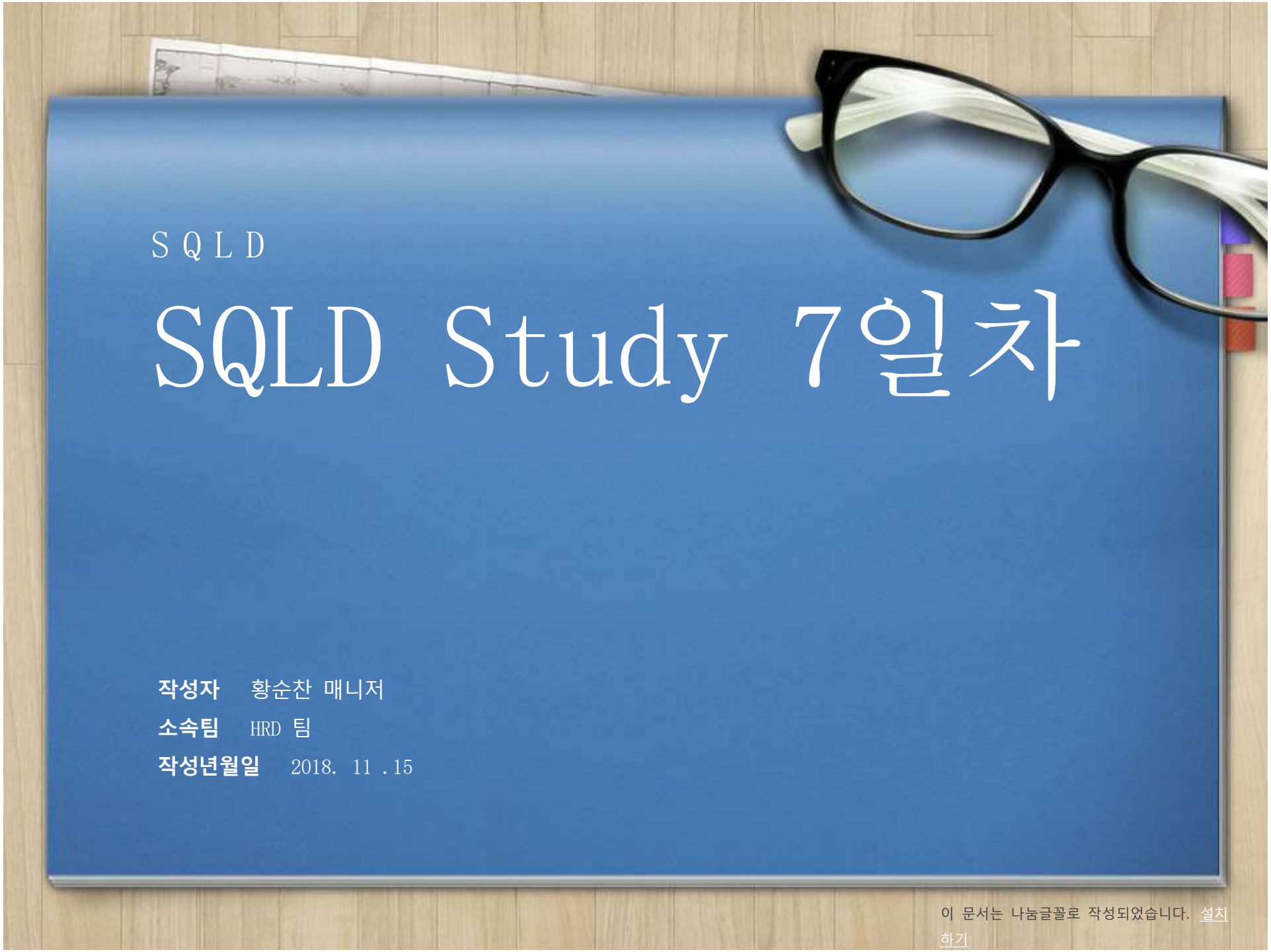


S Q L D

SQLD Study 7일차

작성자 황순찬 매니저
소속팀 HRD 팀
작성년월일 2018. 11 .15

0. 금일의 키포인트

- 포인트

서브쿼리의 정의

뷰의 대한 정의

ROLLUP 함수

0. 금일의 키포인트

- 포인트

CUBE 함수

그룹내 순위 함수

프로시저와 트리거의 차이

목차

1. SQL 기본

- 서브쿼리
- 그룹 함수(GROUP FUNCTION)
- 윈도우 함수(WINDOW FUNCTION)
- DCL(DATA CONTROL LANGUAGE)
- 절차형 SQL

- 서브쿼리

정의

- 하나의 SQL문 안에 포함되어 있는 또 다른 SQL문
- 알려지지 않은 기준을 이용한 검색을 위해 사용
- 메인 쿼리와 서브쿼리

메인쿼리-->

```
SELECT...
FROM...
WHERE...
        (SELECT...
        FROM...
        WHERE...)
```

<--서브쿼리

-> 서브쿼리는 메인 쿼리의 컬럼을 사용할 수 있지만 메인 쿼리는 서브쿼리 컬럼을 사용할 수 없다.

-> 서브 쿼리 컬럼을 사용한다면 조인 방식을 변환하거나 함수, 스칼라등을 사용

1. SQL기본

- 서브쿼리

정의

- 조인은 집합간의 곱의 관계이다. 1:1 관계라면 1레벨의 집합 생성. 1:M의 관계라면 M레벨의 집합, M:N의 관계라면 M-N 레벨 집합

- 서브 쿼리 사용시 주의사항
  -> 서브 쿼리를 괄호로 감싸서 사용.

  -> 단일행 또는 복수행 비교 연산자와 함께 사용가능하며,
  단일행 비교연산자는 서브쿼리의 결과가 반드시 1건 이하여야 하고
  복수행 비교연산자는 서브쿼리의 결과 건수와 상관이 없다.

  -> 서브쿼리에서는 ORDER BY를 사용하지 못하며, 쿼리의 단 한개만 사용가능.

- 서브쿼리의 사용이 가능 한 곳.
  -> SELECT절
  -> FROM절
  -> WHERE절
  -> HAVING절
  -> ORDER BY절
  -> INSERT 문의 VALUES 절
  -> UPDATE 문의 SET절

- 서브쿼리

정의

- 서브 쿼리의 종류는 동작하는 방식이나 반환되는 데이터의 형태에 따라 분류

- 동작하는 방식에 따른 서브쿼리

서브쿼리 종류	설명
Un-Correlated(비연관) 서브쿼리	서브쿼리가 메인쿼리 컬럼을 가지고 있지 않는 형태의 서브쿼리이다. 메인 쿼리(서브쿼리가 실행된 결과)을 제공하기 위한 목적으로 주로 사용한다.
Correlated(연관) 서브쿼리	서브쿼리가 메인쿼리 컬럼을 가지고 있는 형태의 서브쿼리이다. 일반적으로 메인쿼리가 먼저 수행되어 읽혀진 데이터를 서브쿼리에서 조건이 맞는지 확인하고자 할 때 주로 사용된다.

-> 서브쿼리는 메인쿼리 안에 포함되기 때문에 항상 메인쿼리에서 읽혀진 데이터에 대해 서브쿼리에서 해당 조건이 만족하는지 확인하는 방식. 실제 실행 순서는 달라질 수 있다.

- 서브쿼리

정의

- 반환되는 데이터 형태에 따른 서브쿼리 분류

서브쿼리 종류	설명
Single Row 서브쿼리 (단일 행 서브쿼리)	서브쿼리의 실행 결과가 항상 1건 이하인 서브쿼리를 의미한다. 단일 행 서브쿼리는 단일 행 비교 연산자와 함께 사용된다. 단일 행 비교 연산자에는 = , < , <= , > , >= , <>이 있다.
Multi Row 서브쿼리 (다중 행 서브쿼리)	서브쿼리의 실행 결과가 여러 건인 서브쿼리를 의미한다. 다중 행 서브쿼리는 다중 행 비교 연산자와 함께 사용된다. 다중 행 비교 연산자에는 IN, ALL, ANY , SOME, EXISTS가 있다.
Multi Column 서브쿼리 (다중 컬럼 서브쿼리)	서브쿼리의 실행 결과로 여러 컬럼을 반환한다. 메인쿼리의 조건절에 여러 컬럼을 동시에 비교할 수 있다. 서브쿼리와 메인쿼리에서 비교하고자 하는 컬럼 개수와 컬럼의 위치가 동일해야 한다.

- 서브쿼리

단일행 서브쿼리

- 서브쿼리가 단일 행 비교연산자와 함께 사용할 때는 서브쿼리의 결과 건수가 반드시 1건 이하여야 한다.
  -> 서브쿼리의 결과 건수가 2건 이상 반환되면 SQL문은 실행시간 오류가 발생

- 서브쿼리의 예제

```
SELECT  PLAYER_NAME 선수명, POSITION 포지션 , BACK_NO 백넘버
FROM            PLAYER
WHERE   TEAM_ID = (SELECT     TEAM_ID
                           FROM       PLAYER
                           WHERE      PLAYER_NAME =  '정남일' )
ORDER BY PLAYER_NAME;
```

- 테이블 전체에 하나의 그룹함수를 적용할 때는 그 결과값이 1건이 생성되기 때문에 단일행 서브쿼리로써 사용가능.

- 서브쿼리

다중행 서브쿼리

- 서브쿼리의 결과가 2건 이상 반환될 수 있다면 반드시 다중행 비교 연산자와 함께 사용.

- 다중행 비교 연산자 종류

다중 행 연산자	설명
IN (서브쿼리)	서브쿼리의 결과에 존재하는 임의의 값과 동일한 조건을 의미한다.(MUTIPLE OR 조건)
비교연산자 ALL (서브쿼리)	서브쿼리의 결과에 존재하는 모든 값을 만족하는 조건을 의미한다. 비교 연산자로 ">"를 사용했다면 메인쿼리는 서브쿼리의 모든 결과 값을 만족해야 하므로, 서브쿼리 결과의 최대값보다 큰 모든 건이 조건을 만족한다.
비교연산자 ANY (서브쿼리)	서브쿼리의 결과에 존재하는 어느 하나의 값이라도 만족하는 조건을 의미한다. 비교연산자로 ">"를 사용했다면 메인쿼리는 서브쿼리의 값들 중 어떤 값이라도 만족하면 되므로, 서브쿼리의 결과의 최소값보다 큰 모든 건이 조건을 만족한다.(SOME은 ANY와 동일함)
EXISTS (서브쿼리)	서브쿼리의 결과를 만족하는 값이 존재하는 지 여부를 확인하는 조건을 의미한다. 조건을 만족하는 건이 여러 건이더라도 1건만 찾으면 더 이상 검색하지 않는다.

- 서브 쿼리의 실행 결과가 2건 이상 나오는 모든 경우에 다중 행 비교연산을 사용.

- 서브쿼리

다중 컬럼 서브 쿼리

- 서브쿼리의 결과로 여러 개의 컬럼이 반환되어 메인 쿼리의 조건과 동시에 비교되는 것을 의미.

- SQL_SERVER에서는 적용 되지 않음.

```
SELECT TEAM_ID 팀코드, PLAYER_NAME 선수명, POSITION 포지션, HEIGHT 키
FROM          PLAYER
WHERE (TEAM_ID,HEIGHT) IN (SELECT TEAM_ID, MIN(HEIGHT)
                           FROM PLAYER
                           GROUP BY TEAM_ID)
ORDER BY TEAM_ID, PLAYER_NAME;
```

- 서브쿼리

연관 서브 쿼리

- 서브쿼리 내에서 메인 쿼리 컬럼이 사용된 서브쿼리

- EXISTS 서브쿼리는 항상 연관서브쿼리로 사용.
  Exists 서브쿼리의 특징은 아무리 조건을 만족하는 건이 여러건이더라도 조건을 만족하는 1건만 찾으면 추가 검색을 하지 않는다.

- 서브쿼리

그 밖에 위치에서 사용하는 서브쿼리

1) SELECT 절에 서브쿼리 사용.

- SELECT절에 사용하는 서브쿼리인 스칼라 서브쿼리

- 스칼라 서브쿼리는 한 행, 한 컬럼만을 반환하는 서브쿼리
  -> 컬럼을 쓸 수있는 대부분의 곳에서 사용가능.

- 스칼라 서브 쿼리 또한 단일 행 서브쿼리이기 때문에 결과가 2건 이상 반환되면 SQL문내에서 오류 반환.

2) FROM절에서 서브쿼리 사용하기

- FROM 절에서 사용되는 서브쿼리를 인라인 뷰라고 함.

- SQL문이 실행 될 때만 임시적으로 생성되는 동적인 뷰이기 때문에 데이터베이스에 해당 정보가 저장되지 않는다.

- 일반적인 뷰 = 정적 뷰 , 인라인 뷰는 동적 뷰 라고 한다.

- 인라인 뷰의 컬럼은 SQL문 자유롭게 참조 할 수 있다.

- 인라인 뷰에서는 ORDER BY절을 사용할 수 있으며, 인라인 뷰에 먼저 정렬을 수행하고 정렬된 결과 중에서 일부분을 추출하는 것을 TOP-N이라 함.

- 서브쿼리

그 밖에 위치에서 사용하는 서브쿼리

3) HAVING 절에서 서브쿼리 사용하기

- HAVING절은 그룹 함수와 함께 사용될 때 그룹핑된 경과에 대해 부가적인 조건을 주기 위해 사용

4) UPDATE 문의 SET 절에서 사용

- 서브쿼리를 사용한 변경 작업을 할 때 서브쿼리의 결과가 NULL을 반환할 경우 해당 컬럼의 결과가 NULL이 될 수 도 있기 때문에 주의

5) INSERT 문의 VALUES절에서 사용하기

- 서브쿼리

뷰(VIEW)

- 테이블은 실제로 데이터를 가지고 있는 반면, 뷰는 실제 데이터를 가지고 있지 않다.
- 뷰는 단지 뷰 정의(view definition) 만을 가지고 있다.
- 질의에서 뷰가 사용되면 뷰 정의를 참조해서 DBMS 내부적으로 질의를 재작성(REWRITE)하여 질의 수행
- 실제 데이터를 가지고 있지 않지만 테이블이 수행하는 역할을 수행하기 때문에 가상 테이블(virtual table)이라고도 한다.
- 뷰 사용의 장점.

뷰의 장점	설명
독립성	테이블 구조가 변경되어도 뷰를 사용하는 응용 프로그램은 변경하지 않아도 된다.
편리성	복잡한 질의를 뷰로 생성함으로써 관련 질의를 단순하게 작성할 수 있다. 또한 해당 형태와 SQL문을 자주 사용할 때 뷰를 이요하면 편리하게 사용할 수 있다.
보안성	직원의 급여정보와 같이 숨기고 싶은 정보가 존재한다면, 뷰를 생성할 때 해당 컬럼을 빼고 생성함으로써 사용자에게 정보를 감출 수 있다.

- 서브쿼리

뷰(VIEW)

- 뷰는 Create View를 통해 생성가능
  -> 뷰를 포함하는 뷰를 잘못 생성하는 경우 성능의 문제가 발생할 수 있으므로 수행원리를 잘 이해해야 한다.

- 뷰에 대한 예제

```
CREATE VIEW V_PLAYER_TEAM AS
SELECT P.PLAYER_NAME, P.POSITION, P.BACK_NO, P.TEAM_ID, T.TEAM_NAME
FROM PLAYER P, TEAM T
WHERE P.TEAM_ID = T.TEAM_ID;
```

- VIEW 삭제 하기.

```
DROP VIEW V_PLAYER_TEAM;
DROP VIEW V_PLAYER_TEAM_FILTER;
```

- 그룹함수(GROUP FUNCTION)

데이터 분석의 개요

- ANSI/ISO SQL 표준은 데이터 분석을 위해 3가지 함수 정의
  - -> Aggregation Function
  - -> Group Function
  - -> Window Function

- Aggregation Function
  - Group Aggregation Function이라고 부르며, Group Function의 한 부분으로 분류(Count, Sum, Avg, Max등 집계함수 포함)

- Group Function
  - 하나의 SQL로 테이블을 한번만 읽어서 빠르게 리포트 작성 가능
    - -> 소계/합계를 표시하기 위해 grouping 함수와 Case 함수를 이용하여 원하는 포맷 보고서 가능

- 그룹함수(GROUP FUNCTION)

데이터 분석의 개요

- 그룹함수로는 집계함수를 제외하고 소그룹간의 소계를 계산하는 ROLL UP 함수, Group by 항목들간 다차원적인 소계를 계산할 수 있는 CUBE 함수, 특정 항목에 대한 소계를 계산하는 Grouping set 함수
  -> ROLL UP은 GROUP BY의 확장된 형태로 사용하기 쉬우며 병렬로 수행 가능하기 때문에 효과적이며, 계층적 분류를 포함하고 있는 데이터 집계에 적합.

  -> CUBE는 결합 가능한 모든 값에 대하여 다차원적인 집계를 생성. ROLL UP에 비해 다양한 데이터를 얻을 수 있지만 시스템 부하가 있다.

  -> Grouping Sets는 원하는 부분의 소계만 손쉽게 추출 가능.

  -> ROLL UP, CUBE , Grouping sets 결과에 대해 정렬이 필요한 경우 ORDER BY 절에 정렬 컬럼을 명시

- WINDOW FUNCTION
  -> 분석함수나 순위함수로 알려져 있는 윈도우 함수는 데이터 웨어하우스에서 발전한 기능이다.

- 그룹함수(GROUP FUNCTION)

ROLLUP 함수

- ROLL UP에 지정된 Grouping Column의 List는 Subtotal을 생성하기 위해 사용되며, Grouping Columns의 수는 N이라고 했을때 N+1 Level의 Subtotal이 생성된다.
  -> ROLL UP의 인수는 계층 구조 이므로 인수순서가 바뀌면 수행 결과로 바뀌게 되므로 인수의 순서에도 주의

1) 일반적인 Group by절 사용
   - 정렬이 필요한 경우 order by절에 명시적으로 정렬 컬럼이 표시 되어야 한다.

```
SELECT DNAME, JOB,
        COUNT(*)  "total Empl"
        SUM(SAL)  "Total Sal"
FROM EMP , DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY DNAME, JOB;
```

- 그룹함수(GROUP FUNCTION)

ROLLUP 함수

1-1) GROUP BY + ORDER BY 사용

```
SELECT DNAME, JOB,
        COUNT(*)  "total Empl"
        SUM(SAL)  "Total Sal"
FROM EMP , DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY DNAME, JOB
ORDER BY DNAME, JOB;
```

2) ROLLUP 함수 추가

- ROLL UP의 경우 계층간 집계에 대해서 LEVEL 별 순서를 정렬하지만, 계층 내 GROUP BY수행시 생성되는 표준집계에는 별도의 정렬을 지원하지 않는다.

```
SELECT DNAME, JOB,
        COUNT(*)  "total Empl"
        SUM(SAL)  "Total Sal"
FROM EMP , DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY ROLLUP (DNAME, JOB);
```

- 그룹함수(GROUP FUNCTION)

ROLLUP 함수

2-2) ROLLUP 함수 + ORDER BY절 사용.

```
SELECT DNAME, JOB,
        COUNT(*)  "total Empl"
        SUM(SAL)  "Total Sal"
FROM EMP , DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY ROLLUP (DNAME, JOB);
ORDER BY DNAME, JOB;
```

3) GROUPING 함수 사용

- ROLL UP, CUBE,GROUPING SETS 등 새로운 그룹함수를 지원하기 위해 GROUPING 함수가 추가
  - -> ROLL UP이나 CUBE에 의한 소계가 계산된 결과에는 GROUPING(EXPR) = 1이 표시
  - -> 그 외의 결과에는 GROUPING(EXPR) = 0이 표시
- GROUPING 함수와 CASE/DECODE를 이용해, 소계를 나타내는 필드에 원하는 문자열을 지정할 수 있음.

- 그룹함수(GROUP FUNCTION)

ROLLUP 함수

4) GROUPING 함수 + CASE 사용.

4-2) ROLL UP 함수 일부 사용.

```
SELECT
 CASE GROUPING(DNAME) WHEN 1 THEN  'All Departments'  ELSE DNAME END AS DNAME,
 CASE GROUPING(JOB) WHEN 1 THEN  'ALL jobs'   ELSE JOB END AS JOB,
COUNT(*)        "Total Empl",
SUM(SAL)        "Total Sal"
FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY DNAME, ROLLUP(JOB);
```

- 그룹함수(GROUP FUNCTION)

ROLLUP 함수

4) GROUPING 함수 + CASE 사용.

4-3) ROLLUP 함수 결합 컬럼 사용

```
SELECT DNAME, JOB , MGR, SUM(SAL)     “Total Sal”
FROM    EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY ROLLUP (DNAME, (JOB, MGR));
```

- 그룹함수(GROUP FUNCTION)

CUBE 함수

- ROLLUP에서 단지 가능한 Subtotal 만을 생성했지만, CUBE는 결합 가능한 모든 값에 대해 다차원 집계사용.
- CUBE를 사용할 경우에 내부적으로 Grouping Columns의 순서를 바꾸어서 또 한번의 Query를 추가 수행.
- Grouping Columns가 가질수 있는 모든 경우에 대해 CUBE를 사용하는 것이 바람직하나 ROLLUP에 비해 시스템에 많은 부담을 주므로 사용에 주의
- CUBE함수의 경우 ROLLUP과 달리 평등한 관계이므로 인수의 순서가 바뀌는 경우 행 간 정렬 순서로 바뀌지만 결과 그대로이다.

- 그룹함수(GROUP FUNCTION)

CUBE 함수

5) CUBE 함수 이용

- CUBE는 GROUPING COLUMNS이 가질 수 있는 모든 경우의 수에 대하여 SUBTOTAL을 생성하는데 GROUPING COLUMNS 수가 N이라고 가정하면 2의 N승 LEVEL의 SUBTOTAL을 생성.

5-2) UNION ALL사용 SQL

- UNION ALL은 SET OPERATION내용으로, 여러 SQL문장을 연결하는 역할을 할 수 있다.
  CUBE SQL과 결과 데이터는 같으나 행들의 정렬은 다르다.

- CUBE함수를 사용하기 전에는 테이블을 여러번 엑세스를 했지만 CUBE사용 SQL에서는 한번으로 줄일 수 있다.

-> 수행 속도 및 자원 사용율을 줄일 수 있으며, SQL문장도 짧아져 가독성도 높였다.
실행 결과 ROLLUP 함수도 똑같은 개선효과를 갖는다.

- 그룹함수(GROUP FUNCTION)

GROUPING SET 함수

- GROUP BY SQL 문장을 여러번 반복하지 않아도 원하는 결과를 얻을 수 있다.

- GROUPING SETS에 표시된 인수들에 대해 개별 집계를 구할 수 있으며, 이 때 표시된 인수들 간에는 계층구조인 ROLLUP과 달리 평등한 관계이므로, 순서가 바뀌어도 결과는 바뀌지 않는다.

- GROUPING SETS함수도 결과에 대한 정렬이 필요한 경우 ORDER BY를 명시적으로 표시한다.

- 그룹함수(GROUP FUNCTION)

GROUPING SET 함수

1) 일반 그룹함수를 이용한 SQL
   - 별도의 ORDER BY 조건을 명시하지 않았기 때문에 컬럼에 대해 정렬되있지 않다.

2) GROUPING SET의 사용 SQL
   - grouping set 함수 사용하여 UNION ALL을 사용한 일반 그룹 함수 SQL과 같은 결과를 얻을 수 있으며, 괄호로 묶은 집합 별로 집계를 구할 수 있다.

3) GROUPING SET을 사용 SQL(순서변경)
   - GROUPING SETS 인수들은 평등한 관계이므로 인수의 순서과 바뀌어도 결과는 같다.

4) 3개의 인수를 이용한 GROUPING SETS 이용

- 윈도우 함수

WINDOW FUNCTION 개요

- 절차형 프로그램을 작성하거나 INLINE VIEW를 이용해 복잡한 SQL문을 작성하는 것을 쉽게 정의하기 위해 만든 함수
  (복잡한 플그램을 하나의 SQL문장으로 쉽게 해결)

- 분석 함수(ANALYTIC)나 순위(RANK)로도 알려진 윈도우 함수는 데이터 웨어하우스에서 발전한 기능

- 기존에 사용하던 집계함수도 있고, 새로이 WINDOW함수 전용으로 만들어진 기능도 있다.
  또한, 다른 함수와는 달리 중첩해서 사용 못하지만 서브쿼리에서는 사용할 수 있다.

- 윈도우 함수

WINDOW FUNCTION 개요

- 윈도우 함수의 종류

1) 그룹내 순위 함수 : RANK, DENSE_RANK, ROW_NUMBER 함수
: ANSI/ISO, ORACLE, SQL SERVER등 지원

2) 그룹내 집계 관련 함수 : SUM,MAX,AVG,COUNT 등
: ANSI/ISO, ORACLE,SQL SERVER등 지원
->SQL SERVER는 집계함수는 뒤에 설명할 OVER절내의 ORDER BY를 사용할 수 없다.

3) 그룹내 행 순서 관련 함수 : FIRST_VALUE,LAST_VALUE,LANG,LEAD
: ORACLE에서 FIRST_VALUE,LAST_VALUE 지원(MAX,MIN)
: lag,LEAD는 DW에서 유용하게 사용되는 기능

4) 그룹내 비율 관련 함수 : CUME_DIST,PERSENT_RANK,NTILE,RATIO_TO_REPORT함수
-> COME_DIST,PERSENT_RANK 함수는 ANSI/ISO와ORACLE지원
-> NTILE 함수는 ANSI 표준에는 없지만 ORACLE과 SQL SERVER 에서 지원

5) 선형 분석을 포함한 통계 분석 함수 : PASS

- 윈도우 함수

WINDOW FUNCTION 개요

- WINDOW FUNCTION SYNTAX

-> WINDOW 함수에는 OVER 문구가 키워드로 필수 포함

```
SELECT WINDOW_FUNCTION(ARGUMENTS) OVER
([PARTITION BY 컬럼] [ORDER BY 절]  [WINDOWING 절])
FROM 테이블 명;
```

-> 설명

1. WINDOW FUNCTION : 기존에 사용하던 함수도 있고, 새롭게 WINDOW 함수용으로 추가된 함수도 있다.

2. ARGUMENT(인수) : 함수에 따라 0 ~ N 개의 인수가 지정.
3. PARTITION BY : 전체 집합을 기준에 의해 소그룹으로 나눌 수 있다.
4. ORDER BY : 어떤 항목에 순위를 지정할 지 ORDER BY 절을 기술
5. WINDOWING : 함수의 대상이 되는 행 기준의 범위를 강력하게 지정
   ROWS는 물리적 결과 행의 수 , RANGE는 논리적인 값에 의해 범위를 나타내며, 둘 중 하나만 선택하여 사용
   -> SQL SERVER에서는 지원하지 않는다.

- 윈도우 함수

그룹내 순위 함수

1) RANK 함수 : RANK 함수는 ORDER BY를 포함한 QUERY문에서 특정항목에 대한 순위를 구하는 함수

-> 특정 범위 내에서 순위를 구할 수 있고, 전체 데이터에 대한 순위도 가능. 동일한 값에 대해서 동일한 순위를 지정.

2) DENSE_RANK 함수 : RANK 함수와 흡사하나, 동일한 순위를 하나의 건수로 취급하는 점이 다름.

3) ROW_NUMBER 함수 : ROW_NUMBER 함수는 RANK와 DENSE_RANK함수가 동일한 값에 대해서는 동일한 순위를 부여하는데 반해 동일한 값이라도 고유한 순위 부여.

-> 동일한 값에 대한 순서까지 관리 하고 싶으면 ROW_NUMBER() OVER(ORDER BY SAL, DESC ENAME) 같이 ORDER BY절을 이용해 추가적인 정렬 기준을 정의

- 윈도우 함수

일반 집계 함수

1) SUM 함수 : 파티션별 윈도우의 함을 구할 수 있다.

```
SELECT SUM(SAL) OVER(PARTITION BY MGR) MGR_SUM
FROM EMP;
```

2) MAX 함수 : 파티션별 윈도우의 최대값을 구할 수 있다.

```
SELECT MAX(SAL) OVER(PARTITION BY MGR) MGR_MAX
FROM EMP;
```

3) MIN 함수 : 파티션별 윈도우의 최소값을 구할 수 있다.

```
SELECT MIN(SAL) OVER(PARTITION BY MGR ORDER BY HIREDATE) MGR_MIN
FROM EMP;
```

4) AVG 함수 : AVG함수와 파티션별 ROWS 윈도우를 이용해 원하는 조건에 맞는 데이터에 대한 통계값을 구할 수 있다.

```
SELECT ROUND(AVG(SAL) OVER(PARTITION BY MGR ORDER BY MGR ORDER BY HIREDATE ROWS
BETWEEN 1 PERCEDING AND 1 FOLLOWING)) AS MGR_AVG
FROM EMP;
```

5) COUNT 함수 : COUNT 함수와 파티션별 ROWS 위도우를 이용해 원하는 조건에 맞는 데이터에 대한 통계값을 구할 수 있다.

- 윈도우 함수

그룹 내 행 순서 함수

1) FIRST_VALUES 함수 : 파티션별 윈도우에서 가장 먼저 나온 값을 구함.
-> SQL SERVER에서 지원하지 않고 MIN 함수를 활용하여 같은 결과를 얻을 수 있다.

: FIRST_VALUE는 다른 함수와 달리 공통 등수를 인정하지 않고 처음 나온 행만 처리

; 공통 등수가 있을 경우에 의도적으로 세부 항목을 정렬하려면 INLINE VIEW를 사용하거나, OVER()내의 ORDER BY절에 컬럼 추가

2) LAST_VALUES 함수 : 파티션별 윈도우에서 가장 나중에 나온 값을 구한다.

: SQL SERVER에서는 지원하지 않으며, MAX함수를 활용하여 같은 결과를 얻음.

: 공통 등수가 있을 때 FIRST_VALUE와 같다.

- 윈도우 함수

그룹 내 행 순서 함수

3) LAG 함수 : 파티션별 윈도우에서 이전 몇번째 행의 값을 가져올 수 있다.
SQL_SERVER에서 지원하지 않는다.

: NVL, INNULL의 기능과 같다.

4) LEAD 함수 ; 파티션별 윈도우에서 이후 몇번재 값을 가져올 수 있다.
참고로 SQL_SERVER에서 지원하지 않는다.

: LEAD 함수는 3개의 ARGUMENTS 까지 사용할 수 있는데, 두번재 인자는 몇번째 후의 행을 가져올지 결정하며 세번재 인자는 NULL이 들어올 때 다른 값으로 바꿔준다.
-> NVL과 ISNULL 기능과 같다.

- 윈도우 함수

그룹 내 비율 함수

1) RATI_TO_REPORT 함수 : 파티션 내 전체 SUM(컬럼)값에 대한 행별 컬럼 값의 백분율을 소수점으로 구할 수 있다.

: 결과값을 > 0 && <=1 의 범위를 가지며, 개별 RATIO의 합을 구하면 1이 된다.
(SQL_SERVER에서는 지원하지 않는다.)

```
SELECT ENAME, SAL, ROUND(RATIO_TO_REPORT(SAL) OVER (0, 2) AS R_R
FROM EMP;
```

2) PERCENT_RANK 함수 : 파티션별 윈도우에서 제일 먼저 나오는 것을 0으로, 제일 늦게 나오는 것을 1로 하여, 값이 아닌 행의 순서별 백분율을 구한다.

: 결과 값은 >=0 && <=1의 범위를 가진다.
(SQL_SERVER는 지원하지 않는다.)

```
SELECT PERCENT_RANK() OVER(PARTITION BY DEPTNO ORDER BY SAL DESC)
FROM EMP;
```

- 윈도우 함수

그룹 내 비율 함수

3) CUME_DIST 함수 : 파티션별 위도우의 전체건수에서 현재 행보다 작거나 같은 건수에 대해 누적 백분율을 구한다.

: 결과값을 >0 && <=1의 범위를 가진다.
(SQL_SERVER는 지원하지 않는다.)

: 다른 WINDOW 함수의 경우 동일 순서면 앞 행의 함수 결과 값을 따르는데, CUME_DIST의 경우는 동일 순서면 뒤 행의 결과 값을 기준으로 한다.

4) NTILE함수 : 파티션별 전체 건수를 ARGUMENT 값으로 N등분한 결과를 구 할 수 있다.

- DCL(DATA CONTROL LANGUAGE)

DCL의 개요

- 유저를 생성하고 권한을 제어 할 수 있다.

유저와 권한

- 대부분의 데이터베이스는 데이터 보호와 보완을 위해 유저와 권한을 관리한다.

- ORACLE에서 제공하는 유저들

유저	역할
SCOTT	ORACLE 테스트용 샘플 유저 DEFALUT 패스워드 : TIGER
SYS	DBA ROLE을 부여받은 유저
SYSTEM	데이터베이스의 모든 시스템 권한을 부여받은 DBA 유저 ORCLE 설치 완료 시에 패스워드 설정.

- DCL(DATA CONTROL LANGUAGE)

유저와 권한

- ORACLE은 유저를 통해 데이터베이스에 접속을 하는 형태
  -> ID와 PW방식으로 인스턴스에 접속을 하고 그에 해당하는 스키마에 오브젝트 생성 등의 권한을 부여

- SQL_SERVER는 인스턴스에 접속하기 위해 로그인이란 것을 생성하며, 인스턴스 내에 존재하는 다수의 데이터베이스에 연결하여 작업하기 위해 유저를 생성한 후 로그인과 유저를 매핑해줘야 한다.

  : 특정 유저는 특정 데이터베이스 내의 특정 스키마에 대해 권한 부여

- DCL(DATA CONTROL LANGUAGE)

유저와 권한

- SQL_SERVER 로그인의 2가지 방법

-> WINDOW 인증 방식으로 WINDOW에 로그인 한 정보를 가지고
SQL_SERVER에 접속하는 방식.

WINDOW 인증은 기본인증 모드이며, SQL_SERVER 인증보다 훨씬 안전

SQL SERVER가 WINDOW에서 제공하는 자격 증명을 신뢰하므로
WINDOW 인증을 사용한 연결을 트러스트된 연결이라 한다.

-> 혼합 모드 방식으로 기본적으로 WINDOW 인증으로도 SQL_SERVER에 접속
가능하며, ORACLE의 인증과 같은 방식으로 사용자 아이디와 비밀번호로
SQL SERVER에 접속하는 방식

SQL 인증을 사용할 때는 강력한 암호(숫자 + 문자 + 특수문자 등) 을 사용.

- DCL(DATA CONTROL LANGUAGE)

유저와 권한

1) 유저 생성과 시스템 권한 부여

- 롤(ROLE)을 이용하여 간편하고 쉽게 권한을 부여
  ->새로운 유저를 생성하려면 CREATE USER를 사용

```
<ORACLE>
GRANT CREATE USER TO SCOTT;

CONN SCOTT/TIGER

CREATE USER PJS IDENTIFIED BY KOREA7;
```

- SQL_SERVER는 유저를 생성하기 전 먼저 로그인을 생성
  로그인을 생성할 수 있는 권한을 가진 로그인은 기본적인 SQL이다.

```
<SQL_SERVER>
CREATE LOGIN PJS WITH PASSWORD = ‘KOREA7’ , DEFAULT_DATABASE = AdventureWorks
```

- DCL(DATA CONTROL LANGUAGE)

유저와 권한

1) 유저 생성과 시스템 권한 부여

- SQL_SERVER에서 유저는 데이터베이스마다 존재하며, 유저를 생성하기 위해서는 유저가 속할 DATABASE로 이동해야 한다.
  -> 유저가 로그인하려면 CREATE SESSION 권한을 부여 받아야 한다.

- 로그인 권한만 부여됫기 때문에 테이블 생성시 테이블 생성 권한도 받아야 한다. 그렇지 않으면 ERROR가 된다.

2) OBJECT에 대한 권한 부여

- 오브젝트 권한은 특정 오브젝트인 테이블, 뷰 등에 대한 SELECT, INSERT, DELETE, UPDATE 작업 명령어를 의미

- 오브젝트 권한과 오브젝트 관계

- DCL(DATA CONTROL LANGUAGE)

유저와 권한

2) OBJECT에 대한 권한 부여

- 오브젝트 권한과 오브젝트 관계(ORACLE 사례)

객체권한	테이블	VIEWS	SEQUENCE	PROCEDURE
ALTER	O		O	
DELETE	O	O		
EXECUTE				O
INDEX	O			
INSERT	O	O		
REFERENCES	O			
SELECT	O	O	O	
UPDATE	O	O		

- DCL(DATA CONTROL LANGUAGE)

유저와 권한

2) OBJECT에 대한 권한 부여

- 오브젝트 권한과 오브젝트 관계(SQL SERVER사례)

객체권한	테이블	VIEWS	FUNCTION	PROCEDURE
ALTER	O		O	
DELETE	O	O	O	
EXECUTE				O
INDEX	O			
INSERT	O	O		
REFERENCES	O			
SELECT	O	O	O	
UPDATE	O	O		

- DCL(DATA CONTROL LANGUAGE)

유저와 권한

2) OBJECT에 대한 권한 부여

- 모든 유저는 각각 자신이 생성한 테이블 외에 다른 테이블에 접근하려면 해당 테이블에 대한 오브젝트 권한을 소유자로 부터 받아야 한다.

- SQL_SERVER도 마찬가지이며, 다른점은 오브젝트가 유저 소유가 아닌 스키마가 소유하게 되며, 유저는 스키마에 대한 특정권한을 갖는다.

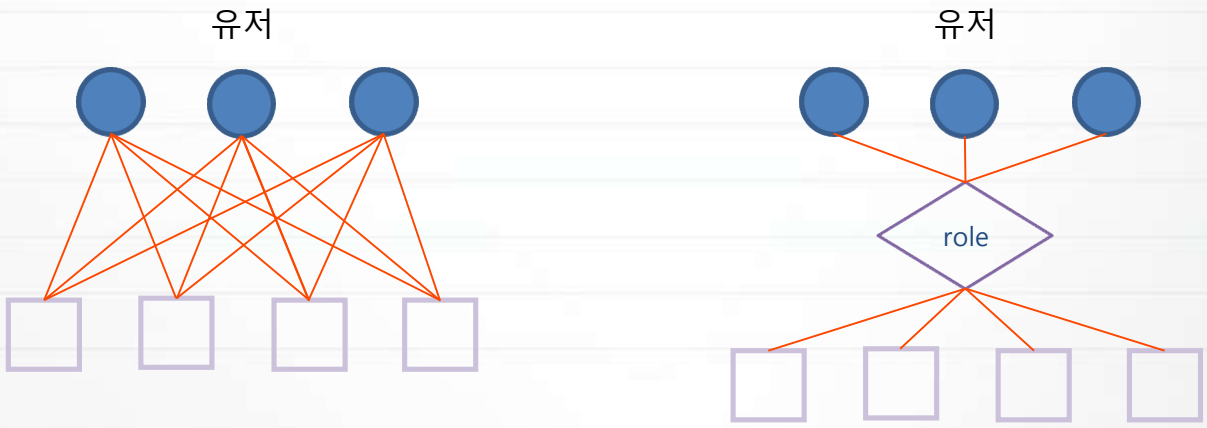
- DCL(DATA CONTROL LANGUAGE)

ROLE을 이용한 권한 부여

- DBA는 유저를 생성할 대마다 권한을 부여하거나 변경해야 하는데 유저가 늘어날 수록 자주 권한을 변경해야 하는데 그와 같은 번거로움 등의 문제를 줄이기 위해 ROLE 제공

- DBA는 ROLE을 생성하고, ROLE에 각종 권한을 부여한 후 ROLE을 다른 ROLE이나 유저에게 부여할 수 있다. 또한, ROLE에 포함되어 있는 권한들이 필요한 유저에게는 해당 ROLE만을 부여하여 빠르고 정확하게 처리.

- ROLE에는 시스템 권한고 오브젝트 권한을 모두 부여할 수 있으며 ROLW은 유저에게 직접부여 될 수 도 있고, 다른 ROLE에 포함하여 유저에게 부여 된다.
- ROLE에 대한 개념.



- DCL(DATA CONTROL LANGUAGE)

ROLE을 이용한 권한 부여

- ORACLE에서는 기본적으로 몇가지 ROLE을 제공
  그 중 많이 사용하는 ROLE은 CONNECT 와 RESOURCE이다.

- CONNECT는 CREATE SESSION과 같은 로그인 권한이 포함되어 있고 RESOURCE는
  CREATE TABLE과 같은 오브젝트의 생성 권한이 포함되며 일반적으로 유저를 생성할 때
  CONNECT 와 RESOURCE을 사용하여 기본 권한 부여

- CONNECT ROLE과 RESOURCE ROLE에 포함된 권한 목록(ORACLE 사례)

CONNECT	RESOURCE
ALTER SESSION	CREATE CLUSTER
CREATE CLUSTER	CREATE INDEXTYPE
CREATE DATABASE LINK	CREATE OPERATOR
CREATE MENU_SEQUENCE	CREATE PROCEDURE
CREATE SESSION	CREATE MENU_SEQUENCE
CREATE SYNONYM	CREATE TABLE
CREATE TABLE	CREATE TRIGGER

- DCL(DATA CONTROL LANGUAGE)

ROLE을 이용한 권한 부여

- USER를 삭제하는 명령어는 DROP USER이고, CASCADE 옵션을 주면 해당 유저가 생성한 오브젝트를 먼저 삭제한 후 유저를 삭제

- SQL SERVER에서는 ROLE을 생성하여 사용하기 보다는 기본적으로 제공되는 ROLE에 멤버로 참여하는 방식으로 사용

- 서버 수준 역할 명 (SQL_SERVER)

서버 수준 역할명	설명
PUBLIC	모든 SQL SERVER 로그인은 PUBLIC 서버 역할에 속한다. 서버 보안 주체에게 보안 객체에 대한 특정 사용 권한이 부여되지 않았거나 거부된 경우 사용자는 해당 개체에 대해 PUBLIC으로 부여된 사용 권한을 상속 받는다. 모든 사용자가 개체를 사용할 수 있도록 하려는 경우에만 개체에 PUBLIC 권한을 할항해야 한다.
BULKADMIN	BULK INSERT문을 수행할 수 있다.
DBCREATOR	데이터베이스를 생성, 변경, 삭제 및 복원할 수 있다.
DISKADMIN	디스크 파일을 관리하는데 사용
PROCESSADMIN	SQL SERVER의 인스턴스에서 실행중인 프로세스를 종료할 수 있다.

- DCL(DATA CONTROL LANGUAGE)

ROLE을 이용한 권한 부여

- 서버 수준 역할 명 (SQL_SERVER)

서버 수준 역할명	설명
SECURITYSADMIN	로그인 및 해당 속성을 관리한다. 서버 및 데이터베이스 수준의 사용 권한을 부여 , 거부, 취소 할 수 있다. 또한, 로그인의 암호를 다시 설정 할 수 있다.
SERVERADMIN	서버 차원의 구성 옵션을 변경하고 서버를 종료할 수 있다.
SETUPADMIN	연결된 서버를 추가하거나 제거할 수 있다.
SYSADMIN	서버에서 모든 작업을 수행할 수 있다. 기본적으로 WINDOWS BUILTIN₩ADMINISTORS 그룹의 멤버인 로컬 관리자 그룹은 SYSADMIN 고정 서버 역할의 멤버이다.

- SQL SERVER에서는 ORACLE과 같이 ROLE을 자주 사용하지 않는다.
  대신 위에서 언급한 서버수준 역할 및 데이터베이스 수준 역할을 이용하여 로그인 및 사용자 권한을 제어한다.

- 인스턴스 수준을 요구하는 로그인에는 서버 수준을 역할하며, 데이터베이스 수준을 요구하는 사용자에게는 데이터베이스 수준 역할을 부여

- 절차형 SQL

절차형 SQL의 개요

- 절차형 SQL을 이용하면 SQL문의 연속적인 실행이나 조건에 따른 분기처리를 이용하여 특정기능을 수행하는 저장 모듈을 생성

PL / SQL의 개요

1) PL/SQL의 특징

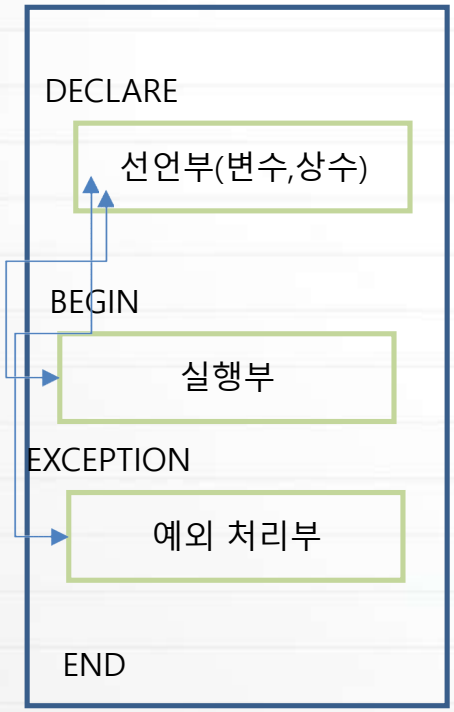
- ORACLE의 PL/SQL은 BLOCK 구조로 되어있고 BLOCK 내에는 DML문장과 QUERY 문장 그리고 절차형 언어 등을 사용할 수 있으며, 절차적 프로그램밍을 가능하게 하는 트랜잭션 언어이다.

- 저장 모듈이란 PL/SQL 문장을 데이터베이스 서버에 저장하여 사용자와 어플리케이션 사이에서 공유할 수 있도록 만든 일종의 SQL 컴포넌트 프로그램
  -> 독립적으로 실행되거나 다른 프로그램으로 부터 실행 될 수 있는 완전한 실행 프로그램이다.
  -> ORACLE의 저장 모듈에는 PROCEDURE, USER , DEFINED FUNCTION TIRGGER 등이 있다.

- 절차형 SQL

PL / SQL의 개요

2) PL / SQL의 구조



-> DECLARE : BEGIN ~ END 절에서 사용될 변수와 인수에 대한 정의 및 데이터 타입을 선언

-> BEGIN ~ END : 개발자가 처리하고자 하는 SQL문과 여러 비교 , 제어문을 이용하여 필요한 조직을 처리하는 실행부

-> EXCEPTION : BEGIN~ END절에서 실행되는 SQL문이 실행 될때 에러가 바생하면 그 에러를 어떻게 처리할 것인지 정의

- 절차형 SQL

PL / SQL의 개요

3) PL / SQL의 기본 문법

- USER DEFINED FUNCTION이나 TRIGGER의 생성방법이나 사용목적은 다르지만 기본 문법은 비슷

```
CREATE [OR REPLACE] Procedure[Procedure_name]
(ARGUMENT1 [MODE] DATA_TYPE1,
ARGUMENT1 [MODE] DATA_TYPE2, ...)
IS[AS]
BEGIN..
```

- 프로시저를 삭제하는 명령어

```
DROP PROCEDURE[PRODEDURE_NAME
```

- 프로시저는 데이터베이스내에 저장되며, 개발자가 자주 실행해야 하는 조직을 절차적 언어를 이용하여 작성한 프로그램 모듈

- 절차형 SQL

PL / SQL의 개요

3) PL / SQL의 기본 문법

- [MODE] 부분에 지정 할 수 있는 매개변수 유형은 3가지
  -> IN은 운영체제에서 프로시저로 전달될 변수의 MODE
  -> out 프로시저에서 처리된 결과가 운영체제로 전달
  -> INOUT은 INT과 OUT 모두 사용가능.

- T-SQL 개요

T-SQL 특징

- 근본적으로SQL_SERVER를 제어하기 위한 언어 이며, ANSI_ISO 표준의 SQL에 약간의 기능을 추가해 보완적으로 만든것.

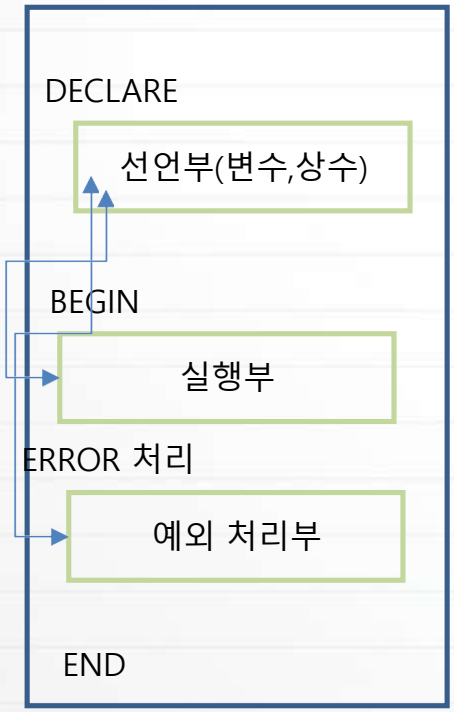
- T-SQL의 프로그래밍 기능

1. 변수 선언 기능 @@ = 전역변수 이며, @ = 지역변수 이다.
2. 지역변수 = 사용자가 자신의 연결 시간 동안만 사용
   전역변수 = 이미 SQL 서버에 내장된 값

3. 데이터 유형 제공 INT, FLOAT, VARCHAR등 존재
4. 산술 연산자, 비교연산자, 논리연산자 등 사용 가능
5. IF-ELSE와 WHILE, CASE-THEN 사용이 가능
6. 주석 가능.

- T-SQL 개요

T-SQL 구조



-> DECLARE : BEGIN ~ END 절에서 사용될 변수와 인수에 대한 정의 및 데이터 타입 선언

-> BEGIN ~ END : 개발자가 처리하고자 하는 SQL문과 여러가지 비교문, 제어문을 이용하여 필요한 로직을 처리

-> ERROR 처리: BEGIN ~ END 절에서 실행되는 SQL문이 실행될 때 ERROR가 발생하면 그 에러를 어떻게 처리 할 지 정의

- T-SQL 개요

T-SQL 기본문법

- USE DEFINETION FUNCTION이나 TRIGGER의 생성 방법과 사용목적은 STORED PROCEDURE와 다르지만 기본적인 분법은 비슷하다.

```
CREATE Procedure [schema_name]Procedure_name
@parameter1 data_type1 [mode],
@parameter2 data_type2 [mode],
...
WITH <proc_option>
AS
....
BEGIN
```

- 생성된 프로시저를 삭제하는 명령어

```
DROP Precedure [schema_name][Procedure_name];
```

- T-SQL 개요

T-SQL 기본문법

- 프로시저의 변경이 필요한 경우 ORACLE은 CREATE OR REPLACE와 같이 하나의 구문을 처리하지만 SQL SERVER는 CREATE 구문을 ALTER 구문으로 변경

- @PARAMETER는 프로시저가 호출될 대 프로시저 안으로 어떤 값이 들어오거나 혹은 프로시저에서 처리나 결과값을 리턴시킬 매개변수를 지정.

- [mode] 부분에 지정할 수 있는 매개 변수의 유형은 4가지 존재
  - -> varying : 결과 집합이 출력 매개변수로 사용되도록 지정한 CURSOUR 매개변수만 적용
  - -> default : 기본값이 지정되어 있으면 해당 매개변수를 지정하지 않아도 프로시저가 지정된 기본 값으로 정상적 수행
  - -> out, out put : 프로시저에서 처리된 결과 값은 excute문 호출시 반환
  - -> readonly : 자주 사용되지 않고, 지정하면 수정할 수 없다.

- T-SQL 개요

T-SQL 기본문법

- with 부분에 지정할 수 있는 옵션은 3가지

-> Recomplie : 데이터베이스 엔진에서 현재 프로시저의 계획을 캐시하지 않고 프로시저가 런타임에 컴파일

-> encryption : 원본 테스트가 알아보기 어려운 형식으로 변환

-> Excate AS : 해당 저장 프로시저를 실행할 보안 컨테스트 지정.

- T-SQL 개요

Procedure의 생성과 활용

- 프로시저를 작성하면서 주의해야 할 점

1. PL/SQL에서 사용하는 SELECT문장에서는 다양한 변수가 있다.
   SCALAR 변수는 사용자의 임시 데이터를 하나만 저장할 수 있는 변수이며,
   거의 모든 형태의 데이터 유형 지정.

2. PL/SQL에서 사용하는 SELECT 문장은 결과값이 반드시 있어야 하며, 그 결과 역시
   반드시 하나여야 한다.

3. T-SQL을 비롯하여 일반적으로 대입 연산자는 “=”을 사용하지만 PL/SQL에서는
   “:=”를 사용한다.

4. Exception에는 When-then절을 사용하여 에러의 여러 종류별로 적절히 처리한다.
   Other를 이용하여 모든 Error를 처리 할 수 있지만 정확하게 Error를 처리하는 것이 좋다.

- T-SQL 개요

User Defined Function의 생성과 활용

- Procedure처럼 절차형 SQL을 로직과 함께 데이터베이스내에 저장해 놓은 명령문의 집합

- FUNCTION이 PROCEDURE와 다른점은 RETURN을 사용해서 하나의 값을 반드시 되돌려 줘야 한다.

  -> FUNCTION은 PROCEDURE와는 달리 SQL 문장에서 특정 작업을 수행하고 반드시 수행 결과 값을 리턴한다.

- T-SQL 개요

TRIGGER 생성과 활용

- 특정한 테이블에 INSERT, UPDATE, DELETE와 같은 DML문이 수행 됫을 때 데이터베이스에서 자동으로 동작하도록 작성된 프로그램

- 테이블과 뷰, 데이터 베이스 작업을 대상으로 정의
  전체 트랜잭션 작업에 대해 발생되는 TRIGGER와 각 행에 대해 발생하는 TRIGGER가 있다.

- Trigger에서 사용하는 레코드 구조체 비교.(1)

구분	:OLD	:NEW
INSERT	NULL	입력된 레코드 값
UPDATE	UPDATE되기 전의 레코드 값	UPDATE된 후의 레코드 값
DELETE	레코드가 삭제되기 전 값	NULL

- T-SQL 개요

TRIGGER 생성과 활용

- Trigger에서 사용하는 레코드 구조체 비교.(2)

구분	DELETE	INSERTED
INSERT	NULL	입력된 레코드 값
UPDATE	UPDATE가 되기 전의 레코드 값	UPDATE된 후의 레코드 값
DELETE	레코드가 삭제되기 전 값	NULL

- ROLLBACK을 하면 하나의 트랜잭션이 취소가 되어 TRIGGER로 입력된 정보까지 하나의 트랜잭션으로 인식하여 두 테이블 모두 입력 취소

- TRIGGER는 데이터베이스에 의해 자동 호출되지만 결국 INSERT,UPDATE,DELETE문과 하나의 트랜잭션 안에서 일어나는 일련의 작업들이라 할 수 있다.

- 데이터 베이스 보안의 적용, 유효하지 않은 트랜잭션의 예방, 업무 규칙, 자동 적용 제공 등에 사용.

- T-SQL 개요

PROCEDURE와 트리거의 차이점

- 프로시저는 BEGIN~END절 외에 COMMIT, ROLLBACK과 같은 트랜잭션 종료 명령어를 사용할 수 있지만, 데이터베이스 트리거는 BEGIN ~ END절 내에 사용할 수 없다.

- 프로시저와 트리거의 차이점.

프로시저	트리거
CREATE Procedure 문법사용	CREATE TRIGGER 문법 사용
EXECUTE 명령어로 실행	생성 후 자동으로 실행
COMMIT, ROLLBACK 실행 가능	COMMIT, ROLLBACK 실행 안됨.

– 장 요약

제 1절 표준조인

ANSI/ISO 표준 SQL에서 규정한 INNER JOIN, USING 조건절, ON 조건절, CROSS JOIN, OUTER JOIN 문법을 통해 사용자는 테이블 간의 JOIN조건을 FROM 절에서 명시적으로 정의할 수 있다.

제 2절 집합 연산자

두 개 이상의 테이블에서 JOIN을 사용하지 않고 SET 연산자는 여러 개의 SQL문을 연결하여 데이터를 결합하는 방식을 사용한다. UNION은 합집합, UNION ALL은 확장된 합집합, INTERSECT는 교집합, EXCEPT / MINUS는 차집합을 나타낸다.

제 3절 계층형 질의와 셀프 조인

테이블에 계층형 데이터가 존재하는 경우 데이터를 조회하기 우해서 계층형 질의를 사용한다. 셀프 조인이란 동일 테이블 사이의 조인을 말하며, FROM 절에 동일 테이블이 두 번 이상 나타난다.

– 장 요약

제 4절 서브쿼리

서브쿼리란 하나의 메인쿼리 안에 포함되어 있는 종속적인 SQL 문장을 말하는 것으로, 위치나 기능에 따라 NESTED SUBQUERY, INLINE VIEW, SCALAR SUBQUERY로 구분할 수 있다.

제 5절 그룹 함수

데이터 분석을 위한 GROUP FUNCTION으로는 소그룹 간의 소계를 계산하는 ROLLUP 함수, GROUP BY 항목들 간의 다차원적인 소계를 계산할 수 있는 CUBE 함수, 특정 항목에 대한 소계를 계산하는 GROUPING SETS 함수가 있다.

제 6절 윈도우 함수

데이터 분석을 위한 WINDOW FUNCTION은 부분적이나마 행과 행간의 관계를 쉽게 정의하기 위해 만든 함수이다. WINDOW FUNCTION을 이용한 순위(RANK) 관련 함수는 RANK, DENSE_RANK,ROW_NUMBER 함수가 있으며, 그 외 그룹 내 집계(AGGREGATE) 관련 함수, 그룹 내 비율 관련 함수 등이 있다.

– 장 요약

제 7절 DCL

유저를 생성하고 권한을 제어할 수 있는 DCL(DATA CONTROL LANGUAGE)명령어가 있고, GRANT 문장을 통해 권한을 생성하고 REVOKE 문장을 통해 권한을 회수한다.

제 8절 절차형 SQL

절차형 SQL을 이용하여 SQL문장의 조건에 따른 분기 처리나 SQL 문장의 연속적인 실행을 이용하여 특정 기능을 수행하는 저장 모듈을 생성할 수 있다. 절차형 SQL을 이용하여 PROCEDURE, TRIGGER, USER DEFINED FUNCTION을 만들 수 있다.

감사합니다!