

대분류/20  
정보통신

중분류/01  
정보기술

소분류/02  
정보기술개발

세분류/04  
DB엔지니어링

능력단위/14

NCS학습모듈

# SQL 응용

LM2001020414\_16v3



교육부

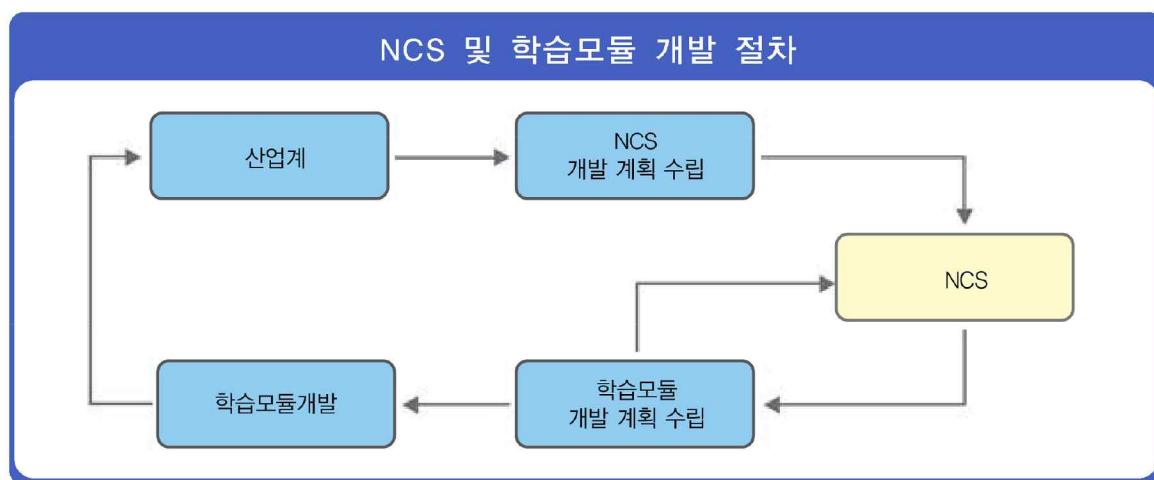
NCS 학습모듈은 교육훈련기관에서 출처를 명시하고 교육적 목적으로 활용할 수 있습니다. 다만 NCS 학습모듈에는 국가(교육부)가 저작재산권 일체를 보유하지 않은 저작물들(출처가 표기되어 있는 도표, 사진, 삽화, 도면 등)이 포함되어 있으므로 이러한 저작물들의 변형, 복제, 공연, 배포, 공중 송신 등과 이러한 저작물들을 활용한 2차 저작물의 생성을 위해서는 반드시 원작자의 동의를 받아야 합니다.

# NCS 학습모듈의 이해

\* 본 학습모듈은 「NCS 국가직무능력표준」 사이트(<http://www.ncs.go.kr>)에서 확인 및 다운로드 할 수 있습니다.

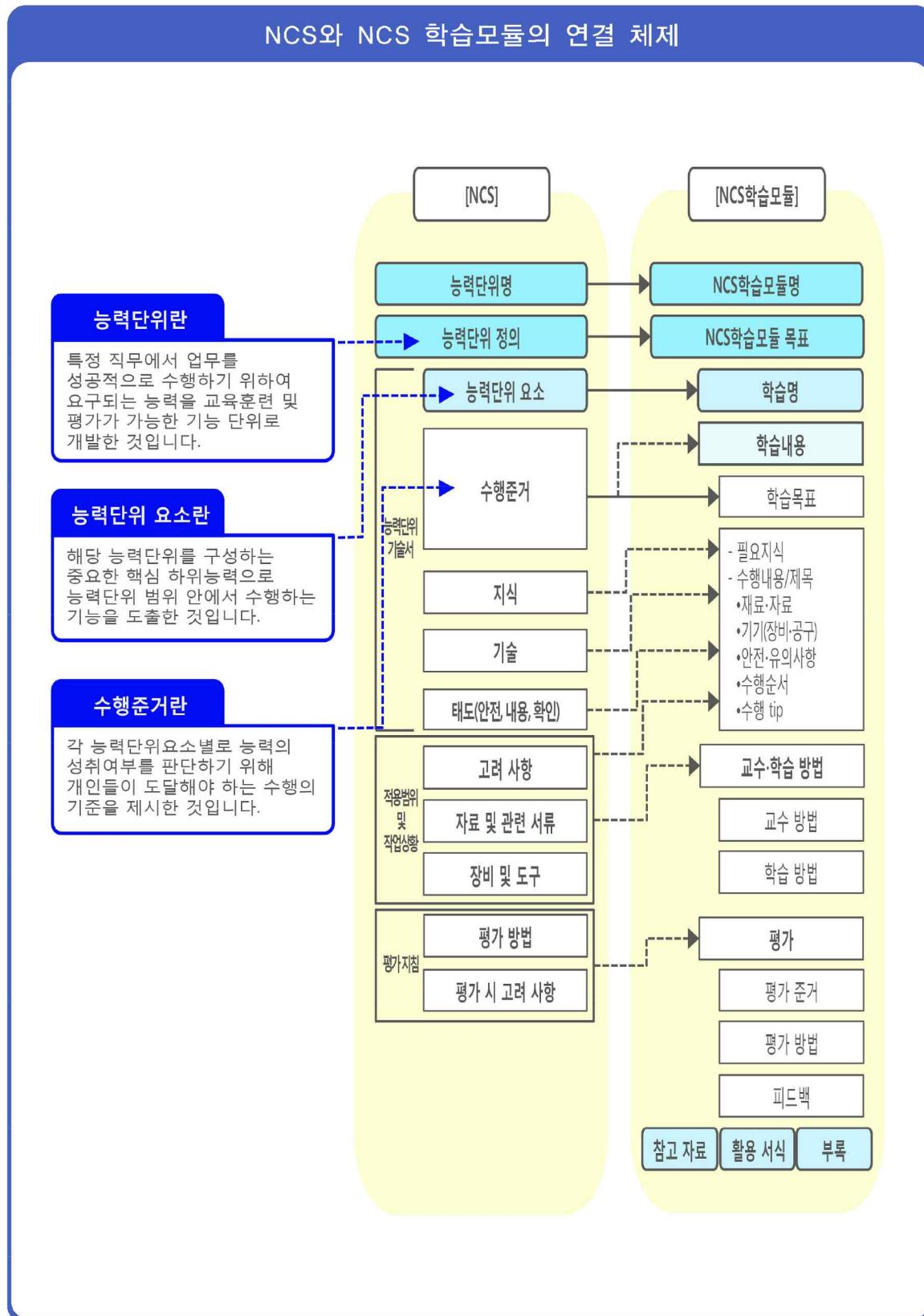
## (1) NCS 학습모듈이란?

- 국가직무능력표준(NCS: National Competency Standards)이란 산업현장에서 직무를 수행하기 위해 요구되는 지식·기술·소양 등의 내용을 국가가 산업부문별·수준별로 체계화한 것으로 산업현장의 직무를 성공적으로 수행하기 위해 필요한 능력(지식, 기술, 태도)을 국가적 차원에서 표준화한 것을 의미합니다.
- 국가직무능력표준(이하 NCS)이 현장의 ‘직무 요구서’라고 한다면, NCS 학습모듈은 NCS의 능력단위를 교육훈련에서 학습할 수 있도록 구성한 ‘교수·학습 자료’입니다. NCS 학습모듈은 구체적 직무를 학습할 수 있도록 이론 및 실습과 관련된 내용을 상세하게 제시하고 있습니다.



- NCS 학습모듈은 다음과 같은 특징을 가지고 있습니다.
  - 첫째, NCS 학습모듈은 산업계에서 요구하는 직무능력을 교육훈련 현장에 활용할 수 있도록 성취목표와 학습의 방향을 명확히 제시하는 가이드라인의 역할을 합니다.
  - 둘째, NCS 학습모듈은 특성화고, 마이스터고, 전문대학, 4년제 대학교의 교육기관 및 훈련기관, 직장교육기관 등에서 표준교재로 활용할 수 있으며 교육과정 개편 시에도 유용하게 참고할 수 있습니다.

- NCS와 NCS 학습모듈 간의 연결 체계를 살펴보면 아래 그림과 같습니다.



## (2) NCS 학습모듈의 체계

- NCS 학습모듈은 1.학습모듈의 위치, 2.학습모듈의 개요, 3.학습모듈의 내용 체계, 4.참고 자료, 5.활용 서식/부록으로 구성되어 있습니다.

### 1. NCS 학습모듈의 위치

- NCS 학습모듈의 위치는 NCS 분류 체계에서 해당 학습모듈이 어디에 위치하는지를 한 눈에 볼 수 있도록 그림으로 제시한 것입니다.

예시 : 이 · 미용 서비스 분야 중 네일미용 세분류

### NCS-학습모듈의 위치

대분류	이용 · 숙박 · 여행 · 오락 · 스포츠
중분류	이 · 미용
소분류	이·미용 서비스

#### 세분류

헤어미용	능력단위	학습모듈명
피부미용	네일 샵 위생 서비스	네일숍 위생서비스
메이크업	네일 화장을 제거	네일 화장을 제거
<b>네일미용</b>	<b>네일 기본 관리</b>	<b>네일 기본관리</b>
이용	네일 랩	네일 랩
	네일 팁	네일 팁
	젤 네일	젤 네일
	아크릴릭 네일	아크릴 네일
	평면 네일아트	평면 네일아트
	융합 네일아트	융합 네일아트
	네일 샵 운영관리	네일숍 운영관리

#### 학습모듈은

NCS 능력단위 1개당 1개의 학습모듈 개발을 원칙으로 합니다. 그러나 필요에 따라 고용 단위 및 교과단위를 고려하여 능력단위 몇 개를 묶어서 1개의 학습모듈로 개발할 수 있으며, NCS 능력단위 1개를 여러 개의 학습 모듈로 나누어 개발할 수도 있습니다.

## 2. NCS 학습모듈의 개요



### 구성

- NCS 학습모듈 개요는 학습모듈이 포함하고 있는 내용을 개략적으로 설명한 것으로서 **학습모듈의 목표**, **선수 학습**, **학습모듈의 내용 체계**, **핵심 용어**로 구성되어 있습니다.

#### 학습모듈의 목표

해당 NCS 능력단위의 정의를 토대로 학습목표를 작성한 것입니다.

#### 선수 학습

해당 학습모듈에 대한 효과적인 교수·학습을 위하여 사전에 이수해야 하는 학습모듈, 학습 내용, 관련 교과목 등을 기술한 것입니다.

#### 학습모듈의 내용 체계

해당 NCS 능력단위요소가 학습모듈에서 구조화된 방식을 제시한 것입니다.

#### 핵심 용어

해당 학습모듈의 학습 내용, 수행 내용, 설비·기자재 등 가운데 핵심적인 용어를 제시한 것입니다.



### 활용 안내

예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈

#### 네일 기본관리 학습모듈의 개요

##### 학습모듈의 목표

고객의 네일 보호와 미적 요구 충족을 위하여 효과적인 네일 관리로 프리에지 형태 만들기, 큐티를 정리하기, 컬러링하기, 보습제 도포하기, 마무리를 할 수 있다.

##### 선수학습

네일숍 위생서비스(LM1201010401\_14V2)

##### 학습모듈의 내용체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 프리에지 형태 만들기	1-1. 네일 파일에 대한 이해와 활용 1-2. 프리에지 형태 파일링	1201010403_12v2.1	프리에지 모양 만들기
2. 큐티를 정리하기	2-1. 네일 기본관리 매뉴얼 이해 2-2. 큐티를 관리	1201010403_14v2.2	큐티를 정리하기
3. 컬러링하기	3-1. 컬러링 매뉴얼 이해 3-2. 컬러링 방법 선정과 작업 3-3. 웰 컬러링 작업	1201010403_14v2.3	컬러링
4. 보습제 도포하기	4-1. 보습제 선정과 도포 4-2. 각질제거	1201010403_14v2.4	보습제 바르기
5. 네일 기본관리 마무리하기	5-1. 유분기 제거 5-2. 네일 기본관리 마무리와 정리	1201010403_14v2.5	마무리하기

##### 학습모듈의 목표는

학습자가 해당 학습모듈을 통해 성취해야 할 목표를 제시한 것으로, 교수자는 학습자가 학습모듈의 전체적인 내용흐름을 파악할 수 있도록 지도하는 것이 필요합니다.

##### 선수 학습은

교수자나 학습자가 해당 모듈을 교수 또는 학습하기 이전에 이수해야 할 학습내용, 교과목, 핵심 단어 등을 표기한 것입니다. 따라서 교수자는 학습자가 개별 학습, 자기 주도 학습, 방과 후 활동 등 다양한 방법을 통해 이수할 수 있도록 지도하는 것이 필요합니다.

##### 핵심 용어는

학습모듈을 통해 학습되고 평가되어야 할 주요 용어입니다. 또한 당해 모듈 또는 타 모듈에서도 핵심 용어를 사용하여 학습내용을 구성할 수 있으며, 「NCS 국가 직무능력표준」사이트([www.ncs.go.kr](http://www.ncs.go.kr))에서 색인(찾아보기) 중 하나로 이용할 수 있습니다.

##### 핵심 용어

프리에지, 니퍼, 퓨셔, 폴리시, 네일 파일, 스웨어형, 스웨어 오프형, 라운드형, 오발형, 포인트형

### 3. NCS 학습모듈의 내용 체계



- NCS 학습모듈의 내용은 크게 **학습**, **학습 내용**, **교수·학습 방법**, **평가**로 구성되어 있습니다.

#### 학습

해당 NCS 능력단위요소 명칭을 사용하여 제시한 것입니다.

학습은 크게 학습 내용, 교수·학습 방법, 평가로 구성되며 해당 NCS 능력단위의 능력단위 요소별 지식, 기술, 태도 등을 토대로 학습 내용을 제시한 것입니다.

#### 학습 내용

학습 내용은 학습 목표, 필요 지식, 수행 내용으로 구성하였으며, 수행 내용은 재료·자료, 기기(장비·공구), 안전·유의 사항, 수행 순서, 수행 tip으로 구성한 것입니다. 학습모듈의 학습 내용은 업무의 표준화된 프로세스에 기반을 두고 실제 산업현장에서 이루어지는 업무활동을 다양한 방식으로 반영한 것입니다.

#### 교수·학습 방법

학습 목표를 성취하기 위한 교수자와 학습자 간, 학습자와 학습자 간의 상호작용이 활발하게 일어날 수 있도록 교수자의 활동 및 교수 전략, 학습자의 활동을 제시한 것입니다.

#### 평가

평가는 해당 학습모듈의 학습 정도를 확인할 수 있는 평가 준거, 평가 방법, 평가 결과의 피드백 방법을 제시한 것입니다.



예시 : 네일미용 세분류의 ‘네일 기본관리’ 학습모듈의 내용

학습 1	프리에지 형태 만들기(LM1201010403_14v2.1)
학습 2	큐티클 정리하기(LM1201010403_14v2.2)
<b>학습 3 컬러링하기(LM1201010403_14v2.3)</b>	
학습 4	보습제 도포하기(LM1201010403_14v2.4)
학습 5	네일 기본관리 미무리하기(LM1201010403_14v2.5)

#### 학습은

해당 NCS 능력단위요소 명칭을 사용하여 제시하였습니다. 학습은 일반교과의 ‘대단원’에 해당되며, 모듈을 구성하는 가장 큰 단위가 됩니다. 또한 완성된 직무를 수행하기 위한 가장 기초적인 단위로 사용할 수 있습니다.

#### 학습내용은

요소 별 수행준거를 기준으로 제시하였습니다. 일반교과의 ‘중단원’에 해당합니다.

#### 학습목표는

모듈 내의 학습내용을 이수했을 때 학습자가 보여줄 수 있는 행동수준을 의미합니다. 따라서 일반 수업시간의 과목목표로 활용할 수 있습니다.

#### 3-1. 컬러링 매뉴얼 이해

##### 학습목표

- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 얇게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시를 일plex 없이 균일하게 도포할 수 있다.
- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 톱코트를 바를 수 있다.

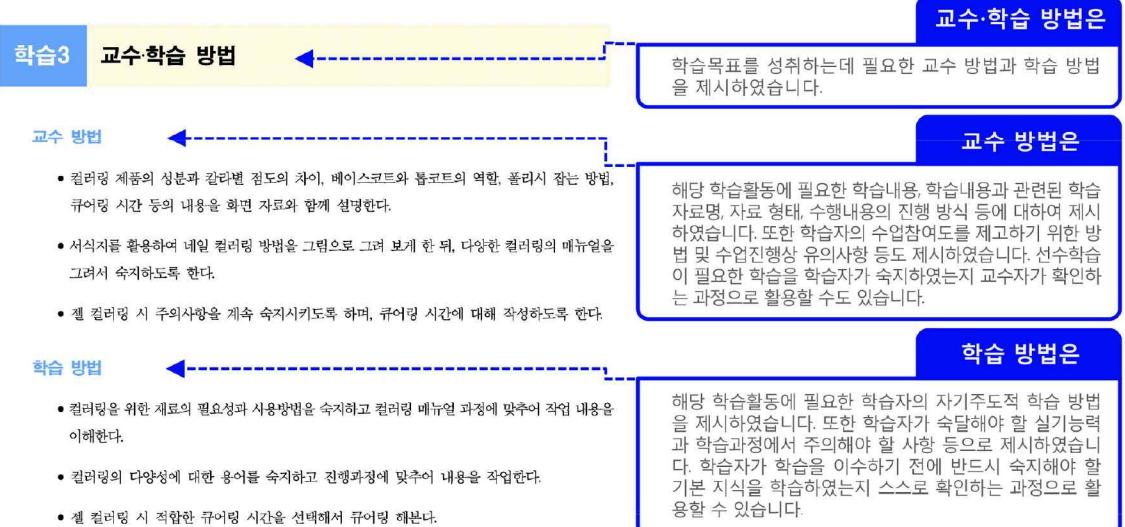
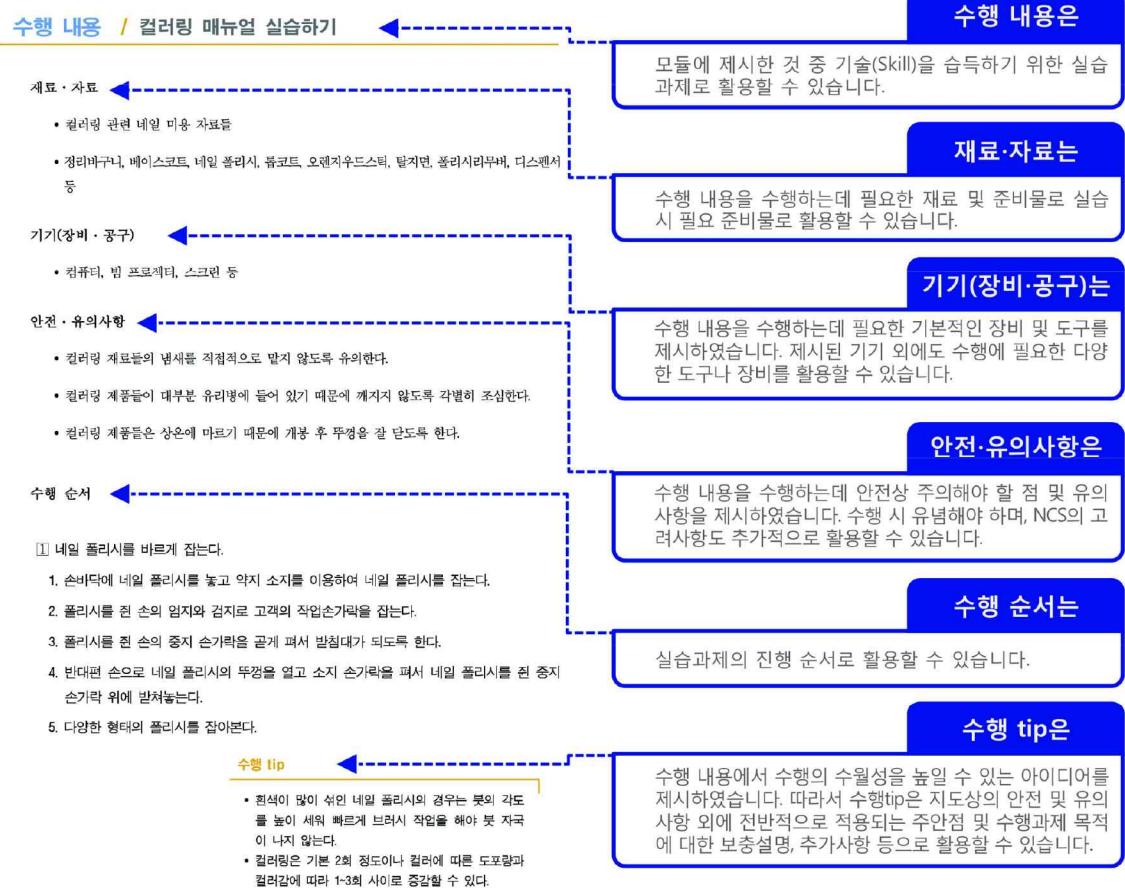
##### 필요 지식 /

##### ① 컬러링 매뉴얼

컬러링 작업 전, 아세톤 또는 네일 폴리시 리무버를 사용하여 손톱표면과 큐티클 주변, 손톱 일부 부분까지 깨끗하게 유분기를 제거해야 한다. 컬러링의 순서는 Base coating 1회 → Polishing 2회 → 컬러수정 → Top coating 1회 → 최종수정의 순서로 한다. 베이스코트는 착색을 방지하고 빌릴성 향상을 위해 가장 먼저 도포하며 컬러링의 마지막에 컬러의 유지와 광택을 위해 톱코트를 도포한다. 네일 보강제(Nail Strengthener)를 바를 시에는 베이스코트를 도포하기 전에 사용한다.

#### 필요지식은

해당 NCS의 지식을 토대로 해당 학습에 대한 이해와 성과를 높이기 위해 알아야 할 주요 지식을 제시하였습니다. 필요지식은 수행에 꼭 필요한 핵심 내용을 위주로 제시하여 교수자의 역할이 매우 중요하며, 이후 수행순서 내용과 연계하여 교수·학습으로 진행할 수 있습니다.



## 학습3 평가

## 평가 준거

- 평가는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.

- 평가는 다음 사항을 평가해야 한다.

학습내용	학습목표	성취수준
		상 중 하
컬러링 매뉴얼 이해	- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 알게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시를 얼룩 없이 균일하게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 툴코트를 바를 수 있다.	

## 평가 방법

- 작업장 평가

학습내용	평가 항목	성취수준
		상 중 하
컬러링 매뉴얼 이해	- 고객의 요구에 따라 네일 폴리시 색상의 침착을 막기 위한 베이스코트를 아주 알게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시를 얼룩 없이 균일하게 도포할 수 있다.	
	- 작업 매뉴얼에 따라 네일 폴리시 도포 후 컬러 보호와 광택 부여를 위한 툴코트를 바를 수 있다.	

## 피드백

## 1. 작업장 평가

- 작업 결과물을 확인하여 수정사항을 제시하고 수정 부분을 인지하도록 한다.

## 4. 참고 자료

## 참고 자료

• 김미원(2011).『Nail Study』. 서울: 사)한국네일자식서비스협회.

• 민방경(2015).『미용사(네일)필기』. 서울: 예문사.

• 박은주(2014).『네일미용』. 서울: 정담미디어.

## 5. 활용 서식/부록

## 활용서식

## 프리에지 형태 실습지

## 1. 프리에지 형태의 이해

모양	이름	특징
	( ) Square nail	- 강한 느낌의 사각형태 - 네일의 양끝 모서리 부분이 90° 사각의 형태이다. - 발톱의 형태 활용 - 내인성 발톱의 보정시에 적용

해당 NCS 능력단위 평가방법과 평가 시 고려 사항을 준용하여 작성하였습니다. 교수자 및 학습자가 평가항목 별 성취수준을 확인하는데 활용할 수 있습니다.

## 평가 준거는

학습자가 해당 학습을 어느 정도 성취하였는지를 평가하기 위한 기준을 제시하고 있습니다. 학습목표와 연계하여 단위수업 시간에 평가항목 별 성취수준을 평가하는데 활용할 수 있습니다.

## 평가 방법은

NCS 능력단위의 평가방법을 준용하였으며, 평가 준거에 따른 평가방법을 2개 이상 제시하였습니다. 평가방법으로는 포트폴리오, 문제해결 시나리오, 서술형 시험, 논술형 시험, 사례연구, 평가자 체크리스트, 작업장 평가 등이 있으며, NCS의 능력단위 요소 별 수행 수준을 평가하는데 가장 적절한 방법을 선정하여 활용할 수 있습니다.

## 피드백은

평가 후에 학습자들에게 평가 결과를 피드백하여 부족한 부분을 알려주고, 학습 결과가 미진한 경우, 해당 부분을 다시 학습하여 학습목표를 달성하는 데 활용할 수 있습니다.

## 참고자료는

해당 학습모듈의 필요지식에 대한 출처와 인용한 참고자료 및 사이트를 제시하였습니다.

## 활용서식은

평가 서식, 실험시트 등 교수학습 시 활용 가능한 다양한 서식들로 구성하였습니다. 과제 진행에서 평가에 이르기까지 필요한 서식을 해당 학습모듈의 특성에 맞춰 개발하거나 기존의 양식을 활용하여 제시하였습니다.

## 부록

## 네일 기본관리 도구와 재료 목록

목록	비고	준비
위생가운	흰색	작업자 착용
위생 미스트	흰색	작업자 착용
보호안경	투명한 렌즈 (안경으로 대체 가능)	작업자 착용
재료정리함	제질, 색상 무관	작업대

활용서식 이외에 교수학습과정에서 참고할 수 있는 자료가 있는 경우 제시하였습니다.

## 부록은

# [NCS-학습모듈의 위치]

대분류	정보통신
중분류	정보기술
소분류	정보기술개발

세분류	능력단위	학습모듈명
SW아키텍처	데이터베이스 요구사항 분석	데이터베이스 요구사항 분석
응용SW 엔지니어링	개념데이터 모델링	개념데이터 모델링
임베디드SW 엔지니어링	논리 데이터베이스 설계	논리 데이터베이스 설계
DB엔지니어링	물리 데이터베이스 설계	물리 데이터베이스 설계
NW엔지니어링	데이터베이스 구현	데이터베이스 구현
보안엔지니어링	데이터 품질관리	데이터 품질관리
UI/UX엔지니어링	데이터 전환 설계	데이터 전환 설계
시스템SW 엔지니어링	데이터 전환	데이터 전환
	데이터베이스 성능확보	데이터베이스 성능확보
	데이터 표준화	데이터 표준화
	SQL활용	SQL활용
	SQL응용	SQL응용

---

# 차 례

---

학습모듈의 개요	1
학습 1. 절차형 SQL 작성하기	
1-1. 프로시저 및 호출문 작성	3
1-2. 사용자 정의함수 및 호출쿼리 작성	15
1-3. 트리거 작성	24
• 교수 · 학습 방법	32
• 평가	34
학습 2. 응용 SQL 작성하기	
2-1. 집계성 SQL 작성	38
2-2. 특정 기능 수행 SQL문 작성	52
2-3. DCL 명령문 작성	65
• 교수 · 학습 방법	74
• 평가	76
참고 자료	79



# SQL 응용 학습모듈의 개요

## 학습모듈의 목표

관계형 데이터베이스에서 SQL을 사용하여 응용시스템의 요구 기능에 적합한 데이터를 정의하고 조작하며 제어할 수 있다.

## 선수학습

데이터베이스 요구사항 분석, 데이터 표준화(2001020409\_16v3), SQL 활용(2001020413\_16v3)

## 학습모듈의 내용 체계

학습	학습 내용	NCS 능력단위 요소	
		코드 번호	요소 명칭
1. 절차형 SQL 작성하기	1-1. 프로시저 및 호출문 작성 1-2. 사용자 정의함수 및 호출쿼리 작성 1-3. 트리거 작성	2001020414_16v3.1	절차형 SQL 작성하기
2. 응용 SQL 작성하기	2-1. 집계성 SQL 작성 2-2. 특정 기능 수행 SQL문 작성 2-3. DCL 명령문 작성	2001020414_16v3.2	응용 SQL 작성하기

## 핵심 용어

데이터 질의어(DQL: Data Query Language), 데이터 조작어(DML: Data Manipulate Language), 데이터 제어어(DCL: Data Control Language), 트랜잭션(Transaction)



## 학습 1

# 절차형 SQL 작성하기

## 학습 2

## 응용 SQL 작성하기

## 1-1. 프로시저 및 호출문 작성

### 학습 목표

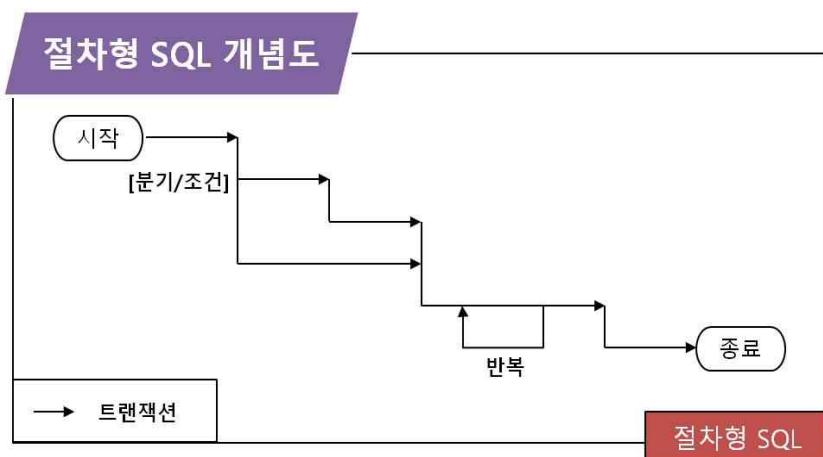
- 반복적으로 사용하는 특정 기능을 수행하기 위해 여러 개의 SQL 명령문을 포함하는 프로시저를 작성하고 프로시저 호출문을 작성할 수 있다.

### 필요 지식 /

#### ① 절차형 SQL

##### 1. 절차형 SQL의 개념

타 개발 언어와 유사하게 SQL에도 절차 지향적인 프로그램이 가능하며, 이를 이용하여 SQL문의 연속적인 실행이나 조건에 따른 분기, 반복 등의 제어를 활용하여 다양한 기능을 수행하는 저장 모듈을 생성하고 쉽게 활용할 수 있다. 이는 DB 작업의 고효율화를 기반으로 높은 생산성을 확보할 수 있게 해 준다. 다음은 절차형 SQL 내의 작업 진행을 간략히 표시한 추상적 개념도이다.



[그림 1-1] 절차형 SQL 개념도

##### 2. 절차형 SQL의 특징

절차형 SQL의 특징은 다음과 같다.

- DBMS 엔진에서 직접 실행한다.

- BEGIN/END의 Block화된 구조로 되어 있어 각 기능별로 모듈화가 가능하다.
  - 조건문, 반복문 등 단일 SQL 문장으로는 실행하기 어려운 연속적인 작업을 처리하는 데에 적합하다.
  - 일련의 작업에 필요한 데이터를 DBMS 내부에서 직접 처리하기 때문에 일반적으로 Input/Output Packet이 적다.
  - 타 절차형 언어에 비해 작업의 효율성은 낮은 편이다.
  - DBMS 벤더별 차이가 있어서 벤더가 다른 환경에 이식하는 경우 수정 및 재컴파일이 필요하다.

### 3. 절차형 SQL의 구성

프로시저, 사용자 정의함수 및 트리거 등이 다소 차이가 있으나 필수적 요소를 고려한 구성은 아래와 같다. 아래의 요소만 갖추어도 정상적인 컴파일도 가능하고 실행도 가능하다.



[그림 1-2] 절차형 SQL 구성도

- DECLARE: 대상이 되는 프로시저, 사용자 정의함수 등을 정의
  - BEGIN: 프로시저, 사용자 정의함수가 실행되는 시작점
  - END: 프로시저, 사용자 정의함수가 실행되는 종료점

#### 4. 절차형 SQL의 유형

DBMS 벤더에 따라 P제품(O사), T제품(M사), S제품(D사) 등의 절차형 SQL이 있다. 벤더에 따라 특성성 차이가 있으나 기본적 맥락은 동일하다. 다만, 벤더가 다른 환경에 이식하는 경우 수정 및 재컴파일이 필요할 수 있다.

## (1) P제품

O사의 SQL 언어를 확장하기 위해 사용하는 절차형 SQL 언어이다. P제품의 특징은 다음과 같다.

- O사와 P제품을 지원하는 어떤 서버로도 프로그램을 옮길 수 있는 이식성을 가진다.
- P프로그램을 입력받으면 SQL 문장과 프로그램 문장을 구분하여 처리한다. 즉, 프로그램 문장은 P 엔진이 처리하고 SQL 문장은 O사 서버의 SQL Statement Executor가 실행하도록 작업을 분리하여 처리한다.

## (2) T제품

M사에서 ANSI / ISO 표준의 SQL에 약간의 기능을 더 추가하여 보완적으로 만든 것이다.

- 변수 선언 기능 @@이라는 전역 변수(시스템 함수)와 @이라는 전역 변수가 있다.
- 지역 변수는 사용자가 자신의 연결 시간 동안만 사용하기 위해 만들어지는 변수이며, 전역 변수는 이미 SQL 서버에 내장된 값이다.
- 흐름 제어 사용이 가능하며 연산자 사용, 데이터 유형 제공이 가능하다.

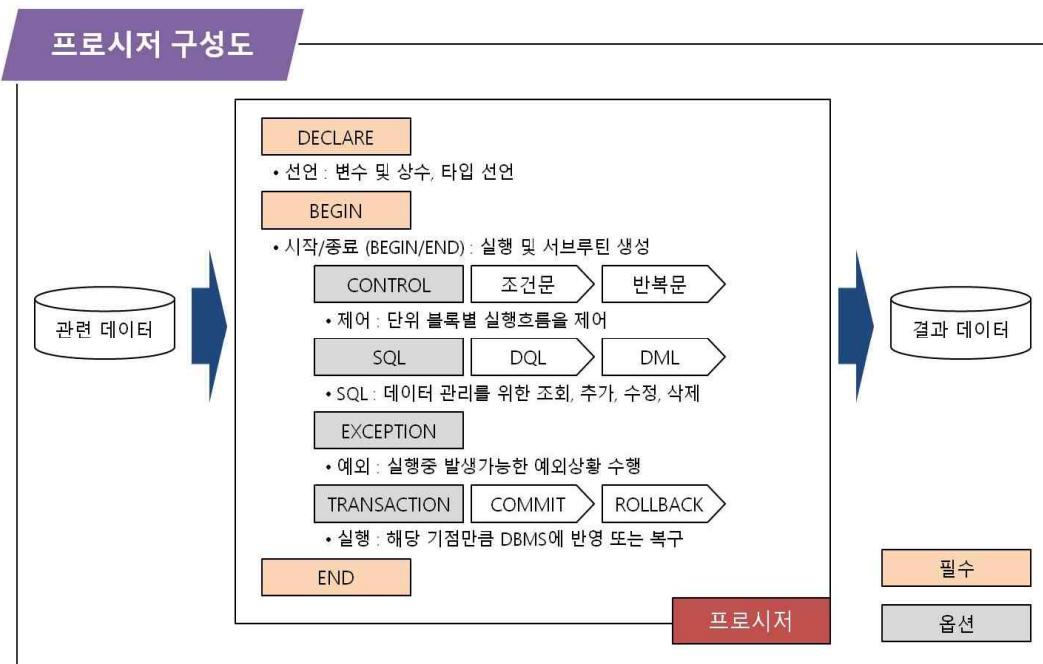
## ② 프로시저 작성

### 1. 프로시저의 개념

프로시저는 절차형 SQL을 활용하여 특정 기능을 수행할 수 있는 트랜잭션 언어이다. 프로시저 호출을 통해 실행되며, 이를 통해 일련의 SQL 작업을 포함하는 데이터 조작어(DML: Data Manipulate Language)를 수행하는 것이 일반적이다. 시스템에서의 일일 마감 작업, 또는 일련의 배치 작업 등을 프로시저를 활용하여 관리하고 주기적으로 수행하기도 한다.

### 2. 프로시저의 구성

일반적인 프로시저의 구성은 다음과 같다.



[그림 1-3] 프로시저 구성도

- DECLARE: 프로시저의 명칭, 변수와 인수 그리고 그에 대한 데이터 타입을 정의하는 선언부이다.
- BEGIN / END: 프로시저의 시작과 종료를 표현하는 데 필수적이며, BEGIN / END를 쌍을 이루어 추가하므로 Block을 구성한다. 다수 실행을 제어하는 기본적 단위가 되며 논리적 프로세스를 구성한다.
- CONTROL: 기본적으로는 순차적으로 처리한다. 특정 비교 조건에 따라(IF) 참인 Block 또는 문장을 실행하거나(THEN), 기타의 경우에도 조건에 맞는 Block 또는 문장을 실행 한다(ELSIF 또는 ELSE). 조건에 따라 반복을 수행할 수도 있다(LOOP).
- SQL: DQL(SELECT), DML(INSERT, UPDATE, DELETE)을 주로 사용한다. 자주 사용하지는 않으나 DDL(TRUNCATE 등)을 사용하기도 한다.
- EXCEPTION: BEGIN ~ END절에서 실행되는 SQL문이 실행될 때 예외 발생 시 예외 처리 방법을 정의하는 처리부이다.
- TRANSACTION: 프로시저에서 수행된 DML 수행 내역의 DBMS의 적용 또는 취소 여부를 결정하는 처리부이다. 일반 SQL과 동일하게 최종 COMMIT / ROLLBACK 시점 이후부터 실행된 DML의 적용 / 취소를 수행한다.

### 3. 프로시저의 문법

상기 설명한 프로시저의 구성에 기반한 프로시저의 문법은 아래와 같다.

<표 1-1> 프로시저 문법

```

CREATE [OR REPLACE] PROCEDURE [PROCEDURE_NAME]
(PARAMETER_1 [MODE] DATA_TYPE1,
PARAMETER_2 [MODE] DATA_TYPE2,
...
)
IS [AS]
...
BEGIN
...
[SQL]
[CONTROL]
...
[EXCEPTION]
...
[COMMIT/ROLLBACK]
END;

```

- CREATE 명령어로 DBMS 내에 프로시저 생성이 가능하다.
- [OR REPLACE] 명령은 기존 프로시저 존재 시에 현재 컴파일하는 내용으로 덮어쓴다는 (Overwrite) 의미이다. 기존 동명의 프로시저가 존재하고 CREATE 명령문만 사용하면 컴파일 시에는 에러가 발생한다.
- PARAMETER는 외부에서 프로시저 호출 시 변수를 입력 또는 출력할 수 있다.  
MODE는 변수의 입력/출력을 구분하며 다음과 같이 사용한다.
  - (가) IN: 운영 체제에서 프로시저로 전달되는 MODE
  - (나) OUT: 프로시저에서 처리된 결과가 운영 체제로 전달되는 MODE
  - (다) INOUT: IN과 OUT의 두 가지 기능을 동시에 수행하는 MODE

#### 4. 프로시저의 예시

아래는 매출 마감 처리를 하는 간단한 프로시저이다.

<표 1-2> 프로시저 예시

```

CREATE OR REPLACE PROCEDURE SALES_CLOSING -- 매출마감처리
(V_CLOSING_DATE IN CHAR(8))           -- 마감일자
IS
BEGIN
  V_SALES_TOT_AMT NUMBER;             -- 매출총액

  SELECT SUM(SALES_AMT)               -- 매출액
    INTO V_SALES_TOT_AMT
   FROM SALES_LIST_T                -- 판매내역
  WHERE SALES_DATE = V_CLOSING_DATE; -- 판매일자

EXCEPTION
  WHEN NO_DATA_FOUND THEN          -- 데이터 미존재시
    SET V_SALES_TOT_AMT = 0 ;       -- 매출총액에 0 입력

  INSERT INTO                      -- 삽입
    SALES_CLOSED_T                -- 마감내역 TABLE
    ( SALES_DATE
      , SALES_TOT_AMT
    )
  VALUES ( V_CLOSING_DATE
        , V_SALES_TOT_AMT
      );
EXCEPTION
  WHEN NO_DATA_FOUND THEN          -- 데이터 미존재시

```

```
SET SALES_TOT_AMT = 0 ;  
  
COMMIT;  
END;
```

- 마감 일자(V\_CLOSING\_DATE)를 입력변수로 선언한다.
- 내부에서 매출 총액 변수(V\_SALES\_TOT\_AMT)를 선언한다.
- 판매 내역 테이블(SALES\_LIST\_T)에서 마감 일자의 매출액 합계(SALES\_AMT)를 계산하여 매출 총액 변수에 입력한다.
- 예외 사항으로 데이터 미존재 시 매출 총액은 0으로 입력한다.
- 마감 내역 테이블(SALES\_CLOSED\_T)에 마감 일자와 매출 총액을 삽입한다.
- 종료 시 COMMIT으로 트랜잭션을 완료 처리한다.

### ③ 프로시저 호출문 작성

작성된 프로시저는 외부 호출을 통해서 실행된다. 응용 프로그램에서 호출하거나 내부 스케줄러에 의해 배치 작업을 수행하는 경우 등에 호출되어 프로시저가 사용된다.

#### 1. 프로시저 호출문의 문법

SQL 명령어를 활용하여 작성된 프로시저를 호출한다.

<표 1-3> 프로시저 호출 문법

```
SQL> EXECUTE [PROCEDURE_NAME] (PARAMETER_1, PARAMETER_2, PARAMETER_3, ...);
```

SQL TOOL을 활용하여 직접 실행시키는 경우에는 EXECUTE, 줄여서 EXEC 명령어를 실행하여 프로시저를 실행한다. 프로시저에 입출력 변수가 존재하는 경우 데이터 유형 및 MODE에 변수를 입력하여 실행한다. 데이터 유형의 경우 자동 변환이 되어고 별도 오류가 발생하지 않는 경우가 많지만 가급적 프로시저에서 선언한 데이터 타입과 동일하게 입출력 변수를 넣어서 실행시킨다.

SQL TOOL 등을 사용하지 않고 응용 프로그램이나 타 프로시저 등에서 호출 시에는 프로그램에 따라 별도 명령어 없이 “@프로시저명” 또는 프로시저 명칭만 적어서 호출 가능하다.

#### 2. 프로시저 호출문 예제

위에서 제시된 프로시저 예제를 호출하는 경우를 예시로 한다.

<표 1-4> 프로시저 호출 예시

```
SQL> EXECUTE SALES_CLOSING( '20170425' );
```

## 수행 내용 / 프로시저 및 호출문 작성하기

---

### 재료 · 자료

- 논리 E-R 다이어그램
- 물리 E-R 다이어그램
- 테이블 설계서
- 뷰 명세서
- 인덱스 설계서
- 데이터 표준 정의서
- 데이터 요구사항

### 기기(장비 · 공구)

- 컴퓨터
- DBMS(DataBase Management System) 설치 프로그램
- DBMS 클라이언트 프로그램

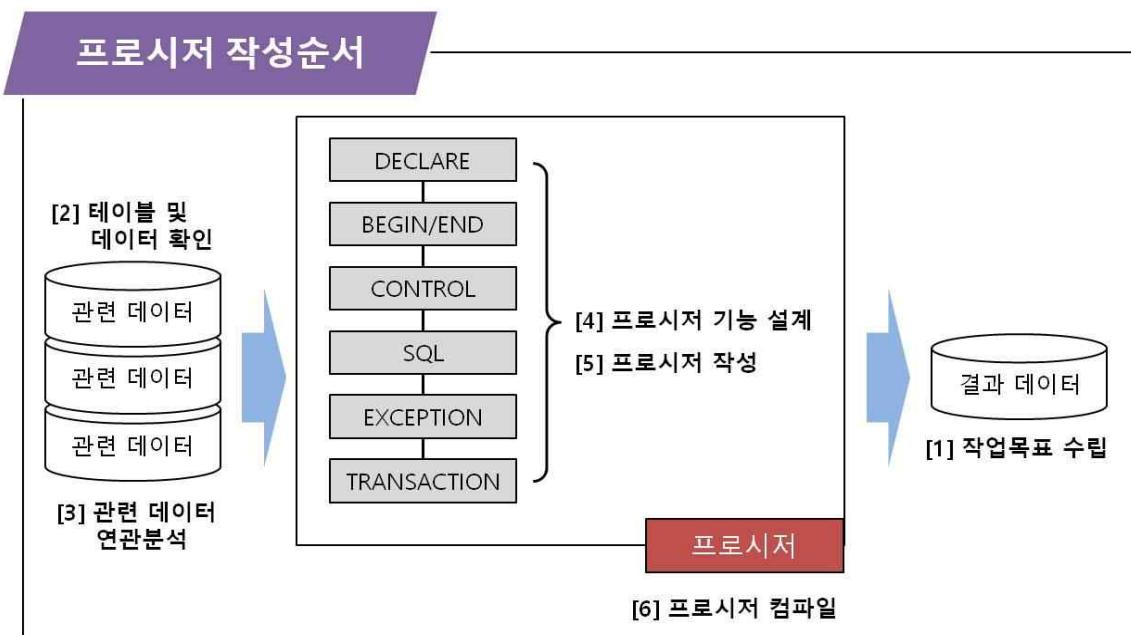
### 안전 · 유의 사항

- 실습 후 사용한 PC의 DBMS 접속 종료

### 수행 순서

#### ① 프로시저를 작성한다.

1. 프로시저 작업 목표를 수립한다.
  - (1) 데이터 작업을 통해 구현하고자 하는 데이터 변경 항목을 정의한다.
  - (2) 단순 데이터 작업이 아닌, 일련의 절차를 통해 구현 가능한지 검토하고, 해당되는 경우 프로시저를 사용하는 것을 결정한다.
  - (3) 프로시저를 통해서 구현하려고 하는 기능이나 일련의 작업에 대한 대략적 개념을 상기한다.



[그림 1-4] 프로시저 작성 순서

2. 생성할 프로시저에 관련되거나 필요한 테이블 및 데이터를 확인한다.

- (1) 프로시저 내에서 구현하고자 하는 결과 데이터 생성을 위한 요소 데이터가 맞는지 확인한다.
- (2) 필요로 하는 모든 데이터를 조사하였는지 확인하고, 추가로 필요한 데이터가 없는지 점검한다.
- (3) 관련된 모든 데이터가 포함된 기존 테이블 및 구조를 확인한다.
- (4) 다수의 테이블로부터 데이터를 검색하는 경우에는 조인(JOIN)이나 집합 명령어 등을 활용할 수 있다.

3. 관련된 기존 테이블 및 데이터 간의 관계를 분석한다.

- (1) 필요로 하는 모든 데이터 조회 및 접근이 가능하도록 한다. 조사한 테이블의 구조 및 주키(PK : Primary Key), 외래키(FK : Foreign Key)를 확인하여 테이블 간의 관계를 정의할 수 있도록 한다.
- (2) 관련된 데이터를 분석한다. 문자열, 코드, 숫자, 날짜 등 데이터 형식도 확인하고 설계 시 반영 가능하도록 누락된 사항이 없는지 점검한다.
- (3) 구현에 필요한 모든 데이터를 취합하였는지 확인하고, 필요시 이전 단계로 돌아가 필요한 데이터 및 관련 테이블을 조사한다. 설계 단계에서 누락된 데이터를 확인하고 작성하는 것도 무방하나, 전체 프로세스 및 데이터 흐름 작성 측면에서 필요로 하는 데이터는 모두 취합하고 설계하는 것이 바람직하다.

#### 4. 프로시저의 기능을 설계한다.

##### (1) 프로시저의 변수를 분석한다.

(가) 프로시저 외부와 연동되어야 하는 입력, 출력변수를 분석한다.

###### 1) 프로시저 외부와 연동되는 입력변수를 분석한다.

여기서 입력변수는 테이블로부터 데이터를 조회하는 것이 아니라, 프로시저 수행 시 외부에서 변수(Parameter)로 입력을 통해 넣는 값을 의미한다. 예를 들면, 일별 판매액 집계 시 대상이 되는 일자 등은 프로시저 수행 시 지정하게 되며, 이 값이 외부의 입력변수이다.

###### 2) 프로시저 외부와 연동되는 출력변수를 분석한다.

프로시저의 출력변수는 사용자 정의함수에서의 결과와는 조금 다르게 프로시저 실행에 대한 결과나 오류코드 및 내용 등의 문장(Comment)을 사용하는 경우가 대다수이다. 필수적인 것은 아니지만 프로시저 실행에 대한 요약이나 결론 등을 나타낼 때 좋으니 참조한다.

(나) 프로시저 내부에서 생성하고 활용되어야 하는 변수를 분석한다.

별도의 변수 선언이나 값 등을 입력하고 계속 사용하는 값을 대상으로 하는 것이 좋다. 모든 값을 내부 변수로 사용할 필요는 없으며, 예를 든다면 자주 사용되는 현재의 시간이나 목표일 등의 값이다.

##### (2) 프로시저의 데이터 흐름도를 작성한다.

(가) 주요 데이터값을 위주로 전체적인 흐름을 구상한다. 가령 프로시저 초반에는 필요로 하는 값을 조회하고 원하는 중간 결과값 등을 산출하여 마지막에는 원하는 데이터를 입력하는 것 정도가 이에 해당한다.

(나) 조건, 반복, 분기 등의 데이터 흐름을 위주로 프로세스를 작성한다. 동일한 결과를 얻는 프로시저를 작성하더라도 어떻게 데이터 흐름을 설계하느냐에 따라 전체적 성능이나 소요시간 등의 차이가 발생할 수 있다. 단순한 구조를 유지하며, 효과적 흐름이 가능하도록 전체 흐름을 설계한다.

(다) 데이터 흐름도 작성은 프로시저 작성 과정 중 가장 핵심적인 사항이다. 단독으로 설계하는 것보다 관련자들과의 논의를 통해 진행하는 것도 추천한다.

##### (3) 예외 처리를 설계한다.

(가) 프로시저 내부에서 사용되는 SQL문의 예외 발생 상황을 설계한다. 데이터 질의어(DQL: Data Query Language)의 경우에는 조회되는 데이터가 없거나 단일 내부 변수값에 다수 결과값을 입력하는 등의 경우가 있을 수 있다.

(나) 발생 가능한 상황별 예외를 정의하고 처리 방안을 구상한다. 해당 프로시저가 실행되며 발생 가능한 상황을 가정하고 사전에 정의함으로써, 실행 중 가능한

에러 상황을 방지하고 처리 방안을 제시한다.

(4) 프로시저 실행 시의 예기치 않은 상황을 고려했는지 확인한다.

- (가) 프로시저 실행이 실행 빈도에 영향을 주지 않도록 작성한다. 예를 들면 하루에 한 번 실행되는 것을 가정하고 작성하더라도 운영환경에서의 예기치 않은 사정에 의해 두 번 이상 실행될 수 있음을 염두에 둔다. 이런 경우 두 번 이상 실행이 되지 않거나, 실행이 되더라도 결과에 영향을 주지 않도록 설계한다.
- (나) 트랜잭션(Transaction)의 위치를 고민한다. 내부적으로 간단한 경우에는 정상 종료 시에만 프로시저 마지막에 반영(COMMIT)하는 것이 가장 좋다는 것을 숙지하고 설계한다. 참고로, 복잡한 프로시저나 내부에서 다루는 데이터의 용량이 매우 큰 경우, 마지막에 반영하기에는 DBMS 메모리에 많은 부담을 주게 된다. 이런 경우는 중간중간 반영을 하고 별도의 DB LOG를 적재하거나 테이블에 작업내역을 삽입하는 경우도 있음을 숙지한다.

5. 프로시저를 작성한다.

(1) 프로시저의 변수를 선언한다.

- (가) 설계 단계에서 작성한 프로시저 외부와 연동되어야 하는 입력, 출력변수를 선언하고 활용한다.
- (나) 설계 단계에서 작성한 프로시저 내부에서 생성하고 활용되어야 하는 변수를 선언하고 활용한다.

(2) 프로시저의 기능 설계를 바탕으로 작성한다.

- (가) 설계한 데이터 프로세스를 중심으로 코드를 작성한다. 가독성 확보 및 블록별 실행 순서 명시화를 위한 줄바꿈(Indentation)에 유의하며 작성한다.
- (나) 조건, 반복, 분기 등 사전에 정의해 놓은 데이터 흐름 설계를 기반으로 코드를 작성한다.

(3) 예외 처리를 구현한다.

- (가) 설계 단계에서 구상한 프로시저 내부에서 사용되는 SQL문의 예외 발생 상황에 서의 코드를 작성한다.
- (나) 발생 가능한 상황별 예외를 정의하고 처리 방안을 작성한다.

(4) 코딩 규칙을 준수하였는지 확인한다.

- (가) 소스 가독성을 높이기 위한 줄바꿈(Indentation)을 엄격히 적용한다. 기본적으로 SQL 문장의 가독성을 확보하기 위해 줄바꿈을 적용한다. 프로시저의 경우는 SQL뿐만 아니라 내부에서의 블록화된 부분 및 조건문, 반복문 등의 다양한 명령어가 사용되므로 특히 더 유의하여 줄바꿈 방식을 적용한다.

- (나) 내외부 변수 사용 시 변수 명명 방법을 준수하였는지 확인한다. 데이터사전(DD : Data Dictionary)을 사용하는 것이 일반적이며, 용어의 통일을 활용하여 혼동을 최소화하여 생산성 저하를 방지할 수 있다.
- (다) 주석문을 가급적 많이 작성한다. 주요변수나 데이터 진행흐름, 로직 등을 설계자나 작성자는 알 수 있지만 타 작업자는 쉽게 이해하지 못하는 경우가 대다수이다. 이런 경우 가독성 증대를 위해 주석문을 적극 활용한다.

## 6. 프로시저를 컴파일한다.

- (1) 프로시저를 컴파일한다. 컴파일하게 되면 DBMS에 반영되고, 그때부터 사용 가능하다.
- (2) 컴파일 과정 중 에러가 발생할 수 있다. 그럴 경우 컴파일 과정에서 발생하는 메시지를 참조하여 원인을 해결한다. 에러의 경우는 매우 다양하며, 컴파일하기 전에 알기가 어려울 수 있기 때문에 에러를 해결하는 과정 중에 필요한 시간이 많이 걸릴 수 있다. 결과 메시지를 최대한 참조하여 에러를 해결하고, 완료되면 재컴파일한다. 대개의 경우 프로시저 컴파일 오류는 문법적 오류로 인해 발생하며, 이 부분에 문제가 없을 경우 컴파일은 정상적으로 가능하다.

## ② 프로시저 호출문을 작성한다.

### 1. 프로시저의 외부 입출력 변수를 확인하고 호출문을 작성한다.

- (1) 입력, 출력 데이터는 프로시저에서 선언된 데이터 유형(Type: 문자열, 숫자 및 날짜 등)과 일치하게 작성한다. 호출 시 에러가 발생하지 않을 수 있지만 DBMS에서 변환하는 과정 등에서 비효율적 동작을 필요로 할 수 있으며 정확한 프로그래밍 작성을 위해 지양하도록 한다.
- (2) 데이터 타입 외 기타 Validation을 고려하여 변수를 작성한다. 예를 들면 8자리의 문자열로 선언된 변수에 10자리의 문자열을 호출한다거나 존재하지 않는 2월 30일의 날짜를 호출하는 경우이다.

### 2. SQL 명령문을 실행하여 프로시저를 호출한다.

- (1) 정상적으로 종료되는 경우 의도한 바와 같이 작업이 되었는지 확인한다.
  - (가) 프로시저 작성 시 수립한 목표대로 작업이 완료되었는지 확인한다. 대개의 경우 프로시저는 데이터 조작어(DML : Data Manipulation Language)를 수반하고 있다는 사실을 상기한다.

- (나) 데이터 질의어(DQL : Data Query Language) 명령어를 사용하여 추가, 수정되거나 삭제된 데이터를 확인한다. 프로시저 실행 후 정상 수행 내역의 확인을 위한 데이터 질의 쿼리를 작성하는 것도 방법이지만, 더 좋은 것은 프로시저를 설계하고 작성한 다음 곧바로 확인이 가능하도록 쿼리를 미리 작성해 놓는 것이다.
- (2) 비정상적으로 종료되는 경우에는 원인을 확인한다.
- (가) 프로시저 실행 시 나타나는 오류메시지를 참조하여 어느 과정에서 어떻게 에러가 발생했는지 확인한다.
- (나) 메시지를 참조하여 프로시저를 수정하거나 호출문을 수정 후 재호출한다.
- 1) 프로시저가 컴파일은 되었으나, 수행 과정 중 에러가 발생할 수 있다. 단순 에러이거나 또는 근본적 문제로 인한 것일 수 있다. 에러가 발생하는 부분을 확인하고 이를 수정한다. 일반적으로 프로시저 호출 과정 중 발생하는 에러의 경우 문제가 되는 프로시저의 줄(Row)과 오류원인을 표현하는 경우가 많다. 이를 참조하여 조치하고 재컴파일한다.
  - 2) 프로시저 자체에는 문제가 없으나, 외부 호출문으로 인해 오류가 발생할 수도 있다. 오류 메시지를 확인하여 외부 호출문이나 호출 변수 등을 수정한다.
  - 3) 프로시저를 재호출한다. 근본적 해결이 되지 않았다면 방금 수정한 문제가 다시 발생할 수 있다. 이런 경우 좀더 분석을 하고 내용을 확인하여 근본적 조치를 취한다. 또한 발생했던 문제가 해결된다 하더라도, 다른 문제가 다시 발생할 수도 있다. 메시지를 참조하여 프로시저를 수정하고 재컴파일하는 과정을 반복한다.

### 수행 tip

- 프로시저는 보통 복합적인 DML 작업의 일괄 처리를 위해 많이 사용하며, 프로시저의 사용 횟수와 처리 되는 데이터의 규모에 비례하여 효율성이 증대된다.
- 소규모 프로시저는 단일 트랜잭션 위주로 개발한다. 이를 위해 정상 완료 시에만 종료 전에 Commit을, 그 밖의 경우에는 Rollback을 하고 종료시키는 것이 권장된다.
- 프로시저는 외부 호출 시의 시기나 빈도 등에 의한 영향이 최소화되도록 프로시저 내부에 실행에 대한 조건을 추가하여 중복 호출 시에도 영향이 없도록 작성하는 것이 좋다.

# 1-2. 사용자 정의함수 및 호출쿼리 작성

## 학습 목표

- 일련의 연산 처리 결과가 단일값으로 반환되는 사용자 정의함수를 작성하고 사용자 정의함수를 호출하는 쿼리를 작성할 수 있다.

## 필요 지식 /

### ① 사용자 정의함수 작성

#### 1. 사용자 정의함수의 개념

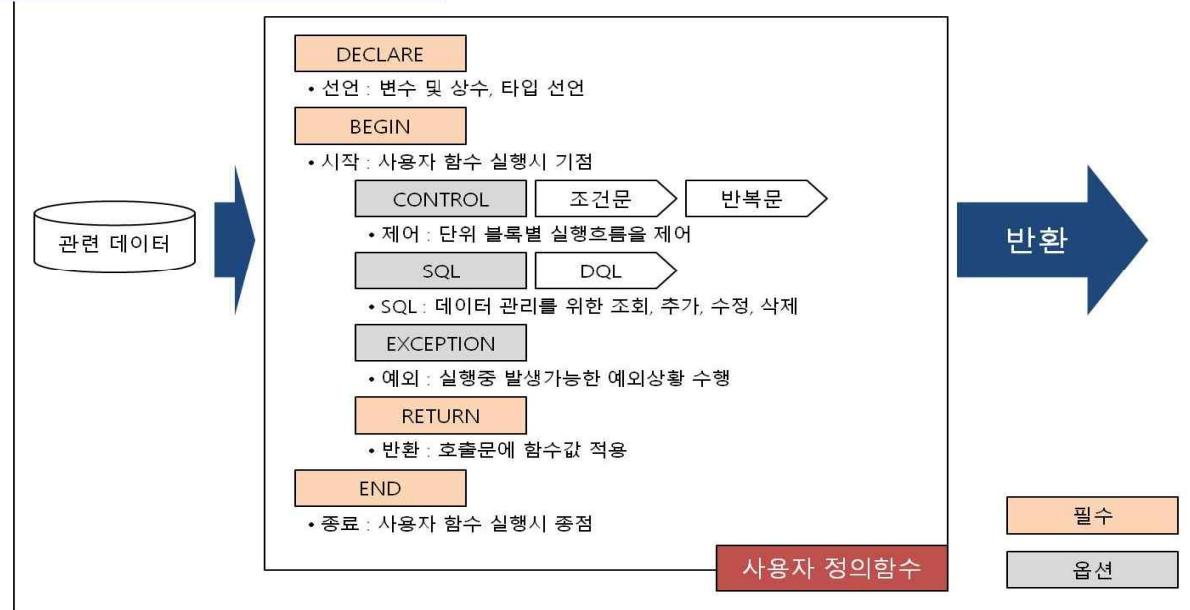
사용자 정의함수는 프로시저와 동일하게 절차형 SQL을 활용하여 일련의 연산 처리 결과를 단일값으로 반환할 수 있는 절차형 SQL이다. DBMS에서 제공되는 공통적 함수 이외에 사용자가 직접 정의하고 작성한다. 사용자 정의함수의 호출을 통해 실행되며, 반환되는 단일값을 조회 또는 삽입, 수정 작업에 이용하는 것이 일반적이다.

기본적인 개념 및 사용법, 문법 등은 상기 언급된 프로시저와 동일하며, 종료 시 단일값을 반환한다는 것이 프로시저와의 가장 큰 차이점이다.

#### 2. 사용자 정의함수의 구성

기본적인 사항은 프로시저와 동일하고 반환에서의 부분만 프로시저와 다르다.

### 사용자정의함수 구성도



[그림 1-5] 사용자 정의함수 구성도

### 3. 사용자 정의함수의 문법

프로시저 문법과 유사하고, 위에서 언급한 바와 같이 결과를 리턴하는 부분이 차이가 있다.

<표 1-5> 사용자 정의함수 문법

```
CREATE [OR REPLACE] FUNCTION [FUNCTION_NAME]
(PARAMETER_1 [MODE] DATA_TYPE_1,
PARAMETER_2 [MODE] DATA_TYPE_2,
...
)
IS [AS]
...
BEGIN
...
[SQL]
[CONTROL]
...
[EXCEPTION]
...
RETURN [VALUE];
END;
```

- CREATE 명령어로 DBMS 내에 사용자 정의함수 생성이 가능하다.
- [OR REPLACE] 명령을 통해 기존 사용자 정의함수 존재 시에 현재 컴파일하는 내용으로 Overwrite할 수 있다. 만약, 기존 동명의 프로시저가 존재하고 CREATE 명령문만 사용하여 컴파일 시에는 에러가 발생한다.
- RETURN 명령을 통해 사용자 정의함수 종료 시 사용자 정의함수를 호출한 쿼리에 반환하는 단일값을 정의한다.

#### 4. 사용자 정의함수의 예제

다음 사용자 정의함수는 생일을 입력받아 나이를 출력하는 간단한 예시이다.

<표 1-6> 사용자 정의함수 예제

```
CREATE OR REPLACE FUNCTION GET_AGE
(V_BIRTH_DATE IN CHAR(8))
IS
BEGIN
    V_CURRENT_YEAR CHAR(4);          -- 현재 연도
    V_BIRTH_YEAR   CHAR(4);          -- 생년
    V_AGE          NUMBER;           -- 나이

    SELECT TO_CHAR(SYSDATE,'YYYY')      -- 현재 일자에서 년도만 추출
          , SUBSTR(V_BIRTH_DATE,1,4)    -- 생년월일에서 앞 4자리 추출
        INTO V_CURRENT_YEAR           -- 현재 연도(4자리)
          , V_BIRTH_YEAR              -- 생년월일 년도(4자리)
    FROM DUAL ;

    SET AGE = TO_NUMBER(V_CURRENT_YEAR) - TO_NUMBER(V_BIRTH_YEAR) + 1;
    -- 현재 연도와 생년의 차에 1을 더함

    RETURN AGE;                    -- 나이를 반환

END;
```

- 생년월일(V\_BIRTH\_DATE)을 입력변수로 선언한다.
- 현재 연도(V\_CURRENT\_YEAR) 변수와 생년(V\_BIRTH\_DATE) 변수를 문자열 4자리로 선언한다.
- 반환할 대상인 나이(V\_AGE) 변수를 숫자 형식으로 선언한다.
- 현재 일자(SYSDATE)를 조회하고 연도 4자리만 파싱하여 현재 연도 변수에 입력한다.
- 생년월일 8자리에서 4자리만 파싱하여 생년 변수에 입력한다.
- 현재 연도와 생년을 숫자 형식으로 변환한 후 두 수의 차에 1을 더하고 나이 변수에 입력한다.
- 구해진 나이값(한국 기준)을 반환한다.

## ② 사용자 정의함수 호출쿼리 작성

작성된 사용자 정의함수는 프로시저와 동일하게 외부에서의 호출을 통해 실행된다. 비교적 단순한 결과값을 구하는 것이 보통이며, 또는 타 시스템에 정보 제공 등 은닉을 통해 캡슐화를 제공하는 데에도 많이 사용된다.

### 1. 사용자 정의함수 호출쿼리 작성 문법

DQL 또는 DML 문장을 활용하여 사용자 정의함수를 호출한다.

<표 1-7> 사용자 정의함수 예제

```
SQL> SELECT [USER_FUNCTION_NAME] (PARAMETER_1, PARAMETER_2...) FROM DUAL;  
  
SQL> UPDATE [TABLE_NAME] SET [COLUMN_NAME] = [USER_FUNCTION_NAME] (PARAMETER_1,  
... ) WHERE ...
```

직접 사용자 정의함수 결과값을 데이터 질의어에 활용하거나(첫번째 예제), 사용자 정의함수 결과값을 데이터 조작어에 직접 적용하게 하여(두번째 예제) 활용 가능하다. 그 밖의 부분은 프로시저의 호출과 유사하다.

### 2. 사용자 정의함수 호출쿼리 작성 예제

위에서 제시된 사용자 정의함수를 호출하는 쿼리를 예시로 한다.

<표 1-8> 사용자 정의함수 예제

```
SQL> SELECT GET_AGE('19900101') FROM DUAL;  
  
SQL> UPDATE EMPLOYEE_INFO_T SET AGE = GET_AGE(BIRTH_DATE) WHERE EMPLOYEE_ID =  
'2017001';
```

첫 번째 사용자 정의함수는 생년월일값(1990년 1월 1일)을 가지고 연령값을 가져오며, 두 번째 사용자 정의함수는 직원 아이디(EMPLOYEE\_ID)값을 활용하여 생일(BIRTH\_DATE) 컬럼 내의 값을 직접 활용하여 연령값에 수정한다.

## 수행 내용 / 사용자 정의함수 및 호출쿼리 작성하기

### 재료 · 자료

- 논리 E-R 다이어그램
- 물리 E-R 다이어그램
- 테이블 설계서
- 뷰 명세서
- 인덱스 설계서
- 데이터 표준 정의서
- 데이터 요구사항

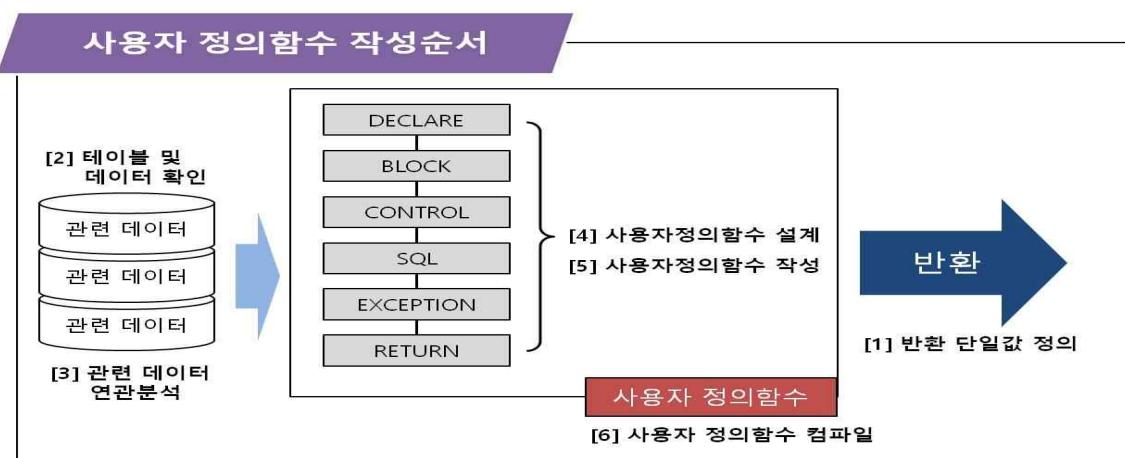
### 기기(장비 · 공구)

- 컴퓨터
- DBMS(DataBase Management System) 설치 프로그램
- DBMS 클라이언트 프로그램

### 안전 · 유의 사항

- 실습 후 사용한 PC의 접속 종료

### 수행 순서



[그림 1-6] 사용자 정의함수 작성 순서

① 사용자 정의함수를 작성한다.

1. 사용자 정의함수 반환값을 정의한다.

(1) 작성하려고 하는 사용자 정의함수가 반환할 단일값을 정의한다.

(2) 일련의 연산 처리 기능을 통해 구현하고자 하는 절차의 전체적 개념을 구상한다.

2. 생성할 사용자 정의함수와 관련된 기존 테이블 및 데이터를 확인한다.

(1) 사용자 정의함수 내에서 구현하고자 하는 결과 데이터 생성을 위한 요소 데이터가 맞는지 확인한다.

(2) 필요로 하는 모든 데이터를 조사하였는지 확인하고, 추가로 필요한 데이터가 없는지 확인한다.

(3) 관련 데이터가 포함된 기존 테이블 및 구조를 확인한다.

3. 관련된 기존 테이블 및 데이터 간의 관계를 분석한다.

(1) 필요로 하는 모든 데이터의 조회 및 접근이 가능하도록 한다. 조사한 테이블의 구조 및 주키(PK : Primary Key), 외래키(FK : Foreign Key)를 확인하여 테이블 간의 관계를 정의할 수 있도록 한다.

(2) 관련된 데이터를 분석한다. 문자열, 코드, 숫자, 날짜 등 데이터 유형도 확인하고 설계 시 반영 가능하도록 누락된 사항이 없는지 점검한다.

(3) 구현에 필요한 모든 데이터를 취합하였는지 확인하고, 필요시 이전 단계로 돌아가 필요한 데이터 및 관련 테이블을 조사한다. 설계 단계에서 누락된 데이터를 확인하고 작성하는 것도 무방하나, 전체 프로세스 및 데이터 흐름 작성 측면에서 필요로 하는 데이터는 모두 취합하고 설계하는 것이 바람직하다.

4. 사용자 정의함수의 기능을 설계한다.

(1) 사용자 정의함수의 변수를 분석한다.

(가) 사용자 정의함수 외부와 연동되어야 하는 입력변수를 분석한다.

여기서 입력변수는 테이블로부터 데이터를 조회하는 것이 아니라, 사용자 정의함수 수행 시 외부에서 변수(Parameter)로 입력을 통해 넣는 값을 의미한다. 프로시저와 동일하다. 분석에 그치지 않고 검증(Validation)까지 고려하는 것이 가장 좋다. 예를 들어, 비율을 입력받는 사용자 정의함수에서는 150의 숫자가 들어오게 되면 ‘올바르지 않은 입력 값입니다’ 등의 값을 표시하고(0부터 100까지의 수를 입력받아야 하기 때문에) 바로 에러코드나 기타 문제 발생 시 반환할 값을 반환하는 경우이다.

(나) 프로시저 내부에서 생성하고 활용되어야 하는 변수를 분석한다.

별도의 변수 선언이나 값 등을 입력하고 계속 사용하는 값을 대상으로 하는 것이 좋다. 모든 값을 내부 변수로 사용할 필요는 없으며, 예를 든다면 자주 사용되는 현재의 시간이나 목표일자 등의 값이다. 프로시저와 동일하다.

(2) 프로시저의 데이터 흐름도를 작성한다.

(가) 주요 데이터값을 위주로 전체적인 흐름을 구상한다.

사용자 정의함수 초반에는 필요로 하는 값을 조회하고 원하는 중간 결과값 등을 산출하여 마지막에는 원하는 데이터를 조회하는 것 정도가 이에 해당한다.

(나) 조건, 반복, 분기 등의 데이터 흐름을 위주로 프로세스를 작성한다.

동일한 결과를 얻는 함수를 작성하더라도 어떻게 데이터 흐름을 설계하느냐에 따라 전체적 성능이나 소요시간 등이 차이가 발생할 수 있다. 단순한 구조를 유지하며, 효과적 흐름이 가능하도록 전체 흐름을 설계한다.

(3) 예외 처리를 설계한다.

(가) 사용자 정의함수 내부에서 사용되는 SQL문의 예외 발생 상황을 설계한다.

데이터 질의어(DQL: Data Query Language)의 경우에는 조회되는 데이터가 없거나 단일 내부 변수값에 다수 결과값을 입력하는 등의 경우가 있을 수 있다.

(나) 발생 가능한 상황별 예외를 정의하고 처리 방안을 구상한다.

해당 사용자 정의함수가 실행되며 발생 가능한 상황을 가정하고 사전에 정의함으로써 실행 중 가능한 여러 상황을 방지하고 처리 방안을 제시한다.

(4) 내부에 데이터 조작어(DML: Data Manipulation Language) 사용을 지양한다.

사용자 정의함수는 단일 반환값을 계산하는 데 목적을 둔다. 내부에서 데이터 삽입, 수정, 삭제를 할 필요가 있다면 함수가 아니라 프로시저를 사용하는 것을 고려하여 대체한다. 일반적으로 사용자 정의함수를 호출할 때 별도의 DML 작업이 없는 것으로 이해하고 호출한다는 것을 감안하여 작성 목적에 맞게 작성한다. 이는 일반적 절차형 SQL을 사용하는 작업자들 간의 통념이다.

## 5. 사용자 정의함수를 작성한다.

(1) 사용자 정의함수의 변수를 선언한다.

(가) 사용자 정의함수 외부와 연동되어야 하는 입력변수를 선언 및 활용한다.

(나) 사용자 정의함수 내부에서 생성하고 활용되어야 하는 변수를 선언 및 활용한다.

(2) 사용자 정의함수의 기능 설계를 바탕으로 코드를 작성한다.

(가) 설계한 데이터 프로세스를 중심으로 코드를 작성한다. 가독성 확보 및 추후 수정 등의 과정에 대비하여 줄바꿈에 유의하며 작성한다.

- (나) 조건, 반복, 분기 등의 데이터 흐름을 기반으로 코드를 작성한다.
- (3) 예외 처리를 구현한다.
- (가) 설계한 내용을 바탕으로 사용자 정의함수 내부에서 사용되는 SQL문의 예외 발생 상황의 코드를 작성한다.
- (나) 발생 가능한 상황별 예외를 정의하고 처리 방안을 작성한다.
- (4) 코딩 규칙을 준수하였는지 확인한다.
- 소스 가독성을 높이기 위한 줄바꿈이나内外부 변수 사용 시의 변수 명명 방법, 주석 문 추가 등을 준수하였는지 확인한다. 프로시저의 경우와 같다.
6. 사용자 정의함수를 컴파일한다.
- (1) 사용자 정의함수를 컴파일한다. 컴파일 이후 DBMS에 반영되고 사용 가능하다. 프로시저와 같다.
- (2) 에러가 발생하는 경우, 메시지를 참조하여 원인 해결 후 재컴파일한다. 프로시저와 동일하게 발생하는 메시지를 참조하여 원인을 해결한다.

## ② 사용자 정의함수를 호출하기 위한 쿼리를 작성한다.

1. 프로시저의 외부 입출력 변수를 확인하고 호출쿼리를 작성한다.
  - (1) 입력, 출력 데이터는 사용자 정의함수에서 선언된 데이터 유형(Type: 문자열, 숫자 및 날짜 등)과 일치하게 작성한다. 호출 시 에러가 발생하지 않을 수 있으나 수행 시의 효율적 작성과 명확한 정의를 통한 프로그래밍 습관 배양을 위해 정확한 내용을 입력하도록 한다.
  - (2) 데이터 타입 외 기타 Validation을 고려한 변수를 작성한다. 예를 들어 최대자릿수를 넘어가는 문자열 변수를 호출한다거나 비율 숫자를 입력하는 부분에 200 등의 숫자를 입력하는 경우이다.
2. SQL 명령문을 실행하여 사용자 정의함수를 호출한다.

- (1) 호출쿼리가 정상적으로 실행되는 경우
- 데이터 질의어(DQL: Data Query Language) 명령어의 결과값을 사용하여 사용자 정의 함수 반환값을 확인하는데, 의도한 바와 같은 결과가 도출되는지를 확인한다. 에러는 없었다고 하더라도 의도하지 않는 결과가 나오는 경우 사용자 정의함수 내부 및 호출 과정 등을 확인하여 필요한 부분을 수정한다. 일부 DB TOOL의 경우 디버깅모드(Debugging Mode)를 제공하는데, 이를 활용하면 좀더 쉽게 확인 가능하다.

## (2) 호출쿼리 실행 시 에러가 발생하는 경우

### (가) 사용자 정의함수 실행 시 나타나는 메시지를 참조한다.

에러가 발생하는 부분을 확인한다. 프로시저와 동일하게 사용자 정의함수도 호출 과정 중 발생하는 에러의 경우 문제가 되는 사용자 정의함수의 줄(Row)과 에러 사유를 표현한다. 이를 참조하도록 한다. 사용자 정의함수에서 가장 빈번한 오류는 반환 결과값이 없거나 2개 이상인 경우이다. 단일 값을 반환한다는 사실에 유의한다.

### (나) 메시지를 참조하여 사용자 정의함수를 수정하거나 호출문을 수정 후 재호출한다.

사용자 정의함수 내의 로직 또는 다른 부분이 잘못되었는지 확인하고 재컴파일한다. 아닐 경우 호출쿼리에서 문제가 있을 수 있으므로 그 부분을 수정하도록 한다. 수정이 완료되면 재호출하고, 결과를 확인하여 의도하는 결과를 최종적으로 얻을 수 있도록 조치한다.

## 수행 tip

- 사용자 정의함수는 단순한 단일값의 반환 과정의 단순화를 위해 많이 사용된다.
- 조인되는 테이블이 2~3개 정도 수준이고 다수 절차를 통한 단일값 조회 수준의 자주 사용되는 SQL을 대체하기 위해 사용자 정의함수를 작성하고 활용하는 경우가 많다.
- 사용자 정의함수는 결과로 표시되는 ROW 수에 비례하여 내부적으로 실행된다. 실행결과가 많은 경우 사용자 정의함수를 남용하게 되면 전체 성능 측면에서는 비효율적일 수 있으므로 호출쿼리의 특성을 사전에 파악하고 적용하도록 한다.
- 사용자 정의함수 내에서도 DML 문장 수행이 가능하기는 하나 대개의 경우 DQL 문장만 사용하여 결과값을 반환시킨다.

# 1-3. 트리거 작성

## 학습 목표

- 하나의 이벤트가 발생하면 관련성이 있는 몇 개의 테이블 간에 연속적으로 데이터 삽입, 삭제, 수정을 할 수 있는 트리거를 작성할 수 있다.

## 필요 지식 /

### ① 트리거 작성

#### 1. 트리거의 개념

특정 테이블에 삽입, 수정, 삭제 등의 데이터 변경 이벤트가 발생하면 DBMS에서 자동적으로 실행되도록 구현된 프로그램을 트리거라고 한다. 이벤트는 전체 트랜잭션 대상과 각 행에 의해 발생되는 경우 모두를 포함할 수 있으며 테이블과 뷰(View), DB 작업을 대상으로 정의할 수 있다.

#### 2. 트리거의 목적

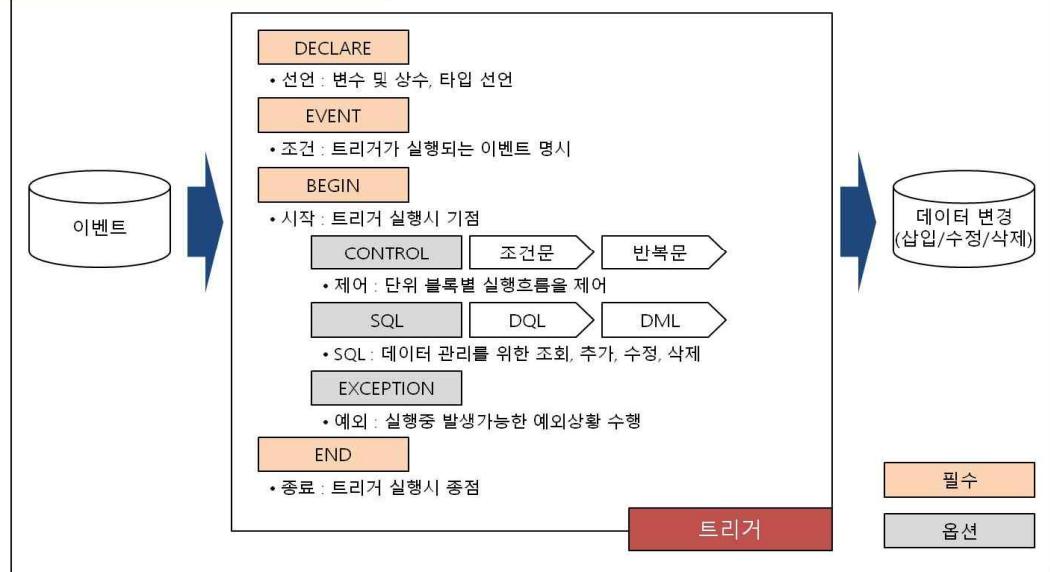
특정 테이블에 대한 데이터 변경을 시작점으로 설정하고, 그와 관련된 작업을 자동적으로 수행하기 위해 트리거를 사용한다. 일반적으로 이벤트와 관련된 테이블의 데이터 삽입, 추가, 삭제 작업을 DBMS가 자동적으로 실행시키는 데 활용되나, 데이터 무결성 유지 및 로그 메시지 출력 등의 별도 처리를 위해 트리거를 사용하기도 한다.

#### 3. 트리거의 구성

앞에서 설명한 프로시저나 사용자 정의함수와 기본적 문법은 같다. 일반적인 트리거의 구성도는 [그림 1-7]과 같다.

반환이 없다는 점, DML을 주된 목적으로 한다는 점에서는 프로시저와 유사하나, EVENT 명령어를 통해 트리거 실행을 위한 이벤트를 인지한다는 점과 외부 변수 IN, OUT이 없다는 점이 프로시저나 사용자 정의함수와 다르다.

## 트리거 구성도



[그림 1-7] 트리거 구성도

### 4. 트리거의 문법

위에서 나온 트리거의 구성을 기반으로 하는 트리거의 문법은 다음과 같다.

<표 1-9> 트리거 문법

```

CREATE [OR REPLACE] TRIGGER [TRIGGER_NAME]
AFTER [TRANSACTION TYPE]
ON [TABLE NAME]
[FOR EACH ROW]
BEGIN

...
[SQL]
[CONTROL]

...
[EXCEPTION]

...
END;
    
```

대상 테이블(TABLE\_NAME)에 벌어진 이벤트 유형(TRANSACTION TYPE, INSERT / UPDATE / DELETE) 및 이벤트 순서(BEFORE / AFTER)에 맞게 트리거 수행을 위한 조건을 입력한다. 매번 변경되는 데이터 행의 수만큼 실행을 위한 명령어(FOR EACH ROW)를 정의하기도 한다. 트리거에는 레코드 구조체라는 개념이 존재하는데, 대상 테이블의 데이터 변경을 이벤트로 처리하기 때문에 변경 전후의 데이터값을 각각 구분할 때 사용된다. P언어에서는

:OLD, :NEW로 사용되며, T언어에서는 deleted와 inserted로 사용된다. 각각 이전 · 이후 데이터를 의미하며, INSERT 시에는 이전 값은 NULL, DELETE 시에는 이후 값이 NULL을 의미한다.

## 5. 트리거 작성의 예제

다음은 직원 정보 변경 시 변경된 데이터를 이력 테이블에 적재하는 간단한 트리거의 예시이다.

<표 1-10> 트리거 예제

```
CREATE OR REPLACE TRIGGER PUT_EMPLOYEE_INFO_HISTORY -- 직원 정보 이력 입력
AFTER UPDATE -- 수정 후
ON EMPLOYEE_INFO_T -- 직원 정보 테이블 대상
FOR EACH ROW -- ROW 단위 건별

BEGIN
    INSERT INTO EMPLOYEE_INFO_H -- 직원 정보 이력 테이블
    (
        EMPLOYEE_ID,
        SEQ_VAL,
        EMPLOYEE_NAME,
        EMPLOYEE_DEPT,
        EMPLOYEE_ADDRESS,
        EMPLOYEE_CP_NO
    )
    (
        :OLD.EMPLOYEE_ID --기존 ID
        , SEQ_VAL.NEXT_VAL --이력 순번
        , :NEW.EMPLOYEE_NAME --신규 성명
        , :NEW.EMPLOYEE_DEPT --신규 부서
        , :NEW.EMPLOYEE_ADDRESS --신규 주소
        , :NEW.EMPLOYEE_CP_NO --신규 핸드폰 번호
    );
END;
```

- 직원 정보(EMPLOYEE\_INFO\_T) 테이블 수정 후(AFTER UPDATE) 실행된다.
- 직원 정보 이력(EMPLOYEE\_INFO\_H) 테이블에 데이터를 삽입(INSERT)한다.
- 기존 직원 ID(:OLD.EMPLOYEE\_ID) 기준이며, 기타 직원 정보는 변경된 정보(:NEW.)를 삽입한다.

## 6. 트리거 작성 시 주의 사항

### (1) 데이터 제어어(DCL, Data Control Language) 사용 불가

트리거 내에는 COMMIT, ROLLBACK 등의 DCL을 사용할 수 없다. 쉽게 말하면 Auto Commit을 하는 결과와 같다고 할 수 있다. 트리거 내에 COMMIT이나 ROLLBACK을 사용하고 컴파일하면 에러가 나는 것을 볼 수 있다. 여기서 하나 더 유의할 부분은 트리거로 인해 발생하는 모든 후속 절차에도 동일하게 적용된다는 것이다. 트리거 내에서 타 프로시저를 호출하는 것도 가능한데, 해당 프로시저 내에 COMMIT이 포함되어 있으면 이로 인해 오류가 유발된다.

### (2) 오류에 특히 주의할 것

트리거 실행 중 오류가 발생하게 되면 트리거 실행의 원인을 제공한 데이터 작업에도 영향을 주는 경우가 많다. 즉, 특정 테이블에 데이터를 추가한 후 발생하는 트리거에서 오류가 발생할 경우에는 트리거 이후의 작업이 진행되지 않는 것 뿐만 아니라, 더 나아가 데이터가 추가되지 않는다. 결국 트랜잭션을 합치는 과정에서 더 높은 무결성 및 품질을 요구한다고 볼 수 있으므로 이런 부분에서 주의를 요한다.

## 수행 내용 / 트리거 작성하기

---

### 재료 · 자료

- 논리 E-R 다이어그램
- 물리 E-R 다이어그램
- 테이블 설계서
- 뷰 명세서
- 인덱스 설계서
- 데이터 표준 정의서
- 데이터 요구사항

### 기기(장비 · 공구)

- 컴퓨터
- DBMS(DataBase Management System) 설치 프로그램
- DBMS 클라이언트 프로그램

### 안전 · 유의 사항

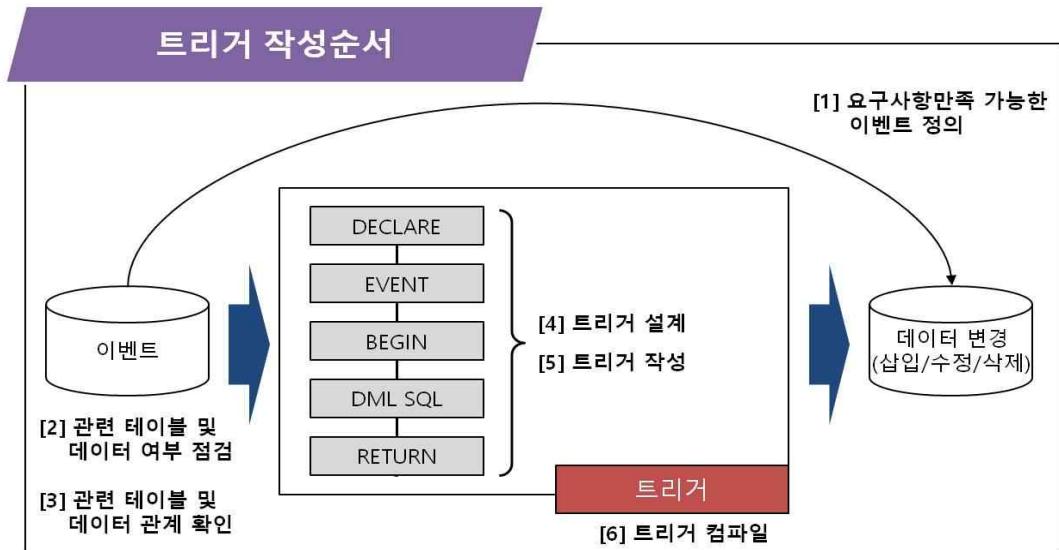
- 실습 후 사용한 PC의 접속 종료

### 수행 순서

#### ① 트리거를 작성한다.

1. 인식이 가능한 이벤트를 정의한다.
  - (1) 요구사항 만족을 위한 데이터의 트랜잭션을 확인한다.
  - (2) 해당 데이터의 트랜잭션을 인식할 수 있는 이벤트가 가능한지 확인한다.
2. 대상 이벤트와 관련된 테이블 및 데이터를 확인한다.
  - (1) 관련된 데이터 및 SQL 명령문을 통해 추가, 수정, 삭제하고자 하는 데이터를 확인한다.

(2) 관련 데이터가 포함된 기존 테이블 및 구조를 확인한다.



[그림 1-8] 트리거 작성 순서

3. 기존 테이블 및 데이터 간의 관계를 분석한다.

- (1) 관련 테이블 간의 상호 구조 및 PK(Primary Key), FK(Foreign Key)를 확인한다.
- (2) 결과값 반환을 위한 관련 데이터를 분석한다.

4. 트리거의 기능을 설계한다.

- (1) 트리거의 데이터 흐름도를 작성한다.
  - 주요 데이터 속성을 위주로 전체적 흐름을 구상한다.
  - 조건, 반복, 분기 등의 데이터 흐름을 위주로 프로세스를 작성한다.
- (2) 트리거의 변수를 분석한다.
  - 트리거 내부에서 생성하고 활용되어야 하는 변수를 분석한다.
  - 이벤트를 통한 변경 전후 데이터를 구분한다.  
:NEW, :OLD 등의 구분을 통한 변경 전후 데이터를 구분하는 부분이 가장 실수하기 쉬운 부분이다. 데이터 인과 관계를 통한 구성을 고려하고 작성 시 주의한다.
- (3) 예외 처리를 설계한다.
  - 트리거 내부에서 사용되는 SQL문의 예외 발생 상황을 설계한다. 예외 상황은 앞에 나온 프로시저나 사용자 정의함수와 같다.
  - 발생 가능한 상황별 예외를 정의하고 처리 방안을 구상한다.

5. 트리거를 작성한다.

(1) 트리거의 변수를 선언한다.

- (가) 트리거 내부에서 생성하고 활용되어야 하는 변수를 선언하고 활용한다.
- (나) 이벤트를 통해 변경되기 전의 데이터와 변경된 데이터를 구분하여 변수를 활용하는 부분에 주의한다.

(2) 트리거의 기능 설계를 바탕으로 작성한다.

- (가) 설계한 데이터 프로세스를 중심으로 코드를 작성한다.
- (나) 조건, 반복, 분기 등의 데이터 흐름을 기반으로 코드를 작성한다.

(3) 예외 처리를 구현한다.

- (가) 트리거 내부에서 사용되는 SQL문의 예외 발생 상황의 코드를 작성한다.
- (나) 발생 가능한 상황별 예외를 정의하고 처리 방안을 작성한다.

6. 트리거를 컴파일한다.

(1) 트리거를 컴파일하여 DBMS에 반영한다.

(2) 에러가 발생하는 경우, 메시지를 참조하여 원인 해결 후 재컴파일한다.

② 트리거 실행을 위한 이벤트를 발생시킨다.

1. 테이블 내에 데이터 삽입, 수정, 삭제 등 사전에 정의한 이벤트를 발생시킨다.

2. 트리거가 자동 실행된다.

(1) 이벤트가 정상 완료되는 경우이다. 정상 완료되는 경우 별도의 메시지가 없다.

(가) 트리거가 실행되었는지 확인한다. 해당 트리거가 Valid 상태인지 확인한다.

(나) 해당 트리거로 인한 데이터 삽입, 수정, 삭제가 되었는지 확인한다.

데이터 질의어(DQL: Data Query Language)를 통해 의도한 데이터 변경이 수반되었는지 확인한다.

(2) 이벤트가 비정상 완료되는 경우이다. 메시지가 발생하므로 이를 참조한다.

(가) 트리거와는 무관한 에러가 발생하는 경우

이벤트가 정의된 테이블에 삽입, 수정, 삭제 등의 데이터 변경 처리 자체에서 에러가 발생한 경우이다. 메시지를 참조하면 해당 테이블에서의 데이터 변경 처리 과정 중 문제가 생긴 것을 알 수 있다. 트리거로 인한 것은 아니지만 정확한 테스트를 할 수 없으므로, 에러의 원인을 찾아서 해결한 후 다시 이벤트를 실행시킨다.

#### (나) 트리거로 인한 에러가 발생하는 경우

해당 트리거명과 트리거 내의 줄(Row)수 및 에러 내용을 메시지에서 확인할 수 있다. 트리거는 호출쿼리가 별도로 없으므로 대개의 경우 트리거 내부의 문법이나 로직에서 오류가 발생할 가능성이 높다. 원인을 파악하고 수정하여 다시 컴파일한다. 컴파일 후에는 다시 이벤트를 실행시킨다.

#### 수행 tip

- 트리거에 관련된 이벤트는 대부분의 경우 사전에 정해 놓은 테이블 내에 데이터가 삽입, 수정, 삭제 등 변경되는 경우를 많이 선정한다.
- 대상 테이블 데이터 변경 시 자동적으로 변경 이력을 적재 또는 로그 데이터를 적재하기 위해서 트리거를 많이 사용하기도 하나, 트리거에서 오류가 발생하는 경우 관련된 이벤트 자체가 미수행되므로, 트리거를 사용할 경우에는 이로 인한 오류나 기타 영향이 없도록 유의해야 한다.
- 상기 특징을 이용하여 입력되는 데이터값의 무결성 유지를 위해 트리거를 사용하기도 한다.
- 트리거 실행 후 이로 인한 다른 트리거가 실행될 수 있는데, 이런 경우 순환 오류 등이 발생 가능하므로 가급적 단일 연결 수준의 트리거만 사용하도록 한다.

## 학습 1 교수 · 학습 방법

### 교수 방법

- SQL에 대한 학습자의 이해 및 활용 수준을 확인하고, 주요 사항 및 TIP 위주로 설명한다.
- 일관성 유지, 객관성 확보, 효율성 제고 관점에서의 Data Flow 방안을 제시한다.
- 제어문, 조건문 등의 명령문과 데이터 질의어(DQL: Data Query Language), 데이터 조작어 (DML: Data Manipulation Language) 간 상호 연계하여 활용할 수 있도록 교육한다.
- Exception 등 예외 처리의 중요성을 주지시키고 소규모 프로시저인 경우 단일 처리(One Transaction)를 유지할 수 있도록 설명한다. 쉽게 말하면 중간에 에러 발생 시 룰백 (Rollback), 최종 완료 시에 커밋(Commit)하도록 한다.
- 프로시저 사용을 통한 일괄 처리 작업의 유용성 제시 및 사용 예를 설명하고, 규모 및 반복 횟수에 비례하여 전체 효용성이 증대한다는 사실을 설명한다.
- 프로시저의 진행 상태를 호출 시에도 인지할 수 있도록 Message Output 처리 또는 로그 테이블에 데이터를 적재하는 방법에 대해서도 설명한다. 로그 테이블에 데이터 적재 시에는 단일 처리(One Transaction)를 유지하지 않도록 설명한다.
- 사용자 정의함수를 통해 내부 연산 처리와 상관없이 외부로 반환되는 결과값을 통해 쉽게 활용이 가능하다는 사실을 설명한다.
- 트리거 사용을 통해 DBMS에서의 변경 이벤트를 인지하여 데이터 삽입, 수정, 삭제가 가능 함을 설명한다.

### 학습 방법

- SQL에 대한 이해 및 활용 수준을 상기하고, 주요 사항 및 TIP 위주로 숙지한다.
- 일관성 유지, 객관성 확보, 효율성 제고 관점에서의 Data Flow 방안을 구성한다.
- 제어문, 조건문 등의 명령문과 데이터 질의어(DQL: Data Query Language), 데이터 조작어 (DML: Data Manipulation Language) 간 상호 연계하여 활용할 수 있도록 구성한다.

- Exception 등 예외 처리의 중요성을 인식하고 소규모인 경우 One Transaction을 유지할 수 있도록 구성한다.
- 프로시저 사용을 통한 일괄 처리 작업의 유용성 제시 및 사용 예를 이해하고, 규모 및 반복 횟수에 비례하여 전체 효용성이 증대한다는 사실을 숙지한다.
- 프로시저의 진행 상태를 외부 호출 시에도 인지할 수 있도록 Message Output 처리 또는 로그 테이블에 데이터를 적재하는 방법에 대해서도 숙지한다.
- 사용자 정의함수를 통해 내부 연산 처리와 상관없이 외부로 반환되는 결과값을 통해 쉽게 활용이 가능하다는 사실을 숙지한다.
- 트리거 사용을 통해 DBMS에서의 변경 이벤트를 인지하여 데이터 삽입, 수정, 삭제가 가능함을 숙지한다.

## 학습 1 평 가

### 평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	학습 목표	성취수준		
		상	중	하
프로시저 및 호출문 작성	- 반복적으로 사용하는 특정 기능을 수행하기 위해 여러 개의 SQL 명령문을 포함하는 프로시저를 작성하고 프로시저 호출문을 작성할 수 있다.			
사용자 정의함수 및 호출쿼리 작성	- 일련의 연산 처리 결과가 단일값으로 반환되는 사용자 정의함수를 작성하고 사용자 정의함수를 호출하는 쿼리를 작성할 수 있다.			
트리거 작성	- 하나의 이벤트가 발생하면 관련성이 있는 몇 개의 테이블 간에 연속적으로 데이터 삽입, 삭제, 수정을 할 수 있는 트리거를 작성할 수 있다.			

### 평가 방법

- 문제 해결 시나리오

학습 내용	평가 항목	성취수준		
		상	중	하
프로시저 및 호출문 작성	- 절차형 SQL을 통해 달성하고자 하는 목표 수립을 위해 프로그램을 효율적으로, 합리적으로 작성할 수 있는 능력			
	- 내부 데이터 프로세스 수립 시 단순하게, 신뢰성을 유지할 수 있게, 유지 보수성을 확보할 수 있게 작성할 수 있는 능력			
	- 줄바꿈(Indentation)을 엄격히 적용하고 규정된 용어를 사용하며 필수적 사항에 대한 주석을 추가하여 가독성이 확보된 프로시저를 작성할 수 있는 능력			
	- 프로시저 호출을 위한 호출문 작성 시 단순하고 합리적으로 작성할 수 있는 능력			

사용자 정의함수 및 호출쿼리 작성	<ul style="list-style-type: none"> <li>- 절차형 SQL을 통해 달성하고자 하는 목표 수립을 위해 프로그램을 효율적으로, 합리적으로 작성할 수 있는 능력</li> <li>- 내부 데이터 프로세스 수립 시 단순하게, 신뢰성을 유지할 수 있게, 유지 보수성을 확보할 수 있게 작성할 수 있는 능력</li> <li>- 줄바꿈(Indentation)을 엄격히 적용하고 규정된 용어를 사용하며 필수적 사항에 대한 주석을 추가하여 가독성이 확보된 사용자 정의함수를 작성할 수 있는 능력</li> <li>- 사용자 정의함수 호출을 위한 호출문 작성 시 단순하고 합리적으로 작성할 수 있는 능력</li> </ul>			
	<ul style="list-style-type: none"> <li>- 절차형 SQL을 통해 달성하고자 하는 목표 수립을 위해 프로그램을 효율적으로, 합리적으로 작성할 수 있는 능력</li> <li>- 내부 데이터 프로세스 수립 시 단순하게, 신뢰성을 유지할 수 있게, 유지 보수성을 확보할 수 있게 작성할 수 있는 능력</li> <li>- 줄바꿈(Indentation)을 엄격히 적용하고 규정된 용어를 사용하며 필수적 사항에 대한 주석을 추가하여 가독성이 확보된 프로시저를 작성할 수 있는 능력</li> </ul>			
	<ul style="list-style-type: none"> <li>- 절차형 SQL을 통해 달성하고자 하는 목표 수립을 위해 프로그램을 효율적으로, 합리적으로 작성할 수 있는 능력</li> <li>- 내부 데이터 프로세스 수립 시 단순하게, 신뢰성을 유지할 수 있게, 유지 보수성을 확보할 수 있게 작성할 수 있는 능력</li> <li>- 줄바꿈(Indentation)을 엄격히 적용하고 규정된 용어를 사용하며 필수적 사항에 대한 주석을 추가하여 가독성이 확보된 프로시저를 작성할 수 있는 능력</li> </ul>			
	<ul style="list-style-type: none"> <li>- 절차형 SQL을 통해 달성하고자 하는 목표 수립을 위해 프로그램을 효율적으로, 합리적으로 작성할 수 있는 능력</li> <li>- 내부 데이터 프로세스 수립 시 단순하게, 신뢰성을 유지할 수 있게, 유지 보수성을 확보할 수 있게 작성할 수 있는 능력</li> <li>- 줄바꿈(Indentation)을 엄격히 적용하고 규정된 용어를 사용하며 필수적 사항에 대한 주석을 추가하여 가독성이 확보된 프로시저를 작성할 수 있는 능력</li> </ul>			
트리거 작성	<ul style="list-style-type: none"> <li>- 절차형 SQL을 통해 달성하고자 하는 목표 수립을 위해 프로그램을 효율적으로, 합리적으로 작성할 수 있는 능력</li> <li>- 내부 데이터 프로세스 수립 시 단순하게, 신뢰성을 유지할 수 있게, 유지 보수성을 확보할 수 있게 작성할 수 있는 능력</li> <li>- 줄바꿈(Indentation)을 엄격히 적용하고 규정된 용어를 사용하며 필수적 사항에 대한 주석을 추가하여 가독성이 확보된 프로시저를 작성할 수 있는 능력</li> </ul>			
	<ul style="list-style-type: none"> <li>- 절차형 SQL을 통해 달성하고자 하는 목표 수립을 위해 프로그램을 효율적으로, 합리적으로 작성할 수 있는 능력</li> <li>- 내부 데이터 프로세스 수립 시 단순하게, 신뢰성을 유지할 수 있게, 유지 보수성을 확보할 수 있게 작성할 수 있는 능력</li> <li>- 줄바꿈(Indentation)을 엄격히 적용하고 규정된 용어를 사용하며 필수적 사항에 대한 주석을 추가하여 가독성이 확보된 프로시저를 작성할 수 있는 능력</li> </ul>			

• 평가자 질문

학습 내용	평가 항목	성취수준		
		상	중	하
프로시저 및 호출문 작성	- 프로시저의 Input, Output, Process 설명 능력			
	- 프로시저의 Data Flow의 합리적 설계 방안 답변 능력			
	- 프로시저 컴파일에서 오류 발생 시 대처 방안 답변 능력			
	- 프로시저 호출 과정에서 에러 발생 시 대처 방안 답변 능력			
	- 프로시저 실행 후 합리적 결과 검증 방안 제시 능력			
사용자 정의함수 및 호출쿼리 작성	- 사용자 정의함수의 Input, Return, Process 설명 능력			
	- 사용자 정의함수의 Data Flow의 합리적 설계 방안 답변 능력			
	- 사용자 정의함수 컴파일에서 오류 발생 시 대처 방안 답변 능력			
	- 사용자 정의함수 호출 과정에서 에러 발생 시 대처 방안 답변 능력			

트리거 작성	- 트리거 작성 시 합리적 이벤트 선정 설명 능력		
	- 트리거 Data Flow의 합리적 설계 방안 답변 능력		
	- 트리거 컴파일에서 오류 발생 시 대처 방안 답변 능력		
	- 트리거 이벤트 발생 시, 트리거 실행 중 에러 발생 시 대처 방안 답변 능력		

- 평가자 체크리스트

학습 내용	평가 항목	성취수준		
		상	중	하
프로시저 및 호출문 작성	- 매개 변수를 맞게 사용하여 구현 및 호출하는 능력			
	- DQL을 사용하여 데이터를 조회하는 능력			
	- DML을 사용하여 데이터를 삽입, 수정, 삭제하는 능력			
	- 반복문, 조건문, 분기문 등의 명령어 사용 능력			
	- 단순하고 효율적인 데이터 흐름을 통해 쉽고 적은 분량으로 작성하는 능력			
	- 예외 사항에 대한 처리 방안 작성 능력			
	- 줄바꿈 및 주석, 용어 통일 등을 통해 가독성을 확보하도록 작성하는 능력			
	- 컴파일 가능하게 프로시저를 작성하는 능력			
	- 호출 시 오류를 발생시키지 않고 정상적 종료까지 도달하게 하는 능력			
	- 결과 검증을 통해 기대하는 결과를 도출하는 능력			
사용자 정의함수 및 호출쿼리 작성	- 매개 변수를 맞게 사용하여 구현 및 호출하는 능력			
	- DQL을 사용하여 데이터를 조회하는 능력			
	- 단순하고 효율적인 데이터 흐름을 통해 쉽고 적은 분량으로 작성하는 능력			
	- 예외 사항에 대한 처리 방안 작성 능력			
	- 줄바꿈 및 주석, 용어 통일 등을 통해 가독성을 확보하도록 작성하는 능력			
	- 컴파일 가능하게 사용자 정의함수를 작성하는 능력			
	- 호출 시 오류를 발생시키지 않고 정상적 종료까지 도달하는 능력			
	- 호출 시 기대했던 단일값의 정상 반환을 달성하는 능력			
	- 이벤트를 맞게 선정하고 이로 인한 트리거를 실행 가능하게 작성하는 능력			
	- DML을 사용하여 데이터를 삽입, 수정, 삭제하는 능력			
트리거 작성				

<ul style="list-style-type: none"> <li>- 줄바꿈 및 주석, 용어 통일 등을 통해 가독성을 확보하도록 작성하는 능력</li> <li>- 컴파일 가능하게 트리거를 작성하는 능력</li> <li>- 트리거 이벤트 발생 시 오류를 발생시키지 않고 정상적으로 트리거가 실행되게 하는 능력</li> <li>- 트리거 이벤트 발생 후 수립했던 목표대로 데이터 삽입, 삭제, 수정이 되도록 작성하는 능력</li> </ul>			
--	--	--	--

## 피드백

### 1. 문제 해결 시나리오

- 초기 요구사항과 이벤트, 절차형 SQL 내부에서의 데이터 플로우 부분이 가장 중요하다. 이에 대한 이해가 부족한 경우 충분히 설명하고 부족한 부분을 보완하도록 한다.
- 절차형 SQL 활용을 통해 수립 가능한 목표 수준을 확인, 인지시키고 이해가 부족한 경우 부족한 부분을 보완하도록 한다.
- 내부 데이터 프로세스의 합리적 수행 경로를 확인, 인지시키고 이해가 부족한 경우 부족한 부분을 보완하도록 한다.

### 2. 평가자 질문

- 절차형 SQL의 사용 목적, 활용성, 사용 시 장단점 및 유의점에 대해 답변할 수 있도록 하고 이해가 부족한 경우 부족한 부분을 보완하도록 한다.
- 외부 변수 및 이벤트 작성법에 대해 답변할 수 있도록 하고, 이해가 부족한 경우 부족한 부분을 보완하도록 한다.
- 합리적 내부 데이터 프로세스 수립 및 작성에 대해 답변할 수 있도록 하고, 이해가 부족한 경우 부족한 부분을 보완하도록 한다. 모듈화에 대한 장점과 구성방법 등도 주지시킨다.
- 예상 가능한 예외 상황 발생 시 대처 방안에 대해 답변할 수 있도록 하고, 이해가 부족한 경우 부족한 부분을 보완하도록 한다. 예외가 발생하지 않는 경우도 많지만, 실제 발생시 내부 오류가 될 수 있으므로 소홀히 하지 않아야 함을 주지시킨다.
- 컴파일이나 실행 시 에러가 발생할 경우의 대처 방안에 대해 답변할 수 있도록 하고, 이해가 부족한 경우 부족한 부분을 보완하도록 한다.

### 3. 평가자 체크리스트

- SQL이나 절차형 SQL 문법의 정상적 사용을 통한 작성 여부를 확인하고, 이해가 부족한 경우 부족한 부분을 보완하도록 한다.
- 반복문, 조건문, 분기문 등 문법의 정상적 사용을 통한 작성 여부를 확인하고, 이해가 부족한 경우 부족한 부분을 보완하도록 한다.
- 작성한 절차형 SQL의 정상적 컴파일 여부를 확인하고, 불가한 경우 부족한 부분을 보완하도록 한다. 정상적 컴파일을 하는 역량 자체보다도, 컴파일시 발생하는 오류 등을 보면서 스스로 조치하고 수정할 수 있는 역량을 키우는 것이 중요하다.
- 외부 호출 또는 이벤트를 통한 실행 시의 정상 실행 여부를 확인하고, 불가한 경우 부족한 부분을 보완하도록 한다.

## 학습 2

## 응용 SQL 작성하기

## 2-1. 집계성 SQL 작성

## 학습 목표

- 원도우 함수와 그룹 함수를 사용하여 순위와 소계, 중계, 총합계를 산출하는 DML(Data Manipulation language) 명령문을 작성할 수 있다.

## 필요 지식 /

## ① 데이터 분석 함수의 개념

## 1. 데이터 분석 함수의 정의 및 목적

관계형 데이터베이스에서는 단일 행 기준의 처리가 주로 이루어진다. 그러나 총합, 평균 등의 데이터 분석을 위해서는 복수 행 기준의 데이터를 모아서 처리하는 것이 필수적이다. 이와 같은 다중 행 처리를 목적으로 하는 다중 행 함수가 존재한다.

## 2. 데이터 분석 함수의 특성

데이터 분석을 위한 다중 행 함수의 공통적인 특성은 아래와 같다.

- 단일 행을 기반으로 산출하지 않고 복수 행을 그룹별로 모아 놓고 그룹당 단일 계산 결과를 반환한다.
- GROUP BY 구문을 활용하여 복수 행을 그룹핑한다.
- SELECT, HAVING, ORDER BY 등의 구문에 활용한다.

## 3. 데이터 분석 함수의 유형

SQL 표준에서는 데이터 투플 간의 상호 연관 및 계산 분석을 위한 세 가지 함수가 있다.

- 집계 함수(AGGREGATE FUNCTION)
- 그룹 함수(GROUP FUNCTION)
- 원도우 함수(WINDOW FUNCTION)

집계 함수는 명령어도 상대적으로 단순하며 이를 기본으로 하여 그룹 함수나 원도우 함수에도 적용된다. 집계 함수도 그룹 함수의 한 부분이며, 통합하여 언급하기도 한다.

## ② 집계 함수

### 1. 집계 함수의 구문

상기 언급한 바와 같이 GROUP BY 구문을 활용하여 복수 행을 그룹핑하여 분석 결과 데이터를 반환한다. 그룹 함수의 구문은 아래와 같다.

<표 2-1> 집계 함수 구문

```
SELECT [COLUMN_1, COLUMN_2, COLUMN_3]
      , <GROUP FUNCTION>
    FROM <TABLE_NAME>
  [WHERE ...]
 GROUP BY <COLUMN_1, COLUMN_2, COLUMN_3...>
 [HAVING 조건식(GROUP FUNCTION 포함)]
```

GROUP BY 구문 뒤에는 테이블을 구분하는 컬럼을 기재하여 그룹화한다. 여기에 지정하는 열을 집약키 또는 그룹화 열이라 칭한다.

HAVING 구문은 그룹화된 집합에 대한 조건 지정 시 사용한다. WHERE 조건으로 지정된 데이터 집합으로부터 그룹화된 집합에 대한 조건 선택 시에 HAVING을 사용하는 것이다. HAVING 구문은 선택적이며, 상수나 집약 함수, 집약키를 사용할 수 있다.

### 3. 집계 함수의 분류

자주 사용되는 집계 함수는 아래와 같다. 집계 특성상 숫자 유형의 계산에 사용되는 것이 대다수이나, MAX/MIN 또는 COUNT와 같이 문자열 유형의 최대/최소나 건수 계산 등에도 사용 가능하다.

<표 2-1> 집계 함수 분류

집계 함수	내용	입력값
COUNT(입력값)	복수 행의 줄(ROW) 수	*이나 상수인 경우 NULL이 포함된 행의 수. 컬럼 적용 시 해당 입력값 내의 NULL이 아닌 ROW 수 산출
SUM(입력값)	복수 행의 해당 컬럼(COLUMN) 간의 합계	상수 입력 시 해당 ROW 수 * 상수의 값(1인 경우라면 ROW COUNT). 컬럼 적용 시 해당 입력값 내의 NULL이 아닌 값의 합계
AVG(입력값)	복수 행의 해당 컬럼 가의 평균	해당 입력값 내의 NULL이 아닌 값의 평균
MAX(입력값)	복수 행의 해당 컬럼 중 최댓값	해당 입력값 내의 최댓값 출력. 문자열이나 날짜 데이터 형식에도 적용 가능
MIN(입력값)	복수 행의 해당 컬럼 중 최솟값	해당 입력값 내의 최솟값 출력. 문자열이나 날짜 데이터 형식에도 적용 가능
STDDEV(입력값)	복수 행의 해당 컬럼 가의 표준편차	해당 입력값 내의 표준편차
VARIAN(입력값)	복수 행의 해당 컬럼 간의 분산	해당 입력값 내의 분산

출처: 한국데이터베이스진흥원(2011). 『SQL 전문가 가이드』. 한국데이터베이스진흥원. p. 255.

#### 4. GROUP BY 구문

SQL에서는 WHERE 구문을 활용하여 조건별 대상 ROW를 선택한다. 그러나 복수 ROW 대상의 데이터 분석 시 그룹핑 대상이 되는 부분을 선별할 필요가 있다. GROUP BY는 그 와 같은 경우에 사용하며, 아래와 같은 특성을 가진다.

- NULL값을 가지는 ROW는 제외한 후 산출한다.
- SELECT에서 사용하는 것과 같은 ALIAS 사용이 불가하다.
- WHERE 구문 안에 포함되지 않는다.
- WHERE 구문은 GROUP BY보다 먼저 실행되고, 대상이 되는 단일 행을 사전에 선별하는 역할을 한다.

GROUP BY 구문은 실제 구체적 데이터 분석값을 보고자 하는 컬럼 단위를 선정할 때 사용되는 기준이 되며, 이 부분의 조정을 통해 사용자가 원하는 분석 데이터를 볼 수 있게 해 준다.

#### 5. HAVING

HAVING 구문은 WHERE 구문 내에는 사용할 수 없는 집계 함수의 구문을 적용하여 복수 행의 계산 결과를 조건별로 적용하는 데 사용된다. 일반적으로 GROUP BY 뒤에 기재하며, GROUP BY 구문의 기준 항목이나 소그룹 집계 함수를 활용한 조건을 적용하는 데 사용한다. 쉽게 생각하면 GROUP BY 및 집계 함수에 대한 WHERE 구문이다.

### ③ 그룹 함수

#### 1. 그룹 함수의 개념

집계 함수와 유사한 개념이고 또한 집계 함수를 포함하기도 하나, 여기서는 소그룹 간의 소계 및 중계 등의 중간 합계 분석 데이터를 산출하는 부분을 말한다. 소계 및 총계 등을 구하기 위해서 집계 함수만 사용한다면 레벨별 집계를 위한 각 단계별 데이터 질의어 (DQL: Data Query Language)를 UNION ALL 등으로 결합하고 표시하는 단계를 거쳐야 한다. 그러나 그룹 함수를 사용한다면 단일 DQL만으로도 원하는 작업을 할 수 있다. 다음의 예제를 통해 해당 개념을 이해할 수 있다.

<표 2-2> 동일한 결과 표시를 위해 사용한 SQL 집계 함수(상)와 그룹 함수(하)

```
SELECT NATION_NM          -- 국가명칭
      , COUNTY_NM           -- 구역명칭
      , SUM(SALES_AMT)      -- 매출액
   FROM LOCAL_SALES_RESULT
  GROUP BY NATION_NM
      , COUNTY_NM
```

```
UNION ALL

SELECT NATION_NM
      , ''
      , SUM(SALES_AMT)
   FROM LOCAL_SALES_RESULT
  GROUP BY NATION_NM
```

```
UNION ALL

SELECT ''
      , ''
      , SUM(SALES_AMT)
   FROM LOCAL_SALES_RESULT
```

---

```
-----  
SELECT NATION_NM          -- 국가명칭
      , COUNTY_NM           -- 구역명칭
      , SUM(SALES_AMT)       -- 매출액
   FROM LOCAL_SALES_RESULT
  GROUP BY ROLLUP (NATION_NM, COUNTY_NM)
```

위쪽의 SQL과 아래쪽의 SQL은 동일한 결과를 도출한다. 여기서 소계나 총계를 구하는 데에 있어서는 그룹함수가 적절함을 판단할 수 있다.

## 2. 그룹 함수의 유형

### (1) ROLLUP

ROLLUP에 의해 지정된 컬럼은 소계 등 중간 집계값을 산출하기 위해 사용된다. 지정 컬럼의 수보다 하나 더 큰 레벨만큼의 중간 집계값이 생성된다. ROLLUP의 지정 컬럼은 계층별로 구성되기 때문에 순서가 바뀌면 수행 결과가 바뀜을 유의한다. ROLLUP은 다음과 같이 사용한다.

<표 2-3> ROLLUP 활용 구문

```
SELECT [COLUMN_1, COLUMN_2, COLUMN_3]
      , <GROUP FUNCTION>
   FROM <TABLE_NAME>
  [WHERE ...]
 GROUP BY [COLUMN...] ROLLUP <COLUMN...>
  [HAVING ...]
  [ORDER BY ...]
```

소계 집계 대상이 되는 컬럼을 ROLLUP 뒤에 기재한다. 소계 집계 대상이 아닌 경우 GROUP BY 뒤에 기재한다. SELECT 뒤에 포함되는 컬럼이 GROUP BY 또는 ROLLUP 뒤에 기재되어야 한다는 점만 숙지하고 쿼리를 작성한다.

소계나 총합계 등의 경우는 집계 대상이 되는 행의 그룹핑 앞 또는 뒤에 표현하는 것이 일반적이다. ORDER BY 구문을 활용해 계층 내 정렬에 사용이 가능하며, SQL의 결과를 보다 체계적으로 보여 준다.

## (2) CUBE

CUBE는 결합 가능한 모든 값에 대해 다차원 집계를 생성하는 그룹 함수이며, 가능한 한 소집계만을 생성하는 ROLLUP과 구별된다. CUBE는 내부적으로 대상 컬럼의 순서를 변경하여 또 한 번의 쿼리를 수행한다. 또 총계는 양쪽 쿼리에서 모두 수행 후 한 쪽에서 제거되는 과정에 의해 ROLLUP에 비해 계산이 많다. CUBE 함수의 사용법도 ROLLUP과 유사하며 다음과 같다.

<표 2-4> CUBE 활용 구문

```
SELECT [COLUMN_1, COLUMN_2, COLUMN_3]
      , <GROUP FUNCTION>
      FROM <TABLE_NAME>
      [WHERE ...]
      GROUP BY [COLUMN...] CUBE <COLUMN...>
      [HAVING ...]
      [ORDER BY ...]
```

## (3) GROUPING SETS

GROUPING SETS를 이용해 다양한 소계 집합을 만들 수 있다. 집계 대상 컬럼들에 대한 개별 집계를 구할 수 있으며, ROLLUP이나 CUBE와는 달리 컬럼 간 순서와 무관한 결과를 얻을 수 있다. 또 ORDER BY를 사용하여 집계 대상 그룹과의 표시 순서를 조정하여 체계적으로 보여 준다. GROUPING SETS의 사용법은 다음과 같다.

<표 2-5> GROUPING SETS 활용 구문

```
SELECT [COLUMN_1, COLUMN_2, COLUMN_3]
      , <GROUP FUNCTION>
      FROM <TABLE_NAME>
      [WHERE ...]
      GROUP BY [COLUMN...] GROUPING SETS <COLUMN...>
      [HAVING ...]
      [ORDER BY ...]
```

#### ④ 윈도우 함수

##### 1. 윈도우 함수의 개념

윈도우 함수는 데이터베이스를 사용한 온라인 분석 처리 용도로 사용하기 위해서 표준 SQL에 추가된 기능이다. 온라인 분석 처리는 시장 분석, 통계 작성, 경영 계획 분석 및 수립 등 비즈니스 현장에서 자주 사용되는 분석이 포함되며, 데이터 기반 의사 결정의 증가에 따라 그 중요성이 더욱 증가하는 실정이다. 윈도우 함수를 이런 이유로 OLAP(OnLine Analytical Processing) 함수라고도 한다.

##### 2. 윈도우 함수의 구문

윈도우 함수의 구문은 다음과 같다.

<표 2-6> 윈도우 함수의 구문

```
SELECT <WINDOWS_FUNCTION> [ARGUMENTS] OVER  
    ([PARTITION BY <COLUMN_1, COLUMN_2, COLUMN_3...>])  
    [ORDER BY <COLUMN_A, COLUMN_B, COLUMN_C...>]  
FROM TABLE
```

PARTITION BY는 선택 항목이며, 순위를 정할 대상 범위의 컬럼을 설정한다. PARTITION BY 구에는 GROUP BY 구가 가진 집약 기능이 없으며, 이로 인해 ROW 수가 줄어들지 않는다. 참고로, PARTITION BY를 통해 구분된 레코드 집합을 윈도우라고 하며, 윈도우 함수의 유래는 이에 기인한다.

윈도우 함수에는 OVER 문구가 필수적으로 포함되며, ORDER BY 뒤에는 SORT 컬럼을 입력한다. 즉, 어떤 열을 어떤 순서로 순위를 정할지를 지정하며 SELECT 문의 마지막에 사용하는 ORDER BY와 동일하게 사용한다.

##### 3. 윈도우 함수의 예시

윈도우 함수의 간단한 예제는 다음과 같다.

<표 2-7> 윈도우 함수의 예시

```
SELECT MAJOR, SUBJECT, STUDENT_NAME, GRADE  
    , RANK()      OVER (ORDER BY GRADE DESC) AS GRADE_RANK  
    , DENSE_RANK() OVER (ORDER BY GRADE DESC) AS GRADE_DENSE_RANK  
    , ROW_NUMBER() OVER (ORDER BY GRADE DESC) AS GRADE_ROW_NUMBER  
    , RANK()      OVER (PARTITION BY MAJOR ORDER BY GRADE DESC) AS GRADE_RANK_IN_SUB  
FROM STUDENT_GRADE
```

학생들의 GRADE를 기준으로 하여 순위를 계산하는 예시이다. RANK(), DENSE\_RANK(), ROW\_NUMBER() 등의 목적에 맞는 윈도우 함수를 사용할 수 있으며, PARTITION BY 구문 뒤에는 순위의 기준이 되는 대상을 조정할 수 있다.

## 4. 윈도우 함수의 분류

### (1) 집계 함수

앞에서 설명한 집계 함수와 동일하다. 대다수의 DBMS에서 지원한다.

### (2) 순위 함수

#### (가) RANK

레코드의 순위를 계산한다. 동일 순위의 레코드 존재 시 후순위는 넘어간다.

예) 2위가 3개인 레코드인 경우: 2위, 2위, 2위, 5위, 6위, ...

#### (나) DENSE\_RANK

레코드의 순위를 계산한다. 동일 순위의 레코드 존재 시에도 후순위를 넘어가지 않는다.

예) 2위가 3개인 레코드인 경우: 2위, 2위, 2위, 3위, 4위, ...

#### (다) ROW\_NUMBER

레코드의 순위를 계산한다. 동일 순위의 값이 존재해도 이와 무관하게 연속 번호를 부여한다.

예) 2위가 3개인 레코드인 경우: 2위, 3위, 4위, 5위, 6위, ...

### (3) 행순서 함수(SQL Server에서는 지원하지 않음)

#### (가) FIRST\_VALUE

파티션별 윈도우에서 가장 먼저 나오는 값을 찾는다. 집계 함수의 MIN과 동일한 결과를 출력한다.

#### (나) LAST\_VALUE

파티션별 윈도우에서 가장 늦게 나오는 값을 찾는다. 집계 함수의 MAX와 동일한 결과를 출력한다.

#### (다) LAG

파티션별 윈도우에서 1부터 이전 몇 번째 행의 값을 가져온다.

#### (라) LEAD

파티션별 윈도우에서 1부터 이후 몇 번째 행의 값을 가져온다.

### (4) 그룹 내 비율 함수

RATIO\_TO\_REPORT, PERCENT\_RANK, CUME\_DIST, NTILE 등의 그룹 내 비율 함수가 있다.

## 수행 내용 / 집계 SQL 작성하기

---

### 재료 · 자료

- 논리 E-R 다이어그램
- 물리 E-R 다이어그램
- 테이블 설계서
- 뷰 명세서
- 인덱스 설계서
- 데이터 표준 정의서
- 데이터 요구사항

### 기기(장비 · 공구)

- 컴퓨터
- DBMS(DataBase Management System) 설치 프로그램
- DBMS 클라이언트 프로그램

### 안전 · 유의 사항

- 실습 후 사용한 PC의 DBMS 접속 종료

### 수행 순서

#### ① 순위 조회 수행 SQL을 작성한다.

1. 순위를 조회하는 대상 항목을 정의한다.
  - (1) 구하고자 하는 순위 대상 및 값이 있는지를 확인한다.
  - (2) 테이블 내의 순위 대상이 되는 컬럼 및 속성(문자열, 숫자 또는 날짜)을 확인한다.
2. 윈도우 함수를 이용하여 순위를 구하는 SQL을 작성한다.  
집계 함수를 사용하여 순위 산출도 가능하지만, 더 좋은 방법은 윈도우 함수를 이용하는 것이다. 집계 함수보다 간단하게 순위 산출이 가능하며 이런 이유로 인해 유지 보수 및 수정도 쉽다. 다음은 작업 순서이며, 예제는 맨 아래와 같다.

- (가) 순위의 대상이 되는 컬럼을 확인한다. 컬럼은 숫자뿐만 아니라 문자열, 날짜 등의 데이터 형식도 가능하다.
- (나) 순위를 정의하는 방식을 결정한다. 동일 값에 동일 순위를 부여하며 후순위를 넘어 가는 경우에는 RANK(), 동일 값에 동일 순위를 부여하며 후순위를 넘어가지 않는 경우에는 DENSE(), 동일 값에 동일 순위를 부여하지 않는 경우는 ROW\_NUMBER()를 사용한다. 아래 예제에서 각각의 함수 결과를 확인하여 참조한다.
- (다) OVER ( ORDER BY 뒤에 순위의 대상이 되는 컬럼을 기재한다. 이때 순위를 산정하는 그룹을 나눌 것인지를 판단하고, 나누게 된다면 PARTITION BY로 구분한다. PARTITION BY는 다중 컬럼으로도 가능하다. 컬럼명 뒤에는 DESC 또는 ASC를 기재하여 내림차순 또는 올림차순을 구분한다. ASC는 기재하지 않아도 무방하다. 아래 예제에서 OVER ( ) 내부에 PARTITION BY 유무에 따라 달라지는 결과를 확인하여 참조한다.
- (라) 그룹핑의 구분 대상이 되는 컬럼을 맨 앞의 SELECT 뒤에 순서대로 기재한다.
- (마) 대상 테이블 또는 데이터 그룹을 FROM 이하에 기재한다.

#### WINDOWS FUNCTION EXAMPLE

```

SELECT MAJOR, SUBJECT, STUDENT_NAME, GRADE (라)
    , RANK()          OVER (
    , DENSE_RANK()   OVER (           (다)
    , ROW_NUMBER()   OVER (
    , RANK() (나)    OVER (PARTITION BY MAJOR ORDER BY GRADE DESC) AS GRADE_RANK_IN_SUB
FROM STUDENT_GRADE (마)

```

MAJOR	SUBJECT	STUDENT_NAME	GRADE (가)	GRADE_RANK	GRADE_DENSE_RANK	GRADE_ROWNUMBER	GRADE_RANK_IN_SUB
COMPUTER	SW ENGINEERING	JOHN	4.5	1	1	1	1
COMPUTER	SW ENGINEERING	PETER	2.0	9	6	9	4
COMPUTER	SW ENGINEERING	SCOTT	4.0	2	2	2	2
COMPUTER	SW ENGINEERING	ROBERT	1.5	10	7	10	5
COMPUTER	SW ENGINEERING	JESSICA	3.5	4	3	4	3
COMPUTER	NETWORK	SUSAN	2.5	8	5	8	5
COMPUTER	NETWORK	JAMES	4.0	2	2	3	1
COMPUTER	NETWORK	TAILOR	3.0	5	4	5	2
COMPUTER	NETWORK	ALICE	3.0	5	4	6	2
COMPUTER	NETWORK	SMITH	3.0	5	4	7	2

[그림 2-1] 순위 함수 사용 예제

② 소계, 중계, 총합계 수행 SQL을 작성한다.

1. 소계나 합계 등 합하여 더하고자 하는 대상 항목을 정의한다.
  - (1) 구하고자 하는 순위 대상 및 값을 확인한다.
  - (2) 테이블 내의 순위 대상이 되는 컬럼 및 속성(문자열, 숫자 또는 날짜)을 확인한다.
2. 소계, 중계, 총합계를 구하는 SQL을 작성한다.
  - (1) 그룹 함수 SQL을 이용하여 소계, 중계, 총합계를 산출한다.

소계나 총계를 ROW 기준으로 표시하는 경우에는 GROUP BY와 ROLLUP을 이용하여 소계 등을 구하는 것이 가장 일반적이다. 작성 순서는 다음과 같다.

    - (가) 조회 대상 테이블을 확인하고 FROM 뒤에 기재한다.
    - (나) 집계 함수를 사용해야 하는 컬럼을 확인한다.
    - (다) 총계 대상 컬럼을 확인한다. 총계 산출 시 기준이 되는 컬럼을 GROUP BY절 뒤에 넣는다.
    - (라) 소계의 기준이 되는 컬럼을 확인한다. 소계 산출 시 기준이 되는 컬럼을 ROLLUP 구문 뒤에 넣는다.
    - (마) (다), (라)에서 기재한 컬럼을 참조하여 조회 대상 컬럼을 SELECT 뒤에 기재한다.

---

GROUP FUNCTION EXAMPLE 1

---

(마) (나)  
SELECT COLLEGE, MAJOR, ITEM, PRICE, SUM(COUNT) AS COUNT, SUM(AMOUNT) AS AMOUNT  
FROM PURCHASE\_LIST (가)  
GROUP BY COLLEGE, ROLLUP(MAJOR, ITEM, PRICE)  
(다) (라)

---

COLLEGE	MAJOR	ITEM	PRICE	COUNT	AMOUNT
ENGINEERING	COMPUTER ENG.	BALLPEN	200	7	1,400
ENGINEERING	COMPUTER ENG.	FOUNTAIN PEN	4,000	2	8,000
ENGINEERING	COMPUTER ENG.	PENCIL	300	11	3,300
ENGINEERING	COMPUTER ENG.	ERASER	500	5	2,500
ENGINEERING	COMPUTER ENG.	RULER	1,000	3	3,000
ENGINEERING	COMPUTER ENG.			28	18,200
ENGINEERING	ELECTRONICS ENG.	BALLPEN	200	5	1,000
ENGINEERING	ELECTRONICS ENG.	FOUNTAIN PEN	4,000	1	4,000
ENGINEERING	ELECTRONICS ENG.	PENCIL	300	1	300
ENGINEERING	ELECTRONICS ENG.	ERASER	500	5	2,500
ENGINEERING	ELECTRONICS ENG.	RULER	1,000	4	4,000
ENGINEERING	ELECTRONICS ENG.			16	11,800
ENGINEERING				44	30,000

[그림 2-2] 그룹 함수를 활용한 소계, 총계 사용 예제

(2) 윈도우 함수 SQL을 이용하여 소계, 중계, 총합계를 산출한다.

SUM과 OVER (PARTITION BY ...)를 사용하여 소계를 구할 수 있다. 소계나 총계를 ROW 단위로 기재할 경우에는 뒤에 나올 집계 함수와 비슷한 형태로 구성하는 번거로움으로 인해 이런 경우에는 그룹 함수가 가장 적당하나, 여기서는 윈도우 함수로도 가능함을 확인하기로 한다. 작성 순서는 다음과 같다.

- (가) 집계 대상 테이블 또는 데이터 그룹으로부터 조회한다(SELECT).
- (나) UNION ALL을 사용한다(소계 함수 구문을 행으로 추가하기 위함).
- (다) 집계 함수를 사용해야 하는 컬럼을 확인한다. SUM이나 COUNT 등 적용할 함수를 확인한다.
- (라) OVER (PARTITION BY ~ 뒤에 소계 그룹핑 구분 대상이 되는 컬럼을 추가한다. 집계 함수를 사용하는 경우의 GROUP BY 뒤에 기재할 컬럼과 동일한 컬럼을 추가하면 된다.

---

GROUP FUNCTION EXAMPLE 2

---

(가) SELECT COLLEGE, MAJOR, ITEM, PRICE, COUNT, AMOUNT  
FROM PURCHASE\_LIST

(나)  
UNION ALL  
(마)

SELECT DISTINCT COLLEGE, MAJOR, '' AS ITEM, '' AS PRICE  
, SUM(COUNT) OVER (PARTITION BY COLLEGE, MAJOR) AS COUNT  
(다), SUM(AMOUNT) OVER (PARTITION BY COLLEGE, MAJOR) AS AMOUNT

(바) FROM PURCHASE\_LIST (라)  
UNION ALL

SELECT DISTINCT COLLEGE, '' AS MAJOR, '' AS ITEM, '' AS PRICE  
, SUM(COUNT) OVER (PARTITION BY COLLEGE) AS COUNT  
, SUM(AMOUNT) OVER (PARTITION BY COLLEGE) AS AMOUNT  
FROM PURCHASE\_LIST

---

COLLEGE	MAJOR	ITEM	PRICE	COUNT	AMOUNT
ENGINEERING	COMPUTER ENG.	BALLPEN	200	7	1,400
ENGINEERING	COMPUTER ENG.	FOUNTAIN PEN	4,000	2	8,000
ENGINEERING	COMPUTER ENG.	PENCIL	300	11	3,300
ENGINEERING	COMPUTER ENG.	ERASER	500	5	2,500
ENGINEERING	COMPUTER ENG.	RULER	1,000	3	3,000
ENGINEERING	COMPUTER ENG.			28	18,200
ENGINEERING	ELECTRONICS ENG.	BALLPEN	200	5	1,000
ENGINEERING	ELECTRONICS ENG.	FOUNTAIN PEN	4,000	1	4,000
ENGINEERING	ELECTRONICS ENG.	PENCIL	300	1	300
ENGINEERING	ELECTRONICS ENG.	ERASER	500	5	2,500
ENGINEERING	ELECTRONICS ENG.	RULER	1,000	4	4,000
ENGINEERING	ELECTRONICS ENG.			16	11,800
ENGINEERING				44	30,000

[그림 2-3] 윈도우 함수를 활용한 소계, 총계 사용 예제

- (마) SELECT 뒤에 DISTINCT를 추가한다. 윈도우 함수를 사용하는 경우 GROUP BY를 사용하지 않아 집계 대상 데이터를 그룹화하지 않아서 ROW가 줄어들지 않기 때문에 소계나 총계의 표시 ROW를 한 줄만 남기기 위해서이다.
- (바) 추가로 중계나 총합계가 필요한 경우 (나)~(마)까지의 과정을 반복한다. 단계가 증가함에 따라 집계 그룹 대상을 축소시킨다.

(3) 집계 함수 SQL을 이용하여 소계, 중계, 총합계를 산출한다.

SUM과 GROUP BY, UNION ALL 등을 활용하여 소계, 중계 및 총합계 산출도 가능하다. 물론 이 경우도 그룹 함수가 더 작성하기도 편하고 그만큼 효율도 높지만 다양하게 SQL을 활용하는 역량 증대 측면에서 해보도록 한다. 작성 순서는 다음과 같다.

- (가) 집계 대상 테이블 또는 데이터 그룹으로부터 조회한다(SELECT).
- (나) UNION ALL을 사용한다(소계 함수 구문을 행으로 추가하기 위함).
- (다) 집계 함수를 사용해야 하는 컬럼을 확인한다.
- (라) 사용 대상 집계 함수를 확인한다. 합계의 경우 SUM, 건수의 건수 COUNT를 사용한다. 해당 컬럼에 NULL이 가능한지의 여부를 확인하고, NULL이 가능한 컬럼도 합계나 건수를 사용해야 한다면 NVL을 활용하여 NULL을 제거한다.
- 예) SUM(NVL(AMOUNT,0)), COUNT(NVL(TRY\_CNT,1))

GROUP FUNCTION EXAMPLE 3																																																																																													
<b>(가)</b> SELECT COLLEGE, MAJOR, ITEM, PRICE, COUNT, AMOUNT FROM PURCHASE_LIST																																																																																													
<b>(나)</b> UNION ALL <b>(마)</b> SELECT COLLEGE, MAJOR, '' AS ITEM, '' AS PRICE, SUM(COUNT) AS COUNT, SUM(AMOUNT) AS AMOUNT FROM PURCHASE_LIST GROUP BY COLLEGE, MAJOR																																																																																													
<b>(다) (라)</b> <b>(라)</b> SELECT COLLEGE, '' AS MAJOR, '' AS ITEM, '' AS PRICE, SUM(COUNT) AS COUNT, SUM(AMOUNT) AS AMOUNT FROM PURCHASE_LIST GROUP BY COLLEGE																																																																																													
<table border="1"> <thead> <tr> <th>COLLEGE</th><th>MAJOR</th><th>ITEM</th><th>PRICE</th><th>COUNT</th><th>AMOUNT</th></tr> </thead> <tbody> <tr><td>ENGINEERING</td><td>COMPUTER ENG.</td><td>BALLPEN</td><td>200</td><td>7</td><td>1,400</td></tr> <tr><td>ENGINEERING</td><td>COMPUTER ENG.</td><td>FOUNTAIN PEN</td><td>4,000</td><td>2</td><td>8,000</td></tr> <tr><td>ENGINEERING</td><td>COMPUTER ENG.</td><td>PENCIL</td><td>300</td><td>11</td><td>3,300</td></tr> <tr><td>ENGINEERING</td><td>COMPUTER ENG.</td><td>ERASER</td><td>500</td><td>5</td><td>2,500</td></tr> <tr><td>ENGINEERING</td><td>COMPUTER ENG.</td><td>RULER</td><td>1,000</td><td>3</td><td>3,000</td></tr> <tr><td>ENGINEERING</td><td>ELECTRONICS ENG.</td><td>BALLPEN</td><td>200</td><td>5</td><td>1,000</td></tr> <tr><td>ENGINEERING</td><td>ELECTRONICS ENG.</td><td>FOUNTAIN PEN</td><td>4,000</td><td>1</td><td>4,000</td></tr> <tr><td>ENGINEERING</td><td>ELECTRONICS ENG.</td><td>PENCIL</td><td>300</td><td>1</td><td>300</td></tr> <tr><td>ENGINEERING</td><td>ELECTRONICS ENG.</td><td>ERASER</td><td>500</td><td>5</td><td>2,500</td></tr> <tr><td>ENGINEERING</td><td>ELECTRONICS ENG.</td><td>RULER</td><td>1,000</td><td>4</td><td>4,000</td></tr> <tr><td>ENGINEERING</td><td>COMPUTER ENG.</td><td></td><td></td><td></td><td>28</td><td>18,200</td></tr> <tr><td>ENGINEERING</td><td>ELECTRONICS ENG.</td><td></td><td></td><td></td><td>16</td><td>11,800</td></tr> <tr><td>ENGINEERING</td><td></td><td></td><td></td><td></td><td>44</td><td>30,000</td></tr> </tbody> </table>							COLLEGE	MAJOR	ITEM	PRICE	COUNT	AMOUNT	ENGINEERING	COMPUTER ENG.	BALLPEN	200	7	1,400	ENGINEERING	COMPUTER ENG.	FOUNTAIN PEN	4,000	2	8,000	ENGINEERING	COMPUTER ENG.	PENCIL	300	11	3,300	ENGINEERING	COMPUTER ENG.	ERASER	500	5	2,500	ENGINEERING	COMPUTER ENG.	RULER	1,000	3	3,000	ENGINEERING	ELECTRONICS ENG.	BALLPEN	200	5	1,000	ENGINEERING	ELECTRONICS ENG.	FOUNTAIN PEN	4,000	1	4,000	ENGINEERING	ELECTRONICS ENG.	PENCIL	300	1	300	ENGINEERING	ELECTRONICS ENG.	ERASER	500	5	2,500	ENGINEERING	ELECTRONICS ENG.	RULER	1,000	4	4,000	ENGINEERING	COMPUTER ENG.				28	18,200	ENGINEERING	ELECTRONICS ENG.				16	11,800	ENGINEERING					44	30,000
COLLEGE	MAJOR	ITEM	PRICE	COUNT	AMOUNT																																																																																								
ENGINEERING	COMPUTER ENG.	BALLPEN	200	7	1,400																																																																																								
ENGINEERING	COMPUTER ENG.	FOUNTAIN PEN	4,000	2	8,000																																																																																								
ENGINEERING	COMPUTER ENG.	PENCIL	300	11	3,300																																																																																								
ENGINEERING	COMPUTER ENG.	ERASER	500	5	2,500																																																																																								
ENGINEERING	COMPUTER ENG.	RULER	1,000	3	3,000																																																																																								
ENGINEERING	ELECTRONICS ENG.	BALLPEN	200	5	1,000																																																																																								
ENGINEERING	ELECTRONICS ENG.	FOUNTAIN PEN	4,000	1	4,000																																																																																								
ENGINEERING	ELECTRONICS ENG.	PENCIL	300	1	300																																																																																								
ENGINEERING	ELECTRONICS ENG.	ERASER	500	5	2,500																																																																																								
ENGINEERING	ELECTRONICS ENG.	RULER	1,000	4	4,000																																																																																								
ENGINEERING	COMPUTER ENG.				28	18,200																																																																																							
ENGINEERING	ELECTRONICS ENG.				16	11,800																																																																																							
ENGINEERING					44	30,000																																																																																							

[그림 2-4] 집계 함수를 활용한 소계, 총계 사용 예제

- (마) (가)에서 작성한 내용을 복사하여 집계 그룹 대상 컬럼은 변경하지 않고 소계, 중계 및 총합계에 따라 합이 되는 컬럼을 ‘’ 등으로 변환하고 GROUP BY에서도 제거한다.
- (바) 추가로 중계나 총합계가 필요한 경우 (나)~(마)까지의 과정을 반복한다. 단계가 증가함에 따라 집계 그룹 대상을 축소시킨다.

③ 집계성 SQL 명령문을 실행한다.

1. 작성한 집계성 SQL 명령문을 실행한다.

(1) DB TOOL 등을 이용하여 작성한 SQL문을 실행한다.

(2) 실행하면서 오류가 나는 경우 이에 대응한다.

실행 시 오류가 발생할 경우에는 해당 오류 메시지를 참조한다. 오류 원인을 확인하고 SQL을 수정하거나 데이터 수정 등 기타 사항을 조치한 후 재실행한다.

2. 기대하는 결과가 나오는지 확인한다.

(1) 결과가 조회되었는지 확인한다.

SQL 수행과정중 오류가 발생되지 않는가 확인한다. 오류가 발생할 경우, ROW로 표시되는 오류부분 및 메시지 내용을 확인하여 조치하도록 한다.

(2) 결과는 조회되나 기대한 결과값이 아닐 경우 원인을 확인한다.

(가) 윈도우 함수 SQL 사용한 순위 산출 시 정상적 결과가 나오지 않았을 경우 PARTITION BY 이후에 열거되는 컬럼을 위주로 하여 검증하고, ORDER BY에서의 ASC, DESC 등 순서 표시 유형도 정상적으로 입력했는지를 확인한다.

(나) 그룹 함수 SQL 사용한 소계 등 산출 시 정상적 결과가 나오지 않았을 경우 ROLLUP 및 GROUP BY 이후의 컬럼을 혼동하지 않고 잘 기재했는지 확인한다. 소계, 중계 및 총계까지 모두 산출하는 경우라면 ROLLUP 컬럼에 합계가 되는 컬럼이 모두 포함되어야 함을 인식하고, 그렇지 않은 경우라면 GROUP BY와 ROLLUP 간 분배하는 SQL 구문 및 사용법을 다시 확인한다.

### 수행 tip

- 집계 함수의 사용 난이도가 가장 낮으므로 집계 함수를 먼저 익숙하게 활용하고 그다음 그룹 함수, 윈도우 함수를 학습하고 익히는 것이 무난하다.
- 집계 함수와 그룹 함수는 사용법이 유사하나 윈도우 함수는 이외는 상이하다. 구하고자 하는 결과값에 따라 어떤 함수를 사용할 경우 가장 효율적인지 학습자들에게 숙지시킨다.
- 수행 순서의 예제에서도 볼 수 있지만, 데이터 집계 결과를 얻는 데 윈도우 함수나 그룹 함수, 집계 함수를 모두 사용할 수 있으나 윈도우 함수나 그룹 함수를 쓸수록 필요한 문장이 적다. 이는 작성 시 효율뿐만 아니라 추후 운영 및 유지 보수 시에도 많은 이점을 가지고 있는 것이며, 이 점을 학습자들에게 숙지시킨다.

## 2-2. 특정 기능 수행 SQL문 작성

### 학습 목표

- 응용시스템에서 사용하는 특정 기능을 수행하기 위한 SQL문을 작성할 수 있다.

### 필요 지식 /

#### ① 응용시스템 DBMS 접속

##### 1. 응용시스템 DBMS 접속 개념

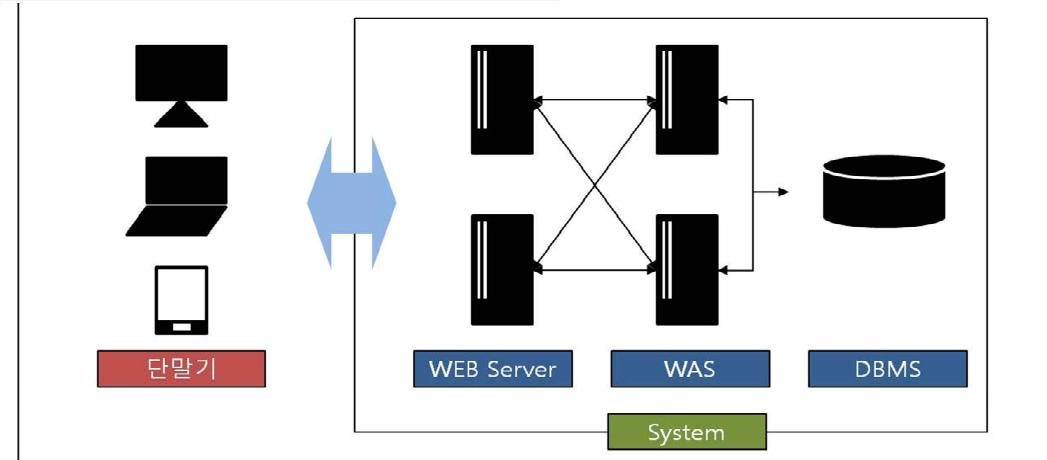
###### (1) 응용시스템과 DBMS와의 관계

DBMS에서 사용되는 SQL은 DB 관리도구를 사용하여 직접 접속해 실행이 가능하고, 역시 응용시스템에서도 호출 방식을 통해 사용이 가능하다. 사용자는 응용시스템을 통해 DBMS에 접속하여 조회하고, 저장하는 것이 가장 일반적인 형태이다. 응용시스템은 입력변수(Input Parameter)를 전달하고 SQL을 실행하여 결과를 응용시스템을 통해 사용자에게 전달하는 일종의 매개체인 것이다.

###### (2) 웹 프로그램 구조(WEB Application Architecture)

웹 응용시스템은 단순하게 표현하면 사용 단말기의 웹 브라우저로부터 요청을 받고 전달하는 웹 서버(WEB Server), 웹 서버로부터 서비스 요청을 받고 트랜잭션을 생성하여 DBMS에 요청하고 응답을 받아서 다시 웹 서버로 전송하는 웹 응용 프로그램 서버(WAS: WEB Application Server)로 구성된다. DBMS에서 실행하는 SQL문은 웹 응용 프로그램 서버에서 실행되며, 간단하게 개념적으로 표현하면 아래와 같다.

WEB Application Archtecture



[그림 2-5] 단순 웹 응용시스템의 구조

규모가 그리 크지 않은 응용 프로그램의 경우는 웹 서버(WEB Server)와 웹 응용 프로그램 서버(WAS)를 통합하여 사용할 수도 있으나, 여러 클라이언트로부터의 요청과 업무 로직, 데이터 접속 및 SQL을 하나의 서버에서보다 역할에 따라 분리하여 사용하는 것이 좀 더 바람직한 경우가 많다.

## 2. 응용시스템 DBMS 접속 기술

### (1) 자바 데이터베이스 연결(JDBC: Java DataBase Connectivity)

언어마다 DBMS에 접근할 수 있게 하는 기술은 여러 가지가 있으나 기본적인 개념은 유사하다. JAVA는 자바 데이터베이스 연결(JDBC: JAVA DataBase Connectivity)을 사용한다. JDBC는 SQL을 사용하여 DBMS에 질의하고 데이터를 조작하는 API (Application Programming Interface)를 제공한다. JDBC API를 사용하는 방법은 앞에서 순수한 JAVA 언어로 JDBC를 사용하는 방법과 동일하다. 아래는 JDBC API를 사용하여 Input Parameter가 존재하는 SELECT SQL을 수행하는 코드 예제이다.

<표 2-8> JDBC API를 이용한 SQL 작성 예제

```
private static final String SQL_GETID = "SELECT * FROM USER_INFO WHERE USER_NAME = ?";  
public List <CustomEntity> findUserId(String user_name)  
{  
    List<CustomEntity> customers = new ArrayList<CustomEntity>();  
    try  
    {  
        connection = dataSource.getConnection();  
        sql_exec = connection.prepareStatement(SQL_GETID);  
        sql_exec.setString(1, user_name);  
        sql_result = sql_exec.executeQuery();  
        UserEntity userInfo = null;  
  
        while(sql_result.next())  
        {  
            userInfo = new UserEntity();  
            userInfo.setId(sql_result.getString("user_id"));  
            userInfo.add(userInfo);  
        }  
    }  
  
    catch(SQLException e)  
    {  
        ....  
    }  
}
```

## (2) MyBatis

MyBatis는 SQL Mapping 기반 오픈 소스 Access Framework이다. MyBatis는 DBMS에 질의하기 위한 SQL 쿼리를 별도의 XML 파일로 분리하고 Mapping을 통해서 SQL을 실행한다. MyBatis는 DBMS 의존도가 높고 SQL 질의 언어를 사용하기 때문에 SQL 친화적인 국내 실무 개발 환경에 맞아서 많이 사용된다. MyBatis의 장점은 다음과 같다.

- 복잡한 JDBC 코드를 단순화할 수 있다.
- SQL을 거의 그대로 사용 가능하다.
- Spring 기반 프레임워크와 통합 기능을 제공한다.
- 우수한 성능을 보여 준다.

여기서는 MyBatis를 위주로 응용시스템에서 사용하는 SQL 작성법을 학습한다.

## ② MyBatis SQL 작성 문법

### 1. MyBatis SQL 문법 기본 구조

MyBatis 설정 파일은 데이터 소스 연결, SQL 매퍼 파일, 타입 별칭(Type Alias), 타입 핸들러(Type Handler)와 같은 속성 설정이 가능하다. 또 다양한 속성(Attribute)을 조정하여 다중 접속, 연결 문자 등의 조정이 가능하다. 응용 SQL 작성을 위해 필요하기는 하나, 내용도 방대하고 SQL 작성 부분과는 조금 무관하여, 여기서는 MyBatis 설치 및 설정 관련 부분은 다루지 않고 MyBatis XML 작성에 대해 설명하도록 한다. MyBatis SQL 작성 문법은 다음과 같다.

<표 2-9> MyBatis SQL 작성 문법

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="[SITE URL & Project Structure Tree]"
  <[SQL TYPE] id = [SQL ID] resultType = [RESULT TYPE]>

  [SQL]

</ [SQL TYPE]>
<mapper>
```

실제 SQL을 작성하는 부분은 mapper 이후의 SQL Type 및 SQL 부분이다. MyBatis도 XML 문법을 사용하기 때문에 SQL Type을 시작할 때 “<”, 종료할 때 “</” Tag를 사용하여 표시한다. SQL Type은 SELECT, INSERT, UPDATE, DELETE 등의 타입 중에서 선택한다. SQL 부분에 실제 SQL을 추가하고 이를 실행한다.

## 2. DQL, DML SQL 문장의 Input Parameter 입력 방법

응용시스템을 통해 SQL을 실행할 때 가장 중요한 부분 중 하나는 입력변수에 대한 처리 방법이다. MyBatis에서는 #{Parameter Name}으로 처리한다. 간단한 예는 다음과 같다.

<표 2-10> Input Parameter 입력 방법 예시

```
<mapper>
  <select id = "findId" resultType = "map">
    SELECT USER_ID
    FROM USER_INFO
    WHERE USER_NM = #{user_nm}
      AND BIRTH_DAY = #{user_birth_day}
  </select>
</mapper>
```

이 예제는 웹사이트 등에서 사용자명과 생년월일을 입력받아 ID를 표시하는 SQL 예제이다. 응용시스템의 UI(User Interface, 화면 등)를 통해서 Input Parameter를 입력받고 이것을 SQL문에 적용한 후 실행하는 것이 일반적이다. SQL문은 미리 작성해 놓고, 사용자가 입력하는 항목에 따라 다른 조건의 SQL을 실행하는 것이 그 목적이다.

SELECT문뿐만 아니라 INSERT, UPDATE, DELETE문에도 동일하게 사용 가능하다.

## 3. 동적 SQL

MyBatis를 이용한 SQL의 강점 중 하나는 조건에 따라 SQL 구문 자체를 변경할 수 있다는 것이다. 이를 동적 SQL(Dynamic SQL)이라 하며, 응용 프로그램 실행 시 수행할 SQL문이 결정된다. 위에서 설명한 Input Parameter만 변경하는 것이 아니라 SQL에 포함된 다양한 부분을 조건에 따라 변경 가능하다. 다음의 예제를 참조하자.

<표 2-11> MyBatis 조건별 실행

```
<mapper>
  <select id = "findId" resultType = "map">
    SELECT USER_ID
    FROM USER_INFO
    WHERE USER_NM = #{user_nm}
    <if input_para="user_birth_day != null">
      AND BIRTH_DAY = #{user_birth_day}
    </if>
    <if input_para="user_email != null">
      AND EMAIL = #{user_email}
    </if>
  </select>
</mapper>
```

<if~> 구문을 사용해서 user\_birthday가 있을 때에만 “AND BIRTH\_DAY =” 조건문이 실행될 수 있도록 한 것이다. user\_nm은 필수적으로 입력되는 값이나, 그 외 user\_birth\_day나 user\_email 값이 선택적으로 입력된다면, 오류나 잘못된 결과를 얻는 상황 방지를 위해 위와 같이 SQL을 작성해야 한다. 위의 경우에는 상기 if 문이 없어도 정상적으로 실행 가능한 SQL 작성은 가능하나, 가독성이나 효율성 측면에서는 <if~>의 방법으로 사용하는 것이 좋다. <if~> 외에도 다중 문자열 등의 반복적 입력 시 사용하는 <foreach>, if보다 좀 더 경우별 SQL문 적용을 위한 <choose when otherwise~> 등이 사용된다.

이 부분에서 짚고 넘어가야 할 것은 응용시스템에서 사용하는 SQL을 작성할 때에는 결국 UI에서 어떻게 값이 입력되고, 입력 항목 검증 방법이 어떻게 되는지, 필수값 여부 등 복합적 요소들의 관계를 고려하여 작성해야 한다는 것이다. 사용자가 원하는 서비스 제공을 위해 응용시스템은 SQL을 통해 DBMS로부터 데이터를 얻거나 반영하기 때문에 SQL도 통합 응용시스템의 부분으로서 역할한다는 것을 인식할 필요가 있다.

#### 4. 절차형 SQL 호출

DQL, DML SQL뿐만 아니라 학습 1에서 언급된 절차형 SQL도 실행 가능하다. 절차형 SQL 중 사용자 정의함수는 DQL이나 DML에 포함하여 사용하기 때문에 별도 호출의 의미가 없으며, 트리거 역시 DBMS에서 바로 실행되므로 별도 호출이 불필요하다. 이 중에서 프로시저를 호출하는 예제는 다음과 같다.

<표 2-12> MyBatis에서 프로시저 호출

```
<select id = "execSalesDailyBatch" resultType = "map" statementType = "CALLABLE">
    { CALL RUN_SALES_AMT_BATCH(#{"TARGET_DATE"}) }
</select>
```

여기서 주의해야 할 부분은 프로시저를 호출하는 경우 statementType을 ‘CALLABLE’로 설정해야 한다는 것이며, 프로시저 호출 전에도 ‘CALL’ 문장을 사용해야 한다는 것이다. SQL Type은 SELECT나 INSERT 등 어느 것을 사용해도 무방하나, Transaction을 하는 것이 주된 기능이라면 INSERT나 UPDATE를 사용하는 것이 의미상 전달에 좋다.

## 수행 내용 / 특정 기능 수행 SQL문 작성하기

---

### 재료 · 자료

- 논리 E-R 다이어그램
- 물리 E-R 다이어그램
- 테이블 설계서
- 뷰 명세서
- 인덱스 설계서
- 데이터 표준 정의서
- 데이터 요구사항

### 기기(장비 · 공구)

- 컴퓨터
- DBMS(DataBase Management System) 설치 프로그램
- DBMS 클라이언트 프로그램
- Spring, J2EE 등의 JAVA Framework(MyBatis, Maven 등 포함)가 설치된 PC

### 안전 · 유의 사항

- 실습 후 사용한 PC의 접속 종료

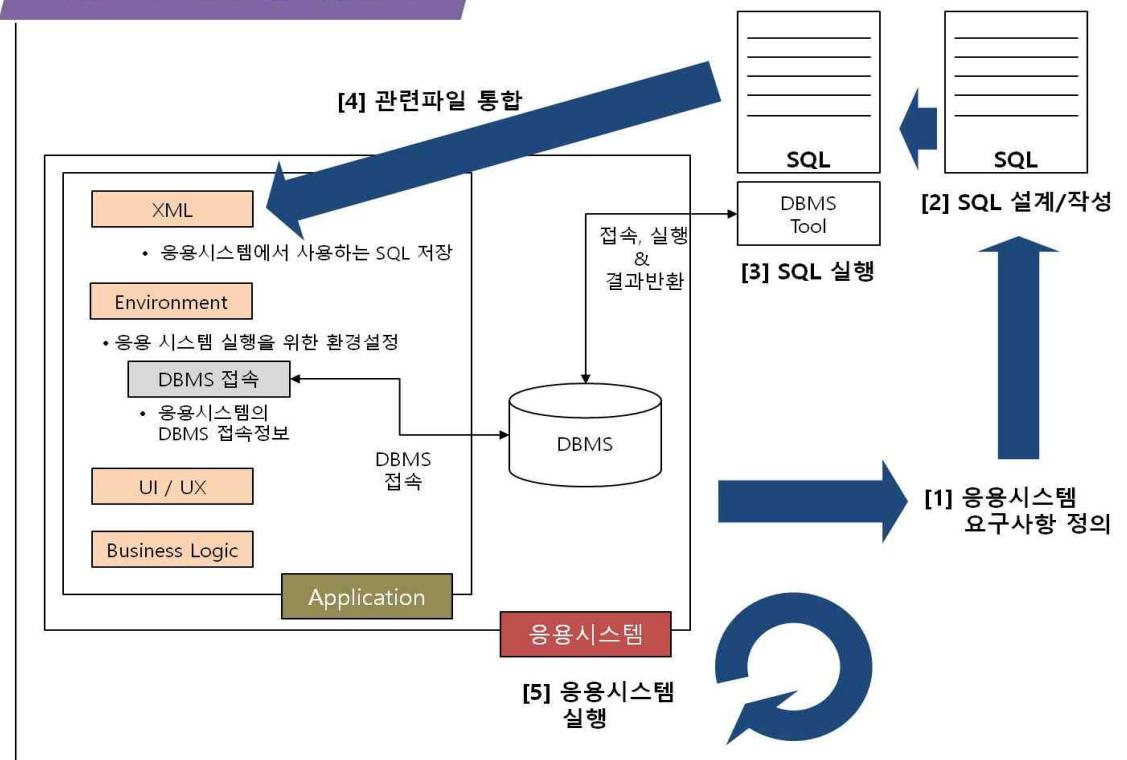
### 수행 순서

① 응용시스템에서 필요로 하는 요구사항을 정의한다.

1. 전체 요구사항을 모두 취합한다.

응용시스템을 통해서 구현해야 하는 서비스의 기능을 모두 정리한다. 요구사항의 성격 및 기능, 상황에 맞게 분류하여 체계화한다.

## 응용 시스템 SQL 작성순서



[그림 2-6] 응용시스템 사용 SQL 작성 순서

### 2. 취합된 요구사항을 분석한다.

#### (1) DBMS 질의가 필요한 부분을 선별한다.

응용시스템에서 요구하는 사항 중 DBMS에 질의를 통해 구현 가능한 기능을 선별하고 기능을 분석한다. DBMS가 아닌 파일 시스템이나 기타 응용시스템 기능을 통해 제공되는 서비스는 배제하는 것이다.

#### (2) 질의 유형(조회, 삽입, 수정, 삭제)을 파악한다.

DBMS로부터 가져오는 것인지 또는 반영하는 것인지, 아니면 두 가지 모두 수행하는 것인지를 구분한다. 두 가지를 모두 수행하는 경우 서비스 제공을 위한 질의 유형의 순서를 파악한다. 간단히 조회 또는 저장 등 개념적 수준으로 요구사항을 분석한다.

#### (3) 서비스의 가변값을 파악한다.

응용시스템에서 요구하는 기능은 불변값과 가변값을 모두 내포한다. 가령, 사용자가 성명 및 생년월일을 입력하여 ID를 찾는 서비스를 사용하고자 할 때, 사용자의 성명과 생년월일은 사용자별로 각각 다르다. 그러나 해당 기능에서 성명과 생년월일을 가지고 ID를 반환한다는 사실은 변하지 않는다. 이런 경우 가변값은 성명과 생년월일이다.

#### (4) 서비스의 가변 항목을 파악한다.

(3)에서 파악한 가변값을 제외하고 달라질 수 있는 부분을 파악한다. 주의해야 할 것은 값이 바뀌는 것보다 더 큰 부분에서 변경되는 부분을 파악하는 것이다. 위의 경우와

유사한 경우를 계속 예로 들면, 사용자가 성명이나 생년월일을 넣지 않고 이메일 주소만 가지고 ID를 찾는다거나 추가로 개인 확인 문구 등을 넣는 것이다. 이런 경우에는 가변값만으로 처리가 불가하며, 입력하는 항목 자체가 달라진다는 것을 파악해야 한다.

## ② 구현을 위한 SQL을 설계한다.

1. 기능 분류를 통해 DQL, DML 또는 프로시저 중 어느 것을 사용할 것인지를 결정한다.

### (1) 질의 유형으로부터 필요한 SQL을 정의한다.

DBMS로부터 가져오는 것은 데이터 질의어, DBMS에 반영하는 것은 데이터 조작어를 사용한다. 두 가지가 모두 필요한 경우 순차적으로 사용하는데, 대개 응용 프로그램에서 데이터 조작어는 단독으로 사용되지 않는다. 그 이유는 DBMS에 반영한 결과를 바로 사용자에게 전달하는 것이 일반적이기 때문이다. 몇 건을 반영하였는지, 반영된 결과 현재는 이렇다는 등의 Refresh가 수반되는 경우가 많기 때문에 데이터 조작어 사용 이후 데이터 질의어를 사용하는 것을 염두에 두도록 한다.

### (2) 프로시저를 작성해야 할지를 판단한다.

앞에서 학습한 절차형 SQL, 특히 프로시저의 사용 여부를 결정한다. 이는 데이터 조작어, 데이터 질의어를 사용하는 경우 모두에 해당된다. 데이터 저장 기능이 필요하지 않더라도 복잡하고 다중 트랜잭션을 적용한 결과를 조회해야 하는 경우, 서브쿼리(Sub Query)나 다중 조인 등 다소 복잡한 DQL로 작성할 것인지 또는 해당 내용 조회를 위한 데이터를 프로시저를 통해 사전 또는 요청 시 관련 데이터를 생성한 다음, 이것을 단순한 DQL로 작성하여 조회할 것인지의 판단이 필요하다. 여기서 판단 기준은 다음을 참조한다.

- 해당 조회 결과 자체에 대해 지속적 보관의 가치가 있는가?
- 해당 조회 결과는 차후에 재조회 시 변경을 허용하지 않는가? (보통 일마감이나 월마감의 데이터는 생성 후 변경을 허락하지 않는다.)
- 해당 조회를 위한 SQL을 만들었을 때 내부 복잡도 및 다량 데이터로 인해 이에 대한 실행 시간이 많이 소요되어 응용 프로그램 사용자가 불편을 느끼는가? 만약 그렇다면, 해당 조회를 위한 데이터를 사전에 만들어 놓을 수 있는가?
- SQL 내에 서브쿼리나 다중 조인, 그룹 함수나 사용자 정의함수가 많음으로써 복잡도가 높아져서 가독성이 떨어지고 추후 유지 보수가 어려울 것으로 예상되는가?
- SQL을 단계별로 좀 더 분할하면 다소 코드의 길이는 길어지더라도 쉽게 이해가 가능하여 유지 보수나 재활용성을 높일 수 있겠는가?

여기서 ‘예’가 많을수록 프로시저를 만드는 것을 검토한다. 결과 데이터를 생성하고 이를 테이블에 반영한 후, 해당 데이터를 단순하게 조회하는 것이 더 간단하고 분

산된 구조가 될 수 있다. 예를 들어 응용시스템을 통해 일별 매출액 집계 등을 보고 싶다고 한다면, 사전에 정해진 시간마다 일마감 집계 데이터를 생성하고 나중에 이를 간단하게 조회하는 것이 효율적이며, 이러한 이유로 많은 시스템이 이와 같은 구조로 운영하고 있다. 다소 복잡하고 일괄 처리가 필요한 기능은 절차형 SQL을 사용하여 구현하고, 단발적인 SQL은 DQL이나 DML을 활용하는 것이 가장 효과적임을 인식한다. 결국 응용시스템에서 요구하는 기능 또는 서비스의 수준을 DBMS 및 구현된 시스템 모델링 관점에서 판단하는 것이 중요하다.

(3) 사용자 함수를 작성해야 할지를 판단한다.

조회 또는 입력하는 항목 중 산출 과정이 다소 복잡하고 별도의 관리가 필요하다고 판단되는 경우, 이를 위한 사용자 함수를 작성하는 것까지 포함하여 설계한다. 다만, 무분별한 사용자 함수의 사용은 SQL 실행 효율을 저하시킬 수 있기 때문에 주의하고, 공통적으로 사용되는 횟수가 높고 사용자 함수가 포함되는 결과 ROW가 적을수록 사용자 함수의 활용이 관리 · 기술적 측면에서 효율적이다.

(4) MERGE를 잘 활용한다.

INSERT와 UPDATE를 모두 필요로 한다고 하더라도 데이터를 저장하는 대상의 중복 여부와 다중 트랜잭션 기능인지를 판단하여, 가능한 경우 MERGE SQL을 활용하여 사용하는 것이 효과적이다. MERGE SQL이 DBMS 내에서 INSERT 또는 UPDATE보다 효율적인 수행이 가능하고 작성되는 SQL 분량도 적어지기 때문이다.

## 2. 구현을 위한 구체적 SQL을 설계한다.

(1) 단순한 구조를 유지할 수 있도록 설계한다.

SQL 전체적으로는 단순한 단일 기능 수행을 위주로 하여 실행되도록 설계한다. 가독성 측면뿐만 아니라 추후 모듈(Module) 측면에서의 재활용성을 고려할 때, 가급적 단순하게 작성하고 이를 유지하는 것은 중요하다. 응용 프로그램에서 사용되는 SQL은 DB 관리도구에서 일회성으로 사용되고 곧바로 폐기될 수 있는 단발성 SQL과는 다르게 해당 응용 프로그램의 폐기 시까지 사용되는 것을 전제로 함을 숙지한다.

(2) 관련된 테이블을 확인한다.

DQL 또는 DML 대상이 되는 모든 관련 테이블을 확인한다. 두 개 이상의 테이블을 동시에 활용해야 하는 경우 테이블 간의 조인(JOIN) 관계를 확인하여 기본키(PK: Primary Key)와 외래키(FK: Foreign Key)를 참조로 하여 SQL을 작성한다. 서브쿼리(Sub Query)를 사용하는 경우도 유사하다.

(3) 부모-자식 관계 간의 테이블에 삽입하는 경우에는 순서를 확인한다.

부모, 자식 테이블에 모두 데이터를 삽입하는 경우에는 반드시 부모 테이블에 먼저 데이터를 삽입한 후 자식 테이블에 삽입한다. DBMS에 실제 케스케이드(CASCADE) 관계 제약이 적용되어 있는 경우에 자식 테이블부터 먼저 INSERT한다면 SQL 실행 시 오류가 발생한다. 이와 같은 경우 우선 처리 순위를 분명히 하는 태도를 습득한다.

(4) 데이터를 확인한다.

서비스 처리에 필요한 모든 데이터를 확인한다. 데이터의 값뿐만 아니라 유형(날짜, 문자열, 숫자 등)과 같이 사용에 필요한 속성을 확인한다.

(5) 입력변수(Input Parameter)를 확인한다.

SQL에 포함되는 데이터 항목 중에 응용시스템의 유저 인터페이스(UI: User Interface, 일반적으로 화면을 뜻하기도 함.)를 통해서 입력받는 항목을 선별한다. 분석 과정에서 확인한 가변값에 매칭되는 데이터를 구분하는 것이다. 이때 응용 프로그램을 통해서 입력받는 항목은 별도로 리스트 등을 통해 인지할 수 있도록 관리한다.

③ SQL을 작성하고 DB 관리도구를 통해 실행한다.

1. 설계된 SQL을 작성한다.

(1) DQL, DML을 활용하여 SQL문을 작성한다.

관련 테이블 및 데이터 간의 관계 및 값을 유의하여 SQL을 작성한다. SELECT, INSERT, UPDATE, DELETE가 위주가 될 것이며, 서비스 및 기능 특성에 따라 사용자 정의함수나 프로시저를 포함하는 경우는 이를 먼저 작성한다. 절차형 SQL을 작성하는 경우에는 당연히 이를 호출하는 SQL까지 작성해야 한다.

(2) 입력변수(Input Parameter)를 구별한다.

응용시스템의 유저 인터페이스(UI : User Interface)를 통해 입력받는 항목 부분이다. 이 부분을 실제 들어올 만한 값으로 미리 입력하고 SQL을 실행해도 좋으나, 가급적 입력변수로만 처리해 놓고 실행 시 입력하는 방향으로 작성한다. 만약 SQL을 빈번하게 실행하며 작성하는 상황이라면 미리 입력하는 것도 무방하나, 이런 경우에도 가급적 마지막 단계에는 입력변수로 설정하도록 한다. DB 관리도구마다 입력변수 표시(아래에서의 :IN\_YEAR) 방법은 상이하므로 이 부분은 별도로 확인해야 하며, 이에 대한 문장의 비교는 다음과 같다.

<표 2-13> Binding을 하지 않은 SQL(상)과 Binding을 한 SQL(하)

```
SELECT YEAR  
    , SUM(SALES_AMT) AS SUM_SALES_AMT  
  FROM SALES_LIST  
 WHERE YEAR = '2017';  
  
-----  
SELECT YEAR  
    , SUM(SALES_AMT) AS SUM_SALES_AMT  
  FROM SALES_LIST  
 WHERE YEAR = :IN_YEAR ;
```

(3) DML을 작성하는 경우, 데이터 변경 확인 및 데이터 원상 복구를 위한 DQL을 작성한다.

(가) 데이터 변경 확인을 위한 DQL을 작성한다.

DML을 수행하면 테이블 내의 데이터가 변경된다. 대개의 DB 관리도구에서는 DML 수행 결과를 간단히 ROW 단위로 표시해 주는데, 이것만으로도 작업이 되었음을 알 수는 있으나, 그다음 실제 데이터의 변경된 내역을 직접 확인할 수 있는 SELECT 문이 필요하다.

(나) 데이터 원상 복구를 위한 DML을 작성한다.

필요한 경우, 재테스트 또는 원상 복구 이후 다른 테스트를 위해 수행한 DML을 되돌릴 수 있는 DML을 추가로 작성하고, 작성된 SQL 실행 후 원상 복구를 위한 DML을 실행한다.

2. 작성된 SQL을 실행한다.

(1) 작성된 SQL을 실행한다.

DB 관리도구의 기능을 활용하여 SQL을 실행한다.

(2) 필요시 입력변수를 입력한다.

입력변수에 맞는 실제값을 기재하여 실행한다. DB TOOL에 따라 실행 시마다 입력 변수값을 넣는 경우와 실행 창마다 입력변수를 사전에 입력하고 실행하는 경우가 있다.

(3) 실행 오류가 발생할 경우 이에 맞게 대응한다.

만약 실행 중 오류가 발생하는 경우에는 실행 오류 메시지를 참조하여 오류 원인을 분석한다. 대개의 경우는 SQL 문장 오류로부터 기인하나, 경우에 따라 다른 데이터에 의해 발생하는 경우도 있다. 확인된 오류 원인을 제거하거나 해결하고, SQL을 재실행한다.

### 3. 실행 결과를 확인한다.

#### (1) 결과를 확인한다.

의도한 바와 같은 결과가 도출되었는지 확인한다.

#### (2) 의도하지 않은 결과 발생 시 이에 맞게 대응한다.

조회하고자 하는 내용이 다르게 나오거나 다르게 수정되었을 경우, 데이터 구성 및 인과 관계 등을 참고로 하여 원인을 분석한다. 확인된 원인을 제거하거나 해결하고, SQL을 재실행한다.

## ④ 응용시스템의 관련 파일에 통합한다.

### 1. 작성된 SQL을 MyBatis의 XML 파일에 적재한다.

#### (1) MyBatis 파일을 작성한다.

다양한 설정을 입력하고, 기능을 간략히 표현할 수 있도록 명명한 SQL ID를 생성한다.

#### (2) SQL을 이식한다.

Text 복사 기능 등을 이용하여 DB TOOL에서 작성한 SQL문을 MyBatis 파일에 붙여 넣는다.

### 2. 응용시스템에서 실행 가능한 형태로 변환한다.

#### (1) 입력변수를 확인하고 형식을 조정한다.

MyBatis에서의 입력변수 입력 형식인 #{}를 이용하여 SQL의 입력변수를 대체한다.

#### (2) 필요시 조건문 등을 추가한다.

응용시스템에서의 변수에 따라 SQL 문장의 변경이 가능한 <if> 등이 필요한 경우 필요한 곳에 입력한다. XML Tag라서 종료 부분이 반드시 필요하다는 것을 숙지하고, 이때 이로 인해 SQL 문법이 틀어지지 않도록 주의한다.

#### (3) 가독성을 확보한다.

줄바꿈을 적용하고 SQL 구조를 좀 더 쉽게 이해할 수 있도록 하여 조건문이나 실행에 대한 직관성을 증대시킨다.

## ⑤ 응용시스템을 실행한다.

### 1. 응용시스템의 서비스를 기동한다.

응용시스템 서비스를 시작한다. MyBatis의 XML 내용은 서비스를 개시할 때 적용되므로, 변경 사항이 있을 경우 서비스를 재시작해야 한다.

## 2. 응용시스템을 실행한다.

응용시스템을 시작하여 의도한 작업 또는 서비스를 실행한다. UI를 통해서 실행하는 것이 일반적이나, 경우에 따라서는 타 Event로 실행이 가능하다.

## 3. 결과를 확인한다.

DBMS에 질의한 결과가 정상적으로 응용시스템에서 조회되는지 또는 응용시스템에서 저장한 내역이 정상적으로 DBMS에 반영되었는지 등을 확인한다. DBMS로 인한 오류가 발생할 경우 SQL 문장 및 Input Parameter, 결과 Return 방식 등을 확인하고 원인 제거 후 재 실행한다.

### 수행 tip

- 여기서는 응용시스템을 설치하고 정상적인 기능 구동을 위한 설치 과정은 생략하고 기본적 개념만 언급한다. 추가 지식이 필요한 경우 JAVA 및 관련 Frame Work 등을 찾아서 학습하도록 한다.
- SQL 문장이 단순하거나 실행의 무결성이나 결과의 정합성에 대한 확신이 있다면 수행 순서 중 ③번 과정은 생략해도 무방하다. 다만 문제가 발생하는 경우, 이를 위한 디버깅 작업에 소요되는 시간 및 노력은 ③번 과정을 수행했을 때가 항상 더 효율적임을 상기한다. 복합 작업은 나누어서 분석하는 것이 보통 더 효과적이다.

## 2-3. DCL 명령문 작성

### 학습 목표

- 사용자의 그룹을 정의하여 사용자를 생성 또는 변경할 수 있고 사용자의 권한 부여와 회수를 위한 DCL(Data Control Language) 명령문을 작성할 수 있다.

### 필요 지식 /

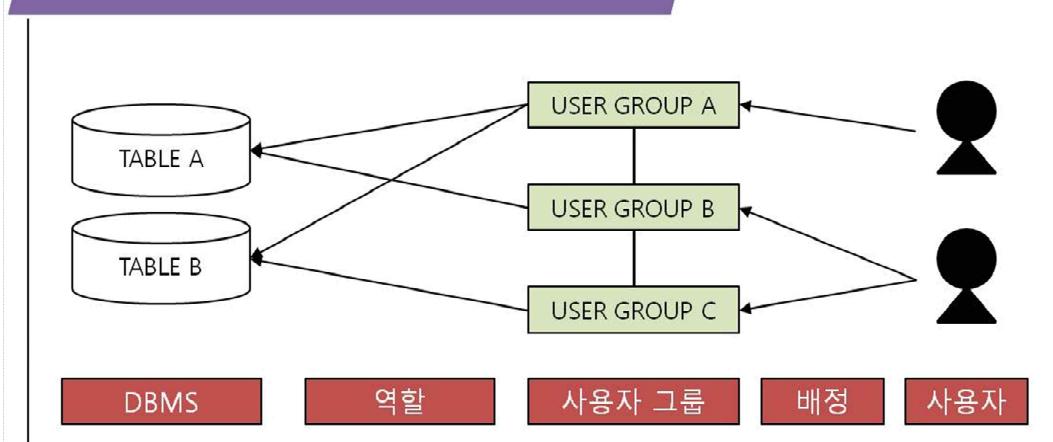
#### ① 사용자 그룹 정의

##### 1. 사용자 그룹 관리의 개념

업무상 회사 내 타부서 간 또는 외부 시스템 등과 데이터를 공유하기 위해 DB를 오픈하는 경우가 발생한다. 이런 경우 전체 DB에 대해 오픈하는 경우는 거의 없고, 필요로 따라 제한적인 DB만을 오픈한다. 여기서 해당 DB에 접속하여 사용하는 대상자를 사용자라고 하며, 다수의 사용자가 접속하는 사용자를 사용자 그룹이라고 한다. 사용자 그룹은 동일한 권한과 제약을 가지는 사용자들이 공통으로 사용하는 계정이다.

DBMS에서의 사용자 그룹의 관리는 역할기반 접근제어(RBAC: Role Based Access Control) 그룹 관리를 기반으로 한다. 사용자를 일일이 개별적으로 분할하지 않고 수행하는 역할을 기반으로 그룹핑하여 먼저 분할하고 사용자 그룹에 권한을 부여한다. 그 다음 개별 사용자를 사용자 그룹에 맞추어 배정하여 결국 사용자는 사용자 그룹에 속한 권한을 사용하게 한다.

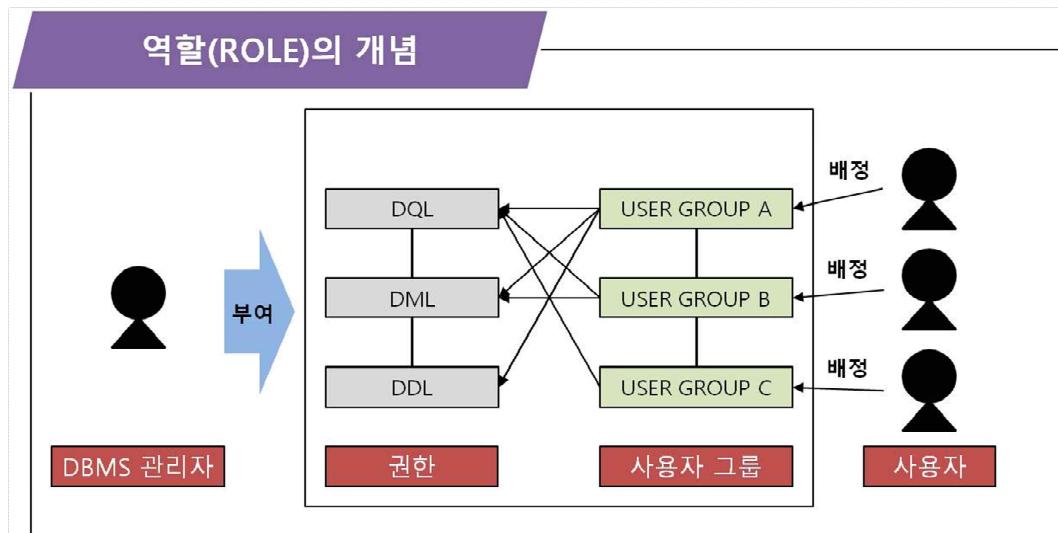
#### 역할기반 사용자그룹관리의 개념



[그림 2-7] 역할 기반 접근 제어(RBAC) 방식의 사용자 그룹 관리

## 2. 역할(ROLE) 부여를 통한 사용자 그룹 관리

사용자 그룹 관리 시 해당 사용자 그룹에 다양한 권한을 부여할 수 있다. 데이터 질의어(DQL: SELECT), 데이터 조작어(DML: INSERT, UPDATE, DELETE), 데이터 정의어(DDL: CREATE, ALTER, DROP) 모두 부여가 가능한데, 앞의 DQL 및 DML 뒤의 DDL은 분리하는 것이 일반적이다. DQL과 DML에 제한적 권한을 부여하더라도 DDL 권한이 있는 경우 DDL을 통해 생성한 테이블에 대해 오너(Owner)가 되기 때문에 이 부분이 혼동되는 경우가 많다. 또한 SQL 문장의 종류가 아주 많아서 일일이 설정하는 것이 매우 복잡하고 관리하는 것도 어렵다. 그렇기 때문에 일반 사용자 권한 개념으로 DQL, 중간 관리자 권한으로 DML까지 부여하고 DB관리자 권한 개념으로 DDL을 부여하는 방법이 사용된다. 역할(ROLE)은 DB에서 사용자 그룹과 권한 사이에서 중개 역할을 하며, 이를 통해 빠르고 정확하게 필요한 권한을 부여할 수 있게 한다.



[그림 2-8] 별도 권한(ROLE)을 사용자 그룹에게 부여하고 배정하는 개념

### ② 사용자 그룹 생성과 변경

#### 1. 사용자 그룹 생성

DBMS 제공 벤더에 따라 사용자 그룹 관리에 대해서는 SQL문 및 접근 관점이 많이 상이하다. O사에서는 사용자 그룹 로그인을 통해 DBMS에 접근하여 사용한다. 아이디와 비밀번호 방식으로 인스턴스 접속 후 그에 해당하는 권한을 받고 사용하는 것이다. S사는 인스턴스 접속을 위해 로그인이라는 것을 생성하고, 인스턴스 내 다수의 DBMS에 접근을 위해 사용자 그룹을 생성한 후 로그인과 사용자 그룹을 매핑하는 방식으로 사용한다.

사용자 그룹 생성을 위해서는 사용자 그룹 생성 권한이 있어야 하며, O사와 S사 SQL이 다르다. O사의 경우에는 사용자 그룹을 생성하고 난 다음, 사용자 그룹에 Session 생성 권한도 별도로 부여해야 한다. 그 이후 해당 사용자 그룹으로 로그인이 가능하다.

<표 2-14> O사 사용자 그룹 생성 SQL

```
CREATE USER [USER_GROUP_ID] IDENTIFIED BY [USER_GROUP_PASSWORD];
GRANT CREATE SESSION TO [USER_GROUP_ID];
```

<표 2-15> S사 사용자 그룹 생성 SQL

```
CREATE LOGIN [USER_GROUP_ID] WITH PASSWORD = [USER_GROUP_PASSWORD],
DEFAULT_DATABASE = [DATABASE_SESSION]
```

## 2. 사용자 그룹 변경

생성해 놓은 사용자 그룹을 삭제하거나 패스워드를 변경할 수 있다. 역시 O사와 S사가 각각 다르다.

<표 2-16> O사 사용자 그룹 변경 SQL

```
ALTER USER [USER_GROUP_ID] IDENTIFIED BY [USER_GROUP_PASSWORD];
```

## ③ 사용자 권한 부여 및 회수

### 1. 사용자 권한 부여

객체의 소유자는 객체에 대한 모든 권한을 가지며, 자신이 소유한 특정 권한을 GRANT문을 사용하여 다른 사용자에게 부여할 수 있다. 부여하는 권한에 따라 아래와 같이 분류가 가능하다.

#### (1) 데이터 질의어

SELECT 권한을 아래 양식과 같이 타 사용자 그룹에 부여한다. 이때 USER\_GROUP 대신 PUBLIC을 선택하면 특정 사용자 그룹이 아닌 전체 사용자 그룹에 공통적으로 적용이 가능하다. WITH GRANT OPTION의 경우 권한을 부여한 사용자 그룹이 또 다른 사용자 그룹에게 부여받은 권한을 재부여할 수 있다는 의미이다. 즉, 이 권한을 갖는 사용자들이 체인 형식으로 증가할 수 있는 것이다.

<표 2-17> O사 및 S사의 DQL 권한 부여 문법

```
GRANT SELECT ON [OBJECT] TO [USER_GROUP] (WITH GRANT OPTION)
```

#### (2) 데이터 조작어

INSERT, UPDATE, DELETE 권한을 타 사용자 그룹에 부여한다. 문법과 사용 방식 등은 DQL과 동일하며, WITH GRANT OPTION도 이와 동일하다. DML SQL 3가지는 기존 데이터를 변경한다는 점에서는 동일하므로, 일반적으로 3가지를 동시에 부여하는 경우가 많다. 물론 그중에서 선별적으로만 부여하는 것도 가능하다.

### (3) 데이터 정의어

CREATE, ALTER, DROP 등의 권한을 부여한다. DQL, DML 권한 부여 사용법과 거의 유사하나 대상 사용자 그룹을 기재한다는 점에서 다르다.

## 2. 사용자 권한 회수

사용자에게 부여된 권한을 다시 회수할 수 있다. 권한을 부여하는 것과 반대 개념이며, 사용하는 방법도 유사하다. REVOKE를 사용하여 회수하며, TO가 아닌 FROM으로 사용자 그룹을 선택한다. 아래와 같이 사용한다.

<표 2-18> O사 및 S사의 DQL 권한 회수 문법

```
REVOKE [PRIVILEGE] ON [OBJECT] FROM [USER_GROUP]
```

## 수행 내용 / DCL 명령문 작성하기

---

### 재료 · 자료

- 논리 E-R 다이어그램
- 물리 E-R 다이어그램
- 테이블 설계서
- 뷰 명세서
- 인덱스 설계서
- 데이터 표준 정의서
- 데이터 요구사항

### 기기(장비 · 공구)

- 컴퓨터
- DBMS(DataBase Management System) 설치 프로그램
- DBMS 클라이언트 프로그램

### 안전 · 유의 사항

- 실습 후 사용한 PC의 접속 종료

### 수행 순서

#### ① 요구사항을 정의하고 분석한다.

1. 구현하고자 하는 시스템에 요구되는 권한 내용에 대해 확인한다.
2. 전체 요구사항을 모두 취합한다.
3. 사용자별, 객체별, 역할별로 체계적으로 정리하여 쉽게 이해가 되도록 한다.

#### ② 작성 대상 사용자 그룹을 정의한다.

1. 수행의 주체가 되는 대상을 정리한다.
  - (1) 실제 사용자 그룹으로 생성하고 로그인해야 하는 수행 주체가 되는 대상을 선별한다.
  - (2) 수행 주체는 가급적 세분류로 분할하되, 추후 통합 여지를 남긴다.

2. 수행 주체 대상 간의 유사성을 비교한다.
  - (1) 수행 주체의 역할, 수행 기능을 위주로 정리한다.
  - (2) 유사성이 일정 수준 이상 도출되는 수행 주체 대상을 정리한다.
3. 사용자 그룹을 정의한다.
  - (1) 대상 수행 주체에 포함되는 수행 기능을 통합하여 하나로 생성한다.
  - (2) 사용자 그룹 명칭을 정한다.

[3] 사용자 그룹을 매핑한다.

1. 정의된 사용자 그룹을 나열하고 수행 기능도 표시한다.
  - (1) 횡으로 사용자 그룹을 나열하고 종으로 수행 기능을 나열한다.
  - (2) 사용자 그룹별 적용해야 하는 수행 기능을 표시한다.

<표 2-2> 사용자 그룹 매핑 표시 예시

구분	고객관리팀 개발자	상품개발팀 개발자	경영정보팀 개발자
고객 ID 정보	SELECT INSERT/UPDATE	SELECT	SELECT
고객 개인 정보	SELECT INSERT/UPDATE	-	-
상품 정보	SELECT	SELECT INSERT/UPDATE/DELETE	SELECT
상품 원가 정보	SELECT	SELECT INSERT/UPDATE/DELETE	SELECT
판매 정보	-	SELECT INSERT/UPDATE	SELECT
매출액 정보	-	SELECT INSERT/UPDATE	SELECT

2. 사용자 그룹 생성 및 권한 부여, 회수 테스트를 위한 시나리오를 작성한다.
  - (1) 사용자 그룹별 테스트할 내용을 사전에 정의한다.
  - (2) 권한 부여자와 권한 수여자, 두 가지 사용자 그룹을 통한 이중 테스트가 가능하도록 준비한다.

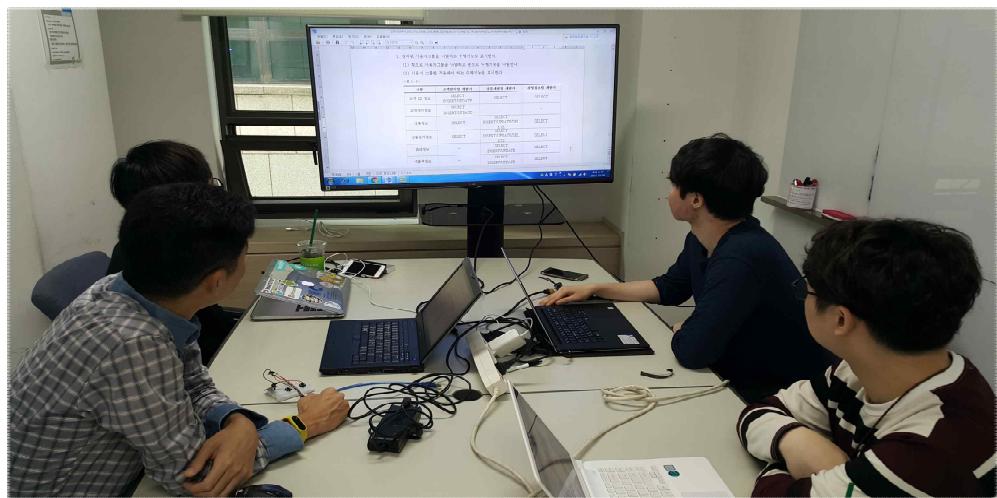
④ 생성된 사용자 그룹을 상호 검증한다.

1. 사용자 그룹 매핑사항을 공유한다.

사용자 그룹 매핑사항이 실제 그룹별 권한관리에서 가장 중요한 설계부분이며, 이 부분을 충분히 검증하고 이후 단계를 진행하는 것이 중요하다.

2. 필요시 회의 등을 통해서 수정, 보완한다.

메일이나 다른 커뮤니케이션을 통해서도 가능하나 관련자들이 모여서 작성된 사항을 공유하고 논의하여 수정, 보완하는 것이 가장 좋다.



[그림 2-9] 사용자 그룹 매핑 사항의 공유, 검토 회의 예시

⑤ 사용자 그룹을 생성한다.

1. 사용자 그룹 생성 명령문을 작성한다.

(1) 사용하는 DBMS 종류에 따른 사용자 그룹 생성 명령문을 확인한다.

(2) 사용자 그룹 생성 명령문 SQL을 작성한다.

2. 사용자 그룹 생성 명령문 실행

(1) 작성한 사용자 그룹 생성 명령문을 실행한다.

(2) 정상적으로 실행되지 않은 경우 에러 메시지를 확인하고 관련하여 필요한 조치를 행한다.

<표 2-19> O사 및 S사의 사용자 그룹 생성 예제

```
CREATE USER HR_DEPT IDENTIFIED BY HR_DEPT0236 ;  
GRANT CREATE SESSION TO HR_DEPT ;
```

```
-----  
CREATE LOGIN HR_DEPT WITH PASSWORD = HR_DEPT0236, DEFAULT_DATABASE = EMPLOYEE_INFO;
```

3. 사용자 그룹의 생성을 확인한다.

(1) 생성자가 사용자 그룹이 생성되었는지를 확인한다.

(2) 신규 생성된 사용자 그룹으로 로그인하여 정상 로그인이 되는지를 확인한다.

## ⑥ 사용자의 권한을 부여한다.

1. 권한 부여 명령문을 작성한다.

(1) 사용하는 DBMS 종류에 따른 권한 부여 명령문을 확인한다.

(2) 권한 부여 명령문 SQL을 작성한다.

2. 권한 부여 명령문을 실행한다.

(1) 작성한 권한 부여 명령문을 실행한다.

(2) 정상적으로 실행되지 않은 경우 에러 메시지를 확인하고 관련하여 필요한 조치를 행한다.

3. 권한 부여 상태를 확인한다.

(1) 권한이 정상적으로 부여되었는지 권한 부여자가 직접 확인한다.

(2) 권한을 부여받은 사용자 그룹으로 로그인하여 정상적으로 권한을 부여받았는지를 확인한다.

## ⑦ 사용자의 권한을 회수한다.

1. 권한 회수 명령문을 작성한다.

(1) 사용하는 DBMS 종류에 따른 권한 회수 명령문을 확인한다.

(2) 권한 회수 명령문 SQL을 작성한다.

2. 권한 회수 명령문을 실행한다.

(1) 작성한 권한 회수 명령문을 실행한다.

(2) 정상적으로 실행되지 않은 경우 에러 메시지를 확인하고 관련하여 필요한 조치를 행한다.

3. 권한 회수를 확인한다.

(1) 권한이 정상적으로 회수되었는지 권한 부여자가 직접 확인한다.

(2) 권한이 회수된 사용자 그룹으로 로그인하여 정상적으로 권한이 회수되었는지를 확인한다.

### 수행 tip

- 시스템 규모 및 업무 환경 등에 따라 다르지만 대개의 경우 사용자 그룹에는 DQL, DML까지만 부여하고, 그 이상의 DDL과 같은 SQL은 DBMS 관리자가 요청을 받아서 수행하는 경우가 대다수이다. 일반 사용자 그룹이 DDL을 사용하는 경우 테이블 생성 등이 자유로워져서 데이터 모델 관리가 쉽지 않기 때문이다.
- DML 권한을 부여하지 않는 것도 충분히 가능하나, 다른 방법으로 VIEW만 사용 가능한 사용자 그룹을 생성하고 관리하는 경우도 있다. 이 경우 테이블에 대한 접근이 직접 되지 않으며 특정 컬럼 등도 제약이 가능하므로 경우에 따라서는 유용하다.
- 결국 사이트의 상황 및 작업자의 역량 및 관리수준 등을 고려하여 부여가능한 ROLE을 정하고 운영하는 것이 가장 좋다.

## 학습 2 교수 · 학습 방법

### 교수 방법

- SQL에 대한 학습자의 이해 및 활용 수준을 확인하고, 주요 사항 및 TIP 위주로 설명한다.
- 학습자가 각 SQL 명령문 간에 연계되어 활용할 수 있도록 교육한다.
- 윈도우 함수와 그룹 함수를 혼합 사용하여 다양한 방법으로 순위와 소계 등 집합 결과를 구할 수 있음을 설명한다.
- 순위와 소계 등을 구하는 방법은 다양하나, 결과에서 구하고자 하는 양식이나 형태에 따라 가장 적합하고 효율적인 방법이 있을 수 있음을 설명한다.
- 응용시스템에서 사용하는 SQL 작성은 DBMS와 응용시스템 간 접속 설정이 필요함을 설명한다.
- 사용자 그룹 관리를 통해 DBMS 접근을 통제하여 데이터의 무결성 및 보안을 유지할 수 있음을 설명한다.
- 부여하는 ROLE의 수준 및 등급으로 인해 미칠 수 있는 영향 정도를 설명하고, 제약과 활용 측면에서 최적화된 관리 수준을 부여할 수 있음을 설명한다.

### 학습 방법

- SQL에 대한 이해 및 활용 수준을 상기하고, 주요 사항 및 TIP 위주로 숙지한다.
- 원활한 진행을 위한 기본적 SQL 활용의 사전 지식 습득은 필수이며, 각 SQL 명령문 간에 연계되어 활용할 수 있도록 숙지한다.
- 윈도우 함수와 그룹 함수를 혼합 사용하여 다양한 방법으로 순위와 소계 등 집합 결과를 구할 수 있음을 숙지한다.
- 순위와 소계 등을 구하는 방법은 다양하나, 결과에서 구하고자 하는 양식이나 형태에 따라 가장 적합하고 효율적인 방법이 있을 수 있음을 숙지한다.
- 응용시스템에서 사용하는 SQL 작성은 DBMS와 응용시스템 간 접속 설정이 필요함을 파악한다.

- 사용자 그룹 관리를 통해 DBMS 접근을 통제하여 데이터의 무결성 및 보안을 유지할 수 있음을 숙지한다.
- 부여하는 ROLE의 수준 및 등급으로 인해 미칠 수 있는 영향 정도를 이해하고, 제약과 활용 측면에서 최적화된 관리 수준을 부여할 수 있음을 학습한다.

## 학습 2 평 가

### 평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	학습 목표	성취수준		
		상	중	하
집계성 SQL 작성	- 원도우 함수와 그룹 함수를 사용하여 순위와 소계, 중계, 총합계를 산출하는 DML(Data Manipulation language) 명령문을 작성할 수 있다.			
특정 기능 수행 SQL문 작성	- 응용시스템에서 사용하는 특정 기능을 수행하기 위한 SQL문을 작성할 수 있다.			
DCL 명령문 작성	- 사용자의 그룹을 정의하고 사용자를 생성 또는 변경 할 수 있고 사용자의 권한 부여와 회수를 위한 DCL(Data Control Language) 명령문을 작성할 수 있다.			

### 평가 방법

- 문제 해결 시나리오

학습 내용	평가 항목	성취수준		
		상	중	하
집계성 SQL 작성	- 요구되는 사항에 대해 이를 명확히 정의 및 분류하고 관련 사항을 분석하여 해결을 위한 설계 구상 후 구현하고 검증이 가능하도록 처리할 수 있는 능력			
	- SQL 문장 실행 시 에러가 발생하면 그 에러를 참조로 하여 원인 파악 후 정상적인 실행이 가능하게 SQL문을 수정할 수 있는 능력			
	- SQL 문장 실행 시 기대하는 결과가 나오지 않았을 때, 원인을 파악한 후 SQL문을 수정하거나 기타 데이터를 조작할 수 있는 능력			
	- 단일 테이블 대상 데이터 조회, 서브쿼리나 조인을 사용하는 다수 테이블 대상 데이터 조회, DBMS 제공 함수나 사용자 정의함수를 사용하는 DQL 명령문을 작성 할 수 있는 능력			

특정 기능 수행 SQL문 작성	<ul style="list-style-type: none"> <li>- 요구되는 사항에 대해 이를 명확히 정의 및 분류하고 관련 사항을 분석하여 해결을 위한 설계 구상 후 구현하고 검증이 가능하도록 처리할 수 있는 능력</li> <li>- SQL 문장 실행 시 에러가 발생하면 그 에러를 참조하여 원인 파악 후 정상적인 실행이 가능하게 SQL문을 수정할 수 있는 능력</li> <li>- SQL 문장 실행 시 기대하는 결과가 나오지 않았을 때, 원인을 파악한 후 SQL문을 수정하거나 기타 데이터를 조작할 수 있는 능력</li> <li>- 단일 테이블 대상 데이터 조회, 서브쿼리나 조인을 사용하는 다수 테이블 대상 데이터 조회, DBMS 제공 함수나 사용자 정의함수를 사용하는 DQL 명령문을 작성 할 수 있는 능력</li> <li>- 응용시스템에서의 변수입력 도출 및 올바른 결과전달을 통해 응용시스템에서의 정상적 SQL 실행을 하게 할 수 있는 능력</li> </ul>		
	<ul style="list-style-type: none"> <li>- 요구되는 사항에 대해 이를 명확히 정의 및 분류하고 관련 사항을 분석하여 해결을 위한 설계 구상 후 구현하고 검증이 가능하도록 처리할 수 있는 능력</li> <li>- SQL 문장 실행 시 에러가 발생하면 그 에러를 참조하여 원인 파악 후 정상적인 실행이 가능하게 SQL문을 수정할 수 있는 능력</li> <li>- SQL 문장 실행 시 기대하는 결과가 나오지 않았을 때, 원인을 파악한 후 SQL문을 수정하거나 기타 데이터를 조작할 수 있는 능력</li> </ul>		
DCL 명령문 작성	<ul style="list-style-type: none"> <li>- 다양한 방법으로 구성 및 설계가 가능한 사용자 그룹과 수행 기능을 적절히 생성하고 매핑하는 설계 수준</li> <li>- 사용자 그룹의 단순성 및 확장 가능성 설계 수준</li> </ul>		
	<ul style="list-style-type: none"> <li>- 그룹 함수를 사용하여 필요한 데이터를 조회하는 능력</li> <li>- 윈도우 함수를 사용하여 필요한 데이터를 조회하는 능력</li> <li>- SQL 실행 시 오류가 발생하지 않고 정상적으로 실행 및 결과가 산출되도록 하는 능력</li> <li>- 결과 검증을 통해 기대하는 결과를 도출하는 능력</li> </ul>		

• 평가자 체크리스트

학습 내용	평가 항목	성취수준		
		상	중	하
DML 명령문 작성	- 그룹 함수를 사용하여 필요한 데이터를 조회하는 능력			
	- 윈도우 함수를 사용하여 필요한 데이터를 조회하는 능력			
	- SQL 실행 시 오류가 발생하지 않고 정상적으로 실행 및 결과가 산출되도록 하는 능력			
	- 결과 검증을 통해 기대하는 결과를 도출하는 능력			

특정 기능 수행 SQL문 작성	<ul style="list-style-type: none"> <li>- 응용시스템에 적용할 SQL 문법을 이해하고 적용하는 능력</li> </ul>		
	<ul style="list-style-type: none"> <li>- 응용시스템의 입력변수를 적절히 선별하여 SQL 문법에 적용하는 능력</li> </ul>		
	<ul style="list-style-type: none"> <li>- 응용시스템 실행 시 DBMS 오류 없이 실행 결과가 나오도록 하는 능력</li> </ul>		
	<ul style="list-style-type: none"> <li>- 응용시스템 실행 시 기대하는 결과값이 정상적으로 도출되도록 하는 능력</li> </ul>		
DCL 명령문 작성	<ul style="list-style-type: none"> <li>- 수행 주체를 정확하게 분석하여 적합한 사용자 그룹으로 생성하는 능력</li> </ul>		
	<ul style="list-style-type: none"> <li>- 수행 기능을 정의하고 이를 사용자 그룹과 적절하게 매핑하여 권한을 배정하는 능력</li> </ul>		
	<ul style="list-style-type: none"> <li>- DCL 명령문을 정상적으로 작성하여 오류가 발생하지 않고 정상적으로 실행되게 하는 능력</li> </ul>		
	<ul style="list-style-type: none"> <li>- 권한 부여 현황 검증 시 기대하는 결과를 도출하는 능력</li> </ul>		

## 피드백

### 1. 문제 해결 시나리오

- 집계 함수 SQL 및 일반적인 데이터 질의어(DQL)에 익숙한 수준이 되어야 원활한 그룹 함수 SQL 활용이 가능하다. 사전 단계에 미숙한 경우 그 부분을 좀 더 학습하여 그룹 및 윈도우 함수를 활용하는 데 부족함이 없도록 한다.
- 사용자 그룹 생성 및 권한 부여는 요구사항 층계 관점에 뜯지않게 사용자 그룹 구성의 적절성이 중요한 항목이다. 사용자 그룹 구성의 적절성에는 사용자 그룹의 단순성, 확장 가능성 및 관리 편의성 등이 포함된다.

### 2. 평가자 체크리스트

- 그룹 함수와 윈도우 함수의 SQL 작성 여부를 확인하고, 이해가 부족한 경우 부족한 부분을 보완하도록 한다.
- 응용시스템 실행 시 오류가 나는 경우, DBMS로부터 발생하는 경우도 있지만 그 외의 원인으로 인해 발생하는 경우도 많다. 진행하는 학습 범위를 검토하여, 응용시스템을 포함하여 체크하는 경우가 아니라면 실행 시 발생하는 SQL 오류 메시지에 좀 더 집중하여 학습하도록 진행한다.
- 작성한 SQL의 실행 시 오류가 나는 경우, 또는 기대한 결과가 나오지 않을 경우 원인을 파악하며 정상적 결과를 산출하기 위한 과정을 숙지시키고 보완하도록 한다.
- 사용자 그룹 생성 및 권한 부여에서는 SQL 오류보다 권한이 정상적으로 부여 또는 회수되지 않은 경우가 많다. 권한 설계에서는 매트릭스 형태의 설계가 중요함을 인식시킨다.

# 참고자료

---



- 김승기(2016). 『데이터베이스 배움터』. 생능출판.
- 김태근(2012). 『오라클 SQL & PL/SQL』. 프리렉.
- 마크(2015). 『SQL 더 쉽게, 더 깊게』. 제이펍.
- 전병선(2014). 『All-in-One JAVA 애플리케이션 개발』. 와우북스.
- 한국데이터베이스진흥원(2011). 『SQL 전문가 가이드』. 한국데이터베이스진흥원. p.255.
- 한국데이터베이스진흥원(2013). 『SQL 전문가 가이드(The Guide for SQL Professional)』.
- T-SQL. <https://docs.microsoft.com/ko-kr/sql/t-sql/language-reference>에서 2017. 06. 06. 검색.
- Stored Procedure. [https://en.wikipedia.org/wiki/Stored\\_procedure](https://en.wikipedia.org/wiki/Stored_procedure)에서 2017. 06. 06. 검색.



## NCS학습모듈 개발이력

발행일	2015년 12월 31일		
세분류명	DB엔지니어링(20010204)		
개발기관	한국소프트웨어기술진흥협회, 한국직업능력개발원		
집필진	편홍열(에이비엔아이)* 강성구(명지대학교) 김승현(경희대학교) 박미화(투이컨설팅) 박준자(한국오라클) 임영섭(비투엔컨설팅) 장온순(한국IT컨설팅) 장인혁(청운)	검토진	김보운(이화여자대학교) 여권동(NDS시스템) 정금묵(베이스존) 주선태(T3Q) 진권기(이비스톰)
			* 표시는 대표집필자임
발행일	2017년 12월 31일		
학습모듈명	SQL응용(LM2001020414_16v3)		
개발기관	(사)한국정보통신기술사협회, 한국직업능력개발원		
집필진	김광국((주)코스콤)* 김동욱(한국기술교육대학교) 김준범(SK주식회사) 송정현((주)씨에이에스)	검토진	이주연((주)씨에이에스) 장경애((사)한국정보통신기술사협회)
			* 표시는 대표집필자임
발행일	2018년 12월 31일		
학습모듈명	SQL응용(LM2001020414_16v3)		
개발기관	한국직업능력개발원		

## SQL응용(LM2001020414\_16v3)

저작권자	교육부
연구기관	한국직업능력개발원
발행일	2018.12.31

\* 이 학습모듈은 자격기본법 시행령(제8조 국가직무능력표준의 활용)에 의거하여 개발하였으며, NCS통합포털사이트(<http://www.ncs.go.kr>)에서 다운로드 할 수 있습니다.



[www.ncs.go.kr](http://www.ncs.go.kr)