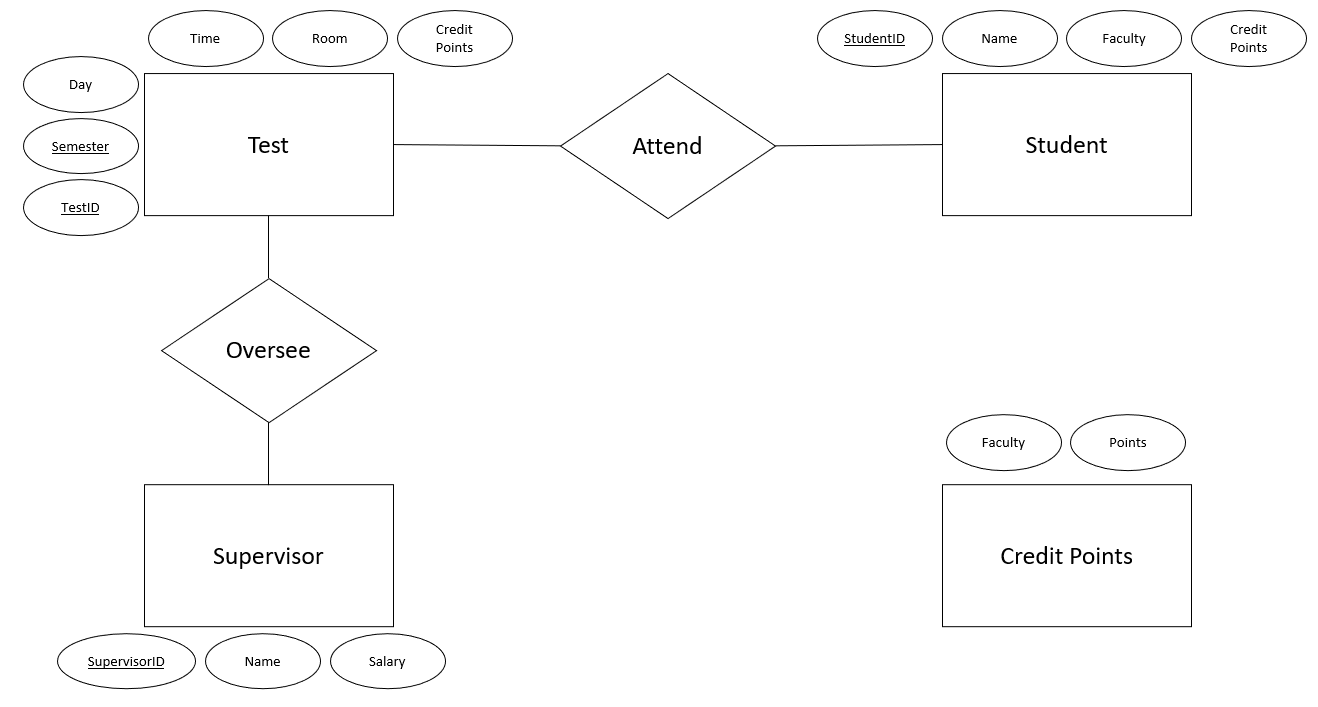
**Database Systems - 236363 HW2 – Dry**

**Design:**

בחרנו ב-ERD הבא על מנת לתחזק את מסד הנתונים שלנו ונפרט עליו כעת:

**Test** – יחס זה מכיל בתוכו את כל הטסטים המתקיימים, כאשר TestID, Semester הם המפתחות ו-Day, Time, Room, Credit Points הם ה-Attributes שלו. בהוספת רשומה ליחס מתבצע וידוא שכל הערכים הם לא NULL ובנוסף מתבצע וידוא ש:

* TestID > 0
* Room > 0
* Time is BETWEEN 1 AND 3
* Semester is BETWEEN 1 AND 3
* Day is BETWEEN 1 AND 31

**Student** – יחס זה מכיל בתוכו את כל הסטודנטים, כאשר StudentID הוא המפתח ו-Name, Faculty, Credit Point הם ה-Attributes שלו. בהוספת רשומה ליחס מתבצע וידוא שכל הערכים הם לא NULL ובנוסף מתבצע וידוא ש:

* StudentID > 0
* Credit Points >= 0

**Supervisor** – יחס זה מכיל בתוכו את כל המשגיחים, כאשר SupervisorID הוא המפתח ו-Name, Salary הם ה-Attributes שלו. בהוספת רשומה ליחס מתבצע וידוא שכל הערכים הם לא NULL ובנוסף מתבצע וידוא ש:

* SupervisorID > 0
* Salary >= 0

**Oversee** – זהו Relationship בין Test ו-Supervisor ומכיל בתוכו משגיחים ואת הטסטים עליהם הם משגיחים. כאשר TestID, Semester הם FK של היחס Test ו-SupervisorID הוא FK של היחס Supervisor. שלשה זו מייצגת מפתח ביחס ה-Oversee ורשומה ביחס זה מייצגת משגיח בעל SupervisorID אשר משגיח על מבחן TestID בסמסטר Semester.

**Attend** – זהו Relationship בין Test ו-Student ומכיל בתוכו סטודנטים ואת הטסטים אליהם הם נגשים. כאשר TestID, Semester הם FK של היחס Test ו-StudentID הוא FK של היחס Student. שלשה זו מייצגת מפתח ביחס ה-Attend ורשומה ביחס זה מייצגת סטודנט בעל StudentID אשר ניגש למבחן TestID בסמסטר Semester.

**Credit Points** – היחס ההתחלתי אותו קיבלנו מכיל בתוכו את כל הפקולטות ומספר הנק' הדרוש לסיום בפקולטה. כאשר Faculty הוא מפתח ו-Points הם ה- Attributes שלו. בתוך היחס קיימות רשומות התחלתיות וה-API שלנו לא מוסיף או מוחק מהיחס.

**clearTables** – פונקציה שמוחקת את כל הרשומות מכל היחסים ב-DB, אך לא מוחקת את היחסים עצמם. מתבצע באמצעות TRUNCATE TABLE בשאילתא.

**dropTables** – פונקציה שמוחקת את כל היחסים וה-VIEWS מה-DB באמצעות DROP TABLE או DROP VIEW בשאילתא בהתאמה. חשוב למחוק קודם את כל ה-VIEWS שכן הם תלויים בשאר היחסים ב-DB ולכן אי שמירה על סדר זה תגרום לשגיאה.

**CRUD and Basic API:**

**addTest** – פונקציה פשוטה המכניסה רשומה ליחס Test באמצעות INSERT INTO בשאילתא.

**getTestProfile** – פונקציה השולפת מבחן ספציפי תוך מהיחס Test באמצעות השוואה בשאילתא למפתח (ה-ID והסמסטר) שקיבלה בתור ארגומנט.

**deleteTest** – פונקציה פשוטה המוחקת רשומה מהיחס Test באמצעות DELETE FROM בשאילתא.

**addStudent** – פונקציה פשוטה המכניסה רשומה ליחס Student באמצעות INSERT INTO בשאילתא.

**getStudentProfile** – פונקציה השולפת מבחן ספציפי תוך מהיחס Student באמצעות השוואה בשאילתא למפתח (ה-ID) שקיבלה בתור ארגומנט.

**deleteStudent** – פונקציה פשוטה המוחקת רשומה מהיחס Student באמצעות DELETE FROM בשאילתא.

**addSupervisor** – פונקציה פשוטה המכניסה רשומה ליחס Supervisor באמצעות INSERT INTO בשאילתא.

**getSupervisorProfile** – פונקציה השולפת מבחן ספציפי תוך מהיחס Supervisor באמצעות השוואה בשאילתא למפתח (ה-ID) שקיבלה בתור ארגומנט.

**deleteSupervisor** – פונקציה פשוטה המוחקת רשומה מהיחס Supervisor באמצעות DELETE FROM בשאילתא.

**studentAttendTest** – פונקציה פשוטה המכניסה רשומה ליחס Attend באמצעות INSERT INTO בשאילתא.

**studentWaiveTest** – פונקציה פשוטה המוחקת רשומה מהיחס Attend באמצעות DELETE FROM בשאילתא.

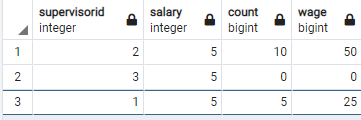
**supervisorOverseeTest** – פונקציה פשוטה המכניסה רשומה ליחס Oversee באמצעות INSERT INTO בשאילתא.

**supervisorStopsOverseeTest** – פונקציה פשוטה המוחקת רשומה מהיחס Oversee באמצעות DELETE FROM בשאילתא.

**averageTestCost –** לצורך פתרון התרגיל תחילה ביצענו תת-שאילתא המוצאת את המשכורת הממוצעת למשגיח פר מבחן. על תת-שאילתא זו סכמנו את הממוצעים פר מבחן וחילקנו במספר המבחנים כדי לקבל את הנדרש.

**getWage** – לצורך פתרון התרגיל יצרנו VIEW בשם Salaries המכיל את SupervisorID המשכורת שלו לטסט, מספר הטסטים שהוא משגיח עליהם וכפל בין 2 העמודות שזהו השכר שהוא עתיד לקבל.

לבסוף באמצעות שאילתה פשוטה שלפנו את השכר של משגיח המתאים לSupervisorID וקיבלנו את הWage.



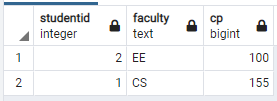
**supervisorOverseeStudent –** ביצענו NATURAL JOIN בין Attend ל-Oversee , קיבצנו את התוצאה על פי זוגות של סטודנטים ומשגיחים כך שיש לפחות 2 איברים בכל קבוצה, ובחרנו את הסטודנטים. הוספנו DISTINCT כדי לא לקבל כפילויות וסידרנו לפי StudentID בסדר יורד.

**TestsThisSemester** – מטבלת טסט שלפנו את הטסטים אשר קורים הסמסטר סידרנו לפי TestID בסדר יורד והגבלנו לעד 5 רשומות.

**studentHalfWayThere –** כדי לראות כמה נק"ז יש לכל הסטודנטים ביצענו NATURAL JOIN בין Student ל-CreditPoints ומשם בחרנו את כל הסטודנטים שמקיימים שהנק"ז שלהם אכן גדול ממחצית הנק"ז הדרוש לסיום בהתאם לפקולטה בה הם נמצאים. אם הנק"ז הדרוש אי-זוגי, כאשר חילקנו אותו ב-2 לקחנו ערך שלם עליון (במקום ערך שלם תחתון שנלקח בברירת מחדל בחלוקה של int) כדי לשמור על נכונות השאילתא.

**studentCreditPoints -** לצורך פתרון התרגיל השתמשנו ב-VIEW בשם CpAfterAttend המכיל את StudentID הפקולטה שלו, ומספר הנקודות שיהיו לו לאחר שהוא יגש לכל המבחנים אליהם הוא רשום.

לבסוף באמצעות שאילתה פשוטה שלפנו את נק' הזכות של הסטודנט המתאים ל-StudentID וקיבלנו את נק' הזכות של הסטודנט לאחר שייגש למבחנים אשר הוא רשום אליהם.

****

**getMostPopularTest** – כדי לראות את כל המבחנים אליהם רשומים סטודנטים ביצענו NATURAL JOIN בין Student ל-Attend קיבצנו את התוצאה על פי מבחנים וספרנו את הקבוצות. מכל זה בחרנו את מספר המבחן וכמות הופעותיו (מספר הקבוצות שלו). לבסוף, בסידרנו את הרשומות בתוצאה בסדר יורד והגבלנו את מספר הרשומות לאחד כדי לקבל בתוצאה את הרשומה עם המבחן הפופולרי ביותר.

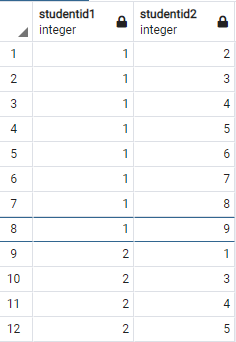
**getConflictingTests** – ביצענו מכפלה קרטזית בין Test לTest ממנה לקחנו את כל הטסטים בעלי ID שונה שהסמסטר, היום והשעה שלהם זהים, הוספנו DISTINCT כדי לא לקבל כפילויות וסידרנו לפי TESTID בסדר עולה.

**graduateStudents** – לצורך פתרון התרגיל השתמשנו ב-VIEW בשם CpAfterAttend המכיל את StudentID הפקולטה שלו, ומספר הנקודות שיהיו לו לאחר שהוא יגש לכל המבחנים אליהם הוא רשום. כדי לראות כמה נק"ז יש לכל הסטודנטים לאחר שיגשו לכל המבחנים שאליהם הם רשומים ביצענו NATURAL JOIN בין Student ל- CpAfterAttend ומשם בחרנו את כל הסטודנטים שהנק"ז שלהם גדול מהנק"ז הדרוש לסיום בהתאם לפקולטה בה הם נמצאים. סידרנו לפי StudentID בסדר עולה והגבלנו את מספר הרשומות ל-5 כנדרש.

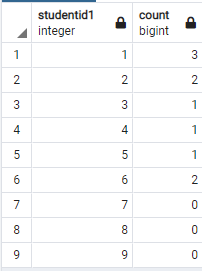
**getCloseStudents** – לצורך פתרון התרגיל יצרנו 3 VIEWS עליהם נפרט כעת:

1. StudentWithALLStudents – מכיל רשימה studentid1,studentid2 של כל הסטונדטים עם כל הסטודנטים האחרים:

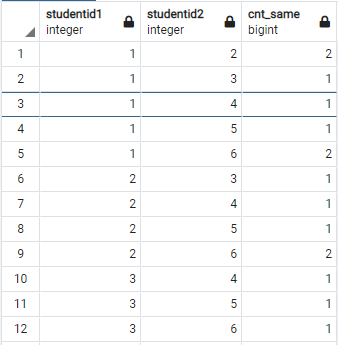
בצורה יותר פורמאלית:



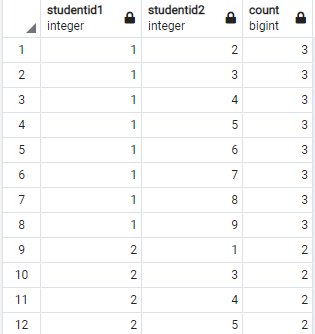
1. StudentTestCount – מכיל רשימה studentid1,count של כל הסטונדטים עם כמות הטסטים אליהם הם ניגשים:



1. Same – לכל זוג סטודנטים שונים StudentID1,StudentID2 מחזיר כמה טסטים הם לוקחים יחד.



על VIEWS 1,2 עשינו NATURAL JOIN וקיבלנו יחס שבו יש את StudentID1,StudentID2 וכמה טסטים לקח StudentID1



ליחס זה עשינו LEFT OUTER JOIN עם טבלת SAME וכך קיבלנו זוג סטודנטים StudentID1,StudentID2 כמה טסטים לקח StudentID1 וכמה טסטים לוקח StudentID2 יחד איתו.

מטבלה זו שלפנו את Student 2 בסדר יורד בהגבלה עד 10 לכל המקומות שStudentID1 = input וגם שכמות הטסטים שלוקח Student2 כפול 2 היא גדולה או שווה מכמות הטסטים שלוקח StudentID1 סה"כ וקיבלנו את המבוקש.