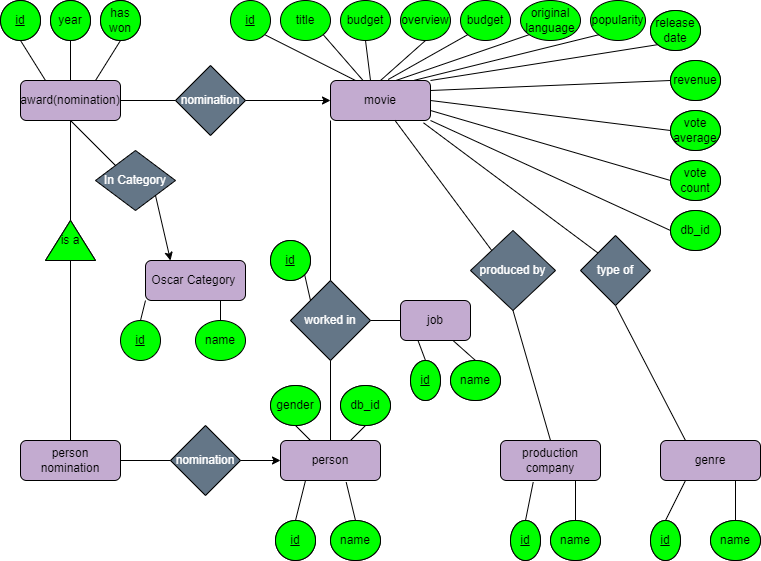
Software documentation



הסבר על בניית הסכמה:

* לכל סרט שמרנו את רשימת הז'אנרים ורשימת חברות הפקה אשר הפיקו את הסרט. לכל סרט ייתכנו מס' חברות הפקה שהפיקו את הסרט וכמה ז'אנרים בו הוא מתויג, וגם להיפך. לכן, יצרנו טבלאות אשר מקשרות בין הסרט לז'אנר/הסרט לחברת ההפקה.
* את טבלת הperson\_movie\_job יצרנו על מנת לשמור את תפקידי האנשים בסרט מסוים.

ייתכן שאדם מסוים שיחק בשני תפקידים שונים באותו הסרט ולכן לכל רשומה כזו בטבלה יש ID שמייחד אותה. שאר העמודות הינן foreign keys

* קטגוריה אחת יכולה להשתתף בכמה מועמדויות וכך המצב גם עבור סרט ולכן

היחס של one to many ושימוש ב foreign keys

* קובץ הCSV בו עבדנו על מנת לחלץ את כל המועמדויות לאוסקר בקטגוריות השונות הכיל את שנת המועמדות, הקטגוריה של הפרס, הסרט שהיה מועמד וכתלות בקטגוריה- האדם שזכה בפרס על אותו הסרט. לכן, יצרנו טבלה אחת שתכיל את כל המועמדויות של הסרטים לאוסקר (award) וטבלה נוספת, אשר במידת הצורך מקשרת בין אותה המועמדות לשחקן/במאי (person\_award).  
  כך בעצם אנחנו יכולים להבדיל בין פרסים אשר ניתנו לשחקנים ופרסים שנתנו לסרט. בנוסף, העיצוב מאפשר לנו להימנע מערכי NULL כאשר בעמודת הID של השחקן כאשר המועמדת היא על קטגוריה של סרט. ולכן העיצוב הנ"ל הינו האופטימלי ביותר.

אופטימיזציות:

* שימוש ב foreign keys בכל הטבלאות שמקשרות בין מידע של טבלאות, על מנת לשמור על אמינות המידע ולהקטין גדלי הטבלאות.
* שימוש בnot null בכל מקום שאפשר על מנת לאפשר לאינדקסים לעבוד בצורה המיטבית ביותר ולחסוך בדיקות של האם הערך קיים.
* שימוש בdata types קטנים ככל האפשר על מנת לשמור על גודל הרשומה כקטנה ככל האפשר, לדוגמא בעמודת הhas\_won אשר מתארת אם אותה מועמדות זכתה או לא בטבלת הaward, השתמשנו בBIT שיכול להיות 0 או 1. דוגמא נוספת היא שמירת שנת המועמדות לאוסקר בSMALLINT.  
  במקום בINT מקטינה את גודל הרשומה ומאפשרת יצירת אינדקס על עמודה זו בגודל קטן יותר.
* שימוש באינדקסים - ייצרנו 4 אינדקסים:
  + יצרנו אינדקס HASH על עמודת השם של טבלת האנשים כיוון שרוב השאילתות על טבלה הזו הן שאילתות point query על השם של האדם.
  + יצרנו אינדקס HASH על עמודת השם של טבלת הסרטים כיוון שרוב השאילתות המשתמשות/מתבצעות על טבלה זו הן שאילתות point query על שם הסרט.
  + יצרנו אינדקס B-TREE על עמודות הID והYEAR של טבלת הaward כיוון שכל השאילתות שלנו שמשתמשות בטבלת המועמדויות, משתמשות תמיד בעמודת הID. בחלקן, השאילתות מפלטרות לפי טווח של שנים. ולכן אינדקס מבצע אופטימיזציה לשאילתות שמשתמשות בטבלה זו.
  + יצרנו אינדקס HASH על עמודות הmovie\_id וjob\_id בטבלת הperson\_movie\_job מכיוון שחלק מהשאילתות שמשתמשות בטבלה זו הן שאילתות point query על העמודות הנ"ל. בשאילתות אלה הפילטור תמיד מתבצע על עמודות הmovie\_id ולעיתים בנוסף גם על job\_id.
* בנוסף לאינדקסים שיצרנו, נוספו האינדקסים על הPK והFK של כל טבלה שנוצרו אוטומטית ביצירת הטבלאות.

(בפועל כל האינדקסים הם מסוג B-TREE מכיוון שmysql לא אפשר ליצור אינדקס מסוג HASH.)

תיאור 7 השאילתות המרכזיות שלנו:

1. Full text query שאילתה שמקבלת טקסט חופשי (free\_text) ומחזירה רשימה של 10 הסרטים שמכילים את המחרוזת הנ"ל. רשימת הסרטים מוחזרת בסדר יורד לפי הפופולריות.

"Select \* from movie where match (title) against ('{free\_text}\*' IN BOOLEAN MODE) order by popularity desc limit 10"

אופטימיזציות:   
- יצרנו full text indexעל עמודת השם של הסרט כדי לייעל את החיפוש לפי ביטוי רגולרי על שם הסרט.  
  
בנינו את הטבלה של הסרטים כך שאין סרטים עם שמות שהם null.

1. שאילתה שמקבלת שם של סרט ומחזירה את מספר הזכיות שלו באוסקר

SELECT count(\*) as numOfWins FROM movie as m, award as a"

WHERE m.title = "{movie\_title}" and m.id=a.movie\_id and a.has\_won=1"

אופטימיזציות:

- יצרנו index על עמודת השם של הסרט כדי לייעל את החיפוש לפי שם הסרט.

- השאילתה משתמשת בindex על עמודת הmovie\_id בטבלת הaward.

בנינו את הטבלה של הסרטים כך שאין סרטים עם שמות שהם null.

שימוש בnot null בעמודת הmovie\_id בטבלת הaward על מנת לחסוך בדיקות של הערך שהוא לא null.

1. שאילתה שמקבלת שם של אדם ומחזירה את מספר הסרטים שהיו מועמדים לאוסקר בו הוא השתתף.

SELECT COUNT(\*) FROM

( SELECT distinct movie.title AS movieTitle,

jobInMovie.job\_name AS role

FROM movie, jobInMovie,

person\_movie\_job,

person

WHERE person.name = "{person\_name}"

AND person.id = person\_movie\_job.person\_id

AND person\_movie\_job.job\_id = jobInMovie.id

AND person\_movie\_job.movie\_id = movie.id

) AS personMovies

אופטימיזציות:

- יצרנו index על עמודת השם של האדם כדי לייעל את החיפוש לפי person.name.

- השאילתה משתמשת בindex על עמודת ה- person\_movie\_job.person\_id, שמייעלת את הפילטור לפי עמודה זו.

- השאילתה משתמשת בindex על עמודת ה- movie.id, שמייעלת את הפילטור לפי עמודה זו, וגם מייעל את ה- distinct בשאילתה הפנימית.

בנינו את הטבלה person כך שאין שמות של אנשים שהם null.

שימוש בnot null בעמודות בטבלה person\_movie\_job על מנת לחסוך בדיקות של הערך שהוא לא null.

1. שאילתה שמקבלת כפרמטרים :

טווח של שנים, רשימת קטגוריות category1, category2.. ורשימת ג'אנרים genre1, genre2.., ומחזירה את 5 האנשים שזכו בהכי הרבה אוסקרים לפי פילטורים אלו (בסדר יורד)

SELECT person.name AS name,

count(\*) AS numOfAwards

FROM person,

award\_person,

movie\_genre,

award

WHERE person.id=award\_person.person\_id

AND award.movie\_id = movie\_genre.movie\_id

AND ( movie\_genre.genre\_id = {genre1} OR movie\_genre.genre\_id = {genre2} …)

AND (award.oscar\_category\_id = { category1} OR award.oscar\_category\_id = { category2} …)

AND award.year >= {min\_year}

AND award.year <= {max\_year}

AND award.has\_won = 1

AND award.id = award\_person.award\_id

group by person.name order by numOfAwards desc limit 5

אופטימיזציות:

- השתמשנו ב- index-ים על העמודות:

award.id, award\_person.award\_id person.id, award\_person.person\_id, award.movie\_id, movie\_genre.movie\_id,

בשביל לייעל על פעולות ה-join.

השתמשנו ב-index-ים על העמודות:

movie\_genre.genre\_id, award.oscar\_category\_id, award.year

בשביל לייעל את החיפושים לפי עמודות אלו.

שימוש בnot null בעמודות שהן FK בכל הטבלאות, על מנת לחסוך בדיקות של הערך שהוא לא null.

1. שאילתה שמקבלת כפרמטרים :

טווח של שנים, רשימת קטגוריות category1, category2.. , רשימת ג'אנרים genre1, genre2.., ומשתנה בוליאני only\_winners ומחזירה את 5 הסרטים המועמדים לאוסקר הכי רווחיים לפי קריטריונים אלו.

SELECT title AS title,

MAX(revenue) AS maxRevenue

FROM movie,

award,

movie\_genre

WHERE movie.id=award.movie\_id

AND award.movie\_id = movie\_genre.movie\_id

AND ( movie\_genre.genre\_id = {genre1} OR movie\_genre.genre\_id = {genre2} …)

AND (award.oscar\_category\_id = { category1} OR award.oscar\_category\_id = { category2} …)

AND award.year >= {min\_year}

AND award.year <= {max\_year}

AND award.has\_won = { only\_winners }

group by title order by maxRevenue desc limit 1