

יום מספר 4

שימוש במילת המפתח - **LIKE** - שני תוים בשימוש _ (קו תחתון) ו- % . קו תחתון מייצג תו בודד כלשהו ו- % מייצג בין 0 לאינספור. למשל, מצא את כל הסטודנטים ששםם מתחיל באותיות: Ma, או מצא את כל הספריות שנימצאות בערים שיש בשם העיר האות y. (LIKE 'Ma%', LIKE '%y%')

יש גם את האפשרות לבחירת תו ממחרוזת: LIKE '[abcd]%' אחד מ- a, b, c ואח"כ כל דבר. אפשרי גם להוציא תוים מסויימים על ידי: LIKE '[^abc_]'

קומבינציה של AND ו- OR

```
select * from CUSTOMER
where LastName='Smith' and (FirstName='John' or FirstName='Smith')
```

עבור שיפור ביצועים

SELECT TOP number/percent

```
SELECT TOP 2 * FROM Customers
SELECT TOP 50 PERCENT * FROM Customers
```

נחזור על שימוש במילת המפתח - **DISTINCT**

```
SELECT DISTINCT column1, column2,.....columnN
FROM table_name
WHERE [condition]
```

כך נימנע משליפת שורות כפולות. דוגמה לשימוש: שלוף רשימה של כל הערים שבהן גרים סטודנטים, ללא חזרה על עיר פעמיים או יותר.

```
select distinct scity
from dbo.student
```

פקודת UNION מאפשרת לנו לאחד יותר משאלתה אחת. שתיהן צריכות להיות זהות מבנית (אותו מספר עמודות ורצוי אותו סוג ערכים) כדי שזה יעבוד. הן יכולות להיות מטבלאות שונות. זה לא כ"כ נפוץ ולכן ניראה רק דוגמה טריביאלית שעבורה UNION לא ממש נחוץ:

קודם ניצור את בסיס הנתונים workplace1 ע"י הרצת: (createdbs/workplace1_person_create)

```
use workplace1
create table person
(
  personID int not null,
  lastName varchar(30),
  firstName varchar(20)
)
```

ונכניס לו נתונים (אם עדיין לא קיים) בעזרת:

```
populate/SQLQuery_workplace1_insert_person.sql
```

```

use workplace1
insert into person
(personID, lastName, firstName, gender)
values
(274, 'Kidman', 'Nicole', 'f'),
(275, 'Einstein', 'Albert', 'm'),
(276, 'Cohen', 'Yossi', 'm')

```

הרץ את: createdbs /SQLQuery_workplace_person_alter002.sql

וכעת לפעולת ה- UNION

```

select fullname as NAME
from dbo.person
where personID = 274
union
select fullname as NAME
from dbo.person
where personID = 276

```

איך הייתם משיגים אותה תוצאה בדרך אחרת? (רמז: אתם כבר יודעים כמה דרכים עבור זה).

כעת נמשיך לדבר על join

מה ההבדלים בין: INNER JOIN ו- FULL JOIN, RIGHT OUTER, LEFT OUTER?

בעיקרון משתמשים בסוגים אחרים של JOIN (אחרים מ- INNER) כאשר יש מקרים שלא בטוחים באמינות הנתונים או שהנתונים באים ממקורות שונים ורוצים לתאם ביניהם. (day4_outer_join.sql)

דוגמה לשימוש:

נתונות הטבלאות הבאות:

Employee		Location	
EmpID	EmpName	EmpID	EmpLoc
13	Jason	13	San Jose
8	Alex	8	Los Angeles
3	Ram	3	Pune, India
17	Babu	17	Chennai, India
25	Johnson	39	Bangalore, India

שימו לב שבטבלה Employee ישנו עובד בשם Johnson שאיננו נמצא בטבלת המקומות וכן שישנו עובד בעל מספר זיהוי 39 בטבלת המקומות שאינו נמצא בטבלת העובדים.

אם נכתוב Join פנימי (INNER JOIN) על מספר זיהוי עובד, יושמטו עובדים 25 ו-39. נסו זאת.

אבל אנחנו מעוניינים להתאים בין שמות העובדים ומיקומם, ורוצים לדעת גם אם לא ידוע מיקומו של עובד. הטכניקה תהיה להשתמש במשהו שניקרא: LEFT JOIN או LEFT OUTER JOIN

לדוגמה:

```
select * from employee e
left outer join location l
on e.empID = l.empID;
```

אפשר גם להשמיט את מילת המפתח: OUTER ולכתוב:

```
select * from employee e
left join location l
on e.empID = l.empID;
```

התוצאה תהיה:

e.EmpID	e.EmpName	l.EmpID	l.EmpLoc
13	Jason	13	San Jose
8	Alex	8	Los Angeles
3	Ram	3	Pune, India
17	Babu	17	Chennai, India
25	Johnson	NULL	NULL

המילה LEFT מרמזת לו שמהטבלה שנימצאת בצד שמאל של ה- join ילקחו גם ערכים שאינם תואמים לטבלה מימין. דבר דומה יקרה אם ניכתוב:

```
select * from employee e
right outer join location l
on e.empID = l.empID;
```

e.EmpID	e.EmpName	e.EmpID	l.EmpLoc
13	Jason	13	San Jose
8	Alex	8	Los Angeles
3	Ram	3	Pune, India
17	Babu	17	Chennai, India
NULL	NULL	39	Bangalore, India

תנחשו מה יקרה אם ניכתוב:

```
select * from employee e
full outer join location l
on e.empID = l.empID;
```

פונקציות נוספות

day4_substr

```
SELECT x = SUBSTRING('abcdef', 2, 3);
```

התוצאה של x תהיה: bcd

```
USE library;
```

```
SELECT isbn, SUBSTRING(title, 1, 10) AS 'Abbreviated'  
FROM book
```

בחירה של חלק מעמודה. שימושי כאשר העמודה היא מחרוזת. הפרמטר הראשון הוא העמודה, השני מיקום - מתחיל לספור משמאל (מבוסס על 1 כתו השמאלי ביותר) והפרמטר השלישי הוא האורך הרצוי.

החזר את 5 התוים השמאליים של העמודה:

day4_substr

```
SELECT LEFT(author, 5)  
FROM book
```

ועבור תוים מימין יש כמובן את: RIGHT

- LTRIM/RTRIM - מיועדות לחתוך רווחים מימין או משמאל של תוכן העמודה.

דוגמאות להסבת סוגי נתונים מסוג אחד לאחר:

```
-- Use CAST and CONVERT
```

```
USE bank;
```

```
select account_number, balance
```

```
FROM dbo.account
```

```
WHERE CAST(balance AS char(7)) LIKE '3%';
```

```
select account_number, balance
```

```
FROM dbo.account
```

```
WHERE CONVERT(char(7), balance) LIKE '3%';
```

```
(day4_ltrim_cast)
```

GROUP BY

נניח שרצינו לדעת מהטבלה של הספריות שמשייכת סיפריה לעיר, כמה ספריות נימצאות בכל עיר בטבלה. על ידי הסתכלות, אנחנו יכולים להבחין שניו יורק די שולטת, אבל אנחנו רוצים מספר מדויק של כולן.

ישנו מבנה מיוחד ב-SQL שמאפשר **צבירה של שורות על פי שוויון בשדה מסוים**, ולמעשה הצגה של סיכום. במקרה שלנו תחשבו על זה כעל מצב שבו אנחנו 'מכווצים את הטבלה' לשורות סכום עבור ערים, כלומר ליד כל עיר, ניתן רק את מספר הספריות שבה, ללא פרטים נוספים.

הטכניקה היא על ידי כתיבת שאילתה מהסוג הבא:

```
select city, count(*) as 'Number of libraries'
from dbo.library
group by city
```

יש לזכור שבשאילתה מסוג זה יכולים להישלף אך ורק השדות עליהם מסכמים (במקרה הזה שדה אחד ושמו **city**) ובנוסף שדה הסיכום עצמו.

תחשבו אם יש הגיון בנסיון לשלוף באותה שאילתה גם את שם הספרייה.

נוסיף עוד טוויסט לעלילה. כעת רוצם אותו דבר כמו קודם, אולם נירצה להציג רק את הערים בהן יש לפחות 2 ספריות.

```
select city, count(*) as 'Number of libraries'
from dbo.library
group by city
having count(*) >=2
```

טוויסט נוסף בעלילה: בנוסף למה שהיה עד כה, נירצה גם את הספריות רק מערים ששמן מתחיל ב-S זה משהו שכבר למדנו, אבל כעת אתם צריכים לשלב גם את ה-group by וגם את ה-where. תנסו.

```
select city, count(*) as 'Number of libraries'
from dbo.library
where city like 'S%'
group by city
having count(*) >=2
```

למי זה הצליח? אם כן היכן מיקמתם את משפט ה-where?

תזכרו שה-having עבור ה-group by הוא למעשה כמו ה-where עבור ה-select. פשוט מגביל את הבחירה. ולכן הם גם צריכים להיות צמודים בהתאמה, כ"א אחד לאן שהוא שייך.

ישנו עוד סוג של group by מלבד ספירת מספר השורות שעונות על שדה הסיכום. זה קורה במצב של עמודות שניתנות לסיכום. לדוגמה בטבלה Available ישנו שדה שניקרא: noOfCopies. אם נירצה לדעת עבור כל ספר (מספר זיהוי) כמה עותקים ישנם ממנו, בכל הספריות הקיימות - איך לדעתכם תיראה השאילתה?

ובאותה נשימה, את כל הספרים עבורם יש יותר מ-4 עותקים כ"א.

לכתוב שאילתה שתיתן את רשימת מספרי זיהוי הספרים (isbn) של הספרים שנישאלו על ידי שלושה סטודנטים לפחות, מהספריות שנימצאות בניו יורק.

פיתרון

מצד אחד יש לנו את המשימה לבדוק את טבלת ההשאלות - borrow כיוון שרוצים לבדוק ספרים שהושאלו מספר פעמים (מעל 3), אבל מצד שני רוצים רק מספריות שנימצאות בניו יורק, ובטבלה זו למרבה הצער, אין לנו את הערים של הספריות, אלא רק את שמותיהן (LNAME (השדות הם: ID, ISBN, LNAME)

ולכן על מנת קודם כל, בשלב ראשון, למצוא רק את הספרים שהושאלו מספריות ניו יורקיות, עלינו לחבר עם: join בין הטבלה borrow והטבלה: library (שבה יש שדות: LNAME ו-CITY), בצורה הבאה (inner join על isbn וגם שהעיר היא ניו יורק):

```
select isbn
from borrow b, library l
where l.lname=b.lname
and l.city='New York'
```

day4_ny_borrow

שאלתה זו מבודדת לנו את כל הספרים בטבלה: borrow (כלומר אלה שנישאלו) מספריות בניו יורק. זהו השלב הראשון. כעת המשימה בשלב השני היא "לקבץ" את הספרים שנישאלו מספר פעמים שווה או יותר מ-3, ולכן נעזר בהוראת ה-GROUP BY שלמדנו, ונפעיל אותה על הרשימה שהתקבלה מהשלב הראשון (כלומר על השאלתה שכתבנו כבר) בצורה הבאה:

```
select isbn, count(*)
from
  ( select isbn
    from borrow b, library l
    where l.lname=b.lname
    and l.city='New York'
  ) as subq1 -- Alias for the sub-query
group by isbn
having count(*)>=3
```

day4_subquery1

שימו לב מה קרה כאן. הפכנו את השאלתה מהשלב הראשון, לתת-שאלתה ומיקמנו אותה בתוך שאלתת ה-group by, כלומר התייחסנו אל השאלתה מהשלב הראשון כמו לטבלה עצמאית.

מה קיבלנו בשלב השני? רשימה של מספרי ספרים ועל יד כל מספר ספר את מספר הפעמים שהוא מופיע ב-borrow (כלומר הושאל) ואלה כמובן רק מספריות ניו יורקיות.

כעת מה שנותר לעשות זה משהו פשוט. הרי בתוצאה הסופית לא מעניין אותנו כמה סטודנטים בדיוק שאלו ספרים אלה, אם זה 3 או 7 או 100, רק את מספרי הספרים, ולכן מכל הקונסטרוקציה שבנינו, אנחנו ניצור תת-שאלתה, וכעת נכתוב:

```
select isbn
from
  ( כל הקונסטרוקציה )
```

ולכן התשובה הסופית תהיה:

```

select subq2.isbn
from (
  select isbn, count(*)
  from (
    select isbn
    from borrow b, library l
    where l.lname=b.lname
  ) subq1
  group by isbn
  having count(*)>=3
) subq2

```

day4_sub_sub_query

שאלתה נוספת שאפשר לנסות

לכתוב שאלתה שתיתן את רשימת כל הסטודנטים ששאלו את כל הספרים ששאל הסטודנט בעל ID של '384'

אם הסתבכתם עם זה. כדי לצאת מההסתבכות, אם שאלתה טיפה מורכבת, הייתי ממליץ לנתח ולפרק אותה לגורמים. לעיתים קרובות יש להתחיל מהחלק הפנימי והחוצה. במקרה זה קודם נכתוב מהם כל הספרים ששאל הסטודנט בעל ID של '384'.

```

select isbn
from borrow
where ID='314'

```

ישנו רק ספר אחד ששאל סטודנט זה: 0802134297

כעת נותר לנו לבחור מאותה טבלה (borrow) את רשימת הסטודנטים (ID) אבל כאשר לכל סטודנט הושאלו (לפחות) כל הספרים שהצגנו בשאלתה הפנימית. לשם כך עלינו להשתמש בפקודה שניקראת: ALL משולבת עם סימן השיוויון. זה אומר: תן לי את הערך שמתאים לכל הערכים בתת-השאלתה.

```

select ID
from borrow
where isbn = ALL
(
  select isbn
  from borrow
  where ID='314'
)

```

day4_all_any

כעת זה נראה פשוט, לא?

שימו לב שיצאו עוד 3 סטודנטים ששאלו את ספר. אם נשנה ל '384' יהיה רק אחד.

אם נשנה את ה- ALL ל- ANY וגם את ה-ID ל-295, נקבל רשימה ארוכה של סטודנטים - מישהו יודע מדוע?

תראו כמה ספרים 295 לקח (ולכן אם רוצים את אלה שלקחו איזה ספר שהוא מרשימה זו, יהיו הרבה כאלה)

תת-שאלות, או שאלות פנימיות או מקוננות (Sub queries)

אלה שאלות שנימצאות בתוך ביטוי ה-WHERE של השאלה המרכזית (או החיצונית). תת שאלה מיועדת על מנת להחזיר ערך או ערכים שעל פיהם יכולה השאלה הראשית לבסס תנאים נוספים לבחירת שורות.

תת-שאלות ניתנות לשימוש במשפטי DELETE, SELECT, INSERT, UPDATE, ויכולות להכיל פעולות כמו =, >, <, <=, >=, BETWEEN, IN וכד'.

כללים לכתיבת תת-שאלה

היא חייבת להיות מוקפת בין סוגריים רגילים
לא ניתן להשתמש בפקודה ORDER BY (אפשר בחיצונית)
אם תת השאלה מחזירה יותר משורה אחת, ניתן להתייחס אליה רק עם פעולות מתאימות כגון IN
ניתן גם לכתוב תת שאלה על אותה טבלה כמו השאלה החיצונית.

שאלה - מצא את כל הספרים שכמותם בטבלת הזמינות גבוהה מהכמות הממוצעת של ספר בטבלת הזמינות.

(day4_above_avg)

```
SELECT ISBN
FROM Available
WHERE noOfCopies > (
  SELECT AVG(noOfCopies)
  FROM Available)
```

פקודת EXISTS - פקודה זו משמשת כקישור בין השאלה ותת השאלה. התנאי מתקיים אם לפחות שורה אחת בתת השאלה מוחזרת. מאד לא יעיל, כי תת השאלה מתבצעת בכל פעם מחדש על כל החזר של שורה בשאלה החיצונית. ניראה דוגמה בפיתרון עבודת הבית.

כתוב/כתבי שאלה שתתן את רשימת שמות הסטודנטים ששאלו ספר שכתב המחבר Sam Wang, וגם לא שאלו ספר של המחבר Sherman Chow

כתוב/כתבי שאלה שתתן את רשימת שמות הסטודנטים ששאלו ספר שכתב המחבר Sam Wang, וגם לא שאלו ספר של המחבר Sherman Chow

על מנת לפתור נשתמש, בין היתר במילת המפתח: Except אשר 'מחסירה' תוצאות שאלה אחת מאחרת בצורה הבאה:

```
select sname from student
where id in
  (select id
   from book bk, borrow br
```



```
where bk.isbn = br.isbn and author = 'Sam Wang')
except
select sname
from student where id in
(select id from book bk, borrow br where bk.isbn =
br.isbn and author = 'Sherman Chow');
```

day4_library_except

מה עם הרעיון הבא::

```
select sname from student
where id in
(select id
from book bk, borrow br
where bk.isbn = br.isbn and author = 'Sam Wang'
and author != 'Sherman Chow')
```

day4_library_except_failed

זה יתגל כפיתרון סרק כיוון שאותו שדה (עמודה) של author אם משויים אותו לערך אחד (Sam Wang) ומוסיפים תנאי נוסף של אי שיוויון של אותו שדה לערך אחר ('Sherman Chow'), הרי מובן מאליו שהתנאי השני אינו מוסיף להגבלת מספר השורות החוזר.

ולכן אין לנו ברירה, אלא לייצר שאילתה נוספת שתענה על התנאי השני בניפרד, ואז "להחסיר" את השנייה מהראשונה.

שאילתה שאולי הזכרנו

מצא את מספרי הזהות של כל התלמידים ששאלו לפחות ספר אחד מספריה שממוקמת בעיר מגוריהם (כלומר לכל תלמיד עיר מגוריו).

day4_q5

נורמליזציה

- נורמליזציה (או נירמול) - Normalization. הוא תהליך של הבאת מסד הנתונים למצב שהנתונים נישמרים עם כמה שפחות כפילויות.

- לרוב מדברים בתיכנון של בסיס נתונים על 3 צורות של נירמול (1NF, 2NF, 3NF). הראשונה מתיחסת לכפילויות מידע. טבלה היא מבנה דו-מימדי (שורות ועמודות). כל שורה מזהה דבר מה' שונה מכל שורה אחרת. למשל טבלה שבה יש עמודה עבור שם ועמודה עבור תאריך לידה. מטבע הדברים יתכנו שורות כפולות - זה אינו בצורה נורמלית ראשונה. פיתרון אפשרי להוסיף עמודה עבור מספר זהות.

כל עמודה צריכה להכיל ערך מסוג מסוים. אם בעמודה של תאריך הלידה פעם מופיע חודש ושנה ועבור שורה אחרת מופיעים יום, חודש ושנה - זה אינו בצורה נורמלית ראשונה. כמו כן באותו תא בטבלה אסור שיהיו מספר ערכים, למשל אם הייתה עמודה עבור עיר לידה, עבור כל אדם יהיה רק ערך אחד, לא יכול להיות משהו כמו: London, New York.
 עוד סוג הפרת מצב של 1NF, אם ישנן עמודות בטבלה בעלות אותה משמעות (חוזרות), לדוגמה אם ניצור טבלה בעלת עמודות חוזרות בכל אחד ילד של האדם שמתואר בשורה. מה החיסרון במצב כזה?

- המצב הזה אינו יציב, כי לכל אדם תיאורטית יהיה מספר שונה של עמודות. אם נניח שלא יכולים ליות יותר מ 20 ילדים, גם נזכו עמודות מיותרות על הרבה אנשים וגם אולי נחמיץ איזה בדואי שנשוי ל 4 נשים ויש לו 28 ילדים (או את גואל רצון)

- הפיתרון במקרה זה יהיה לחלק את הטבלה לשתי טבלאות. באחת נתונים של האדם בלבד, ובשנייה נתונים שמשייכים אדם לילדיו, כאשר בכל שורה מופיע האדם עם ילד אחר בטבלה השניה.

לפני הנירמול:

<u>Person ID</u>	<u>name</u>	<u>email</u>	<u>phone</u>	<u>child1</u>	<u>child2</u>	<u>child3</u>
0001	Itamar	it@gmail.com	05211111111	Sam	Dina	Maya
0002	Dimon	dd@yahoo.com	05022222222	Riva	Shani	Leeam

Persons (table)

<u>Person ID</u>	<u>name</u>	<u>email</u>	<u>phone</u>
0001	Itamar	it@gmail.com	05211111111
0002	Dimon	dd@yahoo.com	05022222222

PersonChildren

<u>Person ID</u>	<u>Sequence</u>	<u>Child</u>
0001	01	Sam
0001	02	Dina
0001	03	Maya
0002	01	Riva
0002	02	Shani
0002	03	Leeam

היתרונות של הצורה הנורמלית הראשונה ברורים. אין צורך לשנות את מבנה הטבלה ולהוסיף עמודות אם לאדם יש יותר ילדים משלושה, אין ביזבוז של זיכרון במידה ולאדם ישנם פחות משלושה ילדים. כמוכן כפילות הנתונים היא מינימלית, (PersonID)

טבלה שנימצאת בצורה נורמלית ראשונה ומפירה את הצורה הנורמלית השניה, היא טבלה שבה יש תלות בין חלק מהמפתח הראשי ועמודות אחרות. למשל טבלה שבה יש שם עובד, כישורים וכתובת עבודה. המפתח הראשי מורכב משם + כישורים, אבל כתובת העבודה תלויה רק בשם העובד. דוגמה:

Employee skills

Name	Skill	Work Address
Brown	programmer	61 Main St, NY , NY
Tony	MD	14 E63rd St, Boston, USA
Brown	QA	61 Main St, NY , NY
Brown	Systems analysy	61 Main St, NY , NY

שימו לב שעבור Brown ישנה כפילות, מחזיקים את כתובת מקום עבודתו מספר פעמים.

כדי לפתור זאת נפרק את הטבלה לשתיים. באחת נישמור שם וכישורים (2 עמודות) ובשניה שם וכתובת (2 עמודות). כלומר צורה נורמלית שניה מכילה גם את הראשונה פלוס מה שהזכרנו.

Employees

Name	Work Address
Brown	61 Main St, NY , NY
Tony	14 E63rd St, Boston, USA

Employee skills

Name	Skill
Brown	Programmer
Brown	AQ
Brown	Systems analysy
Tony	MD

עבור צורה נורמלית שלישית ניראה דוגמה לטבלה שנימצאת בצורה נורמלית שניה ולא שלישית:

Tournament winners

Tournament	year	Winner	Winner DOB
Indiana Invitational	2010	Al Fredrickson	July 15, 1985
NY Open	2012	Bob Albertson	August 1, 1992

US Closed	2013	Al Fredrickson	July 15, 1985
NY Open	2014	Chip Masterson	November 3, 1990

כל שורה בטבלה זו צריכה לומר לנו מי ניצח טורניר מסוים ולכן עמודות שהן יכולות להיות מפתח למשל, הינן: **Tournament + year** (במינימום כדי להבטיח יחודיות)

העמודה של Winner DOB תלויה ב-Winner - עובדה שיכולה לגרום עידכונים לא עיקביים, כי תאריך הלידה יחזור על עצמו אם אותו אדם זכה בטורניר זהה בשנים שונות או באותה שנה בטורנירים שונים.

- זהו דפקט קצת קשה יותר לאבחנה, אחד השדות תלוי בשדה אחר שאינו מפתח ראשי (כלומר תלות עקיפה במפתח הראשי) - לשם כך נפצל את הטבלה לשתיים בצורה שאחת תכיל את המפתח הראשי (טורניר ושנה) ואת אחד השדות (המנצח) וטבלה שניה שמייחסת את המנצח לתאריך לידתו.

Tournament winners

Tournament	year	winner
Indiana Invitational	2010	Al Fredrickson
NY Open	2012	Bob Albertson
US Closed	2013	Al Fredrickson
NY Open	2014	Chicp Mastersin

Winner DOB

Winner	DOB
Al Fredrickson	July 15, 1985
Bob Albertson	August 1, 1992
Chicp Mastersin	November 3, 1990

הרעיון של הנירמול זה להפחית כפילות נתונים גם מבחינת שימורם וגם מבחינת עידכונם

ופישוט מקסימלי למבנה הטבלאות, אך ורק לנתונים ההכרחיים בכל טבלה ע"מ לשלוף כל סוג של נתון מהטבלאות. כל זה שייך לתחום של תיכנון בסיסי נתונים שבחלקו הינו מדע ובחלקו גם אמנות. יש סיכוי שנעבוד בחברות Startup ולכן כדאי לדעת מעט על נושא זה, גם אם זה לא העיקר של הלימוד.

ההגדרה ה"יבשה" של צורה שלישית נורמלית היא מתמטית אזוטרית, אבל בקירוב אפשר לומר כמו בבית משפט שכל עמודה שאיננה מפתח ראשי, חייבת לומר משהו על המפתח, על כל המפתח וכלום מלבד המפתח.

כיוון שהבחור מ.י.ב.מ. שהמציא את המודל היחסי קוראים לו : Codd, ישנה בדיחת קרש שבאנגלית נישמעת טוב: (עמודה מכונה גם: attribute)

Bill Kent: "[Every] non-key [attribute] must provide a fact about the key, the whole key, and nothing but the key."^[7] - so help me [Codd](#)!^[8]