SQL

- יצירת יחסים והכנסת רשומות חדשות
- שליפת נתונים מתוך יחס אחד
- שליפת נתונים מתוך מספר יחסים
- פעולת הקבצה ופונקציות הקבצה
- פונקציות שימושיות
- טריגרים ופונקציות בפורמט PostgreSQL
- ⇐ תרגול

דף נוסחאות - חזרה לבחינה - SQL

פסוקי DDL (Data Definition Language) ב-SQL		
משפט	תחביר	הסבר
יצירת טבלה	create table *table_name* (*column_name* data_type,...)	לאחר הפתיח יש לרשום את שם הטבלה ולאחר מכן בתוך סוגריים עגולים את כל עמודות הטבלה עם פסיק מפריד בין אחת לשניה. 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 *table_name*	בכדי למחוק טבלה שלמה מבסיס הנתונים יש לרשום drop table *table_name*. אם רוצים לבקש למחוק את הטבלה רק אם היא קיימת יש לרשום drop table 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 *table_name* values ('*value*','*value*'...),(...),(...)...	בכדי להכניס רשומות חדשות לטבלה קיימת יש לרשום insert into *table_name* 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 *table_name* set *column_name*='*value*' where *column_name*='*value*'	עדכון של תכונות ספציפיות בתוך רשומה קיימת ביחס: update *table_name* set *column_name*='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 *table_name* where *column_name*='*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 *table_name* 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 *column_name datatype*, מחיקת עמודה קיימת - DROP COLUMN *column_name*, שינוי שם עמודה - RENAME COLUMN *column_name* TO *new_column_name*, הוספת אילוץ - ADD *CONSTRAINT*
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 (true, yes, on, 1) או שקר (false, no, off, 0) או ערך ריק (null)
	date	תאריך - הפורמט המומלץ הינו: yyyy-mm-dd (לדוגמה, העשרים לנובמבר 88 ייכתב כך: 1988-11-20)
אילוצים 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 (*column_name,...*)
	foreign key (*column_name*) references *table_name*	מפתח זר - בסיום ההכרזות על כל התכונות ביחס, עבור כל תכונה שהיא מפתח זר (מפתח של טבלה אחרת) רושמים foreign key (*column_name*) ומיד אחר כך את הטבלה שממנה המפתח הזר לקוח באמצעות references *table_name*. 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 (*column_name* ='*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 (*salary* >= 0)

דף נוסחאות - חזרה לבחינה - SQL

שאילתות SQL		
מושג	**תחביר**	**הסבר**
הטלה	select	בחירה של עמודות מסוימות מתוך היחס הנמצא ב-from
מהיחס/ים	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: **select distinct**(column_name) או לדוגמא: **select count** (**distinct** (column_name))
שרשור 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: table_A natural join table_B from
מכפלה קרטזית	,	מחבר כל שורה ביחס הראשון עם כל שורה ביחס השני: table_A , table_B **from**. תכונות בעלות אותו שם לא מתבטלות וביחס התוצאה יש שתי עמודות שונות עבור תכונות אלו ולכן חייבים לכתוב מאיזה טבלה מבקשים את התכונה: table_A.column_name **select**
נתינת שם ליחס	as	מאפשר לקרוא ליחס בשם מקוצר. שימושי מאוד במכפלה קרטזית בין שני יחסים שהם אותו היחס: table_A **as** A1, table_A **as** A2 **select** A1.column_name אפשר להשתמש גם אחרי שאילתה מקוננת ואז יש להגדיר את שמות העמודות של השאילתה: Select A.c1 **from** (**select** column_name1, column_name2 **from** table_A) **as** A(c1,c2))
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 שנעשית לפני הקיבוץ)
שאילתות מקוננות	all, any	**all** 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: column_name >= **all**(**select** column_name **from** table_A)
	in, not in	משמש לבדיקה האם ערך כלשהו נמצא או לא נמצא ביחס המופיע אחריו. לדוגמא: column_name **in** (**select** column_name **from** table_A)
	exists, not exists	exists מחזיר "אמת" אם השאילתה המופיעה אחריו מכילה לפחות רשומה אחת. not exists מחזיר "אמת" אם השאילתה המופיעה אחריו ריקה לחלוטין.
יצירת יחס חדש	with …. as	משמש ליצירת יחס חדש לצורך שימוש עתידי בשאילתה אחרת. יעיל בשאילתות מורכבות. **with** name(col1, col2…) **as** (**select** columnA **as** col1, columnB **as** col2 **from** table_name **where**….)
חלק מתאריך	date_part	משמש לקבלת חלק מסוים מתוך תאריך. שנה: **date_part('year'**,dateName), חודש: **date_part('month'**,dateName), יום: **date_part('day'**,dateName).
תאריך נוכחי	current_date	מחזיר את התאריך הנוכחי ברגע שבו מבוצעת השאילתה. שימושי כאשר נדרש שהשאילתה תהיה נכונה בכל תאריך.
פעולות על יחסים	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: union all
חיפוש במחרוזת	like / ilike	כאשר נדרש לבדוק האם מחרוזת כלשהי מכילה תת מחרוזת אחרת: column_name **like** 'sub string'. אם רוצים לאפשר שתת המחרוזת תתחיל באיזשהו תו אחר: '_text', אפשר גם לאפשר כמה תווים: '%text'. בכדי לאפשר תו נוסף בסוף הטקסט: 'text_', וכמה תווים: 'text%'. שימוש ב-ilike מתעלם מאותיות רישיות.

דף נוסחאות - חזרה לבחינה - SQL

טריגר בפורמט PostgreSQL

create or replace function trig_name() returns trigger as$$	פסוק ליצירת הטריגר trig_name
declare	פתיחת אזור להכרזה על משתנים. אם אין שימוש במשתנים אין צורך ב-declare
varName1 integer; varName2 boolean; varName3 numeric(4,2); varName4 varchar(20);	שמות המשתנים והסוג שלהם. • משתנה מסוג numeric משמש למספרים עשרוניים, יש להגדיר את כמות הספרות בסך הכל (בדוגמא-4) וכן את כמות הספרות אחרי הנקודה (בדוגמא-2), לדוגמא עבור המספר 123.58 יש להגדיר: numeric(5,2). • משתנה מסוג varchar משמש לקליטת מילים, יש להגדיר את אורך המילה המקסימלית האפשרית (בדוגמא-20).
begin	התחלת הפונקציה עצמה. גוף הפונקציה יכול לכלול לולאות for, משפטי if, ופעולות נוספות כמו הכנסה של רשומה ליחס כלשהו: insert into table_name values(A,B,C,D…)
If(condition) then	התחלת משפט "אם". בתוך התנאי אפשר לבדוק האם מתקיים משהו בתוך יחס הקיים במערכת (באמצעות שאילתה רגילה), אפשר להשוות את אחד הערכים בשורה החדשה שמנסים להכניס לבסיס הנתונים, לדוגמא: if(new.columnA in (select columnA from tale_name))
raise notice 'error message for the user';	הקפצת הודעת שגיאה למשתמש
return null;	החזרת ערך ריק למשתמש ומניעת הכנסת הרשומה החדשה לבסיס הנתונים
end if;	סיום משפט "אם"
return new;	הכנסת הרשומה החדשה לבסיס הנתונים
end; $$ language plpgsql;	סיום הפונקציה
create trigger T1 before insert or update on table_name	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
for each row	האם לבצע את הטריגר לכל שורה בנפרד: for each row, או לכל ההוראה ביחד: for each statement
when (condition)	אם רוצים להפעיל את הטריגר רק במקרים מסוימים אפשר להשתמש ב- when (לא חובה).
execute procedure trig_name();	זימון הפונקציה והפעלתה (אפשר לשלוח לפונקציה פרמטרים לשימושה בתוך הסוגריים)