פתרונות לשאלות חידוד של חלק העיצוב בבחינת 2025
מיכל בוכבינדר

1. לרוב המוחלט של ספרים יש סופר יחיד, למיעוט יש סופרים מרובים לאותו ספר.
   a. היחס בין ספרים וסופרים בעולם הוא רבים לרבים (MANY TO MANY), אמנם אין הרבה ספרים שלהם כמה כותבים, אך ישנם ספרים כאלו.
   הרבה ספרים יכולים להיכתב ע"י סופר אחד, אך לא הרבה סופרים יצרו ספר בודד.
   b. למרות שבעולם האמיתי נוכל לראות מקרים של הרבה סופרים שכותבים ספר אחד, בעיני עדיף לבחור ביחס מצמצם שלא ישפיע על רוב הנתונים ואני אבחר ביחס של סופר אחד לספר אחד (הרבה ספרים יכולים להיכתב ע"י סופר אחד). אם נבחר ליצג יחס של סופר יחיד לספר, אין סיבה ליצור טבלה שתקשר במיוחד בין ספרים לסופרים. ניתן ליצור טבלת BOOKS, ובה עמודת AUTHOR שתייצג את הסופר עבור כל ספר.
   היתרון בכך הוא חיסכון במקום (במקום שניצור טבלת BOOKS_AUTHOR, הסתפקנו בעמודה).
   חסרונות לשיטה זו, הוא איבוד מידע (לא נייצג את המקרים המועטים בהם הרבה סופרים כותבים ספר), וגם מאחר וויתרנו על טבלת קישור המחברת בין FK של ספרים ל-FK של סופרים, יש יותר סכנה בשגיאות (ייתכן ושם הסופר יכיל שגיאת כתיב, וגם רגישויות לאותיות קטנות/גדולות) מה שעלול לגרום לאנליזה לא מדויקת (כמו מספר הספרים שסופר כתב).
   c. אם נבחר לא להציג את היחס בטבלה נפרדת, אייצג את הקישור בטבלת BOOKS עם צירוף עמודה נפרדת - נשים לב שכללי הנירמול נשמרים (יחס של אחד לרבים).
2. יש ספרים שהסופר שלהם אינו ידוע.
   a. לדעתי לא, מאחר והדבר יאפשר שגיאות רבות במערכת, ובסעיף הבא ישנו פיתרון עדיף לבעיה זו.
   b. במקום להגדיר שדה NULL, ניתן לרשום UNKNOWN - הבעייתיות בדרך זו היא שUNKNOWN יחשב ככותב של כלל הספרים הללו.
   c. ניתן לציין "מחברו של X". כך הסופרים הלא ידועים יהיו עם ישות אך לא ישות אחת מאוחדת. כך, ברור שלא ידוע מיהו הסופר, והדבר לא יאפשר בלבול בין מקרים בהם הסופר אינו ידוע למקרים בהם קרתה שגיאה בהזנת המידע למערכת.
3. הועלה הרעיון לפצל את טבלת ספרים לטבלת ספרים ולטבלת פרטי הספר - אך זהו רעיון מאוד לא מומלץ, עדיף לייצג ישות בטבלה אחת מכיוון שהיחס בין ספר לפריטו הוא אחד לאחד.. לעיתים יש אילוצים. לדוגמה, אנחנו נחזיק סריקה דיגיטלית של הספר בנפח גבוה מאד. לרוב השימושים אין צורך בה ועדיף שתהיה בצד כדי לא לפגוע בביצועים.
   a. היחס בין ספר לפרטי הספר הוא בדומה ליחס הספר והסופר - אחד לאחד.
   יחס הספר לכלל פרטיו יהיה מעין מפתח זיהוי ייחודי - ולכן הספר ביחס לכלל פרטיו יהיה יחס של אחד לאחד.

   b. **היתרונות:**
   - **ביצועים**: טבלת הספרים נשארת "צרה". יותר שורות בעמוד זיכרון/דיסק, פחות קריאות.
   - **שדות "כבדים" בצד**: טקסטים ארוכים (תקציר), שם הסופר, הוצאה, עוברים לטבלה נפרדת ולא מנפחים את טבלת הליבה.
   - **הרשאות והפרדה לוגית**: קל לתת גישה רק לטבלת הספרים ולחסום/להקשיח גישה לשדות רגישים/פחות נחוצים בטבלת הנתוני ספרים.

החסרונות:

- אי חיסכון במקום: (יצירת טבלה נוספת) ניתן לאחד את הנתונים לטבלה אחת, ועדיין לשמור על כללי הנירמול.
- קשה יותר להבנה למשתמשים מתחילים: המודל פחות אינטואיטיבי.
- **מורכבות בשאילתות**: אם רוב הבקשות צריכות "גם וגם", תצטרכי JOIN כמעט בכל מקום = יותר קוד/תחזוקה וגם יותר עלות.
- **סיכון לחוסר שלמות**: רשומת בטבלת ספרים בלי רשומת בטבלת נתוני ספרים (או להפך) היא חסרת משמעות. יש לאכוף 1:1 היטב.
- **אובר־נורמליזציה**: אם כמות "הפרטים" קטנה וקלה, פיצול לא באמת שווה את המורכבות.

4. הועלה הרעיון לייצג את מספר העותקים כעמודה בטבלת הספרים - רעיון פחות מומלץ.
   a. החיסרון בשיטת ייצוג זו, היא שלא נוכל לבצע מעקב איזה עותק קיים אצל איזה לקוח, ולמעשה לא נוכל להבדיל בין העותקים הזהים הקיימים בידי לקוחות שונים.
   למשל, בטבלת הספרים, כאשר נחפש את הספר "ההוביט", נשים לב כי ישנו מזהה הספר (ID), שם הספר ("ההוביט"), ומספר עותקיו (למשל, 2).
   אם, למשל שני לקוחות שונים השאילו את העותקים, ונניח שאחד מהעותקים הוא ממהדורה ישנה יותר, לא נוכל לדעת איזה עותק הושאל למי.
   בנוסף, תיווצר בעייתיות בטבלת ההשאלות - שני לקוחות שונים יכולים להשאיל באותו זמן ספר עם אותו ID? זוהי שגיאה.
   לכן – הדרך הנכונה תהיה לקיים טבלת עותקים (book_copies), שבה כל עותק יקבל מזהה ייחודי משלו (copy_id) עם קישור ל־book_id. כך ניתן יהיה לנהל השאלות, סטטוס, מיקום וכו, או לייצר book_id עבור כל ספר ועותק.
   b. **התשובה תלויה במבנה.**
   לאור הסעיף הקודם, עם שדה פשוט של "מספר עותקים" לא ניתן באמת לדעת כמה מהם הושאלו – אין מידע עליהם כפריטים נפרדים.
   במודל עם שדה מספרי בלבד אי אפשר לדעת כמה עותקים הושאלו או אצל מי, אולי יהיה אפשר לשער כמה נשארו להשאלה, אבל בלי דיוק.
   במודל עם טבלת עותקים, בהחלט אפשר לדעת כמה עותקים הושאלו – ע"י חיבור לטבלת השאלות ושאילתה כמו:

```
SELECT COUNT(*)
FROM loans
WHERE return_date IS NULL;
```

5. ניתן לייצג השאלה שלא הסתיימה ע"י שדה תאריך החזרה, או ע"י משתנה בינארי.
   a. החסרונות בתאריך החזרה:
   שדה בינארי, יכול להחליף את עמודת תאריך ההחזרה , אך הוא לא יציג לנו את תאריך ההחזרה של לקוח - לא נוכל לדעת כמה זמן לקוח משאיל כל ספר והאם יש צורך להאיץ בו.
   במקרה בו נשתמש בעמודה בינארית יחד עם תאריך החזרה, החיסרון יהיה עוד עמודת מידע שתתפוס מקום במקום עמודה אחת.

b. האילוצים שהייתי מציעה לעמודת תאריך ההחזרה הם - שיהיו מטייפ DATE, שיהיו קטנים בהכרח מהתאריך של היום או זהים לו , ושיהיה גדול מתאריך ההשאלה (לא לקבוע החזרה עתידית).
בעמודת תאריך ההשאלה הייתי מציבה אילוץ של NOT NULL, ושלא יהיה קטן מהתאריך הנוכחי (לא לאפשר השאלות עבר).

6. הבעייתיות בעמודת גיל היא שהיא אינה מתעדכנת אוטומטית. לכן, כדי לדעת את הגיל נשתמש בעמודת תאריך לידה, דבר שאינו מצריך תיחזוק.
7. אם הספריה מורכבת מכמה תתי ספריות (ספרית נוער וילדים לעומת ספרות בוגרת), היה צורך באובייקט ספריה שהיה מפריד בין הספריות השונות, ובשיוך הספרים אליהן.

8. היתרונות בהוספת ה"קישוטים", הוא שהדבר עלול לעזור בהמחשת והבנת קשרים נוספים בין ישויות ואובייקטים (אם הקשר בין סופר לספר לא היה מובן או אינטואטיבי בהתחלה, אולי השוואה בין הקשר של הוצאה לספר תחדד).
החסרונות, בזמן בחינה הוספת המידע "המיותר" עלול לגזול זמן.
השורה התחתונה - יש ערך בקישוטים הללו בעולם האמיתי, אלו יכולים להיות רעיונים טובים ויצירתיים.
אולם בבחינה זה עלול להיות מוקד לטעויות ולכן עדיף להימנע. אם למרות הכל ראינו בכך ערך, אז לנמק.

שאלות ייצוג כללי

1. אילוצים מומלצים על השדה מחיר:
   - Not null - אין היגיון בשורה המתארת מוצר או טרנזאקציה והיא ללא המחיר.
   - Check (price) > 0 למנוע ערכים שליליים או אפסיים.
   - לפי התחום העסקי, אפשר להחליט על ערך מקסימום ולאכוף אותו.
   - להגדרים שהערכים יהיו דצימליים (עם מספר ספרות מוגדר לאחר הנקודה.

   פרטי מידע שכדאי לדעת:
   - האם מדובר במחיר לצרכן? סיטונאי? האם יש מבצע?
   - באיזו מטבע מדובר? (אם יש צורך לקשר ל־Currency)
2. יתרונות:
   - גמישות גבוהה - מאפשר אחסון של מבנים שונים ללא שינוי בסכימה.
   - פשטות בכתיבה הראשונית - חוסך צורך בהקמה ותחזוקה של טבלאות משנה.

   חסרונות:
   - קושי בשליפות SQL - קשה למיין, לסנן או לבצע JOIN על ערכי JSON.
   - ביצועים - חיפושים בתוך טקסטים או JSON לרוב איטיים.
   - קושי בוולידציה - אין בקרה על מבנה/סוגי הערכים בתוך הטקסט. אין בקרה על המבנה שעלול להשתנות מרשומה לרשומה.

3. יש כמה דרכים לשמור כתובת בבסיס נתונים, ולכל אחת יתרונות וחסרונות משלה:
   כתובת בשדה טקסט אחד (VARCHAR):
   זוהי הדרך הכי פשוטה – כל הכתובת (רחוב, עיר, מיקוד וכו') מוזנת לתוך שדה אחד. זה נוח להזנה ולתצוגה מהירה, אבל מצד שני אין שליטה על מבנה הכתובת, קשה לחפש לפי חלקים מסוימים כמו עיר או מיקוד, וזה גם רגיש לשגיאות כתיב או הבדלים בפורמט.
   על מנת ליצור סדר יותר, ניתן לבצע
   חלוקה לעמודות נפרדות (למשל: רחוב, עיר, מיקוד, מדינה):

במקרה הזה כל חלק בכתובת נשמר בשדה משלו. זה מאפשר חיפושים מדויקים (למשל: כל הלקוחות מתל אביב), וגם שומר על אחידות. מצד שני, זה דורש יותר עבודה בהזנה ותחזוקה, וצריך ליצור התנייה שתוודא שכל השדות באמת מוזנים.
החיסרון בשיטה זו, היא שלא כל המדינות בעולם מובנות באותו האופן - ולכן יהיו המון עמודות עם NULLים בלתי נמנעים (למשל מקומות שבהם אין חלוקה ל-STATE אלא לREGION או ל-DISTIRICT).

דרך נוספת, היא באמצעות מיקומי GPS - פורמט זה נוח מאוד עבור שליחים, אך החיסרון בו הוא שהוא לא מאוד מובנה. במקרים בהם יש צפיפות בין בניינים עלול להיות חוסר דיוק, וגם לא ניתן לדעת קומה ומספר דירה אם מסתמכים אך ורק על פורמט זה (נשים לב שבWOLT נעזרים בשתי דרכי ייצוג המיקום).

4. **ייצוג:**

a. טבלה אחת בשם Animals, עם העמודות הבאות לדוגמה:
   - animalID
   - Name
   - Species
   - Type
   - Legs
   - HasFur
   - CanFly
   - LivesInWater
   - Weight

יתרונות:

- פשוטה ליישום – טבלה אחת.
- נוחה לשליפות כלליות על כל בעלי החיים.
- אין צורך ב־JOINים בין טבלאות.

חסרונות:

- הרבה ערכים יהיו NULL או לא רלוונטיים (למשל, CanFly לא רלוונטי לדגים).
- קשה לשמור על שלמות נתונים (למשל – דולפין הוא יונק אך אין לו רגליים,
- שדות רבים עלולים להיות מבלבלים ולא מתאימים לכל סוגי החיות.
- ישנם מקרים שדעת הציבור אינה בקורצליה עם דעת הביולוגים - למשל ליוויתן הוא בעל 4 רגליים ולא עם 0 כפי שמרביתנו היינו חושבים.
- טבלה מאוד עמוסה וארוכה ועקב כך מועדת להרבה שגיאות.

B. **ייצוג:**
טבלה Animals בסיסית עם שדות כלליים: AnimalID, Species, Type
ונייצר טבלאות נפרדות לכל תת־סוג:

- Mammals: שדות כמו HasFur, IsAquatic, Legs
- Birds: שדות כמו CanFly, Wingspan
- Fish: שדות כמו Gills, WaterType

יתרונות:

- התאמה טובה יותר למאפייני כל תת־סוג.
- מבנה נתונים "נכון יותר" ומנורמל.
- קל להוסיף מאפיינים ספציפיים לכל קבוצה.

חסרונות:

- שליפות מורכבות שדורשות JOIN על טבלאות שונות - ניהול מורכב יותר של המידע.
- קשה לבצע שאילתה אחידה על כל החיות.

C. תכונת מספר הרגליים רלוונטית לפי סוג הייצוג שאנחנו בוחרים - האם מדובר בייצוג של חיות ספציפיות (כמו בגן חיות) או בייצוג של מספר הרגליים לפי זנים. אם מדובר בייצוג לפי זנים, שמירה של השדה "מספר רגליים" אינה בהכרח רלוונטית וחשובה. במקרה והחלטנו כי ציון מספר הרגליים אינו מהותי לנו, ניתן לוותר על עמודת מספר הרגליים. במקרה ואנו מציגים את מספר הרגליים של חיות ספיציפיות ונרצה להתייחס אל החריגות, נוכל לשמור את מספר הרגליים לסוג חיה וניצור טבלה שתכיל את מספר הרגליים לחיות הספציפיות הבודדות שחורגות מהכלל - כך נחסוך במקום ונימנע מכפילויות.

מבנה טבלאות מומלץ:

במקום שדה מספר רגליים כללי, ניתן לשקול אחת מהחלופות הבאות:

1. **שדה קטגוריאלי במקום מספרי**:
   - לדוגמה: LimbsStructure = 'bipedal' | 'quadruped' | 'winged' | 'limbless
   - זה נותן תחושת כלליות, אך משמר את המידע בצורה אחידה יותר – וגם שומר מקום לחריגים.
2. **טבלת תכונות לפי קטגוריה ביולוגית וגם לפי תוכנת הזן הספציפי.**
   - נשמור על היררכיה של "מחלקות ביולוגיות" ונקשר לכל אחת מאפיינים קבועים (כגון מספר רגליים סטנדרטי, סוג תנועה ועוד).
   - נשמור את מספר הרגליים **ברמת הזן** (ברירת־מחדל לכל בני אותו סוג)
   - וניצור **טבלת חריגות לפרטים** בודדים בלבד (Animals_Overrides) שבה נרשום את מספר הרגליים רק למי שחורג מהכלל.

יתרון: מאפשר דיוק.
חיסרון: יוצר המון עומס, וחשוב לציין שישנם לא מעט זנים יוצאי דופן, אך הם לא הרוב.

עבור לוויתן נציין - 4 רגליים, עבור פינגווין נציין 0.
אם נרצה לספור עבור חיה ספציפית בגן חיות או שמורת טבע ולא עבור סוג זן, נוכל לנהל שדה נוסף – למשל HasFullLimbs או ליצור טבלת MedicalConditions.

D. נוכל לעשות שימוש בטבלת קשרים מרובים (many to many):
כך למשל נוכל ליצור טבלת abilities (למשל: Fly, Swim, Glide, Dive), וטבלת AnimalAbilities עם שדות AnimalID ו- AbilityID.
כך פינגווין יכול להיות גם Bird בטבלה אחת, וגם עם יכולת Swim בטבלה נפרדת, בעוד יונק כמו עטלף יהיה עם Fly.

E. **כן, לפינגווין יש נוצות**

- **משמעות ביולוגית:** פינגווינים הם עופות, ולכן יש להם נוצות. במקרה הזה, הנוצה היא מאפיין של קבוצת־על (עופות), והופעתה בפינגווין מאשרת את שיוכו להיררכיה הזו – גם אם הפונקציונליות של הנוצה השתנתה.
- **השלכה היררכית:** היררכיית הקלאסים שומרת על עקרון ההכלה - הפינגווין יורש תכונות של "עוף" ולכן נשאר קונסיסטנטי עם ההיררכיה.

אם התשובה הייתה שלילית:

- **משמעות בעייתית:** זו עלולה לשבור את ההיגיון ההיררכי. אם הפינגווין הוא תת-מחלקה של עופות, וציינו שכל עוף צריך שיהיו לו נוצות – אז הפינגווין מפר את הכלל הזה.
- **בעיית עיצוב:** נדרשת התמודדות עם חריגות מהכלל, למשל יצירת קלאס חדש נפרד (ולא תת-קלאס של עוף), או הפעלת תנאים שמתירים חריגות - ראו פיתרון לשאלה 4C.

תכונת הנוצות מזכירה את תכונת מספר הרגליים (סעיף C).