תוכן עניינים

SQL

SQL היא שפת שאילתות המסוגלת לגשת ולבצע מניפולציות על מאגרי נתונים.

יכולות השפה

- הרצת שאילתות מול בסיסי נתונים
- שליפת מידע מבסיסי נתונים
- הזנת מידע לבסיס נתונים
- מחיקת מידע מבסיס נתונים
- עדכון מידע לבסיס נתונים
- יצירת בסיס נתונים חדש
- יצירת טבלאות בבסיס נתונים
- יצירת נהלים בבסיס נתונים
- יצירת תצוגות חדשות לצפייה בבסיס נתונים
- ניהול הרשאות לגישה לבסיסי נתונים.

פקודות SQL בסיסיות

- **SELECT** –שליפת נתונים לצפייה מבסיס נתונים קיים.
- **UPDATE** – עדכון מידע קיים בבסיס הנתונים.
- **DELETE** – מחיקת מידע קיים מבסיס נתונים.
- **INSERT INTO** – הזנת נתונים לבסיס נתונים קיים.
- **CREATE DATABASE** – יצירת בסיס נתונים חדש.
- **ALTER DATABASE** – עדכון בסיס נתונים קיים (הגדרות בלבד).
- **DROP DATABASE** – מחיקת בסיס נתונים.
- **BACKUP DATABASE** – גיבוי בסיס נתונים.
- **CREATE TABLE** – יצירת טבלה חדשה.
- **ALTER TABLE** – עדכון טבלה קיימת (הגדרות בלבד).
- **DROP TABLE** – מחיקת טבלה.
- **CREATE INDEX** – יצירת אינדקס (לצורך שליפת נתוני חיפוש מהירים יותר).
- **DROP INDEX** – מחיקת אינדקס.
- **SELECT DISTINCT** – שליפת נתונים תוך התעלמות מערכים כפולים בטור נבחר.

WHERE

פקודה המשמשת לסינון תוצאות.
<u>אופרטורים הניתנים לשימוש ב WHERE</u>:

- = שווה ל..
- < / =< גדול מ.. / גדול או שווה ל..
- > / => קטן מ.. / קטן או שווה ל...
- <> לא שווה ל.. (כמו !=)
- BETWEEN חיפוש בתוך טווח מוגדר מראש
- LIKE חיפוש תבנית מוגדרת מראש (<u>SQL WILDCARD CHARACTER</u>)
- IN הגדרת מספר ערכים לחיפוש בתוך טבלה (קיצור לשימוש בOR)

AND OR NOT

AND ו OR משמשים לביצוע שאילתת חיפוש בעלת יותר מתנאי אחד.
NOT משמש לשלילת תנאי שהוגדר אחריו – ז"א שהחיפוש יחזיר את הערכים שתוצאת התנאי שחל עליהם יהיה NOT TRUE.
ניתן לבצע שילוב של AND OR ו-NOT ללא הגבלה בשאילתה אחת.

NULL VALUES

במידה וטור בטבלה לא הוגדר כ- NOT NULL, יהיה ניתן לשמור בו ערכים ריקים מתוכן.
ניתן לחפש תאים ריקים בטבלאות ע"י שימוש בפקודה IS NULL (או IS NOT NULL – בדרך השלילה).

פקודות SQL נוספות

- **ORDER BY** – סידור סדר תוצאות החיפוש לפי טור שהוגדר מראש.
  ברירת המחדל היא סדר עולה (**ASC**), בכדי להציג בסדר הפוך יש להשתמש ב **DESC**.
- **TOP** \ **ROWNUM** \ **LIMIT** – משמשים להגבלת מספר התוצאות שיתקבל בחיפוש.
  על מנת להשתמש בנתון אחוזי ניתן להשתמש ב-PERCENT.
- **MIN()** and **MAX()** – איתור הערך הנמוך / הגבוה ביותר בעמודה נבחרת.
- **COUNT()** – מחזיר את מספר השורות שהתקבלו לפי פרמטרי החיפוש.
- **AVG()** – מחזיר את ערך הממוצע של כל הערכים בעמודה (עמודה חייבת להיות מוגדרת INT).
- **SUM()** – מחזיר תוצאת חיבור של כל הערכים בעמודה (עמודה חייבת להיות מוגדרת INT).

SQL WILDCARD CHARACTERS – שיפור תוצאות חיפוש

בכדי להרחיב את טווח החיפוש של הסנן LIKE, ניתן להשתמש בסימנים מיוחדים אשר יוכלו לעזור לצמצם או להגדיל את תוצאות החיפוש בהתאם לפרמטרים שנכניס.

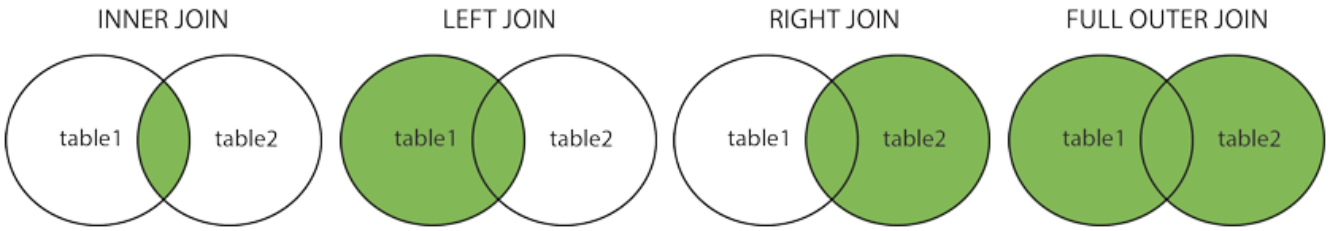
<u>סימנים מיוחדים שניתן לשלב עם LIKE:</u>

%	– מייצג תו או מספר תווים.
_	– מייצג תו בודד.
%[bsp]	– מייצג ערך שהאות הראשונה שלו <u>מכילה</u> את אחת מהאותיות b, s, p.
%[a-c]	– מייצג ערך שהאות הראשונה שלו <u>בטווח</u> שבין a לבין c (a, b, c).
%[!bsp]	– מייצג ערך שהאות הראשונה שלו <u>אינה מכילה</u> את אחת מהאותיות b, s, p.

SQL JOINS - איחוד טבלאות

ON *<table>* **JOIN** היא פונקציה בSQL, המאפשרת לאחד 2 טבלאות או יותר לטבלה אחת, בהתבסס על מזהה זהה ביניהם.
קיימים מספר סוגי JOIN:

- INNER JOIN - מחזיר ערכים בעלי מזהה ייחודי ב2 הטבלאות
- OUTER JOIN - LEFT: מחזיר את כל הערכים מהטבלה השמאלית ורק את הערכים בעלי המזהה מהטבלה הימנית.
  RIGHT: מחזיר את כל הערכים מהטבלה הימנית ורק את הערכים בעלי המזהה מהטבלה השמאלית.
  FULL: מחזיר את כל התוצאות כאשר יש התאמת מזהה גם בטבלה הימנית וגם בשמאלית.



- SELF JOIN – הוספת טבלה נוספת לטבלה הקיימת ע"י שימוש בהוספתה ל FROM עם פסיק מפריד.

SQL UNION – שימוש בכמה פעולות SELECT

UNION נועד על מנת לבצע 2 או יותר פעולות SELECT בשאילתה אחת (לשלוף מידע מכמה טבלאות).
שימוש ב- UNION מבצע DISTINCT אוטומטית (מתעלם מערכים כפולים)
בכדי לשלוף את כל המידע הקיים ב2 הטבלאות, ניתן להשתמש ב UNION ALL.

ניתן להשתמש ב UNION אך ורק בתנאים הבאים:

- מספר עמודות שווה בין 2 הטבלאות.
- סוג המידע צריך להיות תואם בין 2 הטבלאות.
- בכל SELECT העמודות חייבות להיות בסדר זהה.

SQL ALIASES (AS) – הגדרת כינויים לעמודות

ניתן להגדיר את תוצאות החיפוש עם עמודות מותאמות אישית ע"י שימוש ב AS ובחירת שם חדש לעמודה.
תכונה זו משמשת בעיקר לטעמי נוחות משתמש.

פונקציות צבירה

פונקציית צבירה היא פונקציה המחזירה שורה אחת מעמודה שנבחרה.
פונקציות הצבירה בSQL הינן: COUNT, MAX, MIN, SUM, AVG, GROUP BY

GROUP BY – איחוד ערכים

GROUP BY היא פונקציית צבירה המשמשת לאיחוד ערכים - מחזירה תוצאה של שורה אחת מכל ערך מהעמודה שנבחרה.

HAVING

HAVING נוצר ב SQL מפני ש WHERE לא יכול לכלול בתוכו פונקציות צבירה, ולכן יש לכלול את פונקציות הצבירה בתוכו.

EXISTS OPERATOR

הפונקציה EXISTS נועדה על מנת לבחון אם ערך מסויים קיים בשאילתת משנה ולהחזיר תוצאות שערכן הוא TRUE.
EXISTS יוכנס בתוך WHERE ושאילתת המשנה תוכנס בתוך

```
<query> WHERE EXISTS (<sub query>)
```

ANY & ALL

תפקידן של ALL ו ANY הוא להחזיר TRUE או FALSE:

- ALL - במידה וכל התנאים מתקיימים בשאילתת המשנה.
- ANY – במידה וחלק מהתנאים מתקיימים בשאילתת המשנה.

ניתן להשתמש בפונקציה בצירוף אופרטור (=, <>, !=, >, =<, <, =>) בתוך HAVING או WHERE,
אחריה יש להוסיף בסוגריים את שאילתת המשנה. לדוגמה:

```
<query> WHERE `foo` = ANY (<sub query>)
```

SELECT INTO

SELECT * INTO מבצע העתקה של כל הנתונים של טבלה קיימת לטבלה חדשה (כמו גיבוי).
במידה ויש צורך בהעתקה לבסיס נתונים קיים, יתווסף IN כדוגמת התחביר הבא:

```
SELECT * INTO <newtable> IN externaldb.mdb FROM <oldtable> WHERE <condition>;
```

במידה ואף אחד מהנתונים לא עומד על התנאי של השאילתה, תיווצר טבלה חדשה ריקה.

INSERT INTO SELECT

INSERT INTO SELECT מבצע העתקה מטבלה אחת לטבלה אחרת.

```
INSERT INTO <table2> SELECT * FROM <table1> WHERE <condition>;
```

- INSERT INTO SELECT מחייב תאימות של סוג מידע (INT \ VARCHAR) בין 2 הטבלאות.
- הנתונים הקיימים בטבלת היעד אינם מושפעים מהפעולה.

COMMENTS – הערות

הערות נועדו על מנת לתאר את הקוד או בכדי לנטרל הרצה של שורות בקוד.

-- <comment>	– שורת קוד אחת מסומנת כהערה
/* <comment> */	– קטע קוד מסומן כהערה

CASE

CASE עובר על סדרת תנאים ומחזיר את התנאי הראשון שתוצאתו תהיה TRUE (כמו IF THEN ELSE Statements).
במידה ואף אחד מהתנאים לא יחזיר תוצאת TRUE, יוחזר NULL.
CASE נרשם בתחביר הבא:

```
CASE
    WHEN <condition1> THEN <result1>
    WHEN <condition2> THEN <result2>
    WHEN <condition3> THEN <result3>
    ELSE result
END;
```

פונקציות NULL

במידה ונבצע פעולה כלשהי על תא שערכו ריק מתוכן, נקבל תוצאת NULL.
בכדי להתגבר על בעיה זו קיימות מספר פונקציות שנועדו לבחון האם הערך ריק או לא.
פונקציות הNULL משתנות בהתאם לסביבת העבודה:

- **MySQL**
  (**IFNULL**(*<value>*, *<new_value>*) - מאפשר להזין ערך אחר במידה והערך בתא ריק.
  (**COALESCE**(*<value>*, *<new_value>*) – מחזיר את הערך הראשון שאינו NULL מתוך רשימה.
- **SQL Server**
  (**ISNULL**(*<value>*, *<new_value>*) – מאפשר להזין ערך אחר במידה והערך בתא ריק.
- **MS Access**
  (**IsNull**(*<value>*, *<new_value>*) – פונקציה זו מחזירה TRUE (-1) אם התא ריק, ו FALSE (0) אם לא.
- **Oracle**
  (NVL(*<value>*, *<new_value>*) – מאפשר להזין ערך אחר במידה והערך בתא ריק.

SQL STORED PROCEDURES – שמירת שאילתות

PROCEDURE נועד על מנת לשמור שאילתות בכדי להריץ אותם מאוחר יותר.
התחביר נראה כך:

```
CREATE PROCEDURE <procedure_name>
AS
SELECT * FROM <column> WHERE <condition>
GO;
```

על מנת להריץ את השאילתה בהמשך, יש להשתמש בפקודה:

```
EXEC <procedure_name>;
```

DATES – תאריכים ושעות

ניתן למקד שאילתות באמצעות שימוש בתאריך או שעה, אותם יש לרשום בפורמט הבא:

- פורמט תאריך – YYYY-MM-DD
- פורמט תאריך ושעה – YYYY-MM-DD HH:MI:SS
- פורמט שנה – YY or YYYY

SQL CONSTRAINTS

SQL constraints נועדו על מנת להגדיר את מאפייני הערכים בעמודה שבטבלה ובכדי לוודא שהערך שהוכנס זהה לשאר הערכים האחרים שכבר נמצאים בטבלה או שיוכנסו בעתיד.
תחביר:

```
CREATE TABLE <table_name> (
    <column1> <datatype> constraint,
    <column2> <datatype> constraint,
    <column3> <datatype> constraint,
    ....
);
```

- NOT NULL – מחייב הזנת נתונים לעמודה.
- UNIQUE – מחייב הזנת ערך ייחודי לעמודה (ערך שונה משאר הערכים באותה עמודה)
- PRIMARY KEY – מחייב הזנת נתונים + ערך ייחודי לעמודה (שילוב של NOT NULL ו- UNIQUE)
- FOREIGN KEY – מזהה ייחודי שקיים גם בטבלה אחרת, מיועד על מנת לחבר 2 טבלאות (או יותר).
- CHECK – מחייב הזנת נתונים שעומדים בתנאי ספציפי שאותו נגדיר.
- DEFAULT – ערך ברירת מחדל שיוכנס לטבלה במידה ולא יוכנס ערך אחר במקומו.
- AUTO INCREMENT – אופציה שבמידה ומסומנת בV, לכל רשומה חדשה יצורף מזהה ייחודי באופן אוטומטי.

SQL VIEWS

ניתן לבצע שמירת תצוגה וירטואלית של מה שנרצה לחפש, תחת שם חדש שנקרא VIEW.
לצורך יצירת VIEW יש להשתמש בתחביר הבא:

```
CREATE VIEW <view_name> AS
SELECT <column1>, <column2>, ...
FROM <table_name>
WHERE <condition>;
```

כשנרצה להשתמש בתצוגה הוירטואלית, נתייחס אליה כאל טבלה רגילה:

```
SELECT * FROM <view_name>;
```

<u>פקודות VIEW נוספות</u>:

- `CREATE OR REPLACE VIEW` – עדכון VIEW קיים.
- `DROP VIEW <view_name>` - מחיקת VIEW.