대분류 / 20	중분류 / 01	소분류 / 02	세분류 / 02	학습모듈 / 05
정보통신	정보기술	정보기술개발	응용SW엔지니어링	

05
데이터 입출력 구현

LM2001020205_14v2

KRIVET 한국직업능력개발원
한국소프트웨어기술진흥협회 KOSTA Korea Software Technology Association
교육부

데이터 입출력 구현 학습모듈의 개요

학습모듈의 목표

응용소프트웨어가 다루어야 하는 데이터 및 이들 간의 연관성, 제약조건을 식별하여 논리적으로 조직화하고, SW아키텍처에 기술된 데이터저장소에 조직화된 단위의 데이터가 저장될 최적화된 물리적 공간을 구성하고 데이터 조작언어를 이용하여 데이터 입출력을 구현할 수 있다.

선수학습

소프트웨어 공학 이해, 데이터모델링 개념 이해, 데이터모델링 표기법 및 도식 기술 등

학습모듈의 내용체계

학습	학습내용	NCS 능력단위요소		
		코드번호	요소명칭	수준
1. 논리 데이터저장소 확인하기	1-1 논리 데이터모델 검증	2001020205_14v2.1	논리 데이터저장소 확인하기	4
2. 물리 데이터저장소 설계하기	2-1 물리 데이터모델 설계 2-2 물리 데이터모델 구성	2001020205_14v2.2	물리 데이터저장소 설계하기	5
3. 데이터 조작 프로시저 작성하기	3-1 데이터 조작 프로시저 작성 3-2 데이터 조작 프로시저 테스트	2001020205_14v2.3	데이터 조작 프로시저 작성하기	3
4. 데이터 조작 프로시저 최적화하기	4-1 데이터 조작 프로시저 성능개선	2001020205_14v2.4	데이터 조작 프로시저 최적화하기	4

핵심 용어

논리 데이터모델, 물리 데이터모델, 데이터 조작 프로시저, 데이터 조작 프로시저 최적화

학습 1	**논리 데이터저장소 확인하기 (LM2001020205_14v2.1)**
학습 2	물리 데이터저장소 설계하기(LM2001020205_14v2.2)
학습 3	데이터 조작 프로시저 작성하기(LM2001020205_14v2.3)
학습 4	데이터 조작 프로시저 최적화하기 (LM2001020205_14v2.4)

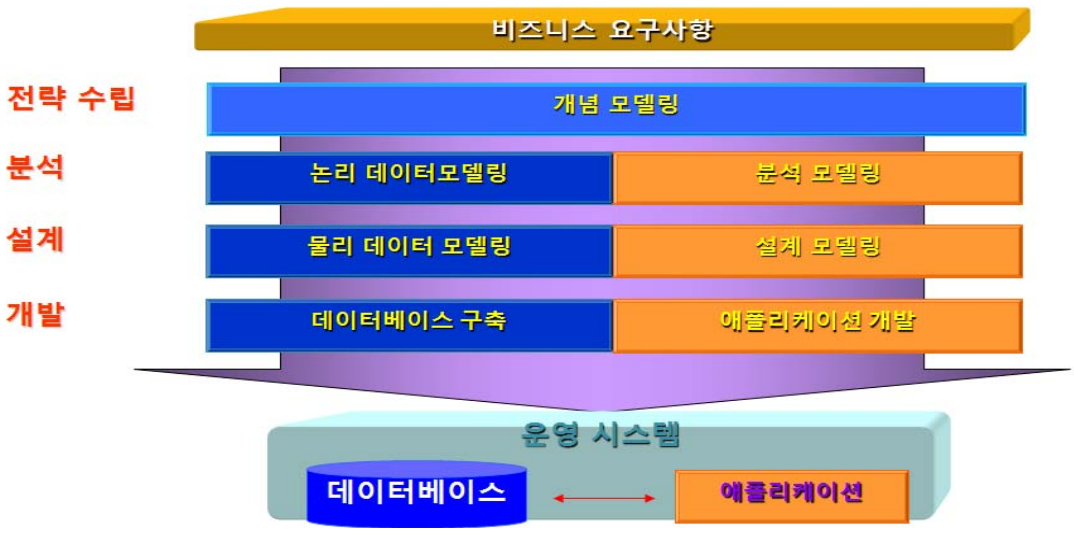
1-1. 논리 데이터모델 검증

학습 목표

- 업무 분석가, 데이터베이스 엔지니어가 작성한 논리 데이터저장소 설계 내역에서 정의된 데이터의 유형을 확인하고 식별할 수 있다.
- 논리 데이터저장소 설계 내역에서 데이터의 논리적 단위와 데이터 간의 관계를 확인할 수 있다.
- 논리 데이터저장소 설계 내역에서 데이터 또는 데이터간의 제약조건과 이들 간의 관계를 식별할 수 있다.

필요 지식 /

1 일반적인 시스템 개발 절차



[그림 1-1] 시스템 개발 절차

일반적으로 시스템 개발은 데이터 관점과 프로세스 관점의 두가지로 진행되는데, 개념 모델링을 통해 개발 범위를 파악하고, 업무 중심의 분석(논리 데이터 모델링, 분석 모델링)단

계를 거쳐 개발하고자 하는 환경을 고려한 설계(물리 데이터 모델링, 설계 모델링)단계로 구체화되어 개발(데이터베이스 구축, 애플리케이션 개발)단계로 진행된다.

2 데이터 모델링 개요

1. 데이터 모델링 정의

기업의 정보 구조를 실체(Entity)와 관계(Relation)를 중심으로 명확하고 체계적으로 표현하여 문서화하는 기법을 말한다.

2. 데이터 모델링 목적

(1) 연관 조직의 정보요구에 대한 정확한 이해를 할 수 있다.

(2) 사용자, 설계자, 개발자 간에 효율적인 의사소통 수단을 제공한다.

(3) 데이터 체계 구축을 통한 고품질 S/W와 유지보수 비용의 감소효과를 기대할 수 있다.

(4) 신규 또는 개선 시스템의 개발 기초를 제공한다.

3. 데이터 모델링 특성

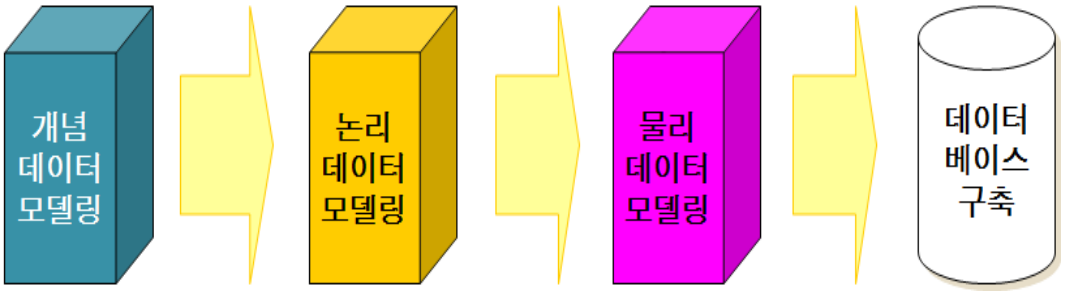
(1) 데이터 중심 분석을 통한 업무 흐름 파악이 용이하다.

(2) 데이터 무결성을 보장할 수 있다

(3) 데이터의 공유를 통한 중복을 제거하고 일관성 있는 정보를 제공받을 수 있다.

3 데이터 모델링 절차

데이터 모델링은 개념 모델링, 논리 모델링, 물리 모델링을 통해 데이터베이스를 구축하는 일련의 절차를 거쳐 진행된다.



[그림 1-2] 데이터 모델링 절차

1. 개념 데이터 모델링

전사의 정보요건을 표현한 상위수준의 모델로서,

(1) 주요 엔터티타입, 기본 속성, 관계, 주요 업무기능 등을 포함한다.

(2) 모든 업무 영역을 포함하고, 주제 영역에 포함되는 중심 엔터티타입 간의 관계를 파악하여 주요 업무 규칙을 정의한다.

(3) 논리 데이터 모델의 기초가 된다.

2. 논리 데이터 모델링

개념모델로부터 업무영역의 업무 데이터 및 규칙을 구체적으로 표현한 모델로서,

(1) 모든 업무용 엔터티타입, 속성, 관계, 프로세스 등을 포함한다.

(2) 모든 업무 데이터를 정규화(Normalization) 하여 모델링한다.

(3) 모든 업무 규칙과 관계를 완전하고 정확하게 표현한다.

(4) 성능 혹은 기타 제약 사항과는 독립적인 모델로서, 특정 DBMS로부터 독립적이라 할 수 있다.

3. 물리 데이터 모델링

설계단계에서 시스템의 설계적 및 정보 요건을 정확하고 완전하게 표현한 모델로서,

(1) 데이터베이스 생성을 위한 물리 구조로 변환한다.

(2) 시스템 설계 요건 반영을 위한 아래와 같은 오브젝트를 추가한다.

(가) 설계용 엔터티 타입

(나) 설계용 속성

(3) 설계와 성능을 고려한 조정을 수행한다.

(가) 적용 DBMS 특성 고려

(나) 엔터티 타입의 분리 또는 통합 검토

(다) 반정규화(Denormalization)

(라) 관계의 해제

(4) 적용 DBMS에 적합한 성능조정을 수행한다

(가) 인덱스 추가 및 조정

(나) 테이블 스페이스 조정

(다) 인덱스 스페이스 조정

4 논리 데이터 모델링 개요

1. 논리 데이터 모델링 정의

(1) 데이터베이스 개발 과정의 첫 단계로 전략수립 및 분석 단계에서 실시한다.

(2) 데이터 구조에 대한 논리적 정의단계로서 정확한 업무 분석을 통한 자료의 흐름을 분석하여 현재 사용 중인 양식, 문서, 장표를 중심으로 자료항목을 추출하여 추출된 엔터티(Entity)와 속성(Attribute)들의 관계(Relation)를 구조적으로 정의하는 단계이다.

(가) 엔터티: 관리할 대상이 되는 실체

(나) 속성: 관리할 정보의 구체적 항목

(다) 관계: 엔터티간의 대응관계

2. 논리 데이터 모델링 특성

(1) 논리적 데이터 모델링 시 요구사항을 충분히 수집하지 않으면 다음 단계의 요구사항 변경에 따른 많은 비용이 발생한다.

(2) 모든 이해당사자들과 의사소통의 보조자료로서 E-R 모델을 활용한다.

(3) 논리적 모델은 H/W나 S/W에 독립적이다.

3. 정규화(Normalization)

(1) 정의

중복성을 최소화하고 정보의 일관성을 보장하기 위한 개념

(2) 목적

(가) 데이터 중복 배제로 데이터 관리 편의성 제고 및 자료 저장 공간의 최소화

(나) 데이터 모형 단순화

(다) 데이터 구조의 안정성 및 무결성 유지

(라) 속성의 배열상태 검증

(마) 엔터티와 속성의 누락 여부 검증 수단

(바) 자료검색과 추출의 효율성을 추구

(3) 특징

(가) 어떠한 관계구조가 바람직한 것인지, 바람직하지 못한 관계를 어떻게 분해하여야 하는지에 관한 구체적인 판단기준을 제공

(나) 정규화된 데이터 모델은 정확성, 일치성, 단순성, 비중복성, 안정성 보장

(4) 유형

(가) 제1정규화

1) 반복되는 속성이나 Group 속성 제거

2) 새로운 실체와 1:N의 관계 추가

3) 모든 속성은 반드시 하나의 값을 가져야 함(반복형태가 있어서는 안됨)

(나) 제2정규화

1) 주식별자에 완전하게 종속되지 않는 속성 제거

2) 불완전 함수적 종속(Non Fully Dependency) 제거

3) 모든 속성은 반드시 UID전부에 종속되어야 함(UID일부에만 종속되어서는 안됨)

(다) 제3정규화

1) 비식별자에 종속되는 속성 제거

2) 주식별자에 이행종속(Transitive Dependency) 되는 속성 제거

3) UID가 아닌 모든 속성 간에는 서로 종속될 수 없음(속성간 종속성 배제)

(라) 제4정규화

1) 실제로 거의 고려되지 않는 정규화

2) 주식별자에 다가종속(Multi-Valued Dependency)되는 속성을 두가지 이상 두지 않음

3) 2차 정규화된 테이블은 다대다 관계를 가질 수 없음

4) 어떠한 관계구조가 바람직한 것인지, 바람직하지 못한 관계를 어떻게 분해하여야 하는지에 관한 구체적인 판단기준을 제공

(4) 정규화 수준에 따른 장단점

정규화 수준이 높을수록,

(가) 장점

1) 유연한 데이터 구축이 가능

2) 데이터의 정확성 높아짐

(나) 단점

1) 물리적 접근이 복잡

2) 길이가 짧은 데이터 생성으로 과도한 조인 발생

4. 모델 작성 기법

(1) 엔터티들은 정렬하여 배열한다.

(2) 업무흐름의 진행 순서와 관련된 엔터티는 진행순서를 고려하여 좌에서 우, 상에서 하로 중심부에 배열한다.

(3) 중심에 배열된 엔터티와 관계를 가진 엔터티를 가까이 배열한다.

(4) 관계는 사선이 아닌 수직, 수평선을 사용한다.

(5) 공간을 활용하여 복잡해 보이지 않도록 배열한다.

(6) 교차선이 생기거나 관계선이 너무 길지 않도록 배열한다.

(7) 관계있는 엔터티끼리 그룹핑한다.

수행 내용 / 논리 데이터모델 검증하기

재료・자료

- ERD 작성표준, 개발 시 활용 데이터 저장소에 대한 매뉴얼 및 개발 가이드, 모델링 검토기준

기기(장비・공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 데이터저장소 설계 프로그램

안전・유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

현행 업무와 시스템에 대한 현황 파악 및 신규 데이터베이스의 분석내용을 파악하여, 데이터에 대한 정확한 요구사항 분석이 이루어졌고, 데이터의 흐름, 데이터 모델링에 대한 기준, 절차 및 방법, 엔터티 및 프로세스 간 연관성, 데이터 접근권한, 통제 등에 대한 분석이 적절하게 수행되었는가를 확인하고 검증하기 위해 다음의 순서로 진행한다.

[1] 구축 시스템의 업무관련 아래 유형의 데이터가 식별되었는지 확인한다

1. 업무처리를 위한 입력데이터, 출력데이터가 식별되었는지 확인한다.
2. 관련 업무와의 연계 데이터가 식별되었는지 확인한다.
3. 신규 데이터 요구사항이 식별되었는지 확인한다.

[2] 현재 운영중인 시스템의 데이터 현황 분석이 적절하게 이루어졌는지 확인한다

1. 시스템별 데이터베이스 구조, 분산, 백업현황이 정확히 파악되었는지 확인한다.
2. 데이터 속성, 공통코드가 정확히 파악되었는지 확인한다.
3. 외부 연계 데이터 속성이 정확히 파악되었는지 확인한다.

[3] 데이터베이스에 대한 사용자 요구사항이 도출되고 분석되었는지 확인한다.

1. 저장 데이터 볼륨, 분산구조, 제약조건에 관한 요구사항이 도출되고 분석되었는지 확인한다.
2. 타 시스템과의 연계 데이터에 대한 요구사항이 도출되고 분석되었는지 확인한다.

3. 데이터베이스 백업 및 복구 정책에 대한 요구사항이 도출되고 분석되었는지 확인한다.
4. 초기 데이터 구축방안에 대한 요구사항이 도출되고 분석되었는지 확인한다.
5. 기존 데이터의 전환방안에 대한 요구사항이 도출되고 분석되었는지 확인한다.
6. 무결성, 데이터 보안성 수준에 대한 요구사항이 도출되고 분석되었는지 확인한다.

4 데이터의 흐름이 명확하게 정의되었는지 확인한다.

1. 프로세스별 입출력 데이터가 명확히 정의되었는지 확인한다.
2. 프로세스 간 공유 및 연계 데이터가 명확히 정의되었는지 확인한다.

5 데이터베이스 설계 기준이 적절히 정의되었는지 확인한다.

1. 데이터베이스 설계 표준 지침의 작성 여부 및 적절한지 확인한다.
2. 데이터 모델 설계 지침의 작성 여부 및 적절한지 확인한다.

6 데이터 모델링이 충분하고 적절하게 수행되었는지 확인한다.

1. 논리 데이터 모델의 개념 완전성 및 정확성에 대해 확인한다.
2. 논리 데이터 모델의 중복 최소화를 통한 무결성이 보장되었는지 확인한다.
3. 엔터티 간 관계(Relationship)의 업무 규칙 부합 여부를 확인한다.
4. 데이터 모델 관련 산출물 작성의 적합성 및 일관성에 대해 확인한다.

7 엔터티 및 프로세스 간의 연관(데이터 생성, 조회, 수정, 삭제 등)관계가 명확한지 확인한다.

8 데이터에 대한 접근권한 및 통제가 명확히 분석되었는지 확인한다.

1. 데이터 접근권한 및 통제가 명확히 분석되었는지 확인한다.
2. 데이터 중요도 및 데이터 암호화 대상 소프트웨어 아키텍처 설계 가이드라인을 확인한다.

수행 tip

- 논리 데이터 모델은 구축하고자 하는 업무 시스템의 사용자 요구사항과 데이터 흐름이 누락되지 않고 반영되었는지와 모델링 표준의 준수도, 그리고 중복회피를 위한 정규화가 수행되었는지가 중요한 검증 포인트이다.

학습 1 교수 · 학습 방법

교수 방법

- 교수자의 주도로 ERD 표기법과 의미에 대한 내용을 파워포인트(PPT) 자료로 제시한 후 설명한다.
- 교수자의 주도로 분석 단계에서 작성된 논리 데이터 모델을 이해하고, 모델의 표준화 준수정도와 정규화 측면에서 검증할 내용을 중심으로 PPT 자료로 제시한 후 설명한다.
- 교수자의 주도로 논리 데이터 모델이 물리 데이터 모델에 어떤 측면에서 필요하고 다른지를 이해하기 위하여 해당 내용을 중심으로 PPT 자료로 제시한 후 설명한다.
- 모델링 도구들에 대한 이해 및 이전 사용 경험 여부에 대해 파악한 후 수업을 진행하고, 필요시 모델링 도구 매뉴얼 및 관련 홈페이지 등을 방문하여 관련 자료를 제시한 후 설명한다.
- 정규화 과정에 대한 학습 시 실제 데이터 구조를 가진 실 사례를 준비하여 각 정규화 과정에서 어떻게 정규화가 진행되는지를 설명함으로써, 실무 능력을 확보할 수 있도록 지도한다.

학습 방법

- 작성된 ERD 표기법과 의미에 대하여 명확히 이해할 수 있도록 학습한다.
- 작성된 논리 데이터 모델을 읽어 업무의 데이터 흐름을 이해할 수 있고, 모델의 활용 용도에 대해 이해한다.
- 논리 데이터 모델과 물리 데이터모델의 작성 필요성 및 차이점을 이해할 수 있도록 학습한다.
- 모델링을 위한 데이터 모델 작성과정과 도구활용법을 이해하고 실습한다.
- 정규화 과정에서 각 정규화가 진행되는 과정을 실제 데이터 구조를 활용함으로써 명확히 이해하고 실습한다.

학습 1 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.

- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준 상	중	하
논리 데이터모델 검증	- 업무 분석가, 데이터베이스 엔지니어가 작성한 논리 데이터저장소 설계 내역에서 정의된 데이터의 유형을 확인하고 식별할 수 있다.			
	- 논리 데이터저장소 설계 내역에서 데이터의 논리적 단위와 데이터 간의 관계를 확인할 수 있다.			
	- 논리 데이터저장소 설계 내역에서 데이터 또는 데이터간의 제약조건과 이들 간의 관계를 식별할 수 있다.			

평가 방법

- 평가자 체크리스트

학습내용	평가항목	성취수준 상	중	하
논리 데이터모델 검증	- 논리 데이터모델의 완전성			
	- 논리 데이터모델의 표준 준수도와 정규화 수준			

- 구두발표

학습내용	평가항목	성취수준 상	중	하
논리 데이터모델 검증	- 논리 데이터모델의 완전성			
	- 논리 데이터모델의 표준 준수도와 정규화 수준			

피 드 백

1. 체크리스트를 통한 관찰
   - 검증과정에서 학습자의 경험과 지식뿐만 아니라 분석, 의견 조율 및 조정 능력을 확인하고 개선해야 할 사항을 알려준다.
   - 체크리스트에 위배되는 항목에 대해 명확히 정리하여 돌려준다.
   - 정규화 과정에 위배되는 항목에 대해 명확히 정리하여 돌려준다.
2. 구두 발표
   - 발표한 내용 중 미비사항이나 보완해야 할 사항을 정리하여 돌려준다.
   - 논리 데이터모델을 가독성 및 완정성 측면에서 보완해야 할 사항을 정리하여 돌려준다.

학습 1	논리 데이터저장소 확인하기(LM2001020205_14v2.1)
학습 2	**물리 데이터저장소 설계하기 (LM2001020205_14v2.2)**
학습 3	데이터 조작 프로시저 작성하기(LM2001020205_14v2.3)
학습 4	데이터 조작 프로시저 최적화하기 (LM2001020205_14v2.4)

2-1. 물리 데이터모델 설계

학습 목표

- 논리 데이터저장소 설계를 바탕으로 응용소프트웨어가 사용하는 데이터저장소의 특성을 반영한 물리 데이터저장소 설계를 수행할 수 있다.
- 논리 데이터저장소 설계를 바탕으로 목표 시스템의 데이터 특성을 반영하여 최적화된 물리 데이터저장소를 설계할 수 있다.

필요 지식 /

물리 데이터 모델링은 논리모델을 적용하고자 하는 기술에 맞도록 상세화해 가는 과정이다. 따라서 앞으로 기술되는 내용은 특정 적용기술이나 DBMS를 전제할 수밖에 없기 때문에 시장 점유율을 고려하여 범용적으로 활용되는 기술과 제품을 선택하여야 한다.
앞으로 제시되는 내용은 시장에서 대부분 활용되고 있는 관계형 데이터베이스(RDBMS)의 오라클 데이터베이스를 기준으로 제시한다.

① 반정규화(Denormalization) 개념

1. 정의

정규화에 충실하여 모델링을 수행하면 종속성, 활용성은 향상되나 수행속도가 증가하는 경우가 발생하여 이를 극복하기 위해 성능에 중점을 두어 정규화하는 방법

2. 특징

(1) 데이터 모델링 규칙에 얽매이지 않고 수행한다.

(2) 시스템이 물리적으로 구현되었을 때 성능향상을 목적으로 한다.

3. 사용 시기

(1) 정규화에 충실하였으나 수행속도에 문제가 있는 경우

(2) 다량의 범위를 자주 처리해야 하는 경우

(3) 특정범위의 데이터만 자주 처리하는 경우

(4) 처리범위를 줄이지 않고는 수행속도를 개선할 수 없는 경우

(5) 요약 자료만 주로 요구되는 경우

(6) 추가된 테이블의 처리를 위한 오버헤드를 고려하여 결정

(7) 인덱스의 조정이나 부분범위처리로 유도하고, 클러스터링을 이용하여 해결할 수 있는지를 철저히 검토 후 결정

2 반정규화(Denormalization) 유형

1. 중복 테이블 추가

(1) 용도

(가) 다량의 범위를 자주 처리하는 경우

(나) 특정 범위의 데이터만 자주 처리되는 경우

(다) 처리범위를 줄이지 않고는 수행속도를 개선할 수 없는 경우

(2) 방법

(가) 집계 테이블의 추가

활용하고자 하는 집계정보를 위한 테이블을 추가하고, 각 원본테이블에 트리거를 등록시켜 생성하여 활용하는데, 이때 트리거의 오버헤드에 유의해야 한다.

(나) 진행 테이블의 추가

이력관리 등의 목적으로 사용되며 활용도가 좋아지도록 기본키를 적절히 설정하여야 한다.

(다) 특정 부분만을 포함하는 테이블 추가

거대한 테이블의 특정 부분만을 사용하는 경우 자주 사용되는 부분으로 새로운 테이블 생성하여 활용한다.

2. 테이블 조합

(1) 용도

대부분 처리가 두 개 이상의 테이블에 대해 항상 같이 일어나는 경우에 활용한다.

(2) 방법

해당 테이블을 통합하여 설계한다.

(3) 고려사항

(가) 데이터 액세스가 보다 간편하지만 Row수가 증가하여 처리량이 증가하는 경우가 발생될 수 있으므로 이를 고려해야 한다.

(나) 입력, 수정, 삭제 규칙이 복잡해질 수 있음에 유의해야 한다.

(다) Not Null, Default, Check 등의 Constraint를 완벽히 설계하기 어려운 점이 있다.

3. 테이블 분할

(1) 용도

(가) 칼럼의 사용빈도의 차이가 많은 경우

(나) 각각의 사용자가 각기 특정한 부분만 지속적으로 사용하는 경우

(다) 상황에 따라 SUPER-TYPE을 모두 내려 SUB-TYPE 별로 분할하거나 SUPER-TYPE만은 따로 테이블을 생성하는 경우

(2) 방법

(가) 수직 분할

칼럼별 사용빈도의 차이가 많은 경우 자주 사용되는 칼럼들과 그렇지 않은 칼럼으로 분류하여 테이블을 분할하는 방법이다.

(나) 수평 분할

특정 범위별 사용 빈도의 차이가 많은 경우 해당 범위 별로 테이블을 분할하는 방법이다.

(3) 고려사항

(가) 특정 칼럼 또는 범위를 사용하지 않는 경우 수행속도에 많은 영향이 있음을 고려해야 한다.

(나) 기본키의 유일성 관리가 어려워진다.

(다) 액세스 빈도나 처리할 데이터양이 적은 경우는 분할이 불필요함을 고려하여야 한다.

(라) 분할된 테이블은 오히려 수행속도를 나쁘게 하기도 함에 유의하여야 한다.

(마) 데이터 프로세싱 관점이 아니라 검색에 중점을 두어 결정하여야 한다.

4. 테이블 제거

(1) 용도

테이블 재정의나 칼럼의 중복화로 더 이상 액세스 되지 않는 테이블 발생할 경우

(2) 방법

해당 테이블을 삭제한다.

(3) 고려사항

(가) 관리 소홀로 인해, 누락시 유지보수 단계에서 많이 발생하는 현상임을 고려해야 한다.

(나) 유지보수 단계에서 초기 설계 시 예상하지 못했던 새로운 요구사항이 증가하게 되면 눈앞의 해결에만 급급하여 테이블의 추가나 변경이 함부로 일어나게 되어 시스템은 일관성과 통합성이 무너지게 된다는 사실을 간과해서는 안된다.

5. 칼럼의 중복화

(1) 용도

(가) 자주 사용되는 칼럼이 다른 테이블에 분산되어 있어 상세한 조건에도 불구하고 액세스 범위를 줄이지 못하는 경우

(나) 대량 데이터에서 Row별 연산 결과를 얻고자 할 때 성능향상을 위한 파생(Derived) 칼럼을 추가할 경우

(다) 기본키의 형태가 적절하지 않거나 너무 많은 칼럼으로 구성된 경우

(라) 정규화 규칙에 얽매이지 않으면서 성능향상을 목적으로 한 반정규화(Denomalization)를 통한 중복 데이터를 허용하는 경우

(2) 방법

필요한 해당 테이블이나 칼럼을 추가한다.

(3) 고려사항

(가) 테이블 중복과 칼럼의 중복을 고려한다.

(나) 데이터 일관성 및 무결성에 유의해야 한다.

(다) SQL Group Function을 이용하여 해결 가능한지 검토한다.

(라) 저장공간의 지나친 낭비를 고려해야 한다.

수행 내용 / 물리 데이터모델링하기

재료・자료

• ERD 작성표준, 개발 시 활용 데이터 저장소에 대한 매뉴얼 및 개발 가이드, 모델링 검토기준

기기(장비・공구)

• 컴퓨터, 인터넷, 형상관리 프로그램, 데이터저장소 설계 프로그램

안전・유의사항

• 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

논리 데이터 모델로부터 물리 데이터 모델로 변환하는 순서는, 단위 엔터티를 테이블로, 속성을 칼럼으로, UID를 기본키(Primary key)로, 관계를 외래키(Foreign ley)로 변환한 후, 칼럼 유형과 길이를 정의하고, 데이터 처리 범위와 빈도수를 분석하여 반정규화를 고려하는 순서로 진행한다.

1 단위 엔터티를 테이블로 변환한다.

1. 논리모델에서 정의된 엔터티는 물리모델에서 테이블로 변환한다.
2. 변환 방법
   (1) 일반적으로 테이블과 엔터티 명칭을 동일하게 하는 것을 권고한다.
   (2) 엔터티는 한글명을 사용하고, 테이블은 소스코드의 가독성을 위해 영문명을 사용한다.
   (3) 메타데이터시스템과 같은 사전에 표준화된 용어가 있을 경우 메타에 등록되어 있는 단어를 사용하여 명명한다.

2 속성을 칼럼으로 변환한다.

1. 논리모델에서 정의된 속성은 물리모델에서 칼럼으로 변환한다.
2. 변환방법
   (1) 칼럼의 명칭은 속성의 명칭과 반드시 일치할 필요는 없으나, 개발자와 사용자 간 의사소통을 위하여 가능한 한 표준화된 약어를 사용하도록 한다.
   (2) SQL 예약어(Reserved word) 사용은 피해야 한다.

(3) SQL 문장의 가독성을 높이기 위해 칼럼명칭은 가능한 한 짧은 것이 좋다.

(4) 특정 테이블에 칼럼을 정의한 후, 한 로우(row)에 해당하는 샘플 데이터를 작성하여 칼럼의 정합성을 검증한다.

(5) 칼럼명으로 복합단어를 사용할 경우 미리 정의된 표준에 의해 명명하여야 한다.

3 UID를 기본키(Primary Key)로 변환한다.

1. 논리모델에서 정의된 UID는 물리모델에서 기본키로 변환한다.

2. 변환방법

(1) 엔터티의 UID에 해당하는 모든 속성에 대해 기본키로 선언하고, Not Null, Unique 등의 제약조건을 추가적으로 정의한다.

(2) 관계에 의한 외래키가 기본키에 포함될 수 있다.

4 관계를 외래키(Foreign Key)로 변환한다.

1. 논리모델에서 정의된 관계는 외래키로 변환한다.

2. 변환방법

(1) n 관계에서 1 영역에 있는 기본키를 n 영역의 외래키로 선언한다.

(2) 외래키명은 1 영역의 기본키 이름을 그대로 사용하나 다른 의미를 가질 경우 변경하여 명명할 수 있다.

(3) 순환 관계에서 자신의 기본키는 외래키로 정의된다.

5 칼럼 유형(Type)과 길이(Length)를 정의한다.

1. 정의된 각 칼럼에 대해, 적용 DBMS에서 제공하는 데이터유형 중 적절한 유형을 정의하고, 해당 데이터의 최대 길이를 파악하여 길이를 설정한다.

2. (Oracle 예시) 자주 사용되는 데이터유형

(1) CHAR: 고정길이 문자열 Data 최대 2000Byte까지 저장 가능

(2) VARCHAR2: 가변길이 문자열 Data 최대 4000Byte까지 저장 가능

(3) NUMBER: 38 자릿수의 숫자 저장가능

(4) DATE: 날자값 저장

(5) BLOB, CLOB: Binary, Text Data 최대 4GB까지 저장 가능

6 반정규화(Denormalization)한다.

1. 중복 테이블 추가

   (1) 집계 테이블 추가

   (2) 진행 테이블 추가

   (3) 특정부분만을 포함하는 테이블 추가

2. 테이블 조합

   (1) 1:1 관계의 테이블 조합

   (2) 1:M 관계의 테이블 조합

   (3) 슈퍼타입 서브타입 테이블 조합

3. 테이블 분할

   (1) 수직 분할

   (2) 수평 분할

4. 테이블 제거

   테이블 재정의 또는 칼럼의 중복화로 더 이상 액세스 되지 않는 테이블이 발생할 경우 제거

5. 칼럼의 중복화

   일반적으로 조인 프로세스를 줄이기 위해 칼럼의 중복을 허용

수행 tip

- 물리 데이터 모델은 구축하고자 하는 업무 시스템의 논리 데이터 모델을 적용 데이터베이스에서 지원하는 적절한 구성요소로 매핑하였는지와, 성능향상을 위한 반정규화가 수행되었는지가 중요한 검증 포인트이다.

2-2. 물리 데이터저장소 구성

학습 목표

- 물리 데이터저장소 설계에 따라 데이터저장소에 실제 데이터가 저장될 물리적 공간을 구성할 수 있다.

필요 지식 /

물리 데이터 모델링은 논리모델을 적용하고자 하는 기술에 맞도록 상세화해 가는 과정이다. 따라서 앞으로 기술되는 내용은 특정 적용기술이나 DBMS를 전제할 수 밖에 없기 때문에 시장 점유율을 고려하여 범용적으로 활용되는 기술과 제품을 선택하여야 한다.
앞으로 제시되는 내용은 시장에서 대부분 활용되고 있는 관계형 데이터베이스(RDBMS)의 오라클 데이터베이스를 기준으로 제시한다.

① 테이블 제약조건

실무에서 주로 사용하는 테이블 제약조건으로는 다음 두가지가 있다.

1. Delete Constraint

참조된 기본키의 값이 삭제될 경우의 처리내용을 정의한다.

(1) Cascade: 참조한 테이블에 있는 외부키와 일치하는 모든 Row가 삭제된다.

(2) Restricted: 참조한 테이블에 있는 외부키에 없는 것만 삭제 가능하다.

(3) Nullify: 참조한 테이블에 정의된 외부키와 일치하는 것을 Null로 수정한다.

2. Update Constraint

참조된 기본키의 값이 수정될 경우의 처리내용을 정의한다.

(1) Cascade: 참조한 테이블에 있는 외부키와 일치하는 모든 Row가 수정된다.

(2) Restricted: 참조한 테이블에 있는 외부키에 없는 것만 수정가능하다.

(3) Nullify: 참조한 테이블에 정의된 외부키와 일치하는 것을 Null로 수정한다.(해당 칼럼이 Null을 허용할 경우만)

2 인덱스 설계

1. 인덱스 적용 기준

(1) 인덱스 칼럼의 분포도가 10 ~ 15% 이내인 경우

> **분포도 = (1 / 칼럼값의 종류) X 100**
> **= (칼럼값의 평균 Row수 / 테이블의 총 Row수) X 100**

(2) 분포도가 범위 이상이더라도 부분처리를 목적으로 하는 경우

(3) 입출력 장표 등에서 조회 및 출력 조건으로 사용되는 칼럼인 경우

(4) 인덱스가 자동 생성되는 기본키와 Unique키의 제약조건을 사용할 경우

2. 인덱스 칼럼 선정

(1) 분포도가 좋은 칼럼은 단독적으로 생성하여 활용도를 향상시킨다.

(2) 자주 조합되어 사용되는 칼럼은 결합 인덱스로 생성하여 활용한다.

(3) 결합 인덱스는 구성되는 칼럼순서 선정(사용빈도, 유일성, Sort,...)에 유의해야 한다.

(4) 가능한 한 수정이 빈번하지 않은 칼럼을 선정한다.

3. 설계시 고려사항

(1) 새로 추가되는 인덱스가 기존 액세스 경로에 영향을 미칠 수 있음에 유의한다.

(2) 지나치게 많은 인덱스는 오버헤드로 작용한다.

(3) 인덱스는 추가적인 저장공간이 필요함을 고려해야 한다.

(4) 넓은 범위를 인덱스 처리 시 오히려 전체 처리보다 많은 오버헤드를 발생시킬 수 있음에 유의해야 한다.

(5) 인덱스와 테이블 데이터의 저장 공간을 적절히 분리될 수 있도록 설계해야 한다.

3 뷰 설계

1. 뷰 속성

(1) REPLACE: 뷰가 이미 존재하는 경우 재생성

(2) FORCE: 기본 테이블의 존재 여부에 관계 없이 뷰 생성

(3) NOFORCE: 기본 테이블이 존재할 때 만 뷰 생성

(4) WITH CHECK OPTION: Sub-Query 내의 조건을 만족하는 행만 변경

(5) WITH READ ONLY: DML 작업 불가

2. 뷰 설계 시 고려사항

(1) 최종적으로는 테이블을 액세스하는 것이므로 사용에 따라 수행속도에 문제가 발생할 수 있다.

(2) 뷰내의 SELECT 문의 조건은 가능한 한 최적의 액세스 경로를 사용할 수 있도록 하거나 그럴 수 없다면 뷰를 사용한 SQL의 WHERE 절에서는 반드시 양호한 액세스 경로가 되도록 하여야 한다.

4 클러스터 설계

1. 적용 기준

(1) 분포도가 넓을수록 오히려 유리(인덱스의 단점을 해결)한 기법

(2) 액세스 기법이 아니라 액세스 효율 향상을 위한 물리적 저장 방법

(3) 분포도가 넓은 테이블의 클러스터링은 저장 공간의 절약 가능

(4) 다중(일반적으로 6) 블록 이상의 테이블에 적용

(5) 대량의 범위를 자주 액세스하는 경우 적용

(6) 인덱스를 사용한 처리 부담이 되는 넓은 분포도에 활용

(7) 여러 개의 테이블이 번번히 조인을 일으킬 때 활용

(8) 반복 칼럼이 정규화에 의해 어쩔 수 없이 분할된 경우 활용

2. 클러스터 설계시 고려사항

(1) 검색 효율은 높여 주나 입력, 수정, 삭제 시는 부하가 증가함을 고려하여야 한다.

(2) Union, Distinct, Order by, Group by가 빈번한 칼럼이면 고려해 보아야 한다.

(3) 수정이 자주 발생하지 않는 칼럼은 고려 대상이다.

(4) 처리 범위가 넓어 문제가 발생하는 경우는 단일 테이블 클러스터링을, 조인이 많아 문제가 발생되는 경우는 다중 테이블 클러스터링을 고려하여야 한다.

5 파티션 설계

1. 파티션 종류

(1) 범위분할(Range Partitioning): 지정한 열의 값을 기준으로 분할

(2) 해시분할(Hash Partitioning): 해시 함수에 따라 데이터를 분할

(3) 조합분할(Composite Partitioning): 범위분할에 의해 데이터를 분할한 다음 해시 함수를 적용하여 다시 분할

2. 장점

(1) 데이터 액세스 범위를 줄여 성능 향상

(2) 전체 데이터의 훼손 가능성이 감소 및 데이터 가용성 향상

(3) 각 분할 영역을 독립적으로 백업하고 복구가능

(4) Disk Striping로 I/O 성능을 향상(Disk 컨트롤러에 대한 경합의 감소)

3. 파티셔닝 순서

(1) 파티션의 종류 결정

(2) 파티션 키의 선정

(가) I/O 분산을 어떻게 할 것인가를 고려하여 선정한다.

(나) 액세스 유형에 따라 파티셔닝이 이루어질 수 있도록 파티션 키를 선정한다.

(다) 이력 데이터의 경우, 생성주기 또는 소멸주기가 파티션과 일치하도록 한다.

(3) 파티션 수의 결정

⑥ 디스크 구성 설계

1. 정확한 용량을 산정하여 디스크 사용의 효율을 높인다.

2. 업무량이 집중되어 있는 디스크를 분리하여 설계함으로써 집중화된 디스크에 대한 입출력 부하를 분산한다.

3. 입출력 경합을 최소화하여 데이터의 접근 성능을 향상시킨다.

(1) 테이블 객체를 위한 테이블 스페이스와 인덱스 객체를 위한 테이블스페이스를 분리 구성한다.

(2) 테이블 스페이스와 템포러리 스페이스를 분리 구성한다.

(3) 테이블을 마스터 테이블과 트랜잭션 테이블로 분류한다.

4. 시스템의 구성(Disk의 구성)에 따라 테이블스페이스의 개수와 사이즈 등을 결정한다.

5. 파티션할 테이블은 별도로 분류한다.

수행 내용 / 물리 데이터저장소 구성하기

재료 · 자료

- ERD 작성표준, 개발 시 활용 데이터 저장소에 대한 매뉴얼 및 개발 가이드, 모델링 검토기준

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 데이터저장소 설계 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

물리 데이터모델링이 완료되면, 모델링 결과에 따라 디스크라는 물리 데이터저장소에 다양한 오브젝트를 구성해야 하는데, 데이터저장소의 주요 오브젝트로는 테이블, 인덱스, 뷰, 클러스터, 파티션 등이 있다. 이러한 오브젝트는 디스크 구성 설계를 통해 구성하게 된다.

1 다양한 오브젝트를 설계한다.

1. 테이블 제약조건 설계

   참조 무결성을 관리하기 위한 제약조건(Constraint)을 정의한다.

2. 인덱스 설계

   부분 범위 데이터 검색 시 전체 테이블을 검색하지 않고 빠른 검색을 위해 특정 칼럼들에 대해서 미리 인덱싱(정렬) 작업을 함으로써 해당 인덱스를 이용하여 빠른 검색을 할 수 있도록 하는 기법을 말한다.

3. 뷰 설계

   테이블을 기초로 하는 가상(논리) 테이블을 말한다.

4. 클러스터 설계

   지정된 칼럼 값의 순서대로 데이터 행을 저장하는 방법으로, 하나 혹은 그 이상의 테이블을 같은 클러스터내 저장이 가능하다.

5. 파티션 설계

   대용량DB는 몇 개의 중요한 트랜잭션 테이블에서 데이터가 증가하므로, 보다 작은 단위로 나눔으로써 성능 저하 방지와 관리의 용이성을 위해 사용하는 기법을 말한다.

2 디스크 구성 설계를 한다.

위의 다양한 오브젝트의 디스크 구성 설계 시, 본 장의 필요지식에서 제시된 디스크 구성 설계 시 고려사항을 참조하여 설계하도록 한다.

수행 tip

- 물리 데이터 모델링 결과에 따라 물리 데이터 저장소의 다양한 오브젝트의 특성을 고려하여 디스크 구성설계가 적절히 수행되었는지가 중요한 검증 포인트이다.

학습 2 교수 · 학습 방법

교수 방법

- 교수자의 주도로 분석 단계에서 작성된 논리 데이터 모델을 이해하고, 적용하고자 하는 데이터베이스의 특성과 다양한 데이터 저장소의 오브젝트 특성을 고려하여 매핑하는 물리 데이터 모델 내용을 PPT 자료로 제시한 후 설명한다.
- 정규화된 논리 데이터모델을 업무처리 범위와 빈도수를 고려한 반정규화를 통해 전체적인 데이터베이스 성능을 보장할 수 있도록 하는 필요성과 과정을 도식화하여 제시하고 설명한다.
- 실무에서 가장 많이 활용하는 테이블 인덱스 선정기준에 따라 인덱스 칼럼을 선정하여 인덱스를 설계하는 과정을 이해할 수 있도록 PPT 자료로 제시한 후 설명한다.
- 테이블 제약조건과 뷰, 클러스터, 파티션 등의 다양한 오브젝트 특성을 고려하여 이를 적용하는 과정을 이해할 수 있도록 설명한다.
- 데이터베이스에서 제공하는 다양한 데이터 저장소 오브젝트의 특성을 고려하여 전반적인 데이터베이스 성능을 보장할 수 있도록 해야 하는 필요성과 과정을 이해할 수 있도록 설명한다.

학습 방법

- 작성된 논리 데이터 모델을 읽어 업무를 이해하여 적용 데이터베이스 특성을 고려한 물리 모델링 과정과 데이터 저장소의 다양한 오브젝트 특성과 활용 용도에 대해 이해한다.
- 반정규화 과정과 데이터 저장소의 오브젝트 생성 방법을 이해하고 실습한다.
- 반정규화 유형별 특성과 적용에 따른 장단점을 이해하여 적용할 수 있도록 학습한다.
- 인덱스 내부 메커니즘 기반 하에, 인덱스 선정기준에 따른 인덱스 대상 칼럼 선정 방안을 명확히 이해하여 데이터 액세스 속도를 보장할 수 있도록 학습한다.
- 데이터베이스에서 제공하는 다양한 데이터 저장소 오브젝트 특성을 이해함으로써 데이터베이스 성능을 보장할 수 있도록 학습한다.

학습 2 평 가

평가 준거

● 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.

● 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준 상	중	하
물리데이터모델 설계	- 논리 데이터저장소 설계를 바탕으로 응용소프트웨어가 사용하는 데이터저장소의 특성을 반영한 물리 데이터저장소 설계를 수행할 수 있다.			
	- 논리 데이터저장소 설계를 바탕으로 목표 시스템의 데이터 특성을 반영하여 최적화된 물리 데이터저장소를 설계할 수 있다.			
물리데이터저장소 구성	- 물리 데이터저장소 설계에 따라 데이터저장소에 실제 데이터가 저장될 물리적 공간을 구성할 수 있다.			

평가 방법

● 평가자 체크리스트

학습내용	평가항목	성취수준 상	중	하
물리데이터모델 설계	- 논리 데이터 모델의 물리 데이터 모델로의 매핑 적정성 - 반정규화 적용범위 및 타당성			
물리데이터저장소 구성	- 데이터베이스 지원 데이터저장소 오브젝트 특성 이해 - 오브젝트 설계 기준의 적절성			

● 구두발표

학습내용	평가항목	성취수준 상	중	하
물리데이터모델 설계	- 논리 데이터 모델의 물리 데이터 모델로의 매핑 적정성 - 반정규화 적용범위 및 타당성			
물리데이터저장소 구성	- 데이터베이스 지원 데이터저장소 오브젝트 특성 이해 - 오브젝트 설계 기준의 적절성			

피 드 백

1. 체크리스트를 통한 관찰
   - 검증과정에서 학습자의 경험과 지식뿐만 아니라 분석, 의견 조율 및 조정 능력을 확인하고 개선해야 할 사항을 알려준다.
   - 체크리스트에 위배되는 항목에 대해 명확히 정리하여 돌려준다.
   - 논리데이터 모델의 물리 데이터 모델로의 매핑 과정을 파악하여, 미비사항을 정리하여 돌려준다.
   - 반정규화 적용범위 및 적정성에 따라 반정규화 되었는지 판단해보고, 미비사항을 정리하여 돌려준다.
2. 구두 발표
   - 발표한 내용 중 미비사항이나 보완해야 할 사항을 정리하여 돌려준다.
   - 물리 데이터모델링의 반정규화의 필요성과 적용 후 개선된 점을 파악하여, 실무에서 자주 활용될 수 있도록 한다.
   - 데이터베이스 지원 데이터저장소 오브젝트 선정과정을 파악해 보고, 설계기준에 미비한 점을 정리하여 돌려준다.

학습 1	논리 데이터저장소 확인하기(LM2001020205_14v2.1)
학습 2	물리 데이터저장소 설계하기(LM2001020205_14v2.2)
학습 3	**데이터 조작 프로시저 작성하기 (LM2001020205_14v2.3)**
학습 4	데이터 조작 프로시저 최적화하기 (LM2001020205_14v2.4)

3-1. 데이터 조작 프로시저 개발

학습 목표

- 응용소프트웨어 설계와 물리 데이터저장소 설계에 따라 데이터 저장소에 연결을 수행하는 프로시저를 작성할 수 있다.
- 응용소프트웨어 설계와 물리 데이터저장소 설계에 따라 데이터 저장소로부터 데이터를 읽어 오는 프로시저를 작성할 수 있다.
- 응용소프트웨어 설계와 물리 데이터저장소 설계에 따라 데이터 변경 내용 또는 신규 입력된 데이터를 데이터 저장소에 저장하는 프로시저를 작성할 수 있다.

필요 지식 /

데이터 조작을 위해서 사용하는 언어를 SQL(Structured Query Language)이라고 하는데, SQL은 일정한 데이터 집합으로부터 보다 쉽게 자료를 검색하고 입력, 수정, 삭제와 같은 조작을 할 수 있도록 고안된 언어를 말한다.

일반적으로 SQL은 데이터 정의어(DDL: Data Definition Language), 데이터 조작어(DML: Data Manipulation Language), 데이터 제어어(DCL: Data Control Language) 등의 세가지로 분류한다.

1 SQL 분류

1. 데이터 정의어(DDL: Data Definition Language)

데이터를 저장하고 있는 테이블 등의 구조를 생성하고 변경하기 위하여 사용되는 명령어들을 말하는 것으로, 명령어 수행이 되면 이전 상태로 복귀할 수 없으므로 신중히 사용하여야 한다.

(1) 종류

CREATE, DROP, RENAME, ALTER, TRUNCATE 등이 있다.

(2) CREATE: 오브젝트 생성

(가) 문법

CREATE TABLE My_table(my_field1 NUMBER, my_field2 VARCHAR2(20), my_field3 DATE NOT NULL, CONSTRAINT Tbl_Col_pk Primary Key(my_field1));

(나) 예시

CREATE TABLE dept1 (deptno number(2), dname varchar2(14), loc varchar2(13));

(3) DROP: 오브젝트 삭제

(가) 문법

DROP TABLE My_table;

(나) 예시

DROP TABLE dept1;

(4) RENAME: 오브젝트 이름 변경

(가) 문법

RENAME My_Table TO My_Backup_Table;

(나) 예시

RENAME dept1 TO dept_copy;

(5) ALTER: 오브젝트 구조 변경

(가) 문법

ALTER TABLE My_Table [ADD/MODIFY] (my_field2 varchar2(30));

(나) 예시

ALTER TABLE dept_copy MODIFY (loc varchar2(14));

(6) TRUNCATE: 오브젝트 자름

(가) 문법

TRUNCATE TABLE My_Table;

(나) 예시

TRUNCATE TABLE dept_copy;

2. 데이터 조작어(DML: Data Manipulation Language)

데이터베이스에 있는 데이터를 변경하거나 검색하기 위하여 사용되는 명령어들을 말하며, 이 명령어는 트랜잭션 제어어(Transaction Control Language)를 활용하여 실행 전 상태로 복귀가능한 명령어이다.

(1) 종류

INSERT, UPDATE, DELETE 등이 있다

(2) INSERT: 데이터 입력

(가) 문법

INSERT INTO My_Table [(empno)] [VALUES (1111); / SELECT .. ;]

1) 입력하고자 하는 테이블의 모든 칼럼 데이터를 입력한다면 칼럼명을 명시하지 않아도 되나, 특정 칼럼만을 입력하고자 한다면 반드시 칼럼명을 명시하여야 한다.

2) 반드시 칼럼명 수와 VALUES 절의 수는 동일해야 한다.

3) 기존에 존재하는 테이블 데이터로부터 특정 테이블로 데이터를 복사하고자 한다면 'INSERT INTO emp(empno) SELECT id FROM emp_src'와 같이 사용할 수 있다.

(나) 예시

INSERT INTO dept_copy(deptno, hr_limit) VALUES(92,10);

(3) UPDATE: 데이터 수정

(가) 문법

UPDATE emp SET empno = 1234 [, ename = 'James']

WHERE empno = 1111;

(나) 예시

UPDATE dept_copy SET hr_limit=20 WHERE hr_limit is null;

(4) DELETE: 데이터 삭제

(가) 문법

DELETE [FROM] My_Table WHERE my_field2 = 'ABCD';

(나) 예시

DELETE dept_copy WHERE deptno > 90;

3. 데이터 제어어(DCL: Data Control Language)

사용자별로 데이터베이스에 접근할 수 있는 권한을 부여하거나 회수하는 명령어들을 말한다.

(1) 종류

ROLE, GRANT, REVOKE 등이 있다.

(2) ROLE: 롤

(가) 문법

CREATE ROLE Role_name;

1) Role_name 선언 후 GRANT로 권한을 Role_name으로 부여

2) Oracle에서는 일반적으로 많이 사용하는 권한을 묶어 3가지 기본 Role을 제공

- CONNECT: 데이터베이스 접속 권한
- RESOURCE: Object 생성권한
- DBA: 모든 권한

(나) 예시

1) CREATE ROLE manager;

2) GRANT create table, create view TO manager;

3) GRANT manager TO scott;

(3) GRANT: 권한 및 롤 부여

(가) 문법

GRANT 부여할 권한 유형 TO User [Role_name];

(나) 예시

1) GRANT connect, resource to scott;

2) GRANT SELECT ON emp TO scott[PUBLIC] [With Grant/Admin Option];

Grant/Admin Option은 둘 다 실행 권한을 받은 사용자가 다시 다른 사용자에게 실행 권한을 부여해 줄 수 있게 해주는 option이다. 다만, 두 Option 간 차이는,

- With Grant Option: revoke 시 다른 사용자에게 부여된 권한도 함께 회수된다.
- With Admin Option: revoke 시 다른 사용자에게 부여한 권한은 함께 회수되지 않으므로 Admin Option의 사용은 신중을 기해야 한다.

(4) REVOKE: 권한 및 롤 회수

(가) 문법

REVOKE 회수할 권한 유형 FROM User;

(나) 예시

1) REVOKE connect, resource FROM scott;

2) REVOKE SELECT ON emp FROM scott;

2 트랜잭션 제어어(TCL: Transaction Control Language)

트랜잭션 제어어는 트랜잭션의 DML작업단위를 제어하는 명령어이다.

1. 종류

COMMIT, ROLLBACK, SAVEPOINT 등이 있다.

(1) COMMIT: 트랜잭션을 완료하여 데이터 변경사항을 최종 반영

(가) COMMIT 이후 데이터 상태

1) 데이터에 대한 변경 사항이 물리적인 디스크에 반영된다.

2) COMMIT 이전 데이터는 복구할 수 없다.

3) 모든 사용자가 변경된 결과를 볼 수 있다.

4) 관련된 행에 대해 잠금(Lock)이 풀리며 다른 사용자들이 조작할 수 있다.

(나) 문법

COMMIT;

(2) ROLLBACK: 데이터 변경사항을 이전 상태로 되돌리는 명령어

(가) ROLLBACK 이후 데이터 상태

1) ROLLBACK된 DML 문장은 메모리 상의 Buffer에만 영향을 미치기 때문에 복구가 가능하다.

2) 관련된 행에 대한 잠금(Lock)이 풀리게 된다.

(나) 문법

ROLLBACK;

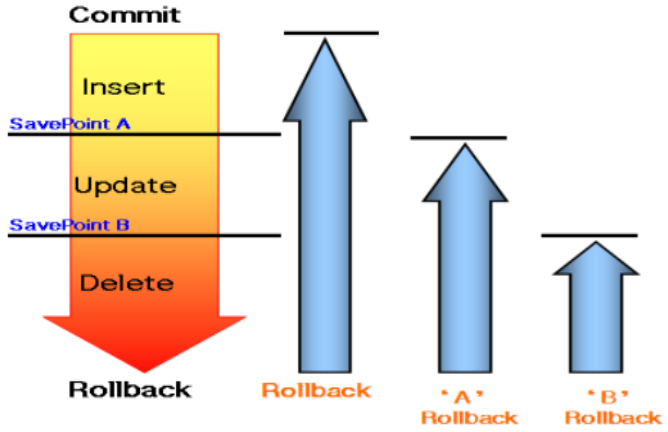
(3) SAVEPOINT: 지정된 특정 시점까지 Rollback할 수 있는 명령어

(가) 문법

SAVEPOINT 특정지점 명칭;

(나) SAVEPOINT 활용 예시

```
1) INSERT…;
2) SAVEPOINT A;
3) UPDATE ...;
4) SAVEPOINT B;
5) DELETE …;
6) ROLLBACK TO A;
```



[그림 3-1] SavePoint 적용범위

3 데이터 검색어(SELECT)

1. 문법

SELECT [DISTINCT] {*, column [alias], . . .}

FROM table_name

[WHERE condition]

[GROUP BY column]

[HAVING condition]

[ORDER BY {column, expression} [ASC ¦ DESC]];

2. 항목 설명

(1) DISTINCT: 중복되는 행을 제거하는 옵션

(2) *: 테이블의 모든 column을 출력

(3) alias: 해당 column에 대해서 다른 이름을 부여할 때 사용
alias 지정은 space 다음에 alias명 또는 AS alias로 지정 가능

(4) table_name: 질의 대상 테이블명

(5) WHERE: 조건을 만족하는 행들만 검색
condition 은, column, 표현식, 상수 및 비교 연산자

(6) GROUP BY: 그룹핑하고자 하는 단위 지정

(7) HAVING: 그룹핑한 결과값에 대한 조건 검색

(8) ORDER BY: 질의 결과 정렬을 위한 옵션(ASC:오름차순(Default), DESC내림차순)

4 절차형 데이터 조작 프로시저 및 저장형 객체 활용

1. 절차형 데이터 조작 프로시저

본 학습모듈은 표준 SQL을 기본으로 하지만, 데이터베이스에 종속적인 특정 영역에 대해서는 시장 점유율을 고려하여 Oracle 데이터베이스를 전제하기로 하였으므로, 절차형 데이터 조작 프로시저 또한 Oracle에서 제공하는 PL/SQL을 기준으로 설명하기로 한다.

(1) PL/SQL 개요

최근의 프로그래밍 언어의 특성을 수용한, SQL의 확장 기능이라 할 수 있다.

(가) 사용시 장점

1) Compile이 필요 없어 script 생성 및 변경 후 바로 실행이 가능하다.

2) 프로그램 개발의 모듈화가 가능하다.

- 블록 내에서 논리적으로 관련된 문장들을 그룹화할 수 있다.
- 강력한 프로그램을 작성하기 위해 서브 블록들을 큰 블록에 포함할 수 있다.
- 복잡한 문제에 대한 프로그래밍이 적절히 나뉘어진 모듈들의 집합으로 구성할 수 있다.

3) 식별자를 선언할 수 있다.

- 변수, 상수 등을 선언하여 해당 식별자를 SQL과 절차적인 프로그램에서 사용할 수 있다.
- 데이터베이스의 테이블과 Record를 기반으로 하는 Dynamic한 변수 선언이 가능하다.
- 단일형 데이터 타입과 복합형 데이터 타입을 선언할 수 있다.

4) 절차적 언어 구조로 된 프로그램을 작성할 수 있다.

- IF문을 통해 조건에 따라 일련의 문장을 실행할 수 있다.
- LOOP문을 사용하여 일련의 문장을 반복적으로 실행할 수 있다.
- Explicit Cursor를 이용한 Multi-row 처리가 가능하다.

5) ERROR 처리가 가능하다.

- Exception 처리 루틴을 이용하여 Oracle Server 에러를 처리할 수 있다.
- 사용자 정의 에러를 선언하고 Exception 처리 루틴으로 처리가 가능하다.

6) 성능 향상을 기대할 수 있다.

- PL/SQL은 네트워크 부하를 줄여 프로그램의 성능을 향상시킬 수 있다.
- PL/SQL은 여러 SQL문장을 BLOCK으로 묶고 한번에 BLOCK전부를 서버로 전송하기 때문에 통신량을 줄일 수 있어 성능향상을 기대할 수 있다.

(나) PL/SQL 구조

PL/SQL은 프로그램을 논리적인 블록으로 나누게 하는 구조화된 블록 언어로서, 다음과 같은 블록 구조로 구성된다.

1) 선언부 (DECLARE, Optional)

실행부에서 참조할 모든 변수, 상수, CURSOR, EXCEPTION을 선언한다.

2) 실행부 (BEGIN/END, Mandatory)

- BEGIN과 END 사이에 기술되는 영역이다.
- 데이터베이스 데이터를 처리할 SQL문과 PL/SQL 블록을 기술한다.

3) 예외 처리부 (Exception, Optional)

실행부에서 에러가 발생했을 때 수행될 문장을 기술

(2) PL/SQL 처리 절차

(가) PL/SQL로 작성된 Block을 Oracle 서버로 보내면 PL/SQL 엔진이 SQL문과 Non SQL문을 구분한다.

(나) Non SQL문은 PL/SQL Engine내의 Procedural Statement Executor가, SQL문은 SQL Statement Executor가 처리하게 된다

(다) Non SQL문은 Client환경에서, SQL문은 서버에서 실행하게 된다.

(라) 따라서, PL/SQL을 사용하게 되면 서버의 작업 양을 줄이게 되므로 네트워크 부하

를 감소시켜 수행성능을 증가시키는 잇점이 있다.

(3) PL/SQL 프로그래밍 가이드

(가) PL/SQL Block내에서는 한 문장이 종료할 때마다 ';' 을 기술한다.

1) END 뒤에도 ';' 을 사용하여 Block이 끝났다는 것을 명시하여야 한다.

2) PL/SQL을 실행은 "/" 을 사용하고, 성공적으로 실행 된다면 "… successfully completed" 라는 메시지가 출력되므로 이를 확인하여야 한다.

(나) PL/SQL Block의 작성은 편집기를 통해서나 SQL*Plus에서 바로 작성하여 실행할 수 있고, PL/SQL 실행 시 발생한 Error는 show errors 명령어로 확인한다.

(다) PL/SQL 블록을 개발할 때 명확한 코드 생성과 유지보수를 위하여 프로그래밍 가이드를 정의, 준수, 수행하는 것이 좋다.

(라) 코드의 가독성을 높이기 위하여 들여쓰기를 하도록 한다.

2. PL/SQL을 활용한 저장형 객체 활용

PL/SQL로 작성할 수 있는 저장형 객체로는 Stored Function, Stored Procedure, Stored Package, Trigger 등이 있는데, 각 저장형 객체의 특성과 작성을 위한 문법은 다음과 같다.

(1) Stored Function

(가) 특성

1) 보통 값을 계산하고 결과값을 반환하기 위해서 많이 사용한다.

2) 대부분 구성이 프로시저와 유사하지만 IN 파라미터만 사용할 수 있다.

3) 반드시 반환될 값의 데이터 타입을 RETURN문에 선언해야 한다.

4) PL/SQL블록 내에서 RETURN문을 통해서 반드시 값을 반환해야 한다.

(나) 문법

```
SQL> CREATE OR REPLACE FUNCTION function name
        [(argument...)]
        RETURN  datatype  -- 반환되는 값의 datatype임
        IS
            [변수 선언 부분]
        BEGIN
           [PL/SQL Block]
              -- PL/SQL 블록에는 적어도 한 개의 RETURN 문이 있어야 함
              -- PL/SQL Block은 함수가 수행할 내용을 정의한 몸체부분임
      END;
```

(2) Stored Procedure

(가) 특성

1) 특정 작업을 수행할 수 있는, 이름이 있는 PL/SQL 블록이다.

2) 매개 변수를 받을 수 있고 반복적으로 사용할 수 있는 Object이다.

3) 보통 연속 실행 또는 구현이 복잡한 트랜잭션을 수행하는 PL/SQL블록을 DB에 저장하기 위해 생성한다.

4) 생성 방법

- CREATE OR REPLACE 구문을 사용하여 생성한다
- IS 로 PL/SQL의 블록을 시작한다.
- LOCAL 변수는 IS 와 BEGIN 사이에 선언한다.

(나) 문법

```
SQL> CREATE OR REPLACE procedure_name
                IN argument
                OUT argument
                IN OUT argument
          IS
              [변수의 선언]
        BEGIN
              [PL/SQL Block]
               -- SQL문장, PL/SQL제어 문장
              [EXCEPTION]  --> 선택
               -- error가 발생할 때 수행하는 문장
        END;
```

1) Parameter

- 실행 환경과 프로그램 사이에 값을 주고받는 역할을 한다.
- 블록 안에서의 변수와 똑같이 일시적으로 값을 저장하는 역할을 한다.

2) Parameter의 타입

- IN: 실행환경에서 Program으로 값을 전달
- OUT: Program에서 실행환경으로 값을 전달
- INOUT: 실행환경에서 프로그램으로 값을 전달하고, 다시 프로그램에서 실행환경으로 변경된 값을 전달한다.

(3) Stored Package

(가) 특성

1) 패키지는 오라클 데이터베이스에 저장되어 있는 서로 관련있는 PL/SQL 프로

시저와 함수들의 집합이다.

2) 패키지는 선언부와 본문 두 부분으로 나누어진다.

- 패키지 선언부: 선언절은 패키지에 포함될 PL/SQL 프로시저나, 함수, 커서, 변수, 예외절을 선언하고, 패키지 선언부에서 선언한 모든 요소들은 패키지 전체에 적용되며, 선언부에서 선언한 변수는 PUBLIC 변수로 사용됨
- 패키지 본문: 패키지에서 선언된 부분의 실행을 정의하는 영역이며, 실제 프로시저나 함수의 내용에 해당하는 내용으로 구성된다.

(나) 문법

1) 패키지 선언부

```
CREATE [OR REPLACE] PACKAGE 패키지명 IS ¦ AS
[변수 선언절] [커서 선언절] [예외 선언절]
[PROCEDURE 선언절] [FUNCTION 선언절]
END 패키지명;
/
```

2) 패키지 본문

```
CREATE [OR REPLACE] PACKAGE BODY 패키지명 IS ¦ AS
[변수 선언절] [커서 선언절] [예외 선언절]
[PROCEDURE 선언절] [FUNCTION 선언절]
END 패키지명;
/
```

(4) Trigger

(가) 특성

1) INSERT, UPDATE, DELETE문이 TABLE에 대해 행해질 때 묵시적으로 수행되는 PROCEDURE이다.

2) Trigger는 TABLE과는 별도로 DATABASE에 저장된다.

3) Trigger는 VIEW에 대해서가 아니라 TABLE에 관해서만 정의될 수 있다.

4) DBMS_OUTPUT.PUT_LINE 을 출력하기 위해 'set serveroutput on' 을 사용한다.

(나) 문법

```
CREATE [OR REPLACE] TRIGGER 트리거명
[시점] [이벤트] [OF] ON 테이블명
[FOR EACH ROW]
[WHEN]
DECLARE
변수 선언 . . .
```

BEGIN

. . .

END ;

/

1) 시점

- BEFORE: INSERT, UPDATE, DELETE문이 실행되기 전에 트리거가 실행된다.
- AFTER: INSERT, UPDATE, DELETE문이 실행된 후 트리거가 실행된다.

2) 이벤트

trigger_event: INSERT, UPDATE, DELETE 중에서 한 개 이상 올 수 있다.

3) FOR EACH ROW

- 이 옵션이 있으면 행 트리거가 된다.
- 행 트리거: 칼럼의 각각 행의 데이터 행 변화가 생길 때마다 실행되며, 그 데이터 행의 실제 값을 제어할 수 있다.
- 문장 트리거: 트리거 사건에 의해 단 한번 실행되며, 칼럼의 각 데이터 행을 제어할 수 없다.

수행 내용 / 데이터 조작 프로시저 개발하기

재료・자료

- ERD 작성표준, 개발 시 활용 데이터 저장소에 대한 매뉴얼 및 개발 가이드

기기(장비・공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 데이터저장소 설계 프로그램

안전・유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

물리 데이터 모델 설계가 완료되면, 설계된 내용대로 데이터베이스를 구축한 후, 개발자는 1. 구축 데이터 저장소에 연결을 수행하여, 2. 데이터 저장소 오브젝트를 생성하고, 3. 데이터의 입력 및 변경(수정, 삭제)을 수행한 후, 4. 저장된 데이터를 검색하는 프로시저를 작성하는 순서로 애플리케이션을 개발하게 되며, 5. 절차형 데이터 조작 프로시저 작성함으로써 데이터 활용능력을 확장할 수 있다.

[1] 데이터 저장소에 연결한다.

일반적으로 많이 사용하고 있는 Java 환경의 경우, 구축된 데이터베이스로의 연결은 JDBC를 통해 다음과 같은 순서로 연결하게 된다.

1. JDBC 개념

   Java 환경에서 데이터베이스 내의 존재하는 데이터를 활용하기 위해 SQL이 필요하게 되는데 이를 연결해 주는 응용프로그램 인터페이스를 말한다.

2. 연결 순서

   (1) 드라이버 로딩

   DB와 연결하기 위해 DBMS에서 제공하는 jar파일 드라이버를 메모리에 적재한다.

   (예) oracle.jdbc.driver.OracleDriver;

   (2) Connection

   해당 드라이버를 사용하여 DB를 연결한다

   (예) String url = "jdbc:oracle:thin:@localhost:1521:ORCL" ;

   conn=DriverManager.getConnection(url,"scott" ,"tiger");

   (3) 쿼리 전달

   쿼리를 DB로 전달하기 위해 Statement, PreparedStatement 객체를 생성한다.

   (예) pstmt=conn.prepareStatement(sql);

   (4) 결과 수신

   전달된 쿼리의 수행으로 인한 반환 값을 수신한다.

   (예) ResultSet rs=pstmt.executeQuery();

[2] 데이터 저장소를 정의한다.

1. 데이터 저장소를 생성한다.
2. 생성된 데이터 저장소를 변경한다.
3. 데이터 저장소를 삭제한다.

3 데이터 조작 프로시저를 작성한다.

1. 생성된 데이터 저장소에 데이터를 입력한다.
2. 입력된 데이터를 수정한다.
3. 저장된 데이터를 삭제한다.

4 데이터 검색 프로시저를 작성한다.

1. 검색 조건에 맞는 데이터를 조회한다.
2. 다양한 함수를 활용하여 데이터를 조회한다.

5 절차형 데이터 조작 프로시저를 작성한다.

1. SQL문의 확장으로, DML 문장과 검색 프로시저를 블록 구조에 절차적 단위(IF, LOOP, FOR등)로 된 코드를 포함함으로써 절차적 프로그래밍을 가능하게 한 강력한 트랜잭션 처리언어인 PL/SQL을 활용하여 프로시저를 작성한다.
2. PL/SQL로 작성할 수 있는 저장형 프로시저 객체 유형을 정의한다.
   저장형 프로시저로 작성할 수 있는 객체 유형은 다음과 같다.
   (1) Stored Function
   (2) Stored Procedure
   (3) Stored Package
   (4) Trigger
3. 정의한 객체를 생성한다.
4. 생성하여 저장된 프로시저 객체를 활용한다.

수행 tip

- 데이터 저장소의 오브젝트에 대한 DDL, DML, DCL, TCL, SELECT를 활용하여 원하는 데이터를 추출할 수 있는지와 절차형 프로그래밍을 통해 저장형 객체를 생성하여 활용할 수 있는지가 중요한 검증 포인트이다.

3-2. 데이터 조작 프로시저 테스트

학습 목표

- 구현된 데이터 조작 프로시저를 테스트할 수 있는 테스트 케이스를 작성하고 단위 테스트를 수행하기 위한 테스트 조건을 명세화 할 수 있다.

필요 지식 /

Oracle DBMS는 모든 데이터조작 프로시저에 대한 테스트 환경으로 SQL*Plus라는 도구를 제공하므로, 개발자는 데이터조작 프로시저 테스트를 위해 해당 도구 활용을 위한 SQL*Plus 명령어에 대한 사전 지식이 필요하다. 특히 절차형 SQL인 PL/SQL의 경우는 디버깅을 위한 환경과 관련 명령어가 추가적으로 요구되므로 효율적인 테스트를 위해 이와 관련된 지식을 충분히 숙지하여 익숙하게 활용할 수 있어야 한다.

1 SQL*Plus 활용

SQL*Plus 로그인 및 활용을 위한 SQL*Plus 명령어는 다음과 같다.

1. 개요

[그림 3-2] SQL*Plus 로그인 및 SQL*Plus 명령어와 SQL 차이점

2. SQL과 SQL*Plus 차이점

SQL은 데이터를 조작하는 표준 언어인 반면 SQL*Plus는 이러한 SQL을 DBMS 서버에 전송하여 처리할 수 있도록 하는 Oracle에서 제공하는 도구인 것이 가장 큰 차이점이라 할 수 있다.

<표 3-1> SQL과 SQL*Plus 차이점

SQL	SQL*Plus
데이터베이스와 통신하는 언어	SQL 명령어를 서버에 전송하는 Tool
ANSI 표준에 기초	Oracle사 제공 Tool
데이터와 테이블에 대한 정의가 가능	데이터에 대한 어떤 정의도 불가능
SQL buffer를 사용	SQL buffer를 사용하지 않음
여러 행 입력 가능	여러 행 입력할 수 없음
명령어 실행시 종료문자(;) 사용	명령어 실행시 종료문자(;) 사용 안함
키워드를 축약할 수 없음	키워드를 축약할 수 있음

3. SQL*Plus 명령어 유형

(1) 파일 명령어: SAVE, GET, SPOOL 등

(2) 편집 명령어: A, C, L, I, DEL, n(숫자) 등

(3) 실행 명령어: START, @, RUN, / 등

(4) 환경 명령어: SET HEAD[LINE/PAGE/PAUSE] ON[OFF] 등

(5) 형식 명령어: COLUMN, TTITLE, BTITLE, BREAK 등

(6) 대화 명령어: DEFINE, PROMPT, ACCEPT 등

4. SQL*Plus 명령어 유형별 처리 내용

<표 3-2> SQL*Plus 명령어 내용

유형	명령어	내용
파 일 명령어	EDIT {파일명}	버퍼의 내용을 편집기로 불러온다
	SAVE {파일명}	버퍼의 내용을 파일에 저장한다
	START {파일명} (=@)	저장된 SQL script를 실행한다
	GET {파일명}	파일의 내용을 버퍼로 읽어온다
	SPOOL {파일명}	조회결과를 파일로 저장한다
	SPOOL OFF	
	HOST (=!와 동일한 효과)	운영체제(shell)로 빠져 나간다
	EXIT	운영체제(O/S) prompt로 빠져 나간다
	CONNECT {uid/pwd}	다른 사용자로 접속할 때 사용한다
	COL col FOR '999,999' [A15]	Col 내용을 일정 Format으로 변경한다

유형	명령어	내용
편 집 명령어	A {문자스트링}	현재 버퍼의 끝에 새로운 문자 스트링을 추가
	C	현재 행의 문자열을 치환한다
	L	버퍼의 전체 리스트를 출력한다
	I	버퍼에 새로운 행을 추가한다
	DEL n	현재 행을 삭제한다
	N(숫자)	현재 행을 출력한다
	CLEAR BUFFER	버퍼의 전체 내용을 삭제한다
실 행 명령어	START {파일명}	SQL script를 실행할 때
	@ {파일명}	SQL script를 실행할 때:START와 동일
	RUN {파일명}	버퍼의 내용을 실행할 때
	/	버퍼의 내용을 실행할 때
환 경 명령어	SET ECHO {off¦on}	SQL script를 실행할 때 명령어의 출력여부
	FEED[BACK] {6¦n¦off¦on}	조회결과 메시지 출력여부
	HEAD[ING] {on¦off}	칼럼의 Head 출력여부
	LINE[SIZE] {80¦n}	출력될 한 라인의 길이
	PAGE[SIZE] {24¦n}	출력 Page 당 라인 수
	PAU[SE] {off¦on}	화면 이동제어(한 Page씩 보고 싶을 때)
	SQLPREFIX {#¦c}	SQL명령어 사이에 SQL*Plus 명령어를 사용할 때
	NULL {text}	NULL값을 대체할 text 정보를 설정할 때
	SERVEROUTPUT {on¦off}	PL/SQL 처리결과를 화면에 출력하고자 할때
	SPACE {1¦n}	출력된 칼럼 간의 여유공간을 설정할 때
	UNDERLINE {기호¦on¦off}	칼럼의 heading 밑에 사용될 Underline 을 설정
	WRAP {on¦off}	칼럼들이 지정된 Linesize를 초과할때 출력여부
형 식 명령어	COLUMN	칼럼의 FORMAT을 변경할 때
	TTITLE	보고서의 제목을 설정할 때
	BTITLE	보고서의 꼬리말을 설정할 때
	BREAK	칼럼 또는 행의 값이 바뀔 때 마다 새로운 보고서 Format을 설정할 때
대 화 명령어	DEFINE	CHAR 데이터형의 사용자 변수를 생성
	UNDEFINE	정의한 사용자 변수를 해제
	PROMPT	PROMPT 지정
	ACCEPT	변수를 생성하여 특정 칼럼에 가변 값을 입력

2 PL/SQL 테스트를 위한 SQL*Plus 활용

1. PL/SQL 테스트

(1) DBMS_OUTPUT 패키지 활용

메시지를 버퍼에 저장하고 버퍼로부터 메시지를 읽어오기 위한 인터페이스를 제공하는 패키지인 DBMS_OUTPUT을 코드에 포함하여야 한다.

(가) 패키지 제공 메소드

1) DISABLE: 메시지 버퍼 내용 삭제

2) ENABLE: 메시지 버퍼 내용 할당

3) PUT: 메시지 버퍼에 저장되는 메시지의 마지막 라인 끝에 새로운 라인문자(EOL)가 추가되지 않음

5) PUT_LINE: PUT과 달리 메시지 끝에 새로운 라인문자가 추가됨

6) GET_LINE: 한 번 호출될 때마다 하나의 라인만을 읽어옴

7) GET_LINES: 지정된 라인을 읽어옴

(나) 활용 예시

1) 예시

```
SQL> CREATE OR REPLACE PROCEDURE Type_Test
( p_empno IN emp.empno%TYPE )
        IS
            -- %TYPE 데이터형 변수 선언
            v_empno emp.empno%TYPE;
            v_ename emp.ename%TYPE;
            v_sal    emp.sal%TYPE;
        BEGIN
            DBMS_OUTPUT.ENABLE;
            -- %TYPE 데이터형 변수 사용
            SELECT empno, ename, sal
            INTO v_empno, v_ename, v_sal
            FROM emp
            WHERE empno = p_empno ;
            -- 결과값 출력
            DBMS_OUTPUT.PUT_LINE( '사원번호: ' || v_empno );
            DBMS_OUTPUT.PUT_LINE( '사원이름: ' || v_ename );
            DBMS_OUTPUT.PUT_LINE( '사원급여: ' || v_sal );
        END;
       /
```

(2) 실행 방법

(가) PL/SQL의 처리결과를 화면에 출력하기 위한 SERVEROUTPUT을 ON 시키고, 실행하고자 하는 PL/SQL 블록 또는 저장객체명을 호출한다.

(나) 활용 예시

SQL> SET SERVEROUTPUT ON

SQL> EXECUTE Type_Test(7369);

(다) 처리결과 예시

사원번호: 7369

사원이름: SMITH

사원급여: 800

(3) 오류 조치 및 확인

PL/SQL 실행 시 오류가 발생하면 'SHOW ERRORS' 명령어를 통해 오류내용을 확인하고 조치한다.

2. 저장 객체 테스트

(1) Stored Function

(가) 함수의 반환값을 저장할 변수를 선언한다.

(예시) SQL> VAR salary NUMBER;

(나) EXECUTE 문을 이용해 함수를 실행한다.

(예시) SQL> EXECUTE:salary:= FC_update_sal(7369);

(다) SQL에서 선언된 변수의 출력은 PRINT문을 사용하고, 함수의 반환값을 저장한 변수값을 확인한다.

(예시) SQL> PRINT salary;

(2) Stored Procedure

(가) 실행하기 전 프로시저 실행 후 변경될 이전의 값을 확인한다.

(나) EXECUTE 문을 이용해 실행

(예시) SQL> set serveroutput on

SQL> execute update_sal(7369);

(다) 실행 이후 프로시저에서 처리하는 대로 관련 데이터가 수정된 것을 확인한다.

(3) Stored Package

(가) 패키지의 실행은 패키지명.프로시저(함수) 명으로 기술한다.

(나) DBMS_OUTPUT.PUT_LINE을 출력하기 위해 set serveroutput on을 실행한다.

SQL> SET SERVEROUTPUT ON ;

(다) 다음 예시와 같이 실행함으로써 수행 결과를 확인한다.

(예시) SQL> exec emp_info.all_emp_info;

(4) Trigger

(가) DBMS_OUTPUT.PUT_LINE을 출력하기 위해 set serveroutput on을 실행한다.

SQL> SET SERVEROUTPUT ON ;

(나) Trigger가 처리될 조건에 부합되는 SQL을 실행하여 데이터의 처리결과를 확인한다.

수행 내용 / 데이터 조작 프로시저 테스트

재료 · 자료

- ERD 작성표준, 개발 시 활용 데이터 저장소에 대한 매뉴얼 및 개발 가이드

기기(장비 · 공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 데이터저장소 설계 프로그램, 데이터베이스에서 제공하는 테스트 유틸리티 프로그램

안전 · 유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

작성한 데이터조작 프로시저를 테스트하는 방법은 단문 형태의 SQL과 절차형인 PL/SQL이 다른데, Oracle DBMS의 SQL은 1. SQL*Plus를 통해 SQL을 입력한 후 나타나는 처리 결과를 보며 테스트 할 수 있고, 2. PL/SQL은 테스트를 위해 블록내에 디버깅을 위한 환경을 코드형태로 작성한 후 실행함으로써 테스트를 하게 되는데 이 또한 SQL*Plus 상에서 관련 명령어를 입력하여 나타나는 내용으로 테스트 할 수 있다. 이를 위한 수행순서를 두가지 유형으로 나누어 살펴본다.

[1] 단문 형태의 SQL을 테스트한다.

1. 테스트 유형 적용 범위

   SQL(DDL, DML, DCL), TCL, SELECT문을 대상으로 한 테스트 방법이다.

2. SQL*Plus를 통한 테스트 순서

(1) Oracle DBMS를 설치하면 SQL*Plus 도구가 포함되어 설치된다.

(2) O/S 프롬프트 상태에서 Sqlplus uid/pwd를 입력하여 SQL*Plus 모드로 전환한다.

(3) 테스트하고자 하는 SQL(DDL, DML, DCL), TCL, SELECT 문장을 입력한다.

(4) SQL*PLUS 명령어로서 출력되는 내용을 파악하여 테스트한다.

2 절차형 SQL문을 테스트한다.

1. 테스트 유형 적용 범위

PL/SQL과 PL/SQL로 생성한 저장형 객체(Function, Procedure, Package, Trigger 등)를 대상으로 한 테스트 방법이다.

2. SQL*Plus를 통한 테스트 순서

(1) Oracle DBMS를 설치하면 SQL*Plus 도구가 포함되어 설치된다.

(2) O/S 프롬프트 상태에서 Sqlplus uid/pwd를 입력하여 SQL*Plus 모드로 전환한다.

(3) 테스트하고자 하는 PL/SQL 블록 내 소스코드에 디버깅을 위해 DBMS에서 제공되는 패키지 호출 로직을 추가하여 작성한다.

(4) SQL*PLUS 명령어인 EXECUTE 명령어를 통해 실행한 후, 처리결과로 출력되는 내용을 파악하여 테스트한다.

(5) 처리결과에 오류가 있다면 'SHOW ERRORS' 명령어를 통해 오류내용을 파악하여 조치한다.

수행 tip

- 작성된 단문SQL과 절차형 SQL을 테스트하기 위한 유틸리티 환경설정과, 처리 결과를 보고 오류를 수정할 수 있는지가 중요한 검증 포인트이다.

학습 3 교수 · 학습 방법

교수 방법

- 물리 데이터 모델을 기반으로 다양한 데이터베이스 오브젝트를 생성하고 조작하고 검색하는 데이터 조작어 전반을 이해하고, 활용할 수 있도록 PPT 자료로 제시한 후 설명한다.
- 물리 데이터 모델을 기반으로 데이터 조작 프로시저 작성에 필요한 SQL(DDL, DML, DCL) 문장을 이해하고, 활용할 수 있도록 PPT 자료로 제시한 후 설명한다.
- 물리 데이터 모델을 기반으로 데이터 조작 프로시저 작성에 필요한 트랜잭션 제어어(TCL) 문장을 이해하고, 활용할 수 있도록 PPT 자료로 제시한 후 설명한다.
- 물리 데이터 모델을 기반으로 데이터를 검색하는 SELECT 문장을 이해하고, 활용할 수 있도록 PPT 자료로 제시한 후 설명한다.
- 데이터 조작 결과에 대한 테스트를 위해 관련 유틸리티 환경설정과 결과를 이해할 수 있도록 관련 자료를 제시한 후 설명한다.

학습 방법

- 물리 데이터 모델을 기반으로 다양한 데이터베이스 오브젝트를 생성하고 조작하고 검색하는 데이터 조작어 전반을 이해하고, 습득한다.
- 물리 데이터 모델을 기반으로 데이터 조작 프로시저 작성에 필요한 SQL(DDL, DML, DCL) 문장을 이해하고, 습득한다.
- 물리 데이터 모델을 기반으로 데이터 조작 프로시저 작성에 필요한 트랜잭션 제어어(TCL) 문장을 이해하고, 습득한다.
- 물리 데이터 모델을 기반으로 데이터를 검색하는 SELECT 문장을 이해하고, 습득한다.
- 작성된 데이터 조작문을 테스트하기 위한 관련 유틸리티 활용법을 이해하고, 그 결과를 판단하여 정상 SQL로 조정해 나갈 수 있도록 실습한다.

학습 3 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.

- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준		
		상	중	하
데이터 조작 프로시저 작성	- 응용소프트웨어 설계와 물리 데이터저장소 설계에 따라 데이터 저장소에 연결을 수행하는 프로시저를 작성할 수 있다.			
	- 응용소프트웨어 설계와 물리 데이터저장소 설계에 따라 데이터 저장소로부터 데이터를 읽어 오는 프로시저를 작성할 수 있다.			
	- 응용소프트웨어 설계와 물리 데이터저장소 설계에 따라 데이터 변경 내용 또는 신규 입력된 데이터를 데이터 저장소에 저장하는 프로시저를 작성할 수 있다.			
데이터 조작 프로시저 테스트	- 구현된 데이터 조작 프로시저를 테스트할 수 있는 테스트 케이스를 작성하고, 단위 테스트를 수행하기 위한 테스트 조건을 명세화할 수 있다.			

평가 방법

- 문제해결 시나리오

학습내용	평가항목	성취수준		
		상	중	하
데이터 조작 프로시저 작성	- 데이터 조작을 위한 SQL 습득 정도			
데이터 조작 프로시저 테스트	- 데이터 조작 프로시저 테스트 환경 구축 - 테스트 결과 이해 및 데이터 조작문 조정			

- 구두발표

학습내용	평가항목	성취수준		
		상	중	하
데이터 조작 프로시저 작성	- 데이터 조작을 위한 SQL 습득 정도			
데이터 조작 프로시저 테스트	- 데이터 조작 프로시저 테스트 환경 구축 - 테스트 결과 이해 및 데이터 조작문 조정			

피 드 백

1. 문제해결 시나리오.
   - 문제해결을 위한 절차나 기법, 그리고 해결을 위한 접근방법의 적정성을 판단하여 개선해야 할 사항을 알려준다,
   - 물리 데이터 모델을 기반으로 다양한 데이터베이스 오브젝트를 생성하고 조작하고 검색하는 과정에서 발생한 문제의 구조 및 문장의 미비사항을 정리하여 돌려준다.
   - 데이터 조작 프로시저 테스트를 위한 환경구축 절차와 테스트 케이스 작성 방안의 미비사항을 정리하여 돌려준다.
2. 구두 발표
   - 발표한 내용 중 미비사항이나 보완해야 할 사항을 정리하여 돌려준다.
   - 데이터 조작 프로시저 테스트 케이스 작성 절차 및 결과에 대한 정확성을 판단하여 미비사항을 정리하여 돌려준다.

학습 1	논리 데이터저장소 확인하기(LM2001020205_14v2.1)
학습 2	물리 데이터저장소 설계하기(LM2001020205_14v2.2)
학습 3	데이터 조작 프로시저 작성하기(LM2001020205_14v2.3)

학습 4 데이터 조작 프로시저 최적화하기 (LM2001020205_14v2.4)

4-1. 데이터 조작 프로시저 성능개선

학습 목표

- 프로그래밍 언어와 도구에 대한 이해를 바탕으로 응용소프트웨어 설계, 물리 데이터 저장소 설계와 운영 환경을 고려하여 데이터 조작 프로시저의 성능을 예측할 수 있다.
- 업무 분석가에 의해 정의된 요구사항을 기준으로, 성능측정 도구를 활용하여 데이터 조작 프로시저의 성능을 측정할 수 있다.
- 실 데이터를 기반으로 테스트를 수행하여 데이터 조작 프로시저의 성능에 영향을 주는 병목을 파악할 수 있다.
- 테스트 결과와 정의된 요구사항을 기준으로 데이터조작 프로시저의 성능에 따른 이슈 발생 시 이에 대해 해결할 수 있다.

필요 지식 /

작성한 데이터 조작 프로시저를 적용하는 과정에서 성능개선이나 자원의 효율적 사용의 필요성이 제기되는바, 이를 해결하기 위해서는 성능 최적화 방안을 통해 업무의 중요도나 트랜잭션 빈도, 그리고 사용하는 사용자 수에 따라 우선순위를 부여한 뒤 우선순위가 높은 SQL부터 최적화를 진행하게 된다.

이런 성능 최적화는 무엇보다도 개발자 스스로가 최적화를 위한 필요지식을 학습하고 이해하여, 이를 습득함으로써 성능 최적화에 대한 부담을 줄이는 것이 무엇보다 우선적으로 필요하다.

관련 지식으로는 성능을 분석할 수 있는 1. APM(Application Performance Management)의 이해와, 2. 모니터링 결과 문제시되는 SQL에 대한 처리흐름, 그리고 3. 해당 SQL이 DBMS 내에서 어떻게 동작하는지를 파악할 수 있는 다양한 유틸리티(Oracle의 경우 TKPROF, EXPLAIN PLAN 등)를 활용하여 그 결과를 분석하는 방법 등이 있다.

1 APM(Application Performance Management) 도구의 이해

1. APM 정의

APM은 Application Performance Management(Monitoring)의 약자로서, 운영 중인 시스템에 대한 가용성 확보, 다운타임 최소화 등을 통해 안정적인 시스템 운영을 위하여, 부하량과 접속자 파악 및 장애진단 등을 목적으로 하는 성능 모니터링 도구를 말한다.

2. APM 유형

APM 유형은 크게 두가지(리소스, 엔드투엔드) 영역으로 나누어 정의해 볼 수 있다.

(1) 애플리케이션 수행시 리소스 모니터링

(가) 모니터링 대상 자원은 CPU, 메모리, 네트워크, 디스크 등이 있다.

(나) 대표적인 오픈 소스로는 Nagios, Zabbix, Cacti 등이 있다.

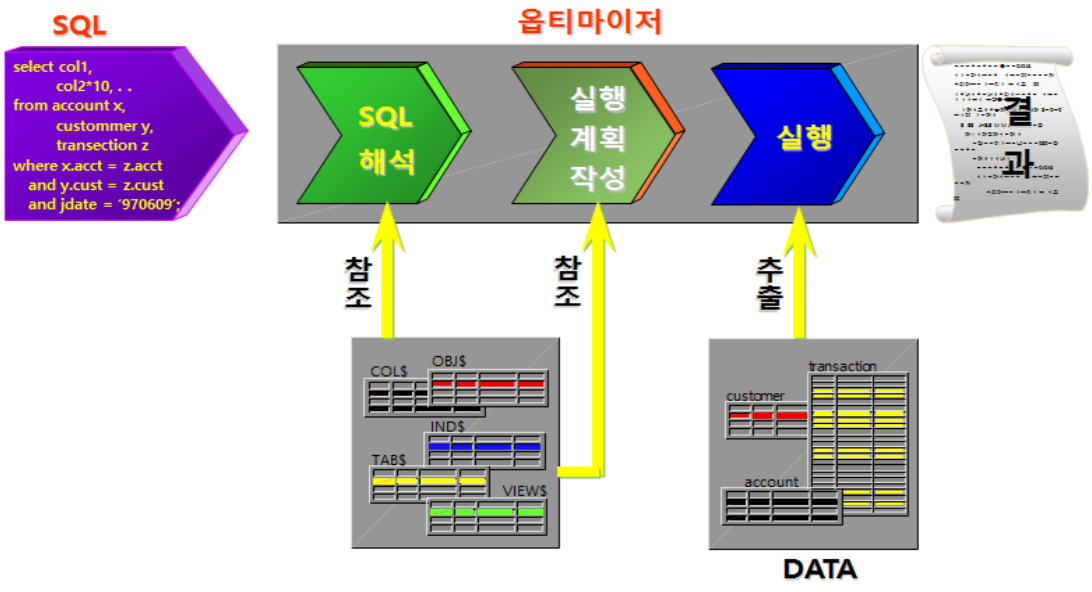
(2) 애플리케이션 수행을 위한 엔드투엔드(End to End) 모니터링

(가) 모니터링 대상을 애플리케이션 수행 관점으로 보아, 비즈니스 트랜잭션 관리 및 최종 사용자 등 엔드투엔드 모니터링으로 본다.

(나) 대표적인 오픈 소스로는 VisualVM이 있고, 상용 제품으로는 제니퍼, 파로스, 시스마스터 등이 있다.

2 SQL 처리 흐름

1. SQL 처리 흐름도



[그림 4-1] SQL 처리 흐름도

(1) 구분분석 단계

(가) 먼저, 사용자가 요청한 SQL문이 데이터베이스에서 처음 사용된 문장인지 이미 사용된 문장인지를 공유 풀 영역을 검색하여 확인한다. 이는 이미 사용된 문장이라면 구문분석(Parsing)이라는 작업을 할 필요가 없고, 처음 사용되었다면 정상적으로 구문분석 작업을 수행해야 하기 때문이다.

(나) 작성된 SQL문이 문법에 따라 정상적으로 작성되었는지를 분석하고, SQL 내에 포함된 테이블, 뷰 등이 데이터베이스에 존재하는 오브젝트인지를 확인한다.

(다) 이후, 옵티마이저는 SQL문을 가장 빠르게 데이터를 검색해 줄 수 있는 실행계획을 찾는다.

(2) 실행 단계

(가) 구문분석이 정상적으로 실행되면 서버 프로세스는 메모리 영역의 데이터베이스 버퍼 캐시영역을 검색하여 해당 테이블의 데이터가 다른 사용자의 다른 SQL문에 의해 이미 데이터버퍼 캐시영역에 존재하는지를 검색한다.

(나) 데이터버퍼 캐시영역에 존재한다면, 테이블의 해당 데이터 파일로부터 테이블을 읽지 않고 캐시영역의 데이터를 그대로 추출한다.

(다) 만약, 존재하지 않는다면 정의된 테이블의 해당 데이터 파일로부터 테이블을 읽어서 데이터버퍼 캐시영역에 저장한다.

(라) SQL문이 SELECT가 아닌 UPDATE, DELETE, INSERT문 등의 DML문장이었다면 데이터 버퍼캐시 영역에서 새로운 데이터로 변경, 삭제 또는 입력하게 된다.

(3) 추출 단계

(가) 실행단계가 끝나면 서버 프로세스는 데이터버퍼 캐시영역에서 관련 테이블 데이터를 읽어서 사용자가 요청한 클라이언트로 보내주게 된다.

(나) SELECT문을 실행하는 경우에만 추출단계가 실행되고, UPDATE, INSERT, DELETE문 실행 시는 추출단계는 실행되지 않는다.

2. SQL 작성시 고려사항

(1) 개발자는 SQL 특성을 충분히 이해하여 SQL문을 적절히 구사할 수 있는 능력을 기본적으로 갖추어야 한다.

(2) 사용자가 SQL 작성시, 옵티마이저가 실행계획을 수립한 후 실행되는 일련의 과정을 이해하고 작성하여야 한다.

(3) 구문분석 단계시 옵티마이저가 수립한 실행계획에 따라 엄청난 수행속도 차이가 발생할 수 있음을 이해하여야 한다.

(4) 특정 SQL이 실행될 때 옵티마이저에 의해 수립된 실행계획은 제어하기가 어렵지만, 옵티마이저가 비정상적으로 동작된다면, 이를 추적하여 개발자가 원하는 실행계획으로 동작될 수 있도록 조정하는 과정이 필요하다.

(5) 무엇보다도 개발자는 옵티마이저가 정상적인 실행계획을 수립할 수 있도록 종합적이고 전략적인 포인트를 SQL에 부여하여 작성하여야 한다.

(6) 좋은 SQL은, 추출되는 결과를 추론하여 SQL을 집합적으로 접근하여 작성하여야 한다.

3 SQL 성능 최적화를 위한 유틸리티 활용

시장에서 많이 사용하고 있는 Oracle DBMS 경우, SQL 문제점을 파악하고 개선하여 SQL 성능을 최적화하기 위해, TKPROF 및 EXPLAIN PLAN이라는 도구를 제공하고 있다.

만약 SQL문이 적절히 작성되지 않았다면 전반적인 처리 효율성이 떨어질 수 있고, 이때 처리 성능의 통계치 정보를 파악하기 위해 TKPROF 도구 활용을 고려해야 하고, EXPALIN PLAN은 SQL이 사용하는 액세스 경로를 파악하기 위해 활용할 수 있는 도구이다.

1. TKPROF 활용

실행되는 SQL문장에 대해 분석정보를 제공하여 사용자(프로그래머,..)가 특정 SQL문장을 어떻게 사용해야 할 것인지에 대한 가이드라인을 제공해 주는 도구로서, EXPLAIN PLAN과 병행하여 사용하는 것이 좋다.

(1) Trace 결과로 파악할 수 있는 분석정보 내용

(가) Parse, Execute, Fetch수

(나) CPU 시간/경과된 시간

(다) 물리적/논리적 Reads

(라) 처리된 로우수

(마) 라이브러리 캐시 Misses

(바) 파싱이 발생할 때의 사용자

(사) 커밋(Commit)/롤백(Rollback)

(2) Trace 유형

(가) Instance Level 추적

1) 지속적인 설정 방법이다.

2) 모든 SQL 수행에 대한 Trace 파일을 생성하여 많은 부하가 발생한다.

(나) Session Level 추적

1) 임시적인 설정 방법이다.

2) 특정 프로세스별로 추적 파일을 생성한다.

(3) Trace 유형 활용

(가) Instance Level로 모든 SQL을 Trace하는 경우는 거의 없고, DB 응용프로그램에서 사용되는 특정 SQL에 대해서만 Trace하는 Session Level을 일반적으로 활용한다.

(나) Trace를 Enable하는 것은 DB 부하가 수반되므로, 필요할때만 사용하고 평상시 개발 및 운영환경에서는 Disable 하는것이 좋다.

(4) Trace 관련 파라미터 설정 및 확인

(가) 관련 파라미터

1) timed_statistics (default=false): CPU시간, 실행시간등 시간에 관련된 정보를 표시하기 위해 사용됨

2) max_dump_file_size (default=500): Trace 파일의 최대 크기(단위: OS블럭수)

3) user_dump_dest: Trace 파일이 생성될 디렉토리

(나) 파라미터 설정 확인

1) Sys user로 DB connect: sqlplus "/ as sysdba"

2) SQL Mode에서 show parameter 또는 show parameter parameter_name

(5) TKPROF 활용법

SQL Trace가 생성한 Trace 파일을 분석하여 사용자가 읽을 수 있는 형태로 변환시켜 주는 Oracle 제공 도구이다.

(가) 적용 명령어

```
tkprof tracefile_name outputfile_name [sort=options] [print=n]
[explain=userid/passwd] [insert=filename] [sys=no] [aggregate=no]
[record=filename] [table=schema.tablename]
```

(나) 명령어 옵션

1) tracefile_name: *.trc file

2) Outputfile_name: 분석될 내용이 저장될 파일명으로 *.prf로 생성된다.

3) Sort

분석된 SQL문이 출력되는 순서를 결정해 주는 옵션으로, 지정한 파라미터 통계값에 대해 역순으로 출력하며, 하나 이상의 파라미터를 지정해 주려면 sort=(EXECPU, PRSCPU)와 같이 기술하면 된다. 이때 지정한 값을 서로 더하여 두값의 합이 가장 많은 SQL문부터 출력하게 된다.

- fchqry: 추출(Fetch)중 일관성 읽기를 위한 Buffer수
- execpu: 실행에 필요한 cpu 시간

- exedsk: 실행 중 디스크 읽기 횟수

4) Print=n: 처음 n문장을 출력함

5) Explain=iserid/passwd: Trace를 수행한 uid/pwd

6) Insert=filename: tkprof_table을 생성하고 Trace 정보를 저장하는 Script File 생성

7) Sys=no

- 재귀적(어떤 문장에 대해 오라클이 생성하고 실행하는 부가적인) SQL 문장을 보여주지 않는다.

- SQL 실행시 sys로 실행한 Data Dictionary 조회 등의 sql은 분석하지 않겠다는 옵션이다.

8) aggregate=no: 동일한 sql을 하나로 집계하지 않는 Option

9) Record=filename: Trace File에서 발견된 Query문만 저장하는 File

10) Table=schema.tablename: Plan_table명을 달리했을 때 그 Table을 지정

(다) 명령어 활용 예시

tkprof ora_219.trc 219.lst explain=scott/tiger

(라) 관련정보 파악 절차

1) Data Dictionary 변경

SQL> Analyze table table명 estimate(compute) statistics;

2) 환경변수 점검

SQL> show parameter parameter_name(특히 user_dump_dest)

3) 해당 SQL 과 관련된 Index 정보 파악

SQL> select * from user_ind_columns where table_name= 'emp' ;

(마) TKPROF 결과항목 의미

1) PARSE

SQL 구문 분석에서 발생하는 통계치로서 공유 SQL 영역에서 찾아온 것도 포함됨

2) EXECUTE

명령문을 실행하면서 발생하는 통계치로서 주로 INSERT, UPDATE, DELETE에 관련된 정보임

3) FETCH

추출 시에 발생하는 통계치로서 SELECT문이 실행되면서 추출된 통계치임

4) Count

- 문장이 분석되고 실행된 횟수

- Query가 Parse, Execute, Fetch한 횟수

5) CPU

- 각 처리 단계별 CPU소모 시간(초)
- Shared pool에서 문장 발견시 0임

6) Elapsed: 각 처리 단계의 시작에서 종료까지 총 경과 시간(초)

7) Disk: 각 처리 단계별 물리적인 디스크 블록을 읽은 횟수

8) Query

- 읽기 일관성(Consistent Read)과 관련된 것으로 각 처리 단계별로 읽은 변경된 버퍼 블록수
- 다른 Session이 올려놓은 메모리상의 데이터블럭의 Read Count

9) Current

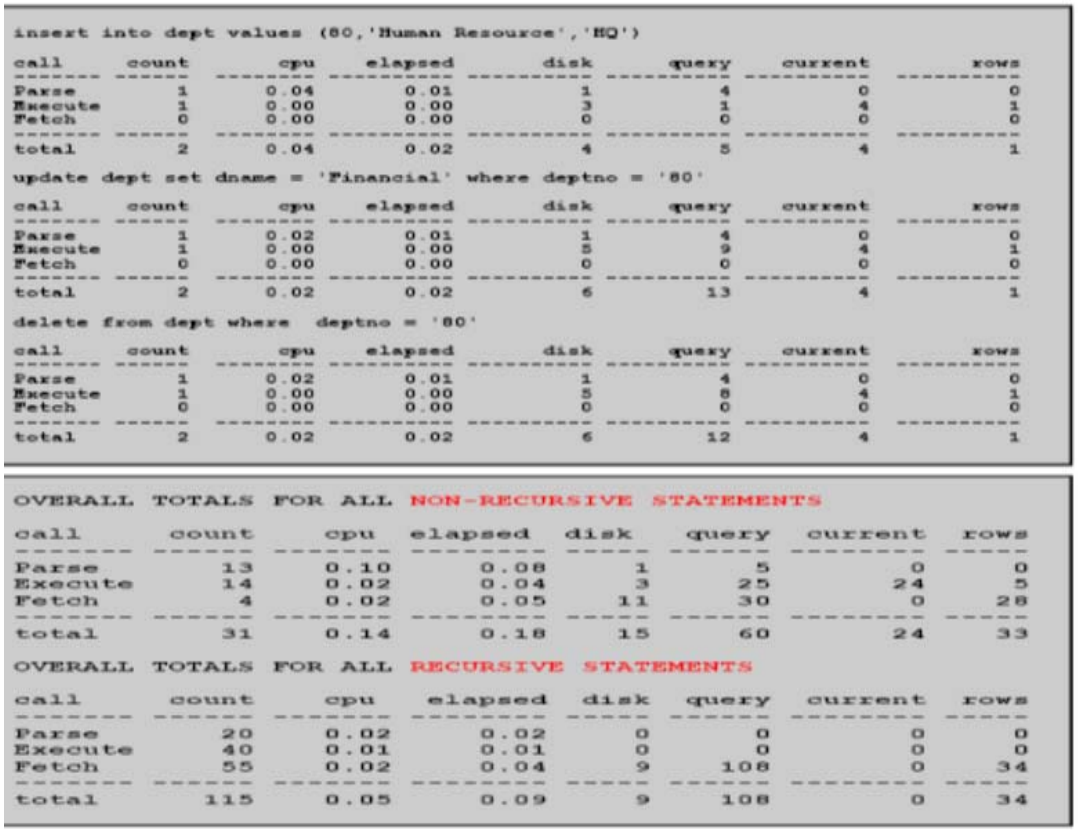
- 각 처리 단계별 현 세션에만 유효한 버퍼 블록을 접근한 수(Current Read)
- 주로 INSERT, UPDATE, DELETE 작업시 발생함
- 본인 session에서 올려놓은 메모리상의 Data block 수

10) Rows

각 처리 단계별 읽은 총 행수로서,

- Fetch 단계: SELECT에 의해 질의된 행수
- Execute 단계: INSERT, UPDATE, DELETE문에서 처리된 행수

(6) TKPROF 실행 후 결과 화면(예시)



```
insert into dept values (80,'Human Resource','HQ')

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.04       0.01          1          4          0           0
Execute      1      0.00       0.00          3          1          4           1
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.04       0.02          4          5          4           1

update dept set dname = 'Financial' where deptno = '80'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.02       0.01          1          4          0           0
Execute      1      0.00       0.00          5          9          4           1
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.02       0.02          6         13          4           1

delete from dept where  deptno = '80'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.02       0.01          1          4          0           0
Execute      1      0.00       0.00          5          8          4           1
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.02       0.02          6         12          4           1

OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       13      0.10       0.08          1          5          0           0
Execute     14      0.02       0.04          3         25         24           5
Fetch        4      0.02       0.05         11         30          0          28
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       31      0.14       0.18         15         60         24          33

OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse       20      0.02       0.02          0          0          0           0
Execute     40      0.01       0.01          0          0          0           0
Fetch       55      0.02       0.04          9        108          0          34
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      115      0.05       0.09          9        108          0          34
```

[그림 4-2] TKPROF 실행 후 결과 화면

2. EXPLAIN PLAN 활용

EXPLAIN PLAN은 사용자들이 SQL문의 액세스 경로를 확인하여 성능개선을 할 수 있도록 SQL문을 분석하고 해석하여 실행계획을 수립하고, 관련 테이블(plan_table)에 저장하도록 지원해 주는 도구이다.

(1) Explain Plan 준비

(가) 해당 사용자로 DB 접속하여 PLAN Table 생성

```
SQL> @$ORACLE_HOME/rdbms/admin/utlxplan
```

(나) PLUSTRACE Role 생성

1) Sys user 로 DB 접속: sqlplus "/ as sysdba"

2) PLUSTRACE role 생성 script 실행한다.

```
SQL> @ORACLE_HOME/sqlplus/admin/plustrce
```

3) 사용자에게 PLUSTRACE 권한을 부여한다.

```
SQL> grant plustrace to scott;
```

(2) Explain Plan 실행

(가) 해당 사용자로 DB 접속하여 Autotrace mode를 on으로 전환한다.

1) Autotrace 명령어를 사용하기 전에 반드시 PLAN_TABLE이 생성되어야 하고, PLUSTRACE 권한을 가지고 있어야 한다.

2) Autotrace Mode 전환(중지)

```
SQL> set autotrace on[off, traceonly]
```

- On: SQL문의 실행 결과와 실행계획과 통계정보를 보여 주는 옵션
- OFF: AUTOTRACE를 해지하는 옵션
- TRACEONLY: 실행계획과 통계정보만을 제공하는 옵션

3) Mode를 Off로 전환할 때까지 Plan_table에 write된다.

4) Autotrace Mode를 확인한다.

```
SQL> show autotrace
```

(나) Plan_Table을 확인한다.

1) 확인 예시

```
SQL> SELECT a.ename, a.sal, b.dname
        FROM emp a, dept b
        WHERE a.deptno = b.deptno;
```

2) 선택된 Row를 확인한다.

3) Plan 내용을 확인한다.

(다) PLAN_TABLE 결과 항목 의미

1) Recursive call: 재귀 호출의 횟수

2) DB block gets: 현재의 블록이 요구된 횟수

3) Consistent gets: 한 블록에 대해 요구된 Consistent Read 횟수

4) Physical reads: 디스크로부터 읽어 들인 데이터 블록의 총 개수

5) Redo size: Redo 로그가 만들어진 크기

6) Byte sent via SQL*Net to client: 클라이언트로 보내진 바이트 수

7) Byte received via SQL*Net from client: 클라이언트로부터 받은 바이트 수

8) Sort (Memory): 메모리에서 일어난 소트 수

9) Sort (Disk): 디스크에서 일어난 소트 수

10) Row processed: 연산을 하는 동안 처리한 row 수

(3) EXPLAIN PLAN & Autotrace 결과 화면(예시)

```
SQL> set autotrace on
SQL> select * from dept where deptno = 10;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK

Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'DEPT'

Statistics
----------------------------------------------------------
         178  recursive calls
           0  db block gets
          27  consistent gets
           7  physical reads
           0  redo size
         629  bytes sent via SQL*Net to client
         655  bytes received via SQL*Net from client
           2  SQL*Net roundtrips to/from client
           2  sorts (memory)
           0  sorts (disk)
           1  rows processed
```

[그림 4-3] EXPLAIN PLAN & Autotrace 결과 화면

수행 내용 / 데이터 조작 프로시저 성능개선하기

재료・자료

- ERD 작성표준, 개발 시 활용 데이터 저장소에 대한 매뉴얼 및 개발 가이드, 모델링 검토기준

기기(장비・공구)

- 컴퓨터, 인터넷, 형상관리 프로그램, 데이터저장소 설계 프로그램, 데이터 입출력 성능 측정 프로그램

안전・유의사항

- 실습 후에는 컴퓨터의 전원을 끈다.

수행 순서

데이터 입출력을 활용한 애플리케이션 개발 후, 개발된 전체적인 시스템 성능 최적화를 위해서는 시스템의 모든 구성요소(하드웨어, 시스템 소프트웨어, 네트워크, 애플리케이션, SQL 등 포함)를 대상으로 최적화 절차를 수행하게 되는데, 특히 개발자 관점에서는 애플리케이션과 SQL 영역에 집중하여 성능개선 작업을 진행하게 된다.

따라서, 이번 장에서는 개발된 시스템의 성능 최적화를 위한 전반적인 절차를 먼저 파악해 보고, 개발자가 집중해야 할 영역 중 본 학습모듈 대상영역을 고려하여 SQL 성능 최적화를 위한 수행순서를 중심으로 살펴본다.

[1] 시스템 성능 개선영역 및 절차를 이해한다.

일반적으로 시스템 성능은 애플리케이션만을 대상으로 수행하는 것이 아닌 시스템을 구성하는 모든 구성요소를 대상으로 하여야만 한다. 그럼으로써 보다 효율적이고, 상대적인 성능개선 소요 시간 또한 줄일 수 있음을 고려하여, 주요한 성능개선 대상 영역과 수행 주체를 알아보면 다음과 같다.

1. 인프라 성능개선을 통해 시스템 성능을 개선할 수 있도록 한다.

(1) 시스템 구성 최적화: 시스템 관리자

(가) 웹서버/애플리케이션서버/DB서버 구성 분리

(나) 커널 파라미터

(다) 디스크 컨트롤러 구성 등

(2) 운영체제 최적화: 시스템 관리자

(가) 시스템 자원: CPU, 메모리 등

(나) 커널 파라미터

(다) 디스크 구성 기법: Raid 수준 등

(3) 미들웨어 최적화: 미들웨어 관리자

(가) Instance 개수

(나) Instance별 환경: JVM(Heap size 등), GC(Garbage Collection) 주기

(다) DB 연결 수: ConnectionPool 등

(2) 네트워크 최적화: 네트워크 관리자

2. 데이터베이스 성능개선을 통해 시스템 성능을 개선할 수 있도록 한다.

(1) 데이터모델 최적화: 데이터모델러

(2) 데이터베이스 최적화: DBA

(가) 구성: 공유 메모리(Oracle의 경우 SGA), 초기 파라미터

(나) 물리적 구조 최적화

(다) 메모리 할당 최적화

(라) I/O 분산 최적화

(마) 메모리 경합 최적화

3. 업무 영역 성능개선을 통해 시스템 성능을 개선할 수 있도록 한다.

(1) 업무기능 최적화: 비즈니스 아키텍트, 업무 분석가

(2) 업무프로세스 설계 최적화: 업무 설계자

4. 애플리케이션 성능개선을 통해 시스템 성능을 개선할 수 있도록 한다.

(1) 언어(Java 또는 C) 최적화: 개발자

(가) 메모리 효율적 사용

(나) 캐싱(Caching), 락킹(Locking), 루핑(Looping) 등

(다) 예외(Exception) 처리

(2) 알고리즘: 개발자

5. SQL 성능개선을 통해 시스템 성능을 개선할 수 있도록 한다.

(1) SQL 성능튜닝: 개발자

(2) 인덱스 조정: 개발자/DBA

2 SQL 성능개선 순서를 이해한다.

1. 문제 있는 SQL 식별

(1) 문제 있는 SQL을 식별하기 위해서는, 애플리케이션의 성능을 관리하거나 모니터링하기 위한 툴인 APM(Application Performance Management) 등을 활용한다.

(2) Oracle의 경우, TKPROF 또는 SQL_Trace와 같은 유틸리티를 사용하여 성능에 문제가 있는 SQL을 확인한다.

2. 옵티마이저(Optimizer) 통계 확인

옵티마이저(Optimizer)는 개발자가 작성한 SQL을 가장 빠르고 효율적으로 수행할 최적의 처리경로를 생성해 주는 데이터베이스 핵심모듈로서, Oracle은 CBO(Cost Based Optimizer)와 RBO(Rule Based Optimizer) 모드를 지원하고, 이중 비용기반인 CBO 모드를 기본으로 지원하고 있다.

비용기반 옵티마이저 모드에서 최적의 처리경로를 생성하기 위해서는 옵티마이저가 활용하는 통계정보를 주기적으로 현행화하여야 하는데 이를 위한 문장은 다음과 같다.

```
Analyze Object_type Object_name Operation STATISTICS;
```

(1) Object_name: TABLE, INDEX, CLUSTER 중 선택하여 기술한다.

(2) Operation

(가) COMPUTE

정의된 Object_name에 대하여 통계정보를 정확하게 계산하는 방법으로, 가장 정확한 통계를 얻을 수 있지만 처리 속도는 가장 느리다.

(나) ESTIMATE

데이터 딕셔너리의 값과 데이터 샘플링 정보를 기반으로 통계치를 예상하는 방법으로, COMPUTE보다 덜 정확하지만 처리속도가 훨씬 빠르다.

(다) DELETE

정의된 Object_name에 대한 모든 통계 정보를 삭제 한다.

(라) 예시: ANALYZE TABLE emp COMPUTE STATISTICS ;

3. 실행계획 검토

Driving 테이블이 최상의 필터를 가지고 있는지를 중심으로 검토한다. 즉 처리량이 작은 Table을 Driving 테이블로 지정되었는지 확인하다.

4. SQL 문 재구성

(1) 가능한 한 where = 을 많이 써서 범위가 아닌 특정 값 지정으로 인한 범위를 줄여 처리속도가 빠르도록 한다.

(2) where 절의 칼럼에 연산자를 사용하여 칼럼변경이 발생하면 인덱스를 활용하지 못하게 됨을 이해하여 칼럼 변경 연산자를 쓰지 않도록 한다.

(3) 부분범위 처리의 경우 Sub-Query에 Exists 사용하여 불필요한 검색을 하지 않도록 한다.

(4) 옵티마이저가 비정상적인 실행계획을 수립하여 처리된다면, 힌트로서 옵티마이저의 액세스 경로 및 조인 순서를 제어할 수 있도록 한다.

5. 인덱스 재구성

(1) 성능에 중요한 액세스 경로를 고려하여 인덱스화한다.

(2) 실행계획을 검토하여 기존 인덱스의 열 순서를 변경하거나 추가 할 수가 있도록 한다.

(3) 인덱스 추가 시 정상적으로 처리되고 있던 다른 SQL에 심각한 영향을 줄 수 있으므로 관련된 주요 SQL 질의결과를 함께 검토하여야 한다.

(4) 한가지 인덱스로 읽기만 하는 코드와 같은 테이블은 Index-Organized Table을 고려한다.

(5) 사용하지 않는 불필요한 인덱스들은 제거한다.

6. 실행계획 유지관리

데이터베이스 버전 업그레이드나, 데이터의 시스템 이동 등 시스템 환경의 변경 사항 발생시에도 실행계획이 유지되고 있는지 모니터링하고 관리한다.

3 TKPROF 수행 순서를 이해한다.

1. 수행하고 하는 DB User로 Connect하여 Trace Mode를 설정

```
SQL> alter session set sql_trace=true;
SQL> alter session set timed_statistics=true;
```

2. Trace 하고자 하는 SQL 실행 후 exit

3. USER_DUMP_DEST dir의 Trace File 확인: ls -alt

4. Trace File 분석 utility(tkprof)를 활용하여 분석

```
tkprof xxx.trc result.lst sys=no explain=uid/pwd
```

5. 최종 파일(result.lst)을 열어서 결과분석

수행 tip

- 개발된 데이터 조작 프로시저의 성능개선 절차와 데이터 입출력 성능 측정을 위한 프로그램(유틸리티)사용법을 통해 문제 있는 SQL을 식별하고 조정함으로써 성능개선 작업을 수행할 수 있는지가 중요한 검증 포인트이다.

학습 4 교수 · 학습 방법

교수 방법

- 데이터 조작 프로시저가 데이터베이스에서 어떻게 동작하고 수행되는지에 대한 원리를 이해하고, 성능개선 절차와 유틸리티를 활용하여 성능을 개선해 가는 내용을 PPT 자료로 제시한 후 설명한다.
- 데이터베이스 조작 프로시저 성능문제 발생 시, 성능개선을 위한 인프라, 데이터베이스, 업무, 애플리케이션, 그리고 SQL 성능개선 영역의 항목에 대해 전체적인 관점을 도식하며 설명한다.
- 성능 진단을 위한 관련 유틸리티 환경 설정 방법과 실행 후 결과물을 이해할 수 있는 자료를 제시한 후 설명한다.
- SQL 성능 개선을 위한 세부 절차에 대해 각 단계별 사례를 제시해 가며 설명한다.
- 문제 있는 SQL을 식별하고, 해당 SQL이 데이터베이스 내에서 어떻게 동작하는지를 판단하여 이를 개선할 수 있는 SQL로 전환할 수 있도록 관련 자료를 제시한 후 설명한다.

학습 방법

- 데이터베이스 조작 프로시저 성능문제 발생 시, 성능개선을 위한 인프라, 데이터베이스, 업무, 애플리케이션, 그리고 SQL 성능개선 영역의 항목에 대해 이해함으로써, 전체적인 관점에서 성능개선 절차를 숙지하여 활용할 수 있도록 학습한다.
- SQL 동작과정을 이해할 수 있고, 유틸리티 활용법에 대해 이해한다.
- 성능진단 및 개선을 위한 관련 유틸리티 활용법을 이해하고 실습한다.
- SQL 성능 개선을 위한 세부 절차에 대해 각 단계별 사례에 대해 이해하고 학습한다.
- 개발자 수준에서 문제 있는 SQL을 성능이 보장될 수 있는 SQL로 전환하는 방법을 이해하고 실습한다.

학습 4 평 가

평가 준거

- 평가자는 학습자가 수행 준거 및 평가 시 고려사항에 제시되어 있는 내용을 성공적으로 수행하였는지를 평가해야 한다.

- 평가자는 다음 사항을 평가한다.

학습내용	평가항목	성취수준 상	중	하
데이터조작 프로시저 성능개선	- 프로그래밍 언어와 도구에 대한 이해를 바탕으로 응용소프트웨어 설계, 물리 데이터저장소 설계와 운영 환경을 고려하여 데이터 조작 프로시저의 성능을 예측할 수 있다.			
	- 업무 분석가에 의해 정의된 요구사항을 기준으로, 성능측정 도구를 활용하여 데이터 조작 프로시저의 성능을 측정할 수 있다.			
	- 실 데이터를 기반으로 테스트를 수행하여 데이터 조작 프로시저의 성능에 영향을 주는 병목을 파악할 수 있다.			
	- 테스트 결과와 정의된 요구사항을 기준으로 데이터 조작 프로시저의 성능에 따른 이슈 발생 시 이에 대해 해결할 수 있다.			

평가 방법

- 문제해결 시나리오

학습내용	평가항목	성취수준 상	중	하
데이터조작 프로시저 성능개선	- 성능개선을 위한 문제해결 절차의 이해			
	- 성능진단을 위한 유틸리티 사용법 및 결과물 이해			
	- 애플리케이션 주요 메커니즘 및 적용 개발언어 분석 능력			

- 구두발표

학습내용	평가항목	성취수준 상	중	하
데이터조작 프로시저 성능개선	- 데이터 조작 프로그래밍 언어의 이해			
	- 성능진단을 위한 유틸리티 사용법 및 결과물 이해			
	- 애플리케이션 주요 메커니즘 및 적용 개발언어 분석 능력			

피 드 백

1. 문제해결 시나리오.
   - 문제해결을 위한 절차나 기법, 그리고 해결을 위한 접근방법의 적정성을 판단하여 개선해야 할 사항을 알려준다,
   - 데이터베이스 조작 프로시저 성능문제 발생시, 성능개선을 위한 인프라, 데이터베이스, 업무, 애플리케이션, 그리고 SQL 성능개선 영역의 항목에 대해 상세히 알려준다.
   - 특히 SQL 영역의 성능개선을 위해 활용가능한 tkprof 유틸리티 사용절차에 대해 알려줌으로써 성능개선을 위한 시나리오를 수립할 수 있도록 한다.
2. 구두 발표
   - 발표한 내용 중 미비사항이나 보완해야 할 사항을 정리하여 돌려준다.
   - 데이터베이스 조작 프로시저 성능문제 해결을 위한 전반적인 절차나 접근 방법에 대해 미비점이나 보완사항을 정리하여 돌려준다.

참고자료

- 한국전산원(2005). 『정보시스템 구축운영 기술 가이드라인』.
- 한국정보화진흥원(2009.5.28). 『정보시스템 감리지침』.
- 박현철외8명(2005). 『UML 이해와 실제』, 한국소프트웨어컴포넌트컨소시엄.

NCS 학습모듈 개발진

(대표집필자)

강석진(이비스톰)

(집필진)

김보운(이화여자대학교)
김홍진(LG CNS)
유은희
장현섭((주)커리텍)
주선태(T3Q)
진권기(이비스톰)
최재준

(검토진)

김승현(경희대학교)
엄기영(우리에프아이에스)
장온순(한국IT컨설팅)
조상욱(세종대학교)
조성호(삼성카드)

(개발기관)

최기원(한국소프트웨어기술진흥협회)
이두현(한국소프트웨어기술진흥협회)

(연구기관)

옥준필(한국직업능력개발원)
김상진(한국직업능력개발원)
김성남(한국직업능력개발원)
김지영(한국직업능력개발원)
문한나(한국직업능력개발원)
홍서희(한국직업능력개발원)

*표시는 NCS 개발진임

www.ncs.go.kr