استفاده از توابع نگاشت و فیلتر و کاهش باعث افزایش کارایی برنامه می شوند؟ آیا می توانید سرعت اجرای این توابع را با برنامه نویسی رویه ای توسط حلقهٔ تکرار مقایسه کنید؟

بله، استفاده از توابع نگاشت و فیلتر و کاهش در زبان SQL می تواند باعث افزایش کارایی برنامه شود. این توابع به ما اجازه می دهند که بر روی دنباله هایی از عناصر عملیات انجام دهیم بدون اینکه نیاز به نوشتن حلقه های تکرار یا شرط های کنترل جریان داشته باشیم. این توابع همچنین می توانند به صورت موازی اجرا شوند و از منابع پردازشی بهتری استفاده کنند. برای مثال، اگر بخواهیم تعداد کاراکترهای هر خط در یک جدول را بدست آوریم، می توانیم از تابع map استفاده کنیم:

```
SELECT map(line, length(line)) FROM table;
```

این تابع برای هر خط در جدول یک جفت از خط و طول آن را برمی گرداند. اگر بخواهیم فقط خط هایی را که بیشتر از 10 کاراکتر دارند را انتخاب کنیم، می توانیم از تابع filter استفاده کنیم:

```
SELECT filter(map(line, length(line)), (k, v) -> v > 10)
FROM table;
```

این تابع برای هر جفت خط و طول آن یک شرط را بررسی می کند و فقط جفت هایی را که شرط را برآورده می کنند را برمی گرداند. اگر بخواهیم مجموع طول همه خط ها را محاسبه کنیم، می‌توانیم از تابع reduce استفاده کنیم:

```
SELECT reduce(map(line, length(line)), 0, (acc, x) -> acc +
x) FROM table;
```

این تابع یک مقدار اولیه (0) و یک تابع دوم (جمع) را گرفته و برای هر جفت خط و طول آن مقدار اولیه را با طول خط جمع می کند و نتیجه را برمی گرداند.

اگر بخواهیم همین کار را با برنامه نویسی رویه ای توسط حلقه تکرار انجام دهیم، باید کد بیشتری بنویسیم و ممکن است با خطاهای بیشتری روبرو شویم. برای مثال، اگر بخواهیم مجموع طول همه خط ها را محاسبه کنیم، باید چنین کدی بنویسیم:

```
DECLARE @total INT = 0;
DECLARE @line VARCHAR(100);
DECLARE @length INT;
DECLARE cursor CURSOR FOR SELECT line FROM table;
OPEN cursor;
FETCH NEXT FROM cursor INTO @line;
WHILE @@FETCH_STATUS = 0
BEGIN
 SET @length = LEN(@line);
 IF @length > 10
 BEGIN
  SET @total = @total + @length;
 END
 FETCH NEXT FROM cursor INTO @line;
END
CLOSE cursor;
DEALLOCATE cursor;
SELECT @total;
```

همانطور که می بینید، این کد خوانایی کمتری دارد و از ساختارهای پیچیده تری مانند مکانیزم cursor استفاده می کند. همچنین این کد به صورت ترتیبی اجرا می شود و از منابع پردازشی کمتری بهره می برد. بنابراین، استفاده از توابع نگاشت و فیلتر و کاهش در زبان SQL می تواند باعث افزایش کارایی برنامه شود.

مکانیزم Cursor:

مکانیزم cursor در زبان SQL یک اشیای پایگاه داده است که به ما اجازه می دهد تا بر روی سطوح نتیجه یک جستجوی کار کنیم. با استفاده از cursor، ما می توانیم هر سطر را به صورت جداگانه معامله کنیم و از منابع پردازشی بهتری استفاده کنیم. برای استفاده از cursor، ما باید چند مرحله را طی کنیم:

- اول، یک cursor را تعریف کنیم. برای این کار، ما باید نام cursor را بعد از کلمه DECLARE با نوع CURSOR و یک جمله SELECT را تعریف کنیم. جمله SELECT باید نتیجه جستجوی خود را تعریف کند.

- دوم، cursor را باز و پر دهیم. برای این کار، ما باید جمله SELECT را در حالت OPEN قرار دهیم.

- سوم، هر سطر را از cursor در یک یا چند متغیر دریافت کنیم. برای این کار، ما باید جملات FETCH NEXT FROM و INTO را در حالت WHILE قرار دهیم.

- چهارم، cursor را بسته و منصوب کنیم. برای این کار، شما باید جملات CLOSE و DEALLOCATE را در حالت WHILE قرار دهیم.

برای مثال، فرض کنید بخواهید نام و سن همه دانش آموزانی را که در کلاس 10 هستند را از یک جدول به نام students بدست آورید. برای این کار، می توانید از یک cursor به نام student_cursor استفاده کنید:

```
-- تعریف cursor
DECLARE student_cursor CURSOR FOR
SELECT name, age FROM students
WHERE class = 10;
```

```
-- باز و پر کردن cursor
OPEN student_cursor;

-- ایجاد متغیرهای محلی
DECLARE @name VARCHAR(50);
DECLARE @age INT;

-- دریافت هر سطر از cursor
FETCH NEXT FROM student_cursor INTO @name, @age;

-- حلقه تکرار برای پردازش هر سطر
WHILE @@FETCH_STATUS = 0
BEGIN
-- نمایش نام و سن دانش آموز
PRINT @name + ' is ' + CAST(@age AS VARCHAR) + ' years
old.';
-- دریافت سطر بعدی از cursor
FETCH NEXT FROM student_cursor INTO @name, @age;
END

-- بسته و منصوب کردن cursor
CLOSE student_cursor;
DEALLOCATE student_cursor;
```

این کد برای هر سطر در جدول students که در کلاس 10 هستند، نام و سن آنها را نمایش می دهد. اما اگر بخواهیم همین کار را با استفاده از تابع map انجام دهیم، می توانیم از یک جمله SQL ساده تر استفاده کنیم:

```
-- استفاده از تابع map
SELECT map(name, age) FROM students
WHERE class = 10;
```

این جمله برای هر سطر در جدول students که در کلاس 10 هستند، یک جفت از نام و سن آنها را برمی گرداند. همانطور که می بینید، استفاده از تابع map کد را خواناتر و کوتاه تر می کند.

منابع:

- https://ocw.mit.edu
- https://learn.microsoft.com
- https://developerhowto.com
- https://stackoverflow.com
- https://docs.databricks.com/en/sql
- https://www.sqlservertutorial.net
- https://www.sqlshack.com