קורס SQL למתחילים

הרצאה 10 – תרגיל – פתב"ס

גיבוי דטאבייסים

פקודה לגיבוי מסד נתונים, למשל בשם Sample, נראית כך:

```
BACKUP DATABASE [Sample] TO DISK = 'c:\tmp\baksSample.bak'
```

בהרצאה ראינו כיצד ניתן להרכיב רצף משפטי SQL, שמגבה את כל הדטאבייסים של המשתמש באמצעות שימוש ב- SELECT יחיד, ולאחר מכן להריץ אותו באמצעות EXECUTE.
כדי לקבל את שמות כל מסדי הנתונים של המשתמש, השתמשנו בשאילתה הבאה:

```
select name
from sys.databases
where database_id>5
```

כתבו קוד שפותח Cursor מול תוצאות השאילתה הזאת, ואז עבור כל שורה (כל שם של DB) מייצר את ה- string של פקודת הגיבוי (ב- syntax שמופיע למעלה) ומריץ אותה באמצעות EXECUTE.
לנוחיותכם, מצורף בנספח הטקסט של דמו השימוש ב- Cursor שהוצג בהרצאה.

פיתרון:

```
DECLARE dbCursor CURSOR FOR
SELECT name
FROM sys.databases
WHERE database_id >=5

OPEN dbCursor

DECLARE @dbName nvarchar(200)

FETCH NEXT FROM dbCursor INTO @dbName

WHILE @@FETCH_STATUS=0
BEGIN
	DECLARE @str nvarchar(max) = REPLACE('BACKUP DATABASE [$DBNAME$] TO DISK = ''c:\tmp\$DBNAME$.bak''', '$DBNAME$', @dbName)

	EXECUTE (@str)

	FETCH NEXT FROM dbCursor INTO @dbName
END


CLOSE dbCursor
```


להרצאות, תרגילים ופתרונות ניתן להיכנס לאתר הקורס.

```
DEALLOCATE dbCursor
```

הסכימה של הפיתרון הזה זהה לסכימה של הפיתרון שהוצג בהרצאה.
בשלב הראשון, אנחנו מייצרים את ה- cursor מול השאילתה. לאחר מכן, אנחנו פותחים אותו ומתחילים את האיטרציה. אנחנו מגדירים משתנה @str שמכיל את הפקודה שצריך לגיבוי דטאבייס, כאשר איפה שאמור להופיע השם אנחנו שמים $DBNAME בטקסט ועושים לזה REPLACE לשם של ה- DB בשורה שעליה מצביע ה- cursor.
אנחנו מריצים באמצעות EXECUTE ומקדמים את ה- cursor הלאה.
כמובן, צריך בסוף לא לשכוח לשחרר משאבים.

יחס של טוב

אנחנו מגדירים יחס "X ענה ל-Y" בין שני משתמשים X ו-Y, אם X ענה לשאלה כלשהי שנשאלה ע"י Y.

אנחנו יכולים לדמיין גרף מכוון שנוצר ע"י היחס הזה: אם אני נקודת ההתחלה שלו, אז יש לי צלע אל כל האנשים שעניתי לשאלה שלהם. לכל אחד מהם, יש צלע לכל האנשים שהוא ענה לשלה שלהם וכן הלאה.

כתבו שאילתה שבהינתן Id של משתמש מקור ופרמטר "עומק" שולפת את כל המשתמשים שאפשר להגיע אליהם בגרף ממשתמש המקור שניתן עד לעומק שהוגדר.

למשל, אם שחר ענה לגל ולאבישי, גל ענתה לאיציק ואיציק ענה לענת ולשחר, אז התוצאה תראה כך כאשר מתחילים משחר ומוגדר עומק של עד 10:

עומק	משתמש
0	שחר
1	גל
1	אבישי
2	איציק
3	ענת

שימו לב ששחר לא מופיע בעומק 3 (למרות שאפשר להגיע אליו בעומק 3 במסלול שחר->גל->איציק->שחר) כי אנחנו רוצים למנוע הופעת כאלה שכבר הגענו אליהם קודם בטיול בגרף, כדי להימנע ממעגלים (שיגררו למעשה שליפה אינסופית במקרה שלנו).

עשו את השאילתה הזאת מול הטבלה Posts, כאשר פוסט X הוא מענה לפוסט Y אם ה- ParentId שלו הוא ה- Id של Y.
שימו לב, שההיררכיה בתשובות היא רק רמה אחת: לשאלה יש מספר תשובות, אבל אין תשובות לתשובות.

התחילו את השאילתה ב-

```
DECLARE @sourceUserId int = 285462
DECLARE @maxDepth int = 4
```

רמז 1: הפיתרון שלכם צריך למעשה לבנות שאילתה באופן דינמי (בלולאה) ולהריץ אותה. **לא** נדרש שימוש ב-CURSOR-ים במקרה הזה.

רמז 2: כדי לאגד את התוצאות מכל הרמות, אתם תרצו להגדיר משתנה טבלה. יש להגדיר אותו **בתוך** ה- SQL שנוצר דינמית, כי לא ניתן לגשת מתוך SQL שמורץ עם EXECUTE למשתנה טבלה שמוגדר בחוץ (ניתן לטבלאות זמניות, אבל לא דיברנו עליהם, ואין סיבה שתצטרכו להשתמש בכך בתרגיל הנ"ל).

הערה (לידע כללי): ניתן לפתור את זה באמצעות משהו שנקרא RECURSIVE CTE. אנחנו לא דיברנו על זה בקורס, ואין סיבה שתעשו בזה שימוש (המטרה של התרגיל היא להשתמש ב- SQL דינאמי.

פיתרון:

בשביל לפתור שאלה כזאת אנחנו צריכים לחשוב מה הלוגיקה הכללית שאנחנו רוצים שתהיה לשאילתה הכללית שתורץ בסוף (זאת השאילתה שנייצר עכשיו) הלוגיקה היא משהו כזה:

1. תגדיר משתנה טבלאי של תוצאות, שהיא כמו הטבלה שאנחנו צריכים להחזיר: עמודה שאומרת מה העומק, עמודה של ה-ID של המשתמש ועמודה של השם שלו
2. תוסיף את הפרטים של המשתמש שממנו אנחנו מתחילים עבור עומק 0.
3. לכל מס' עומק מעומק 1 עד העומק המקסימלי:
   A. תשלוף את כל המשתמשים שקיבלו תשובה ע"י אחד מהמשתמשים מהעומק הקודם (העומק הנוכחי פחות 1)
   B. תוסיף אותם למשתנה הטבלאי
4. שלוף את התוכן של המשתנה הטבלאי.

הפיתרון נראה כך:

```
DECLARE @sourceUserId int = 285462
DECLARE @maxDepth int = 4

DECLARE @sourceUserDisplayName nvarchar(200) = (SELECT DisplayName FROM Users WHERE Id = @sourceUserId)
DECLARE @query nvarchar(max) = '
	DECLARE @res table (Depth int, UserId int, DisplayName nvarchar(200))
'

SET @query = @query + '
	INSERT INTO @res(Depth, UserId, DisplayName)
	VALUES(0, ' + CAST(@sourceUserId as nvarchar(10)) + ', ''' + @sourceUserDisplayName + ''')
'
DECLARE @i int = 1
DECLARE @innerQueryTemplate nvarchar(max)= '
		;with sourceUserIds$i$ as (
		SELECT UserId
		FROM @res
		WHERE Depth = $iminusone$
		)
		INSERT INTO @res(Depth, UserId, DisplayName)
		SELECT DISTINCT $i$, q$i$.OwnerUserId, u$i$.DisplayName
		FROM Posts ans$i$
		JOIN Posts q$i$ ON q$i$.Id = ans$i$.ParentId
		JOIN Users u$i$ ON u$i$.Id = q$i$.OwnerUserId
		WHERE ans$i$.OwnerUserId IN (SELECT UserId FROM sourceUserIds$i$) AND
			ans$i$.ParentId IS NOT NULL AND
			q$i$.OwnerUserId NOT IN (SELECT UserId FROM @res)
	'
WHILE @i <= @maxDepth
BEGIN
	DECLARE @istr nvarchar(5) = CAST(@i as nvarchar(5))
	DECLARE @iminusstr nvarchar(5) = CAST(@i-1 as nvarchar(5))
	SET @query = @query + REPLACE(REPLACE(@innerQueryTemplate, '$i$', @istr), '$iminusone$', @iminusstr)
```

```
	SET @i = @i + 1
END
SET @query = @query + '
	SELECT *
	FROM @res
'
PRINT @query
EXECUTE (@query)
```

הפיתרון הזה למעשה מייצר שאילתה שפועלת כמו הפיתרון שתארנו מקודם. כאשר הסיבה שיש קוד מסביב (ולא כתבנו ישר את השאילתה) כי אנחנו מייצרים אותה בלולאה, בהתאם לכמה עומק אנחנו נרצה להגיע אליו.

למעשה, תרחיש של שאילתות ביחסי אבא-בן, כשרוצים לשלוף לעומק ולעשות דברים כמו שעשינו עכשיו, הוא תרחיש שבו נפוץ מאד SQL דינאמי ופעמים רבות גם עדיף **משמעותית** בביצועים שלו לעומת Recursive CTE – שזה הפיתרון ה- set based לבעייה.

נספח – דוגמא לשימוש ב- Cursor

המטרה: להחזיר result-set שכולל שורה עבור כל משתמש שפירסם שאלה עם ה- Id שלו, שמו, וכותרות השאלות שפירסם מופרדות ב- |~|

השיטה:

1. הגדרת משתנה טבלאי שיכיל את התוצאה
2. הגדרת Cursor מול שאילתה שמביאה את ה- Id וה- DisplayName של כל המשתמשים שפירסמו שאלה
3. פתיחת ה- Cursor
4. הגדרת משתנים שמכילים את הערכים של שורה בודדת
5. הבאת השורה הראשונה
6. בלולאה – כל עוד הצלחנו להביא מידע
    1. ביצוע הפעולה שאנחנו רוצים ויצירת ה-string שאנחנו רוצים (שרשור הכותרות)
    2. הכנסת התוצאה למשתנה של התוצאות
    3. הבאת השורה הבאה (קידום הלולאה)
7. סגירת ה- Cursor ושחרורו
8. שליפה של התוצאות

```
--1
```

```sql
DECLARE @results table (UserId int, DisplayName nvarchar(200), SpecialString nvarchar(max))

--2
DECLARE usersCursor CURSOR FOR
SELECT Id, DisplayName
FROM Users
WHERE Users.CreationDate BETWEEN '2016-01-01' AND '2016-02-01' AND
        EXISTS (SELECT NULL FROM Posts WHERE PostTypeId = 1 AND Posts.OwnerUserId = Users.Id)

--3
OPEN usersCursor

--4
DECLARE @currentUserId int, @currentUserDisplayName nvarchar(200)

--5
FETCH NEXT FROM usersCursor INTO @currentUserId, @currentUserDisplayName

WHILE @@FETCH_STATUS=0 --6
BEGIN
       --6.1
       DECLARE @str nvarchar(max) = ''

       SELECT @str = @str + '|~|' + Title
       FROM Posts
       WHERE PostTypeId=1 AND OwnerUserId = @currentUserId

       --6.2
       INSERT INTO @results(UserId, DisplayName, SpecialString)
       VALUES (@currentUserId, @currentUserDisplayName, @str)

       --6.3
       FETCH NEXT FROM usersCursor INTO @currentUserId, @currentUserDisplayName
END

--7
CLOSE usersCursor
DEALLOCATE usersCursor

--8
SELECT *
FROM @results
```