Store Procedure

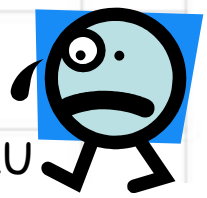
ข้อดีของการใช้ Store Procedure

1. ทำให้โปรแกรมมีลักษณะเป็นโมดูลมากขึ้น โดยสามารถสร้าง SP เพียงครั้งเดียวและเก็บไว้ในฐานข้อมูล เมื่อต้องการก็สามารถเรียกได้หลายครั้ง และสามารถแก้ไขได้อิสระจากโปรแกรมที่ไคลเอ็นต์
2. ทำให้การรันโปรแกรมเร็วขึ้น ถ้าการทำงานนั้นต้องใช้คำสั่ง SQL จำนวนมากทำงานซ้ำ ๆ กัน SP จะทำงานได้อย่างมีประสิทธิภาพ เนื่องจาก Execution Plan จะถูกเก็บไว้ในหน่วยความจำ จึงทำให้การเรียกใช้ SP ครั้งต่อไปทำได้รวดเร็วขึ้น
3. ช่วยลดแบนวิธที่ใช้บนระบบเน็ตเวิร์กลง เพราะการกระทำที่ต้องใช้คำสั่ง T-SQL จำนวนหลายบรรทัดจะสามารถทำผ่านคำสั่งเพียงคำสั่งเดียวได้
4. SP ช่วยเพิ่มความปลอดภัยให้มากขึ้นได้ ผู้ใช้จะถูกกำหนดสิทธิ์ในการรัน SP และผู้เรียกใช้ไม่จำเป็นต้องเข้าใจโครงสร้างของฐานข้อมูล

Store Procedure

```
CREATE PROCEDURE procedure_name
    [{@parameter data_type}[=default][OUTPUT][,...]]
    [WITH ENCRYPTION]
AS
    sql_statement
GO
```

procedure_name	: ชื่อของ Store Procedure
parameter	: ชื่อพารามิเตอร์ที่เรากำหนด พร้อมกับชนิดข้อมูลของพารามิเตอร์นั้น
default	: เป็นการกำหนดค่าดีฟอลต์ให้กับพารามิเตอร์นั้น เมื่อเราไม่ได้กำหนดค่าให้กับพารามิเตอร์นั้นในตอนเรียก
OUTPUT	: กำหนดให้ส่งค่าพารามิเตอร์กลับได้
ENCRYPTION	: เป็นการเข้ารหัสของคำสั่ง SP

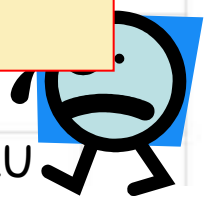


Store Procedure

```
CREATE PROCEDURE my_proc1
      @first int = NULL,
      @second int = 2,
      @third int = 3
AS
      SELECT @first, @second, @third
GO
```

```
CREATE PROCEDURE my_proc2
      @Item1 varchar(11)
AS
      SELECT * FROM TStudent WHERE Std_group = @vItem
GO
```

```
EXECUTE my_proc1 1,2,3;
EXECUTE my_proc2 '501226601';
```



CURSOR

Cursor คือเครื่องมือของ SQL Server สำหรับใช้ในการจัดการข้อมูลในเรคอร์ดเซตในแบบแต่ละเรคอร์ด โดยจะมีพอยต์เตอร์ชี้ไปยังเรคอร์ดที่ทำงาน ซึ่งมีความจำเป็นสำหรับงานที่ต้องเข้าถึงการทำงานในระดับเรคอร์ด

ข้อดีของการใช้เคอร์เซอร์

1. สามารถทำงานอย่างมีเงื่อนไขบนแต่ละเรคอร์ดได้แตกต่างกัน ซึ่งการทำงานอย่างนี้จะต้องรันบนโปรแกรมที่ซับซ้อน
2. การใช้เคอร์เซอร์จัดการแบบเรคอร์ดต่อเรคอร์ดมีประสิทธิภาพกว่าการใช้คำสั่ง UPDATE เพื่อแก้ไขเรคอร์ดเป็นกลุ่ม
3. สามารถควบคุม Transaction ได้ดีกว่า เนื่องจากสามารถทำงานกับข้อมูลในแต่ละเรคอร์ดได้อย่างอิสระ

CURSOR

การวิ่งไปตามแถวข้อมูลซึ่งเป็นผลลัพธ์จากการคิวรีโดยการทำงานของเคอร์เซอร์จะมีขั้นตอนดังนี้

1. ประกาศสร้างเคอร์เซอร์เชื่อมกับคิวรีด้วยคำสั่ง DECLARE CURSOR

   รูปแบบ DECLARE <Alias> CURSOR FOR <SQL Statement>

   Ex. DECLARE tmp CURSOR FOR SELECT Std_id FROM TStudent
   WHERE Std_group = '501226601'

2. เปิดการใช้งานเคอร์เซอร์ด้วยคำสั่ง OPEN

   รูปแบบ OPEN <Alias>

   Ex. OPEN tmp

3. นำเคอร์เซอร์มาวิ่ง ด้วยคำสั่ง FETCH

   รูปแบบ FETCH NEXT FROM <Alias> INTO <parameter[,...]>

   Ex. FETCH NEXT FROM tmp INTO @Std

CURSOR

4. นำข้อมูลจาก แถวที่เคอร์เซอร์จอดอยู่ไปใช้งานที่ละแถว WHILE

   รูปแบบ WHILE @@FETCH_STATUS = 0

   หมายเหตุ @@FETCH_STATUS คือตัวแปร Global ที่ทำหน้าที่ให้ค่าสถานะของการดึงแถวข้อมูล โดยถ้าสามารถดึงได้ปกติจะให้ค่า 0 แต่ถ้าเกิด Error จะให้ค่า -1

5. เมื่อใช้เคอร์เซอร์เสร็จแล้ว ปิดเคอร์เซอร์ด้วยคำสั่ง CLOSE

   รูปแบบ CLOSE <Alias>

   Ex. CLOSE tmp

6. ทวนคืนหน่วยความจำที่ใช้สำหรับเคอร์เซอร์ด้วยคำสั่ง DEALLOCATE

   รูปแบบ DEALLOCATE <Alias>

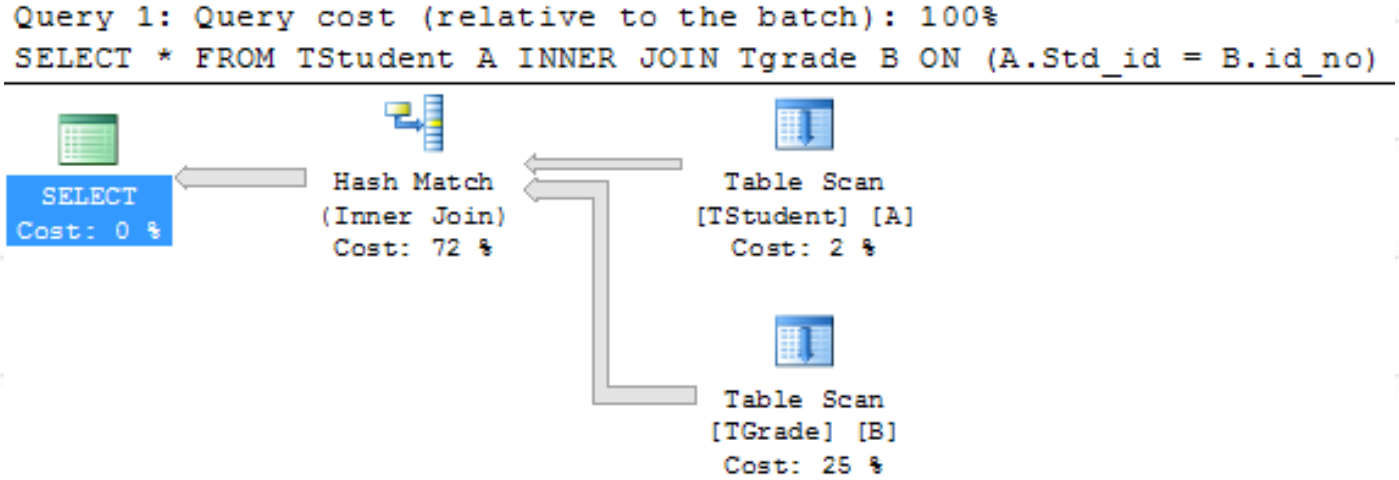
   Ex. DEALLOCATE tmp

คิวรีประสิทธิภาพสูง

คิวรีที่มีประสิทธิภาพคือคิวรีที่ทำงานได้รวดเร็ว กินทรัพยากรของระบบน้อย ดังนั้นเราควรปรับแต่งคิวรีให้มีประสิทธิภาพสูงสุด เราเรียกกระบวนการนี้ว่า Query Optimization โดยมีเครื่องมือช่วยในการวิเคราะห์คิวรีคือ แผนการทำงาน (Execution Plan: EP)

Execution Plan : EP

คือ แผนการทำงานเพื่อเป็นเครื่องมือในการวิเคราะห์หาต้นเหตุทำให้เกิดความช้าได้อย่างรวดเร็ว



Execution Plan

ค่าร้อยละของความสิ้นเปลือง

- Estimated I/O Cost
  ค่าประมาณการความสิ้นเปลืองของอินพุต/เอาต์พุต
- Estimated CPU Cost
  ค่าประมาณการความสิ้นเปลืองของตัวประมวลผล
- Estimated Operator Cost
  ค่าประมาณการความสิ้นเปลืองที่เกิดจากตัวดำเนินการนี้
- Estimated Number of Executions
  จำนวนครั้งที่คาดว่าจะต้องทำ
- Estimated Number of Rows
  จำนวนแถวข้อมูลที่ประเมินได้
- Estimated Row Size
  ขนาดของแถวข้อมูลหนึ่งแถวโดยประเมิน

Execution Plan

ไอคอนในแผนการทำงาน

- Clustered Index Scan (CIC)
  คือตัวดำเนินการสแกนดัชนีแบบคลัสเตอร์ตามชื่อไฟล์ที่ระบุ ซึ่งหากมีคำสั่ง WHERE ร่วมด้วย แถวข้อมูลผลลัพธ์จะถูกกรองตามเงื่อนไข
- NonClustered Index Scan (NIS)
  เหมือนกับ CIC แต่เป็นการกระทำกับดัชนีแบบที่เป็นนอนคลัสเตอร์
- Hash Match (HM)
  สร้างตารางแฮช เพื่อใช้ในการ Join, Distinct, Function และ Union
- Compute Scalar (CS)
  การคำนวณค่าเดี่ยว ๆ การกระทำนิพจน์ที่ให้ค่าเชิงเดี่ยว
- Merge Join (MJ)
  ตัวดำเนินการนี้มักพบเป็นส่วนหนึ่งของกระบวนการ Joint และ Union
- Table Scan (TS)
  คือกระบวนการไล่อ่อนข้อมูลในตารางทุกแถว และให้ผลลัพธ์ทุกแถว ยกเว้นจะมีคำสั่ง WHERE ร่วมอยู่ด้วย

Execution Plan

วิธีปรับปรุงประสิทธิภาพคิวรี

- วิเคราะห์ Execution Plan

  1. อ่าน EP จากมุมบนขวาสุด และไล่การดำเนินการไปตามระดับเหมือนกับโครงสร้าง TREE จนกระทั่งถึง SELECT
  2. พิจารณาทุกกระบวนการว่าตัวดำเนินการบนแผนภูมิที่ปรากฏมีค่าสิ้นเปลืองโดยรวมมากที่สุดคืออะไรบ้าง

- ตัดการ Join ที่ไม่จำเป็น

  การทำ Query Optimization ง่ายที่สุดคือ ตัดการเชื่อมโยงของตารางที่ไม่จำเป็นออกให้มากที่สุด เพราะข้อมูลที่เกิดจากการเชื่อมโยงทำให้สิ้นเปลืองทรัพยากรมาก

- สร้าง Index ให้ครอบคลุม

  ในบางกรณีข้อมูลที่ผ่านการสร้างดัชนีแล้ว แต่ข้อมูลที่ต้องการสืบค้นอาจไม่ครบถ้วน ดังนั้นจึงควรสร้างดัชนีให้ครอบคลุมทุกข้อมูลที่ต้องการค้นหาและเชื่อมโยง

คิวรีประสิทธิภาพสูง

คำแนะนำเพื่อคิวรีประสิทธิภาพสูง

- ใช้ WHERE เพื่อลดข้อมูล
  ช่วยลดค่าสิ้นเปลืองของ I/O ได้มาก
- เลือกเฉพาะคอลัมน์ที่ต้องการ
  เลือเฉพาะบางคอลัมน์แทนที่จะดึงทุกคอลัมน์ด้วย * จะลดค่าสิ้นเปลือง I/O ได้เช่นกัน
- ใช้วิว
  การใช้วิวร่วมกับดัชนีของวิวจะลดเวลา ค่าสิ้นเปลืองของ CPU และ I/O ได้พร้อม ๆ กัน
- ใช้คำสั่ง TOP
  หากต้องการแสดงแถวข้อมูลตั้งแต่ 1 - n โดยใช้คำสั่ง TOP จะทำไม่ต้องสแกนตารางทั้งหมดจึงมีผลทำให้ประสิทธิภาพคิวรีดีขึ้น

คิวรีประสิทธิภาพสูง

คำแนะนำเพื่อคิวรีประสิทธิภาพสูง

- ไม่หาจำนวนแถวด้วย COUNT
  การนับจำนวนแถวข้อมูลด้วย COUNT ต้องสแกนตารางทั้งหมด หากตารางมีขนาดใหญ่จะใช้เวลานานมาก
- หลีกเลี่ยง CURSOR
  คิวรีที่ใช้เคอร์เซอร์อารช้ากว่าคิวรีแบบเซต 30 เท่า
- หลีกเลี่ยง TRIGGER
  การใช้ TRIGGER จะเป็นการจำกัดการทำงานมาก
- หลีกเลี่ยงคำสั่ง HAVING
  การใช้คำสั่ง HAVING ร่วมกับ GROUP BY โดยกรองข้อมูลกลุ่มที่ไม่ตรงเงื่อนไขออกไปนั้น ควรใช้ WHERE ดีกว่า
- หลีกเลี่ยงคำสั่ง DISTINCT
  คำสั่ง Distinct มีผลให้ประสิทธิภาพของคิวรีด้อยลง