Complex Query

(Лаборатори №6)

У. Төрболд

ХШУИС, Мэдээллийн технологийн IV-р түвшний оюутан, turbold1125@gmail.com

1. ОРШИЛ

Энэхүү лабораторид SQL – ийн функцууд буюу тодорхой хэрэглэгчийн тодорхойлсон функц болон Stored procedure буюу урьдчилан эмхэтгэсэн SQL хэллэгийг ашиглан кодыг модульчлах, дахин ашиглах, урьдчилан бэлдсэн код дуудсанаар өгөгдлийн сангийн гүйцэтгэлийг сайжруулна. Энэхүү лабораторид SQL функцуудын тухай ойлголт, stored procedure ашиглан Complex Query бичих үүнийг практик хэрэглээнд хэрхэн ашиглах талаар судлах болно.

2. ЗОРИЛГО

Энэхүү лабораторын зорилго Complex Query (Function, Stored procedure, Trigger) – ын талаар практик дээр хэрэгжүүлэх бөгөөд үүнийг хэрэгжүүлснээр кодын зохион байгуулалт, дахин ашиглах, гүйцэтгэлийг сайжруулах боломжтой болно. Үүний тулд дараах зорилтуудыг хэрэгжүүлэхийг зорьлоо.

- Хүн ам бүртгэлийн мэдээлэлд үндэслэн насаар нь ангилах функц бичих. Энэ функц нь тухайн хотод хэдэн хүн байгаа тоог харуулах, насаар нь харуулах зэргээр хүн амын нягтаршилыг тооцоолох зорилготой.
- Үйлчлүүлэгчдийн оруулсан орлогыг шинжилж, төлбөрийн жагсаалт гаргах.
- Зарагдсан барааны мэдээллийг тоо ширхэг, мөнгөн дүн зэрэг өгөгдлийн хамт жагсааж temporary table – д хадгал. Энэ мэдээллийг өгөгдлийн сангийн хэрэглэгчид байнга, минут тутамд шаардаж үздэг бол энэ жагсаалтыг нэг удаа үүсгээд temporary table – д хадгалаад, дараа нь дахин дуудахад бодолт хийлгүйгээр temporary table – ийг шууд үзүүлдэг байхаар програмчил. Хэрэв ямар нэгэн захиалга хийгдвэл дахин бодолтыг хийж temporary table – д хадгалдаг байхаар хий.
- Зарагдсан барааны мэдээллийг хүснэгтэд нэг удаа хадгалж. Дараа нь дуудахад бодолт хийхгүйгээр шууд үзүүлэх. Хэрэв захиалга хийгдвэл дахин бодолт хийхээр програмчлах.

3. ОНОЛЫН СУДАЛГАА

3.1 Function

SQL функцууд нь параметрүүдийг хүлээн авах, тооцоолол хийх, нэг утгыг буцаах, кодыг дахин ашиглах боломжтой болгодог. Үүний скаляр болон хүснэгтийн функц гэж хоёр төрөлд

хувааж болно. Скаляр функцууд нь бүхэл тоо, мөр, огноо гэх мэт нэг утгыг буцаана. Тэдгээрийг ихэвчлэн тооцоолол, мөрийг удирдах, огнооны үйлдлүүдэд ашигладаг. Хүснэгтийн функц нь хүснэгт хэлбэрээр үр дүнг буцаадаг. Тэд параметрүүдийг хүлээн авч, динамик үр дүнг үүсгэхийн тулд нарийн төвөгтэй байдаг.

Syntax:

```
- Transact-SQL Scalar Function Syntax
CREATE [ OR ALTER ] FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ][ type_schema_name. ] parameter_data_type [ NULL ]
 [ = default ] [ READONLY ] }
    [ ,...n ]
  ]
)
RETURNS return_data_type
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
    BEGIN
        function_body
        RETURN scalar_expression
    END
[ ; ]
```

- **function_name** : Энэ нь заасан өгөгдлийн санд ‘function_name’ гэсэн скаляр функцийг үүсгэнэ. OR ALTER нь функц байгаа бол өөрчлөх боломжийг олгодог.
- **([{ @parameter_name [AS][type_schema_name.] parameter_data_type [NULL] [= default] [READONLY] } :** Энэ нь функцийг оролтын параметрүүдийг тодорхойлдог. Параметр бүрийг **‘@parameter_name’**, өгөгдлийн төрөл **‘parameter_data_type’** болон **NULL**, анхдагч утга буюу **READONLY** зэрэг шинж чанаруудыг зааж өгдөг.
- **RETURNS return_data_type :** Энэ нь функцийг гүйцэтгэсний дараа буцаах өгөгдлийн төрлийг заана.
- **[WITH <function_option> [,...n]] :** Энэ нь шифрлэлт, холболт зэрэг функцэд зориулсан төрөл бүрийн сонголтуудыг тодорхойлно. Жишээ нь хувьсагч зарлах гэх мэт
- **[AS] :** Энэ нь функцийн бие эхэлж буйг заана.
- **BEGIN :** Функцийн процедурын логикийн эхлэлийг тэмдэглэнэ.
- **function_body :** Функцийн гүйцэтгэх логик буюу тооцооллыг агуулна. Функцийн үйлдлийг тодоройлсон кодыг бичнэ.
- **RETURN scalar_expression :** Функцийн буцаах утгыг заана. ‘**scalar_expression’** нь хувьсагч, багана эсвэл скаляр функц байж болно.
- **END :** Функцийн процедурын логикийн төгсгөлийг тэмдэглэнэ.

3.2 Stored Procedure

Stored procedure гэдэг нь энгийн бөгөөд нийлмэжл шинж чанартай, олон дахин ашиглагдах SQL командуудыг нэгтгэсэн өгөгдлийн сангийн объект юм. Хэрэв танд дахин дахин бичдэг query байгаа бол түүнийг stored procedure болгон хадгалаад дуудаж ажиллуулах боломжтой. Stored procedure – д параметрүүдийг дамжуулж болох бөгөөд ингэснээр stored procedure нь дамжуулсан параметрийн утгууд дээр ажиллах боломжтой болно.

Syntax:

```
CREATE PROCEDURE procedure_name
(
```

```
  @parameter1 data_type [DEFAULT value],  -- параметрийн утга, төрөл, default утга өгж болно
  @parameter2 data_type,
  ...
)
AS
BEGIN
  -- процедурын их бие буюу хийх функц
  SELECT ...
  FROM ...
  WHERE ...
  UPDATE ...
  DELETE ...
  DECLARE ...   - variable зарлах боломжтой
END;
```

3.3 Trigger

SQL – д триггер нь датабааз дахь тодорхой хүснэгт дээрх үйл явдлыг автоматаар ажилладаг хадгалагдсан процедур юм.

Syntax:

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF} {INSERT | UPDATE | DELETE} ON table_name
[FOR EACH ROW]
[WHEN (condition)]
BEGIN
    -- Trigger (SQL statements)
END;
```

CREATE TRIGGER нь шинэ триггер үүсгэх эсвэл байгаа нэгийг солиход ашиглана.
BEFORE, AFTER нь хэзээ асах ёстойх заана.
INSERT, UPDATE, DELETE нь триггерийн үйлдлийн төрлийг заана.
WHEN нь нөхцөл

4. ХЭРЭГЖҮҮЛЭЛТ

4.1 Хэрэглэгчийн функц ашиглан Тайлан гаргах. Уг дасгалыг 2 аргаар гүйцэтгэнэ.

1. Аймгийн дугаарыг өгөхөөр санд байгаа бүх насны хүмүүсээс тухайн аймгаас хэдэн хүн байгаа тоог гаргадаг функц бичээд бүх аймгуудын хувьд JOIN хийж тайланг гаргаж хугацааг тэмдэглэж авах.

```
-- Функц зарлах, гараас RegionID авах
CREATE FUNCTION PopulationCountRegion(@RegionID INT)
RETURNS TABLE
AS
RETURN
(
    -- RegionID таарч байгаа хэрэглэгчдийг тоолох
    SELECT COUNT(*) AS PopulationCount
    FROM Users
    WHERE RegionID = @RegionID
);
```

```
DECLARE @StartTime2 DATETIME, @EndTime2 DATETIME, @Method2ExecutionTime INT;
SET @StartTime2 = GETDATE();

-- Функц дуудах, Regions table – аас ID авах
SELECT
    r.Region_name AS RegionName,
    (
        SELECT PopulationCount
        FROM dbo.PopulationCountRegion(r.ID
    ) AS PopulationCount
FROM
    Regions r;

SET @EndTime2 = GETDATE();
SET @Method2ExecutionTime = DATEDIFF(MILLISECOND, @StartTime2, @EndTime2);
```

Үр дүн:

	RegionName	PopulationCount
1	Ulaanbaatar	8
2	Darhan	8
3	Erdenet	4

	Method2ExecutionTime
1	0

2. Аймгийн дугаар, нас гэсэн 2 аргумент аваад тоог нь буцаадаг функц бичээд нүд бүрийн хувьд бодуулах замаар тайлан үүсгэх. Хугацааг тэмдэглэж аваад эхний аргатай харьцуулж үзэх

```
   -- Функц зарлах, гараас Age, RegionID авах
CREATE FUNCTION AgeReport(@Age INT, @RegionID INT)
RETURNS TABLE
AS
RETURN
(
       -- RegionID дахь хэрэглэгчийг тоолох
    SELECT COUNT(*) AS PopulationCount
    FROM Users
    WHERE RegionID = @RegionID AND DATEDIFF(YEAR, DOB, GETDATE()) = @Age
);


DECLARE @StartTime1 DATETIME, @EndTime1 DATETIME, @Method1ExecutionTime INT;
SET @StartTime1 = GETDATE();

-- Функц дуудах, Users table – аас DOB авч Age тооцоолон, параметрээр Age, RegionID
дамжуулах
SELECT
    Age,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 1)) AS Ulaanbaatar,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 2)) AS Darhan,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 3)) AS Erdenet
FROM
```

```
    (SELECT DISTINCT DATEDIFF(YEAR, DOB, GETDATE()) AS Age FROM Users) AS Age;

SET @EndTime1 = GETDATE();
SET @Method1ExecutionTime = DATEDIFF(MILLISECOND, @StartTime1, @EndTime1);

SELECT @Method1ExecutionTime AS Method1ExecutionTime;
```

Үр дүн:

	Age	Ulaanbaatar	Darhan	Erdenet
1	22	2	2	0
2	23	4	2	2
3	24	2	4	2

	Method1ExecutionTime
1	0

4.2 Complex query(function, stored procedure ашигла)

1. Хамгийн их орлого оруулсан үйлчлүүлэгчдийг нийт орлогын 3 хувьтай тэнцэх мөнгийг олгохоор болжээ. Энэ жагсаалтыг үйлчлүүлэгчийн нэр, орлогын нийт мөнгөн дүн, 3 хувьтай тэнцэх мөнгөн дүн зэргээр хэвлэж гаргах complex query бич.

```
-- CalculatePaymentAmount function үүсгэх аргументээр quantity * unitprice буюу нийт дүнг авах
CREATE FUNCTION CalculatePaymentAmount(@TotalSpent DECIMAL(10, 2))
-- буцаах төрөл
RETURNS DECIMAL(10, 2)
AS
BEGIN
    -- DECIMAL төрлийн хувьсагч зарлах
    DECLARE @Payment DECIMAL(10, 2);
    -- 3% дүн = Нийт дүн * 0.03
    SET @Payment = @TotalSpent * 0.03;
    -- 3% дүн буцаах
    RETURN @Payment;
END;

-- CalculateCustomerPayments stored procedure үүсгэх
CREATE PROCEDURE CalculateCustomerPayments
AS
BEGIN
    SELECT
        c.CompanyName,
        TotalSpent = SUM(od.Quantity * od.UnitPrice), -- нийт зарцуулсан дүн
        -- 3% дүнг CalculatePaymentAmount функцыг дуудаж тооцоолоод Payment - д хадгалах
        Payment = dbo.CalculatePaymentAmount(SUM(od.Quantity * od.UnitPrice))
    FROM
        Customers AS c
    INNER JOIN
        Orders AS o ON c.CustomerID = o.CustomerID
    INNER JOIN
        [Order Details] AS od ON o.OrderID = od.OrderID
    GROUP BY  -- бүлэглэх
        c.CompanyName
    ORDER BY  -- буурахаар эрэмбэлэх
        TotalSpent DESC;
END;
```

```
EXEC CalculateCustomerPayments;
```

Үр дүн:

```
EXEC CalculateCustomerPayments;
```

89 %

Results | Messages

	CompanyName	TotalSpent	Payment
1	Save-a-lot Markets	114987.39	3449.62
2	QUICK-Stop	114834.59	3445.04
3	Ernst Handel	112676.68	3380.30
4	Hungry Owl All-Night Grocers	57014.99	1710.45
5	Rattlesnake Canyon Grocery	52263.90	1567.92

2. Зарагдсан барааны мэдээллийг тоо ширхэг, мөнгөн дүн зэрэг өгөгдлийн хамт жагсааж temporary table – д хадгал. Энэ мэдээллийг өгөгдлийн сангийн хэрэглэгчид байнга, минут тутамд шаардаж үздэг бол энэ жагсаалтыг нэг удаа үүсгээд temporary table – д хадгалаад, дараа нь дахин дуудахад бодолт хийлгүйгээр temporary table – ийг шууд үзүүлдэг байхаар программчил. Хэрэв ямар нэгэн захиалга хийгдвэл дахин бодолтыг хийж temporary table – д хадгалдаг байхаар хий.

```
-- Зарагдсан барааны мэдээллийг хадгалах SoldGoods хүснэгт үүсгэх
CREATE TABLE SoldGoods (
    ProductID INT,                          -- Бүтээгдэхүүний ID
    ProductName VARCHAR(100),       -- Бүтээгдэхүүний нэр
    Quantity INT,                           -- Зарагдсан тоо хэмжээ
    Amount MONEY                            -- Нийт дүн
);


-- SoldGoods хүснэгтийг хамгийн сүүлийн мэдээллээр дүүргэх Stored Procedure
CREATE PROCEDURE PopulateSoldGoodsTable
AS
BEGIN
       -- SoldGoods хүснэгтэд байгаа өгөгдлийг арилгах
    TRUNCATE TABLE SoldGoods;
       -- Order Details болон Products хүснэгтийг нэгтгэж SoldGoods хүснэгтэд оруулах
    INSERT INTO SoldGoods (ProductID, ProductName, Quantity, Amount)
    SELECT
        od.ProductID,
        p.ProductName,
        od.Quantity,
              -- Бүтээгдэхүүний нийт үнийг бодох функц дуудах
        dbo.CalculateTotalAmountForProduct(od.ProductID) AS Amount
    FROM
        [Order Details] od
    INNER JOIN
        Products p ON od.ProductID = p.ProductID;
END;

-- Өгөгдсөн productID нийт дүнг тооцоолох функц
CREATE FUNCTION CalculateTotalAmountForProduct (@ProductID INT)
```

```
RETURNS DECIMAL(10, 2)
AS
BEGIN
       -- Нийт дүнг хадгалах хувьсагч зарлах
    DECLARE @TotalAmount DECIMAL(10, 2);
    -- Хувьсагчид тоо хэмжээ болон нэгж үнийг нэгтгэн нийт үнийг олох
    SELECT @TotalAmount = SUM(Quantity * UnitPrice)
    FROM [Order Details]
    WHERE ProductID = @ProductID;
    -- ProductID олдоогүй бол 0 буцаах
    RETURN ISNULL(@TotalAmount, 0);
END;

-- SoldGoods хүснэгтийг анх дүүргэхийн тулд Stored Procedure дуудах
EXEC PopulateSoldGoodsTable

-- Order Details хүснэгтийг өөрчлөхөд SoldGoods хүснэгтийг шинэчлэх триггер
CREATE TRIGGER UpdateSoldGoodsTrigger
ON [Order Details]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    EXEC PopulateSoldGoodsTable; -- SoldGoods шинэчлэхийн тулд Stored Procedure дуудах
END;
```

Үр дүн:

```
EXEC dbo.PopulateSoldGoodsTable
SELECT * FROM SoldGoods;
```

89 %

Results | Messages

	ProductID	ProductName	Quantity	Amount
2112	20	Sir Rodney's Marmalade	1	23635.80
2113	23	Tunnbröd	2	4840.20
2114	32	Mascarpone Fabioli	1	9171.20
2115	39	Chartreuse verte	2	13150.80
2116	41	Jack's New England Clam Chowder	3	9098.10
2117	42	Singaporean Hokkien Fried Mee	10	18.00
2118	46	Spegesild	3	6144.00
2119	52	Filo Mix	2	3383.80
2120	55	Pâté chinois	2	19512.00
2121	60	Camembert Pierrot	2	50286.00
2122	64	Wimmers gute Semmelknödel	2	23009.00
2123	66	Louisiana Hot Spiced Okra	1	3519.00
2124	73	Röd Kaviar	2	4200.00
2125	75	Rhönbräu Klosterbier	4	8650.55
2126	77	Original Frankfurter grüne Soße	2	9685.00

Q | TURBOLD (16.0 RTM) | TURBOLD\turbo (54) | northwind | 00:00:00 | 2,126 rows

Insert хийх үед:

```
INSERT INTO [Order Details]
VALUES (11076, 41, 1.80, 10, 0)
```

89 %

Messages

```
(2127 rows affected)

(1 row affected)

Completion time: 2024-04-24T23:02:53.7107847+08:00
```

```
INSERT INTO [Order Details]
VALUES (11076, 41, 1.80, 10, 0)
```

89 %

Results | Messages

	ProductID	ProductName	Quantity	Amount
2113	20	Sir Rodney's Marmalade	1	23635.80
2114	23	Tunnbröd	2	4840.20
2115	32	Mascarpone Fabioli	1	9171.20
2116	39	Chartreuse verte	2	13150.80
2117	41	Jack's New England Clam Chowder	3	9116.10
2118	42	Singaporean Hokkien Fried Mee	10	18.00
2119	46	Spegesild	3	6144.00
2120	52	Filo Mix	2	3383.80
2121	55	Pâté chinois	2	19512.00
2122	60	Camembert Pierrot	2	50286.00
2123	64	Wimmers gute Semmelknödel	2	23009.00
2124	66	Louisiana Hot Spiced Okra	1	3519.00
2125	73	Röd Kaviar	2	4200.00
2126	75	Rhönbräu Klosterbier	4	8650.55
2127	77	Original Frankfurter grüne Soße	2	9685.00

Q | TURBOLD (16.0 RTM) | TURBOLD\turbo (54) | northwind | 00:00:00 | 2,127 rows

Delete хийх үед:

```
DELETE FROM [Order Details]
WHERE ProductID = 41 AND orderID = 11076
```

%

Messages

```
(2126 rows affected)

(1 row affected)

Completion time: 2024-04-24T23:05:14.9665083+08:00
```

5. ДҮГНЭЛТ

SQL Complex Query нь өгөгдлийн санд логикыг багтаах, тооцоолол хийх хэрэгтэй query юм. Эдгээр нь кодыг оновчтой болгох, засвар үйлчилгээ хийх, гүйцэтгэлийг сайжруулах арга замыг бий болгодог. Энэхүү тайланд дурьдсан функц болон процедурыг хэрэгжүүлснээр тодорхой даалгаврыг автоматжуулж нарийн төвөгтэй үйлдлүүдийг хялбарчилна.

6. Хавсралт

6.1.

```
CREATE FUNCTION PopulationCountRegion(@RegionID INT)
RETURNS TABLE
AS
RETURN
(
    SELECT COUNT(*) AS PopulationCount
    FROM Users
    WHERE RegionID = @RegionID
);

DECLARE @StartTime2 DATETIME, @EndTime2 DATETIME, @Method2ExecutionTime INT;
SET @StartTime2 = GETDATE();

SELECT
    r.Region_name AS RegionName,
    (
        SELECT SUM(PopulationCount)
        FROM dbo.PopulationCountRegion(r.ID)
    ) AS PopulationCount
FROM
    Regions r;

SET @EndTime2 = GETDATE();
SET @Method2ExecutionTime = DATEDIFF(MILLISECOND, @StartTime2, @EndTime2);
```

6.2.

```
CREATE FUNCTION AgeReport(@Age INT, @RegionID INT)
RETURNS TABLE
AS
RETURN
(
    SELECT COUNT(*) AS PopulationCount
    FROM Users
    WHERE RegionID = @RegionID AND DATEDIFF(YEAR, DOB, GETDATE()) = @Age
);

DECLARE @StartTime1 DATETIME, @EndTime1 DATETIME, @Method1ExecutionTime INT;
SET @StartTime1 = GETDATE();

SELECT
    Age,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 1)) AS Ulaanbaatar,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 2)) AS Darhan,
    (SELECT PopulationCount FROM dbo.AgeReport(Age, 3)) AS Erdenet
FROM
    (SELECT DISTINCT DATEDIFF(YEAR, DOB, GETDATE()) AS Age FROM Users) AS Age;

DECLARE @StartTime1 DATETIME, @EndTime1 DATETIME, @Method1ExecutionTime INT;
SET @StartTime1 = GETDATE();

SELECT @Method1ExecutionTime AS Method1ExecutionTime;
```

6.3.

```
CREATE FUNCTION CalculatePaymentAmount(@TotalSpent DECIMAL(10, 2))
RETURNS DECIMAL(10, 2)
```

```
AS
BEGIN
    DECLARE @Payment DECIMAL(10, 2);
    SET @Payment = @TotalSpent * 0.03;
    RETURN @Payment;
END;

CREATE PROCEDURE CalculateCustomerPayments
AS
BEGIN
    SELECT
        c.CompanyName,
        TotalSpent = SUM(od.Quantity * od.UnitPrice),
        Payment = dbo.CalculatePaymentAmount(SUM(od.Quantity * od.UnitPrice))
    FROM
        Customers AS c
    INNER JOIN
        Orders AS o ON c.CustomerID = o.CustomerID
    INNER JOIN
        [Order Details] AS od ON o.OrderID = od.OrderID
    GROUP BY
        c.CompanyName
    ORDER BY
        TotalSpent DESC;
END;

EXEC CalculateCustomerPayments;
```

6.4.

```
CREATE TABLE SoldGoods (
    ProductID INT,
    ProductName VARCHAR(100),
    Quantity INT,
    Amount MONEY
);

CREATE TRIGGER UpdateSoldGoodsTrigger
ON [Order Details]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    EXEC PopulateSoldGoodsTable;
END;

CREATE PROCEDURE PopulateSoldGoodsTable
AS
BEGIN
    TRUNCATE TABLE SoldGoods;
    INSERT INTO SoldGoods (ProductID, ProductName, Quantity, Amount)
    SELECT
        od.ProductID,
        p.ProductName,
        od.Quantity,
        dbo.CalculateTotalAmountForProduct(od.ProductID) AS Amount
    FROM
        [Order Details] od
    INNER JOIN
        Products p ON od.ProductID = p.ProductID;
END;
```

```sql
CREATE FUNCTION CalculateTotalAmountForProduct (@ProductID INT)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @TotalAmount DECIMAL(10, 2);
    SELECT @TotalAmount = SUM(Quantity * UnitPrice)
    FROM [Order Details]
    WHERE ProductID = @ProductID;
    RETURN ISNULL(@TotalAmount, 0);
END;
```