מבחן **SQL** 2025 מועד א׳ – פתרון שאלה 2 תקינות נתונים:

a. כאשר אנחנו מגדירים שדה כ PK, אנחנו כופים ערכים שונים, מצפים שניתן יהיה להזין את הערכים הנדרשים, מעוניינים להשתמש בו לקישור לטבלאות אחרות וחיפוש מהיר כבכל אינדקס. לפי דעתי לא כדאי להגדיר את השדה title כמפתח ראשי מהסיבות הבאות:

i. השדה מוגדר כ־VARCHAR, אך אינדקסים על מחרוזות פחות יעילים לעומת אינדקסים על מספרים, שנועדו לערכים מדויקים עם סדר טבעי.

ii. שימוש ב varchar ב join אם טבלאות אחרות איטי ולא מומלץ.

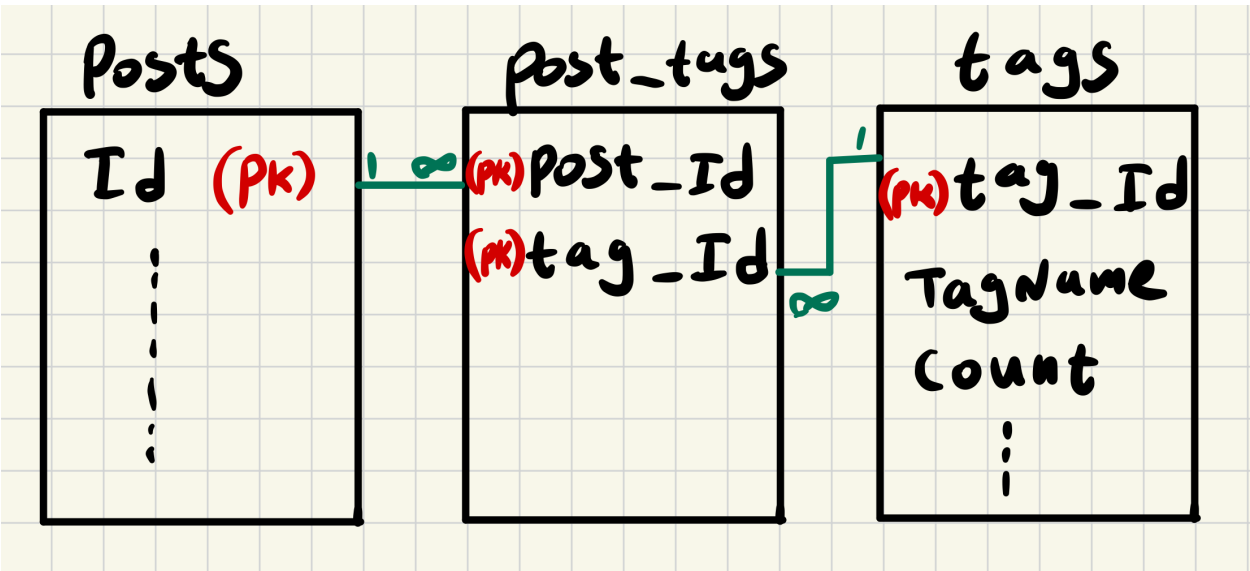
iii. משתמש אשר כותב פוסט לא יכול להכיר את כל הכותרות במערכת, ולכן לא ידע אם קיימת כבר כותרת זהה או דומה וכך לא לחזור על ערכים.

iv. היכרות המשתמש המחפש עם כלל הכותרות במערכת היא מוגבלת, ולכן בעת חיפוש בסגנון "גוגל", אם קיימות מספר כותרות זהות, כולן יופיעו בתוצאות. מצב זה מקשה על המשתמש להבחין בין הפוסטים ולבחור את הפוסט הרצוי.

v. ניתן ורצוי להוסיף אילוץ unique מבלי להגדיר מפתח ראשי. כך לא יהיו כותרות כפולות שיקשו על החיפוש ונבצע join לפי המפתח המספרי. כותבי פוסטים אכן עלולים לנסות להזין כותרת קיימת. האתר יתפוס זאת, יפנה את הכותב אליהם שיחליט האם הם עונים כבר לשאלתו או ינסח הבדל ספציפי יותר בכותרת.

b.

1. היחס בין tags לבין post הינו רבים לרבים. עבור כל post קיימים tags רבים (לדוגמה על sql ו linux) ולכל tag פוסטים רבים (שלל שאלות על sql).
2. ניתן לייצג יחס n:n ע״י טבלה מקשרת בין טבלת posts וטבלת tags. המקור. הטבלה תכיל את מפתחות הישויות המקושרות ותכלול אילוצי FK עליהם למניעת ערכים שאינם בישות המקורר. ראו הצעה לפתרון:



נוסיף את טבלת post tags כך שתתאים לסכמה שלנו:

```
Create table post_tags (
post_Id int(11) not null,
tag_Id int(11) not null,
Primary key (post_Id, tag_Id),
Foreign key (post_Id) references posts(Id) ,
Foreign key (tag_Id) references tags(tag_Id)
);
```

3. שליפת פוסטים שיש להם את tag מס׳ 7.

```
SELECT Id
FROM posts
WHERE
  Tags = '[7]'
  OR Tags LIKE '[7,%'
  OR Tags LIKE '%,7,%'
  OR Tags LIKE '%,7]%';
```

4. שמירת כל ה־tags של פוסט מסוים בשדה אחד הווה pre-compute. בכך היא חוסכת זמן בטעינת הדף שחייבת להיות מהירה. יותר מזה, החישוב פר פוסט נשאר זהה כל עוד לא נוספים תגים וכך אין צורך לחשב עובר כל כניסה לדף. במערכות אמיתיות, נהוג לשמור את שמות התגים עצמם ולא רק את האינדקסים שלהם, כדי לחסוך גם את פיענוחם.

c.

i. היתרון בכפילות זו הוא שוב שימוש ב pre-compute להאצת זמן טעינה של הדף. גם כאן התוצאה היא זהה כל עוד לא מתווספות הערות בכל כניסה.
ii. ישנם מספר חסרונות. ראשית אנו מחזיקים מידע כפול שמבזבז מקום. שנית, אנחנו חייבים לבנות מנגנון עדכון (לדוגמה trigger) שיעדכן את המספר מתווספות הערות.
כפילות היא פתח לאי תאימות בין שתי הטבלאות. במנגנוני העדכון עלולים להיות באגים. בשל כך פעמים רבות כותבים מנגנוני ניטור כדי שיאתרו אי הסכמות במקום להניח שמנגנוני העדכון תקינים.

iii. מציאת אי הסכמות ע״י קוד:

```
SELECT
p.Id AS post_ID,
 max(CommentCount)  as commet_count_field,
IF(COUNT(c.Id) IS NULL, 0, COUNT(c.Id)) AS count_from_comments_tbl
FROM posts AS p
LEFT JOIN comments AS c
ON c.PostId = p.Id
GROUP BY p.Id
Having commet_count_field != count_from_comments_tbl ;
```

לצרכי ביצועים אנחנו שולפים את ערך השדה מטבלת post ומשווים לערך בהערות. הקיבוץ מחשב את מספר ההערות ואנו שולפים את ערך השדה ב max כי הוא ממילא יחיד פר פוסט. את ההשוואה אנחנו מבצעים ב having אחרי הקיבוץ.

מימוש מובנה יותר אך פחות יעיל הוא

```
SELECT
p.Id AS post_ID,
 CommentCount,
IF(COUNT(c.Id) IS NULL, 0, COUNT(c.Id)) AS count_from_comments_tbl
FROM posts AS p
LEFT JOIN
(select
id
, count(*) as count_from_comments_tbl
from
comments
Group by Id ) AS c
ON  p.Id = c.id
where CommentCount != count_from_comments_tbl
Or ( CommentCount !=0 and count_from_comments_tbl is null)
 ;
```

במימוש זה חשוב לטפל במקרים בהם אין הערות כלל, שכן פוסטים אלו לא יופיעו בצד ה join

lv.

כדי למנוע אי־הסכמות בין הנתון בשדה CommentsCount שבטבלת posts לבין הנתונים בטבלת comments, ניתן לבחור באחד משני כיוונים:

1. שמירת הנתון בטבלה אחת בלבד – למחוק את השדה CommentsCount מטבלת posts ולחשב את מספר התגובות בכל גישה לדף באמצעות טבלת הערות. זו דרך פשוטה להמנע כפילות במידע. דרך זו מוותרת על התועלת של ה pre-compute ועצם קיום השדה מראה ש Stack Overflow רואים בה לא כדאית או אפילו אפשרית.
2. שמירת הנתון בשדה ייעודי ואכיפת עדכון אוטומטי – להשאיר את השדה CommentsCount בטבלת posts אך לדאוג לעדכונו בכל שינוי רלוונטי, למשל באמצעות טריגר בבסיס הנתונים או לוגיקה במערכת שמעדכנת את האתר. בנוסף, כדאי להפעיל ניטור שיזהה מקרים של אי־התאמה בין הנתון לבין החישוב בפועל. פתרון זה יעיל יותר בביצועים, במיוחד במערכות עם דרישה למהירות תגובה גבוהה.