

대분류/20
정보통신

중분류/01
정보기술

소분류/02
정보기술개발

세분류/04
DB엔지니어링

능력단위/13

NCS학습모듈

SQL 활용

LM2001020413_19v4



교육부

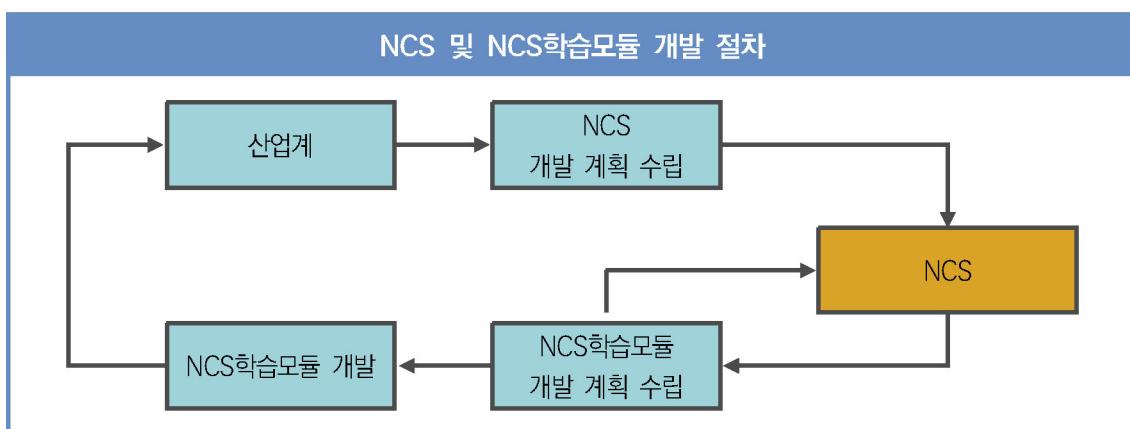
NCS 학습모듈은 교육훈련기관에서 출처를 명시하고 교육적 목적으로 활용할 수 있습니다. 다만 NCS 학습모듈에는 국가(교육부)가 저작재산권 일체를 보유하지 않은 저작물들(출처가 표기되어 있는 도표, 사진, 삽화, 도면 등)이 포함되어 있으므로 이러한 저작물들의 변형, 복제, 공연, 배포, 공중 송신 등과 이러한 저작물을 활용한 2차 저작물의 생성을 위해서는 반드시 원작자의 동의를 받아야 합니다.

NCS학습모듈의 이해

※ 본 NCS학습모듈은 「NCS 국가직무능력표준」사이트(<http://www.ncs.go.kr>)에서 확인 및 다운로드할 수 있습니다.

I NCS학습모듈이란?

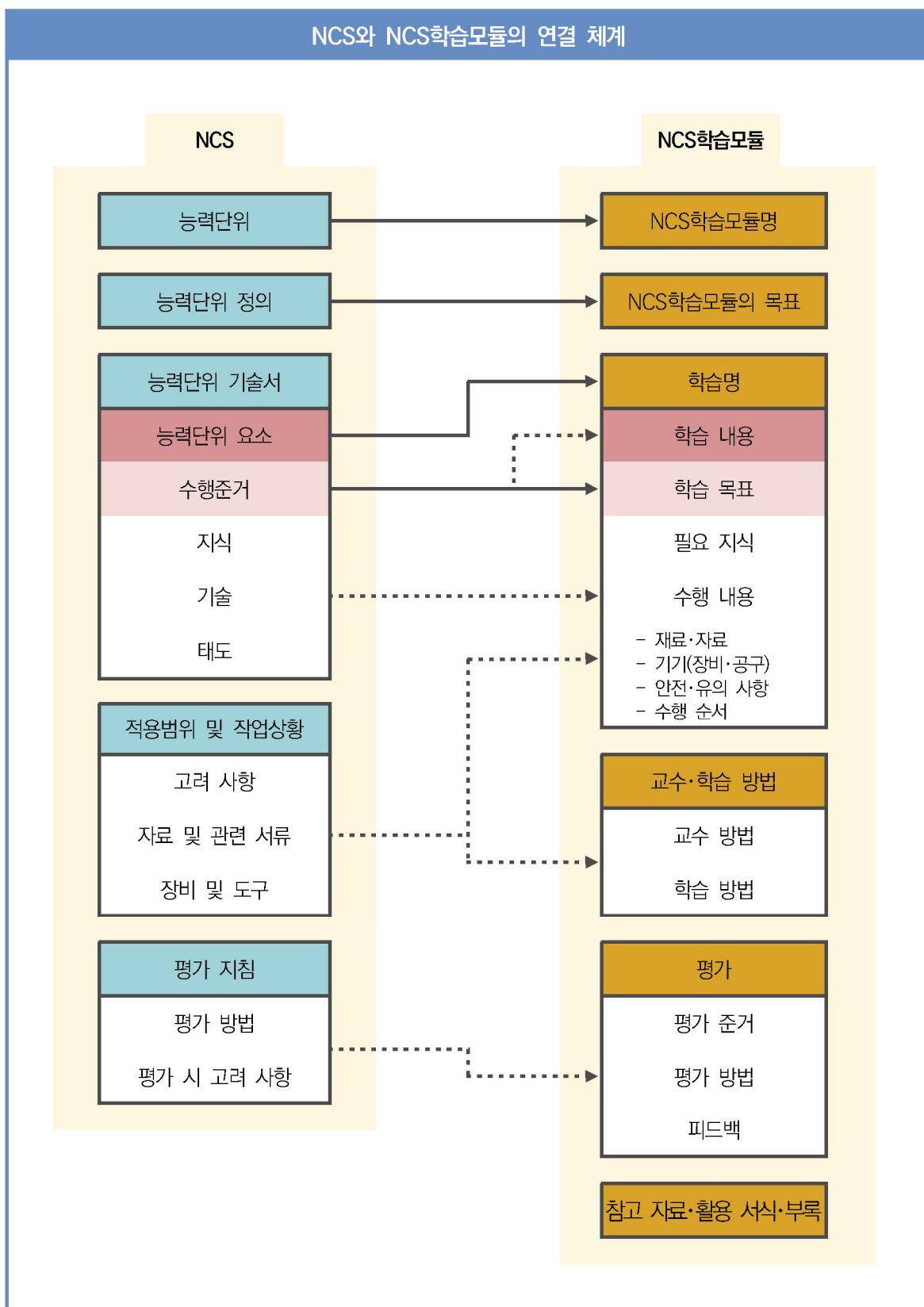
- 국가직무능력표준(NCS: National Competency Standards)이란 산업현장에서 직무를 수행하기 위해 요구되는 지식·기술·소양 등의 내용을 국가가 산업부문별·수준별로 체계화한 것으로 산업현장의 직무를 성공적으로 수행하기 위해 필요한 능력(지식, 기술, 태도)을 국가적 차원에서 표준화한 것을 의미합니다.
- 국가직무능력표준(이하 NCS)이 현장의 ‘직무 요구서’라고 한다면, **NCS학습모듈은 NCS의 능력단위를 교육훈련에서 학습할 수 있도록 구성한 ‘교수·학습 자료’입니다.** NCS학습모듈은 구체적 직무를 학습할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



- **NCS학습모듈은 다음과 같은 특징을 가지고 있습니다.**

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

○ NCS와 NCS학습모듈 간의 연결 체계를 살펴보면 아래 그림과 같습니다.



II NCS학습모듈의 체계

- NCS학습모듈은 1. NCS학습모듈의 위치 , 2. NCS학습모듈의 개요 , 3. NCS학습모듈의 내용 체계 , 4. 참고 자료 , 5. 활용서식/부록 으로 구성되어 있습니다.

1. NCS학습모듈의 위치

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

[NCS-학습모듈의 위치]		
대분류	문화·예술·디자인·방송	
중분류	문화콘텐츠	
소분류	문화콘텐츠제작	
세분류		
방송콘텐츠제작	능력단위	학습모듈명
영화콘텐츠제작	프로그램 기획	프로그램 기획
음악콘텐츠제작	아이템 선정	아이템 선정
광고콘텐츠제작	자료 조사	자료 조사
게임콘텐츠제작	프로그램 구성	프로그램 구성
애니메이션 콘텐츠제작	캐스팅	캐스팅
만화콘텐츠제작	제작계획	제작계획
캐릭터제작	방송 미술 준비	방송 미술 준비
스마트문화앱 콘텐츠제작	방송 리허설	방송 리허설
영사	야외촬영	야외촬영
	스튜디오 제작	스튜디오 제작

학습모듈은

NCS 능력단위 1개당 1개의 학습모듈 개발을 원칙으로 합니다. 그러나 필요에 따라 고용단위 및 교과단위를 고려하여 능력단위 몇 개를 묶어 1개 학습모듈로 개발할 수 있으며, NCS 능력단위 1개를 여러 개의 학습모듈로 나누어 개발할 수도 있습니다.

2. NCS학습모듈의 개요

○ NCS학습모듈의 개요는 학습모듈이 포함하고 있는 내용을 개략적으로 설명한 것으로

학습모듈의 목표, 선수학습, 학습모듈의 내용 체계, 핵심 용어로 구성되어 있습니다.

학습모듈의 목표

해당 NCS 능력단위의 정의를 토대로 학습 목표를 작성한 것입니다.

선수학습

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 체계를 제시한 것입니다.

핵심 용어

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

제작계획 학습모듈의 개요

학습모듈의 목표

본격적인 촬영을 준비하는 단계로서, 촬영 대본을 확정하고 제작 스태프를 조직하며 촬영 장비와 촬영 소품을 준비할 수 있다.

선수학습

제작 준비(LM0803020105_13v1), 섭외 및 제작스태프 구성(LM0803020104_13v1), 촬영 제작(LM0803020106_13v1), 촬영 장비 준비(LM0803040204_13v1.4), 미술 디자인 협의하기(LM0803040203_13v1.4)

학습모듈의 내용체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 촬영 대본 확정하기	1-1. 촬영 구성안 검토와 수정	0803020114_16.3.1	촬영 대본 확정하기
2. 제작 스태프 조직하기	2-1. 기술 스태프 조직 2-2. 미술 스태프 조직 2-3. 전문 스태프 조직	0803020114_16.3.2	제작 스태프 조직하기
3. 촬영 장비 계획하기	3-1. 촬영 장비 점검과 준비	0803020114_16.3.3	촬영 장비 계획하기
4. 촬영 소품 계획하기	4-1. 촬영 소품 목록 작성 4-2. 촬영 소품 제작 의뢰	0803020114_16.3.4	촬영 소품 계획하기

핵심 용어

촬영 구성안, 제작 스태프, 촬영 장비, 촬영 소품

학습모듈의 목표는

학습자가 해당 학습모듈을 통해 성취해야 할 목표를 제시한 것으로, 교수자는 학습자가 학습모듈의 전체적인 내용흐름을 파악하도록 지도할 수 있습니다.

선수학습은

교수자 또는 학습자가 해당 학습모듈을 교수·학습하기 이전에 이수해야 하는 교과목 또는 학습모듈(NCS 능력단위) 등을 표기한 것입니다. 따라서 교수자는 학습자가 개별 학습, 자기 주도 학습, 방과 후 활동 등 다양한 방법을 통해 이수할 수 있도록 지도하는 것을 권장합니다.

핵심 용어는

해당 학습모듈을 대표하는 주요 용어입니다. 학습자가 해당 학습모듈을 통해 학습하고 평가받게 될 주요 내용을 알 수 있습니다. 「NCS 국가직무능력표준」 사이트 (www.ncs.go.kr)의 색인 (찾아보기) 중 하나로 이용할 수 있습니다.

3. NCS학습모듈의 내용 체계

○ NCS학습모듈의 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 내용을 제시한 것입니다.
학습 내용	학습 내용은 학습 목표, 필요 지식, 수행 내용으로 구성되며, 수행 내용은 재료·자료, 기기(장비·공구), 안전·유의 사항, 수행 순서, 수행 tip으로 구성한 것입니다. 학습모듈의 학습 내용은 실제 산업현장에서 이루어지는 업무활동을 표준화된 프로세스에 기반하여 다양한 방식으로 반영한 것입니다.
교수·학습 방법	학습 목표를 성취하기 위한 교수자와 학습자 간, 학습자와 학습자 간 상호 작용이 활발하게 일어날 수 있도록 교수자의 활동 및 교수 전략, 학습자의 활동을 제시한 것입니다.
평가	평가는 해당 학습모듈의 학습 정도를 확인할 수 있는 평가 준거 및 평가 방법, 평가 결과의 피드백 방법을 제시한 것입니다.

학습 1	촬영 대본 확정하기	학습 2 제작 스태프 조직하기	학습은 해당 NCS 능력단위요소 명칭을 사용하여 제시하였습니다. 하나의 학습은 일반교과의 '대단원'에 해당되며, 학습모듈을 구성하는 가장 큰 단위가 됩니다. 또한 하나의 직무를 수행하기 위한 가장 기본적인 단위로 사용할 수 있습니다.
학습 3	촬영 장비 계획하기		
학습 4	촬영 소품 계획하기		

2-1. 기술 스태프 조직

학습 목표 • 프로그램 제작에 적합한 기술 스태프를 조직할 수 있다.

필요 지식

① 기술 스태프의 구성
프로그램의 장르에 따라 구성하는 기술 스태프는 많은 차이가 있다. 같은 장르의 프로그램이라도 그 형식이나 내용, 규모에 따라서 구성되는 기술 스태프의 종류와 인원 수는 천차만별이다.

1. 스튜디오 프로그램
토크쇼, 종합 구성, 예능과 같은 스튜디오 프로그램은 부조정실과 스튜디오를 사용하여 제작하기 때문에 많은 기술 스태프가 필요하다.

학습 내용은
NCS 능력단위요소별 수행준거를 기준으로 제시하였습니다. 일반교과의 '중단원'에 해당합니다.

학습 목표는
학습 내용을 이수할 때 학습자가 갖춰야 할 행동 수준을 의미합니다. 따라서 수업시간의 과목 목표로 활용할 수 있습니다.

필요 지식은
해당 NCS의 지식을 토대로 학습에 대한 이해와 성과를 제고하기 위해 반드시 알아야 할 주요 지식을 제시하였습니다. 필요 지식은 수행에 꼭 필요한 핵심 내용을 위주로 제시하여 교수자의 역할이 매우 중요하며, 이후 수행 순서와 연계하여 교수·학습으로 진행할 수 있습니다.

수행 내용	/ 기술 스태프 구성표 작성하기	
<p>재료·자료</p> <ul style="list-style-type: none"> 방송프로그램 제작 기획서 및 방송 대본, 콘티(continuity), 제작 일정, 운용표 장비 및 시설, 제작 시설 배정 의뢰서 및 배정표, 방송 기술 스태프 데이터베이스(DB) 자료 <p>기기(장비·공구)</p> <ul style="list-style-type: none"> 컴퓨터 등 <p>안전·유의 사항</p> <ul style="list-style-type: none"> 프로그램의 내용과 제작 방법을 분석하고, 각 스태프들의 역할을 신중하게 검토한다. <p>수행 순서</p> <ol style="list-style-type: none"> 방송 대본이나 콘티(continuity), 큐 시트를 분석하고, 프로그램의 내용적 특성, 제작 과정에 대한 자료를 수집한다. 프로그램 제작 방법을 결정한다. <ul style="list-style-type: none"> 스튜디오 녹화를 할 것인가, 야외 촬영을 할 것인가 검토한다. <p>수행 tip</p> <ul style="list-style-type: none"> 스태프의 결정은 스태프 간의 호흡을 중요시하여 선정해야 프로그램의 질을 향상시킬 수 있다. 		<p>수행 내용은</p> <p>해당 학습모듈에서 제시한 내용 중 기술(skill)을 습득하기 위한 실습과제로 활용할 수 있습니다.</p> <p>재료·자료는</p> <p>수행 내용을 수행하는데 필요한 재료 및 준비물로 실습 시 활용할 수 있습니다.</p> <p>기기(장비·공구)는</p> <p>수행 내용에 필요한 기본적인 장비 및 도구를 제시하였습니다. 제시된 기기 외에도 수행에 필요한 다양한 도구나 장비를 활용할 수 있습니다.</p> <p>안전·유의사항은</p> <p>수행 내용을 수행하는 데 있어 안전상 주의해야 할 점 및 유의사항을 제시하였습니다. 실습 시 유념해야 하며, NCS의 고려사항도 추가적으로 활용할 수 있습니다.</p> <p>수행 순서는</p> <p>실습 과제의 진행 순서로 활용할 수 있습니다.</p> <p>수행 tip은</p> <p>수행 내용에서 실습을 용이하게 할 수 있는 아이디어를 제시하였습니다. 수행 tip은 지도상의 안전 및 유의사항 외에 전반적으로 적용되는 주인점 및 수행 과제 목적에 대한 보충설명, 추가사항 등으로 활용할 수 있습니다.</p>
<p>학습2</p> <p>교수·학습 방법</p> <p>교수 방법</p> <ul style="list-style-type: none"> 방송 프로그램의 기술적 요소, 미술 구성 요소, 특수 촬영에 대해서 설명한다. 방송 프로그램 제작에서 각 기술 스태프의 역할에 대해 설명한다. 방송 프로그램을 분석하고 필요한 기술 스태프를 구성할 수 있도록 지시한다. <p>학습 방법</p> <ul style="list-style-type: none"> 방송 프로그램의 기술적 요소, 미술 구성 요소, 특수 촬영에 대해서 알아본다. 프로그램 제작에 필요한 기술 스태프의 역할을 이해하고, 기술 스태프 구성표를 작성한다. 		<p>교수·학습 방법은</p> <p>학습 목표를 성취하는 데 필요한 교수 방법과 학습 방법을 제시하였습니다.</p> <p>교수 방법은</p> <p>해당 학습 활동에 필요한 학습 내용, 학습 내용과 관련된 자료명, 자료 형태, 수행 내용의 진행 방식 등에 대하여 제시하였습니다. 또한 학습자의 수업참여도 제고 방법 및 수업 진행상 유의사항 등도 제시하였습니다. 선수학습이 필요한 학습을 학습자가 숙지하였는지 교수자가 확인하는 과정으로 활용할 수도 있습니다.</p> <p>학습 방법은</p> <p>해당 학습 활동에 필요한 학습자의 자기 주도 학습 방법을 제시하였습니다. 또한 학습자가 숙달해야 할 실기 능력과 학습 과정에서 주의해야 할 사항 등도 제시하였습니다. 학습자가 학습을 이수하기 전 반드시 숙지해야 할 기본 지식을 학습하였는지 스스로 확인하는 과정에 활용할 수 있습니다.</p>

학습2 평가

평가 준거

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

학습 내용	학습 목표	성취수준 상 중 하
기술 스태프 조직	- 프로그램 제작에 적합한 기술 스태프를 조직할 수 있다.	
미술 스태프 조직	- 프로그램 제작에 적합한 미술 스태프를 조직할 수 있다.	
전문 스태프 조직	- 프로그램 특수 촬영을 위한 전문 스태프를 조직할 수 있다.	

평가 방법

- 사례 연구

학습 내용	평가 항목	성취수준 상 중 하
기술 스태프 조직	- 프로그램에서 기술적 요소의 파악 여부 - 기술 스태프의 역할 파악 여부 - 프로그램에 필요한 기술 스태프 구성표 작성 능력	

피드백

- 사례 연구
 - 프로그램을 선택하여 기술 스태프, 미술 스태프, 전문 스태프 구성표를 예시와 같이 작성하였는지 개인별 능력을 평가한 후, 그 결과를 모든 학습자에게 공유하도록 한다.

평가는

NCS 능력단위의 평가 방법과 평가 시 고려사항을 종용하여 작성합니다. 교수자와 학습자가 평가 항목별 성취수준 확인 시 활용할 수 있습니다.

평가 준거는

학습자가 학습을 어느 정도 성취하였는지 평가하기 위한 기준을 제시하고 있습니다. 학습 목표와 연계하여 단위수업 시간에 평가 항목 별 성취수준을 평가하는 데 활용할 수 있습니다.

평가 방법은

NCS 능력단위의 평가 방법을 참고하였으며, 평가 준거에 따른 평가 방법을 2개 이상 제시합니다. 평가 방법의 종류는 포트폴리오, 문제해결 시나리오, 서술형 시험, 논술형 시험, 사례 연구, 평가자 체크리스트, 작업장 평가 등이 있으며, NCS 능력단위 요소 별 수행 수준을 평가하는 데 가장 적절한 방법을 선정하여 활용할 수 있습니다.

피드백은

평가 후에 학습자들에게 평가 결과를 피드백하여 학습 목표를 달성하는 데 활용할 수 있습니다.

4. 참고 자료

참고 자료

- 교육부(2013). 섭외 및 제작스태프 구성(LM0803020104_13v1). 한국직업능력개발원.

참고 자료는

해당 학습모듈에 제시된 인용 자료의 출처를 제시하였습니다. 교수·학습의 과정에서 참고로 활용할 수 있습니다.

5. 활용 서식/부록

활용 서식

스튜디오 기술 스태프 구성표

직종	이름	연락처	소속	특이사항	비고
기술감독					
조명감독					

활용 서식은

평가 서식, 실습 시트 등 교수·학습 시 활용할 수 있는 다양한 서식들로 구성하였습니다. 수행에서 평가에 이르기까지 필요한 서식을 해당 모듈의 특성에 맞춰 개발하거나 기존의 양식을 활용하여 제시하였습니다.

부록

[디지털 텔레비전 방송프로그램 음량 등에 관한 기준]

제정 2014. 11. 29. 미래창조과학부 고시 제2014-87호

제1장 총칙

제1조(목적) 이 고시는 방송법 제70조의2제1항에 따라 방송사업자가 디지털 텔레비전 방송프로그램 및 방송광고의 음량을 일정하게 유지하기 위해 필요한 사항을 규정함을 목적으로 한다.

부록은

활용 서식 이외에 교수·학습 과정에서 참고할 수 있는 자료가 있는 경우 제시하였습니다.

[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	데이터베이스 성능확보	데이터베이스 성능확보
UI/UX엔지니어링	데이터 전환 설계	데이터 전환 설계
시스템SW엔지니어링	데이터 전환	데이터 전환
빅데이터플랫폼구축	SQL활용	SQL활용
핀테크엔지니어링	SQL응용	SQL응용
데이터아키텍처	SQL작성	SQL작성
IoT시스템연동		
인프라스트럭쳐 아키텍처 구축		

차 례

학습모듈의 개요	1
학습 1. 기본 SQL 작성하기	
1-1. 테이블의 데이터 사전 조회	3
1-2. 기본적인 SQL 작성	9
• 교수 · 학습 방법	36
• 평가	37
학습 2. 고급 SQL 작성하기	
2-1. 테이블 외의 데이터 사전 조회	39
2-2. 인덱스와 뷰 작성	45
• 교수 · 학습 방법	63
• 평가	64
참고 자료	66

SQL활용 학습모듈의 개요

학습모듈의 목표

관계형 데이터베이스에서 SQL을 사용하여 목적에 적합한 데이터를 정의하고, 조작하며, 제어할 수 있다.

선수학습

SQL 작성(2001020415_19v1)

학습모듈의 내용 체계

학습	학습 내용	NCS 능력단위 요소	
		코드 번호	요소 명칭
1. 기본 SQL 작성하기	1-1. 테이블의 데이터 사전 조회 1-2. 기본적인 SQL 작성	2001020413_19v4.1	기본 SQL 작성하기
2. 고급 SQL 작성하기	2-1. 테이블 외의 데이터 사전 조회 2-3. 인덱스와 뷰 작성	2001020413_19v4.2	고급 SQL 작성하기

핵심 용어

테이블(Table), 기본키(PK ; Primary Key), 외래키(FK ; Foreign Key), 무결성 제약 조건(Integrity Constraint), 참조 무결성(Referential Integrity), DDL(Data Definition Language), DML(Data Manipulation Language), DCL(Data Control Language), 뷰(View), 인덱스(Index)

1-1. 테이블의 데이터 사전 조회

학습 목표

- 생성된 테이블의 목록, 테이블의 구조와 제약 조건을 파악하기 위해 데이터 사전을 조회하는 명령문을 작성할 수 있다.

필요 지식 /

① 데이터 사전

1. 데이터 사전 개념

데이터 사전(Data Dictionary)에는 데이터베이스의 데이터(사용자 데이터)를 제외한 모든 정보(DBMS가 관리하는 데이터)가 있다. 데이터 사전의 내용을 변경하는 권한은 시스템 사용자(데이터베이스 관리자: DBA)가 가진다. 반면 일반 사용자에게는 단순 조회만 가능한 읽기 전용 테이블 형태가 제공된다.

데이터를 제외한(데이터를 구성하는) 모든 정보라는 점은 데이터의 데이터를 말한다. 따라서 데이터 사전은 메타 데이터(Meta data)로 구성되어 있음을 의미한다.

2. 데이터 사전 내용

데이터 사전 안에 포함된 메타 데이터의 유형은 다음과 같다.

- 사용자 정보(ID, 비밀번호 및 권한 등)
- 데이터베이스 객체(테이블, 인덱스, 뷰 등)
- 무결성 제약 상태
- 함수, 프로시저 및 트리거 정보 등

데이터 사전에 포함된 정보가 메타 데이터라는 것은 모든 DBMS 제품에 공통이지만 데이터 사전을 만드는 방법, 관리하는 방법 등의 차이로 메타 데이터를 구성하는 내용은 제품마다 다르다.

3. 데이터 사전 활용

사용자에게 데이터 사전은 단순 조회의 대상일 뿐이지만 데이터베이스 엔진을 이루는 파서, 옵티마이저와 같은 구성 요소에 대한 데이터 사전은 작업을 수행하는 데 필요한 참조 정보이다. 뿐만 아니라 작업을 수행할 대상이기도 하다.

② 테이블에 대한 기본적인 데이터 사전 조회

1. 오*에서 테이블의 데이터 사전 조회

오* 사용자는 뷰(View)로 데이터 사전에 접근할 수 있다. 오*에서 뷰로 만들어진 데이터 사전은 오브젝트에 접근할 수 있는 사용자 권한에 따라 세 가지로 구분된다.

DBA_ > ALL_ > USER_

오*에서는 이와 같은 지시자 뒤에 오브젝트 이름을 붙이는 형식으로 뷰의 이름이 정해진다. 여기서 오브젝트는 테이블, 칼럼, 제약 조건, 인덱스 등을 의미하므로 다음과 같은 영역별 조회문 구성이 가능하다.

〈표 1-1〉 오* 데이터 사전 영역

영역	검색 범위	테이블 관련 데이터 사전	용도
DBA_	데이터베이스의 모든 객체 조회 가능 (DBA_는 시스템 접근 권한)	DBA_TABLES	모든 테이블 목록 확인
		DBA_TAB_COLUMNS	모든 테이블의 구성 칼럼 목록 확인
		DBA_TAB_COMMENTS	모든 테이블의 커멘트 확인
		DBA_CONSTRAINTS	모든 제약 조건 확인
ALL_	자신의 계정으로 접근 할 수 있는 객체와 다른 계정에 접근 가능한 권한을 가진 모든 객체 조회 가능	ALL_TABLES	권한 있는 테이블 목록 확인
		ALL_TAB_COLUMNS	권한 있는 테이블의 구성 칼럼 목록 확인
		ALL_TAB_COMMENTS	권한 있는 테이블의 코멘트 확인
		ALL_CONSTRAINTS	권한 있는 제약 조건 확인
USER_	현재 자신의 계정이 소유한 객체 조회 가능	USER_TABLES	자기 계정의 테이블 목록 확인
		USER_TAB_COLUMNS	자기 계정의 테이블 구성 칼럼 목록 확인
		USER_TAB_COMMENTS	자기 계정의 테이블의 코멘트 확인
		USER_CONSTRAINTS	자기 계정의 제약 조건 확인

2. M*-SQL에서 테이블의 데이터 사전 검색

데이터 사전은 테이블 형태로 구성되어 있다. 따라서 테이블의 내용을 검색하기 위해서는 해당 테이블의 위치와 이름을 정확히 알고 있어야 한다. 여기서 위치는 데이터베이스를 의미한다.

M*-SQL에서 데이터 사전은 Information_schema라는 데이터베이스 안에 존재한다. 따라서 이 안의 테이블을 조회하기 위해서는 우선 해당 데이터베이스로 이동해서 테이블 목록을 요청해야 한다.

- use Information_schema; -- 이동
- show tables; -- 테이블 목록 보기

테이블 목록으로 데이터 사전을 구성하는 테이블 이름을 확인하고, SELECT문을 통해 해당 테이블의 내용을 조회할 수 있다.

수행 내용 / 테이블 관련 데이터 사전 조회하기

재료·자료

- SQL 언어 문법 학습 자료
- DBMS 매뉴얼 및 튜토리얼 자료

기기(장비 · 공구)

- DBMS 프로그램

안전 · 유의 사항

- 데이터 사전 용도에 대한 추가 학습이 필요하다.
- 데이터베이스 제품별 사용 방법의 차이가 크다. 특히 데이터 사전이란 개념을 구현하고 사용하는 방식에 있어 SQL 사용법은 제품별로 큰 차이가 있기에, 각 제품별 데이터 사전 활용 방법에 대한 학습이 별도로 필요하다.

수행 순서

① 데이터 사전을 검색한다(M*-SQL 기준).

1. 데이터베이스를 선택한다.

데이터 사전 테이블을 관리하는 데이터베이스로 이동하여 테이블의 목록을 살펴보면 다음 [그림 1-1]과 같은 결과를 얻을 수 있다.

```

> use Information_schema;
Database changed
> show tables;
+-----+
| Tables_in_information_schema |
+-----+
| CHARACTER_SETS
| COLLATIONS
| COLLATION_CHARACTER_SET_APPLICABILITY
| COLUMNS
| COLUMN_PRIVILEGES
| ENGINES
| EVENTS
| FILES
+-----+

```

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.6.
[그림 1-1] 데이터 사전 목록 조회1

information_schema라는 데이터베이스로 이동하지 않고 데이터베이스를 SQL문에 지정하여 데이터 사전을 확인할 수 있다. 다음의 [그림 1-2]는 데이터 사전 목록을 조회한 것이며, 데이터 사전의 개수가 많아 10개로 제한한다.

```

> SELECT TABLE_NAME
>   FROM information_Schema.tables
>  LIMIT 10;
+-----+
| TABLE_NAME
+-----+
| ALL_PLUGINS
| APPLICABLE_ROLES
| CHARACTER_SETS
| CHECK_CONSTRAINTS
| COLLATIONS
| COLLATION_CHARACTER_SET_APPLICABILITY
| COLUMNS
| COLUMN_PRIVILEGES
| ENABLED_ROLES
| ENGINES
+-----+
10 rows in set (0.001 sec)

```

출처: 집필진 제작(2021)
[그림 1-2] 데이터 사전 목록 조회2

2. 테이블과 관련된 데이터 사전 테이블을 살펴본다.

테이블과 관련하여 자주 사용하는 데이터 사전은 다음과 같다. 해당 데이터 사전의 의미를 이해하고, 데이터 사전 목록을 조회해본다.

〈표 1-2〉 테이블의 데이터 사전 목록

데이터 사전 테이블	주요 내용
COLUMNS	테이블을 구성하는 칼럼에 대한 정보
COLUMN_PRIVILEGES	테이블 칼럼 권한에 대한 정보 제공
TABLES	데이터베이스에 존재하는 테이블에 대한 정보 제공
TABLE_CONSTRAINTS	테이블에 설정된 제약 사항과 관련된 정보

3. 테이블과 관련된 주요 데이터 사전 내용을 조회한다.

(1) 테이블 목록을 확인한다.

테이블에 대한 데이터 사전의 내용을 확인한다. 실습 편의상 ‘ncs’라는 이름의 데이터베이스가 있다고 가정하고, ncs 데이터베이스에 생성된 테이블 목록을 조회한다. FROM 절에 사용된 tables라는 테이블은 다음 [그림 1-3]과 같이 데이터베이스의 모든 테이블 목록을 제공한다. 아래와 같이 테이블명, 사용한 데이터베이스 엔진, 데이터 길이, 생성된 시점 등의 정보를 제공함을 확인할 수 있다.

```
> SELECT TABLE_SCHEMA, TABLE_NAME, ENGINE, DATA_LENGTH, CREATE_TIME
>   FROM information_Schema.tables
> WHERE table_schema = 'ncs';
+-----+-----+-----+-----+
| TABLE_SCHEMA | TABLE_NAME | ENGINE | DATA_LENGTH | CREATE_TIME |
+-----+-----+-----+-----+
| ncs         | tab1      | InnoDB |       16384 | 2021-06-01 00:09:56 |
| ncs         | tab2      | InnoDB |       16384 | 2021-06-01 00:09:58 |
| ncs         | tab3      | InnoDB |       16384 | 2021-06-01 00:09:59 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

출처: 집필진 제작(2021)

[그림 1-3] tables 데이터 사전 조회

(2) 테이블에 설정된 제약 사항을 확인한다.

다음은 테이블에 설정된 제약 사항 목록을 확인하는 데이터 사전이다. FROM 절에 table_constraints라는 테이블을 통해서 제약 사항(PK, FK) 목록을 확인할 수 있다.

```

> SELECT CONSTRAINT_NAME, TABLE_NAME, CONSTRAINT_TYPE
>   FROM information_Schema.table_constraints
> WHERE table_schema='ncs';
+-----+-----+-----+
| CONSTRAINT_NAME | TABLE_NAME | CONSTRAINT_TYPE |
+-----+-----+-----+
| PRIMARY        | tab1      | PRIMARY KEY    |
| PRIMARY        | tab2      | PRIMARY KEY    |
+-----+-----+-----+
2 rows in set (0.001 sec)

```

출처: 집필진 제작(2021)
[그림 1-4] table_constraints 데이터 사전 조회

수행 tip

- 각 DBMS마다 데이터 사전의 개념은 동일하나 구현 결과는 데이터베이스 제품마다 차이가 크다. 데이터 사전 설계 구조의 작은 차이가 실제 사용 단계 활용 방법에서는 큰 차이로 나타난다.
- 다른 유형의 SQL 명령문과 달리 데이터 사전에 대한 조회 방법은 데이터베이스 제품마다 전혀 다른 방법으로 내용을 조회하고 이해해야 한다.

1-2. 기본적인 SQL 작성

학습 목표

- 조인, 서브쿼리, 집합 연산자를 사용하여 두 개 이상의 테이블로부터 데이터를 조회하는 DML(Data Manipulation Language) 명령문을 작성할 수 있다.
- 테이블의 구조와 제약 조건을 생성, 삭제하고 수정하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.
- 업무 단위인 트랜잭션의 원료와 취소를 위한 DCL(Data Control Language) 명령문을 작성 할 수 있다.

필요 지식 /

① 데이터 정의(DDL문)

1. DDL 개념

DDL(Data Definition Language)은 영문 그대로 ‘데이터를 정의하는 언어’이다. 보다 직관적으로 설명하자면 ‘데이터를 담을 그릇을 생성하는 언어’이다. 이렇게 생성한 그릇을 DBMS에서는 오브젝트(Object)라고 한다. 다음과 같이 DDL을 통해 생성할 수 있는 오브젝트 대상을 확인한다.

〈표 1-3〉 DDL 대상

DDL 대상	설명	비고
스키마(Schema)	- DBMS의 특징과 구축 환경에 기반한 데이터 구조 - 하나의 데이터베이스라고 이해할 수 있음	DBMS마다 차이
도메인(Domain)	- 속성의 데이터 타입과 제약 조건 등을 설정한 정보 - 속성이 가지는 값의 범위로 이해할 수 있음	예를 들어, 주소를 VARCHAR(120)로 정의
테이블(Table)	- 데이터 저장 공간	본 학습 대상
뷰(View)	- 1개 이상의 물리 테이블을 통해서 만드는 가상의 논리 테이블	학습 2-2 참조
인덱스(Index)	- 빠른 검색을 위한 데이터 구조	학습 2-2 참조

2. DDL 유형

오브젝트를 생성, 변경, 그리고 제거하기 위해 다음과 같은 명령어를 사용한다.

〈표 1-4〉 DDL 명령어

구분	DDL 명령어	내용
생성	CREATE	오브젝트 생성
변경	ALTER	오브젝트 변경
삭제	DROP	오브젝트 삭제
	TRUNCATE	오브젝트 내용 삭제(데이터베이스 내의 로깅 작업 건너뜀)

DDL 명령어로 분류되지는 않지만 DDL과 같이 사용되는 명령어가 있다. 비상용 제품인 M*SQL의 경우, 생성된 오브젝트의 목록을 조회하기 위해서는 SHOW 명령문을 사용하며, 내용을 조회하기 위해서는 SELECT문을 사용한다. 상용 제품인 O*의 경우 SELECT로 목록과 내용을 조회한다.

3. DDL 작성

데이터베이스를 구성하기 위해 스키마, 도메인, 테이블, 뷰, 인덱스 등과 같은 오브젝트에 대한 DDL 작업이 필요하지만, 본 학습에서는 테이블만을 대상으로 한다.

(1) 테이블 생성

테이블 생성을 위한 DDL 사용 방법은 다음과 같이 두 가지 종류로 구분할 수 있다.

〈표 1-5〉 테이블 생성 SQL문

구분	문법
신규 생성	CREATE TABLE 테이블명 (칼럼명 데이터 타입 [DEFAULT 값] [NOT NULL] {칼럼명 데이터 타입 [DEFAULT 값] [NOT NULL]}* [PRIMARY KEY (칼럼 리스트),] {[FOREIGN KEY (칼럼 리스트) REFERENCES 테이블 명 [(칼럼명)] [ON DELETE 옵션] [ON UPDATE 옵션] }, }* [CHECK (조건식) UNIQUE(칼럼명)]) ;
다른 테이블 정보를 활용한 테이블 생성	CREATE TABLE 테이블명 AS SELECT문;

위와 같이 다른 테이블을 이용해서 신규 테이블을 생성하는 방법은 DBMS 제품마다 차이가 있다.

(2) 테이블 변경

ALTER를 이용하여 테이블 구조를 변경하는 문법은 다음과 같다.

〈표 1-6〉 ALTER 이용한 테이블 변경 SQL문

구분	문법
열 추가	ALTER TABLE 테이블명 ADD 칼럼명 데이터 타입 [DEFAULT 값]
열 데이터 타입 변경	ALTER TABLE 테이블명 MODIFY 칼럼명 데이터 타입 [DEFAULT 값]
열 삭제	ALTER TABLE 테이블명 DROP 칼럼명

(3) 테이블 삭제, 절단, 이름 변경

DROP TABLE, TRUNCATE TABLE, RENAME TABLE 명령문을 사용하여 테이블 삭제, 테이블의 데이터 삭제, 테이블 이름을 변경할 수 있다. 테이블 존재 및 테이블 내용을 삭제하기 위한 명령어의 사용 문법은 다음과 같다.

〈표 1-7〉 테이블 삭제 및 이름 변경 방법

구분	문법
테이블 삭제	DROP TABLE 테이블명
테이블 내용 삭제	TRUNCATE TABLE 테이블명
테이블명 변경	RENAME TABLE 이전_테이블명 TO 새로운_테이블명 ALTER TABLE 이전_테이블명 RENAME 새로운_테이블명

(4) 제약 조건 적용

다음과 같은 제약 조건을 테이블 생성 과정에 적용할 수 있다. CREATE 문을 통해 테이블 생성 과정에서 제약 조건을 설정할 수 있고, ALTER 문을 통해 이미 생성된 테이블의 제약 조건을 변경할 수 있다.

〈표 1-8〉 테이블 생성에 사용되는 제약 조건

제약 조건	설명
PRIMARY KEY(PK)	<ul style="list-style-type: none"> - 테이블의 기본 키를 정의함 - 기본적으로 NOT NULL, UNIQUE 제약 사항이 설정됨 - 테이블에 외래 키를 정의함
FOREIGN KEY(FK)	<ul style="list-style-type: none"> - 참조 대상을 테이블명(칼럼명) 형식으로 작성해야 함 - 참조 무결성이 위배되는 상황 발생 시, 다음 옵션으로 처리 가능 (CASCADE, NO ACTION, SET NULL, SET DEFAULT)
UNIQUE	<ul style="list-style-type: none"> - 테이블에서 해당하는 열값은 유일해야 함을 의미함 - 테이블에서 모든 값이 다르게 적재되어야 하는 열에 설정함
NOT NULL	<ul style="list-style-type: none"> - 테이블에서 해당하는 열의 값은 NULL 불가능 - 필수적으로 입력해야 하는 항목에 설정함
CHECK	<ul style="list-style-type: none"> - 사용자가 직접 정의하는 제약 조건 - 발생 가능한 상황에 따라 여러 가지 조건을 설정 가능

② 다중 테이블 조회(DML문)

관계형 데이터베이스는 데이터의 중복을 최소화하기 위해 데이터를 분해하여 저장하고 통합하여 사용한다. 데이터를 분해하는 방법에는 정규화 기법이 있고, 데이터를 통합하는 방법에는 다중 테이블 검색 기법이 있다. 다중 테이블을 이용할 수 있는 세부적인 기법은 다음과 같다.

〈표 1-9〉 다중 테이블 검색 방법

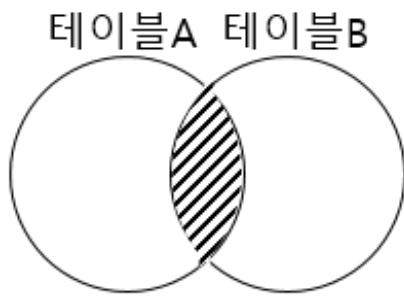
다중 테이블 검색 기법	내용
조인	두 개의 테이블을 결합하여 데이터를 추출하는 기법
서브쿼리	SQL문 안에 포함된 SQL문 형태의 사용 기법
집합 연산자	테이블을 집합 개념으로 조작하는 기법

1. 조인(JOIN)

(1) 조인 개념

조인은 영문 의미 그대로는 결합을 의미하며, 관계형 데이터베이스에서의 조인은 교집합 결과로 데이터 결합 방법을 의미한다. 교집합이 되는 공통점은 다양한 관점에서 정의될 수 있다. 여기서 그 관점을 정의하는 것이 바로 조인의 조건이 된다.

조인은 두 테이블의 공통값을 이용하여 칼럼을 조합하는 수단이다. 보통 PK와 FK값을 결합하여 사용하는 것이 일반적이다. 보다 엄밀하게 말하자면 PK, FK와 관계없이 논리적인 값들의 연관을 사용한다. 세 개 이상의 테이블에 대한 조인(결합)은 두 개의 테이블을 우선 조인하고 그 결과와 나머지 한 개의 테이블을 다시 조인한다.



출처: 집필진 제작(2021)
[그림 1-5] 조인 개념도

(2) 조인 유형

조인은 관계형 데이터베이스의 가장 핵심 기능이자 큰 장점이라 할 수 있다. 조인은 크게 물리적 조인과 논리적 조인으로 구분할 수 있으며, 물리적 조인은 데이터베이스의 성능을 높이기 위한 튜닝 관점에서 다루는 주제로서, 본 학습에서는 사용자가 직접 제어할 수 있는 논리적 조인에 대해 알아본다.

〈표 1-10〉 조인 유형(대분류)

조인 분류	내용
논리적 조인	사용자가 작성한 SQL문에 의해서 표현한 테이블 결합 방식 데이터베이스 관리시스템(DBMS)의 옵티마이저 엔진에 의해 발생하는 테이블 결합 방식
물리적 조인	- Nested Loop Join - Merge Join - Hash Join

논리적 조인은 크게 내부 조인과 외부 조인으로 구분할 수 있으며, 각각의 세부 유형은 다음과 같이 구분할 수 있다.

〈표 1-11〉 조인 유형(세분류)

조인 유형	내용
내부 조인 (Inner Join)	두 테이블에 공통으로 있는 칼럼(열)을 활용하는 유형(공통 칼럼 기반)
동등 조인 (Equi Join)	공통으로 있는 칼럼값이 같은 경우에 레코드 추출
자연 조인 (Natural Join)	두 테이블에 있는 동일한 칼럼명을 기준으로, 모든 칼럼값이 같은 경우에 레코드 추출
교차 조인 (Cross Join)	조인 조건이 없는 모든 데이터의 조합을 추출
외부 조인 (Outer Join)	특정한 테이블의 모든 데이터를 기준으로 다른 테이블의 정보와 비교하여 추출(단, 다른 테이블에 동일한 값이 없어도 특정한 테이블은 출력됨)
왼쪽 외부 조인 (Left Outer Join)	왼쪽 테이블의 모든 데이터와 오른쪽 테이블의 동일한 데이터를 추출(단, 오른쪽 테이블에 동일한 값이 없어도 왼쪽 테이블의 레코드는 출력됨)
오른쪽 외부 조인 (Right Outer Join)	오른쪽 테이블의 모든 데이터와 왼쪽 테이블의 동일 데이터를 추출(단, 왼쪽 테이블에 동일한 값이 없어도 오른쪽 테이블의 레코드는 출력됨)
완전 외부 조인 (Full Outer Join)	양쪽의 모든 데이터를 추출(단, 오른쪽과 왼쪽 테이블에 동일한 값이 없어도 오른쪽, 왼쪽 테이블의 레코드는 출력됨)

내부 조인에서 내부의 의미는 외부 조인에 대비하기 위해 사용되었으므로 수식어가 없는 조인은 내부 조인을 의미한다.

내부 조인의 세부 유형은 조인의 조건에 따라 세분된다. 명시적으로 지정하지 않고, 두 테이블의 컬럼명이 같은 값을 기준으로 하면 자연 조인이고, 특정 컬럼을 지정하면서 같은 값을 비교하면 동등 조인이며, 값이 다른 것을 비교하면 비동등 조인이 된다.

외부 조인은 기준 테이블이 참조 테이블과 조인하되, 참조 테이블에 조인할 데이터가 있는 경우는 참조 테이블의 데이터를 함께 출력하고, 참조 테이블의 조인 데이터가 없는 경우에도 기준 테이블의 모든 데이터를 표시하고 싶은 경우에 사용한다. 보통 기준 테이블의 모든 값에 대해 참조 테이블의 데이터가 반드시 존재한다는 보장이 없는 경우 외부 조인을 사용한다.

2. 서브쿼리(Sub-Query)

(1) 서브쿼리 개념

서브쿼리는 다음 [그림 1-5]와 같이 SQL문 안에 포함된 또 다른 SQL문을 말하며, 아직 확인되지 않은 기준을 위한 검색하는 용도로 사용한다.



출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.14.
[그림 1-6] 서브쿼리 개념도

가장 밖에 위치한 메인쿼리와 그 안에 포함된 서브쿼리 관계는 주종(Master-Slave) 관계를 이루고 있다. 서브쿼리에 작성된 칼럼명은 메인쿼리의 칼럼명을 가져와서 사용할 수 있으나 그 역은 성립하지 않는다.

(2) 서브쿼리 유형

서브쿼리는 자동하는 방식과 반환되는 값의 형태에 따라 서브쿼리 종류를 각각 분류할 수 있다. 우선 동작하는 방식에 따른 서브쿼리의 종류는 다음과 같다.

〈표 1-12〉 서브쿼리 유형 1

서브쿼리 종류	설명
비연관(Un-Correlated) 서브쿼리	서브쿼리 안에 메인쿼리의 칼럼 정보를 가지고 있지 않은 형태 (서브쿼리는 메인쿼리 없이 독자적으로 실행)
연관(Correlated) 서브쿼리	서브쿼리 안에 메인쿼리의 칼럼 정보를 가지고 있는 형태 (메인쿼리의 실행된 결과를 통해서 서브쿼리의 조건이 맞는지 확인)

반환되는 데이터 형태에 따른 서브쿼리의 종류는 다음과 같다.

〈표 1-13〉 서브쿼리 유형 2

서브쿼리 종류	설명
Single Row(단일 행) 서브쿼리	- 서브쿼리의 결과가 항상 1건 이하인 서브쿼리 - 단일 행의 비교 연산자에는 $=$, $<$, \leq , $>$, \geq 등이 사용
Multiple Row(다중 행) 서브쿼리	- 서브쿼리 실행 결과가 여러 건인 서브쿼리 - 다중 행의 비교 연산자에는 IN, EXISTS, ALL, ANY 등이 사용
Multiple Column(다중 칼럼) 서브쿼리	- 서브쿼리의 실행 결과가 2개 이상 칼럼으로 반환되는 쿼리 - 메인쿼리의 조건절에 다수 칼럼을 비교할 때, 서브쿼리와 메인쿼리에서 비교하는 칼럼 개수, 위치가 동일해야 함

3. 집합 연산자

(1) 집합 연산 개념

집합 연산자는 테이블을 하나의 집합이라고 가정하고, 두 테이블 간의 집합 연산

자를 사용해 계산하는 방식이다. 집합 연산자는 2개 이상의 쿼리를 연결하여 하나로 통합해 주는 역할을 수행한다. 즉 집합 연산자는 다수의 쿼리 실행 결과를 하나의 결과로 만든다. 일반적으로 집합 연산자를 사용하는 사례는 서로 다른 테이블 간에 실행 결과를 하나로 합치려고 할 때와 같은 테이블에서 서로 다른 조건으로 쿼리를 실행하여 결과를 합치려고 할 때 사용할 수 있다.

(2) 집합 연산자 유형

테이블을 집합의 개념으로 보고 두 테이블 사이에 가능한 집합 연산은 다음과 같은 종류가 있다.

〈표 1-14〉 집합 연산자 유형

집합 연산자	설명
UNION	2개 이상 SQL문의 실행 결과에 대한 중복을 제거한 합집합
UNION ALL	2개 이상 SQL문의 실행 결과에 대한 중복을 제거하지 않은 합집합
INTERSECTION	2개 이상 SQL문의 실행 결과에 대한 중복을 제거한 교집합
EXCEPT (MINUS)	선행 SQL문의 실행 결과와 후행 SQL문의 실행 결과 사이의 중복을 제거한 차집합(일부 DBMS는 MINUS로 사용)

③ 데이터 제어(DCL 문)

1. DCL 개념

데이터베이스에서 데이터 외의 오브젝트를 조작하려고 하는 경우에 DCL 명령을 사용한다. 이는 Data Control Language라는 약자로, 데이터 제어 언어라고 한다. 다음을 통해 DCL문의 대상과 오브젝트 유형을 확인한다.

〈표 1-15〉 DCL 조작 대상

오브젝트	목적	내용
사용자 권한	접근 통제	DBMS에 접근할 사용자를 등록하고, 특정 사용자에게 데이터베이스의 사용 권한을 부여하는 작업
트랜잭션	안전하고 무결한 거래 보장	동시 다발적으로 발생하는 다수 작업을 독립적이고 안전하게 처리하기 위한 데이터베이스 작업 단위

추가적으로 트랜잭션 제어를 위한 명령어 TCL(Transaction Control Language)라는 용어가 있다. DCL과 TCL은 작업 대상이 상이하기 때문에 서로 다른 개념으로 구분하나, 제어 기능이라는 공통점으로 TCL은 DCL의 일부로 분류하기도 한다. 다음을 통해 유형에

따른 DCL 명령어를 확인한다.

〈표 1-16〉 DCL 명령어

유형	명령어	용도
DCL	GRANT	데이터베이스 사용자 권한 부여
	REVOKE	데이터베이스 사용자 권한 회수
TCL	COMMIT	트랜잭션 확정
	ROLLBACK	트랜잭션 취소
	CHECKPOINT	복귀지점 설정

2. DCL 작성

(1) 사용자 권한 부여

사용자 권한은 시스템 권한과 객체 권한으로 구분한다. 각 권한을 부여하기 위한 명령어 사용법은 다음과 같다.

〈표 1-17〉 권한 부여 명령어 문법

권한	명령어 문법
시스템 권한	GRANT 권한-1, 권한-2 TO 사용자 계정
객체 권한	GRANT 권한-1, 권한-2 ON 객체명 TO 사용자 계정

시스템 권한과 객체 권한의 종류는 다음과 같다.

(2) 사용자 권한 회수

GRANT 명령어에 대응하는 권한 회수/해지 명령어는 REVOKE이며, 권한 유형 별 대응하는 명령어 구조는 다음과 같다.

〈표 1-18〉 권한 회수 명령어 문법

권한	명령어 문법
시스템 권한	REVOKE 권한-1, 권한-2 FROM 사용자 계정
객체 권한	REVOKE 권한-1, 권한-2 ON 객체명 FROM 사용자 계정

3. TCL 활용

(1) 트랜잭션 개념

트랜잭션은 ‘일을 처리하는 단위’를 의미한다. 더 나아가 트랜잭션의 다양한 관점은 다음과 같다.

- 트랜잭션은 논리적인 연산 단위이다.
- 하나 이상의 SQL문이 포함된다.
- 트랜잭션은 거래를 의미한다.
- 거래의 모든 결과가 모두 반영되거나 모두 취소되어야 한다.
- 분해되지 않는 최소 단위이다.

(2) 트랜잭션 제어

트랜잭션을 제어한다는 것은 흐름의 구조를 바꾼다는 것이 아니라 트랜잭션의 결과를 수용하거나 취소하는 것을 의미한다. 이러한 작업을 수행하는 TCL 관련 명령어는 다음과 같다.

〈표 1-19〉 TCL 명령어

명령어	내용	비고
COMMIT	작업거래 내용 확정	
ROLLBACK	작업거래 내용 취소	
CHECKPOINT	작업거래의 저장 시점 설정	ROLLBACK 위치 지정

그리고 사용자 계정, 객체 생성에 관련된 시스템 권한과 해당 객체에 대한 권한은 다음과 같다.

〈표 1-20〉 권한 유형

구분	권한	내용
시스템 권한	CREATE USER	계정 생성
	DROP USER	계정 삭제
	DROP ANY TABLE	테이블 삭제
	CREATE SESSION	데이터베이스 접속
	CREATE TABLE	테이블 생성
	CREATE VIEW	뷰 생성
	CREATE SEQUENCE	시퀀스 생성
	CREATE PROCEDURE	함수 생성
객체 권한	ALTER	테이블 변경
	INSERT	
	DELETE	
	SELECT	데이터 조작
	UPDATE	
	EXECUTE	PROCEDURE 실행

수행 내용 / DDL, DML, DCL 활용하기

재료·자료

- 개념 E-R 다이어그램
- 테이블 설계서
- SQL 언어 문법 학습 자료

기기(장비 · 공구)

- E-R 모델링 도구
- DBMS 프로그램

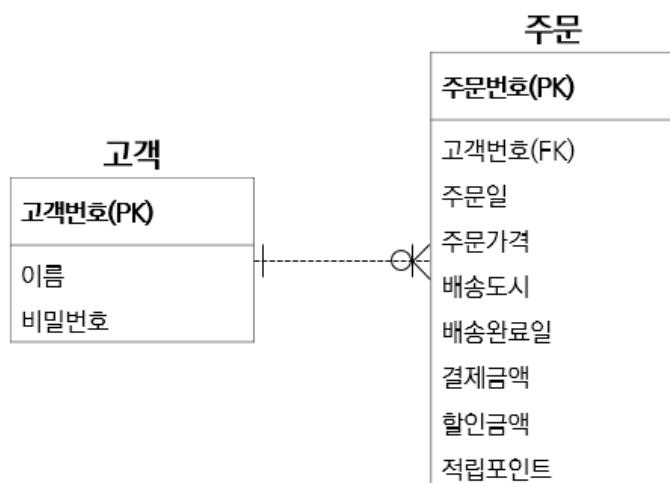
안전 · 유의 사항

- 간접 홍보 효과를 방지하기 위하여 특정 제조사나 특정 제품에 의존적인 내용은 최소화하였으며, 특정 제품에 해당하는 내용을 구분하기 위하여 제품 약어로 표기하였다.
- 제품 약어 M*-SQL과 O*는 대표적인 비상용 제품과 상용 제품을 의미한다.

수행 순서

① 데이터 확인 및 테이블 산출물을 준비한다.

1. 테이블 생성을 위한 모델링 자료를 준비한다.



출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원.

p.19.

[그림 1-7] 테이블 생성 대상인 논리 모델링 모습

위 그림에서 고객은 N개의 주문을 수행하고 있고, 또한 주문 데이터는 고객 데이터 없이 존재할 수 없는 종속되는 구조를 가진다.

2. 데이터를 확인한다.

테이블을 생성하고 변경하기 위해 주문 데이터의 예시를 확인해본다.

주문테이블

주문번호	고객번호	주문일	주문가격	배송 도시	배송완료일	결제금액	할인금액
A0100	24680	20170704	55,000	서울	20170707	45,000	10,000
X0300	13579	20170801	70,000	부산	Null	50,000	20,000

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.20.

[그림 1-8] 주문 데이터 예시

3. 데이터 타입을 결정한다.

데이터 타입은 크게 숫자형, 문자형, 날짜 등의 유형으로 나눌 수 있으며, 구체적인 형식은 NUMERIC, CHARACTER, VARCHAR, DATETIME 등이 있으나, 실제 사용하는 SQL 제품에 따라 사용되는 표현은 조금씩 달라질 수 있다.

[그림 1-7]과 같은 데이터를 저장할 테이블을 만들고자 할 때, 각 칼럼의 값은 크게 문자열과 숫자로 구분할 수 있으므로 각각 다음과 같은 데이터 타입을 사용한다.

- 문자열 - VARCHAR
- 숫자 - DECIMAL, INTEGER

4. 제약 조건이 무엇인지 확인하고 선택한다.

테이블 구조를 정의하면서 데이터 무결성을 유지하기 위해 제약 조건을 이용한다. 제약 조건이 테이블 생성 과정의 필수 요소는 아니지만 데이터베이스의 일관성(Consistency)을 유지하고 잘못된 데이터의 입력과 수정으로부터 데이터베이스를 보호하는 데 필수적인 요소이므로 반드시 정의하는 것이 좋다.

대표적인 제약 조건은 기본 키(Primary Key)와 외래 키(Foreign Key)에 의한 제약 조건이다. 그 밖에도 UNIQUE KEY, PRIMARY KEY, FOREIGN KEY, NOT NULL, CHECK 등이 있다.

본 실습에서 제공된 논리 모델링 산출물을 통해서 고객 테이블은 고객번호를 기본 키(Primary Key)로 설정해야 하고, 주문 테이블은 주문 번호를 기본 키, 고객번호를 외래 키(Foreign Key)로 설정함을 확인할 수 있다. 하지만 테이블의 변경으로 추가적인 제약 조건이 발생할 수 있다.

② 테이블을 생성한다.(M*-SQL 기준)

1. CREATE 문으로 테이블을 생성한다.

[그림 1-7]과 같은 데이터를 저장하기 위한 테이블을 생성하기 위해 다음과 같은 SQL문을 사용한다.(각 데이터 타입의 괄호 안의 숫자는 최대 길이를 의미함)

〈표 1-21〉 고객 테이블 생성 SQL문

위치	고객 테이블 생성 SQL문		
1	CREATE TABLE '고객' (
2	'고객번호'	varchar(16)	NOT NULL,
3	'이름'	varchar(50)	NOT NULL,
4	'비밀번호'	varchar(256)	NOT NULL,
5	PRIMARY KEY ('고객번호')		
6)		

〈표 1-22〉 주문 테이블 생성 SQL문

위치	주문 테이블 생성 SQL문		
1	CREATE TABLE '주문' (
2	'주문번호'	varchar(16)	NOT NULL,
3	'고객번호'	varchar(16)	NOT NULL,
4	'주문일'	varchar(8)	NOT NULL,
5	'주문가격'	decimal(15,2)	NOT NULL,
6	'배송도시'	varchar(256),	
7	'배송완료일'	varchar(8),	
8	'결제금액'	varchar(8),	
9	'할인금액'	decimal(15,2)	NOT NULL,
10	'적립포인트'	decimal(15,2)	NOT NULL,
11	PRIMARY KEY ('주문번호'),		
12	FOREIGN KEY ('고객번호') REFERENCES 고객 ('고객번호')		
13)		

2. SELECT를 이용하여 테이블을 생성한다.

만일 이미 있는 테이블과 동일한 테이블을 생성하고자 한다면 다음과 같이 ‘기존 테이블’에 존재하는 테이블 정보를 이용하여 ‘신규 테이블’을 만들 수 있다.

CREATE TABLE 신규 테이블 AS SELECT * FROM 기존 테이블;

위의 SQL문은 SELECT 절을 통해 ‘기존 테이블’의 속성을 조회하고, 조회 결과를

통해서 ‘신규 테이블’의 속성으로 만드는 방식을 보여주고 있다. 이처럼 SELECT문을 사용해서 테이블을 생성하는 방식은 〈표 1-21〉, 〈표 1-22〉와 같은 테이블 생성 방법과 달리 다음과 같은 특징이 있음에 유의한다.

- 새로 만든 ‘신규 테이블’은 ‘기존 테이블’의 칼럼 정보(데이터 타입, 크기)와 동일하게 적용
- NOT NULL 속성도 동일하게 적용
- 제약 조건은 미적용
- 동일한 칼럼들로 테이블을 새로 만든다면 '*' 사용
- 필요한 칼럼만으로 구성된 테이블을 만들 수 있음
- 제약 조건의 추가는 ALTER TABLE 구문을 사용

3. 테이블의 생성 여부를 확인한다.

테이블 생성 여부를 확인하기 위해 다음과 같은 명령어를 사용한다.

〈표 1-23〉 테이블 확인 SQL문

DBMS 제품	테이블 목록 확인 SQL문
대표적인 상용 O 제품	SELECT * FROM user_tables;
대표적인 비상용 M 제품	SHOW TABLES;

결과에 앞서 생성한 ‘주문’ 테이블이 있으면 테이블 생성이 성공한 것이다. 또는 다음 명령어를 통해 테이블 구조를 확인할 수 있다.

〈표 1-24〉 테이블 구조 보기

구분	내용																																																		
테이블 구조 보기 명령문	DESC '주문';																																																		
명령 결과	<table border="1"> <thead> <tr> <th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th></tr> </thead> <tbody> <tr> <td>주문번호</td><td>varchar(16)</td><td>NO</td><td>PRI</td><td>NULL</td></tr> <tr> <td>고객번호</td><td>varchar(16)</td><td>NO</td><td></td><td>NULL</td></tr> <tr> <td>주문일</td><td>varchar(8)</td><td>NO</td><td></td><td>NULL</td></tr> <tr> <td>주문가격</td><td>decimal(15,2)</td><td>NO</td><td></td><td>NULL</td></tr> <tr> <td>배송도시</td><td>varchar(256)</td><td>YES</td><td></td><td>NULL</td></tr> <tr> <td>배송완료일</td><td>varchar(8)</td><td>YES</td><td></td><td>NULL</td></tr> <tr> <td>결제금액</td><td>varchar(8)</td><td>YES</td><td></td><td>NULL</td></tr> <tr> <td>할인금액</td><td>decimal(15,2)</td><td>NO</td><td></td><td>NULL</td></tr> <tr> <td>적립포인트</td><td>decimal(15,2)</td><td>NO</td><td></td><td>NULL</td></tr> </tbody> </table> <p>출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.22.</p> <p>[그림 1-9] 주문 테이블 구조</p>	Field	Type	Null	Key	Default	주문번호	varchar(16)	NO	PRI	NULL	고객번호	varchar(16)	NO		NULL	주문일	varchar(8)	NO		NULL	주문가격	decimal(15,2)	NO		NULL	배송도시	varchar(256)	YES		NULL	배송완료일	varchar(8)	YES		NULL	결제금액	varchar(8)	YES		NULL	할인금액	decimal(15,2)	NO		NULL	적립포인트	decimal(15,2)	NO		NULL
Field	Type	Null	Key	Default																																															
주문번호	varchar(16)	NO	PRI	NULL																																															
고객번호	varchar(16)	NO		NULL																																															
주문일	varchar(8)	NO		NULL																																															
주문가격	decimal(15,2)	NO		NULL																																															
배송도시	varchar(256)	YES		NULL																																															
배송완료일	varchar(8)	YES		NULL																																															
결제금액	varchar(8)	YES		NULL																																															
할인금액	decimal(15,2)	NO		NULL																																															
적립포인트	decimal(15,2)	NO		NULL																																															

4. CREATE문에 사용된 제약 조건을 조사한다.

테이블 생성을 위해 사용한 <표 1-21>,<표 1-22>와 같은 SQL문에서 무결성을 보장하기 위해 사용된 제약 조건은 다음과 같다.

<표 1-25> 고객 테이블 생성 시 제약 조건

위치	요소	의미
2 ~ 4	NOT NULL	각 칼럼의 값으로 Null 값 허용 여부 지정 - 제약 조건
5	PRIMARY KEY	기본 키(Primary Key) 지정 - 제약 조건

<표 1-26> 주문 테이블 생성 시 제약 조건

위치	요소	의미
6 ~ 8		제약 조건 없음
2 ~ 5, 9 ~ 10	NOT NULL	각 칼럼의 값으로 Null 값 허용 여부 지정 - 제약 조건
11	PRIMARY KEY	기본 키(Primary Key) 지정 - 제약 조건
12	FOREIGN KEY	외래 키(Foreign Key) 지정 - 제약 조건

③ 테이블 구조를 변경한다(M*-SQL 기준).

1. 테이블 구조 변경 대상을 정의한다.

앞에서 생성한 '배송도시'를 '배송도시코드'로 속성 이름을 바꾸면서 동시에 데이터 타입을 바꾸어 보자. 이때 새로운 데이터 타입을 정수형으로 정의한다.

2. 칼럼명과 데이터 타입을 변경한다.

'배송도시'를 '배송도시코드'로 변경하는 SQL문은 다음과 같다.

```
ALTER TABLE '주문' CHANGE '배송도시' '배송도시코드' INT;
```

'ALTER TABLE 테이블명 CHANGE 칼럼명 자료형'을 사용하면 칼럼의 크기와 데이터 타입을 변경할 수 있다. 또한 DEFAULT 값도 수정할 수 있다. DEFAULT 값을 수정하면 수정 이후에 입력되는 값에만 수정된 DEFAULT 값이 적용된다.

3. 칼럼을 추가하고 삭제한다.

다음과 같은 명령어를 통해 칼럼의 추가 및 삭제가 가능하다.

〈표 1-27〉 테이블 변경을 위한 SQL문

구분	SQL문 형식
맨 뒤에	ALTER TABLE ‘테이블명’ ADD ‘칼럼명’ 데이터타입;
추가	맨 앞에 ALTER TABLE ‘테이블명’ ADD ‘새칼럼명’ 데이터타입 FIRST
지정 칼럼 뒤	ALTER TABLE ‘테이블명’ ADD ‘새칼럼명’ 데이터타입 AFTER ‘앞칼럼명’;
삭제	ALTER TABLE ‘테이블명’ DROP ‘칼럼명’;

‘ALTER TABLE 테이블명 ADD 칼럼명 데이터타입’ 문을 사용하여 테이블에 칼럼을 추가하면 칼럼은 테이블의 마지막에 추가된다. 테이블이 가지는 칼럼의 개수에는 제한이 있어서 추가가 안 될 수도 있다.

④ 데이터를 조작한다.

1. 실습 데이터를 삽입한다.

고객 테이블에 칼럼명을 나열하지 않고 복수 개의 레코드를 삽입하는 명령문은 다음과 같다.

```
INSERT INTO ‘고객’ VALUES
('C0001', '홍길동', 'abcd1234'),
('C0002', '양바른', 'ybl1234'),
('C0003', '유코식', 'uu1234'),
('C0004', '김구', 'pass1234'),
('C0005', '신사임당', 'pass1234');
```

주문 테이블에 삽입할 칼럼명을 나열하고 복수 개의 레코드를 삽입하는 명령문은 다음과 같다.

```
INSERT INTO ‘주문’ (주문번호, 고객번호, 주문일, 주문가격, 할인금액) VALUES
('O1001', 'C0001', '20211201', 10000, 100),
('O1002', 'C0001', '20211203', 4500, 45),
('O1003', 'C0004', '20211207', 100000, 1000),
('O1004', 'C0003', '20211207', 55000, 550),
('O1005', 'C0002', '20211217', 85000, 850),
('O1006', 'C0002', '20211218', 23000, 230),
('O1007', 'C0004', '20211221', 5000, 50),
('O1008', 'C0001', '20211222', 8300, 83),
('O1009', 'C0002', '20211225', 45000, 450),
('O1010', 'C0003', '20211231', 9000, 90);
```

즉 하나의 SQL문으로 복수의 레코드를 삽입하려면 각각의 레코드 정보를 연속해서 정의하면 된다.

2. 다중 테이블을 통해 데이터를 조회한다.

(1) 조인으로 다중 테이블을 조회한다.

앞서 고객 테이블과 주문 테이블에 삽입한 데이터를 조회하도록 한다. 우선 조건 절 없이 데이터를 조회해보도록 한다.

```
SELECT 고객.*, 주문.*  
FROM 고객, 주문 ;
```

이는 조건절이 없으므로 연결 가능한 모든 데이터를 조회하는 크로스 조인(Cross Join)의 결과를 출력하게 된다. 출력 건수는 고객 테이블의 5건과 주문 테이블의 10건으로 생성되는 모든 경우의 수인 50건이 조회된다.

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 고객 번호 | 이름 | 비밀번호 | 주문 번호 | 고객 번호 | 주문 일 | 주문 가격 | 배송 도시 | 배송 완료 일 | 결제 금액 | 할인 금액 | 적립 포인트 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| C0001 | 홍길동 | abcd1234 | 01001 | C0001 | 20211201 | 10000.00 | NULL | NULL | NULL | 100.00 | 0.00 |  
| C0002 | 양바른 | yb11234 | 01001 | C0001 | 20211201 | 10000.00 | NULL | NULL | NULL | 100.00 | 0.00 |  
| C0003 | 유코식 | uu1234 | 01001 | C0001 | 20211201 | 10000.00 | NULL | NULL | NULL | 100.00 | 0.00 |  
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |  
| C0002 | 양바른 | yb11234 | 01010 | C0003 | 20211231 | 9000.00 | NULL | NULL | NULL | 90.00 | 0.00 |  
| C0003 | 유코식 | uu1234 | 01010 | C0003 | 20211231 | 9000.00 | NULL | NULL | NULL | 90.00 | 0.00 |  
| C0004 | 김구 | pass1234 | 01010 | C0003 | 20211231 | 9000.00 | NULL | NULL | NULL | 90.00 | 0.00 |  
| C0005 | 신사임당 | pass1234 | 01010 | C0003 | 20211231 | 9000.00 | NULL | NULL | NULL | 90.00 | 0.00 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
50 rows in set (0.001 sec)
```

출처: 집필진 제작(2021)

[그림 1-10] 크로스 조인을 통한 명령문 결과

이번에는 고객 테이블과 주문 테이블에 있는 고객번호로 조인을 수행해보도록 한다. WHERE절을 사용해서 두 테이블의 공통 데이터를 추출하도록 작성한다.

```
SELECT 고객.고객번호, 고객.이름, 주문.주문번호, 주문.주문일, 주문.주문가격  
FROM 고객, 주문  
WHERE 고객.고객번호 = 주문.고객번호;
```

고객번호로 두 테이블에 동시 존재하는 데이터는 총 10건이다.

```
+-----+-----+-----+-----+-----+  
| 고객 번호 | 이름 | 주문 번호 | 주문 일 | 주문 가격 |  
+-----+-----+-----+-----+-----+  
| C0001 | 홍길동 | 01001 | 20211201 | 10000.00 |  
| C0001 | 홍길동 | 01002 | 20211203 | 4500.00 |  
| C0004 | 김구 | 01003 | 20211207 | 100000.00 |  
| C0003 | 유코식 | 01004 | 20211207 | 55000.00 |  
| C0002 | 양바른 | 01005 | 20211217 | 85000.00 |  
| C0002 | 양바른 | 01006 | 20211218 | 23000.00 |  
| C0004 | 김구 | 01007 | 20211221 | 5000.00 |  
| C0001 | 홍길동 | 01008 | 20211222 | 8300.00 |  
| C0002 | 양바른 | 01009 | 20211225 | 45000.00 |  
| C0003 | 유코식 | 01010 | 20211231 | 9000.00 |  
+-----+-----+-----+-----+-----+  
10 rows in set (0.001 sec)
```

출처: 집필진 제작(2021)

[그림 1-11] 내부 조인을 통한 명령문 결과1

마지막으로는 이름이 ‘홍길동’인 고객이 주문한 내역을 조회하고자 한다. 고객번호와 이름, 주문번호, 주문일, 주문가격을 조회하는 명령문은 아래와 같다.

```
SELECT 고객.고객번호, 고객.이름, 주문.주문번호, 주문.주문일, 주문.주문가격  
FROM 고객, 주문  
WHERE 고객.고객번호 = 주문.고객번호  
AND 이름 = '홍길동';
```

또는

```
SELECT 고객.고객번호, 고객.이름, 주문.주문번호, 주문.주문일, 주문.주문가격  
FROM 고객  
INNER JOIN 주문  
    ON 고객.고객번호 = 주문.고객번호  
WHERE 이름 = '홍길동';
```

고객 테이블과 주문 테이블을 내부 조인(Inner Join)하여, 각 테이블에 존재하는 레코드 값을 확인할 수 있다. 홍길동 고객의 주문 데이터가 2건 조회됨을 아래와 같이 확인할 수 있다.

```
+-----+-----+-----+-----+  
| 고객 번호 | 이름      | 주문 번호 | 주문 일   | 주문 가격 |  
+-----+-----+-----+-----+  
| C0001    | 홍길동    | 01001    | 20211201  | 10000.00  |  
| C0001    | 홍길동    | 01008    | 20211222  | 8300.00  |  
+-----+-----+-----+-----+  
2 rows in set (0.000 sec)
```

출처: 집필진 제작(2021)
[그림 1-12] 내부 조인을 통한 명령문 결과2

(2) 서브쿼리로 다중 테이블을 조회한다.

고객 테이블과 주문 테이블 간에 연관 서브쿼리를 사용하여 조인을 수행한다. 예인 쿼리 안의 고객 테이블과 서브쿼리 안의 주문 테이블 간에는 고객번호라는 칼럼으로 연관성이 있다. 즉, 서브쿼리 단독으로는 실행 불가능한 특징을 가지는 연관 쿼리의 사용 예는 다음과 같다.

고객은 다수의 주문을 요청하므로 1명의 고객에 N개의 주문 데이터가 발생할 수 있다. 즉, 고객별로 몇 번의 주문을 수행하였는지 개수를 조회하는 예시이다.

```
SELECT 고객.이름,  
       (SELECT COUNT(1)  
        FROM 주문  
       WHERE 고객.고객번호 = 주문.고객번호) AS 주문횟수  
  FROM 고객;
```

SELECT절에 연관 서브쿼리를 사용하면, 아래와 같이 4명의 고객별로 몇 번의 주문을 했는지 확인할 수 있다.

```
+-----+-----+
| 이름      | 주문 횟수 |
+-----+-----+
| 홍길동      |      2 |
| 양바른      |      4 |
| 유코식      |      2 |
| 김구        |      2 |
+-----+-----+
4 rows in set (0.000 sec)
```

출처: 집필진 제작(2021)
[그림 1-13] 연관 서브쿼리를 통한 명령문 결과

이번에는 비연관 서브쿼리를 수행해본다. 그 전에 서브쿼리 안에 들어가는 쿼리 문을 우선적으로 확인해본다.

```
SELECT 고객번호
FROM 주문;
```

위 쿼리의 결과는 아래와 같다. 주문 테이블에 있는 고객번호 데이터는 총 10건임을 알 수 있다.

```
+-----+
| 고객 번호 |
+-----+
| C0001    |
| C0001    |
| C0004    |
| C0003    |
| C0002    |
| C0002    |
| C0004    |
| C0001    |
| C0002    |
| C0003    |
+-----+
10 rows in set (0.001 sec)
```

출처: 집필진 제작(2021)
[그림 1-14] 비연관 서브쿼리를 통한 명령문 결과1

아래는 NOT IN 구절 안에 ()로 둘러싸인 독립된 SQL문인 비연관 서브쿼리이다. NOT IN은 괄호 안에 존재하지 않는 고객번호에 대해서만 고객 테이블의 고객번호를 검색하게 된다. 이는 고객 테이블이 포함된 메인쿼리와는 별도로 독립적 실행이 가능하기 때문에 비연관 서브쿼리라 한다. 아래 SQL문은 한번도 주문하지 않은 고객의 고객번호와 이름을 조회하는 예시이다.

```
SELECT 고객번호, 이름  
FROM 고객  
WHERE 고객.고객번호 NOT IN ( SELECT 고객번호  
                                FROM 주문 );
```

수행 결과는 다음과 같다.

```
+-----+-----+  
| 고객 번호 | 이름      |  
+-----+-----+  
| C0005    | 신사임당 |  
+-----+-----+  
1 row in set (0.001 sec)
```

출처: 집필진 제작(2021)
[그림 1-15] 비연관 서브쿼리를 통한 명령문 결과2

(3) 집합 연산으로 다중 테이블을 조회한다.

집합 연산의 일종인 UNION의 사용 형식은 다음과 같다.

```
SELECT column1, column2  
UNION [DISTINCT | ALL]  
SELECT column1, column2
```

두 집합 사이에서 집합 연산자가 작동하도록 SELECT문 사이에 연산자를 배치한다. 이때 UNION과 같은 집합 연산에서 다음과 같은 제약 조건이 존재한다.

- UNION으로 연결된 SELECT문에 나타나는 칼럼의 숫자는 같아야 한다.
- UNION으로 대응되는 각 칼럼의 데이터 타입이 같아야 한다.

다음은 UNION 집합 연산자를 사용하여 홍길동의 고객번호과 비밀번호를 세로 방향으로 출력해보도록 한다. 그 전에 집합 연산자로 연결될 각 SQL문과 그에 따른 결과를 확인하도록 한다. 첫 번째 SQL문은 아래와 같다.

```
SELECT 고객번호 FROM 고객 WHERE 이름 = '홍길동';
```

'홍길동'이라는 이름을 가진 고객의 고객번호는 1건 조회됨을 알 수 있다.

```
+-----+  
| 고객 번호 |  
+-----+  
| C0001 |  
+-----+  
1 row in set (0.001 sec)
```

출처: 집필진 제작(2021)
[그림 1-16] 집합 연산자 관련 명령문 결과1

두 번째 SQL문은 아래와 같다.

```
SELECT 비밀번호 FROM 고객 WHERE 이름 = '홍길동';
```

'홍길동'이라는 이름을 가진 고객의 비밀번호는 1건 조회됨을 알 수 있다.

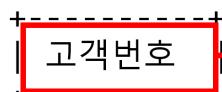
```
+-----+  
| 비밀번호 |  
+-----+  
| abcd1234 |  
+-----+  
1 row in set (0.001 sec)
```

출처: 집필진 제작(2021)
[그림 1-17] 집합 연산자 관련 명령문 결과2

이번에는 위의 두 SQL문을 UNION 집합 연산자로 이은 SQL문이다.

```
SELECT 고객번호 FROM 고객 WHERE 이름 = '홍길동'  
UNION  
SELECT 비밀번호 FROM 고객 WHERE 이름 = '홍길동';
```

UNION으로 묶은 두 개의 집합, 테이블에서 선택된 칼럼의 의미가 달라도 제약 조건을 만족하므로 이 명령은 다음과 같이 적절히 수행된다.

 앞 테이블의 칼럼 이름이 사용됨

```
+-----+  
| 고객번호 |  
+-----+  
| C0001 |  
| abcd1234 |  
+-----+  
2 rows in set (0.000 sec)
```

출처: 집필진 제작(2021)
[그림 1-18] 집합 연산자를 통한 명령문 결과

위와 같은 UNION은 각 집합의 결괏값에서 중복된 레코드는 삭제하기 때문에, 흥길동의 고객번호와 비밀번호가 중복되지 않는다면 UNION 대신 UNION ALL 을 사용하는 것이 바람직하다. UNION ALL은 중복 작업 없이 집합 간의 결과를 통합해주기 때문에, 불필요한 내부 중복 작업을 줄일 수 있기 때문이다.

⑤ 트랜잭션(transaction) 제어 관련 DCL을 수행한다(M*-SQL 기준).

트랜잭션 발생이 필요하므로, 별도의 시나리오를 통해 가정한다.

1. 트랜잭션을 분석한다.

다음과 같은 계좌 이체 사례를 통해서 트랜잭션 과정을 분석하도록 한다.

- 나의 계좌에서 1,000원을 인출한다.
- 상대방의 계좌에 1,000원을 입금한다.

위와 같이 계좌 이체 과정은 최소 2번의 작업이 필요하다. 물론 2번의 작업을 하나로 합쳐서 ‘인출한 후에 입금한다’라는 최소 단위로 말할 수 있지만, 이는 계좌 이체 과정을 사람의 문장으로 통합한 것이지, 실질적인 DBMS 처리 관점에서는 하나의 작업이 될 수 없다.

그렇다면 만약 인출과 입금이라는 2번의 작업 도중에 의도하지 않은 문제가 발생했다면 어떻게 될까? 다시 말하면, 1번이나 2번 작업이 실패했다면 어떠한 결과가 발생할까? 그렇다면 상대방의 계좌에 알 수 없는 1,000원이 입금되거나, 내 계좌에서 1,000원만 인출되고 끝나버리는 현상이 발생할 것이다. 이러한 일련의 문제를 해결하기 위해서 트랜잭션이라는 개념을 사용한다. 트랜잭션은 연관되어 있는 여러 개의 작업을 하나의 작업 단위로 묶어서 ‘분해할 수 없는 최소 단위’로 만드는 것이다. 다음의 통해 트랜잭션의 특징을 확인한다.

〈표 1-28〉 트랜잭션의 특징

특징	내용	비고
원자성 (Atomicity)	트랜잭션 내에서 모두 실행되거나 모두 실행되지 않음	All or Nothing
일관성 (Consistency)	트랜잭션의 실행 전/후 결과가 동일해야 함	무결성
고립성 (Isolation)	트랜잭션이 실행되는 도중에 다른 트랜잭션의 영향을 받지 않음	독립성
지속성 (Durability)	트랜잭션의 결과는 지속적으로 저장되어 보존됨	장애 대응성

데이터베이스에서 일관성은 상태가 항상 같아야 함을 의미한다. 이때 상태라는 것은 이전에 오류가 있었으면 이후에도 오류가 있어야 하며, 이전에 오류가 없었으면 이후에도 오류가 없어야 함을 의미한다. 이 가운데 오류가 없는 상태를 무결성이라고 하

며, 무결성은 정확성, 일관성, 유일성, 신뢰성 있는 상태를 한 번에 표현하는 것이다. 언제 어떠한 형태의 실패에도 안전한 거래를 보장하는 수단이 바로 트랜잭션이다.

2. 메모리 측면의 트랜잭션 조작을 확인한다.

트랜잭션의 수행 원리를 쉽게 이해하기 위하여, 다음을 통해 데이터베이스 관리 방법을 확인한다.

- DBMS에서 수행되는 연산은 메모리에서 처리됨
 - DBMS를 통해 보관해야 할 정보는 하드디스크에 저장됨
 - 하드디스크에 저장된 정보를 메모리로 가져와서 연산함
 - 내부적으로 정해진 시점에 메모리의 정보들은 하드디스크로 전달하여 저장됨
- 위와 같은 메모리를 활용한 데이터베이스 관리 방법은 메모리를 활용하는 TCL문과 대응할 수 있다. 다음을 통해서 TCL 명령어를 확인한다.

〈표 1-29〉 TCL 명령어의 메모리 관점 동작

명령어	내용	메모리 조작
COMMIT	메모리의 내용을 하드디스크에 저장함	영구히 저장
ROLLBACK	메모리의 내용을 하드디스크에 저장하지 않고 버림	메모리 내용 무효화
CHECKPOINT	ROLLBACK 범위 설정을 위해 메모리상 경계를 설정함	상태 기억

3. 트랜잭션 조작을 확인한다.

다음은 임의로 발생된 트랜잭션에 대한 시나리오이다. 1번부터 6번까지 실행된 트랜잭션이 7번에서 ROLLBACK을 수행하려고 한다. 다음 시나리오를 통해서 트랜잭션 순서를 확인한다.

〈표 1-30〉 트랜잭션 시나리오

순서	명령문	비고
1	트랜잭션 시작	
2	INSERT	
3	SAVEPOINT P1	P1이라는 이름으로 복구 지점 설정
4	UPDATE	
5	SAVEPOINT P2	P2라는 이름으로 복구 지점 설정
6	DELETE	
7	(현재 위치에서 ROLLBACK 예정)	복구 지점 사용에 따라 결과 달라짐

7번에서 ROLLBACK을 수행한다면, 다음의 사용 명령에 따라 수행결과가 어떻게 달라지는지 확인한다.

〈표 1-31〉 트랜잭션 ROLLBACK 경우

7번째 명령문	결과
ROLLBACK	1번쨰 명령까지 취소됨 즉, 현재 위치 이전 트랜잭션 처리 내용이 모두 취소됨
ROLLBACK TO P1	1~3까지의 명령은 유효하게 남고, 4~6까지의 내용이 취소됨
ROLLBACK TO P2	1~5까지의 명령은 유효하게 남고, 6 내용이 취소됨

4. Auto Commit을 이용한다.

실무에서 사용하는 TCL문 중에서 자동으로 COMMIT되는 Auto Commit 기능에 대한 이해가 필요하다. 주로 사용하는 DBMS 제품의 대부분은 Auto Commit 기능을 지원하며, M*-SQL 경우에는 기본값으로 '사용함'으로 설정되어 있다. 해당 기능은 COMMIT을 명시적으로 작성하지 않더라도 DML문이 정상적으로 수행되면 자동으로 COMMIT 처리가 된다. 반면 DML문이 실패하면 자동으로 ROLLBACK 처리가 된다. M*-SQL에서는 현재 설정된 Auto Commit 여부를 확인하기 위해서는 다음 명령어를 사용한다.

```
SELECT @@AUTOCOMMIT;
```

위의 수행 결과가 1(True)이라면 Auto Commit이 설정된 것이고 0(False)이라면 Auto Commit이 설정되지 않은 것을 의미한다. Auto Commit에 대해서 원하는 설정값으로 바꾸기 위해서는 다음 명령을 활용한다.

```
SET AUTOCOMMIT = TRUE; -- Auto Commit 설정  
SET AUTOCOMMIT = FALSE; -- Auto Commit 미설정(해지)
```

5. 트랜잭션을 제어한다(M*-SQL 기준).

다음을 통해 트랜잭션으로 인한 데이터의 변경 상태를 확인한다.

<표 1-32> 트랜잭션 제어 SQL문

위치	트랜잭션 제어 SQL문
1	START TRANSACTION;
2	SAVEPOINT P1;
3	INSERT INTO salaries VALUES (100, 500, '2021-07-01', '2022-06-30');
4	SAVEPOINT P2;
5	UPDATE salaries SET salary = 600 WHERE emp_no = 100;
6	ROLLBACK TO P2;
7	ROLLBACK TO P1;

각 위치별 수행 결과는 다음과 같다.

위치 1~3 실행 후 결과는 salaries 테이블에 salary 값이 500인 레코드가 하나 추가된다.

```
> START TRANSACTION;
Query OK, 0 rows affected (0.00sec)

> SAVEPOINT P1;
Query OK, 0 rows affected (0.00sec)

> INSERT INTO salaries (emp_no, salary, from_date, to_date)
VALUES (100, 500, '2021-07-01', '2022-06-30');
Query OK, 1 row affected (0.00sec)

> SELECT * FROM salaries;
+-----+-----+-----+-----+
| emp_no | salary | from_date | to_date |
+-----+-----+-----+-----+
|    100 |     500 | 2021-07-01 | 2022-06-30 |
+-----+-----+-----+-----+
1 row in set (0.00sec)
```

출처: 집필진 제작(2021)

[그림 1-19] 위치 1~3 수행 후 데이터 모습

위치 5의 UPDATE 명령에 의해 salary 값이 600으로 변경되며, 현재의 salary 값은 500이다.

```
> SAVEPOINT P2;
Query OK, 0 rows affected (0.00sec)

> UPDATE salaries SET salary = 600 WHERE emp_no = 100;
Query OK, 1 row affected (0.00sec)
Rows matched: 1 Changed: 1 Warnings : 0

> SELECT * FROM salaries;
+-----+-----+-----+-----+
| emp_no | salary | from_date | to_date |
+-----+-----+-----+-----+
|    100 |     600 | 2021-07-01 | 2022-06-30 |
+-----+-----+-----+-----+
1 row in set (0.00sec)
```

출처: 집필진 제작(2021)

[그림 1-20] 위치 6 수행 후 데이터 모습

저장 지점 P2로 가서 직전의 UPDATE가 취소되면 salary 값은 500이 된다.

```
> ROLLBACK TO P2;  
Query OK, 0 rows affected (0.00sec)
```

```
> SELECT * FROM salaries;  
+-----+-----+-----+-----+  
| emp_no | salary | from_date | to_date |  
+-----+-----+-----+-----+  
| 100    | 500   | 2021-07-01 | 2022-06-30 |  
+-----+-----+-----+-----+  
1 row in set (0.00sec)
```

출처: 집필진 제작(2021)

[그림 1-21] 위치 6 수행 후 데이터 모습

저장 지점 P1으로 가서 INSERT를 취소하면 salaries에 데이터가 없는 최초 상태로 돌아가게 된다.

```
> ROLLBACK TO P1;  
Query OK, 0 rows affected (0.01sec)
```

```
> SELECT * FROM salaries;  
Empty set (0.00sec)
```

출처: 집필진 제작(2021)

[그림 1-22] 위치 7 수행 후 데이터 모습

수행 tip

- SQL 명령문은 데이터베이스 및 SQL 언어의 종류에 따라 조금씩 달라질 수 있다. 그러므로 사용자가 보유한 데이터베이스 및 SQL 언어를 고려하여 수행해야 한다.
- 테이블을 생성할 때, 테이블의 물리적 구성 역시 고려 대상이다. 테이블이 디스크에 저장되는 주요 방식으로 Heap Organized Table, Clustered Index Table, Partitioned Table 등이 있다. 저장 방식의 선택은 테이블의 성능뿐 아니라 확장성, 가용성 등을 고려하여 테이블 저장 형태를 선택해야 한다.
- DML 활용의 예시로 'select *' 형태의 명령문을 제시하고 있지만 이러한 사용 방법은 자제되어야 한다. 모든 칼럼을 조회 대상으로 하는 '*'의 사용은 불필요한 칼럼 정보를 꺼내오는 시간과 이를 전달하는 네트워크 상의 부하를 유발하므로 성능을 위해서는 사용 자제가 권고된다.

학습 1 교수·학습 방법

교수 방법

- DDL 개념과 명령어의 종류에 대해 설명한다.
- DDL 명령어 각각의 사용법을 제시한다.
- 조인과 서브쿼리, 집합 연산자에 대해 설명한다.
- 다중 테이블에 대한 검색 사용법을 제시한다.
- DCL 개념과 명령어의 종류에 대해 안내한다.
- DCL 명령어 각각의 사용법을 보여준다.
- 테이블과 관련된 데이터 사전의 개념과 사용법에 대해 설명한다.

학습 방법

- DDL 개념과 명령어의 종류에 대해 기술한다.
- DDL 명령어 각각의 사용법을 구별한다.
- 조인과 서브쿼리, 집합 연산자에 대해 연습한다.
- 다중 테이블에 대한 검색을 실습한다.
- DCL 개념과 명령어의 종류에 대해 작성한다.
- DCL 명령어 각각의 사용 방법을 요약한다.
- 테이블과 관련된 데이터 사전의 개념과 사용법에 대해 파악한다.

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
테이블의 데이터 사전 조회	- 생성된 테이블의 목록, 테이블의 구조와 제약 조건을 파악하기 위해 데이터 사전을 조회하는 명령문을 작성할 수 있다.			
기본적인 SQL 작성	- 조인, 서브쿼리, 집합 연산자를 사용하여 두 개 이상의 테이블로부터 데이터를 조회하는 DML(Data Manipulation Language) 명령문을 작성할 수 있다.			
	- 테이블의 구조와 제약 조건을 생성, 삭제하고 수정하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.			
	- 업무 단위인 트랜잭션의 완료와 취소를 위한 DCL(Data Control Language) 명령문을 작성할 수 있다.			

평가 방법

- 서술형 시험

학습 내용	평가 항목	성취수준		
		상	중	하
테이블의 데이터 사전 조회	데이터 사전의 개념과 구성 요소를 서술할 수 있는지 평가			
	데이터 사전의 주요 내용을 검색해 의미를 파악할 수 있는지 평가			
기본적인 SQL 작성	원하는 조건의 데이터만 선택적으로 조회할 수 있는 SQL문 작성 능력에 대해 평가			
	테이블 생성문 또는 ALTER문을 통해 제약 조건을 추가할 수 있는지 평가			
	Auto Commit 설정 변경에 따른 Commit 효과를 구분할 수 있는지 평가			

• 사례 연구

학습 내용	평가 항목	성취수준		
		상	중	하
테이블의 데이터 사전 조회	DBMS별 데이터 사전의 구성 요소 조사			
	DBMS별 데이터 사전의 요소별 내용 조사			
	복잡한 조건을 가지는 데이터 SQL문			
기본적인 SQL 작성	데이터 타입별 값의 범위 조사			
	Commit 기능 및 동작 조건 조사			

피드백

1. 서술형 시험
 - 기본적인 SQL 명령문의 사용 능력을 평가한 후, 미흡한 부분에 대해 설명해 준다. 이때 미흡한 부분은 이해 부족에서 발생하는 것이 대부분이기에, 기본 SQL 작성 주제, 영역별 SQL 사용 방법에 대한 지도 이외에, 배경 지식에 대한 설명이 요구된다.
 - 본 학습 모듈에서는 배경 지식을 설명하는 의도로 SQL 사용 방법 이외에 SQL이 무엇인지 생각해 가는 과정을 제시하였다. 이러한 의도를 참고하여 배경 지식에 대한 설명 혹은 학습이 이루어지도록 한다.
 2. 사례 연구
 - SQL 활용의 배경 설명 또는 심화 학습 방향을 설명해 준다.
 - 데이터 타입의 경우, 표현할 수 있는 값의 범위를 설명해 준다.
 - 교육용 데이터베이스를 검색한 후 학습자에게 제공하여, 교육의 밀도를 높인다.

2-1. 테이블 외의 데이터 사전 조회

학습 목표

- 생성된 테이블의 목록, 테이블의 인덱스와 뷰를 파악하기 위해 데이터 사전을 조회하는 명령문을 작성할 수 있다.

필요 지식 /

① 테이블 외의 데이터 사전 검색

데이터 사전에는 데이터베이스 객체인 테이블 외에도 인덱스, 뷰에 대한 메타 데이터(Meta data)를 확인할 수 있다. 테이블뿐만 아니라 인덱스와 뷰에 대한 데이터 사전도 단순히 조회만 가능하며, DBMS 제품에 따라서 데이터 사전의 조회 방식, 관리 방식에 차이가 있다.

1. 오*에서 데이터 사전 검색

앞서 1-1에서 설명한 것과 같이 오* 사용자는 뷔로 데이터 사전에 접근할 수 있다. 오*에서 뷔로 만들어진 데이터 사전은 3가지 영역으로 구분하며, 다음은 오브젝트에 접근 가능한 사용자 권한의 영역이다.

DBA_ > ALL_ > USER_

테이블, 뷔, 인덱스에 대한 오브젝트의 데이터 사전은 다음과 같다.

〈표 2-1〉 오* 데이터 사전 영역

영역	검색 범위	테이블 관련 데이터 사전	용도
DBA_	데이터베이스의 모든 객체 조회 가능 (DBA_는 시스템 접근 권한 의미)	DBA_TABLES DBA_INDEXES DBA_IND_COLUMNS DBA_VIEWS	모든 테이블 목록 확인 모든 인덱스 목록 확인 모든 인덱스의 구성 칼럼 확인 모든 뷰 목록 확인
	자신의 계정으로 접근 가능한 객체와 타 계정의 접근 권한을 가진 모든 객체 조회 가능	ALL_TABLES ALL_INDEXES ALL_IND_COLUMNS	권한 있는 테이블 목록 확인 권한 있는 인덱스 목록 확인 권한 있는 인덱스의 구성 칼럼 확인
		ALL_VIEWS	권한 있는 뷰 목록 확인
		USER_TABLES USER_INDEXES USER_IND_COLUMNS	자기 계정의 테이블 목록 확인 자기 계정의 인덱스 목록 확인 자기 계정의 인덱스 구성 칼럼 확인
USER_	현재 자신의 계정이 소유한 객체 조회 가능	USER_VIEWS	자기 계정의 뷰 목록 확인

2. M*-SQL에서 데이터 사전 검색

M*-SQL에서 데이터 사전은 Information_schema라는 데이터베이스 안에 존재한다. 또는 SQL문에서 information_schema라는 데이터베이스를 지정하고 데이터 사전 목록을 조회할 수 있다.

- use Information_schema; -- information_schema로 이동
- show tables; -- 데이터 사전 목록 보기

또는

```
select *
from information_schema.tables;
```

테이블 목록으로 데이터 사전을 구성하는 테이블 이름을 확인하고, SELECT문을 통해 해당 테이블의 내용을 조회할 수 있다. 테이블과 인덱스, 뷰에 관련된 데이터 사전은 다음과 같다.

〈표 2-2〉 M*-SQL 데이터 사전 영역

데이터베이스 명	데이터 사전 테이블	용도
information_schema	TABLES	데이터베이스에 존재하는 테이블 정보
	TABLE_CONSTRAINTS	테이블에 설정된 제약 사항 정보
	KEY_COLUMN_USAGE	제약 사항을 가지고 있는 키 칼럼에 대한 정보
	COLUMNS	테이블을 구성하는 칼럼에 대한 정보
	VIEWS	DB에 있는 뷰에 대한 정보
	STATISTICS	테이블의 인덱스에 대한 정보

수행 내용 / 테이블 외의 데이터 사전 조회하기

재료·자료

- SQL 언어 문법 학습 자료
- DBMS 매뉴얼 및 튜토리얼 자료

기기(장비 · 공구)

- DBMS 프로그램

안전 · 유의 사항

- 데이터 사전 용도에 대한 추가 학습이 필요하다.
- 데이터베이스 제품별 사용 방법의 차이가 크다. 특히 데이터 사전이란 개념을 구현하고 사용하는 방식에 있어 SQL 사용법은 제품별로 큰 차이가 있기에, 각 제품별 데이터 사전 활용 방법에 대한 학습이 별도로 필요하다.

수행 순서

① 데이터 사전을 검색한다(M*-SQL 기준).

1. 데이터베이스를 선택한다.

데이터 사전 테이블을 관리하는 데이터베이스로 이동하여 테이블의 목록을 살펴보면 다음 그림과 같은 결과를 얻을 수 있다.

```

> use Information_schema;
Database changed
> show tables;
+-----+
| Tables_in_information_schema |
+-----+
| CHARACTER_SETS
| COLLATIONS
| COLLATION_CHARACTER_SET_APPLICABILITY
| COLUMNS
| COLUMN_PRIVILEGES
| ENGINES
| EVENTS
| FILES
.....

```

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.37.
[그림 2-1] 데이터 사전 테이블 목록

2. 테이블 및 인덱스, 뷰 관련된 데이터 사전 테이블을 살펴본다.

테이블 외의 인덱스와 뷰에 대한 데이터 사전을 확인하고 의미를 파악한다.

〈표 2-3〉 데이터 사전 주요 테이블의 핵심 주제

데이터 사전 테이블	주요 내용
COLUMNS	테이블 칼럼에 대한 컬렉션 정보
COLUMN_PRIVILEGES	테이블 칼럼 권한에 대한 정보 제공
TABLES	데이터베이스에 존재하는 테이블에 대한 정보 제공
TABLE_CONSTRAINTS	테이블에 대한 제약 사항에 대한 정보
KEY_COLUMN_USAGE	제약 사항을 가지고 있는 키 칼럼에 대한 정보 제공
VIEWS	DB에 있는 뷰에 대한 정보 제공
STATISTICS	테이블의 인덱스에 대한 정보 제공

3. 테이블 및 인덱스, 뷰 관련된 데이터 사전 내용을 검색한다.

(1) 테이블 목록과 칼럼 목록을 조회한다.

다음과 같은 SQL문을 통해 생성된 테이블 목록과 구성된 칼럼을 확인할 수 있다. columns라는 데이터 사전을 통해서 ncs라는 데이터베이스에 생성된 테이블과 칼럼 목록을 조회한다.

```

> SELECT TABLE_NAME      as 테이블명,
>        COLUMN_NAME     as 컬럼명,
>        ORDINAL_POSITION as 컬럼순서,
>        COLUMN_TYPE      as 데이터타입
>   FROM information_Schema.COLUMNS
> WHERE table_schema='ncs';
+-----+-----+-----+-----+
| 테이블명 | 컬럼명     | 컬럼순서 | 데이터타입 |
+-----+-----+-----+-----+
| tab1    | MEMBER_ID  | 1 | varchar(10) |
| tab1    | NAME       | 2 | varchar(50) |
| tab1    | CODE       | 3 | varchar(2)  |
| tab2    | MEMBER_ID  | 1 | varchar(10) |
| tab2    | NAME       | 2 | varchar(50) |
| tab2    | CODE       | 3 | varchar(2)  |
| tab3    | MEMBER_ID  | 1 | varchar(10) |
| tab3    | NAME       | 2 | varchar(50) |
| tab3    | CODE       | 3 | varchar(2)  |
+-----+-----+-----+-----+
9 rows in set (0.031 sec)

```

출처: 집필진 제작(2021)

[그림 2-2] 테이블 목록과 컬럼 목록의 조회 결과

(2) 기본 키 또는 유일 키를 조회한다.

생성된 기본 키 또는 유일 키(Unique Key)에 대해서 다음과 같이 조회한다. KEY_COLUMN_USAGE라는 데이터 사전을 통해서 ncs 데이터베이스 안에 생성된 키 목록을 조회한다.

```

> SELECT CONSTRAINT_NAME, TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION
>   FROM information_schema.KEY_COLUMN_USAGE
> WHERE TABLE_SCHEMA = 'ncs';
+-----+-----+-----+-----+
| CONSTRAINT_NAME | TABLE_NAME | COLUMN_NAME | ORDINAL_POSITION |
+-----+-----+-----+-----+
| PRIMARY        | tab1      | MEMBER_ID   | 1 |
| PRIMARY        | tab2      | MEMBER_ID   | 1 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

출처: 집필진 제작(2021)

[그림 2-3] key_column_usage 데이터 사전 조회

(3) 인덱스 목록을 조회한다.

생성된 인덱스 목록은 다음과 같이 조회한다. STATISTICS라는 데이터 사전을 통해서 ncs 데이터베이스 안에 생성된 인덱스 목록을 모두 조회한다. 이때 STATISTICS 결과는 KEY_COLUMN_USAGE로 출력된 결과를 포함하게 된다.

```

> SELECT TABLE_NAME  as 테이블명,
>        INDEX_NAME  as 인덱스명,
>        COLUMN_NAME as 인덱스_구성컬럼
>   FROM information_Schema.STATISTICS
> WHERE table_schema='ncs';
+-----+-----+-----+
| 테이블명 | 인덱스명 | 인덱스_구성컬럼 |
+-----+-----+-----+
| tab1    | PRIMARY  | MEMBER_ID
| tab1    | IDX_CODE | CODE
| tab2    | PRIMARY  | MEMBER_ID
+-----+-----+-----+
3 rows in set (0.001 sec)

```

출처: 집필진 제작(2021)
[그림 2-4] statistics 데이터 사전 조회

(4) 뷰 목록을 조회한다.

생성된 뷰 목록은 다음과 같이 조회한다. VIEWS라는 데이터 사전을 통해서 전체 데이터베이스에서 생성된 뷰 목록을 모두 조회한다.

```

> SELECT TABLE_SCHEMA AS 데이터베이스,
>        TABLE_NAME   AS 뷰명,
>        VIEW_DEFINITION AS 뷰_스크립트
>   FROM information_Schema.VIEWS;
+-----+-----+-----+
| 데이터베이스 | 뷰명 | 뷰_스크립트 |
+-----+-----+-----+
| mysql       | user | select mysql`.`global_priv`.... |
+-----+-----+-----+
1 row in set (0.068 sec)

```

출처: 집필진 제작(2021)
[그림 2-5] views 데이터 사전 조회

수행 tip

- 데이터 사전 검색'에서 DBMS 구조를 살펴보는 이유는 트랜잭션, 인덱스, 복구와 같은 SQL 활용에서 DBMS의 구조가 해당 기술의 의도나 방법을 결정하는 데 영향을 끼치기 때문이다.

2-2. 인덱스와 뷰 작성

학습 목표

- 테이블 조회 시간을 단축하기 위해 사용하는 인덱스의 개념을 이해하고 인덱스를 생성하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.
- 먼저 생성된 테이블들을 이용하여 새로운 테이블과 뷰를 생성하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.

필요 지식

① 인덱스 활용

1. 인덱스 개념

인덱스는 저장된 데이터를 빠르게 검색할 수 있는 수단이자, 테이블에 대한 조회 속도를 높여 주는 자료 구조를 말한다. 다음 그림과 같은 인덱스(이름 칼럼)는 테이블에 있는 특정 레코드 위치를 알려주는 수단으로 사용하며, 사용자에 의해서 인덱스가 생성되어야 활용할 수 있다.

index_name		table_create_men			
Index (이름)	주소	일련번호	이름	생년월일	출생지
김구	5	1	장보고	07871111	전라남도 완도군
박문수	4	2	홍길동	14431111	전라남도 장성군
윤동주	6	3	이순신	15450428	서울시 중구 인현동
이순신	3	4	박문수	16911111	경기도 평택시
장보고	1	5	김구	18760829	황해도 해주시
홍길동	2	6	윤동주	19171230	중국 연변 연정시

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.42.

[그림 2-6] 인덱스 개념

반면 PK 칼럼(일련번호 칼럼)으로 만들어진 PK 인덱스는 자동으로 생성된다. 요약하면 PK 칼럼은 기본 키를 생성할 때 데이터베이스에 의해서 자동으로 PK 인덱스가 생성된다.

예를 들어 위의 [그림 2-6]과 같은 테이블에서 일련번호를 기본 키(Primary Key)로 하는 경우, 일련번호에 대한 인덱스는 자동으로 생성되나, 생년월일이나 이름을 기준으로 하는 인덱스는 자동으로 생성되지 않는다. 다음 질의문을 보자.

```
select * from table_create_men where 이름 = '이순신';
```

조건문 where 절에서 '이름'을 비교하고 있다. 이 경우 해당 테이블의 '이름' 칼럼에 인덱스가 없는 경우, 테이블의 전체 내용을 검색(Full Table scan)하게 된다.

반면, 인덱스가 생성되어 있다면 테이블의 일부분을 검색(Unique Scan 또는 Range scan)하여 데이터를 빠르게 찾을 수 있다. 조건절에 '='로 비교되는 칼럼을 대상으로 인덱스를 생성하면 검색 속도를 높일 수 있다. 하지만 자동으로 생성되지 않기 때문에 DB 사용자는 질의문을 분석하여 인덱스를 생성해야 한다.

2. 인덱스 사용

(1) 인덱스 사용 주체

[그림 2-7]에서 '이름' 칼럼에 대한 인덱스가 생성되어 있다면 데이터를 빠르게 찾을 수 있다. 이 때 DBMS는 인덱스를 사용하여 빠른 검색을 수행한다. 이를 위해 DB 사용자는 DBMS가 인덱스를 사용할 수 있게 준비해 주어야 한다. 따라서 사용자 입장에서는 인덱스를 사용한다는 개념보다는 검색 속도를 향상시키기 위해 준비한다는 개념으로 DBMS에 접근해야 한다. 이에 인덱스 준비 방법에 대해 알아보도록 하자.

(2) 인덱스 준비

DB 사용자가 인덱스에 대해 조작할 수 있는 방법으로는 '생성, 삭제, 그리고 변경' 조작이 있다. 참고로 인덱스를 사용하는 명령어는 SQL 표준화에 포함되지 않으므로 DBMS 제품 공급사마다 약간의 차이가 있다. 여기서는 각 조작에 대한 개념을 익히도록 하자.

(가) 인덱스 생성

인덱스 생성 문법은 다음과 같다.

```
CREATE [UNIQUE] INDEX <인덱스명> ON <테이블명> (<칼럼명 나열>);
```

여기서 각각의 파라미터가 의미하는 내용은 다음과 같다.

〈표 2-4〉 인덱스 명령문 요소

파라미터	내용	비고
[UNIQUE]	인덱스 걸린 칼럼에 중복값을 허용하지 않음 (생략 가능)	CREATE 테이블에서 사용하는 UNIQUE 제약 조건과 동일한 의미
<인덱스명>	생성하고자 하는 인덱스 이름을 작성	
<테이블명>	인덱스 대상인 테이블 이름을 작성	
<칼럼명 나열>	인덱스 대상 테이블의 특정 칼럼 이름(들)	복수 칼럼 지정 가능

(나) 인덱스 삭제

인덱스 삭제 명령 형식은 다음과 같다.

```
DROP INDEX <index name>;
```

<index_name>은 생성된 인덱스 이름을 의미한다. 인덱스 관련 명령어에 대한 SQL 표준이 없기에 제품별 Drop 명령문의 사용법은 약간씩 다르다. 보통 인덱스를 테이블의 종속 구조로 생각하여 인덱스를 삭제하기 위해 테이블에 변경을 가하는 형식의 명령을 사용한다. 즉, ALTER TABLE 명령 뒤에 DROP INDEX 명령이 추가되는 형태로 사용된다.

(다) 인덱스 변경

인덱스에 대한 정의를 변경하는 명령문 형식은 다음과 같다.

```
ALTER [UNIQUE] INDEX <index name> ON <table name> (<column(s)>);
```

한 번 생성된 인덱스에 대해 변경이 필요한 경우는 드물다. 또 인덱스 관련 SQL문은 표준화가 안 되었으므로 인덱스 변경에 대한 명령문 지원 여부 및 방법은 벤더별로 다르다. 일부 제품은 인덱스에 대한 변경 SQL문이 없다. 이 경우 기존 인덱스를 삭제하고 신규 인덱스를 생성하는 방식으로 사용이 권고되고 있다.

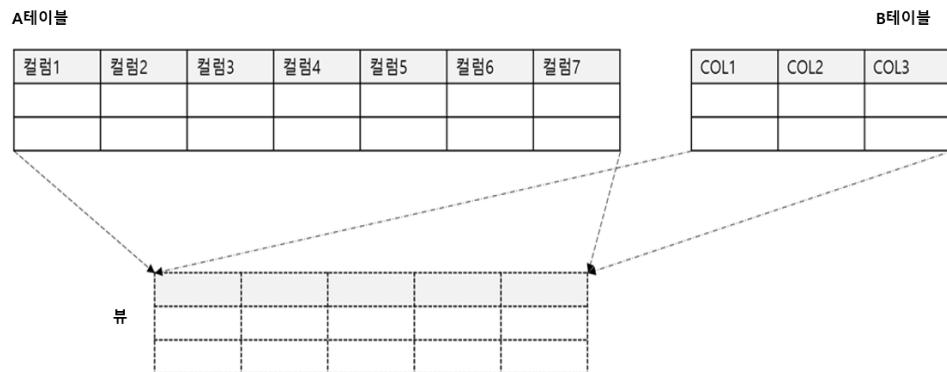
② 뷰(View) 개요

1. 뷰 개념

뷰는 기본 테이블로부터 유도된 가상 테이블로서, 사용자에게(생성 관점 아닌 사용 관점에서)는 물리적으로 존재하지 않지만 테이블로 있는 것처럼 간주된다. 아래 [그림 2-7]은 물리 테이블인 ‘A테이블’과 ‘B테이블’을 활용하여 뷰를 만드는 모습을 나타내고 있다.

뷰는 ‘A테이블’과 같은 하나의 물리 테이블로부터 생성 가능하며, 다수의 테이블 또는 다른 뷰를 이용해 만들 수 있다. 위에서 생성한 뷰는 다음 장에서 학습할 조인 (Join) 기능을 사용하며, 이미 만들어진 뷰가 있다면 조인 없이 하나의 테이블을 대상으로 질의하는 것처럼 SQL문을 작성하면 된다.

```
SELECT * FROM <View Name>;
```



출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.44.

[그림 2-7] 뷰 개념도

즉 FROM 절에 작성한 <뷰>를 통해 뷰를 만들 때 활용한 다수의 <테이블>을 대체할 수 있다. 또한 이러한 기능을 통해 테이블의 중요 데이터 일부만을 제공할 수 있는 등 다음과 같은 장점이 있다.

〈표 2-5〉 뷰의 장점

뷰 장점	내용
논리적 독립성 제공	뷰는 논리 테이블(테이블의 구조가 변경되더라도 뷰를 활용하는 응용 프로그램도 항상 수정할 필요는 없음)
사용자 데이터 관리 용이	다수 테이블에 있는 다양한 데이터에 대해 단순한 질의어 활용 가능
데이터 보안 용이	중요한 보안 데이터가 있는 테이블은 접근하지 못하도록 설정하고, 접근이 가능한 일부 데이터만을 조회할 수 있도록 뷰를 생성함

반면에 다음과 같은 단점이 있다.

〈표 2-6〉 뷰의 단점

뷰 단점	내용
뷰의 인덱스 사용 불가	논리적 생성한 뷰에는 인덱스를 만들 수 없음
뷰 구조 변경 불가	뷰는 삭제 후 재생성을 통해서 뷰의 구조 변경이 가능함
데이터 변경 제약 존재	뷰로 조회된 데이터에 대한 삽입, 변경, 삭제 제약이 있음.

뷰를 사용하기 위해서는 우선 뷰를 만들어야 한다. 뷰의 단순한 사용은 생성 과정에 의존적이다. 즉, 뷰를 어떻게 생성하였느냐에 따라 사용 방법이 달라진다. 뷰의 생성 방법에 대해 알아보도록 하자.

2. 뷰 생성

뷰 생성 명령의 일반 형태는 다음과 같다.

```
CREATE VIEW <뷰 이름>(칼럼 목록) AS <뷰를 통해 보여줄 데이터 조회용 쿼리문>
```

앞의 [그림 2-7]에서 상황별 뷰를 생성하는 방법은 다음과 같다.

〈표 2-7〉 뷰 생성 방법

상황	뷰 생성 쿼리문
테이블A 그대로	CREATE VIEW 뷰A AS select * from 테이블A;
테이블A 일부 칼럼	CREATE VIEW 뷰X AS select 칼럼1, 칼럼2, 칼럼3 from 테이블A;
테이블A와 테이블B 조인 결과	CREATE VIEW 뷰Y AS select * from 테이블A a, 테이블B b where a.칼럼1=b.COL1;

조회문의 다양한 변형에 따라 뷰의 내용이 달라진다. 즉, [그림 2-9]의 경우 ‘테이블A’와 ‘테이블B’로부터 조회 가능한 형태 모두가 뷰로 치환될 수 있다.

3. 뷰 삭제 및 변경

뷰의 구조(정의)를 변경하는 것은 불가능하다. 뷰의 구조가 만들어졌으면, 물리적인 요소는 뷰 이름과 뷰를 조회하기 위한 쿼리문만 해당된다. 이때 뷰의 이름이나 쿼리문을 변경하는 수단은 제공되지 않는다. 이 경우 뷰의 삭제와 재생성을 통해 뷰에 대한 정의 변경이 가능하다. 뷰를 삭제하는 쿼리문은 다음과 같다.

```
DROP VIEW <View Name>;
```

4. 뷰 내용 변경

뷰를 통해 접근 가능한 데이터에 대한 변경이 가능하다. 하지만 모든 경우에 데이터의 변경이 가능한 것이 아니라, 일부 제약이 존재한다. 이러한 제약은 뷰가 논리적으로 생성된 가상 테이블이기 때문에 물리적 상황에 의존적임을 의미한다. 예를 들어 PK(기본 키)에 해당하는 칼럼이 뷰에 정의되어 있지 않다면 INSERT는 당연히 불가능하다.

수행 내용 / 인덱스와 뷰 활용하기

재료·자료

- 테이블 정의서
- 인덱스 설계서
- SQL 트레이스
- SQL 언어 문법 학습 자료
- 뷰 명세서
- E-R 모델링 다이어그램

기기(장비 · 공구)

- DBMS 프로그램

안전 · 유의 사항

- 제품 약어 M*-SQL과 O*는 대표적인 비상용 제품과 상용 제품을 의미하니, 구별해서 이해한다.

수행 순서

① 인덱스를 생성한다.

1. 인덱스 대상 테이블을 확인한다.

다음과 같은 주문 테이블이 있다. ‘주문번호’ 칼럼은 PK이므로 자동으로 인덱스가 생성된다. 하지만 PK 이외의 칼럼에는 인덱스가 생성되지 않은 상태이다. 주문 테이블에 인덱스를 생성해 보자.

주문테이블



주문번호	고객번호	주문일	주문가격	배송 도시	배송완료일	결제금액	할인금액
A0100	24680	20170704	55,000	서울	20170707	45,000	10,000
X0300	13579	20170801	70,000	부산	Null	50,000	20,000

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.47.
[그림 2-8] 인덱스 수행에 사용할 테이블 모습

2. 인덱스를 설계한다.

인덱스 생성 전에 인덱스에 대한 설계가 필요하다. 즉, 인덱스 대상을 결정하는 과정이 필요하다.

(1) 인덱스 설계 과정을 확인한다.

일반적인 인덱스 설계 과정은 다음과 같다.

〈표 2-8〉 인덱스 설계 과정

단계	인덱스 설계 과정	내용
1	접근 경로 수집	<ul style="list-style-type: none">- 접근 경로는 테이블에서 데이터를 찾는 방법을 의미- 수집 방법에는 테이블 스캔과 인덱스 스캔이 있음- 접근 경로 수집 의미는 SQL이 최적화되었을 때 인덱스 스캔을 해야 하는 검색 조건들을 수집하는 것
2	통한 후보 칼럼 선정	<ul style="list-style-type: none">- 수집된 접근 경로에 대해 분포도 조사- 분포도 = 데이터별 평균 로 수 / 테이블의 총 로 수- 일반적으로 분포도가 10~15% 정도이면 인덱스 칼럼 후보로 사용
3	접근 경로 결정	<ul style="list-style-type: none">- 인덱스 후보 목록을 이용하여 접근 유형에 따라 어떤 인덱스 후보를 사용할 것인가를 결정- 만약 누락된 접근 경로가 있다면 분포도 조사를 실시하고 인덱스 후보 목록에 추가 작업을 반복
4	칼럼 조합 및 순서 결정	<ul style="list-style-type: none">- 단일 칼럼의 분포도가 양호하면 단일 칼럼 인덱스로 확정- 하지만 하나 이상의 칼럼 조합이 필요한 경우는 추가 고려하여 인덱스 칼럼 순서를 결정
5	적용 시험	<ul style="list-style-type: none">- 설계된 인덱스를 적용하고 접근 경로별로 인덱스가 사용되는지 시험해야 함- 여러 개의 접근 경로가 존재하는 테이블은 여러 개의 인덱스가 만들어지므로 의도한 실행 계획으로 동작하는지 확인해야 함

출처: 인덱스 설계. DA 가이드(<http://www.dbguide.net>)에서 2017. 06. 24. 검색.

접근 경로라는 것은 ‘검색 조건’을 의미한다. 따라서 접근 경로를 수집한다는 것은 사용하는 질의문의 검색 조건을 살펴보기 위해 수집하는 것이며, 크게 두 가지 방법이 있다.

- 사용하고 있는 애플리케이션 소스 안에 있는 SQL 문장을 수집
- DBMS의 트레이스(Trace) 도구를 사용하여 SQL 문장을 수집

이와 같이 사용하는 SQL 질의문을 수집하고 분석하여 인덱스를 설계한다. 이때 대상 선정과 우선순위 설정에서 다음과 같은 판단 기준을 참고하도록 한다.

(가) 접근 경로 유형

다음과 같은 분류에 속하는 접근 유형을 인덱스 대상으로 고려한다.

〈표 2-9〉 인덱스 접근 경로

접근 경로 유형	내용
반복 수행되는 접근 경로	대표적으로 조인 칼럼을 후보로 선택(주문 1건당 평균 50개의 주문 내역을 가진다면 주문 테이블과 주문 내역 테이블을 이용하여 주문서를 작성하는 SQL은 조인을 위해 평균 50번의 주문 내역 테이블을 반복 액세스하기 때문)
분포도가 양호한 칼럼	주문 번호, 청구 번호, 주민 번호 등은 단일 칼럼 <u>인덱스로도 충분한 수행 속도를 보장받을 수 있는 후보임</u>
조회 조건에 사용되는 칼럼	성명, 상품명, 고객명 등 명칭이나 주문 일자, 판매일, 입고일 등 일자와 같은 칼럼은 조회 조건으로 많이 이용되는 칼럼
자주 결합되어 사용되는 칼럼	판매일 + 판매 부서, 급여일 + 급여 부서와 같이 조합에 의해 사용되는 칼럼
데이터 정렬 순서와 그룹핑 칼럼	조건뿐만 아니라 순방향, 역방향 등의 정렬 순서 고려(인덱스는 구성 칼럼 값들이 정렬되어 있어 인덱스를 이용하면 별도의 ORDER BY, 정렬 작업이 불필요함) 동일한 원리로 그룹핑 단위(GROUP BY)로 사용된 칼럼도 조사
일련번호를 부여한 칼럼	이력을 관리하기 위해서 일련번호를 부여한 칼럼에 대해서도 조사 (마지막 일련번호를 찾는 경우가 빈번히 발생하므로 효과적인 액세스를 위해 필요)
통계 자료 추출 조건	통계 자료는 결과를 추출하기 위해서 넓은 범위의 데이터가 필요. (다양한 추출 조건을 사전에 확보하여 인덱스 생성에 반영)
조회 조건이나 조인 조건 연산자	위에 제시되는 유형의 칼럼과 함께 적용된 =, between, like 등 의 비교 연산자를 병행 조사하여 인덱스 결합 순서를 결정할 때 중요한 정보로 사용하도록 함

출처: 인덱스 설계. DA 가이드(<http://www.dbguide.net>)에서 2017. 06. 24. 검색.

(나) 칼럼 조합 및 순서 결정

복수 개의 칼럼(결합 칼럼)에 대해 인덱스를 사용할 때 순서가 중요하다. 인덱스 칼럼의 순서를 결정할 때 선두에 두어야 할 칼럼을 선택하는 판단 기준은 다음과 같다.

〈표 2-10〉 인덱스 선두 칼럼 결정 요건

선두 칼럼 요건	내용
항상 사용되는 칼럼	칼럼 A, B가 인덱스로 사용될 때 칼럼 A에는 항상 값이 있어야 함(인덱스로 사용될 때 선행되는 칼럼값이 Null인 경우 인덱스를 이용하지 못함)
등자(=) 조건으로 사용되는 칼럼	부등호나 범위 연산(<, >, <=, >=, BETWEEN, LIKE)보다는 등호(=) 연산을 사용하는 칼럼을 선두에 배치하는 것이 좋음.
분포도가 좋은 칼럼	분포도가 좋다는 의미는 데이터값의 중복이 적은 것을 의미함 즉, 값의 중복이 적은 칼럼을 선두로 사용하는 것이 좋음
ORDER BY, GROUP BY 순서	ORDER BY나 GROUP BY 절에 사용되는 칼럼 순으로 인덱스를 생성하는 것이 좋음

출처: 인덱스 설계. DA 가이드(<http://www.dbguide.net>)에서 2017. 06. 24. 검색.

(2) 인덱스 설계 단계를 확인한다.

테이블 구조만 보고 인덱스 대상을 선정할 수는 없다. 해당 테이블을 어떻게 사용하는지 질의문, 특히 검색 조건에 어떠한 칼럼이 사용되는지를 보고 결정해야 한다. 따라서 직접적인 실습 내용은 생략하며, 교수자와 학습자 사이에 해야 할 일을 다음과 같이 제시한다.

(가) 인덱스 설계해 보기

앞서 수행 순서 ①에서 정의한 ‘주문 테이블’에 대해 인덱스를 설계한다.

(나) 인덱스 설계 필요성

인덱스 설계 과정의 필요성에 대해 논의한다.

3. 인덱스를 생성한다.

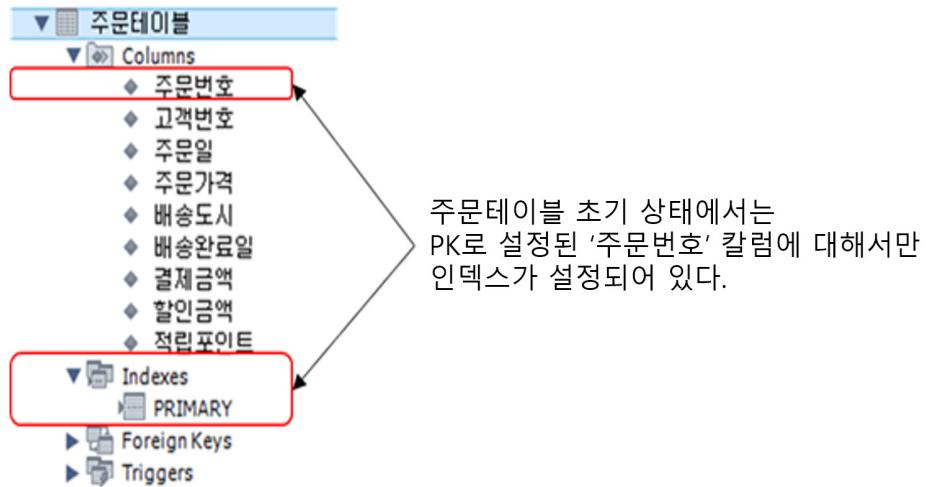
인덱스 설계 과정을 통해 ‘주문 테이블’의 다음과 같은 칼럼에 대해 인덱스가 필요하다고 결정하였다고 하자(수행 순서 ① 참조). 각 경우에 적합한 인덱스를 생성하자.

(1) ‘배송도시’ 칼럼에 대해 인덱스 생성하기

(2) ‘주문일 + 적립 포인트’ 결합 칼럼에 대해 인덱스 생성하기

(3) ‘고객번호’ 칼럼에 대해 UNIQUE 인덱스 생성하기

참고로 인덱스를 생성하기 이전 주문 테이블 상태는 다음과 같다.



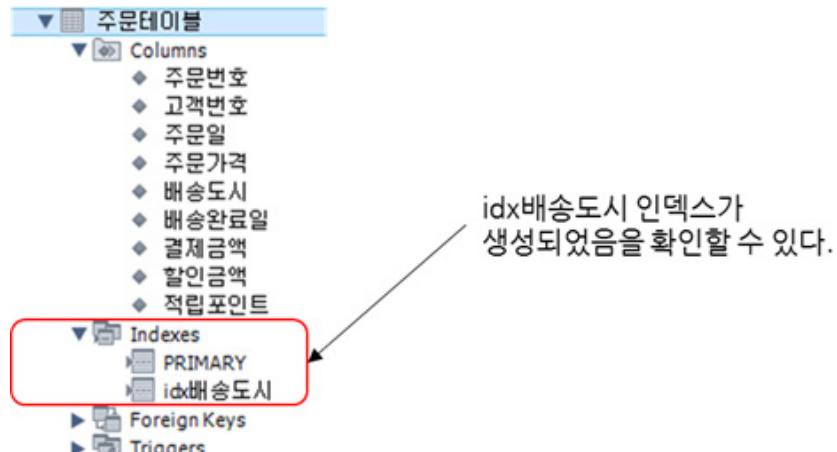
출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.51.
 [그림 2-9] 주문 테이블 초기 상태

(1) '배송도시' 칼럼에 대해 인덱스를 생성한다.

다음과 같은 SQL문을 통해 인덱스를 생성할 수 있다.

```
CREATE INDEX idx배송도시 on 주문 테이블 (배송도시);
```

실행 결과 주문 테이블의 상태에는 다음과 같은 변화가 생겼음을 확인할 수 있다.



출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.51.
 [그림 2-10] 인덱스 생성 후 주문 테이블 모습

M*-SQL에서 인덱스 내용은 다음과 같은 SQL문을 통해 알 수 있다.

```
SHOW INDEX FROM 주문 테이블;
```

인덱스 내용은 다음 그림과 같다.

	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
주문테이블	0	PRIMARY	1		주문번호	A	0	NULL	NULL		BTREE
주문테이블	1	idx배송도시	1		배송도시	A	0	NULL	NULL	YES	BTREE

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.52.

[그림 2-11] 인덱스 정보

(2) '주문일 + 적립 포인트' 칼럼에 대해 인덱스를 생성한다.

다음과 같은 SQL문을 통해 인덱스를 생성할 수 있다.

```
CREATE INDEX idx주문일 포인트 on 주문 테이블 (주문일, 적립 포인트);
```

실행 결과 인덱스의 상태가 다음과 같이 변한 것을 확인할 수 있다.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
주문테이블	0	PRIMARY	1	주문번호	A	0	NULL	NULL		BTREE
주문테이블	1	idx배송도시	1	배송도시	A	0	NULL	NULL	YES	BTREE
주문테이블	1	idx주문일포인트	1	주문일	A	0	NULL	NULL		BTREE
주문테이블	1	idx주문일포인트	2	적립포인트	A	0	NULL	NULL		BTREE

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.52.

[그림 2-12] 주문일, 적립 포인트 추가 인덱스 정보

(3) '고객번호' 칼럼에 대해 UNIQUE 인덱스를 생성한다.

다음과 같은 SQL문을 통해 인덱스를 생성할 수 있다.

```
CREATE UNIQUE INDEX idx고객번호 on 주문 테이블 (고객번호);
```

실행 결과 인덱스의 상태가 다음과 같이 변한 것을 확인할 수 있다.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
주문테이블	0	PRIMARY	1	주문번호	A	0	NULL	NULL		BTREE
주문테이블	0	idx고객번호	1	고객번호	A	0	NULL	NULL		BTREE
주문테이블	1	idx배송도시	1	배송도시	A	0	NULL	NULL	YES	BTREE
주문테이블	1	idx주문일포인트	1	주문일	A	0	NULL	NULL		BTREE
주문테이블	1	idx주문일포인트	2	적립포인트	A	0	NULL	NULL		BTREE

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.52.

[그림 2-13] UNIQUE 인덱스 생성 정보

(4) 기타 인덱스 생성 방법을 조사한다.

이제까지는 인덱스를 생성하기 위하여 CREATE INDEX문을 사용하였다. 인덱스를 생성하는 기타 다른 방법에 대해 조사한다.

4. 인덱스를 삭제한다.

인덱스 생성 과정을 통해 만든 인덱스를 다음 명령을 통해 삭제해 보자.

(1) ‘배송도시’ 인덱스를 삭제한다.

```
ALTER TABLE 주문 테이블 DROP INDEX idx배송도시;
```

(2) ‘주문일 + 적립 포인트’ 인덱스를 삭제한다.

```
ALTER TABLE 주문 테이블 DROP INDEX idx주문일 포인트;
```

(3) ‘고객번호’ 인덱스를 삭제한다.

```
ALTER TABLE 주문 테이블 DROP INDEX idx고객번호;
```

② 뷰 생성

1. 수행에 사용할 뷰를 확인한다.

인덱스 생성 과정에서 사용한 주문 테이블이 있다. 이와 같은 주문 정보를 기반으로 다양한 업무에 활용 가능한 데이터베이스를 구축하고자 한다.

주문테이블

PK 컬럼

주문번호	고객번호	주문일	주문가격	배송 도시	배송완료일	결제금액	할인금액
A0100	24680	20170704	55,000	서울	20170707	45,000	10,000
X0300	13579	20170801	70,000	부산	Null	50,000	20,000

출처: 교육부(2017). SQL 활용(LM2001020413). 한국직업능력개발원. p.53.

[그림 2-14] 뷰 수행에 사용할 데이터

2. 뷰를 생성한다.

(1) 특정 칼럼으로 구성된 뷰를 생성한다.

결산 기능이 필요한 응용 프로그램은 주문 테이블에 정의된 내용 가운데 주문가격 정보만 필요로 한다. 즉, 주문 테이블에 있는 수많은 결제 정보 가운데 다음과 같은 정보만을 해당 응용 프로그램에 제공하고자 할 경우, 뷰를 생성한다.

다음 쿼리문을 이용하여 뷰를 생성한다.

```
CREATE VIEW 주문가격_VIEW AS  
SELECT 주문번호, 고객번호, 주문가격 FROM 주문;
```

위의 SQL문으로 생성된 뷰는 아래 [그림 2-15]와 같다.

주문가격_VIEW

주문번호(PK)
고객번호
주문가격

출처: 집필진 제작(2021)
[그림 2-15] 주문 테이블로
만든 뷰

위와 같이 생성된 뷰를 통하여 주문된 정보를 조회할 수 있다. ‘주문_VIEW’로 조회되는 전체 데이터는 아래 [그림 2-16]과 같다.

```
+-----+-----+-----+
| 주문번호 | 고객번호 | 주문가격 |
+-----+-----+-----+
| 01001   | C0001   | 10000.00 |
| 01002   | C0001   | 4500.00  |
| 01003   | C0004   | 100000.00 |
| 01004   | C0003   | 55000.00  |
| 01005   | C0002   | 85000.00  |
| 01006   | C0002   | 23000.00  |
| 01007   | C0004   | 5000.00   |
| 01008   | C0001   | 8300.00   |
| 01009   | C0002   | 45000.00  |
| 01010   | C0003   | 9000.00   |
+-----+-----+-----+
10 rows in set (0.001 sec)
```

출처: 집필진 제작(2021)
[그림 2-16] ‘주문_VIEW’의 데이터

위와 같이 생성된 뷰에 대해 응용 프로그램은 다음과 같은 쿼리문을 사용할 수 있다. 고객번호에 임의로 ‘C0001’을 입력한 후 결과를 확인한다.

```
SELECT SUM(주문가격) AS 총_주문금액
FROM 주문가격_VIEW
WHERE 고객번호 = 'C0001';
```

여기서 ‘고객번호’는 사용자가 입력한 정보를 의미한다. 그리고 위와 같은 쿼리문 결과로 반환되는 결과로 해당 고객의 총 주문금액을 확인할 수 있다.

```

+-----+
| 총_주문금액 |
+-----+
| 22800.00 |
+-----+
1 row in set (0.001 sec)

```

출처: 집필진 제작(2021)
[그림 2-17] 뷰를 활용한 명령문 결과

이번에는 고객 테이블과 주문 테이블로 조인된 뷰를 만들어보도록 한다. 주문한 고객번호별로 주문한 금액의 최대값을 구하는 뷰이다.

```

CREATE VIEW 최대가격_VIEW AS
SELECT 주문.고객번호, MAX(주문가격) AS 최대가격
FROM 주문, 고객
WHERE 주문.고객번호 = 고객.고객번호
GROUP BY 주문.고객번호;

```

위와 같은 쿼리문 결과는 주문한 고객 4명에 대한 4건이 조회되며, 실제 결과는 아래 [그림 2-18]과 같다.

```

+-----+-----+
| 고객 번호 | 최대가격 |
+-----+-----+
| C0001    | 10000.00 |
| C0002    | 85000.00 |
| C0003    | 55000.00 |
| C0004    | 100000.00 |
+-----+-----+
4 rows in set (0.003 sec)

```

출처: 집필진 제작(2021)
[그림 2-18] 주문 고객 테이블로 만든 뷰

(2) 비교 연산자로 이루어진 뷰를 생성한다.

뷰의 원천 정보를 다양한 형태로 조회할 수 있다. 이러한 조회 형태 가운데 비교 연산자를 가지는 뷰를 생성해 보자. 뷰의 이름을 최근_주문가격_VIEW라고 가정한다.

```

CREATE VIEW 최근_주문가격_VIEW AS
SELECT 주문번호, 고객번호, 주문가격
FROM 주문
WHERE 주문일 >= '20211220';

```

위와 같이 생성된 뷰를 통하여 2021년 12월 20일 이후에 수행된 최근 주문가격 정보를 조회할 수 있다. ‘최근_주문가격_VIEW’로 조회되는 전체 데이터는 다음 [그림 2-19]와 같다.

```

+-----+-----+
| 주문 번호 | 고객 번호 | 주문 가격 |
+-----+-----+
| 01007 | C0004 | 5000.00 |
| 01008 | C0001 | 8300.00 |
| 01009 | C0002 | 45000.00 |
| 01010 | C0003 | 9000.00 |
+-----+-----+
4 rows in set (0.000 sec)

```

출처: 집필진 제작(2021)
[그림 2-19] ‘최근_주문가격_VIEW’의 데이터

(3) 계산 칼럼을 포함하는 뷰를 생성한다.

주문 테이블을 기준으로 고객별로 몇 번의 주문이 존재하는지, 최대 금액과 최소 금액은 얼마인지를 보여 주는 뷰를 생성해 보자. 뷰의 이름은 주문_통계_VIEW라고 가정한다.

```

CREATE VIEW 주문_통계_VIEW AS
SELECT 고객번호, COUNT(주문가격) 주문횟수,
       MAX(주문가격) 최대금액,
       MIN(주문가격) 최소금액
  FROM 주문
 GROUP BY 고객번호;

```

위와 같이 생성된 뷰를 통해서 고객별로 필요한 최대/최소/개수 정보를 뷰의 추 가적인 가공 없이 바로 확인할 수 있다. ‘주문_통계_VIEW’로 조회되는 전체 데이터는 아래 [그림 2-20]과 같다.

```

+-----+-----+-----+-----+
| 고객 번호 | 주문 횟수 | 최대 금액 | 최소 금액 |
+-----+-----+-----+
| C0001 | 3 | 10000.00 | 4500.00 |
| C0002 | 3 | 85000.00 | 23000.00 |
| C0003 | 2 | 55000.00 | 9000.00 |
| C0004 | 2 | 100000.00 | 5000.00 |
+-----+-----+-----+
4 rows in set (0.000 sec)

```

출처: 집필진 제작(2021)
[그림 2-20] ‘주문_통계_VIEW’의 데이터

3. 뷰 내용을 변경한다.

사용자에게 ‘뷰는 테이블’이라고 하였다. 따라서 뷰의 내용을 변경하기 위해서는 UPDATE문을 이용한다. 또 ‘뷰는 논리 테이블’이라고 하였다. 이와 같은 ‘논리’라는 제약에 의해 뷰의 내용에 대한 변경이 불가능한 경우가 발생한다.

(1) INSERT가 불가능한 경우를 조사한다.

앞의 주문 테이블에 대해 다음과 같은 뷰를 생성하였다고 하자.

```
CREATE VIEW 주문_VIEW AS  
SELECT 고객번호, 주문가격 from 주문;
```

즉, 생성된 뷰에는 주문 테이블의 기본 키(PK)에 해당하는 ‘주문번호’가 포함되지 않았다. 이 경우 다음과 같은 레코드를 추가해 보자.

```
INSERT INTO 주문_VIEW VALUES ('C0006', 100);
```

이와 같은 쿼리는 “ERROR 1423 (HY000): Field of view ‘ncs.주문_view’ underlying table doesn't have a default value” 오류가 발생한다. 그 이유는 INSERT 대상이 주문_VIEW이지만 이는 논리 이름이고 실제 대상은 주문 테이블이기 때문이다. 즉, 주문 테이블에 이와 같은 VALUES를 가지는 INSERT 문은 PK가 누락되었기 때문에 불가능하다.

INSERT가 성공하기 위해서는 반드시 PK가 포함되어야 하며, Not Null이나 제약 조건이 정의된 칼럼에도 반드시 대응되는 입력값이 있어야 한다.

(2) UPDATE가 불가능한 경우를 조사한다.

INSERT가 불가능한 경우와 유사한 개념으로 무엇을 어떻게 갱신해야 할지 모르는 칼럼이 하나라도 존재하면 UPDATE도 불가능하다. 보다 구체적인 불가능한 상황은 다음과 같다.

〈표 2-11〉 뷰 업데이트가 불가능한 상황

UPDATE가 불가능한 상황	내용
뷰 칼럼이 산술식, 집계 함수, 상수로부터 유도된 경우	sum, count, avg 등 포함된 뷰
뷰 정의에서 Group By 절이 포함된 경우	물리적으로 존재하는 레코드를 특정할 수 없음
다수 테이블로 뷰가 정의된 경우	대부분 불가능, 뷰 생성 조건이 엄격하다면 UPDATE가 가능할 수 있음

뷰에 대한 UPDATE가 가능한 경우는 갱신할 레코드를 뷰의 근원이 되는 테이블에서 식별 가능한 경우이다. 그러나 GROUP BY나 집계 함수 등이 사용된 뷰에서는 DELETE문과 UPDATE문이 모두 수행되지 않는다. 앞서 생성한 최대가격_VIEW에 대해 다음과 같은 UPDATE문을 수행해 보자.

```

UPDATE 최대가격_VIEW
    SET 고객번호 = 'C1000'
  WHERE 고객번호 = 'C0004';

```

위와 같은 경우 UPDATE문에서는 ERROR 1288 (HY000): The target table **최대가격_VIEW** of the UPDATE is not updatable 오류가 발생한다.

(3) 뷰 내용 변경 가능 및 불가능 조건을 확인한다.

어느 경우에 뷰의 내용 변경이 가능한 것인지, 어느 경우에 뷰의 내용 변경이 절대적으로 불가능한 것인지에 대해 알아보도록 한다.

〈표 2-12〉 뷰 변경이 가능불가능한 경우

경우	변경 여부
뷰가 하나의 테이블에서 정의된 경우	가능
뷰 생성에 사용된 테이블의 PK를 포함하는 경우	가능
뷰 정의에서 집계 함수로 정의된 칼럼이 있는 경우	불가능
뷰 정의에서 DISTINCT가 포함된 경우	불가능
뷰 정의에서 GROUP BY 또는 HAVING이 포함된 경우	불가능
뷰 정의에서 서브쿼리가 포함된 경우	불가능
뷰 정의에 상수, 문자열 등이 포함된 경우	불가능

뷰를 구성하는 모든 테이블의 PK를 포함하고, 산술 연산이나 DISTINCT, GROUP BY, HAVING 및 서브쿼리를 포함하지 않는 뷰에 대해서는 내용 변경이나 삭제가 가능하다.

4. 뷰를 삭제한다.

기준에 생성한 뷰를 삭제해 보자.

```

DROP VIEW 주문가격_VIEW;
DROP VIEW 최근_주문가격_VIEW;
DROP VIEW 주문_통계_VIEW;
DROP VIEW 주문_VIEW;
DROP VIEW 최대가격_VIEW;

```

삭제 이후에는 해당 뷰에 대한 select와 같은 데이터 조회가 불가능하다.

수행 tip

- 인덱스는 사용자가 직접 사용하지 않으므로 간접적으로 인덱스의 효용을 느낄 수밖에 없다. 본 수행 내용 과정을 통해 생성한 인덱스를 통해 검색 속도가 빨라졌는지는 상황에 따라 다르다. 그 이유는 축적된 데이터와 이를 사용하는 검색 조건에 인덱스는 의존적이기 때문이다. 인덱스 학습에서 중요한 것은 인덱스 대상을 결정하는 것이다. 이를 위해 ‘인덱스 설계’에 대한 필요와 이 해가 가장 중요하며, 또한 인덱스를 둘러싼 관련 주제들 역시 중요하다.
- 스무고개 놀이를 통해 어떤 숫자를 찾아가는 과정을 생각해 보자. 가장 효과적인 방법은 중간값을 선택하는 것이다. 컴퓨터에서 이와 같은 방식의 해결 기법을 ‘이진 검색’이라고 한다. 데이터베이스에서 정보를 찾아가는 방법 역시 이진 검색을 응용한 것이다. B트리라는 자료 구조를 통해 분기 횟수를 최소화하여 찾아간다. B트리에는 분기를 할 수 있는 정보가 있으며, 그 정보가 바로 인덱스이다.
- 앞서 “[② 뷰 생성](#)” 과정에서 사용한 다음과 같은 쿼리문(SELECT SUM(주문가격) FROM 주문가격_VIEW WHERE 고객번호 = ‘XXXX’)의 사용은 SW 보안취약점 관점에서 문제가 많다. 하지만 본 학습에서는 보안보다는 사용의 개념을 학습하기 위해 사용된 쿼리문으로 단순 이해의 용도로 사용되었다.

학습 2 교수·학습 방법

교수 방법

- 인덱스 개념에 대해 설명한다.
- 인덱스 생성과 삭제 방법을 제시한다.
- 인덱스 설계 과정에 직접 시연한다.
- 뷰 개념에 대해 안내한다.
- 뷰 생성과 삭제 과정을 보여준다.
- 테이블 외의 데이터 사전의 개념과 사용법에 대해 설명한다.

학습 방법

- 인덱스 개념을 작성한다.
- 인덱스 생성과 삭제 방법을 서술한다.
- 인덱스 설계 과정에 대해 수행한다.
- 뷰 개념에 대해 작성한다.
- 뷰 생성과 삭제 방법을 시연한다.
- 테이블 외의 데이터 사전의 개념과 사용법에 대해 기술한다.

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
테이블 외의 데이터 사전 조회	- 생성된 테이블의 목록, 테이블의 인덱스와 뷰를 파악하기 위해 데이터 사전을 조회하는 명령문을 작성할 수 있다.			
인덱스와 뷰 작성	- 테이블 조회 시간을 단축하기 위해 사용하는 인덱스의 개념을 이해하고 인덱스를 생성하는 DDL(Data Definition Language) 명령문을 작성할 수 있다. - 먼저 생성된 테이블들을 이용하여 새로운 테이블과 뷰를 생성하는 DDL(Data Definition Language) 명령문을 작성할 수 있다.			

평가 방법

- 서술형 시험

학습 내용	평가 항목	성취수준		
		상	중	하
테이블 외의 데이터 사전 조회	인덱스와 뷰에 대한 목록을 도출할 수 있는지 평가			
인덱스와 뷰 작성	생성된 테이블 목록을 검색하고 의미를 파악할 수 있는지 평가			
	검색 속도를 빠르게 하는 인덱스 원리			
	인덱스 설계 과정 및 단계별 유의 사항			
	뷰 생성하기			
	뷰를 이용한 데이터 삽입 및 삭제			

• 사례 연구

학습 내용	평가 항목	성취수준		
		상	중	하
테이블 외의 데이터 사전 조회	DBMS별 인덱스와 뷰 관련 데이터 사전의 구성 요소 조사			
	DBMS별 인덱스와 뷰 관련 데이터 사전의 요소별 내용 조사			
인덱스와 뷰 작성	인덱스 설계를 위한 접근 경로 수집			
	복수 칼럼 순서 변경으로 인덱스 성능 차이 구분			
	보안이 고려된 뷰, 편의성을 위한 뷰 설계			
	뷰를 이용한 데이터 삽입 및 변경이 가능하도록 설계			

피드백

1. 서술형 시험

- 고급 SQL 명령문의 사용 능력을 평가한 후, 미흡한 부분에 대해 설명해 준다. 이때 미흡한 부분은 이해 부족에서 발생하는 것이 대부분이기에, 고급 SQL 작성 주제, 영역별 SQL 사용 방법에 대한 지도 이외에, 개념적 지식에 대한 설명을 제공한다.
- 본 학습 모듈에서는 추가적인 배경 지식을 설명하는 의도로 SQL 사용 방법 이외에 SQL이 무엇인지 생각해 가는 과정을 제시하였다. 이러한 의도를 참고하여 배경 지식에 대한 설명 혹은 학습이 이루어지도록 한다.

2. 사례 연구

- 다양한 문제 상황에 대한 연습이 가능하도록 상황을 제시해 준다.
- 관련된 연관 주제에 대해 호기심을 가질 수 있도록 연관 관계를 제시해 준다.

참고자료



- 교육부(2016). 데이터베이스 구현(LM2001020405_16v3). 한국직업능력개발원.
- 교육부(2017). SQL 활용(LM2001020410_16v3). 한국직업능력개발원.
- 데이터전문가 지식포털. DA가이드/SQL가이드. <http://www.dbguide.net>에서 2017. 7. 23. 검색.
- 한국정보통신기술협회. 정보통신용어사전. <http://terms.tta.or.kr>에서 2017. 9. 7. 검색.
- Git hub. A sample MySQL database. https://github.com/datacharmer/test_db에서 2017. 7. 23. 검색.
- My SQL. Employees Sample Database.
<https://dev.mysql.com/doc/employee/en/employees-installation.html>에서 2017. 7. 23. 검색.
- My SQL. My SQL Document. <https://dev.mysql.com>에서 2017. 7. 23. 검색.
- My SQL Tutorial. My SQL Data Manipulation and Definition.
<http://www.mysqltutorial.org>에서 2017. 7. 23. 검색.
- TTA. <http://terms.tta.or.kr>에서 2017. 8. 7. 검색.
- W3 Resource. SQL, Oracle and MySQL Exercises. <http://www.w3resource.com>에서 2017. 7. 23. 검색.
- W3 Schools. Learn SQL. <http://www.w3resource.com>에서 2017. 7. 23. 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	SQL활용(LM2001020413_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	2021년 12월 31일		
학습모듈명	SQL활용(LM2001020413_19v4)		
개발기관	(사)한국정보통신기술사협회(개발책임자: 온기현), 한국직업능력연구원		
집필진	황극인((주)코스콤)* 양지언(삼성SDS) 엄태명(현대IT&E)	검토진	김희완(삼육대학교) 심형광(한국아이디정보(주))
			*표시는 대표집필자임

SQL활용(LM2001020413_19v4)

저작권자	교육부
연구기관	한국직업능력연구원
발행일	2021. 12. 31.
ISBN	979-11-339-9449-6

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



www.ncs.go.kr