



מערכות בסיסי נתונים

לאוניברסיטה הפתוחה

● מודל היחסים

● SQL

● תרשים ישויות קשרים

● תיכון במסד נתונים

● טיפוסי נתונים מורכבים

● אינדקסים וגיבוב

● עיבוד שאילתות



לקורס המלא ולעוד עשרות סיכומים בחינם:

Click קליק 
by Yehoran

<https://click-go-easy.click/>

לשאלות, הערות,

שיעורים פרטיים ושיעורים קבוצתיים:

יהורן - [0506798719](https://wa.me/0506798719)



מודל היחסים



סימונים מקובלים		
מושג	סימון	הסבר
אופרטור וגם	\wedge	משמש בתנאי של פעולת הבחירה לדרישת קיום שני תנאים או יותר
אופרטור או	\vee	משמש בתנאי של פעולת הבחירה לדרישת קיום לפחות אחד משני תנאים או יותר
אופרטור שלילה	\neg	משמש בתנאי של פעולת הבחירה לדרישת אי-קיום התנאי המופיע מימינו
פעולות שיויון	$=, \neq, <, >, \leq, \geq$	משמשות בתנאי של פעולת הבחירה
יחס	R	הטבלה בבסיס הנתונים עליה מבוצעת השאילתה
איחוד יחסים	\cup	יחס חדש הכולל את כל האיברים בשני היחסים
חיתוך יחסים	\cap	יחס חדש הכולל את האיברים המשותפים לשני היחסים גם יחד
הפרש יחסים	$-$	יחס חדש הכולל את האיברים ביחס משמאל בתנאי שאינם מופיעים ביחס מימין (פעולה שימושית לצורך מציאת ערך קיצון - מבצעים מכפלה קרטזית של היחס עם עצמו ומבקשים רק שורות בהם הערך הנדרש למקסימום גדול / קטן מהערך ביחס השני, בוחרים רק את הערך מיחס אחד ומקבלים את כל השורות מלבד השורה עם הערך הקיצוני הנדרש, מחסרים את התוצאה מכל השורות ומקבלים רק את הערך הקיצוני הנדרש)
בחירה	$\sigma_{condition}(R)$	הצגת כל השורות ביחס R המקיימות את התנאי condition
הטלה	$\Pi_{columns}(R)$	הצגת העמודות columns מתוך היחס R. פעולה זו מצמצמת שורות כפולות
מכפלה קרטזית	\times	חיבור כל אחת מהשורות ביחס אחד עם כל אחת מהשורות ביחס השני
צירוף טבעי	\bowtie	חיבור שני יחסים לפי התכונות בעלות אותו השם: $r(A, B, C) \bowtie s(B, C, D) = \Pi_{r.A, r.B, r.C, s.D}(\sigma_{r.B=s.B \wedge r.C=s.C}(r \times s))$
שכפול יחס	$\rho_{new\ name}(R)$	החזרה של היחס המקורי R תחת שם חדש
השמה	\leftarrow	הצבת יחס כלשהו (מימין) במשתנה חדש (משמאל) – נוח לפישוט של פעולות מורכבות
חילוק	\div או $/$	פעולה המוגדרת אך ורק בין שני יחסים אשר היחס המחלק (מימין) חלקי ממש ליחס המחולק (משמאל) – זאת אומרת, שביחס המחולק יש את כל התכונות שיש ביחס המחלק ועוד תכונות נוספות. הפעולה מחזירה רק את התכונות העודפות שיש ביחס המחולק, והיא מחזירה אך ורק שורות אשר הערך בתכונה העודפת ביחס המחולק מופיע עם כל אחד מהערכים ביחס המחלק.

SQL



פסוקי DDL (Data Definition Language) ב-SQL		
מפשט	תחביר	הסבר
יצירת טבלה	create table <i>table_name</i> (<i>column_name</i> <i>data_type</i> ,...)	לאחר הפתיח יש לרשום את שם הטבלה ולאחר מכן בתוך סוגריים עגולים את כל עמודות הטבלה עם פסיק מפריד בין אחת לשנייה. עמודות שהן מפתח יש לסמן ב-primary key. עמודות שאסור להן להיות ריקות יש לסמן ב-not null. עמודות שהן מפתח זר יש לסמן ב-foreign key ולכתוב מאיזה טבלה הן מגיעות.
מחיקת טבלה	drop table <i>table_name</i>	בכדי למחוק טבלה שלמה מבסיס הנתונים יש לרשום drop table <i>table_name</i> . אם רוצים לבקש למחוק את הטבלה רק אם היא קיימת יש לרשום drop table <i>table_name</i> if exists. כאשר יש טבלאות שיש בהן מפתח זר מהטבלה שאנו מנסים למחוק, אם אנחנו רוצים למחוק גם את הטבלאות האלו נרשום cascade בסיום המשפט אחרי שם הטבלה
הכנסת רשומה	insert into <i>table_name</i> values (' <i>value</i> ',' <i>value</i> '...),(...),(...)	בכדי להכניס רשומות חדשות לטבלה קיימת יש לרשום insert into <i>table_name</i> values, מיד לאחר ה-values יש לרשום את הערכים השונים בתוך סוגריים עגולים עם פסיק מפריד בין תכונה לתכונה. זהו כתיב מקוצר המחייב להכניס ערכים לכל התכונות לפי סדר הופעתן ביחס. אם רוצים להכניס מספר רשומות בבת אחת פשוט רושמים פסיק אחרי הסוגרים העגולים ופותחים סוגריים נוספים - כל סוגריים עגולים שכאלו הם שורה אחת ביחס.
עדכון רשומה	update <i>table_name</i> set <i>column_name</i> ='value' where <i>column_name</i> ='value'	עדכון של תכונות ספציפיות בתוך רשומה קיימת ביחס: update <i>table_name</i> set <i>column_name</i> ='value'. אם לא משתמשים בתנאי where (המופיע מיד אחרי התכונות שרוצים לשנות) שבו מגדירים איזה שורות יושפעו מהשינוי, כל השורות ביחס ישתנו! אם יש כמה שורות ביחס המקיימות את התנאי - כולן ישתנו.
מחיקת רשומה	delete from <i>table_name</i> where <i>column_name</i> ='value'	מחיקה של שורות שלמות הקיימות ביחס. אם לא משתמשים בתנאי where (המופיע מיד אחרי שם הטבלה ממנה מעוניינים למחוק) שבו מגדירים איזה שורות יימחקו, כל השורות ביחס יימחקו! אם יש כמה שורות ביחס המקיימות את התנאי - כולן יימחקו.
עדכון טבלה	alter table <i>table_name</i> change	אם רוצים להוסיף / למחוק / לשנות עמודה או להוסיף / למחוק / לשנות איזשהו אילוץ החל על עמודות כלשהן בטבלה, יש להשתמש בפקודה alter. השינוי יכול להיות: הוספת עמודה חדשה - ADD <i>column_name</i> <i>datatype</i> , מחיקת עמודה קיימת - DROP COLUMN <i>column_name</i> , שינוי שם עמודה - ADD <i>CONSTRAINT</i> , RENAME COLUMN <i>column_name</i> TO <i>new_column_name</i> , הוספת אילוץ - ADD <i>CONSTRAINT</i>
סוגי נתונים (data type)	int / integer	מספר שלם
	numeric(a,b)	מספר ממשי בהצגה עשרונית. כאשר a הוא סך הספרות שיופיעו במספר (משמאל ומימין לנקודה), ו-b הוא מספר הספרות שיופיעו אחרי הנקודה
	char(a)	מחרוזת באורך מדויק של a תווים
	varchar(a)	מחרוזת באורך מקסימלי של a תווים
	bool / boolean	אמת (1, true, yes, on) או שקר (0, false, no, off)
	date	תאריך - הפורמט המומלץ הינו: yyyy-mm-dd (לדוגמה, העשרים לנובמבר 88 ייכתב כך: 1988-11-20)
אילוץ על עמודות	primary key	מפתח ראשי - אם יש רק תכונה אחת שהיא מפתח של היחס אפשר לכתוב primary key מיד אחרי ה-data_type של התכונה, אם יש מספר תכונות שיחד הן מהוות מפתח של היחס, אחרי שמסיימים להגדיר את כל העמודות של הטבלה רושמים (primary key (<i>column_name</i> ,...))
	foreign key (<i>column_name</i>) references <i>table_name</i>	מפתח זר - בסיום ההכרזות על כל התכונות ביחס, עבור כל תכונה שהיא מפתח זר (מפתח של טבלה אחרת) רושמים (foreign key (<i>column_name</i>) ומיד אחר כך את הטבלה שממנה המפתח הזר לקוח באמצעות references <i>table_name</i> . אם רוצים שכאשר רשומה בטבלה שממנה לקוח המפתח הזר תימחק אז גם הרשומות שתלויות בה בטבלה זו יימחקו יש לרשום on delete cascade
	not null	לא ריק - אם רוצים לאסור מצב שבו התכונה ריקה יש לכתוב not null מיד אחרי ה-data_type של התכונה
	unique	ערך ייחודי - אם רוצים לאסור מצב ששתי רשומות שונות יכילו את אותו הערך (אבל לא רוצים להגדיר את התכונה כמפתח) יש לכתוב unique מיד אחרי ה-data_type של התכונה
	check (<i>column_name</i> ='value')	בדיקת ערכים - אם רוצים שרק ערכים מסוימים יוכלו להיכנס לעמודה אפשר לקבוע תנאי על העמודה באמצעות check, לדוגמה אם נרצה שיהיו בעמודה salary רק ערכים חיוביים, נרשום: check (<i>salary</i> >= 0)

SQL



שאלות SQL		
מושג	תחביר	הסבר
הטלה	select	בחירה של עמודות מסוימות מתוך היחס הנמצא ב-from
מייחסים	from	שם הטבלה/הטבלאות מהם השאילתה מבקשת לקרוא נתונים
תנאי על הרשומות	where	רק שורות המקיימות את התנאי יופיעו ביחס התוצאה (לא חובה להציב תנאי)
צמצום כפילויות	distinct	מצמצם שורות זהות. לדוגמא: <code>select distinct(column_name)</code> או לדוגמא: <code>select count (distinct (column_name))</code>
שרשרת תנאים	and, or	and משמש לדרישה שכל התנאים יתקיימו, or משמש לדרישה שלפחות אחד מהתנאים יתקיים
צירוף טבעי	natural join	מחבר שני יחסים יחד לפי התכונה/ות בעלת/ות אותו השם. חיבור זה יוצר טבלה אחת גדולה המכילה את כל התכונות המופיעות בשני היחסים. תכונות בעלות אותו שם מקבלות רק עמודה אחת ולכן לא חייבים לכתוב בבחירה מאיזה יחס התכונה מבוקשת. תחביר: <code>from table_A natural join table_B</code>
מכפלה קרטזית	,	מחבר כל שורה ביחס הראשון עם כל שורה ביחס השני: <code>from table_A, table_B</code> . תכונות בעלות אותו שם לא מתבטלות וביחס התוצאה יש שתי עמודות שונות עבור תכונות אלו ולכן חייבים לכתוב מאיזה טבלה מבקשים את התכונה: <code>select table_A.column_name</code>
נתינת שם ליחס	as	מאפשר לקרוא ליחס בשם מקוצר. שימושי מאוד במכפלה קרטזית בין שני יחסים שהם אותו היחס: <code>select A1.column_name from table_A as A1, table_A as A2</code> אפשר להשתמש גם אחרי שאילתה מקוננת ואז יש להגדיר את שמות העמודות של השאילתה: <code>Select A.c1 from (select column_name1, column_name2 from table_A) as A(c1,c2)</code>
פונקציות הקבצה (אסור להפעיל אחת על השניה)	count	מספר הערכים
	sum	סכום הערכים
	max	הערך המקסימלי
	min	הערך המינימלי
	avg	הערך הממוצע
פעולת הקבצה	group by	מחלק את הטבלה לקבוצות לפי תכונה/ות מסוימת/ות. כל התכונות המופיעות ב-select מחוץ לפונקציית הקבצה כלשהי, חייבות להופיע גם ב-group by. לא משתמשים בפעולת ההקבצה ללא שימוש באחת מפונקציות ההקבצה.
תנאי על הקבוצה	having	משמש לבדיקת תנאי על קבוצת ערכים. הבדיקה נעשית על פונקציית הקבצה והיא נעשית אחרי פעולת ההקבצה. הבדיקה עצמה נעשית אחרי הקיבוץ (לעומת הבדיקה על התנאים ב-where שנועשית לפני הקיבוץ)
שאילתות מקוננות	all, any	משמש לבדיקה אם ערך כלשהו גדול/קטן/שווה/שונה מכל הערכים שביחס המתקבל מהשאילתה המופיעה אחריו, any משמש לבדיקה דומה מערך כלשהו המופיע ביחס. לדוגמא: <code>column_name >= all(select column_name from table_A)</code>
	in, not in	משמש לבדיקה האם ערך כלשהו נמצא או לא נמצא ביחס המופיע אחריו. לדוגמא: <code>column_name in (select column_name from table_A)</code>
יצירת יחס חדש	exists, not exists	מחזיר "אמת" אם השאילתה המופיעה אחריו מכילה לפחות רשומה אחת. not exists מחזיר "אמת" אם השאילתה המופיעה אחריו ריקה לחלוטין.
	with as	משמש ליצירת יחס חדש לצורך שימוש עתידי בשאילתה אחרת. יעיל בשאילתות מורכבות. <code>with name(col1, col2...) as (select columnA as col1, columnB as col2 from table_name where....)</code>
	date_part	משמש לקבלת חלק מסוים מתוך תאריך. שנה: <code>date_part('year',dateName)</code> , חודש: <code>date_part('month',dateName)</code> , יום: <code>date_part('day',dateName)</code> .
פעולות על יחסים	current_date	מחזיר את התאריך הנוכחי ברגע שבו מבוצעת השאילתה. שימושי כאשר נדרש שהשאילתה תהיה נכונה בכל תאריך.
	union	איחוד שני יחסים יחד
	intersect	חיתוך שני יחסים (כל השורות המופיעות בשני היחסים גם יחד)
	except	חיסור יחסים (כל השורות המופיעות ביחס הראשון ולא מופיעות בשני)
		שלושת הפעולות מצמצמות שורות כפולות. כדי להימנע מזה אפשר לכתוב לדוגמא: <code>union all</code>

SQL



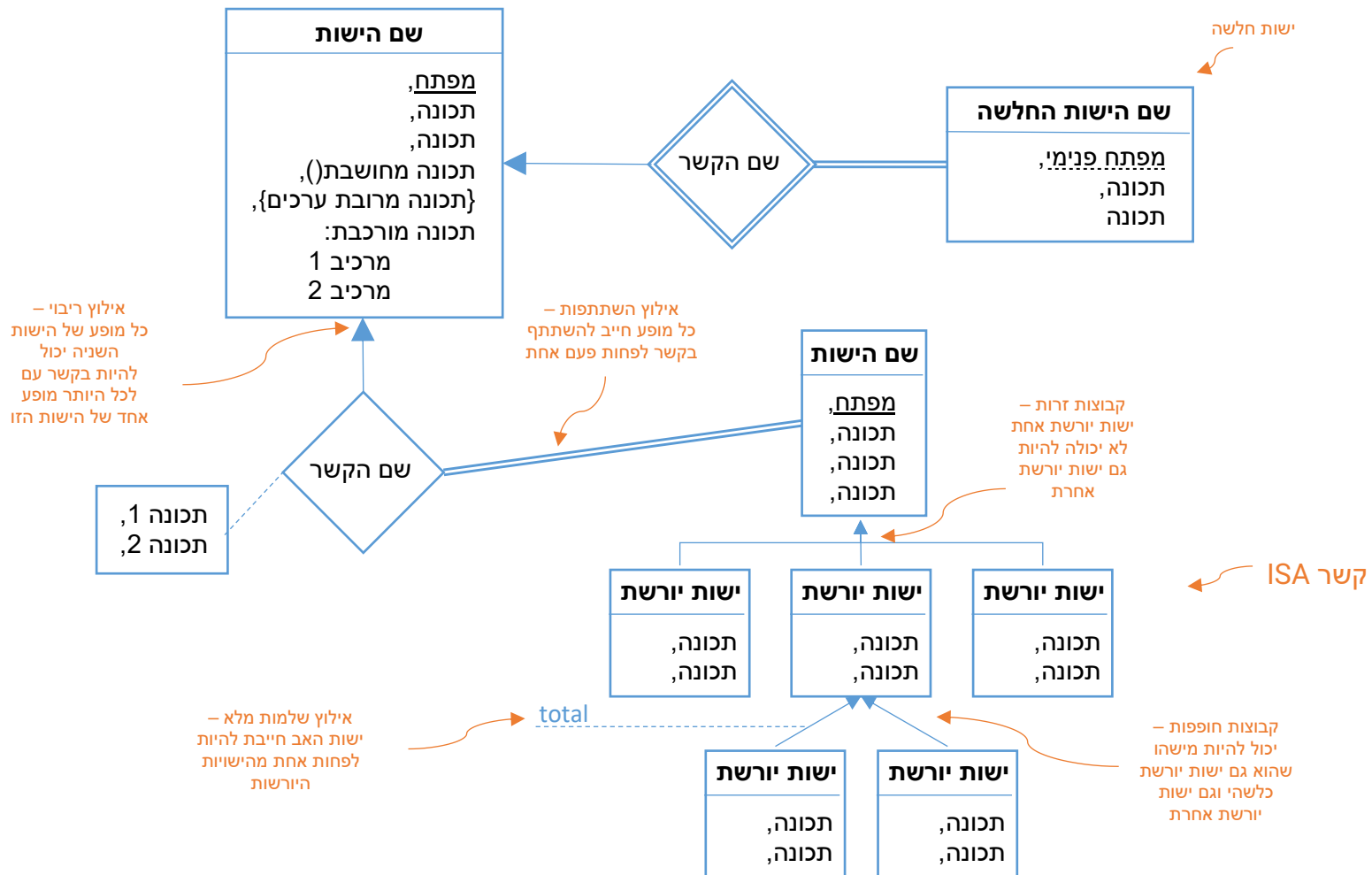
טריגר בפורמט PostgreSQL

פסק ליצירת הטריגר trig_name	create or replace function trig_name() returns trigger as\$\$
פתיחת אזור להכרזה על משתנים. אם אין שימוש במשתנים אין צורך ב- declare שמות המשתנים והסוג שלהם.	declare
<ul style="list-style-type: none"> משתנה מסוג numeric משמש למספרים עשרוניים, יש להגדיר את כמות הספרות בסך הכל (בדוגמא-4) וכן את כמות הספרות אחרי הנקודה (בדוגמא-2), לדוגמא עבור המספר 123.58 יש להגדיר: numeric(5,2). משתנה מסוג varchar משמש לקליטת מילים, יש להגדיר את אורך המילה המקסימלית האפשרית (בדוגמא-20). 	varName1 integer; varName2 boolean; varName3 numeric(4,2); varName4 varchar(20);
התחלת הפונקציה עצמה. גוף הפונקציה יכול לכלול לולאות for , משפטי if , ופעולות נוספות כמו הכנסה של רשומה ליחס כלשהו: insert into table_name values(A,B,C,D...)	begin
התחלת משפט "אם". בתוך התנאי אפשר לבדוק האם מתקיים משהו בתוך יחס הקיים במערכת (באמצעות שאילתה רגילה), אפשר להשוות את אחד הערכים בשורה החדשה שמנסים להכניס לבסיס הנתונים, לדוגמא: if(new.columnA in (select columnA from tale_name))	If(condition) then
הקפצת הודעת שגיאה למשתמש	raise notice 'error message for the user';
החזרת ערך ריק למשתמש ומניעת הכנסת הרשומה החדשה לבסיס הנתונים	return null;
סיום משפט "אם"	end if;
הכנסת הרשומה החדשה לבסיס הנתונים	return new;
סיום הפונקציה	end; \$\$ language plpgsql;
יצירת הטריגר T1 וקביעה מתי הוא יקרה (before/after/instead of) ועבור איזה סוגים של אירועים (הכנסה או עדכון של רשומה) ועל איזו טבלה הטריגר יופעל	create trigger T1 before insert or update on table_name
האם לבצע את הטריגר לכל שורה בנפרד: for each row , או לכל ההוראה ביחד: for each statement	for each row
אם רוצים להפעיל את הטריגר רק במקרים מסוימים אפשר להשתמש ב- when (לא חובה).	when (condition)
זימון הפונקציה והפעלתה (אפשר לשלוח לפונקציה פרמטרים לשימושה בתוך הסוגריים)	execute procedure trig_name();



תרשים ישויות קשרים

איך נראה תרשים אופייני?

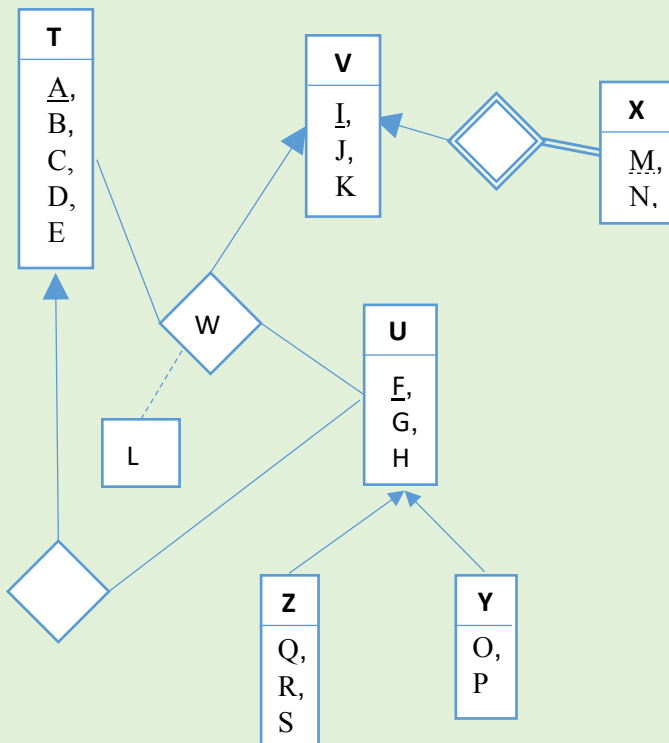


תרשים ישויות קשרים



דוגמא:

T(A,B,C,D,E)
 U(E,G,H,A)
 V(I,J,K)
 W(A,E,I,L)
 X(M,I,N)
 Y(E,O,P)
 Z(E,Q,R,S)



אלגוריתם לשרטוט דיאגרמה על בסיס רשימת יחסים נתונה	
1	יחס המכיל מפתח ייחודי משלו הופך לישות.
2	אם ביחס הישות קיימת תכונה שהיא מפתח ביחס ישות אחר (מפתח זר) אין צורך לכתוב את התכונה הזו כחלק מהתכונות של הישות כיוון שזה מעיד על קשר (רבים ליחיד) בין שתי ישויות אלו. נסמן חץ בכיוון של הישות שהמפתח שלה מופיע כמפתח זר בישות השניה (זו ש"הלוותה" את המפתח שלה לישות השניה).
3	יחסים המכילים מפתחות של שתיים או יותר ישויות אחרות הופכים לקשר בין ישויות אלו. אם יש תכונות נוספות הן הופכות לתכונות של הקשר. אם חלק מהמפתחות הזרים ביחס לא מסומנים כמפתח – נסמן חץ בכיוון של הישות שהמפתח שלה לא מסומן בקו תחתון ביחס הקשר.
4	כאשר ליחס יש שני מפתחות, אחד של עצמו ואחד של ישות אחרת, זה אומר שמדובר בישות חלשה: המפתח של עצמו הופך למפתח חלש (קו תחתון מקוקו) והישות מתחברת בקשר של ישות חלשה (קו כפול סביב הקשר, חץ בכיוון הישות החזקה וקו כפול בכיוון הישות החלשה) עם הישות שהיא מקבלת ממנה את המפתח הזר.
5	כאשר מספר יחסים בעלי אותו המפתח (ושאר התכונות שונות) או כאשר מספר יחסים בעלי אותו המפתח ובעלי מספר תכונות זהות ועוד תכונות נוספות שאינן זהות מדובר בקשר ירשה (ISA). במקרה השני מדובר בהכללה מלאה ולא חופפת.
6	לאחר בניית הדיאגרמה הבסיסית קוראים את המלל המסביר את המשמעות של רשימת היחסים ומפרט את האילוצים השונים החלים על היחסים ובהתאם מעדכנים את הדיאגרמה.



תיכון במסד נתונים

כיסוי קנוני

דוגמא:

$$R = (A, B, C, D, E, G)$$

$$F = \{A \rightarrow C, B \rightarrow DG, D \rightarrow E, BG \rightarrow AE, DE \rightarrow BG\}$$

בדיקת עודפות מצד ימין:

תלות	תכונה	סגור	עודפת	שינוי
$A \rightarrow C$	C	$A^+ = A$	לא	
$B \rightarrow DG$	D	$B^+ = BGAEC$	לא	
$B \rightarrow DG$	G	$B^+ = BDEGAC$	כן	$B \rightarrow D$
$D \rightarrow E$	E	$D^+ = D$	לא	
$BG \rightarrow AE$	A	$BG^+ = BGED$	לא	
$BG \rightarrow AE$	E	$BG^+ = BGACDE$	כן	$BG \rightarrow A$
$DE \rightarrow BG$	B	$DE^+ = DEG$	לא	
$DE \rightarrow BG$	G	$DE^+ = DEB$	לא	

$$F = \{A \rightarrow C, B \rightarrow D, D \rightarrow E, BG \rightarrow A, DE \rightarrow BG\} \quad \text{אוסף תלויות מעודכן:}$$

בדיקת עודפות מצד שמאל:

תלות	תכונה	סגור	עודפת	שינוי
$BG \rightarrow A$	B	$G^+ = G$	לא	
$BG \rightarrow A$	G	$B^+ = BDEGAC$	כן	$B \rightarrow A$
$DE \rightarrow BG$	D	$E^+ = E$	לא	
$DE \rightarrow BG$	E	$D^+ = DEBGAC$	כן	$D \rightarrow BG$

$$F = \{A \rightarrow C, B \rightarrow AD, D \rightarrow EBG\} \quad \text{אוסף תלויות מעודכן:}$$

בדיקת עודפות מצד ימין לאחר איחוד תלויות:

תלות	תכונה	סגור	עודפת	שינוי
$B \rightarrow AD$	A	$B^+ = BDEG$	לא	
$B \rightarrow AD$	D	$B^+ = BAC$	לא	
$D \rightarrow EBG$	E	$D^+ = DBGAC$	לא	
$D \rightarrow EBG$	B	$D^+ = DEG$	לא	
$D \rightarrow EBG$	G	$D^+ = DEBAC$	לא	

$$F_C = \{A \rightarrow C, B \rightarrow AD, D \rightarrow EBG\} \quad \text{כיסוי קנוני:}$$

אלגוריתם למציאת כיסוי קנוני

1	בונים טבלה עם מספר שורות כמספר התכונות הנמצאות בצד ימין של אוסף התלויות. מחשבים את הסגור של צד שמאל של התלות ללא התכונה הנבדקת לעודפות. אם התכונה נמצאת בסגור היא עודפת ומוחקים אותה. אם כל התכונות בצד ימין של תלות כלשהי עודפות, התלות כולה עודפת. אם מחקנו תכונה כלשהי אסור להשתמש בה בחישוב הסגורים לאחר מכן!
2	בסיום בדיקת העודפות מצד ימין כותבים מחדש את אוסף התלויות העדכני
3	בונים טבלה עם מספר שורות כמספר התכונות הנמצאות בצד שמאל של התלויות הכוללות יותר מתכונה אחת בצד שמאל. מחשבים את הסגור של צד שמאל של התלות ללא התכונה הנבדקת לעודפות. אם התכונה נמצאת בסגור, אז גם כל צד ימין של התלות נכנס לסגור והתכונה עודפת. תלות עם תכונה אחת בצד שמאל לא נבדקת לעודפות משמאל.
4	אם באיזשהו שלב הגענו למצב שיש שתי תלויות עם אותו צד שמאל, מאחדים את התלויות לתלות אחת ובודקים עודפות מימין שוב רק לתלות זו.



תיכון במסד נתונים

מציאת כל המפתחות הקבילים

דוגמא:

$$R = (A, B, C, D, E, G)$$

$$F_C = \{A \rightarrow C, B \rightarrow AD, D \rightarrow EBG\}$$

מצאו את כל המפתחות הקבילים של R

אין אף תכונה שלא נמצאת מימין של איזושהי תלות ולכן אין אף תכונה שחייבת להיות חלק מכל מפתח קביל.

התכונות G, E, C מופיעות בצד ימין כלשהו אך לא מופיעות בשום צד שמאל ולכן הן לא יכולות להיות חלק מאף מפתח קביל.

נחשב את הסגורים של התכונות שכן יכולות להיות חלק ממפתח:

$$A^+ = AC$$

$$B^+ = BADEGC$$

$$D^+ = DEBGAC$$

גם B וגם D הם מפתחות קבילים, אין צורך לבדוק צירופים נוספים.

אלגוריתם למציאת כל המפתחות הקבילים

1	בודקים האם בכיסוי הקנוני יש תכונה כלשהי שלא מופיעה בשום צד ימין של אף תלות. אם יש תכונה כזו היא חייבת להיות חלק מכל מפתח קביל.
2	בודקים האם בכיסוי הקנוני יש תכונה כלשהי שמופיעה בצד ימין של תלות כלשהי, אך לא מופיעה בשום צד שמאל של אף תלות, אם יש תכונה כזו, היא לא יכולה להיות חלק מאף מפתח קביל.
3	אם אין אף תכונה שחייבת להיות חלק מכל מפתח קביל, מחשבים את הסגורים של כל אחת מהתכונות שיכולות להיות חלק ממפתח קביל, אם יש תכונות שחייבות להיות חלק מכל מפתח קביל, מחשבים את הסגור של כל התכונות האלו יחד.
4	אם הסגור שחישבנו מכיל את כל התכונות של היחס הרי שהוא מפתח קביל, אם לא, מוסיפים תכונה נוספת (שיכולה להיות חלק ממפתח קביל) ומחשבים את הסגור מחדש עד שמגיעים לכל התכונות ביחס.
5	כאשר מוצאים מפתח קביל בגודל מסוים, יש לבדוק האם קיימים מפתחות קבילים נוספים באותו גודל, וכן לבדוק האם קיימים מפתחות קבילים בגודל גדול יותר שאינם מכילים אף תכונה מהמפתחות שכבר מצאנו.



תיכון במסד נתונים

BCNF

דוגמא:

$$R = (A, B, C, D, E)$$

$$F = \{A \rightarrow CE, BC \rightarrow D, CD \rightarrow AB, BE \rightarrow AD, C \rightarrow E\}$$

פרקו את היחס לפי אלגוריתם פירוק ל-BCNF

$$F_C = \{A \rightarrow C, CD \rightarrow B, BE \rightarrow AD, C \rightarrow E\} \quad R\text{-ל:}$$

ונתון כי המפתחות הקבילים של R הם: (AB, AD, BC, BE, CD)

התלויות $A \rightarrow C, C \rightarrow E$ מפרות BCNF כיוון שהן לא טריוויאליות וצד שמאל שלהן לא מפתח ב-R.

נבחר באופן שרירותי להתחיל מהתלות $A \rightarrow C$ ונפרק לפיה ונרשום את כל התלויות החלות על כל אחד מהפירוקים:

פירוקים	אוסף תלויות	מפתחות קבילים	צורה נורמלית
$R_1 = (A, B, D, E)$	$F_1 = \{BE \rightarrow AD, A \rightarrow E, AB \rightarrow DE, AD \rightarrow BE\}$	BE, AB, AD	3NF
$R_2 = (A, C)$	$F_2 = \{A \rightarrow C\}$	A	BCNF

התלות $A \rightarrow E$ מפרה BCNF. נפרק לפי תלות זו, נחשב את התלויות החלות על כל אחד מהפירוקים ואת המפתחות של כל יחס:

פירוקים	אוסף תלויות	מפתחות קבילים	צורה נורמלית
$R_{11} = (A, B, D)$	$F_{11} = \{AB \rightarrow D, AD \rightarrow B, \}$	AB, AD	BCNF
$R_{12} = (A, E)$	$F_{12} = \{A \rightarrow E\}$	A	BCNF

כל הפירוקים ב-BCNF ולכן כאן מסתיים האלגוריתם. (על פי רוב בשלב זה מתבקשים בשאלה גם לבדוק האם הפירוק משמר תלויות)

יחס כלשהו נמצא ב-BCNF אם עבור כל התלויות בו מתקיים אחד משני התנאים:

- התלות טריוויאלית
- צד שמאל של התלות הוא מפתח ב-R (אם יש ביחס רק 2 תכונות הוא בוודאי ב-BCNF)

אלגוריתם לפירוק יחס ל-BCNF

מחפשים תלות כלשהי ב-R שמפרה BCNF. אם התלות מהצורה $X \rightarrow Y$, מפרקים את היחס לשני יחסים:	1
$R_1 = (R - Y)$ $R_2 = (X, Y)$ זאת אומרת: יחס אחד מכיל את כל התכונות של היחס המקורי מלבד התכונות הנמצאות מימין בתלות המפרה BCNF, והיחס השני מכיל את כל התכונות שבתלות המפרה BCNF	
מוצאים את כל התלויות החלות על כל אחד מהיחסים שיצרנו: מחשבים את הסגור של כל אחת מהתכונות (ושל כל השילובים האפשריים) ביחס המפורק מתוך אוסף התלויות המקורי (ולא מהכיסוי הקנוני) ויוצרים אוסף תלויות המתאים לתכונות שביחס המפורק	2
בודקים אם היחס החדש ב-BCNF, אם לא מפרקים אותו שוב לפי שלבים 1 ו-2	3
* האלגוריתם מבטיח שימור מידע אך לא מבטיח שימור תלויות	



תיכון במסד נתונים

3NF

דוגמא:

$$R = (A, B, C, D, E, G)$$

$$F_C = \{A \rightarrow C, B \rightarrow AD, D \rightarrow EBG\}$$

פרקו את היחס לפי אלגוריתם פירוק ל-3NF

נפרק את היחס לשלושה תתי יחסים לפי התלויות הנתונות:

פירוקים	אוסף תלויות	מפתחות קבילים	צורה נורמלית
$R_1 = (A, C)$	$F_1 = \{A \rightarrow C\}$	A	BCNF
$R_2 = (A, B, D)$	$F_2 = \{B \rightarrow AD\}$	B	BCNF
$R_3 = (B, D, E, G)$	$F_3 = \{D \rightarrow EBG\}$	D	BCNF

A הוא מפתח קביל ב- R_1 , B הוא מפתח קביל ב- R_2 , ו-D הוא מפתח קביל ב- R_3 , לכן כל תתי היחסים ב-BCNF.

אף אחד מהיחסים לא מוכל ביחס אחר ולכן לא צריך לבטל שום יחס.

גם R_2 וגם R_3 מכילים מפתחות קבילים של R ולכן אין צורך להוסיף יחס נוסף.

יחס כלשהו נמצא ב-3NF אם עבור כל התלויות בו מתקיים אחד משלושת התנאים:

- התלות טריוויאלית
- צד שמאל של התלות הוא מפתח ב-R
- כל התכונות בצד ימין של התלות פחות התכונות שבצד שמאל של התלות מוכלות במפתח קביל כלשהו (כל תכונה יכולה להיות חלק ממפתח קביל אחר)

אלגוריתם לפירוק יחס ל-3NF

1	עבור כל תלות הקיימת ב-R יוצרים יחס חדש המכיל את כל התכונות בתלות
2	אם נוצרו שני יחסים כך שאחד מוכל בשני, מוחקים את היחס המוכל: $R_i \subseteq R_j \rightarrow R_i$
3	אם אף אחד מהיחסים שנוצרו לא מכיל מפתח קביל כלשהו (של R המקורי), יוצרים יחס נוסף עבור אחד המפתחות הקבילים
*	האלגוריתם מבטיח שימור מידע ושימור תלויות.
*	היחסים הנוצרים מפירוק זה יכולים להיות ב-BCNF או ב-3NF



תיכון במסד נתונים

שימור מידע

דוגמא:

$$R = (A, B, C, D, E)$$

$$F = \{CD \rightarrow E, A \rightarrow C, AD \rightarrow E, ABC \rightarrow DE\}$$

נתון הפירוק:

$$R_1 = (A, B, C, D)$$

$$R_2 = (A, D, E)$$

האם הפירוק משמר מידע?

החיתוך בין שני היחסים המפורקים הוא: AD

נמצא את התלויות החלות על כל אחד מהיחסים המפורקים:

$$R_1 = (A, B, C, D)$$

$$F_1 = \{A \rightarrow C, ABC \rightarrow D\}$$

$$R_2 = (A, D, E)$$

$$F_2 = \{AD \rightarrow E\}$$

נמצא את המפתחות הקבילים של כל אחד מהיחסים:

$$R_1 = (A, B, C, D)$$

$$F_1 = \{A \rightarrow C, ABC \rightarrow D\} \quad ABC^+ = ABCD$$

$$R_2 = (A, D, E)$$

$$F_2 = \{AD \rightarrow E\} \quad AD^+ = ADE$$

המפתח של R_2 הוא AD וזהו החיתוך בין היחסים המפורקים ולכן הפירוק משמר מידע

פירוק נתון כלשהו של יחס למספר תתי יחסים משמר מידע אם:

$$(R_1 \cap R_2) \rightarrow R_2 \quad \text{או} \quad (R_1 \cap R_2) \rightarrow R_1$$

זאת אומרת, שהתכונות המשותפות לשני היחסים הן מפתח קביל של לפחות אחד מהיחסים

אלגוריתם לבדיקה האם פירוק משמר מידע

1	מחשבים את החיתוך של שני היחסים – מוצאים את כל התכונות המשותפות לשניהם
2	מוצאים את כל התלויות החלות על כל אחד מהיחסים המפורקים
3	מוצאים את כל המפתחות הקבילים של כל אחד מהיחסים המפורקים
4	אם החיתוך שווה למפתח של אחד היחסים הרי שהפירוק משמר מידע. אם לא, הוא לא משמר מידע



תיכון במסד נתונים

שימור תלויות

דוגמא:

$$R = (A, B, C, D, E)$$

$$F = \{A \rightarrow CE, BC \rightarrow D, CD \rightarrow AB, BE \rightarrow AD, C \rightarrow E\}$$

$F_C = \{A \rightarrow C, CD \rightarrow B, BE \rightarrow AD, C \rightarrow E\}$ נתון כיסוי קנוני ל-R: נתון הפירוק:

$$R_1 = (A, C)$$

$$R_2 = (A, B, D)$$

$$R_3 = (A, E)$$

האם הפירוק משמר תלויות?

נמצא את התלויות החלות על כל אחד מהיחסים המפורקים:

$$R_1 = (A, C)$$

$$F_1 = \{A \rightarrow C\}$$

$$R_2 = (A, B, D)$$

$$F_2 = \{AB \rightarrow D, AD \rightarrow B\}$$

$$R_3 = (A, E)$$

$$F_3 = \{A \rightarrow E\}$$

נאחד את כל התלויות יחד:

$$\{A \rightarrow C\} \cup \{AB \rightarrow D, AD \rightarrow B\} \cup \{A \rightarrow E\}$$

אפשר מיד לראות כי התכונה C לא מופיעה בשום צד שמאל של אף תלות

ולכן הסגור שלה הוא $C^+ = C$ ולכן ברור כי לא נוכל להגיע מאוסף תלויות

אלו לתלות $C \rightarrow E$.

מספיק להוכיח כי תלות אחת לא מתקיימת כדי לקבוע כי הפירוק לא משמר תלויות.

פירוק נתון כלשהו של יחס למספר תתי יחסים משמר תלויות אם:

$$(F_1 \cup F_2 \cup \dots \cup F_k)^+ = F^+$$

זאת אומרת, שהסגור של איחוד התלויות מכל הפירוקים שווה לסגור של אוסף התלויות המקורי

אלגוריתם לבדיקה האם פירוק משמר תלויות

מוצאים את כל התלויות החלות על כל אחד מהיחסים המפורקים: מחשבים את הסגור של כל אחת מהתכונות (ושל כל השילובים האפשריים) ביחס המפורק מתוך אוסף התלויות המקורי (ולא מהכיסוי הקנוני) ויוצרים אוסף תלויות המתאים לתכונות שביחס המפורק. בתהליך כזה יכולים "להתגלות" תלויות נוספות, לדוגמא אם ביחס המקורי היו התלויות: $\{A \rightarrow B, B \rightarrow C\}$ והפירוק הוא: $R_2 = (A, B), R_1 = (A, C)$ הרי שעל היחס R_1 אין כביכול שום תלות, כיוון ש-C לא מופיע בתלות $A \rightarrow B$ ו-A לא מופיע בתלות $B \rightarrow C$. אך אם נחשב את הסגור של A נגלה כי: $A^+ = ABC$ ולכן התלות $A \rightarrow C$ חלה על היחס R_1	1
מאחדים את כל אוספי התלויות מכל היחסים המפורקים	2
מחשבים את הסגור של אוסף התלויות המקורי ואת הסגור של איחוד אוספי התלויות של היחסים המפורקים באמצעות: 1. רפלקסיביות - אם: $X \subseteq Y$ אז: $Y \rightarrow X$ 2. טרנזיטיביות - אם: $X \rightarrow Y$ וגם: $Y \rightarrow Z$ אז: $X \rightarrow Z$ 3. איחוד - אם: $X \rightarrow Y$ וגם: $X \rightarrow Z$ אז: $X \rightarrow YZ$ 4. פירוק - אם: $X \rightarrow YZ$ אז: $X \rightarrow Y$ וגם: $X \rightarrow Z$	3
אם הסגורים שווים הפירוק משמר תלויות	4

טיפוסי נתונים מורכבים



דירוג מסמכים

מדד PageRank

מילון	
משמעות	סימון
פרמטר בין 0 ל-1. נהוג להשתמש ב-0.15	δ
מספר הדפים במאגר	N
ההסתברות לעבור מדף i לדף j	$T[i, j]$
הדירוג של הדף j	$P[j]$

נוסחאות	
משמעות	נוסחה
הדירוג של הדף j	$P[j] = \frac{\delta}{N} + (1 - \delta) \cdot \sum_{i=1}^N (T[i, j] \cdot P[i])$

מדד IDF

מילון	
משמעות	סימון
מסמך	d
מונח	t
מספר המונחים שיש במסמך d	$n(d)$
מספר ההופעות של המונח t במסמך d	$n(d, t)$
מספר המסמכים בהם מופיע המונח t	$n(t)$
שאלתה של ביטוי המכיל מספר מונחים	Q

נוסחאות	
משמעות	נוסחה
מדד לשכיחות מונח במסמך	$TF(d, t) = \log \left(1 + \frac{n(d, t)}{n(d)} \right)$
המשקל של המונח t במאגר המסמכים	$IDF(t) = \left(\frac{1}{n(t)} \right)$
הרלוונטיות של המסמך d עבור השאלתה Q הם כל המונחים השייכים לשאלתה בתנאי שהם לא (stop words)	$r(d, Q) = \sum_{t \in Q} TF(d, t) \cdot IDF(t)$

טיפוסי נתונים מורכבים



ייצוג נתונים בבסיס נתונים לא רלציוני - NoSQL

JSON

השפה מכירה ב-5 טיפוסי נתונים: מספר, מחרוזת, בוליאני, מערך ואובייקט. בנוסף, אפשר להזין null כערך. אובייקט הוא רשימה של זוגות מהצורה "תכונה:ערך". רשימת הזוגות סגורה בתוך סוגריים מסולסלים ופסיק מפריד בין זוג לזוג. כאשר הערך יכול להיות כל אחד מהטיפוסים שהשפה מכירה בהם (כולל אובייקט או מערך). שם התכונה הוא מחרוזת ולכן הוא תחום בגרשיים כפולים, והערך יהיה תחום בגרשיים כפולים רק עם הוא מסוג מחרוזת. מערך הוא רשימת ערכים אשר סגורה בתוך סוגריים מרובעים. בין ערך לערך מפריד פסיק, והערכים יכולים להיות כל אחד מטיפוסי הנתונים. אם הערך מסוג מחרוזת הוא תחום בגרשיים כפולים.

לדוגמה:

```
{ "treatment-type": "רופא משפחה",  
  "doctor": {  
    "doctor-name": "משה",  
    "doctor-family": "כהן",  
    "doctor-id": 123 },  
  "patient": {  
    "patient-name": "יוסי",  
    "patient-family": "לוי",  
    "patient-id": 456 },  
  "time-stamp": "12:27:32 18/07/2022",  
  "referral": { "specialist": "אורטופד גב", "hospital": "איכילוב", "referral-reason": "כאבים בגב תחתון" }  
}
```

XML

XML היא שפה של תגיות. לכל ערך יש תגית פותחת ותגית סוגרת. בין שתי תגיות יכולות להיות תגיות נוספות

לדוגמה:

```
<treatment>  
  <treatment-type>רופא משפחה</treatment-type>  
  <doctor>  
    <doctor-name>משה</doctor-name>  
    <doctor-family>כהן</doctor-family>  
    <doctor-id>123</doctor-id>  
  </doctor>  
  <doctor>  
    < patient-name>יוסי </patient-name>  
    < patient-family>לוי </patient-family>  
    < patient-id>456</patient-id>  
  </doctor>  
  <time-stamp>12:27:32 18/07/2022</time-stamp>  
  <referral>  
    <specialist>אורטופד גב</specialist>  
    <hospital>איכילוב</hospital>  
    <referral-reason>כאבים בגב תחתון</referral-  
reason>  
  </referral>  
</treatment>
```



טיפוסי נתונים מורכבים

ייצוג נתונים ב-RDF ושליפתם באמצעות SPARQL

SPARQL

שאלתה מורכבת ממשתנה אשר מקיים את כל השלשות המוגדרות ב-where. אפשר לבקש כמה משתנים. כל משתנה מוגדר עם סימן שאלה לפני שמו. הערכים שיתקבלו הם ערכים המקיימים את כל השלשות המוגדרות ב-where.

לדוגמה:

```
select ?n
where{
  ?did instance of doctor.
  ?did name ?n.
}
```

השאלתה שולפת את השמות של כל הרופאים המוגדרים במאגר

RDF

ייצוג נתונים באמצעות RDF נעשה באמצעות שלשות מהצורה נושא-פרדיקט-מושא. כל שלשה כזו ניתנת לייצוג בגרף באמצעות קשת (הפרדיקט) המחברת בין הנושא (קודקוד) לבין המושא (קודקוד).

לדוגמה:
doctor.
"yosi".

d123 instance of doctor.
d123 name "yosi".



אינדקסים וגיבוב



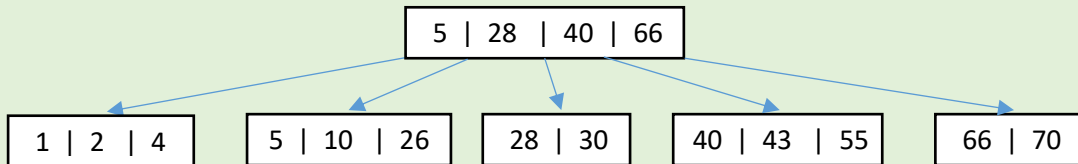
עצי B+

- עץ בעל $n = k$: בכל צומת מלבד השורש יש בין $\lfloor n/2 \rfloor$ ל- n מצביעים. ובין $\lfloor n/2 \rfloor - 1$ ל- $n - 1$ ערכים.

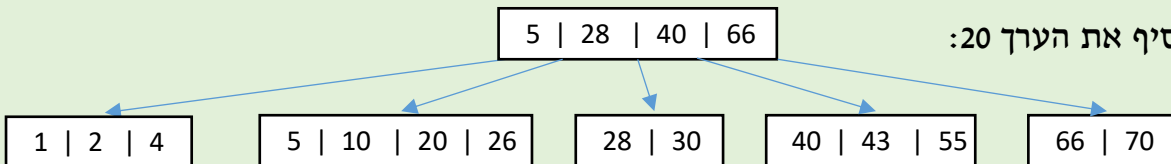
דוגמא:

$$n = 5$$

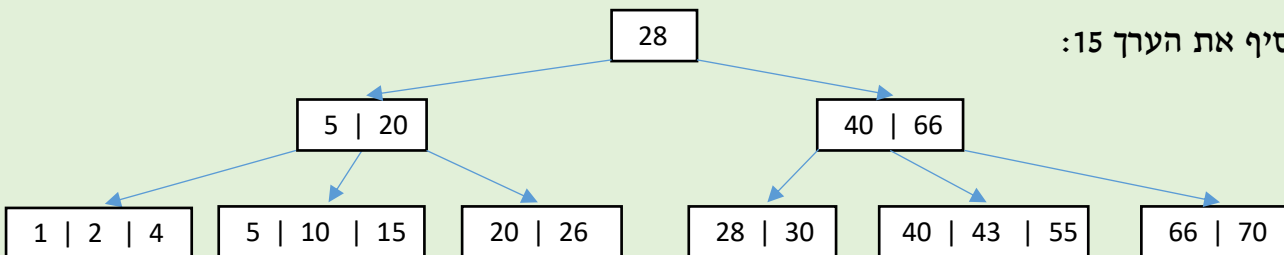
יש להכניס את הערכים
20, 15,



נוסיף את הערך 20:



נוסיף את הערך 15:



אלגוריתם להכנסת ערכים חדשים לעץ

1	מוצאים את העלה אליו הערך החדש צריך להיכנס. אם יש מקום בעלה (כמות הערכים בעלה קטנה מ- $n - 1$) מכניסים את הערך החדש למקום שלו וממשיכים לערך הבא. אם אין מקום בעלה, מפצלים אותו לשני עלים ומכניסים את הערך החדש בעלה המתאים (אם הפיצול אי זוגי אפשר לבחור שצד אחד מקבל מספר גדול מהצד השני ולהיות עקביים לאורך כל הפתרון).
2	לאחר פיצול של עלה לשניים, מעתיקים את הערך הקטן ביותר בעלה עם הערכים הגדולים יותר (העלה הימני מבין השניים) לצומת שמעליו.
3	אם יש מקום בצומת הפנימי מכניסים את הערך לצומת ומסדרים את החיצים כלפי מטה, אם אין מקום בצומת מפצלים אותו לשניים.
4	לאחר פיצול של צומת פנימי לשניים (לפי: $\lfloor (n - 1)/2 \rfloor$ הערכים הגדולים לימין, $\lfloor (n - 1)/2 \rfloor$ הערכים הקטנים לשמאל), מעבירים (ולא מעתיקים) את הערך האמצעי (שבין שני הצמתים) לצומת שמעליו.
5	לאחר סיום סידור העץ מחדש, עוברים לערך החדש הבא ומתחילים מתחילת האלגוריתם



אינדקסים וגיבוב

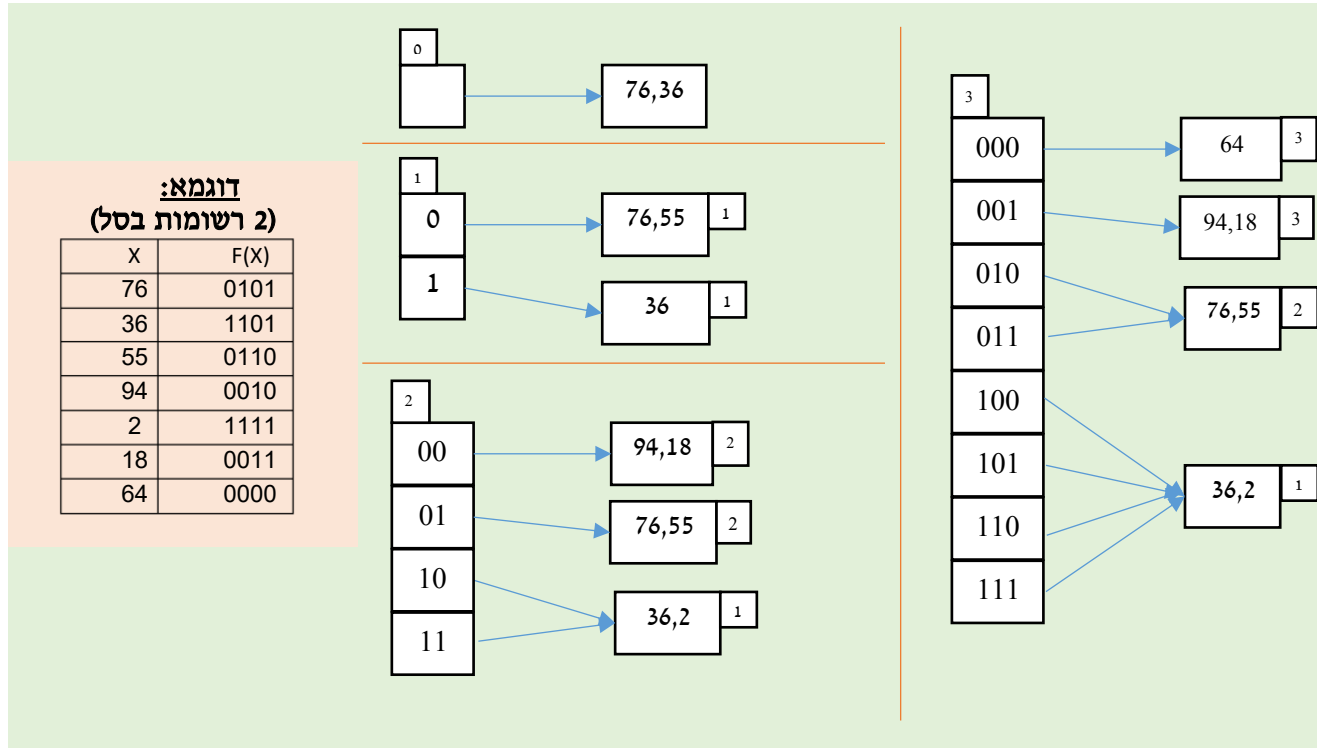
בעץ B+ המכיל x ערכים:	
מינימום עלים:	מקסימום עלים:
$L_{min} = \left\lceil \frac{x}{(n-1)} \right\rceil$	$L_{max} = \left\lceil \frac{x}{\lfloor n/2 \rfloor - 1} \right\rceil$
מינימום רמות:	מקסימום רמות:
<p>1 - רמת העלים (הרמה הנמוכה ביותר), 2 - רמת ההורים - מחשבים את כמות הצמתים המינימלית ברמה השנייה: $I_{2min} = \left\lceil \frac{L_{min}}{n} \right\rceil$ 3 - רמת הסבים - מחשבים את כמות הצמתים המינימלית ברמה השלישית: $I_{3min} = \left\lceil \frac{I_{2min}}{n} \right\rceil$ וכן הלאה, עד שמגיעים ל- $I_{imin} < n$ (כמות צמתים ברמה ה-i פחות מ-n) $i + 1$ - רמת השורש <u>כמות הרמות בפועל:</u> רמת העלים (1) + רמת ההורים (2) + רמת הסבים (3) + ... + רמת השורש ($i + 1$)</p>	<p>1 - רמת העלים (הרמה הנמוכה ביותר), 2 - רמת ההורים - מחשבים את כמות הצמתים המקסימלית ברמה השנייה: $I_{2max} = \left\lceil \frac{L_{max}}{\lfloor n/2 \rfloor} \right\rceil$ 3 - רמת הסבים - מחשבים את כמות הצמתים המקסימלית ברמה השלישית: $I_{3max} = \left\lceil \frac{I_{2max}}{\lfloor n/2 \rfloor} \right\rceil$ וכן הלאה, עד שמגיעים ל- $I_{imax} < n/2$ (כמות צמתים ברמה ה-i פחות מ-$n/2$) $i + 1$ - רמת השורש <u>כמות הרמות בפועל:</u> רמת העלים (1) + רמת ההורים (2) + רמת הסבים (3) + ... + רמת השורש ($i + 1$)</p>
מינימום ערכים בעץ עם l רמות:	מקסימום ערכים בעץ עם l רמות:
<p>מחשבים את כמות הבנים המינימלית שיש לשורש - 2, מחשבים כמות הצמתים המינימלית שיש ברמת הנכדים - $2 \cdot \lfloor n/2 \rfloor$, מחשבים כמות הצמתים המינימלית שיש ברמת הנינים - $2 \cdot \lfloor n/2 \rfloor \cdot \lfloor n/2 \rfloor$ וכן הלאה עד $l - 1$ מחשבים את כמות הערכים המינימלית שיש בעלים - $2 \cdot \left\lfloor \frac{n}{2} \right\rfloor \cdot \left\lfloor \frac{n}{2} \right\rfloor \cdot \dots \cdot (\lfloor n/2 \rfloor - 1)$</p>	<p>מחשבים את כמות הבנים המקסימלית שיש לשורש - n, מחשבים כמות הצמתים המקסימלית שיש ברמת הנכדים - $n \cdot n$, מחשבים כמות הצמתים המקסימלית שיש ברמת הנינים - $n \cdot n \cdot n$ וכן הלאה עד $l - 1$ מחשבים את כמות הערכים המקסימלית שיש בעלים - $n \cdot n \cdot n \cdot \dots \cdot (n - 1)$</p>



אינדקסים וגיבוב

גיבוב מתרחב

- בגיבוב מתרחב נתון כמה רשומות יכולות להיכנס בכל סל.
- גודל המדריך גדל עם הזמן בהתאם לערכים שנדרש להכניס. המדריך גדל כל פעם פי 2.
- אם נדרש לחשב את המספר הבינארי המייצג כל ערך, יש להשתמש בפונקציית הגיבוב הנתונה בתרגיל.



אלגוריתם להכנסת ערכים למבנה גיבוב מתרחב	
1	בהתחלה, גודל המדריך שווה אפס (משתמשים באפס ביטים) והוא מצביע לסל אחד. מכניסים את הערכים הראשונים לסל עד שהוא מתמלא.
2	ברגע שאין מקום בסל שאליו הערך החדש צריך להיכנס, מכפילים את רוחב המדריך פי 2 ולמעשה מוסיפים ביט נוסף.
3	רוחב המדריך לא בהכרח באותו רוחב כמו כל סל. אם תא אחד במדריך מפנה לסל מסוים, רוחב הסל כרוחב המדריך. אם יש שני תאים המפנים לאותו הסל, רוחב הסל קטן באחד מרוחב המדריך. אם יש ארבעה תאים המפנים לאותו סל, רוחב הסל קטן ב-2 מרוחב המדריך.

חישובים שונים למבנה של גיבוב מתרחב	
נדרש	כיצד עושים?
מספר הכניסות המינימלי במדריך בהינתן כמות סלים a	מספר הכניסות במדריך גדל לפי 2^x לכן: נחשב $x = \lceil \log_2 a \rceil$ נחשב $2^x -$ זה מספר הכניסות המינימלי
מספר הכניסות המינימלי במדריך בהינתן כמות ערכים v וגודל סל מקסימלי s	נחשב את כמות הסלים המינימלית האפשרית: $a = \lceil v/s \rceil$ נחשב $x = \lceil \log_2 a \rceil$ נחשב $2^x -$ זה מספר הכניסות המינימלי
רוחב התחילית המינימלית בטבלת כתובות הסלים בהינתן כמות ערכים v וגודל סל מקסימלי s	נחשב את כמות הסלים המינימלית האפשרית: $a = \lceil v/s \rceil$ נחשב $x = \lceil \log_2 a \rceil$ זה רוחב התחילית המינימלית



עיבוד שאילתות

סימונים מקובלים:

N_R – מספר השורות ביחס R

$V(a, R)$ – מספר הערכים השונים המופיעים בתכונה a ביחס R

(אם a מפתח ב- R , הרי ש: $N_R = V(a, R)$)

כללים לחישוב גודל יחס התוצאה של שאילתה

איכות התשובה	דוגמה	גודל יחס התוצאה	מקרה	σ בחירת ערך ספציפי לתכונה
מדויק	$\sigma_{a=8}(R)$	1	בחירת ערך ספציפי לתכונת מפתח	
ממוצע	$\sigma_{a=yossi}(R)$	$\left\lceil \frac{N_r}{V(A, r)} \right\rceil$	בחירת ערך ספציפי לתכונה שאינה מפתח	
ממוצע	$\sigma_{a=5 \wedge B=19}(R)$	$\left\lceil \frac{N_r}{V(A, r)} \right\rceil / V(B, r)$	בחירת ערכים ספציפיים למספר תכונות שאינן מפתח	

פעולת ההטלה מאחדת כפילויות ולכן גודל יחס התוצאה של: $\Pi_a(R)$ הוא לכל היותר: $V(a, R)$

Π
פעולת ההטלה

איכות התשובה	גודל יחס התוצאה	מקרה
מדויק	$N_R \cdot N_S$	אין השוואה בין תכונות (\times) או אין תכונות עם אותו שם (\bowtie)
חסם עליון	$\min(N_S, N_S - (V(a, S) - V(a, R)))$	$\sigma_{R.a=S.a}(R \times S)$ או $R \bowtie S$ כאשר a התכונה המשותפת היא מפתח ב- R
ממוצע	$\left(\frac{N_S}{V(a, S)}\right) \cdot V(a, R)$ (בהנחת התפלגות אחידה, ובתנאי שהתוצאה קטנה מהחסם העליון)	
ממוצע	$\frac{N_R \cdot N_S}{\max(V(a, R), V(a, S))}$	התכונה המשותפת לא מפתח באף אחד מהיחסים או שהיא מפתח בשניהם

\times
מכפלה קרטזית
או
 \bowtie
צירוף טבעי

עיבוד שאילתות



יצירת עץ ביטוי עשויה לסייע לפענוח של כמות השורות שיצאו ביחס התוצאה של שאילתות מורכבות.

עץ הביטוי מורכב מצמתים אשר מייצגים מצבים שונים שיחס התוצאה עובר עד שהוא מגיע לתוצאה הסופית. נהוג לסמן כל שינוי ביחס התוצאה באמצעות צומת בעץ, וכן נהוג שהיחס המקורי ממנו מתחילים מופיע בתחתית העץ וככל שעולים למעלה כך היחס משתנה עד שמגיעים ליחס התוצאה הסופי.

