SQL

Structured Query Language

זו שפת הדטהבייסים הרלציוניים. השפה עצמה מגדירה את כל שלל הפקודות האפשריות שדטהבייס יכול לקבל. כמובן שברמת העקרון היא הייתה אמורה להיות גם סטנדרטית כך שפקודת SQL מדטהבייס אחד (לדוגמא SQLServer) תרוץ בלי בעיה בדטהבייס אחר (לדוגמא Postgres), אבל לא כך הדבר. כל דטהבייס המציא לעצמו "ניב" שונה של SQL למרות שהבסיס מאד מאד דומה.

אפשר לחלק את שפת ה-SQL למספר תתי שפות. למשל, DDL (Data Definition Language) מכילה את שלל הפקודות המטפלות ביצירת סכמה, כלומר יצירת טבלאות, מחיקת טבלאות, שינוי טבלאות וכיו"ב.

אנחנו נלמד את משפחת ה-DML ו-DQL, שזה בהתאמה: Data Modification Language ו-Data Query Language ובמילים אחרות, פקודות ה-CRUD

Crud – create, retrieve, update, delete

SQL – insert, select, update, delete

הסיבה היא שאת רוב פקודות ה-DDL גם כך נייצר בממשקים גרפיים, אך את פקודות ה-CRUD נתכנת באפליקציות שלנו.

הפקודות מתייחסות לא למבנה הטבלה, אלא לנתונים בטבלה.

נתחיל ב-select וזו הפקודה הכי מורכבת.

קודם כל מבנה כללי:

SELECT {'what data'}

FROM {'which tables'}

WHERE {'filter'}

GROUP BY {'aggregate'}

HAVING {'aggregate filter'}

ORDER BY {'columns to order by'}

LIMIT {'limit results'}

התוצאה של פקודת select תהיה תמיד טבלה, מכנים אותה גם result set

בחירת קבוע מתוך הדטהבייס:

SELECT 1;

בחירת חישוב מתוך הדטהבייס:

SELECT 2\*18;

בחירת השעה הנוכחית בדטהבייס:

SELECT now();

בחירת כל הנתונים מטבלה מסויימת

SELECT \*

FROM products;

בחירת עמודות מסויימות בלבד:

SELECT productName, UnitsInStock, UnitPrice

FROM products;

שליטה בשמות העמודת של ה-result set תוך שימוש בפקודת AS:

SELECT productName,

UnitsInStock AS unitsInStock,

UnitPrice AS price

FROM products;

שליטה במיון הרשומות

SELECT productName,

UnitsInStock AS unitsInStock,

UnitPrice AS price

FROM products

ORDER BY price;

שליטה בסדר המיון באמצעות הפקודות ASC למיון עולה ו-DESC למיון יורד:

SELECT productName,

UnitsInStock AS unitsInStock,

UnitPrice AS price

FROM products

ORDER BY price ASC;

SELECT productName,

UnitsInStock AS unitsInStock,

UnitPrice AS price

FROM products

ORDER BY price DESC;

שליטה במיון משני. ניתן להוסיף מיונים משניים כמספר העמודות ב-result set:

SELECT productName,

UnitsInStock AS unitsInStock,

UnitPrice AS price

FROM products

ORDER BY price DESC,

unitsInStock ASC;

צמצום ה-result set על ידי הגבלת מספר השורות בה באמצעות הפקודה limit

SELECT productName,

UnitsInStock AS unitsInStock,

UnitPrice AS price

FROM products

ORDER BY price DESC

LIMIT 10;

pagination - הגבלת מספר התוצאות, החל מ-offset מסויים:

SELECT orderID, OrderDate

FROM orders

ORDER BY OrderDate ASC

LIMIT 20, 10;

כך מיישמים בעצם דפדוף עבור המשתמש. משתמשים ב-limit קבוע, ומשנים כל פעם את ה-offset

בדרך כלל הפקודה LIMIT תגיע אחרי ORDER BY. אין הרבה משמעות ל-LIMIT ללא מיון.

שליפת ערכיים ייחודיים מתוך ה-result set:

SELECT DISTINCT city

FROM employees;

פילטור הרשימה באמצעות הפקודה WHERE:

SELECT \*

FROM products

WHERE UnitPrice <= 10

פילטור לפי תנאי מורכב באמצעות הפקודה AND:

SELECT \*

FROM products

WHERE UnitPrice >= 10 AND UnitPrice <= 20;

פילטור לפי תנאי מורכב באמצעות הפקודה OR:

SELECT \*

FROM products

WHERE UnitPrice <= 10 OR UnitPrice >= 20;

שימוש בפקודה NOT כדי לסנן בצורה בוליאנית:

SELECT \*

FROM products

WHERE NOT Discontinued;

שימוש בפקודה BETWEEN כדי לחסוך >= AND <=:

SELECT \*

FROM products

WHERE unitPrice BETWEEN 10 AND 20;

שימוש בפקודה IN כדי לחסוך פקודות מרובות OR:

SELECT \*

FROM products

WHERE unitPrice IN (10, 15, 20);

חיפוש בשדות טקסטואליים באמצעות הפקודה LIKE. תביא לי את כל המוצרים שמתחילים במחרוזת Chef Anton:

SELECT \*

FROM products

WHERE ProductName LIKE 'Chef Anton%';

פונקציות סקלריות. זוהי פונקציה המקבלת ערך אחד או יותר אך מחזירה ערך בודד, ולכן ניתן להשתמש בה כעמודה ב-result set

למשל הפונקציה FLOOR מעגלת למטה

SELECT ProductName,

FLOOR(UnitPrice)

FROM products;

CEILING יעגל למעלה:

SELECT ProductName,

CEILING(UnitPrice)

FROM products;

או לעגל בכלליות (יעגל למעלה מה שיותר קרוב למעלה, ולמטה מה שיותר קרוב למטה)

SELECT ProductName,

ROUND(UnitPrice)

FROM products;

הפיכת מחרוזת לאותיות גדולות:

SELECT UPPER(ProductName) as upperName,

ROUND(UnitPrice) as roundedPrice

FROM products;

או באותיות קטנות:

SELECT LOWER(ProductName) as upperName,

ROUND(UnitPrice) as roundedPrice

FROM products;

חיתוך חלק התאריך מתוך DATETIME:

SELECT FirstName,

LastName,

DATE(BirthDate)

FROM employees;

או הוצאת רק שנת הלידה מתוך התאריך:

SELECT FirstName,

LastName,

YEAR(BirthDate)

FROM employees;

פירמוט תאריך אמריקאי למשל באמצעות DATE\_FORMAT:

SELECT FirstName,

LastName,

DATE\_FORMAT(BirthDate, '%W, %b %D, %Y')

FROM employees;

פונקציות אגרגציה. אלה הן פונקציות המבצעות חישוב על מספר ערכים ומחזירות ערך בודד

ממוצע:

SELECT AVG(unitPrice)

FROM products;

מקסימום:

SELECT MAX(unitPrice)

FROM products;

מינימום:

SELECT MIN(unitPrice)

FROM products;

סכום:

SELECT SUM(unitPrice)

FROM products;

ספירה:

SELECT COUNT(unitPrice)

FROM products;

כמה שורות יש בטבלה:

SELECT COUNT(\*)

FROM products;

ספירת ערכים ייחודים בטבלה עבור עמודה מסויימת :

SELECT COUNT(DISTINCT city)

FROM employees;

מחיר ממוצע פר קטגוריה באמצעות הפקודה GROUP BY:

SELECT CategoryID,

AVG(UnitPrice)

FROM products

GROUP BY CategoryID;

פילטור אגרגציות באמצעות HAVING:

SELECT CategoryID,

AVG(UnitPrice) AS AveragePrice,

SUM(UnitsInStock) AS TotalInStock

FROM products

GROUP BY CategoryID

HAVING TotalInStock < 200;

איחוד result sets משתי שאילתות שונות באמצעות הפקודה UNION:

SELECT phone

FROM suppliers

UNION

SELECT phone

FROM customers;

הפקודה UNION מפלטרת שורות זהות מה-result set. כדי למנוע ממנה לעשות זאת, יש להשתמש ב-UNION ALL:

SELECT companyName, phone

FROM suppliers

UNION ALL

SELECT companyName, phone

FROM customers;

פקודות UNION ALL הרבה יותר מהירות מפקודות UNION משום שחוסכים מהדטהבייס את הצורך לפלטר שורות כפולות

תת-שאילתא

במקום שהאפליקציה תבצע שתי שאילתות עוקבות לדטהבייס, ניתן להבעיר לדטהבייס שאילתא אחת המורכבת משתי שאילתות:

SELECT ProductID, ProductName

FROM products

WHERE UnitPrice > ( SELECT AVG(unitPrice)

FROM products);

הרבה פעמים נמצא תת שאילתות ביחד עם הפקודה IN:

SELECT ProductID, ProductName

FROM products

WHERE CategoryId IN (SELECT categoryId

FROM categories

WHERE categoryName LIKE 'B%');

JOIN

נניח שאני רוצה להראות נתונים משתי טבלאות בו זמנית. למשל מוצרים ושמות הקטגוריות אליהן הם שייכים. ממה שלמדנו עד כה, ניתן להגשים זאת על ידי שימוש בתת שאילתא, בחלק של ה-SELECT (בניגוד לחלק של ה-WHERE).

SELECT ProductID, productName, (SELECT categoryName

FROM categories

WHERE CategoryID = products.CategoryID)

FROM products;

מה הבעיה? שאילתא זו תהיה מאד לא יעילה. מכיוון שעבור כל מוצר, תרוץ עוד שאילתא על טבלת קטגוריות. אם יש לי 1000 מוצרים, כתבתי כאן שאילתא שתרוץ 1001 פעמים.

לצורך כך המציאו את ה-JOIN

להלן JOIN בסיסי בין שתי טבלאות, ללא תנאי JOIN. מכיוון שהדטהבייס לא יודע על פי מה הוא אמור לצרף את שתי הטבלאות, הוא מייצר מכפלה קרטזית ביניהן. קרי, כל שורה מהמוצרים, תצטרף לכל שורה מהקטגוריות. מספר השורות הסופי, יהיה כמכפלת מספר המוצרים במספר הקטגוריות:

SELECT ProductID, ProductName, CategoryName

FROM products JOIN categories -- ON products.CategoryID = categories.CategoryID;

סביר מאד להניח שמעולם לא נצטרך באמת מכפלה קרטזית, חשוב לזכור לא להשמיט את תנאי ה-JOIN המצויין בפקודה ON:

SELECT ProductID, ProductName, CategoryName

FROM products JOIN categories ON products.CategoryID = categories.CategoryID;

אם אני רוצה לכלול ב-resultSet שדה ששמו זהה בין שתי טבלאות המשתתפות ב-JOIN עליי לציין מפורשות מאיזה טבלה להביאו:

SELECT ProductID, ProductName, CategoryName, products.CategoryID

FROM products JOIN categories ON products.categoryID = categories.CategoryID;

כשעושים JOIN עם resultSet מכובד, נהוג לציין עבור כל שדה מאיזה טבלה הגיע, גם אם אינו ambiguous, ולכן נהוג לתת אליאסים לשמות הטבלאות ולהשתמש בהם במקום שם הטבלה הארוך:

SELECT p.ProductID,

p.ProductName,

c.CategoryName,

p.CategoryID

FROM products AS p JOIN categories as c ON p.categoryID = c.CategoryID;



INNER JOIN - ג'וין המביא רק איברים שיש קשר ביניהם בשתי הטבלאות. זהו ה-JOIN הדיפולטי ב-mysql ולא צריך לציין INNER JOIN למרות שניתן.

LEFT JOIN - ג'וין המביא את כל הנתונים מהטבלה השמאלית ורק את הנתונים שלהם יש קשר אמיתי מהטבלה הימנית:

SELECT p.ProductID,

p.ProductName,

c.CategoryName,

p.CategoryID

FROM products AS p LEFT JOIN categories as c ON p.categoryID = c.CategoryID;

RIGHT JOIN - ג'וין המביא את כל הנתונים מהטבלה הימנית ורק את הנתונים שלהם יש קשר אמיתי מהטבלה השמאלית:

SELECT p.ProductID,

p.ProductName,

c.CategoryName,

p.CategoryID

FROM products AS p RIGHT JOIN categories as c ON p.categoryID = c.CategoryID;

FULL JOIN - ג'וין היביא את כל הנתונים משתי הטבלאות, הן אלו שעומדים בתנאי ה-ON והן אלו שלא. MySQL לא תומך בפול ג'וין ולכן יש ליישם את הטריק הבא: left join שעושה union עם right join:

SELECT p.ProductID,

p.ProductName,

c.CategoryName,

p.CategoryID

FROM products AS p LEFT JOIN categories as c ON p.categoryID = c.CategoryID

UNION

SELECT p.ProductID,

p.ProductName,

c.CategoryName,

p.CategoryID

FROM products AS p RIGHT JOIN categories as c ON p.categoryID = c.CategoryID;

SELF JOIN - לפעמים נרצה לצרף טבלה לעצמה. למשל נרצה לראות טבלת עובדים והבוסים שלהם, כשהבוסים עצמם הם רשומות בטבלת עובדים:

SELECT emps.FirstName, emps.LastName, bosses.FirstName, bosses.LastName

FROM employees AS emps LEFT JOIN employees AS bosses ON emps.ReportsTo = bosses.EmployeeID;

הוספת רשומות. פקודת INSERT

לפקודת ה-INSERT שני נוסחים. נוסח ראשון, שבו אני מציין מפורשות ערך עבור כל אחד מהשדות בטבלה. אם אין לי ערך להכניס עבור שדה, אני יכול להשתמש במילות המפתח DEFAULT או NULL:

INSERT INTO categories VALUES(NULL, 'Kosher', 'Kosher Food', NULL);

INSERT INTO categories VALUES(DEFAULT, 'Gloten Free', 'For celiak people', NULL);

אם אין בידי ערכים לרוב השדות, אפשר להשתמש בנוסח מקוצר המאפשר לציין לאילו שדות אני רוצה להכניס ערכים. לשאר השדות ייכנס ה-DEFAULT או NULL אם אין DEFAULT:

INSERT INTO suppliers(City, CompanyName) VALUES('Washington D.C.', 'White House');

ניתן להכניס מספר רשומות בבת אחת על ידי שימוש ב-SELECT (שימוש לא אפליקטיבי, אלא ידני ודי נדיר, בעיקר נועד לתחזוקת דטהבייסים של פיתוח וכל מיני דברים כאלה):

insert into suppliers(city, companyName) select city, firstName from employees;

עדכון - פקודת UPDATE

מבנה כללי:

UPDATE {which table}

SET {which field} = {which value}

[WHERE] {field} = {value}

אם לא נשתמש ב-WHERE יתעדכנו כל השורות בטבלה. לפעמים נרצה לעשות זאת. רוב הפעמים לא נרצה, היזהרו!

UPDATE employees

SET birthDate = '2000-05-25',

HireDate = now()

WHERE firstName = 'Bar';

מחיקה - פקודת DELETE

DELETE FROM {table name}

[WHERE] {field} = {value}

DELETE ללא WHERE מוחק את כל הנתונים בטבלה. שימו לב, הוא לא מוחק את הטבלה עצמה אלא רק את הנתונים בה (מחיקת טבלה מתבצעת בפקודת ה-DDL DROP TABLE):

DELETE FROM {table name}

[WHERE] {field} = {value}

DELETE FROM employees

WHERE employeeId = 10;