3과목 〈데이터베이스 구축〉

1장 논리 데이터베이스 설계

섹션 76 데이터 베이스 설계 ★

: 데이터베이스 설계란 사용자의 요구를 분석하여 그것들을 컴퓨터에 저장할 수 있는 데이터베이스 구조에 맞게 변형한 후 특정 DBMS로 데이터베이스를 구현하여 일반 사용자들이 사용하게 하는 것

데이터베이스 설계 시 고려 사항
- 무결성 : 삽입, 삭제, 갱신 등의 연산 후에도 데이터베이스에 저장된 데이터가 정해진 제약 조건을 항상 만족해야 한다.
- 일관성 : 데이터베이스에 저장된 데이터들 사이나, 특정 질의에 대한 응답이 처음부터 끝까지 변함없이 일정해야 한다.
- 회복 : 시스템에 장애가 발생했을 때 장애 방생 직전의 상태로 복구할 수 있어야 한다.
- 효율성 : 응답시간의 단축, 시스템의 안정성, 저장 공간의 최적화 등이 가능해야 한다.
- 데이터베이스 확장 : 데이터베이스 운영에 영향을 주지 않으면서 지속적으로 데이터를 추가할 수 있어야 한다.

데이터베이스 설계 순서
1. 요구 조건 분석 : 요구 조건 명세서 작성
2. 개념적 설계 : 개념 스키마, 트랜잭션 모델링, E-R 모델
3. 논리적 설계 : 목표 DBMS에 맞는 논리 스키마 설계,트랜잭션 인터페이스 설계
4. 물리적 설계 : 목표 DBMS에 맞는 물리적 구조의 데이터로 변환
5. 구현 : 목표 DBMS의 DDL로 데이터베이스 생성, 트랜잭션 생성

요구조건 분석 : DB를 사용할 사용자들로부터 필요한 용도를 파악하는 것
- 사용자에 따른 수행 업무와 필요한 데이터의 종류, 용도 등을 수집
- 수집된 조건으로 요구 조건 명세서 작성

개념적 설계(정보 모델링, 개념화)★
: 정보의 구조를 얻기 위하여 현실 세계의 무한성과 계속성을 이해하고, 다른 사람과 통신하기 위하여 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정
- 이 단계에서는 개념 스키마 모델링과 트랜잭션 모델링을 병행 수행한다.
- 요구 분석 단계에서 나온 결과인 요구 조건 명세를 DBMS에 독립적인 E-R 다이어그램으로 작성한다.
- DBMS에 독립적인 개념스키마를 설계한다.

논리적 설계(데이터 모델링)★
: 현실 세계에서 발생하는 자료를 컴퓨터가 이해하고 처리할 수 있는 물리적 저장 장치에 저장할 수 있도록 변환하기 위해 특정 DBMS가 지원하는 논리적 자료 구조로 변환(mapping)시키는 과정이다.
- 개념 세계의 데이터를 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계로 표현되는 논리적 구조의 데이터로 모델화한다.
- 개념적 설계가 개념 스키마를 설계하는 단계라면 논리적 설계에서는 개념 스키마를 평가 및 정제하고 DBMS에 따라 서로 다른 논리적 스키마를 설계하는 단계
- 트랜잭션의 인터페이스를 설계한다.
- 관계형 데이터베이스라면 테이블을 설계하는 단계이다.

물리적 설계(데이터 구조화) ★
: 논리적 설계 단계에서 논리적 구조로 표현된 데이터를 디스크 등의 물리적 저장 장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정이다.
- 물리적 설계 단계에서ㅏ는 다양한 데이터베이스 응용에 대해 처리 성능을 얻기 위해 데이터베이스 파일의 저장 구조 및 액세스 경로를 결정한다.
- 저장 레코드의 양식, 순서, 접근 경로, 조회가 집중되는 레코드와 같은 정보를 사용하여 데이터가 컴퓨터에 저장되는 방법을 묘사한다.
- **물리적 설계 시 고려 사항 :** 트랜잭션 처리량, 응답시간, 디스크 용량, 저장 공간의 효율화 등

데이터베이스 구현
: 논리적/물리적 설계 단계에서 도출된 데이터베이스 스키마를 파일로 생성한는 과정이다.

- 사용하려는 특정 DBMS의 DDL을 이용하여 DB 스키마를 기술한 후 컴파일 하여 빈 데이터베이스 파일을 생성한다
- 생성된 빈 데이터베이스 파일에 데이터를 입력한다.
- 데이터베이스 접근을 위한 응용 프로그램을 작성한다.

섹션 77 데이터 모델의 개념

데이터 모델 : 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형

- 데이터 모델은 데이터, 데이터의 관계,데이터 의미 및 일관성, 제약 조건 등을 기술하기 위한 개념적 도구들의 모임
- 현실 세계를 데이터베이스에 표현하는 중간과정, 즉 데이터베이스 설계 과정에서 데이터의 구조(Schema)를 논리적으로 표현하기 위해 사용되는 지능적 도구

데이터 모델의 구성 요소 : 개체, 속성, 관계
데이터 모델의 종류 : 개념적 / 논리적 / 물리적 데이터 모델
데이터 모델에 표시할 요소 : 구조, 연산, 제약 조건

데이터 모델의 구성 요소
개체 (Entity) : 데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체
속성 (Attribute) : 데이터의 가장 작은 논리적 단위로서 파일 구조상의 데이터 항목 또는 데이터 필드에 해당
관계 (Relationsip) : 개체 간의 관계 또는 속성 간의 논리적인 연결

개념적 데이터 모델
: 현실 세계에 대한 인간의 이해를 돕기 위해 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정

- 개념적 데이터 모델은 **속성들**로 기술된 개체 타입과 이 개체 타입들 간의 관계를 이용하여 현실 세계를 표현한다.
- 현실 세계에 존재하는 개체를 인간이 이해할 수 있는 정보 구조로 표현하기 때문에 정보 모델이라고도 한다. 대표적으로 **E-R 모델**이 있다

논리적 데이터 모델
: 개념적 데이터 모델링 과정에서 얻은 개념적 구조를 컴퓨터가 이해하고 처리할 수 있는 컴퓨터 세계의 환경에 맞도록 변환하는 과정을 말한다.

- 논리적 데이터 모델은 **필드**로 기술된 데이터 타입과 이 데이터 타입들 간의 관계를 이용하여 현실 세계를 표현한다
- 단순히 데이터 모델이라고 하면 논리적 데이터 모델을 의미한다
- 특정 DBMS는 특정 논리적 데이터 모델 하나만 선정하여 사용한다
- 논리적 데이터 모델은 데이터 간의 관계를 어떻게 표현하느냐에 따라 관계 모델, 계층 모델, 네트워크 모델로 구분한다.

논리적 데이터 모델의 품질 검증 p337
: 완성된 논리 데이터 모델이 기업에 적합한지 확인하기 위해 품질 검증

- 개체 품질 / 속성 품질 / 관계 품질 / 식별자 품질 / 전반적인 품질 검증 항목을 통해 확인한다.

데이터 모델에 표시할 요소

1. **구조 (Structure) :** 논리적으로 표현된 개체 타입 들간의 관계로서 데이터 구조 및 정적 성질을 표현한다.
2. **연산 (Operation)** : 데이터베이스에 저장된 실제 데이터를 처리하는 작업에 대한 명세서로서 데이터베이스를 조작하는 기본 도구이다.
3. **제약 조건 (Constraint)** : 데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약 조건

섹션 78 데이터 모델의 구성 요소 - 개체(Entity) (C등급)

: 개체는 데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체이다.

- 개체는 실세계에 독립적으로 존재하는 유형, 무형의 정보러서 서로 연관된 몇 개의 속성으로 구성된다. (예〉 취미, 음악 등)
- 파일 시스템의 레코드에 대응하는 것으로 어떤 정보를 제공하는 역할을 수행
- 영속적(Persistence)으로 존재하는 개체의 집합
- 독립적으로 존재하거나 그 자체로도 구별이 가능하다
- 유일한 식별자(Unique Identifier)에 의해 식별이 가능하다

- 개체는 업무 프로세스에 의해 이용된다
- 다른 개체와 하나 이상의 관계가 존재한다

개체 선정 방법
- 업무 기술서 이용. 실제 업무 담당자와 인터뷰
- 자료 흐름도를 통해 업무 분석을 수행한 경우 자료 흐름도의 자료 저장소를 확인
- BPR(업무 프로세스 재설계)에 의해 업무를 재정의 한 경우 관련 개체 찾기 등

개체명 지정 방법
- 해당 업무에서 사용하는 용어 사용
- 약어는 되도록 제한
- 가능한 단수 명사
- 모든 개체명 유일
- 가능하면 개체가 생성되는 의미에 따라 이름 부여

섹션 79 데이터 모델의 구성 요소 - 속성(Attribute) (C등급)

속성은 데이터베이스를 구성하는 가장 작은 논리적 단위이다
- 파일 구조상의 데이터 항목 또는 데이터 필드에 해당한다.
- 속성은 개체를 구성하는 항목이다
- 속성은 개체의 특성을 기술한다
- 속성의 수를 디그리 혹은 차수라 한다.

속성의 종류: 속성의 특성과 개체 구성 방식에 따라 분류할 수 있다.

특성으로 구분
1. 기본 속성 : 업무 분석을 통해 정의한 속성. 가장 많음
2. 설계 속성 : 원래 업무상 존재하지 않고 설계 과정에서 도출해내는 속성
   - 업무에 필요한 데이터 외에 데이터 모델링을 위해 업무를 규칙화하고 속성을 새로 만들거나 변형시켜 정의하는 속성
3. 파생 속성 : 다른 속성으로부터 계산이나 변형 등의 영향을 받아 발생하는 속성
   - 파생 속성은 가능한 적은 수로 정의하는 것이 좋다.

개체 구성 방식으로 구분
1. 기본키 속성 : 개체를 식별할 수 있는 속성
2. 외래키 속성 : 다른 개체와의 관계에서 포함된 속성
3. 일반 속성 : 개체에 포함되어 있고 기본키, 외래키에 포함되지 않은 속성

속성 후보 원칙 - 원시 속성으로 판단되는 후보 속성은 버리지 않음
- 소그룹별로 속성 후보군을 만들고 가장 근접한 개체에 할당

속성명 지정 원칙 : 웹이나 클라/서버 등 어떠한 환경에서든 사용자 인터페이스에 나타나므로 정확하고 혼란이 없어야함
- 해당 업무에서 사용하는 용어로 지정
- 서술형으로 지정하지 않는다
- 가급적 약어는 쓰지 않는다
- 개체명은 속성명으로 할 수 없다
- 개체에서 유일하게 식별 가능하도록 지정한다

섹션 80 데이터 모델의 구성 요소 - 관계(Relationship) (C등급)

: 관계는 개체와 개체 사이의 논리적인 연결을 의미한다
관계의 형태
: 일 대 일 / 일 대 다 / 다 대 다

업무의 형태에 따라 나뉨
1. 종속 관계(Dependent) : 두 개체 사이의 주·종 관계를 표현한 것으로 식별 관계와 비식별 관계가 있다.
2. 중복 관계(Redundant Relationship) : 두 개체 사이에 2번 이상의 종속 관계가 발생하는 관계
3. 재귀 관계 (Recursive) : 개체가 자기 자신과 관계를 갖는 것으로, 순환 관계라고도 한다.
4. 배타 관계 (Exclusive) : 개체의 속성이나 구분자를 기준으로 개체의 특성을 분할하는 관계, 배타 And 와 배타 Or 관계로 구분한다
   - 배타 And 는 하위 개체들 중 속성이나 구분자 조건에 따라 하나의 개체만 선택할 수 있고, 배타Or은 하나 이상의 개체를 선택할 수 있다.

※식별 관계와 비식별 관계

식별 관계 : 개체 A,B 사이의 관계에서 A 개체의 기본키가 B 개체의 외래키이면서 동시에 기본키가 되는 관계를 말한다.

- B 개체의 존재 여부가 A 개체의 존재 여부에 의존적인 경우 발생
- ER 도형에서 식별 관계는 실선으로 표시

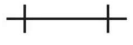
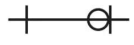
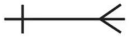
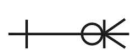
비식별 관계 : 개체 A,B 사이의 관계에서 A 개체의 기본키가 B 개체의 비키본키 영역에서 외래키가 되는 관계를 말한다

- B 개체의 존재 여부가 A 개체의 존재 여부에 관계없이 존재
- 일반적으로 두 개체는 비식별 관계로 존하는 경우가 많다
- ER 도형에서 비식별관계는 점선으로 표시

관계 표기 기호

기호	의미
|	필수(Mandatory)
O	선택적(Optional)
<	다중(Multiple)

관계의 표현

관계	표현	의미
1:1		양쪽에 반드시 1개씩 존재
1:0 또는 1:1		왼쪽에는 반드시 1개, 오른쪽에는 없거나 1개 존재
1:N		왼쪽에는 반드시 1개, 오른쪽에는 반드시 여러 개 존재
1:1 또는 1:N		왼쪽에는 반드시 1개, 오른쪽에는 1개 또는 여러 개 존재
1:0 또는 1:1 또는 1:N		왼쪽에는 반드시 1개, 오른쪽에는 0개 또는 1개 또는 여러 개 존재

섹션 81 식별자(Identifier) (D등급)

하나의 개체 내에서 각각의 인스턴스를 유일하게 구분할 수 있는 구분자로, 모든 개체는 한 개 이상의 식별자를 반드시 가져야한다.

- 식별자는 개체 내에서 대표성 여부, 스스로 생성 여부, 단일 속성 여부, 대체 여부에 따라 다음과 같이 구분한다.

분류	식별자
대표성 여부	주 식별자(Primary Identifier), 보조 식별자(Alternate Identifier)
스스로 생성 여부	내부 식별자(Internal Identifier), 외부 식별자(Foreign Identifier)
단일 속성 여부	단일 식별자(Single Identifier), 복합 식별자(Composit Identifier)
대체 여부	원조 식별자(Original Identifier), 대리 식별자(Surrogate Identifier)

주 식별자 / 보조 식별자

- 주 식별자는 개체를 대표하는 유일한 식별자
- 보조 식별자는 주 식별자를 대신하여 개체를 식별할 수 있는 속성
- 두 식별자 모두 개체를 유일하게 식별할 수 있어야 함
- 한 개체에 주 식별자는 하나만 존재하지만 보조 식별자는 한 개 이상 존재
- 개체를 유일하게 식별할 수 있는 속성이 2개 이상인 경우 업무에 가장 적합한 것을 주 식별자로 정하고, 나머지를 보조 식별자로 지정
- 물리적 테이블에서 주 식별자는 기본키로, 보조 식별자는 유니크 인덱스로 지정되어 사용됨\

(주 식별자의 4가지 특징) : 유일성, 최소성, 불변성, 존재성

내부 식별자 / 외부 식별자

- 내부 식별자는 개체 내에서 스스로 만들어지는 식별자
- 외;부 식별자는 다른 개체와의 관계에 의해 외부 개체의 식별자를 가져와 사용하는 식별자
- 외부 식별자는 자신의 개체에서 다른 개체를 찾아가는 연결자 역할을 함

단일 식별자 / 복합 식별자

- 단일 식별자는 주 식별자가 **한 가지 속성**으로만 구성된 식별자
- 복합 식별자는 주 식별자가 두 개 이상의 **속성**으로 이루어진 식별자

원조 식별자 / 대리 식별자

- 원조 식별자는 업무에 의해 만들어지는 가공되지 않은 원래의 식별자. 본질 식별자라고도 한다
- 대리 식별자는 주 식별자의 속성이 2개 이상인 경우 속성들을 하나의 속성으로

묶어 사용하는 식별자로, 인조 식별자라고도 한다

※대리 식별자의 조건
- 최대한 범용적인 값 사용
- 유일한 값을 만들기 위한 대리 식별자 사용
- 하나의 대리 식별자 속성으로 대체할 수 없는 경우 주의
- 편의성과 단순성, 의미 체계화를 위한 대리 식별자 사용 가능
- 시스템적인 필요성에 의해 내부적으로만 사용하는 대리 식별자 사용 가능

※※후보 식별자 : 개체에서 각 인스턴스를 유일하게 식별할 수 있는 속성 또는 속성 집합들을 의미
- 하나의 개체에는 한 개 이상의 후보 식별자가 있고, 이 중 개체의 대표성을 나타내는 식별자는 주 식별자로, 나머지는 보조 식별자로 지정

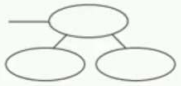
주의 사항
- 각 인스턴스를 유일하게 식별할 수 있어야 한다
- 속성들을 직접 식별할 수 있어야 한다
- 널 값이 될 수 없다
- 속성 집합은 후보 식별자로 지정한 경우 개념적으로 유일해야한다
- 후보 식별자의 데이터는 자주 변경되지 않아야 한다

섹션 82 E-R (개체-관계) 모델

: 개념적 데이터 모델의 가장 대표적인 것으로, 1976년 피터 첸에 의해 제안되고 기본적인 구성 요소가 정립
- E-R 모델은 개체와 개체 간의 관계를 기본 요소로 이용하여 현실 세계의 무질서한 데이터를 개념적인 논리 데이터로 표현하기 위한 방법으로 많이 사용되고 있다
- E-R 모델은 개체 타입과 이들 간의 관계 타입을 이용해 현실 세계를 개념적으로 표현
- E-R 모델에서는 데이터를 개체, 관계, 속성으로 묘사
- E-R 모델은 특정 DBMS를 고려한 것이 아님
- E-R 모델은 다이어그램으로 표현하며, 일대일, 일대다, 다대다를 제한 없이 표현
- 최초에는 개체,속성,관계 같은 개념으로 구성되었으나 나중에는 일반화 계층 같은 복잡한 개념들이 첨가되어 확장된 모델로 발전

E-R 다이어그램(Entitiy-Relationship Diagram)

: E-R 모델의 기본 아이디어를 시각적으로 표현하기 위한 그림으로, 실체 간의 관계는 물론 조직, 사용자, 프로그램, 데이터 등 시스템 내에서 역할을 가진 모든 실체들을 표현한다.

기호	기호 이름	의미
	사각형	개체(Entity) 타입
	마름모	관계(Relationship) 타입
	타원	속성(Attribute)
	이중 타원	다중값 속성(복합 속성)
	밑줄 타원	기본키 속성
	복수 타원	복합 속성 예 성명은 성과 이름으로 구성
	관계	1:1, 1:N, N:M 등의 개체 간 관계에 대한 대응수를 선 위에 기술함
	선, 링크	개체 타입과 속성을 연결

섹션 83 관계형 데이터 모델 (C)

: 관계형 데이터 모델은 가장 널리 사용되는 데이터 모델로, 2차원 적인 표를 이용해서 데이터 상호 관계를 정의하는 DB구조이다.
- 파일 구조처럼 구성한 테이블들을 하나의 DB로 묶어서 테이블 내에 있는 속성들의 관계를 설정하거나 테이블 간의 관계를 설정하여 이용
- 기본키와 이를 참조하는 외래 키로 데이터 간의 관계를 표현
- 계층 모델과 망 모델의 복잡한 구조를 단순화시킨 모델
- 관계형 모델의 대표적인 언어는 SQL
- 일대일, 일대다, 다대다 관계를 자유롭게 표현

※ 액세스에서 관계 설정을 했던 테이블 판을 말함

섹션 84 관계형 데이터베이스의 구조

- 1970년 IBM에 근무하던 코드에 의해 처음 제안
- 관계형 데이터베이스를 구성하는 개체나 관계를 모두 릴레이션이라는 표로 표현
- 릴레이션은 개체를 표현하는 개체 릴레이션, 관계를 나타내는 관계 릴레이션으로 구분할 수 있다.
- 장점 : 간결하고 보기 편리하며, 다른 데이터베이스로의 변환이 용이
- 단점 : 성능이 다소 떨어짐

관계형 데이터베이스의 Relation 구조 ★★★★★

: 릴레이션은 데이터들을 표 형태로 표현한 것으로 구조를 나타내는 **릴레이션 스키마**와 실제 값들인 **릴레이션 인스턴스**로 구성된다.

릴레이션 스키마 : 속성 들의 집합이라고 보면됨 (테이블의 헤더)
도메인 : 어떤 속성이 가지는 데이터 값들의 집합

튜플
- 릴레이션을 구성하는 각각의 행
- 튜플은 속성들의 모임으로 구성됨
- 파일 구조에서 레코드와 같은 의미
- 튜플의 수를 카디널리터(Cardinality) 또는 기수, 대응수라 한다.

속성
- 속성은 데이터베이스를 구성하는 가장 작은 **논리적** 단위
- 파일 구조상의 데이터 항목 또는 데이터 필드에 해당
- 속성은 개체의 특성을 기술
- 속성의 수를 디그리 또는 차수라고 함

도메인
- 하나의 속성이 갖을 수 있는 같은 타입의 원자값들의 집합
- 도메인은 실제 속성 값이 나타날 때 그 값의 합법 여부를 시스템이 검사하는데도 이용됨

릴레이션의 특징
- 한 릴레이션에는 똑같은 튜플이 포함될 수 없으므로 릴레이션에 포함된 튜물들은 모두 상이하다.
- 한 릴레이션에 포함된 튜플 사이에는 순서가 벗다
- 튜플들의 삽입, 삭제 등의 작업으로 인해 릴레이션은 시간에 따라 변한다
- 릴레이션 스키마를 구성하는 속성들의 순서는 중요하지 않다
- 속성은 유일한 식별을 위해 속성의 명칭은 유일해야 하지만, 속성을 구성하는 값은 동일한 값이 있을 수 있다.
- 릴레이션을 구성하는 튜플을 유일하게 식별하기 위해 속성들의 부분집합을 키로 설정한다.
- 속성의 값은 논리적으로 더 이상 쪼갤 수 없는 원자값만을 저장한다

섹션 85 관계형 데이터베이스의 제약 조건 - 키(key)

제약 조건이란 데이터베이스에 저장되는 데이터의 정확성을 보장하기 위해 키를 이용하여 입력되는 데이터에 제한을 주는 것으로 개체 무결성 제약, 참조 무결성 제약 등이 해당된다.

키의 개념 및 종류
: 키는 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 튜플들을 서로 구분할 수 있는 기준이 되는 속성을 말한다
키의 종류에는 **후보키, 기본키, 대체키, 슈퍼키, 외래키** 등이 있다.

1. **후보키 (Candidate Key)** : 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용하는 속성들의 부분집합, 즉 기본키로 사용할 수 있는 속성들을 말한다
 - 하나의 릴레이션내에서는 중복된 튜플들이 있을 수 없으므로 모든 릴레이션에는 반드시 하나 이상의 후보키가 존재한다.
 - 후보키는 릴레이션에 있는 모든 튜플에 대해서 유일성과 최소송을 만족해야만 한다.
※ 최소성은 모든 레코드들을 유일하게 식별하는 데 꼭 필요한 속성으로만 구성되어야한다는 원칙

2. **기본키 (Primary Key)** : 후보키 중에서 특별히 선정된 주키(Main Key)로 중복된 값을 가질 수 없다.
 - 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성이다.
 - 기본키는 후보키의 성질을 갖는다. 즉 유일성과 최소성을 가지며 튜플을 식별하기 위해 반드시 필요한 키이다
 - 기본키는 널값을 가질 수 없다. 즉 튜플에서 기본키로 설정된 속성에는 널값이 있어서는 안된다.

3. **대체키 (Alternate Key)** : 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키를 의미한다.
 - 보조키라고도 한다

4. **슈퍼키** : 한 릴레이션 내에 있는 속성들의 **집합**으로 구성된 키로서 릴레이션을 구성하는 모든 튜플들 중 슈퍼키로 구성된 속성의 집합과 동일한 값은 나타나지 않는다.
 - 슈퍼키는 릴레이션을 구성하는 모든 튜플에 대해 유일성은 만족하지만 최소성은 만족시키지 못한다

5. **외래키 (Foreign Key)** :다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합
 - 외래키는 참조되는 릴레이션의 기본키와 대응되어 릴레이션 간의 참조 관계를 표현하는데 중요한 도구이다
 - 한 릴레이션에 속한 속성 A와 참조 릴레이션의 기본키인 B가 동일한 도메인 상에서 정의되었을 때의 속성 A를 외래키라고 한다
 - 외래키로 지정되면 참조 릴레이션의 기본키에 없는 값은 입력할 수 없다.

섹션 86 관계형 데이터베이스의 제약 조건 - 무결성

무결성 : 데이터베이스에 저장된 데이터 값과 그것이 표현하는 현실 세계의 실제 값이 일치하는 정확성을 의미한다.
 - 무결성 제약 조건은 데이터베이스에 들어 있는 데이터의 정확성을 보장하기 위해 부정확한 자료가 데이터베이스 내에 저장되는 것을 방지하기 위한 제약 조건을 말한다
 - 무결성의 종류에는 **개체 무결성, 도메인 무결성, 참조 무결성, 사용자 정의 무결성** 등이 있다.

1. **개체 무결성 (Entitiy Integrity, 실체 무결성) :** 기본 테이블의 기본키를 구성하는 어떤 속성도 널값이나 중복값을 가질 수 없다는 규정
2. **도메인 무결성 (Domain Integrity, 영역 무결성)** : 주어진 속성 값이 정의된 도메인에 속한 값이어야 한다는 규정. (유효성 검사 규칙에 위배되지 않아야함)
3. **참조 무결성 (Referential Integrity) :** 외래키 값은 널이거나 참조 릴레이션의 기본키 값과 동일해야한다. 즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다는 규정이다.
4. 사용자 정의 무결성 : 속성 값들이 사용자가 정의한 제약 조건에 만족해야 한다는 규정

데이터 무결성의 강화 : 데이터 무결성은 데이터 품질에 직접적인 영향을 미치므로 데이터 특성에 맞는 적절한 무결성을 정의하고 강화해야 한다.
 - 프로그램이 완성되고 데이터가 저장된 상태에서 무결성을 정의할 경우 많은 비용이 발생하므로 데이터베이스 구축 과정에서 정의한다
 - 데이터 무결성은 애플리케이션, 데이터베이스 트리거, 제약 조건을 이용하여 강화한다.

애플리케이션
 - 데이터 생성, 수정, 삭제 시 무결성을 검사하는 코드를 이용해 데이터를 조작하

는 프로그램에 추가
- 코드를 이용한 검사는 복잡하므로 데이터베이스에서 처리하기 어려움
- 사용자 정의 같은 복잡한 것을 구현이 가능하나 관리가 힘들고, 개별적인 시행으로 적정성 검토가 어렵다

데이터베이스 트리거
- 트리거 이벤트에 무결성 조건을 실행하는 절차형 SQL을 추가한다
- 통합관리가 가능하고, 복잡한 요구 조건의 구현이 가능하다
- 다만 운영 중 변경이 어렵고 , 사용상의 주의가 필요하다

제약조건
- 데이터베이스에 제약 조건을 설정하여 무결성을 유지
- 하지만 복잡한 제약 조건 구현과 예외처리가 불가능

섹션 87 관계대수 및 관계해석

관계대수는 관계형 데이터베이스서서 원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 **절차적인** 언어이다.
- 관계대수는 릴레이션을 처리하기 위해 연산자와 연산 규칙을 제공하는 언어로 피연산자가 릴레이션이고, 결과도 릴레이션이다.
- 질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다
- 관계대수에는 관계 데이터베이스에 적용하기 위해 특별히 개발한 순수 관계 연산자와 수학적 집합 이론에서 사용하는 일반 집합 연산자가 있다.

순수 관계 연산자 : Select, Project, Join, Division
일반 집합 연산자 : Union(합집합), Intersection(교집합), Difference(차집합), Cartesian product(교차곱) (데카르트 곱)

1. **Select :** 릴레이션에 존재하는 튜플 중에서 선택 조건을 만족하는 튜플의 부분 집합을 구하여 새로운 릴레이션을 만드는 연산
- 릴레이션의 행에 해당하는 튜플을 구하는 것이므로 수평 연산이라고도 한다
- 연산자의 기호는 그리스 문자 시그마(σ)를 사용한다
- **표기형식** : $\sigma_{<조건>}(R)$ - R은 릴레이션 이름
- 조건에서는 $=, \neq, <, \leq, >, \geq$ 등의 기호를 사용한 비교 연산이 허용 되며, And, OR, Not 등의 논리 연산자를 사용하여 여러 개의 조건을 하나의 조건으로 결합도 가능하다

2. Project : 주어진 릴레이션에서 속성 리스트에 제시된 속성 값만을 추출하여 새로운 릴레이션을 만드는 연산이다. 단 연산 결과에 중복이 발생하면 중복이 제거된다.
- 릴레이션의 열(세로)에 해당하는 속성을 추출하는 것이므로 수직 연산자라고도 한다.
- 연산자 기호는 그리스 문자 파이(π)를 사용한다.
- **표기형식 :** $\pi_{<속성리스트>}(R)$

3. **JOIN :** 공통 속성을 중심으로 두 개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만드는 연산
- Join의 결과로 만들어진 릴레이션의 차수는 조인된 두 릴레이션의 차수를 합한 것과 같다
- Join의 결과는 교차곱(Cartesian Product)을 수행한 다음 Select를 수행한 것과 같다.
- 연산자의 기호는 $\triangleright\triangleleft$를 사용한다
- **표기형식 :** $R \triangleright\triangleleft_{키속성r = 키속성s} S$
- 키 속성 r은 릴레이션 R의 속성이고, 키 속성 s은 릴레이션 S의 속성이다
- 조인 조건이 '='인 경우 동일한 속성이 두 번 나타나는데 자연(Natural) 조인을 이용하면 한번만 표기하게 된다. 이 경우 두 속성의 속성명과 도메인이 같아야한다.

4. **Division :** $X \supset Y$인 두 개의 릴레이션 R(X)와 S(Y)가 있을 때, R의 속성이 S의 속성값을 모두 가진 튜플에서 S가 가진 속성을 제외한 속성만을 구하는 연산이다
- 연산자의 기호는 $\div$를 사용한다한다
- 표기형식 : R[속성r$\div$속성s]S

계속해서 문제를 풀어보자

일반 집합 연산자 : 수학적 집합 이론에서 사용하는 연산자로서 릴레이션 연산에도 그래도 적용 가능

- 일반 집합 연산자 중 합집합, 교집합, 차집합을 처리하기 위해서는 합병 조건을 만족해야한다. (두 릴레이션 간의 속성의 수가 같고, 대응하는 속성별로 도메인이 같아야한다.)(속성의 이름은 달라도 된다)

관계해석(Relational Calculus) ★

: 관계 데이터 모델의 제안자인 코드가 수학의 Predicate Calculus(술어 해석)에 기반을 두고 관계 데이터베이스를 위해 제안했댜.

- 관계해석은 관계 데이터의 연산을 표현하는 방법으로, 원하는 정보를 정의할 때는 계산 수식을 사용
- 관계해석은 원하는 정보가 무엇이라는 것만 정의하는 **비절차적** 특성을 지닌다
- 튜플 관계해석과 도메인 관계해석이 있다
- 기본적으로 관계해석과 관계대수는 관계 데이터베이스를 처리하는 기능과 능력4--면에서 동등하며, 관계대수로 표현한 식은 관계해석으로 표현할 수 있다.
- 질의어로 표현한다
- 주요 논리기호 전칭 정량자 / 존재 전량자 ∀, ∃

섹션 88 정규화 (Normalization)★★★★★

개요 : 함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정이다

- 하나의 종속서이 하나의 릴레이션에 표현될 수 있도록 분해해가는 과정이라 할 수 있다.
- 정규형에는 제 1정규형부터 제 5정규형, BCNF형 까지 있으며 차수가 높아질수록 만족시켜야할 제약 조건이 늘어난다
- 정규화는 데이터베이스의 논리적 설계 단계에서 수행한다
- 정규화는 논리적 처리 및 품질에 큰 영향을 미친다
- 정규화된 데이터의 모델은 일관성, 정확성, 단순성, 비중복성, 안정성 등을 보장한다.
- 정규화 수준이 높을수록 유연한 데이터 구축이 가능하고 데이터의 정확성이 높이지는 반면 물리적 접근이 복잡하고 너무 많은 조인으로 인해 조회 성능이 저하

정규화의 목적★★

- 데이터 구조의 안정성 및 무결성 유지
- 어떠한 릴레이션이라도 데이터베이스 내에서 표현 가능하게 만듬
- 효과적인 검색 알고리즘을 생성가능
- 데이터 중복을 배제하여 이상의 발생 방지 및 자료 저장 공간의 최소화 가능
- 데이터 삽입 시 릴레이션을 재구성할 필요가 없음
- 데이터 모형의 단순화가 가능
- 속성의 배열 상태 검증이 가능
- 개체와 속성의 누락 여부 확인이 가능
- 자료 검색과 추출의 효율성을 추구

이상(Anomaly)의 개념 및 종류 ★★★

: 정규화를 거치지 않으면 데이터베이스 내에 데이터들이 불필요하게 중복되어 릴레이션 조작 시 예기치 못한 곤란한 현상이 있는데, 이를 이상이라 하며 삽입 이상, 삭제 이상, 갱신 이상이 있다.

- **삽입 이상 :** 릴레이션에 데이터를 삽입할 때 의도와는 상관없이 원하지 않은 값들도 함께 삽입되는 현상
- **삭제 이상 :** 릴레이션에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 연쇄가 일어나는 현상
- **갱신 이상 :** 릴레이션에서 튜플에 있는 속성값들을 갱신할 때 일부 튜플의; 정보만 갱신되어 정보에 모순이 생기는 현상

정규화의 원칙

- 정보의 무손실 표현, 즉 하나의 스키마를 다른 스키마로 변환할 때 정보의 손실이 있어서는 안된다
- 분리의 원칙, 즉 하나의 독립된 관계성은 하나의 독립된 릴레이션으로 분리하여 표현해야 한다.
- 데이터의 중복성이 감소되어야 한다.

정규화 과정 ★★★★★★★★

1NF (제 1 정규형) : 1NF는 릴레이션에 속한 모든 도메인이 원자값만으로 되어 있는 정규형이다. 즉 릴레이션의 모든 속성 값이 원자 값으로만 되어 있는 정규형

- 릴레이션의 모든 속성이 단순 영역에서 정의된다.

2NF (제 2 정규형) : 2NF는 릴레이션 R이 1NF이고, 기본키가 아닌 모든 속성이 기본키에 대하여 완전 함수적 종속을 만족하는 정규형이다.

※ 함수적 종속 (Functional Dependency)
- 데이터들이 어떤 기준값에 의해 종속되는 것을 의미
- 예를 들어 '학번'을 기준으로 '이름'은 항상 그 학번에 따라 결정됨

※완전 함수적 종속
: 어떤 테이블 R에서 속성 A가 다른 속성 집합 B 전체에 대해 함수적 종속이지만 속성 집합 B에의 어떠한 진부분 집합 C에는 함수적 종속이 아닐 때 속성 A는 속성 집합 B에 대해 완전 함수적 종속이라 한다.

※부분 함수적 종속
: 어떤 테이블 R에서 속성 A가 다른 속성 집합 B 전체에 대해 함수적 종속이면서 속성 집합 B의 어떠한 진부분 집합에도 함수적 종속일 때, 속성 A는 속성 집합 B에 부분 함수적 종속이라 한다.

※완전/부분 함수적 종속의 이해
- 완전 함수적 종속은 어떤 속성이 기본키에 대해 완전히 종속적일 때를 말한다

예) 〈수강〉 릴레이션이 {학번, 과목명, 성적, 학년)으로 되어있고 (학번,과목명)이 기본키일 때, '성적'은 '학번'과 '과목명'이 같을 경우에는 항상 같은 '성적'이 나온다. 즉, '성적'은 '학번'과 '과목명'에 의해서만 결정되므로 '성적'은 기본키(학번,과목명)에 완전 함수적 종속이 되는 것이다

그러나 '학년'은 '과목명'에 관계없이 '학번'이 같으면 항상 같은 '학년'이 온다. 즉 기본키의 일부인 '학번'에 의해서 '학년'이 결정되므로 '학년'은 부분 함수적 종속이다.

3NF (제 3 정규형) : 3NF는 릴레이션 R이 2NF이고, 기본키가 아닌 모든 속성이 기본키에 대하여 이행적 종속을 만족하지 않는 정규형이다.
- 무손실 조인 또는 종속성 보존을 저해하지 않고도 항상 3NF 설계를 얻을 수 있다.

※이행적 종속은 추이율을 말함

BCNF(Boyce-Codd 정규형)
: BCNF는 릴레이션 R에서 결정자가 모두 후보키(Candidate Key)인 정규형
- 3NF에서 후보키가 여러 개 존재하고 서로 중첩되는 경우에 적용하는, 강한 제3정규형이라고도 한다
- 모든 BCNF(Boyce-Codd Normal Form)가 종속성을 보존하는 것은 아니다

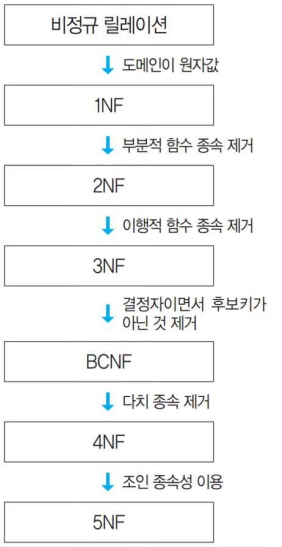
BCNF의 제약 조건
- 키가 아닌 모든 속성은 각 키에 대하여 완전 종속해야 한다
- 키가 아닌 모든 속성은 그 자신이 부분적으로 들어가 있지 않은 모든 키에 대하여 완전 종속해야한다
- 어떤 속성도 키가 아닌 속성에 대해서는 완전 종속할 수 없다

4NF (제 4 정규형) : 4NF는 릴레이션 R에 다치 종속(Multi Valued Dependency,다가 종속) A →→ B 까 성립하는 경우 R의 모든 속성이 A에 함수적 종속 관계를 만족하는 정규형이다

5NF (제 5 정규형) : 5NF는 릴레이션 R의 모든 조인 종속이 R의 후보키를 통해서만 성립되는 정규형이다.

정규화 과정 정리

도부이결다조



섹션 89 반정규화 (Denormalization)

개념 : 시스템의 성능 향상, 개발 및 운영의 편의성을 위해 정규화된 데이터 모델을 통합, 중복, 분리하는 과정으로, 의도적으로 정규화 원칙을 위배하는 행위

- 반정규화를 수행하면 시스템의 성능이 향상되고 관리 효율성은 증가하지만 데이터의 일관성 및 정합성이 저하될 수 있다.
- 과도한 반정규화는 오히려 성능을 저하시킬 수 있다
- 반정규화를 위해서는 사전에 데이터의 일관과 무결성을 우선으로 할지,데이터베이스의 성능과 단순화를 우선으로 할지 결정해야한다.
- 반정규화 방법에는 테이블 통합, 테이블 분할, 중복 테이블 추가, 중복 속성 추가 등이 있다.

1. 테이블 통합 : 두 개의 테이블이 조인되는 경우가 많아 하나아ㅣ 테이블로 합쳐 사용하는 것이 성능 향상에 도움이 돌 경우 수행한다.
- 두 개의 테이블에서 발생하는 프로세스가 동일하게 자주 처리되는 경우, 두 개의 테이블을 이용하여 항ㅅ강 조회를 수행하는 경우 테이블 통합을 고려한다
- 테이블 통합의 종류에는 1:1 관계 테이블 통합, 1:N 관계 테이블 통합, 슈퍼타입/서브타입 테이블 통합이 있다
- 테이블 통합 시 고려사항
  : 데이터 검색은 간편하지만 레코드 증가로 인해 처리량이 증가한다
  : 테이블 통합으로 인해 입력, 수정, 삭제 규칙이 복잡해질 수 있다.
  : Not Null, Default, Check 등의 제약조건을 설계하기 어렵다

2. 테이블 분할 : 테이블을 수직 또는 수평으로 분할한다
- 수평 분할 : 레코드를 기준으로 테이블을 분할
  레코드 별로 사용 빈도의 차이가 큰 경우 사용 빈도에 따라 분할한다.
- 수직 분할 : 하나의 테이블에 속성이 많을 경우 속성을 기준으로 테이블을 분할 갱신 위주의 속성/자주 조회되는 속성 분할/ 크기가 큰 속성 분할/ 보안을 적용해야 하는 속성 분할 등 사용한다
- 테이블 분할 시 고려사항
  : 기본키의 유일성 관리가 어려워진다
  : 데이터 양이 적거나 사용 빈도가 낮은 경우 분할이 필요한지 고려해야 한다
  : 분할된 테이블로 인해 수행 속도가 느려질 수 있다.
  : 데이터 검색에 중점을 두어 테이블 분할 여부를 결정해야 한다

중복 테이블 추가 : 여러 테이블에서 데이터를 추출해서 사용하거나 다른 서버에 저장된 테이블을 이용해야 하는 경우 중복 테이블을 추가하여 작업의 효성을 향상시킬 수 있다.
- 정규화로 인해 수행 속도가 느려지는 경우
- 많은 범위의 데이터를 자주 처리해야 하는 경우
- 특정 범위의 데이터만 자주 처리해야 하는 경우
- 처리 범위를 줄이지 않고는 수행 속도를 개선할 수 없는 경우

중복 테이블을 추가하는 방법
- 집계 테이블의 추가 : 집계 데이터를 위한 테이블을 생성하고, 각원본 테이블에 트리거를 설정하여 사용하는 것으로, 트리거의 오버헤드에 유의해야한다
- 진행 테이블의 추가 : 이력 관리 등의 목적으로 추가하는 테이블로, 적절한 데이터 양의 유지와 활용도를 높이기 위해 기본키를 적절히 설정한다.
- 특정 부분만을 포함하는 테이블의 추가 : 데이터가 많은 테이블의 특정 부분만을 사용하는 경우 해당 부분만으로 새로운 테이블을 생성한다

중복 속성 추가 : 조인해서 데이터를 처리할 떄 데이터를 조회하는 경로를 단축하기 위해 자주 사용하는 속성을 하나 더 추기하는 것이다
- 중복 속성을 추가하면 데이터의 무결성 확보가 어렵고, 디스크 공간이 추가로 필요하다

중복 속성을 추가하는 경우
- 조인이 자주 발생하는 속성인 경우
- 접근 경로가 복잡한 속성인 경우
- 액세스의 조건으로 자주 사용되는 조건의 경우
- 기본키의 형태가 적절하지 않거나 여러 개의 속성으로 구성된 경우

중복 속성 추가 시 고려사항
- 테이블 중복과 속성의 중복 고려, 저장공간의 지나친 낭비 고려
- 데이터 일관성 및 무결성에 유의
- SQL 그룹 함수를 이용해 처리할 수 있어야 함

섹션 90 시스템 카탈로그

시스템 카탈로그의 의미 : 시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스이다.

- 시스템 카탈로그 내의 각 테이블은 사용자를 포함하여 BBMS에서 지원하는 모든 데이터 객체에 대한 정의나 명세에 관한 정보를 유지 관리하는 시스템 테이블
- 카탈로그들이 생성되면 데이터 사전(Data Dictionary)에 저장되기 때문에 좁은 의미로는 카탈로그를 대이터 사전이라고도 한다

시스템 카탈로그 저장 정보 : 시스템 카탈로그에 저장된 정보를 메타-데이터라 함

메타 데이터의 유형

- 데이터베이스 객체 정보 : 테이블, 인덱스, 뷰 등의 구조 및 통계 정보
- 사용자 정보 : 아이디, 패스워드, 접근 권한 등
- 테이블의 무결성 제약 조건 정보 : 기본키, 외래키, NULL값 허용 여부
- 함수, 프로시저, 트리거에 대한 정보

카탈로그의 특징

- 카탈로그 자체도 시스템 테이블로 구성되어 있어 일반 이용자도 SQL을 이용하여 내용을 검색해 볼 수 있다
- INSERT, DELETE, UPDATE문으로 카탈로그를 갱신하는 것은 허용되지 않음
- 데이터베이스 시스템에 따라 상이한 구조를 갖드다
- 카탈로그는 DBMS가 스스로 생성하고 유지한다
- **카탈로그의 갱신 :** 사용자가 SQL문을 실행시켜 기본 테이블, 뷰, 인덱스 등에 변화를 주면 시스템이 자동으로 갱신한다
- **분산 시스템에서의 카탈로그 :** 보통의 릴레이션, 인덱스, 사용자 등의 정보를 포함할 뿐 아니라 위치 투명성 및 중복 투명성을 제공하기 위해 필요한 모든 제어ㅗ 정보를 가져야 한다

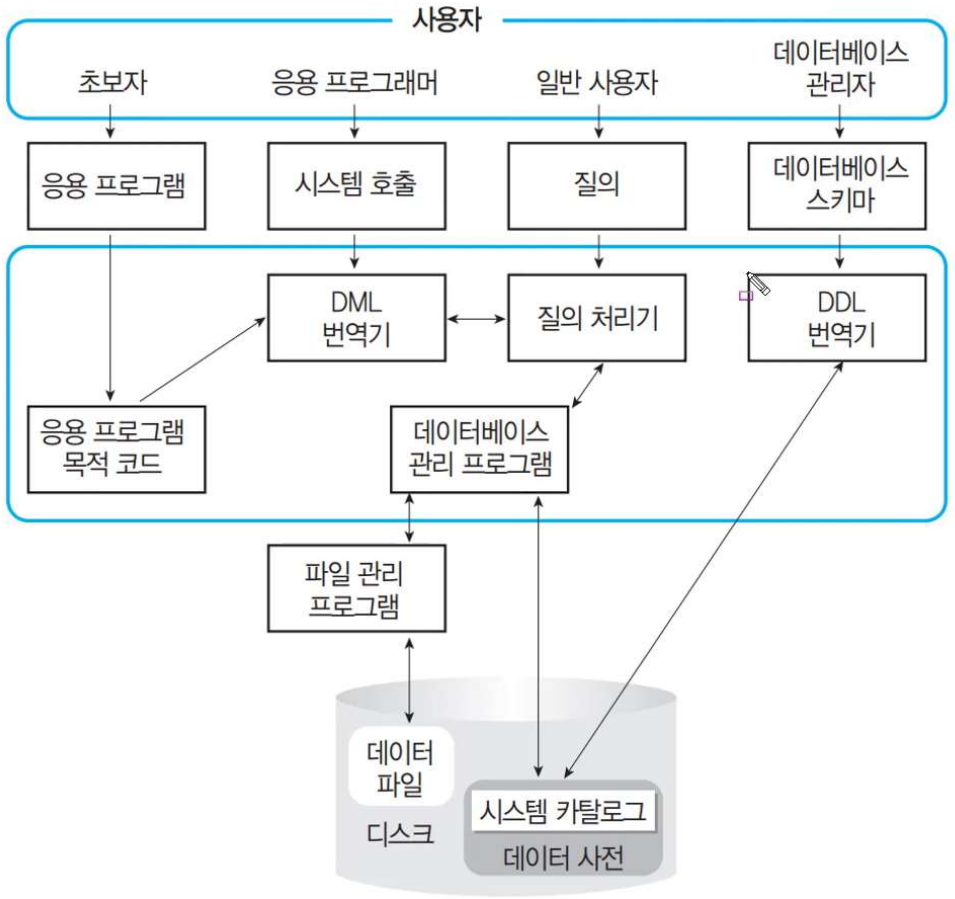
카탈로그/데이터 사전을 참조하기 위한 DBMS 내의 모듈 시스템

- 데이터 정의어 번역기(DDL Compiler) : DDL을 메타 데이터를 갖는 테이블(카탈로그)로 변환하여 데이터 사전에 저장시킨다
- 데이터 조작어 번역기(DML Compiler) : 응용 프로그램에 삽입된 DML문을 주언어로 표현한 프로시저 호출로 변환하여 질의 처리기와 상호 통신한다
- Data Dictionary

: 데이터 사전에 수록된 데이터를 실제로 접근하는 데 필요한 정보를 관리 유지하는 시스템

: 시스템 카탈로그는 사용자와 시스템 모두 접근할 수 있지만 데이터 디렉터리는 시스템만 접근할 수 있다

- 질의 최적화기 : 사용자의 요구를 효율적인 형태로 변환하고 질의를 처리하는 좋은 전략을 모색한다
- 트랜잭션 처리기 : 복수 사용자 환경에서 평행으로 동시에 일어나는 트랜잭션 문제를 해결하여, 각각의 사용자가 데이터베이스 자원을 배타적으로 이용할 수 있도록 한다.



2장 물리 데이터베이스 설계

섹션91 사전 조사 분석 (C)

물리 데이터베이스 설계 : 논리적 구조로 표현된 논리적 데이터베이스를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정

- 물리적 데이터베이스 구조의 기본적인 데이터 단위는 저장 레코드이다.
- 물리적 설계 단계에 꼭 포함되어야 할 것은 저장 레코드의 양식 설계, 레코드 집중(Record Clustrering)의 분석 및 설계, 접근 경로 설계 등이다.
- 물리적 데이터베이스 구조는 데이터베이스 시스템의 성능에 중대한 영향을 미침
- 물리적 데이터베이스 구조는 여러 가지 타입의 저장 레코드 집합이라는 면에서 단순한 파일과 다르다
- 물리적 데이터베이스 구조는 데이터베이스 시스템의 성능에 중대한 영향을 미침
- 물리적 설계 시 고려 사항

: 인덱스 구조/ 레코드 크기/ 파일에 존재하는 레코드의 개수 / 파일에 대한 트랜잭션의 갱신과 참조 성향 / 성능 향상을 위한 개념 스키마의 변경 여부 검토/ 빈번한 질의와 트랜잭션들의 수행속도를 높이기 위한 고려/ 시스템 운용 시 파일 크기의 변화 가능성

- 물리적 설계 전에 기존 시스템을 분석하여 데이터 명명 규칙, 시스템 자원, 데이터베이스 관리 요소 등을 파악해야 한다

물리적 설계 옵션 : 특정 DBMS에서 제공되는 것으로, 데이터베이스 파일에 대한 저장 구조와 접근 경로에 대해 다양한 옵션을 말한다

1. 반응 시간(Response Time) : 트랜잭션 수행을 요구한 시점부터 처리 결과를 얻을때까지의 경과 시간
2. 공간 활용도(Space Utilization) : 데이터베이스 파일과 액세스 경로 구조에 의해 사용되는 저장 공간의 양
3. 트랜잭션 처리량(Treansaction Throughput) : 단위시간 동안 데이터베이스 시스템에 의해 처리될 수 있는 트랜잭션의 평균 개수

데이터 명명 규칙 파악 : 물리 데이터 모델이 적용해야 하는 규칙으로, 조직마다 다를 수 있으므로 물리 데이터 모델의 설계 전에 파악해야 한다

- 데이터 명명 규칙은 데이터 표준회 및 논리 데이터베이스 설계의; 결과물 등을 통해 파악한다
- 물리 데이터베이스 설계와 논리 데이터베이스 설계에 적용되는 명명 규칙은 서로 일관성을 유지해야 한다
- 데이터 명명 규칙은 논리적 데이터 요소를 물리적 데이터 요소로 전환할 때 동일 명칭 부여의 근거로 사용된다
- 데이터 명 규칙을 통해 중복 구축 등을 방지할 수 있다
- 명명 규칙을 파악하려면 도메인과 데이터 사전에 대한 지식이 필요하다
- 도메인 : 객체에 포함된 속성들의 데이터 타입, 크기 등을 표준화 규칙에 따라 일관성 있게 정의한 것을 의미한다

데이터 사전

- 전체 프로젝트 과정에서 일관성 있는 데이터 이름과 인터페이스를 제공하기 위해 데이터 속성의 논리명, 물리명, 용어 정의를 기술해 놓은 것이다
- 데이터 사전은 프로젝트에서 사용하는 명칭 부여의 근거로 사용된다

시스템 자원 파악 : 데이터베이스 설치에 영향을 미칠 수 있는 물리적인 요소들로, 사전에 미리 파악해야 한다

- 시스템 자원은 하드웨어 자원/ 운영체제 및 DBMS 버전 / DBMS 파라미터 정보 등으로 구분한다

1. 하드웨어 자원 : CPU / 메모리 / 디스크 / IO Controller / 네트워크 파악
2. 운영체제 및 DBMS 버전 : 데이터베이스 운영에 영향을 미칠 수 있으므로 관련 요소 등을 파악하고 적절하게 관리
3. DBMS 파라미터 정보

- DBMS 파라미터는 데이터베이스 관리 시스템별로 차이가 많고 관리 방법도 ㄷ 제각각이므로 시스템별 DBMS 파라미터의 종류 및 관리 대상들을 파악한다
- DBMS의 저장공간, 메모리 등에 대한 파라미터, 쿼리에서 활용하는 옵티마이저의 사용 방법등을 파악한다.

데이터베이스 관리요소 파악 : 데이터베이스 운영과 관련된 운영 요소로, 데이터베이스 시스템의 환경에 따라 달라질 수 있으므로 미리 파악한다

- 데이터베이스 관리 요소를 파악한 후 이를 기반으로 데이터베이스 시스템 조사 분석서를 작성한다

- 시스템 조사 분석서를 기반으로 데이터베이스의 구조, 이중화 구성, 분산 데이터베이스, 접근제어/접근통제, DB암호화 등의 범위와 특성을 파악한다

섹션92 데이터베이스 저장 공간 설계 (C)

테이블 : 테이블은 데이터베이스의 가장 기본적인 객체로 행과 열(로우와 컬럼)으로 이루어져 있다.

- 데이터베이스의 모든 데이터는 테이블에 저장된다
- 테이블은 논리 설계 단계의 개체에 대응하는 객체이다
- DBMS 종류에 따라 테이블의 명칭과 기능 등은 약간씩 차이가 있다
- 테이블의 종류로 일반 테이블/ 클러스터 인덱스 테이블 / 파티셔닝 테이블 / 외부 테이블/ 임시 테이블 등이 있다

1. 일반 테이블 : 현재 사용하는 대부분의 DBMS에서 표준 테이블로 사용되는 테이블
   - 테이블에 저장되는 데이터의 행 위치는 속성 값에 상관없이 데이터가 저장되는 순서에 따라 결정된다.
   - (일정한 기중벗이 입력하는대로 바로 저장)

2. 클러스터드 인덱스 테이블 : 기본키나 인덱스키의 순서에 따라 데이터가 저장되는 테이블이다
   - 클러스터 인덱스 테이블은 일반적인 인덱스르 사용하는 테이블에 비해 접근 경로가 단축된다
   - (예를 들어 사원번호의 오름차순 혹은 내림차순으로 정렬되어 테이블에 저장)

3. 파셔셔닝 테이블(Partitioning table) : 대용량의 테이블을 작은 논리적 단위인 파티션으로 나눈 테이블이다
   - 파티셔닝 테이블은 대용량의 데이터를 효과적으로 관리할 수 있지만 파티션 키를 잘못 구성하면 성능 저하 등의 역효과를 낼 수 있다
   - 파티셔닝 방식에 따라 범위 분할(Range), 해시 분할(Hash), 조합 분할(Composite) 등으로 나뉜다.
- 예를 들어 입사일자를 연단위로 묶거나 해시함수를 적용하여 묶거나)

4. 외부 테이블(External Table) : 데이터베이스에서 일반 테이블처럼 이용할 수 있는 외부 파일로, 데이터베이스 내에 객체로 존재한다
   - 외부 테이블은 데이터웨어하우스(Data Warehouse)에서 ETL(Extraction, Transformation, Loding) 등의 작업에 유용하게 사용된다

5. 임시 테이블(Temporary Table) : 트랜잭션이나 세션별로 데이터를 저장하고 처리할 수 있는 테이블
   - 임시 테이블에 저장된 데이터는 트랜잭션이 종료되면 사라진다
   - 임시 테이블은 절차저적인 처리를 위해 임시로 사용하는 테이블이다

컬럼(열) : 컬럼은 테이블의 열을 구성하는 요소로 데이터 타입, 길이 등으로 정의됨

- 데이터 타입은 데이터의 일관성 유지를 위해 사용되는 가장 기본적인 것으로, 도메인을 정의한 경우 도메인에 따라 데이터의 타입과 길이가 정의된다
- 두 컬럼을 비교하는 연산에서 두 컬럼의 데이터 비이나 길이가 다르면 DBMS 내부적으로 데이터 타입을 변환한 후 비교 연산을 수행한다
- 참조 관계인 컬럼들은 데이터 타입과 길이가 일치해야한다
- 데이터 타입과 길이 지정 시 고려 사항
  1. 가변 길이 데이터 타입 : 예상되는 최대 길이로 지정
  2. 고정 길이 데이터 타입 : 최소 길이로 지정
  3. 소수점 이하 자리수 : 소수점 이하 잘릿수는 반올림되어 저장
- 데이터 타입에 따른 컬럼의 물리적인 순서
  1. 고정 길이 컬럼이고 NOT Null인 컬럼 : 앞쪽
  2. 가변 길이 컬럼 : 뒤쪽
  3. Null 값이 많을 것으로 예상되는 컬럼 : 뒤쪽

테이블스페이스 : 테이블이 저장되는 논리적인 영역으로, 하나의 테이블스페이스에 하나 또는 그 이상의 테이블을 저장할 수 있다

- 테이블을 저장하면 논리적으로 테이블스페이스에 저장되고, 물리적으로는 해당 테이블스페이스와 연관된 데이터 파일에 저장된다
- 데이터베이스를 테이블, 테이블스페이스, 데이터 파일로 나눠 관리하면 논리적 구성이 물리적 구성에 종속되지 않아 투명성이 보장된다
- 테이블스페이스는 데이터베이스에 저장되는 내용에 따라 테이블, 인덱스, 임시

등의 용도로 구분하여 설계한다

- 테이블 스페이스 설계 시 고려사항
  1. 테이블스페이스는 업무별로 구분하여 지정한다
  2. 대용량 테이블은 하나의 테이블스페이스에 독립적으로 지정한다
  3. 테이블과 인덱스는 분리하여 저장한다
  4. LOB(Large Object) 타입의 데이터는 독립적인 공간으로 지정한다

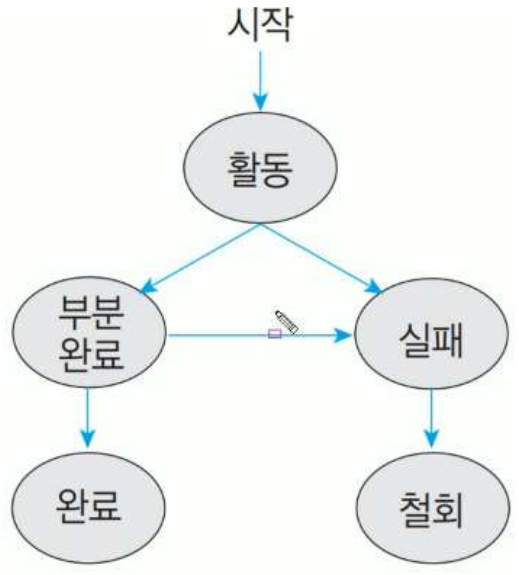
섹션93 트랜잭션 분석 / CRUD 분석★★★★★

트랜잭션의 정의 : 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미

- 트랜잭션은 데이터베이스 시스템에서 병행 제어 및 회복 작업 시 처리되는 작업의 논리적 단위로 사용된다
- 트랜잭션은 사용자가 시스템에 대한 서비스 요구 시 시스템이 응답하기 위한 상태 변환 과정의 작업 단위로 사용된다

※ CRUD가 바로 트랜잭션이 수행하는 연산

트랜잭션의 상태★★



활동 (Active)	트랜잭션이 실행 중인 상태
실패 (Failed)	트랜잭션 실행에 오류가 발생하여 중단된 상태
철회(Aborted)	트랜잭션이 비정상적으로 종료되어 Rollback 연산을 수행한 상태
부분 완료 (Partially Committed)	트랜잭션을 모두 성공적으로 실행한 후 Commit 연산이 실행되기 직전인 상태
완료 (Committed)	트랜잭션을 모두 성공적으로 실행한 후 Commit 연산을 실행한 후의 상태

트랜잭션의 특성 : 다음은 데이터의 무결성(Integrity)를 보장하기 위해 DBMS의 트랜잭션이 가져야할 특성이다.

Atomicity (원자성)	○트랜잭션의 연산은 데이베이스에 모두 반영되도록 완료(Commit) 되든지 아니면 전혀 반영되지 않도록 복구(Rollback)되어야 한다 ○트랜잭션 내이 모든 명려은 반드시 완벽히 수행되어야 하며, 모두가 완벽히 수행되지 않고 어느 할나라도 오류가 발생하면 트랜잭션 전부가 모두 취소되어야 한다.
Consistency (일관성)	○트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환한다 ○시스템이 가지고 있는 고정 요소는 트랜잭션 수행 전과 트랜잭션 수행 완료 후의 상태가 같아야한다
Isolation (독립성, 격리성, 순차성)	○둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행 중에 다른 트랜잭션의 연산이 끼어들 수 없다 ○수행중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조할 수 없다
Durability (영속성, 지속성)	○성공적을 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 한다

CRUD 분석 : CRUD는 생성(Create), 읽기(Read), 갱신(Update), 삭제(Delete)의 앞글자를 따서 만든 용어이며, CRUD분석은 데이터베이스 테이블에 변화를 주는 트랜잭션의 CRUD연산에 대해 CRUD 매트릭스를 작성하여 분석하는 것이다

- CRUD분석으로 테이블에 발생하는 트랜잭션의 주려 발생 횟수를 파악하고 연관된 테이블들을 분석하면 테이블에 저장되는 데이터의 양을 유추할 수 있다
- CRUD 분석을 통해 많은 트랜잭션이 몰리는 테이블을 파악할 수 있으므로 디스크 구성 시 유용한 자료로 활용할 수 있다
- CRUD분석을 통해 외부 프로세스 트랜잭션의 부하가 집중되는 데이터베이스 채

널을 파악하고 분산시킴으로써 연결 지연이나 타임아웃 오류를 방지할 수 있다

CRUD 매트릭스 : 2차원 형태의 표로서 , 행에는 프로세스를 열에는 테이블을, 행과 열이 만나는 위치에는 프로세스가 테이블 발생시키는 변화를 표시하는 업무 프로세스와 데이터 간 상관 분석표이다

- CRUD매트릭스를 통해 프로세스의 트랜잭션이 테이블에 수행하는 작업을 검증
- CRUD 매트릭스의 각 셀에는 C,R,U,D 가 들어가며 복수의 변화를 줄 경우, C〉R〉U〉D의 우선순위를 적용하여 한가지만 적지만, 활용 목적에 따라 복수 기록할 수 있다.

트랜잭션 분석 : 트랜잭션 분석의 목적은 CRUD 매트릭스를 기반으로 테르에 발생하는 트랜잭션양을 분석하여 테이블에 저장되는 데이터의 양을 유추하고 이를 근거로 DB용량을 산정하고 DB 구조를 최적화하는 것이다

- 트랜잭션 분석은 업무 개발 담당자가 수행한다
- 트랜잭션 분석을 통해 프로세스가 과도하게 접근하는 테이블을 확인하여 여러 디스크에 배치함으로써 디스크 입·출력 분산을 통한 성능 향상을 가져올 수 있다

트랜잭션 분석서 : 단위 프로세스와 CRUD 매트릭슬를 이용하여 작성하며, 구성요소에는 단위 프로세스, CRUD 연산, 테이블명, 컬럼명, 테이블 참조횟수, 트랜잭션 수, 발생 주기 등이 있다.

※트랜잭션 수는 주기별로 수행되는 트랜잭션 횟수

섹션94 인덱스 설계

인덱스의 개념 : 데이터 레코드를 빠르게 접근하기 위해 〈키 값, 포인터 〉 쌍으로 구성되는 데이터 구조이다

- 인덱스는 데이터가 저장된 물리적 구조와 밀접한 관계가 있다
- 인덱스는 레코드가 저장된 물리적 구조에 접근하는 방법을 제공한다
- 인덱스를 통해서 파일의 레코드에 대한 액세스를 빠르게 수행할 수 있다
- 레코드의 삽입과 삭제가 수시로 일어나는 경우 인덱스의 개수를 최소로 하는 것이 효율적이다
- 데이터 정의어(DDL)를 이용하여 사용자가 생성, 변경, 제거할 수 있다
- 인덱스가 없으면 특정한 값을 찾기 위해 모든 데이터 페이지를 확인하는 TABLE SCAN이 발생한다
- 기본키를 위한 인덱스를 기본 인덱스라 하고, 기본 인덱스가 아닌 인덱스들을 보조 인덱스ㄹ아고 한다. 대부분의 관계형 데이터베이스 관리 시스템셔셔ㅓ는 모든 기본키에 대하여 자동적으로 기본 인덱스를 생성한다
- 레코드의 물리적 순서가 인덱스의 엔트리 순서와 일치하게 유지되도록 구성되는 인덱스를 클러스터드 인덱스라고 한다.
- 인덱스는 인덱스를 구성하는 구조나 특징에 따라 구분할 수 있다

※클러스티드 인덱스 : 인덱스 키의 순서에 다라 정렬되어 저장되는 방식. 데이터 삽입, 삭제 시 순서 유지를 위해 재정렬 필요. 한 개의 릴레이션에 하나의 인덱스만 생성 가능

※넌클러스티드 인덱스 : 인덱스의 키 값만 정렬되어 있을 뿐 실제 데이터는 정렬되지 않는 방식. 데이터 검색시 먼저 인덱스를 검색하여 실제 위치를 확인해야하므로 검색속도가 느림. 한 개의 릴레이션에 여러 개의 인덱스를 만들 수 있음

1. 트리 기반 인덱스 : 인덱스를 저장하는 블록들이 트리 구조를 이루고 잇는 것으로 상용 DMBMS에서는 트리 구조 기반의 B+ 트리 인덱스를 주로 사용함

○B 트리 인덱스

- 일반적으로 사용되는 인덱스 방식으로, 루트 노드에서 하위 노드로 키 값의 크기를 비교해 나가면서 단말 노드에서 찾고자 하는 데이터를 검색
- 키 값과 레코드를 가리키는 포인터들이 트리 노드에 오름차순으로 저장
- 모든 리프 노드는 같은 레벨에 있음
- 브랜치 블록과 리프 블록으로 구성됨

  브랜치 블록 : 분기를 위한 목적으로 사용되고, 다음 단계를 가리키는 포인터를 지니고 잇음

  리프 블록 : 인덱스를 구성하는 컬럼 데이터와 해당 데이터의 행 위치를 가리키는 레코드 식별자로 구성됨

○B+ 트리 인덱스

- B+ 트리는 B 트리의 변형으로 단말 노드가 아닌 노드로 구성된 인덱스 세트(INDEX SET)와 단말 노드로만 구성된 순차 세트(Sequence Set)로 구성된다
- 인덱스 세트에 있는 노드들은 단말 노드에 있는 키 값을 찾아갈 수 있는 경로

롤만 제공되며, 순차 세트에 있는 단말 노드가 해당 데이터 레코드의 주소를 가리킴
 - 인덱스 세트에 있는 모든 키 값이 단말 노드에 다시 나타나므로 단말 노드만을 이용한 순차 처리가 가능하다

2. 비트맵 인덱스 : 인덱스 컬럼의 데이터를 Bit 값인 0과 1로 변환하여 인덱스 키로 사용하는 방법이다
 - 비트맵 인덱스의 목적은 키 값을 포함하는 행의 주소를 제공하는 것이다
 - 비트맵 인덱스는 분포도가 좋은 컬럼에 적합하며, 성능 향상을 기대할 수 있음
 - 데이터가 Bit로 구성되어 있어 효율적인 논리 연산이 으하고 저장 공간이 작음
 - 비트맵 인덱스는 다중 조건을 만족하는 튜플의 개수 계산에 적합
 - 비트맵 인덱스는 동일한 값이 반복되는 경우가 많아 압축 효율이 좋음

3. 함수 기반 인덱스 : 컬럼의 값 대신 컬럼에 특정 함수나 수식을 적용하여 산출된 값을 사용하는 것으로, B+ 트리 인덱스 또는 비트맵 인덱스를 생성하여 사용함.
 - 함수 기반 인덱스는 데이터를 입력하거나 수정할 때 함수를 적용해야 하므로 부하가 발생할 수 있다
 - 사용된 함수가 사용자 정의 함수인 경우 시스템 함수보다 부하가 더 큼
 - 함수 기반 인덱스는 대소문자, 띄어쓰기 상관없이 조회할 때 유용
 - 적용 가능한 함수의 종류 : 산술식, 사용자 정의 함수, PL/SQL Function, SQL Function, Package, C callout 등

4. 비트맵 조인 인덱스 : 다수의 조인된 객체로 구성된 인덱스로, 단일 객체로 구성된 일반적인 인덱스와 액세스 방법이 다르다
 - 비트맵 조인 인덱스는 비트맵 인덱스와 물리적 구조가 동일함

5. 도메인 인덱스 : 개발자가 필요한 인덱스를 직접 만들어 사용하는 것으로, 확장형 인덱스(Extensible Index)라고도 한다
 - 개발자가 필요에 의해 만들었지만 프로그램에서 제공하는 인덱스처럼 사용할 수도 잇음

 인덱스 설계 : 분명하게 드러난 컬럼에 대해 기본적인 인덱스를 먼저 지정한 후 개발 단계에서 필요한 인덱스의 설계를 반복적으로 진행한다
- 순서 :
  1. 인덱스의 대상 테이블이나 컬럼 등을 선정
  2. 인덱스의 효율성을 검토하여 인덱스 최적화 수행
  3. 인덱스 정의서 작성

인덱스 대상 테이블 선정 기준
- MULTI BLOCK READ 수에 따라 판단 (메모리가 한번에 읽어 올수 있는 블록의 수)
- 랜덤 액세스가 빈번한 테이블
- 특정 범위나 특정 순서로 데이터 조회가 필요한 테이블
- 다른 테이블과 순차적 조인이 발생되는 테이블

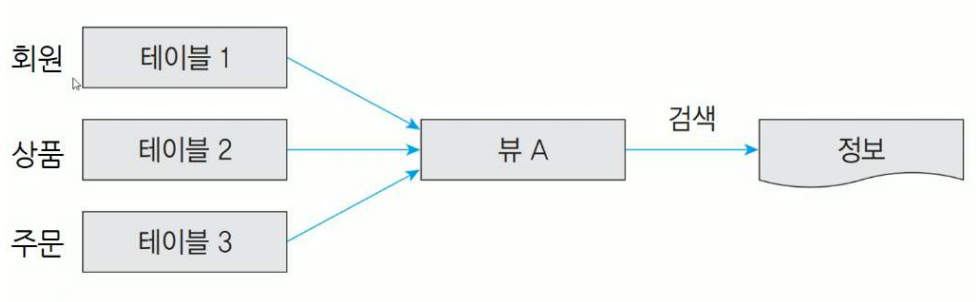
인덱스 대상 컬럼 선정 기준
- 인덱스 컬럼의; 분포도가 10~15% 이내인 컬럼
분포도 = (컬럼값의 평균 Row 수 / 테이블의 총 Row수) * 100
- 분포도가 높아도 부분 처리를 목적으로 하는 컬럼
- 입·출력 장표 등ㅇ에서 조회 및 출력 조건으로 사용되는 컬럼
- 인덱스가 자동 생성되는 기본키와 Unique키 제약 조건을 사용한 컬럼
- 가능한 수정이 빈번하지 않은 컬럼
- ORDER BY, GROUP BY, UNION이 빈번한 컬럼
- 분포도가 좁은 컬럼은 단독 인덱스로 생성
- 인덱스들이 자주 조합되어 사용되는 경우 하나의 결합 인덱스로 생성\

인덱스 설계 시 고려사항
- 새로 추가되는 인덱스는 기존 액세스 경로에 영향을 미칠 수 잇음
- 인덱스를 지나치게 많이 만들면 오버헤드 발생
- 넓은 범위를 인덱스로 처리하면 많은 오버헤드 발생
- 인덱스를 만들면 추가적인 저장 공간 필요
- **인덱스와 테이블 데이터의 저장 공간이 분리되도록 설계해야함**

섹션95 뷰(view 설계)

뷰의 개요 : 뷰는 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된, 이름을 가지는 가상 테이블이다

- 뷰는 저장장치 내에 물리적으로 존재하지 않지만, 사용자에게는 있는 것처럼 간주된다.
- 뷰는 데이터 보정 작업, 처리 과정 시험 등 임시적인 작업을 위한 용도로 활용
- 뷰는 조인문의 사용 최소화로 사용상의 편의성을 최대화한다.
- 뷰를 생성하면 뷰 정의;가 시스템 내에 저장되었다가 생성된 뷰 이름을 질의어에서 사용할 경우 질의어가 실행될 때 뷰에 정의된 기본 테이블로 대체되어 기본 테이블에 대해 실행된다
- 다음 그림은 뷰 A가 테이블 1, 테이블 2, 테이블 3에서 유도되어 생성되며, 뷰 A를 통해 테이블 1, 테이블 2, 테이블 3에 대한 데이터에 접근할 수 있음을 나타낸 것이다.



뷰의 특징 ★★★★★

- 뷰는 기본 테이블로부터 유도된 테이블이기 때문에 기본 테이블과 같은 형태의 구조를 사용하며, 조작도 기본 테이블과 거의 같다
- 뷰는 가상 테이블이기 때문에 물리적으로 구현되어 있지 않다
- 데이터의 논리적 독립성을 제공할 수 있다
- 필요한 데이터만 뷰로 정의해서 처리할 수 있으므로, 관리가 용이하고 명령문이 간단해진다
- 뷰를 통해서만 데이터에 접근하게 하면 뷰에 나타나지 않는 데이터들을 안전하게 보호하는 효율적인 기법으로 사용할 수 있따
- 기본 테이블의 기본키를 포함한 속성(열) 집합으로 뷰를 구성해야만 삽입, 삭제, 갱신 연산이 가능하다
- 일단 정의된 뷰는 다른 뷰의 정의에 기초가 될 수 있다
- 뷰가 정의된 기본 테이블이나 뷰를 삭제하면 그 테이블이나 뷰를 기초로 정의된 다른 뷰도 자동으로 삭제된다.
- 뷰를 정의할때는 CREATE문, 제거할 때는 DROP문을 사용한다 (DDL)

뷰의 장단점

장점	단점
○논리적 데이터 독립성을 제공한다. ○동일 데이터에 대해 동시에 여러 사용자의 상이한 응용이나 요구를 지원해 준다. ○사용자의 데이터 관리를 간단하게 해준다 ○접근 제어를 통한 자동 보안이 제공된다.	○독립적인 인덱스를 가질 수 없다 ○뷰의 정의를 변경할 수 없다 ○뷰로 구성된 내용에 대한 삽입, 삭제, 갱신 연산에 제약이 따른다

뷰의 설계 순서

1. 대상 테이블을 선정
   - 외부 시스템과 인터페이스에 관여하는 테이블
   - CRUD 매트릭스를 통해 여러 테이블이 동시에 자주 조인되어 접근되는 테이블
   - SQL문 작성 시 거의 모든 문장에서 인라인 뷰 방식으로 접근되는 테이블

※인라인 뷰 : FROM절에 사용되는 서블 쿼리

2. 대상 컬럼 선정
   - 보안을 유지해야 하는 컬럼은 주의하여 선별
3. 뷰 정의서 작성

뷰 설계 시 고려 사항

- 테이블 구조가 단순화 될 수 있도록 반복적으로 조인을 설정하여 사용하거나 동일한 조건절을 사용하는 테이블을 뷰로 생성한다
- 동일한 테이블이라도 업무에 따라 테이블을 이용하는 부분이 달라질 수 있으므로 사용할 데이터를 다양한 관점에서 제시해야 한다
- 데이터의 보안 유지를 고려하여 설계한다

섹션96 클러스터 설계

클러스터의 개요 : 데이터 저장 시 데이터 액세스 효율을 향상시키기 위해 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법이다.

- 클러스터링 키로 지정된 컬럼 값의 순서대로 저장되고, 여러 개의 테이블이 하나의 클러스터에 저장된다

클러스터의 특징

- 클러스터링 된 테이블은 데ㅐ이터 조회 속도는 향상시키지만 데이터 입력, 수정, 삭제에 대한 성능은 저하시킨다
- 클러스터는 데이터의 분포도가 넓을수록 유리하다 (인덱스와 반대)
- 데이터 분포도가 넓은 테이블을 클러스터링 하면 저장 공간을 절약할 수 있다
- 클러스터링 테이블은 클러스터링키 열을 공유하므로 저장 공간이 줄어든다
- 대용량을 처리하는 트랜잭션은 전체 테이블을 스캔하는 일이 자주 발생하므로 클러스터링을 하지 않는 것이 좋다
- 처리 범위가 넓은 경우에는 단일 테르 클러스터링을, 조인이 많이 발생하는 경우엔는 다중 테이블 클러스터링을 사용한다
- 파티셔닝된 테이블에는 클러스터링을 할 수 없다
- 클러스터링을 하면 비슷한 데이터가 동일한 데이터 블록에 저장되기 때문에 디스크 I/O가 줄어든다
- 클러스터링된 테이블에 클러스터드 인덱스를 생성하면 접근 성능이 향상된다

클러스터 대상 테이블

- 분포도가 넓은 테이블
- 대량의 범위를 자주 조회하는 테이블
- 입력, 수정, 삭제가 자주 발생하지 않는 테이블
- 자주 조인되어 사용되는 테이블
- ORDER BY, GROUP BY, UNION 이 빈번한 테이블

섹션97 파티션 설계

파티션의 개요 : 데이터베이스에서 파티션은 대용량의 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것을 말한다

- 대용량 DB인 경우 중요한 몇 개의 테이블에만 집중되어 데이터가 증가되므로, 이런 테이블들을 작은 단위로 나눠 분산시키면 성능 저하를 방지할 뿐만 아니라 데이터 관리도 쉬워짐
- 테이블이나 인덱스를 파티셔닝 하면 파티션키 혹은 인덱스키에 따라 물리적으로 별도의 공간에 데이터가 저장된다
- 데이터 처리는 테이블 단위로 이루어지고, 데이터 저장은 파티션별로 수행된다

파티션의 장단점

장점	○데이터의 접근 시 액세스 범위를 줄여 쿼리 성능이 향상 ○파티션별로 데이터가 분산되어 저장되므로 디스크의 성능이 향상 ○파티션별로 백업 및 복구를 수행하므로 속도가 빠름 ○시스템 장애 시 데이터 손상 정도를 최소화할 수 있음 ○데이터 가용성이 향상 ○파티션 단위로 입출력을 분산시킬 수 있음
단점	○하나의 테이블을 세분화하여 관리하므로 세심한 관리가 요구 ○테이블간 조인에 대한 비용이 증가 ○용량이 작은 테이블에 파티셔닝을 수행하면 오히려 성능 저하

파티션의 종류 : 파티셔닝 방식에 따라 나뉨

범위 분할	○지정한 열의 값을 기준으로 범위를 지정하여 분할한다
해시 분할	○해시 함수를 적용한 결과 값에 따라 테이블을 분할 ○특정 파티션에 데이터가 집중되는 범위 분할의 단점을 보완한 것으로, 데이터를 고르게 분산할 때 유용 ○특정 데이터가 어디에 있는지 판단할 수 없다 ○고객번호, 주민번호 등과 같이 데이터가 고른 컬럼에 효과적
조합 분할	○범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할하는 방법 ○범위 분할한 파티션이 너무 커서 관리가 어려울 때 유용
목록 분할	○지정한 열 값에 대한 목록을 만들어 이를 기준으로 분할 예)'국가'라는 열에 '한국','미국','일본'이 있는 경우 '미국'을 제외할 목적으로 '아시아'라는 목록을 만들어 분할함
라운드 로빈	○레코드를 균일하게 분배하는 방식 ○각 레코드가 순차적으로 분배되며, 기본키가 필요 없다

파티션키 선정 시 고려사항

- 파티션키는 테이블 접근 유형에 따라 파티셔닝이 이뤄지도록 선정한다

- 데이터 관리의 용이성을 위해 이력성 데이터는 파티션 생성주기와 소멸주기를 일치시켜야 한다
- 매일 생성되는 날짜 컬럼, 백업의 기준이 되는 날짜 컬럼, 파티션 간 이동이 없는 컬럼, I/O 병목을 줄일 수 있는 데이터 분포가 양호 컬럼 등을 파티션키로 설정

인덱스 파티션 : 파티션된 테이블의 데이터를 관리하기 위해 인덱스를 나눈 것
- 인덱스 파티션은 파티션된 테이블의 종속 여부에 따라 Local Partitoned Index, Global Partitioned Index로 나뉜다

1. Local Partitoned Index : 테이블 파티션과 인덱스 파티션이 1:1 대응하게 파티셔닝 한다
2. Global Partitioned Index : 테이블 파티션과 인덱스 파티션이 독립적을 구성되도록 파티셔닝 한다.

※ Local Partitoned Index이 Global Partitioned Index에 비해 데이터 관리가 쉽다

- 인덱스 파티션은 인덱스 파티션키 컬럼의 위치에 따라 Prefixed Partitoned Index와 Non-Prefixed Partitoned Index로 나뉜다.

1.Prefixed Partitoned Index : 인덱스 파티션키가 인덱스 첫 번쨔 컬럼과 같음
2. Non-Prefixed Partitonedf Index : 다름

각각 위의 2가지, 2가지를 조합하여 총 4개의 인덱스 파티션을 구성하여 사용한다

섹션98 데이터베이스 용량 설계(C)

데이터가 저장될 공간을 정의하는 것
- 데이터베이스 용량을 설계할 때는 테이블에 저장할 데이터양과 인덱스, 클러스터 등이 차지하는 공간 들도 예측하여 반영해야 한다

데이터베이스 용량 설계의 목적
- 데이터베이스의 용량을 정확히 산정하여 디스크의 저장 공간을 효과적으로 사용하고 확장성 및 가용성을 높임
- 디스크의 특성을 고려하여 설계함으로써 디스크의 입출력 부하를 분산시키고 채널의 병목 현상을 최소화
- 데이터 접근성을 향상시키는 설계 방법
    1. 테이블의 테이블스페이스와 인덱스 테이블스페이스를 분리하여 구성
    2. 테이블스페이스와 임시 테이블스페이스를 분리하여 구성
    3. 테이블을 마스터 테이블과 트랜잭션 테이블로 분류
- 데이터베이스에 생성되는 오브젝트의 익스텐트 발생을 최소화하여 성능을 향상

※ 익스텐트(Extent) : 기본 용량이 다 찼을 경우 추가적으로 할당 되는 공간

- 데이터베이스 용량을 정확히 분석하여 테이블과 인덱스에 적합한 저장 옵션을 지정한다

데이터베이스 용량 분석 절차
1. 데이터 예쌍 건수, 행의 길이, 보존 기간, 증가율 등 기초 자료를 수집하여 용량 분석
2. 분석된 자료를 바탕으로 DBMS에 이용될 테이블, 인덱스 등 오브젝트별 용량을 산정
3. 테이블과 인덱스의 테이블스페이스 용량을 산정
4. 데이터베이스에 저장될 모든 데이터 용량과 데이터베이스 설치 및 관리를 위한 시스템 용량을 합해 디스크 용량을 산정

섹션99 분산 데이터베이스 설계

분산 데이터베이스의 정의 : 논리적으로는 하나의 시스템에 속하지만 물리적으로는 네트워크를 통해 여러 개의 컴퓨터 사이트에 분산되어 있는 데이터베이스를 말한다
- 분산 데이터베이스는 데이터의 처리나 이용이 많은 지역에 데이터베이르 위치시킴으로써 데이터의 처리가 간으한 해당 지역에서 해결될 수 있도록 한다

분산 데이터베이스의 구성요소

분산 처리기	자체적으로 처리 능력을 가지며, 지리적으로 분산되어 있는 컴퓨터 시스템을 말함
분산 데이터베이스	지리적으로 분산되어 있는 데이터베이스로서 해당 지역의 특성에 맞게 데이터베이스가 구성된다
통신 네트워크	분산 처리기들을 통신망으로 연결하여 논리적으로 하나의 시스템처럼 작동할 수 있도록 하는 통신 네트워크를 말한다.

분산 데이터베이스 설계시 고려사항
- 작업부하(Work Load)의 노드별 분산 정책
- 지역의 자치성 보장 정책
- 데이터의 읽관성 정책
- 사이트나 회선의 고장으로부터의 회복 기능
- 통신 네트워크를 통한 원격 접근 기능

분산 데이터베이스의 목표★★★

1. 위치 투명성(Location Transparency) : 액세스하려는 데이터베이스의 실제 위치를 알 필요 없이 단지 데이터베이스의 논리적인 명칭만으로 액세스할 수 있다
2. 중복 투명성(Replication Transparency) : 동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고, 시스템은 자동으로 여러 자료에 대한 작업을 수행한다
3. 병행 투명성(Concurrency Transparency) : 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않는다.
4. 장애 투명성(Failure Transparency) : 트랜잭션, DBMS, 네트워크, 컴퓨터 장애에도 불구하고 트랜잭션을 정확하게 처리한다

분산 데이터의 장·단점★

장점	단점
○지역 자치성이 높다	
○자료의 공유성이 향상 된다	
○분산 제어가 가능하다	○DBMS가 수행할 기능이 복잡하다
○시스템 성능이 향상된다	○데이터베이스 설계가 어렵다
○중앙 컴퓨터 장애가 전체 시스템에 영향을 끼치지 않는다	○소프트웨어 개발 비용이 증가한다 ○처리 비용이 증가한다
○효용성과 융통성이 높다	○잠재적 오류가 증가한다
○신뢰성 및 가용성이 높다	
○점진적 시스템 용량 확장이 용이하다	

분산 데이터베이스 설계 : 애플리케이션이나 사용자가 분산되어 저장된 데이터에 접근하게 하는 것을 목적으로 한다
- 잘못 설계된 분산 데이터베이스는 복잡성 증가, 응답 속도 저하, 비용 증가 등의 문제가 발생한다
- 분산 데이터베이스의 설계는 전역 관계망을 논리적 측면에서 소규모 단위로 분할한 후, 분할된 결과를 복수의 노드에 할당하는 과정으로 진행된다. 노드에 할당된 소규모 단위를 분할(Fragment)이라 부른다
- 분산 설계 방법에는 테이블 위치 분산, 분할(Fragmentation). 할당(Allocation)이 있다.

1. 테이블 위치 분산
: 데이터베이스의 테이블을 각기 다른 서버에 분산시켜 배치하는 방법을 의미한다
- 테이블의 위치를 분산할 때는 테이블의 구조를 변경하지 않으며, 다른 데이터베이스의 테이블과 중복되지 않게 배치한다
- 데이터베이스의 테이블을 각각 다른 위치에 배치하려면 해당 테이블들이 놓일 서버들을 미리 설정해야 한다

2. 분할(Fragment) : 분할은 테이블의 데이터를 분할하여 분산시키는 것
- 분할 규칙
  ○완전성 : 전체 테이블을 대상으로 분할해야 함
  ○재구성(Reconstruction) : 분할된 데이터는 관계 연산을 활용하여 본래의 데이터로 재구성할 수 있어야 한다
  ○상호 중첩 배제(Dis-jointness) : 분할된 데이터는 서로 다른 분할의 항목에서 속하지 않아야 한다.
- 주요 분할 방법 : 수평분할 / 수직 분할

3. 할당 : 동일한 분할을 여러 개의 서버에 생성하는 분산 방법으로, 중복이 없는 할당과 중복이 있는 할당으로 나뉜다
○비중복 할당 방식
- 최적의 노드를 선택해서 분산 데이터베이스의 단일 노드에서만 분할이 존재하도록 하는 방식이다
- 일반적으로 애플리케이션에는 릴레이션을 배타적 분할로 분리하기 힘든 요구가 포함되므로 분할된 테이블 간의 의존성은 무시되고 비용 증가, 성능 저하 등의 문제가 발생할 수 있다

○중복 할당 방식 : 동일한 테이블을 다른 서버에 복제하는 방식으로, 일부만 복제하는 부분 복제와 전체를 복제하는 완전 복제가 있다.

섹션100 데이터베이스 이중화 / 서버 클러스터링(C)

데이터베이스 이중화는 시스템 오류로 인한 데이터베이스 서비스 중단이나 물리적 손상 발생 시 이를 복구하기 위해 동일한 데이터베이스를 복제하여 관리하는 것

- 데이터베이스 이중화를 수행하면 하나 이상의 데이터베이스가 항상 같은 상태를 유지하므로 데이터베이스에 문제가 발생하면 복제된 데이터베이스를 이용하여 즉시 문제를 해결하 수 있다.
- 데이터베이스 이중화는 여러 개의 데이터베이스를 동시에 고나리하므로 사용자가 수행하는 작업이 데이터베이스 이중화 시스템에 연결된 다른 데이터베이스에도 동일하게 적용된다
- 데이터베이스 이중화를 이용하면 손쉽게 백업 서버를 운영할 수 있다.

데이터베이스 이중화의 분류 : 데이터베이스 이중화는 변경 내용의 전달 방식에 따라 Eager 기법과 Lazy 기법으로 나뉜다.

Eager 기법	트랜잭션 수행 중 데이터 변경이 발생하면 이중화된 모든 데이터베이스에 즉시 전달하여 변경 내용이즉시 적용되도록 하는 기법
Lazy 기법	트랜잭션 수행이 종료되면 변경 사실을 새로운 트랜잭ㄱ션에 작성하여 각 데이터베이스에 전달되는 기법으로, 데이터베이스마다 새로운 트랜잭션이 수행되는 것으로 간주된다

데이터베이스 이중화 구성 방법 : 활동-대기 방법과 활동-활동 방법이 있음

활동-대기 (Active-Stanby) 기법	○한 DB가 활동 상태로 서비스하고 있으면 다른 DB는 대기하고 있다가 활성 DB에 장애가 발생하면 대기 상태에 있던 DB가 자동으로 모든 서비스를 대신 수행한다 ○구성 방법과 관리가 쉬어 많은 기업에서 활용한다
활동-활동 (Active-Active) 기법	○두 개의 DB가 서로 다른 서비스를 제공하다가 둘 중 한쪽 DB에 문제가 발생하면 나머지 다른 DB가 서비스를 제공한다 ○두 DB가 모두 처리를 하기 때문에 처리율이 높지만 구성 방법 및 설정이 복잡하다

클러스터링(Clustering) : 두 대 이상의 서버를 하나의 서버처럼 운영하는 기술

- 클러스터링은 서버 이중화 및 공유 스토리지를 사용하여 서버의 고가용성을 제공
- 클러스터링에는 고가용성 클러스터링과 병렬 처리 클러스터링이 있다.

고가용성 클러스터링	하나의 서버에 장애가 발생하면 다른노드(서버)가 받아 처리하여 서비스 중단을 방지하는 방식으로, 일반적으로 언급되는 클러스터링이다.
병렬 처리 클러스터링	전체 처리율을 높이기 위해 하나의 작업을 여러 개의 서버에서 분산하여 처리하는 방식이다. ※사용자의 요청을 로드 밸런서(Load Balancer)가 여러 대의 서버로 분산

섹션101 데이터베이스 보안/ 암호화

데이터베이스 보안의 개요 : 데이터베이스의 일부분 또는 전체에 대해서 권한이 없는 사용자가 액세스하는 것을 금지하기 위해 사용되는 기술

- 보안을 위해 데이터 단위는 테이블 전체로부터 특정한 행과 열 위치에 있는 특정한 데이터 값에 이르기까지 다양하다
- 데이터베이스 사용자들은 일반적으로 서로 다른 객체에 대하여 다른 접근 권리 또는 권한을 갖게 된다.

암호화(Encryption) : 데이터를 보낼 때 송신자가 지정한 수신자 이외에는 그 내용을 알 수 없도록 평문을 암호문으로 변환하는 것

- 개인키와 공개키 방식 두가지가 있음

개인키 암호 방식(Private Key Encryption) = 비밀키 : 동일한 키로 데이터를 암호화하고 복호화

- 대칭 암호 방식 또는 단일키 암호화 기법이라고도 함
- 제 3자에게 비밀키가 노출시키지 않고 데이터베이스 사용 권한이 있는 사용자만 나눠가짐
- 종류 : 전위 기법, 대체 기법, 대수 기법, 합성 기법(DES, LUCIFER)

공개키 암호 방식 : 서로 다른키로 데이터를 암호화 및 복호화

- 데이터를 암호화할 때 사용하는 키 (공개키)는 데이터베이스 사용자에게 공개하고, 복호활 때의 키(비밀키는) 관리자가 비밀리에 관리하는 방법
- 비대칭 암호 방식이라고도 하며, 대표적으로 RSA(Rivest Shamir Adleman)가 있다.

섹션102 데이터베이스 보안 - 접근 통제

접근 통제 : 데이터 저장된 체와 이를 사용하려는 주체 사이의 정보 흐름을 제한하는 것이다.

- 접근통제는 데이터에 대해 다음과 같은 통제를 함으로써 불법적인 접근 및 파괴를 예방한다

1. 비인가된 사용자의 접근 감시
2. 접근 요구자의 사용자 식별
3. 접근 요구의 정당성 확인 및 기록
4. 보안 정책에 근거한 접근의 승인 및 거부 등

- 접근통제 기술에는 임의 접근통제(DAC), 강제 접근통제(MAC), 역할기반 접근통제(RBAC)가 있다.

임의 접근통제 (Discretionary Access Control)	○임의 접근통제에는 데이터에 접근하는 사용자의 신원에 따라 접근을 부여하는 방법 ○데이터 소유자가 접근통제 권한을 지정하고 제어한다 ○객체를 생성한 사용자가 생성된 객체에 대한 모든 권한을 부여받고, 부여된 권한을 다른 사용자에게 허가할 수도 잇음 ○임의 접근통제에 사용되는 SQL명령어에는 GRANT와 REVOKE가 있다.
강제 접근통제 (Mandatory Access Control)	○강제 접근통제는 주체와 객체의 등급을 비교하여 접근 권한을 부여하는 방식이다 ○시스템이 접근통제 권한을 지정한다 ○데이터베이스 객체별로 보안 등급을 부여할 수 있고, 사용자별로 인가 등급을 부여할 수 있다 ○주체는 자신보다 보안 등급이 높은 객체에 대해 읽기, 수정, 등록이 모두 불가능하고, 보안 등급이 같은 객체에 대해서는 읽기, 수정, 등록이 가능하고, 보안 등급이 낮은 객체는 읽기가 가능하다
역할기반 접근통제 (Role Based Access Control)	○사용자의 역할에 따라 접근 권한을 부여하는 방식이다 ○중앙관리자가 접근 통제 권한을 지정 ○임의 접근통제와 강제 접근통제의 단점을 보완하였으며, 다중 프로그래밍 환경에 최적화된 방식이다 ○중앙관리자가 역할마다 권한을 부여하면, 책임과 자질에 따라 역할을 할당받은 사용자들은 역할에 해당하는 권을 사용할 수 있다.

\

강제 접근통제 (MAC)의 보안 모델

벨 라파듈라 모델 (Bell-LaPadula Model)	○군대의 보안 레벨처럼 정보의 기밀성에 따라 상하 관계가 구분된 정보를 보호하기 위해 사용 ○보안 취급자의 등급을 기준으로 읽기 권한과 쓰기 권한이 제한 ○자신의 보안 레벨 이상의 문서를 작성할 수 있고, 자신의 보안 레벨 이하의 문서를 읽을 수 있음 예) 보안등급이 2급이면 2~3급은 조회 가능. 1~2급은 작성가능
비바 무결성 모델 (Biba Integrity Model)	○벨 라파듈라 모델을 보안한 수학적 모델로, 무결성을 보장하는 최초의 모델 ○비인가자에 의한 데이터 변형을 방지
클락-윌슨 무결성 모델 (Clark-Wilson)	○무결성 중심의 상업용 모델로, 사용자가 직접 객체에 접근할 수 없고 프로그램에 의해 접근이 가능한 보안 모델
만리장성 모델	서로 이해 충돌 관계에 있는 객체 간의 정보 접근을 통제하는 모델

접근통제 정책 : 어떤 주체가(Who) 언제(When), 어디서(Where), 어떤 객체(What)에게, 어떤 행위(How)에 대한 허용 여부를 정의하는 것으로, 3가지의 정책 있음

신분 기반 정책	○주체나 그룹의 신분에 근거하여 객체의 접근을 제한하는 방법으로, IBP와 GBP가 있다. - IBP(Individual-Based Policy) : 최소 권한 정책으로, 단일 주체에게 하나의 객체에 대한 허가를 부여한다. - GBP(Group-Based Policy) : 복수 주체에 하나의 객체에 대한 허가를 부여함
규칙 기반 정책	○주체가 갖는 권한에 근거하여 객ㅊ체의 접근을 제한. MLP와CBP가 있음 - MLP(Multi-Level Policy) : 사용자 및 객체별로 지정된 기밀 분류에 따른 정책 - CLP(Compartment-Level Policy) : 집단별로 지정된 기밀 허가에 따른 정책
역할 기반 정책	○GBP의 변형된 정책으로, 주체의 신분이 아니라 주체가 맡은 역할에 근거하여 객체의 접근을 제한하는 방법 예) 인사담당자

접근통제 매커니즘 : 정의된 접근통제 정책을 구현하는 기술적인 방법으로, 접근통제 목록, 능력 리스트, 보안 등급, 패스워드, 암호화가 있다.

1. 접근통제 목록(Access Control List) : 객체를 기준으로 특정 객체에 대해 ㅈ

어떤 주체가 어던 행위를 할 수 있는지를 기록한 문서이다
2. 능력 리스트(Capability List) : 주체를 기준으로 주체에게 허가된 자원 및 권한을 기록한 목록이다
3. 보안 등급(Security Level) : 주체나 객체 등에 부여된 보안 속성의 집합으로, 이 등급을 기반으로 접근 승인 여부가 결정된다.
4. 패스워드 : 주체가 자신임을 증명할 때 사용하는 인증 방법이다.
5. 암호화 : 데이터를 보낼 때 지정된 수신자 이외에는 내용을 알 수 없도록 평문을 암호문으로 변환하는 것으로, 무단 도용을 방지하기 위해 주로 사용

접근통제 보안 모델 : 보안 정책을 구현하기 위한 정형화된 모델
1. 기밀성 모델 : 군사적인 목적으로 개발된 최초의 수학적 모델로, 기밀성 보장이 최우선인 모델이다.
- 이 모델은 군대 시스템 등 특수 환경에서 주로 사용된다
- 제약 조건
○단순 보안 규칙 : 주체는 자신보다 높은 등급의 객체를 읽을 수 없다
○★(스타)-보안 규칙 : 주체는 자신보다 낮은 등급의 객체에 정보를 쓸 수 없다
○강한★(스타)-보안 규칙 :주체는 자신과 등급이 다른 객체를 읽거나 쓸 수 없다.

2. 무결성 모델 : 기밀성 모델에서 발생하는 불법적인 정보 변경을 방지하기 위해 무결성을 기반으로 개발된 모델
- 무결성 모델은 데이터의 일관성 유지에 중점을 두어 개발되었다
- 무결성 모델은 기밀성 모델과 동일하게 주체 및 객체의 보안 등급을 기반으로 한다
- 제약 조건
○단순 무결성 규칙 : 주체는 자신보다 낮은 등급의 객체를 읽을 수 없음
○★(스타)-무결성 규칙 : 주체는 자신보다 높은 등급의 객체에 정보를 쓸 수 없음

3. 접근통제 모델 : 접근통제 메커니즘을 보안 모델로 발전시킨 것으로, 대표적으로 접근통제 행렬이 있다.
○접근통제 행렬 : 임의적인 접근통제를 관리하기 위한 보안 모델로, 행은 주체, 열은 객체, 즉 행과 열로 주체와 객체의 권한 유형을 나타냄
- 행 : 주체로서 객체에 접근을 시도하는 사용자
- 열 : 객체로서 객체에 접근통제가 이뤄지는 테이블, 컬럼, 뷰, 등과 같은 데이터베이스 개체이다.
- 규칙 : 주체가 객체에 대하여 수행하는 입력, 수정, 삭제 등의 데이터베이스에 대한 조작이다.
※표로 나타내서 각 속성에 대한 접근 권한을 써놓은 것이다.

접근통제 조건 : 접근통제 매커니즘의 취약점을 보완하기 위해 접근통제 정책에 부가하여 적용할 수 있는 조건이다
1. **값 종속 통제** (Value-Dependent Control) : 일반적으로는 객체에 저장된 값에 상관없이 접근통제를 동일하게 허용하지만 객체의 저장된 값에 따라 다르게 접근통제를 허용해야 하는 경우 사용한다.
예) 납입한 금액에 따라 보안 등급이 설정되고, 그 보안 등급으로 접근통제
2. **다중 사용자 통제** (Multi-User Control) : 지정된 객체에 다수의 사용자가 동시에 접근을 요구하는 경우 사용
예) 다수결을 통해 접근 여부 결정
3. **컨텍스트 기반 통제** (Context-Based Control) : 특정 시간, 네트워크 주소, 접근 경로, 인증 수준 등에 근거하여 접근을 제어하는 방법으로, 다른 보안 정책과 결합하여 보안 시스템의 취약점을 보완할 때 사용
예) 영업시간에만 접근을 가능하게 하거나 함

감사추적 : 사용자나 애플리케이션이 데이터베이스에 접근하여 수행한 모든 활동을 기록하는 기능
- 감사 추적은 오류가 발생한 데이터베이스를 복구하거나 부적나 데이터 조작을 파악하기 위해 사용
- 감사 추적 시 실행한 프로그램, 사용자, 날짜 및 시간, 접근한 데이터의 이전 값 등 및 이후 값등이 저장된다.

섹션103 데이터베이스 백업 (C)
: 전산 장비의 장애에 대비하여 데이터베이스에 저장된 데이터를 보호하고 복구하기 위한 작업으로, 치명적인 데이터 손실을 막기 위해서는 데이터베이스를 정기적으로 백업해야한다.
- DBMS는 DB파괴 및 실행 중단이 발생하면 복구 기능을 제공한다

데이터베이스 장애 유형 : 데이터베이스 장애 유형을 정확히 파악하고 장애에 따른 백업 전략을 세워야 장애 발생 시 복구가 가능하다

- 사용자 실수 : 사용자의 실수로 인해 테이블이 삭제되거나 잘못된 트랜잭션이 처리된 경우
- 미디어 장애 : CPU, 메모리, 디스크 등 하드웨어의 장애나 데이터가 파손됨
- 구문 장애 : 프로그램의 오류나 사용 공간의 부족으로 인해 발생하는 장애
- 사용자 프로세스 장애 : 프로그램이 비정상적으로 종료되거나 네트워크 이상으로 세션이 종료되어 발생하는 오류
- 인스턴스 장애 : 하드웨어 장애, 정전, 시스템 파일 파손 등 비정상적인 요인으로 인해 메모리나 데이터베이스 서버의 프로세스가 중단된 경우

로그 파일 : 데이터베이스의 처리 내용이나 이용 상황 등 상태 변화를 시간의 흐름에 따라 모두 기록한 파일로, 데이터베이스 복구를 위해 필요한 가장 기본적인 자료이다

- 로그 파일을 기반으로 데이터베이스를 과거 상태로 복귀(UNDO)시키거나 현재 상태로 재생(REDO)시켜 데이터베이스 상태를 일관성 있게 유지할 수 있다
- 로그 파일은 트랜잭션 시작 시점, Rollback 시점, 데이터 입력, 수정 삭제 시점 등에서 기록된다
- **로그 파일 내용 :** 트랜잭션이 작업한 모든 내용, 트랜잭션 식별, 트랜잭션 레코드, 데이터 식별자, 갱신 이전 값, 갱신 이후 값 등

데이터베이스 복구 알고리즘 : 동기적/비동기적 갱신에 따라 나뉨

NO-UNDO/ REDO	○데이터베이스 버퍼의 내용을 비동기적으로 갱신한 경우의 복구 알고리즘 ○NO-UNDO : 트랜잭션 완료 전에는 변경 내용이 데이터베이스에 기록되지 않으므로 취소할 필요가 없음 ○REDO : 트랜잭션 완료 후 데이터베이스 버퍼에는 기록되어 있고, 저장되매체에는 기록되지 않으므로 트랜잭션 내용을 다시 실행해야한다.
UNDO/NO-REDO	○데이터베이스 버퍼의 내용을 동기적으로 갱신한 경우의 복구 알고리즘 ○UNDO : 트랜잭션 완료 전에 시스템이 파손되었다면 변경된 내용을 취소함 ○NO-REDO : 트랜잭션 완료 전에 데이터베이스 버퍼 내용을 이미 저장매체에 기록하였으므로 트랜잭션 내용을 다시 실행할 필요가 없음
UNDO/ REDO	○데이터베이스 버퍼의 내용을 동기/비동기적으로 갱신한 경우의 복구 알고리즘 ○데이터베이스 기록 전에 트랜잭션이 완료될 수 있으므로 완료된 트랜잭션이 데이터베이스에 기록되지 못했으면 다시 실행해야한다.
NO-UNDO/ NO-REDO	○데이터베이스 버퍼의 내용을 동기적으로 기록하지만 데이터베이스와는 다른 영역에 기록한 경우의 복구 알고리즘 ○NO-UNDO : 변경 내용은 데이터베이스와 다른 영역에 기록되어 있어 취소할 필요가 없다 ○NO-REDO : 다른 영역에 이미 기록되어 있어 트랜잭션을 다시 수행할 필요가 없다.

백업 종류 : 복구 수준에 따라서 운영체제를 이용하는 물리백업과 DBMS 유틸리티를 이용하는 논리백업으로 나뉜다.

- 물리 백업 : 데이터베이스 파일을 백업하는 방법으로, 백업 속도가 빠르고 작업이 단순하지만 문제 발생 시 원인 파악 및 문제 해결이 어려움
- 논리 백업 : DB 내의 논리적 객체들을 백업하는 방법으로, 복원 시 데이터 손상을 막고 문제 발생 시 원인 파악 및 해결이 수월하지만 백업/복원 시 시간이 많이 소요된다.

섹션104 스토리지

스토리지 : 단일 디스크로 처리할 수 없는 대용량의 데이터를 저장하기 위해 서버와 저장장치를 연결하는 기술이다.

- 종류에는 DAS, NAS, SAN 등이 있다

1. DAS (Direct Attached Storage) : 서버와 저장장치를 전용 케이블로 직접 연결하는 방식으로, 일반 가정에서 컴퓨터에 외장하드를 연결하는 것이 해당된다.
- 서버에서 저장장치를 관리한다
- 저장장치를 직접 연결하므로 속도가 빠르고 설치 및 운영이 쉽다
- 초기 구축 비용 및 유지보수 비용이 저렴하다
- 직접 연결 방식이므로 다른 서버에서 접근할 수 없고 파일을 공유할 수 없다
- 확장성 및 유연성이 상대적으로 떨어진다.
- 저장 데이터가 적고 공유가 필요 없는 환경에 적합하다

2. NAS(Network Attached Storage) : 서버와 저장장치를 네트워크를 통해 연결
 - 별도의 파일 관리 기능이 있는 NAS Storager가 내장된 저장장치를 직접 관리한다
 - Ethernet 스위치를 통해 다른 서버에서도 스토리지에 접근할 수 있어 파일 공유가 가능하고, 장소에 구애받지 않고도 저장장치에 쉽게 접근할 수 있다
 - DAS에 비해 확장성과 유연성이 우수하다
 - 접속 증가 시 성능이 저하될 수 있다.

3. SAN(Storage Area Network) : DAS의 빠른 처리와 NAS의 파일 공유 장점을 혼합한 방식으로, 서버와 장치를 연결하는 전용 네트워크를 별도로 구성하는 방식이다
 - 광 채널 (FC:Fiber Channer) 스위치를 이용하여 네트워크를 구성한다
 - 광 채널 스위치는 서버나 저장장치를 광케이블로 연결하므로 처리 속도가 빠름
 - 저장장치를 공유함으로써 여러 개의 저장장치나 백업 장비를 단일화 가능
 - 확장성, 유연성, 가용성이 뛰어남
 - 높은 트랜잭션 처리에 효과적이나 기존 시스템의 경우 장비의 업그레이드가 필요하고, 초기 설치 시에는 별도의 네트워크를 구축해야 하므로 비용이 많이 든다.

섹션105 논리 데이터 모델과 물리 데이터 모델 변환 (C)

테이블 : 데이터를 저장하는 데이터베이스의 가장 기본적이고 물리적인 오브젝트
 - 구성 요소 : 로우 / 컬럼 / 기본키 / 외래키

1. 엔티티(Entity)를 테이블로 변환 : 논리 데이터 모델에서 정의된 엔티티를 물리 데이터 모델의 테이블로 변환하는 것
 - 엔티티를 테이블로 변환한 후 테이블 목록 정의서를 작성함
  ※테이블 목록 정의서 : 전체 테이블을 목록으로 요약 관리하는 문서.
 - 변환 규칙 :
  엔티티 -〉 테이블
  속성 -〉 컬럼
  주 식별자 -〉 기본키
  외부 식별자 -〉 외래키
  관계 -〉 관계
 - 변환 시 고려 사항
  ○일반적으로 테이블과 엔티티 명칭은 동일하게 하는 것을 권고한다
  ○엔티티는 주로 한글명을 사용하지만 테이블은 소스 코드의 가독성을 위해 영문명을 사용한다
  ○메타 데이터 관리 시스템에 표준화된 용어가 있을 때는 메타에 등록된 단어를 사용해서 명명한다

2. 슈퍼타입/서브타입을 테이블로 변환 : 슈퍼타입/서브타입은 논리 데이터 모델에서 이용되는 형태이므로 물리 데이터 모델을 설계할때는 슈퍼타입/서브타입을 테이블로 변환해야한다.

1)슈퍼타입 기준 테이블 변환 : 서브타입을 슈퍼타입에 통합하여 하나의 테이블로 만드는 것.
 - 서브타입에 속성이나 관계가 적을 경우에 적용하는 방법으로, 하나로 통합된 테이블에는 서브타입의 모든 속성이 포함되어야 한다
 - 장·단점

장점	○데이터의 액세스가 상대적으로 용이하다 ○뷰를 이용하여 각각의 서브타입만을 액세스하거나 수정할 수 있다 ○서브타입 구분이 없는 임의 집합에 대한 처리가 용이하다 ○여러 테이블을 조인하지 않아도 되므로 수행 속도가 빨라진다 ○SQL 문장 구성이 단순해진다
단점	○테이블의 컬럼이 증가하므로 디스크 저장 공간이 증가한다. ○처리마다 서브타입에 대한 구분이 필요한 경우가 많이 발생한다 ○인덱스 크기의 증가로 인덱스의 효율이 떨어진다

2) 서브타입 기준 테이블 변환 : 슈퍼타입 속성들을 각각의 서브타입에 추가하여 서브타들을 개별적인 테이블로 만드는 것이다.
 - 서브타입에 속성이나 관계가 많은 경우 적용함
 - 장·단점

장점	○각 서브타입 속성들의 선택 사양이 명확한 경우 유리하다 ○처리할 때마다 서브타입 유형을 구분할 필요가 없다 ○여러 개의 테이블로 통합하므로 테이블당 크기가 감소해 전체 테이블 스캔 시 유리하다

단점	○수행 속도가 감소할 수 있다 ○복잡한 처리를 하는 SQL의 통합이 어렵다 ○부분 범위에 대한 처리가 곤란해진다 ○여러 테이블을 통합한 뷰는 조회만 가능하다 ○UID(Unique Indentifier, 식별자)의 유지 관리가 어렵다

3) 개별타입 기준 테이블 변환 : 슈퍼타입과 서브타입들을 각각의 개별적인 테이블로 변환하는 것

- 슈퍼타입과 서브타입 테이블들 사이에는 각각 1:1 관계가 형성된다.
- 개별타입 기준 테이블 변환을 적용하는 경우

○전체 데이터에 대한 처리가 빈번한 경우
○서브타입의 처리가 대부분 독립적으로 발생하는 경우
○통합하는 테이블의 컬럼 수가 많은 경우
○서브타입의 컬럼 수가 많은 경우
○트랜잭션이 주로 슈퍼타입에서 발생하는 경우
○슈퍼타입의 처리 범위가 넓고 빈법ㄴ하게 바랫ㅇ하여 단일 테이블 클러스터링이 필요한 경우

- 장·단점

장점	○저장 공간이 상대적으로 작음 ○슈퍼타입 또는 서브타입 각각의 테이블에 속한 정보만 조회하는 경우 문장 작성이 용이하다
단점	○슈퍼타입 또는 서브타입의 정보를 같이 처리하면 항상 조인이 발생하여 성능이 저하한다

3. 속성을 컬럼으로 변환

1) 일반 속성 변환

- 속성과 컬럼은 명칭이 반드시 일치할 필요는 없지만, 개발자와 사용자 간 의사소통을 위하여 가능 한 표준화된 약어를 사용하여 일치시키는 것이 좋음
- 컬럼명은 SQL의 예약어(Reserverd Word) 사용을 피한다
- 컬럼명은 SQL의 가독성을 높이기 위해 가능한 짧게 지정한다
- 복합 단어를 컬럼명으로 사용ㅎ랄 때는 미리 정의된 표준을 따른다
- 테이블의 컬럼을 정의한 후, 한 행에 해당하는 샘플 데이터를 작성해 컬럼의 정합성을 검증한다

2) Primary UID (주 식별자)를 기본키로 변환

3) Primary UID(관계의 UID BAR)를 기본키로 변환 : 다른 엔티티와의 관계로 생성된 Primary UID는 물리 데이터 모델의 기본키로 만듬

4) Secondary(Alternate) UID를 유니크키로 변환

4. 관계를 외래키로 변환 : 논리 데이터 모델에서 정의된 관계는 기본키와 이를 참조하는 외래키로 변환한다

- 1:1 관계 : 개체 A의 기본키를 개체 B의 외래키로 설정하거나 그 반대로해서 추가하여 표현한다
- 1:M 관계 : 개체 A의 기본키를 개체 B의 외래키로 설정하거나 별도의 테이블로 표현함 (기본키가 1이므로 M쪽의 외래키가 됨)
- N:M 관계 : 릴레이션 A와 B의 기본키를 모두 포함한 별도의 릴레이션으로 표현한다. 이때 생성된 별도의 릴레이션을 교차 릴레이션(Intersectiojn Relation) 혹은 교체 엔티티라고 한다. (즉 두 테이블의 기본키를 속성으로 가지는 그 관계만의 테이블이다)
- 1:M 순환 관계 : 개체 A에 개체 A의 기본키를 참조하는 외래키 컬럼을 추가하여 표현한다. 데이터의 계층 구조를 표현하기 위해 주로 사용한다

관리 목적의 테이블/컬럼 추가 : 논린 데이터 모델에는 존재하지 않는 테이블이나 컬럼을 데이터베이스의 관리 혹은 데이터베이스를 이용하는 프로그래밍의 수행 속도를 향상시키기 위해 물리 데이터 모델에 추가할 수 있다.

데이터 타입 선택 : 논리 데이터 모델에서 정의된 논리적인 데이터 타입을 물리적인 DBMS의 물리적 특성과 성능을 고려하여 최적의 데이터 타입과 데이터의 최대 길이를 선택한다

- 주요 타입에 문자 타입, 숫자 타입, 날짜 타입 등이 있음

※ VARCHAR2 타입은 가변길이 문자열 데이터타입

섹션106 물리 데이터 모델 품질 검토 (C)

: 물리 데이터 모델을 설계하고 데이터베이스 객체를 생성한 후 개발 단계로 넘어가기 전에 모델러와 이해관계자들이 모여 수행한다

- 물리 데이터 모델은 시스템 선응에 직접적인 영향을 미치므로 향후 발생할 문제에 대해 면밀히 검토해야 한다
- 물리 데이터 모델 품질 검토의 목적은 데이터베이스의 성능 향상과 오류 예방임
- 물리 데이터 모델을 검토하려면 모든 이해관계자가 동의하는 검토 기준이 필요

물리 데이터 모델 품질 기준

정확성	데이터 모델이 요구사항이나 업무 규칙, 표기법에 따라 정확하게 표현되었음
완전성	데이터 모델이 데이터 모델의 구성 요소를 누락 없이 정의하고 요구사항이나 업무 영역을 누락 없이 반영하였음
준거성	데이터 모델이 데이터 표준, 표준화 규칙, 법적 요건 등을 정확히 준수함
최신성	데이터 모델이이 최근의 이슈나 현행 시스템을 반영하고 있음
일관성	데이터 모델이 표현상의 일관성을 유지하고 있음
활용성	작성된 모델과 설명을 사용자가 충분히 이해할 수 있고, 업무 변화에 따른 데이터 구조의 변경이 최소화될 수 있도록 설계되었음을 의미

※조직 또는 업무 상황에 따라 가감하거나 변형하여 사용

물리 데이터 모델 품질 검토 항목 : 물리 데이터 모델의 특성을 반영한 품질 기준을 작성한 후 이를 기반으로 작성

- 물리 데이터 모델의 전반적인 것을 검토 항목으로 작성
- 체크리스트로 용이하게 품질검토를 하게 함

물리 데이터 모델의 품질 검토 순서

1. 데이터 품질 정책 및 기준 확인
2. 물리 데이터 품질의 특성에 따라 품질 기준 작성
3. 데이터 품질 기준에 따라 체크리스트 작성
4. 논리 데이터 모델과 물리 데이터 모델 비교
5. 각 모델링 단계의 모델러와 이해관계자가 품질 검토 수행
6. 체크리스트 내용을 종합하여 물리 데이터베이스 모델의 품질 검토 보고서 작성

3장 SQL 응용

섹션 107 SQL의 개념

SQL(Structured Query Language)의 개요

- 1974년 IBM 연구소에서 개발한 SEQUEL에서 유래한다
- 국제 표준 데이터베이스 언어이며, 많은 회사에서 관계형 데이터베이스(RDB)를 지원하는 언어로 채택하고 있다.
- 관계대수와 관계해석을 기초로 한 혼합 데이터 언어이다
- 질의어지만 질의 기능만 있는 것이 아니라 데이터 구조의 정의, 데이터 조작, 데이터 제어 기능을 모두 갖추고 있다

SQL의 분류

1. DDL(Data Define Language, 데이터 정의어)

- DDL은 SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다
- 논리적 데이터 구조와 물리적 데이터 구조의 사상을 정의한다
- 데이터베이스 관리자나 데이터베이스 설계자가 사용한다
- DDL의 세 가지 유형

명령어	기능
CREATE	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의
ALTER	TABLE에 대한 정의를 변경하는 데 사용
DROP	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 삭제

2. DML(Data Manipulation Language, 데이터 조작어)

- DML은 데이터베이스 사용자가 응용 프로그램이나 질의어를 통해서 저장된 데이터를 실질적으로 처리하는 데 사용되는 언어이다
- 데이터베이스 사용자와 데이터베이스 관리 시스템 간의 인터페이스를 제공한다
- DML의 네 가지 유형

명령어	기능
SELECT	테이블에서 조건에 맞는 튜플을 검색한다
INSERT	테이블에 새로운 튜플을 삽입한다
DELETE	테이블에서 조건에 맞는 튜플을 삭제한다
UPDATE	테이블에서 조건에 맞는 튜플의 내용을 변경한다

3. DCL(Data Control Language, 데이터 제어어)

- DCL은 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정의하는 데 사용되는 언어이다
- 데이터베이스 관리자가 데이터 관리를 목적으로 사용한다
- DCL의 네 가지 유형

명령어	기능
COMMIT	명령에 의해 수행된 결과를 실제 물리적 디스크로 저장하고, 데이터베이스 조작 작업이 정상적으로 완료되었음을 관리자에게 알려준다.
ROLLBACK	데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래의 상태로 복구한다.
GRANT	데이터베이스 사용자에게 사용 권한을 부여한다
REVOKE	데이터베이스 사용자의 권한을 취소한다

섹션 108 DDL

: DDL은 DB구조, 데이터 형식, 접근 방식 등 DB를 구축하거나 수정할 목적으로 사용하는 언어이다

- DDL은 번역한 결과가 데이터 사전이라는 특별한 파일에 여러 개의 테이블로서 저장된다
- DDL에는 CREATE SCHEMA, CREATE DOMAIN, CREATE TABLE, CREATE VIEW, CREATE INDEX, ALTER TABLE, DROP 등이 있다.

CREATE SCHEMA : 스키마를 정의

- 스키마의 식별을 위해 스키마 이름과 소유권자/허가권자를 정의함

"**CREATE SCHEMA** 스키마명 **AUTORIZATION** USER_ID;“

CREATE DOMAIN : 도메인을 정의

- 임의의 속성에서 취할 수 있는 값의 범위가 SQL에서 지원하는 전체 데이터 타입의 값이 아니고 일부분일 대, 사용자는 그값의 범위를 도메인으로 정의할 수 있다
- 정의된 도메인명은 일반적인 데이터 타입처럼 사용한다

"**CREATE DOMAIN** 도메인명 **[AS]** 데이터타입
    [DEFALT 기본값]
    [CONSTRAINT 제약 조건명 **CHECK** (범위값)];“

- 데이터 타입 : SQL에서 지원하는 데이터 타입
- 기본값 : 데이터를 입력하지 않았을 때 자동으로 입력되는 값

※보기힘든 데이터 타입들 : SMALLINT(2), REAL, DOUBLE PRECISION, DEC(i,j), VARCHAR, VARBIT 등등

CREATE TABLE : 테이블을 정의

"CREATE TABLE 테이블명
    (속성명 데이터_타입 [DEFAULT 기본값][NOT NULL],…
    [, PRIMARY KEY(기본키_속성명,…)]
    [, UNIQUE(대체키_속성명,…)]
    [, FOREIGN KEY(외래키_속성명,…)]
        [REFERENCES 참조테이블(기본키_속성명,…)]
        [ON DELETE 옵션]
        [ON UPDATE 옵션]
    [, CONSTRANT 제약조건명][CHECK (조건식)]); “

- 기본 테이블에 포함될 모든 속성에 대하여 속성명과 그 속성의 데이터 타입, 기본값, 널여부를 지정
- PRIMARY KEY로 사용할 속성 또는 속성 집합을 지정
- UNIQUE는 대체키로 중복된 값을 가질 수 없음
- FOREIGN KEY ~REFERENCES~ 는 참조할 다른 테이블과 그 테이블을 참조할 때 사용할 외래키 속성을 지정함.
- 또한 ON DELETE로 참조 테이블의 튜플이 삭제되었을 때 기본 테이블에 취해야할 사항을 지정한다. 옵션에는 NO ACTION, CASCADE, SET NULL, DEFALT가 있다
- ON UPADTE는 참조 테이블의 참조 속성 값이 변경되었을 때, 기본 테이블에 취해야할 사항을 지정. 옵션에는 NO ACTION, CASCADE, SET NULL, DEFALT가 있다.

○ NO ACTION : 참조 테이블에 변화가 있어도 아무런 조치를 취하지 않음

○ CASCADE : 참조 테이블의 튜플이 사라지면 기본 테이블의 관련 튜플도 모두 삭제되고, 속성이 변경되면 관련 튜플의 속성 값도 모두 변경된다

○ SET NULL : 참조 테이블에 변화 시 기본 테이블의 관련 튜플의 속성 값을 NULL로 바꿈

○ DEFALT : 참조 테이블에 변화 시 기본 테이블의 관련 튜플의 속성 값을 기본값으로 바꿈

- CONSTRANT : 제약 조건의 이름을 지정. 이름을 지정할 필요가 없으면 CHECK로 직접 명시함

CREATE VIEW : 뷰를 정의하는 명령문
"CREATE VIEW 뷰명[(속성명[,속성명,…])]
AS SELECT문 "
- SELECT문을 서브 쿼리로 사용하여 SELECT문의 결과로서 뷰를 생성한다
- 서브 쿼리인 SELECT문에는 UNION이나 ORDER BY 절을 사용할 수 없다
- 속성명을 기술하지 않으면 SELECT문의 속성명이 자동으로 사용된다

CREATE INDEX : 인덱스를 정의
" CREATE [UNIQUE] INDEX 인덱스명
ON 테이블명(속성명[ASC/DESC][,속성명[ASC/DESC]])
[CLUSTER]; "

ALTER TABLE : 테이블에 대한 정의 속성을 추가하거나 제거하거나 속성의 디폴트 값을 변경할 때 사용
"ALTER TABLE 테이블명 ADD 속성명 데이터_타입[DEFAULT '기본값'];"
"ALTER TABLE 테이블명 ALTER 속성명 [SET DEFAULT '기본값'];"
"ALTER TABLE 테이블명 DROP COLUMN 속성명 [CASCADE'];"

DROP : 스키마, 도메인, 기본 테이블 ,뷰 테이블, 인덱스, 제약 조건 등을 제거
"DROP SCHEMA 스키마명 [CASCADE|RESTRICT]"
"DROP DOMAIN 도메인명 [CASCADE|RESTRICT]"
"DROP TABLE 테이블명 [CASCADE|RESTRICT]"
"DROP VIEW 뷰명 [CASCADE|RESTRICT]"
"DROP INDEX 인덱스명 [CASCADE|RESTRICT]"
"DROP CONSTRAINT [CASCADE|RESTRICT]"

- DROP CONSTRAINT : 제약 조건 제거
- RESTRICT : 다른 개체가 제거할 요소를 참조중일 때는 제거를 취소

섹션 109 DCL

: DCL은 데이터의 보안, 무결성, 회복, 병행 제어 등을 정의하는 데 사용하는 언어이다
- DCL은 데이터베이스 관리자(DBA)가 데이터 관리를 목적으로 사용한다
- DCL에는 GRANT, REBOKE, COMMIT, ROLLBACK, SAVEPOINT 등 있다

GRANT/REVOKE : 데이터베이스 관리자가 데이터베이스 사용자에게 권한을 부여하거나 취소하기 위한 명령어
- **사용자 등급을 지정 및 해제하기 위한 명령어**

"GRANT 사용자등급 TO 사용자_ID_리스트 [IDENTIFIED BY 암호];"
"REBOKE 사용자등급 FROM 사용자_ID_리스트;"

※사용자 등급에는 3가지가 있음
1. DBA : 데이터베이스 관리자
2. RESOURCE : 데이터베이스 및 테이블 생성 가능자
3. CONNECT : 단순 사용자

- **테이블 및 속성에 대한 권한 부여 취소**

"GRANT 권한_리스트 ON 개체 TO 사용자 [WITH GRANT OPTION];"
"REVOKE [GRANT OPTION FOR] 권한_리스트 ON 개체 FROM 사용자 [CASCADE];'"

- 권한 종류 : ALL, SELECT, INSERT, DELETE, UPDATE, ALTER 등
- WITH GRANT OPTION : 부여받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한을 부여함
- GRANT OPTION FOR : 다른 사용자에게 권할을 부여할 수 있는 권한을 취소
- CASCADE : 권한 취소 시 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄적으로 취소

COMMIT : 트랜잭션이 성공적으로 끝나면 데이터베이스가 새로운 일관성 상태를 가지기 위해 변경된 모든 내용을 데이터베이스에 반영해야 하는데, 이때 사용하는 명령이 COMMIT 이다

- COMMIT 명령을 실행되지 않아도 DML문이 성공적으로 완료되면 자동으로 COMMIT되고, DML이 실패하면 자동적으로 ROLLBACK이 되도록 Auto Commit 기능을 설정할 수 있다

ROLLBACK : 아직 COMMIT 되지 않은 변경된 모든 내용들을 취소하고 데이터베이스를 이전 상태로 되돌림
- 트랜잭션 상태가 성공적으로 끝내지 못하면 데이터베이스에 비일관성인 상태를 가져다줄 수 있기 때문에 롤백해야함

SAVEPOINT : 트랜잭션 내에 ROLLBACK 할 위치인 저장점을 지정하는 명령어
- 저장점을 지정할때는 이름을 부여하며, ROLLBACK 시 저장된 저장점까지 트랜잭션 처리 내용이 취소된다.

섹션 110 DML

: DML은 데이터베이스 사용자가 응용 프로그래이나 질의어를 통해 저장된 데이터를 실질적으로 관리하는데 사용하는 언어이다
- DML은 데이터베이스 사용자와 데이터베이스 관리 시스템 간의 인터페이스를 제공한다

명령문	기능
SELECT	테이블에서 튜플을 검색
INSERT	테이블에 새로운 튜플 삽입
DELETE	테이블에서 튜플 삭제
UPDATE	테이블에서 튜플의 내용을 갱신

삽입문(INSERT INTO)
"INSERT INTO 테이블명([속성명1,속성명2,…])
VALUES(데이터1, 데이터2,…);“
- 대응하는 속성과 데이터는 개수와 데이터 유형이 일치해야함
- 기본 테이블의 모든 속성을 사용할 때는 속성명을 생략할 수 있다
- SELECT 문을 사용하여 다른 테이블의 검색 결과를 삽입할 수도 있음

삭제문(DELETE FROM~) : 테이블의 특정 튜플을 할 때 사용
"DELETE FROM 테이블명 [WHERE 조건];“
- 모든 튜플을 삭제하려면 WHERE절을 사용
- 모든 레코드를 삭제하더라도 테이블 구조는 남아있어 디스크에서 테이블을 완전히 제거하는 DROP과 다르다

갱신문 : 기본 테이블에 있는 튜플 중에서 특정 튜플의 내용을 변경할 때 사용
"UPDATE 테이블명 SET 속성명=데이터[,속성명=데이터,…] [WHERE 조건];“

섹션 111 DML - SELECT 1

일반 형식
SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭] [,[테이블명.]속성명,…]
[,그룹함수(속성명)[AS 별칭]]
[,Window함수 OVER(PARTITION BY 속성명1,속성명2,…
ORDER BY 속성명3, 속성명4,…)]
FROM 테이블명[,테이블명,…]
[WHERE 조건] [GROUP BY 속성명, 속성명,…] [HAVING 조건]
[ORDER BY 속성명[ASC|DESC]];

- PREDICATE : 불러올 튜플 수를 제한할 명령어를 기술
  ○ALL : 전부를 지정, 주로 생략
  ○DISTINCT : 중복된 튜플이 있으면 그 중 첫 번째 한 개만 검색
  ○DISTINCTROW : 중복된 튜플을 제거하고 한 개만 검색하지만 선택된 속성의 값이 아닌, 튜플 전체를 대상으로 한다. 즉 그 행 전체를 출력

- 속성명 : 검색하여 불러올 속성(열) 또는 속성을 이용한 수식을 지정
  모든 속성을 지정할 때는 * 사용
  두 개 이상 테이블을 대상으로 검색할 때는 테이블명.속성명으로 지정

- AS : 속성 및 연산의 이름을 다른 제목으로 표시하기 위해 사용
- FROM : 질의에 의해 검색될 데이터들을 포함하는 테이블 명을 기술
- WHERE : 검색할 조건 기술
- ORDER BY : 속성명으로 정렬하는 기능
- TOP n : 상위 n개 자료만 표시

※비교 연산자 중 〈〉 는 같지 않음을 나타냄
LIKE 연산자 : % - 모든 문자를 대표
_ - 문자 하나를 대표
- 숫자 하나를 대표

연산자 우선순위 산술>관계>논리

섹션 112 DML - SELECT 2

(위의 일반형식에서 추가)
- 그룹함수 : GROUP BY절에 저장된 그룹별로 속성의 값을 집계할 함수를 기술
- Window 함수 : GROUP BY절을 이용하지 않고 속성의 값을 집계할 함수를 기술
  ○ PARTITON BY : WINDOW 함수가 적용될 범위로 사용할 속성을 지정
  ○ ORDER BY : PARTITION 안에서 정렬 기준으로 할 속성 지정
- GROUP BY절 : 특정 속성을 기준으로 그룹화하여 검색할 때 사용한다. 일반적으로 그룹함수와 함께 사용함
- HAVING 절 : GROUP BY와 함께 사용되며, 그룹에 대한 조건을 지정

※그룹 함수
1. COUNT(속성명) : 그룹별 튜플 수
2. SUM(속성명) : 그룹별 합계
3. AVG(속성명) : 그룹별 평균
4. MAX(속성명) : 그룹별 최대값
5. MIN(속성명) : 그룹별 최소값
6. STDDEV(속성명) : 그룹별 표준편차
7. VARIANCE(속성명) : 그룹별 분산
8. ROLLUP(속성명, 속성명, …)
   - 인수로 주어진 속성을 대상으로 그룹별 소계를 구하는 함수
   - 속성의 개수가 n개이면, n+1레벨까지, 하위 레벨에서 상위 레벨 으로 데이터가 집계된다
9. CUBE(속성명, 속성명, …)
- ROLLUP과 유사한 형태이나 CUBE는 인수를 중진 속성을 대상으로 모든 조합의 그룹별 소계를 구한다
- 속성의 개수가 n개이면, 2^n레벨까지, 상위레벨에서 하위레벨로 데이터가 집계

※WINDOW 함수
- GROUP BY 절을 이용하지 않고 함수의 인수로 지정한 속성을 범위로 하여 속성의 값을 집계
- 함수의 인수로 지정한 속성이 대상 레코드의 범위가 되는데, 이를 윈도우(WINDOW)라 부름
1. ROW_NUMBER() : 윈도우별로 각 레코드에 대한 일련 번호를 반환
2. RANK(): 윈도우별로 순위를 반환하며, 공동순위 반영
3. DENSE_RANK() : 윈도우별 순위를 반환하며, 공동순위 무시하여 순위 부여

집합 연산자를 이용한 통합 질의
: 집합 연산자를 사용하여 2개 이상의 테이블의 데이터를 하나로 통합함
SELECT 속성명1, 속성명2,…
FROM 테이블명
UNION | UNION ALL | INTERACTION | EXCEPT
SELECT 속성명1, 속성명2, …
FROM 테이블명
[ORDER BY 속성명 [ASC|DESC]];

- 두 개의 SELECT 문에 기술한 속성들은 개수와 데이터 유형이 서로 동일해야 함

집합 연산자	설명	집합종류
UNION	두 SELECT문의 조회 결과를 통합하여 모두 출력 중복된 행은 한번만 출력	합집합
UNION ALL	두 SELECT문의 조회 결과를 통합하여 모두 출력 중복된 행도 그래로 출력	합집합
INTERSECT	두 SELECT문의 조회 결과 중 공통된 행만 출력	교집합
EXCEPT	첫 번째 SELECT문의 조회 결과에서 두 번째 SELECT문의 조회 결과를 제외한 행을 출력	차집합

섹션 113 DML - JOIN

: 2개의 테이블에 대해 연관된 튜플들을 결합하여, 하나의 새로운 렐레이션을 반환한다

- JOIN은 크게 INNER JOIN과 OUTER JOIN으로 구분된다
- JOIN은 일반적으로 FROM절에 기술하지만, 릴레이션이 사용되는 어느 곳이나 사용할 수 있다

INNER JOIN : 일반적으로 EQUI JOIN과 NON-EQUI JOIN으로 구분됨

- 조건이 없는 INNER JOIN을 수행하면 CROSS JOIN과 동일한결과를 얻을 수 있음

※CROSS JOIN은 두 테이블의 순서쌍을 결과로 반환함

EQUI JOIN

- EQUI JOIN은 조인 대상 테이블에서 공통 속성을 기준으로 '='(equal) 비교에 의해 같은 값을 가지는 행을 연결하여 결과를 생성하는 JOIN방법이다
- EQUI JOIN에서 JOIN 조건이 '='일 때 동일한 속성이 2번 나타나게 되는데, 이 중 중복된 속ㄱ성을 제거하여 같은속성을 한번만 표기하는 방법을 NATURAL JOIKN이라고 한다
- EQUI JOIN에서 연결 고리가 되는 공통 속성을 JOIN속성이라 한다.

- WHWER절을 이용한 EQUI JOIN의 표기형식

"SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1, 테이블명2, …
WHERE 테이블명1.속성명 = 테이블명2.속성명;“

-NATURAL JOIN을 이용한 EQUI JOIN의 표기형식

"SELECT [테이블명1.]속성명, [테이블명2.]속성명…
FROM 테이블명1 NATURAL JOIN 테이블명2;“

-JOIN ~USING절을 이용한 EQUI JOIN의 표기형식

"SELECT [테이블명1.]속성명, [테이블명2.]속성명…
FROM 테이블명1 JOIN 테이블명2 USING(속성명);“

NON-EQUI JOIN

- 조건에 '='이 아닌 다른 나머지 비교 연산자 〉,〈,〈〉,≤,≥를 사용하는 조인방법

표기형식

"SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1, 테이블명2, …
WHERE (NON-EQUI JOIN 조건);

OUTER JOIN : 릴레이션에서 JOIN 조건에 만족하지 않는 튜플도 결과를 출력하기 위한 JOIN방법으로 , LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN 등이 있다

- LEFT OUTER JOIN : INNER JOIN의 결과를 구한 후 , 우측 항 릴레이션의 어떤 튜플과도 맞지않는 좌측 항의 릴레이션에 있는 튜플들에 NULL값을 붙여서 INNER JOIN의 결과에 추가한다
  - 표기 형식

"SELECT [테이블명1.]속성명, [테이블명2.]속성명…
FROM 테이블명1 LEFT OUTER JOIN 테이블명2
ON 테이블명1.속성명 = 테이블명2.속성명;“

또는

"SELECT [테이블명1.]속성명, [테이블명2.]속성명…
FROM 테이블명1, 테이블명2
WHERE 테이블명1.속성명 = 테이블명2.속성명(+);“

- RIGHT OUTER JOIN : INNER JOIN의 결과를 구한 후 , 좌측 항 릴레이션의 어떤 튜플과도 맞지않는 우측 항의 릴레이션에 있는 튜플들에 NULL값을 붙여서 INNER JOIN의 결과에 추가한다
  - 표기 형식

"SELECT [테이블명1.]속성명, [테이블명2.]속성명…
FROM 테이블명1 RIGHT OUTER JOIN 테이블명2
ON 테이블명1.속성명 = 테이블명2.속성명;“

또는

"SELECT [테이블명1.]속성명, [테이블명2.]속성명…
FROM 테이블명1, 테이블명2
WHERE 테이블명1.속성명(+) = 테이블명2.속성명;“

FULL OUTER JOIN
- LEFT OUTER JOIN과 RIGHT OUTER JOIN을 합쳐 놓은 것
- 표기형식

"SELECT [테이블명1.]속성명, [테이블명2.]속성명…
FROM 테이블명1 FULL OUTER JOIN 테이블명2
ON 테이블명1.속성명 = 테이블명2.속성명;“

SLEF JOIN : 같은 테이블에서 2개의 속성을 연결하여 EQUI JOIN을 하는 조인 방법이다
- 표기 형식

"SELECT [별칭1.]속성명, [별칭1.]속성명,…
FROM 테이블명1 [AS] 별칭1 JOIN 테이블명1 [AS] 별칭2
ON 별칭1.속성명 = 별칭2.속성명;“

또는

"SELECT [별칭1.]속성명, [별칭1.]속성명,…
FROM 테이블명1 [AS] 별칭1, 테이블명1 [AS] 별칭2
WHERE 별칭1.속성명 = 별칭2.속성명;“

4장 SQL 활용

섹션 114 프로시저(Procedure) (C등급)

: 프로시저란 절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어로, 호출을 통해 실행되어 미리 저장해 놓은 SQL 작업을 수행한다
- 프로시저를 만들어 데이터베이스에 저장하면 여러 프로그램에서 호출하여 사용할 수 있다
- 프로시저는 데이터베이스에 저장되어 수행도기 때문에 스토어드(Stored)프로시저라고도 불린다
- 프로시저는 시스템의 일일 마감 작업, 일관(batch) 작업 등에 주로 사용된다

구성 요소
○ DECLARE : 프로시저의 명칭, 변수, 인수, 데이터 타입을 정의하는 선언부이다
○ BEGIN / END : 프로시저의 시작과 끝을 의미한다
○ CONTROL : 조건문 또는 반복문이 삽이되어 순차적으로 처리한다
○ SQL : DML, DCL이 삽입되어 데이터 관리를 위한 조회, 추가, 수정, 삭제 작업을 수행한다
○ EXCEPTION : BEGIN ~ END 안의 구문 실행 시 예외가 발생하면 이를 처리하는 방법을 정의
○ TRANSACTION : 수행된 데이터 작업들을 DB에 적용할지 취소할지를 결정하는 처리부

프로시저 생성 : CREATE PROCEDURE 명령어를 사용
- 표기 형식

```
CREATE [OR REPLACE] PROCEDURE 프로시저명(파라미터)
[지역변수 선언]
BEGIN
    프로시저 BODY;
END;
```

○ OR REPLACE : 선택적인(Optional) 예약어이다. 이 예약어를 사용하면 동일한 프로시저 이름이 이미 존재하는 경우, 기존의 프로서저를 대체할 수 있다
○ 파라미터 : 프로시저 파라미터로 다음 것들이 올 수 있음
- IN : 호출 프로그램이 프로시저에게 값을 전달할 때 지정
- OUT : 프로시저가 호출 프로그램에게 값을 전달할 때 지정
- INOUT : 호출 프로그램이 프로시저에게 값을 전달하고, 프로시저 실행 후 호출 프로그램에 값을 반환할 때 지정
- 또한 파라미터 뒤에 매개변수명과 자료형을 지정해야한다

○ 프로시저 BODY
- 프로시저의 코드를 기록하는 부분
- BEGIN에서 시작하며 END로 끝나면, BEGIN과 END 사에에는 적어도 하나의 SQL문이 있어야 한다

프로시저 실행 : 프로시저를 실행하기 위해서는 EXECUTE 명령어 또는 CALL 명

령어를 사용하며, EXECUTE 명령어를 줄여서 ECEC로 사용하기도 한다

- 표기형식

```
EXECUTE 프로시저명;
CALL 프로시저명;
EXEC 프로시저명;
```

프로시저 제거

```
DROP PROCEDURE 프로시저명;
```

섹션 115 트리거(Trigger)

: 데이터베이스 시스템에서 삽입, 갱신, 삭제 드의 이벤트가 발생할 때마다 관련 작업이 자동으로 수행되는 절차형 SQL이다

- 트리거는 데이터베이스에 저장되며, 데이터 변경 및 무결성 유지, 로그 메시지 출력 등의 목적으로 사용된다.

- 트리거의 구문에는 DCL을 사용할 수 없으며, DCL이 포함된 프로시저나 함수를 호출하는 경우에도 오류가 발생한다

- 트리거에 오류가 있는 경우 트러거가 처리하는 데이터에도 영향을 미치므로 트러거를 생성할 때 세심한 주의가 필요하다

트리거의 구성 : 트리거는 선언, 이벤트, 시작, 종료로 구성되며, 시작과 종료 구문 사이에는 제어(CONTROL), SQL, 예외(EXCEPTION)가 포함된다

- 트리거의 구성도

구성요소

○ DECLARE : 트리거의 명칭, 변수 및 상수, 데이터 타입을 정의하는 선언부

○ EVENT : 트리거가 실행되는 조건을 명시한다

○ BEGIN / END : 트리거의 시작과 종료를 의미한다

○ CONTROL : 조건문 또는 반복문이 삽입되어 순차적으로 처리된다

○ SQL : DML문이 삽입되어 데이터 관리를 위한 조회, 추가, 수정, 삭제 작업을 할 수 있음

○ EXCEPTION : BEGIN ~ END 안의 구문 실행 시 예외가 발생하면 이를 처리하는 방법을 정의

트리거의 생성 : CREATE TRIGGER 명령어를 사용

- 표기형식

```
CREATE [OR REPLACE] TRIGGER 트리거명 동작시기 동작 ON 테이블명
[REGERENCING NEW | OLD AS 테이블명]
[FOR EACH ROW [WHEN 조건식]]
BEGIN
    트리거 BODY;
END;
```

○ OR REPLACE : 선택적인 예약어. 이 예약어를 통해 동일한 이름의 트리거가 존재하는 경우, 기존의 트리거를 대체 가능

○ 동작시기 : 트리거가 실행될 때를 지정한다 종류에는 AFTER/BEFORE가 있음

- AFTER : 테이블이 변경된 후에 트리거가 작동
- BEFORE : 테이블이 변경되기 전에 트리거가 작동

○ 동작 : 트리거가 실행되게 할 작업의 종류를 지정. 종류에는 INSERT, DELETE, UPDATE가 있음

- INSERT : 테이블에 새로운 튜플을 삽입할 때 트리거가 작동
- DELETE : 테이블의 튜플을 삭제할 때 트리거가 실행
- UPDATE : 테이블의 튜플을 수정할 때 트리거가 실행

○ NEW | OLD : 트리거가 적용될 테이블의 별칭을 지정한다

- NEW : 추가되거나 수정에 참여할 튜플들의 집합(테이블)을 의미
- OLD : 수정되거나 삭제 전 대상이 되는 튜플들의 집합(테이블)을 의미

○ FOR EACH ROW : 각 튜플마다 트리거를 적용한다는 의미

○ WHEN 조건식 : 선택적인 예약어로 트리거를 적용할 튜플의 조건을 지정
○ 트리거 BODY : 트리거의 본문 코드를 입력하는 부분으로, 똑같이 BEGIN ~ END 안에 적어도 하나의 SQL문이 존재해야하고, 그렇지 않으면 오류 발생

※NEW OLD로 지정된 테이블 이름 앞에 :을 붙여 트리거 본문에서 사용함

트리거의 제거
DROP TRIGGER 트리거명;

섹션 116 사용자 정의 함수

: 사용자 정의 함수는 프로시저와 유사하게 SQL을 사용하여 일련의 작업을 연속적으로 처리하며, 종료 시 처리 결과를 단일값으로 반환하는 절차형 SQL이다
- 사용자 정의 함수는 데이터베이스에 저장되어 SELECT, INSERT, DELETE, UPDATE 등 DML문의 호출에 의해 실행된다
- 사용자 정의 함수는 예약어 RETURN을 통해 값을 봔한하기 때문에 출력 파라미터가 없음
- 사용자 정의 함수는 INSERT, DELETE, UPDATE를 통한 테이블 조작은 불가하지만, SELECT를 통한 검색·조회만 할 수 있다
- 사용자 정의 함수는 프로시저를 호출하여 사용할 수 없다
- 사용자 정의함수는 SUM(), AVG() 등의 내장 함수처럼 DML문에서 반환값을 활용하기 위한 용도로 사용된다

프로시저 V/S 사용자 정의 함수

구분	프로시저	사용자 정의 함수
반환값	없거나 1개 이상 가능	1개
파라미터	입·출력 가능	입력만 가능
사용 가능 명령문	DML, DCL	SELECT
호출	프로시저, 사용자 정의 함수	사용자 정의 함수
사용방법	실행문	DML에 포함

사용자 정의 함수의 구성 : 사용자 정의 함수의 구성은 프로시저와 유사하다. 프로시저 구성에서 RETURN만 추가하면 된다.

사용자 정의 함수 생성은 CREATE FUNCTION을 이용

```
CREATE [OR REPLACE] FUNCTION 사용자 정의 함수명(파라미터)
[지역변수 선언]
BEGIN
    프로시저 BODY;
    RETURN;
END;
```

- 사용자 정의 함수에서 파라미터는 IN밖에 없음 (출력을 하지 않기 때문)
- 지역변수 선언에서 리턴할 자료형을 입력함 EX) RETURN VARCHAR2

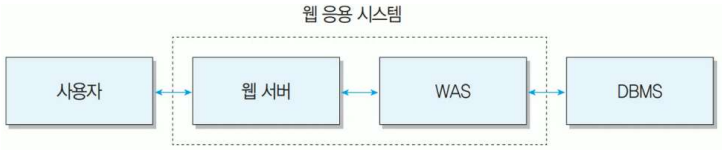
사용자 정의 함수의 실행 : 사용자 정의 함수의 실행은 DML에서 속성명이나 값이 놓일 자리를 대체하여 사용된다.

```
SELECT 사용자 정의 함수명 FROM 테이블명;
INSERT INTO 테이블명(속성명) VALUES(사용자 정의 함수명);
DELETE FROM 테이블명 WHERE 속성명 = 사용자 정의 함수명;
UPDATE 테이블명 SET 속성명 = 사용자 정의 함수명;
```

사용자 정의 함수 제거 : DROP FUNCTION 사용자 정의 함수명;

섹션 117 DBMS 접속 기술 (C등급)

: DBMS 접속은 사용자가 데이터를 사용하기 위해 응용 시스템을 이용하여 DBMS에 접근하는 것을 의미한다
- 응용 시스템은 사용자로부터 매개 변수를 전달받아 SQL을 실행하고 DBMS로부터 전달받은 결과를 사용자에게 전달하는 매게체 역할을 수행한다
- 인터넷을 통해 구동되는 웹 응용 프로그램은 웹 응용 시스템을 통해 DBMS에 접근한다
- 웹 응용 시스템은 웹 서버와 웹 애플리케이션 서버(WAS)로 구성되며, 서비스 규모가 작은 경우 웹 서버와 웹 애플리케이션 서버를 통합하여 하나의 서버만으로 운용할 수 있다.



- 사용자는 웹서버에 접속하여 데이터를 주고받음
- 웹 서버는 수많은 서비스 요청을 처리하기 때문에 사용자가 대용량의 데이터를 요청하면 직접 처리하지 않고 WAS에게 해당 요청을 전달
- WAS는 수신한 요청을 트랜잭션 언어로 변환 후 DBMS에 전달하여 데이터를 받습니다. 이렇게 받은 데이터는 처음 요청한 웹 서버로 다시 전달되어 사용자에게까지 도달하게 된다

DBMS 접속 기술 : DBMS에 접근하기 위해 사용되는 API 또는 API의 사용을 편리하게 도와주는 프레임워크 등을 의미함

1. JDBC (JAVA DATABASE Connectivity) : JAVA 언어로 다양한 종류의 데이터베이스 접속하고 SQL문을 수행할 때 사용되는 표준 API이다
- JDBC는 Java SE(Standard Edition)에 포함되어 있으며, JDBC 클래스는 java, sql, javax.sql에 포함되어 있다
- 접속하려는 DBMS에 대한 드라이버가 필요함

2. ODBC (Open DataBase Connectivity) : ODBC는 데이터베이스에 접근하기 위한 표준 개방형 API로, 개발 언어에 관계없이 사용할 수 있음
- 프로그램 내 ODBC 문장을 사용하여 MS-Access, DBASE, DB2, Excel, Text 등 다양한 데이터베이스에 접근할 수 있다
- ODBC도 접속하려는 DBMS에 맞는 드라이버가 필요하지만, 접속하려는 DBMS의 인터페이스를 알지 못하더라고 ODBC 문장을 사용하여 SQL을 작성하면 ODBC에 포함된 드라이버 관리자가 해당 DBMS의 인터페이스에 맞게 연결해주므로 DBMS의 종류를 몰라도 된다

3. MyBatis : JDBC 코드를 단순화하여 사용할 수 있는 SQL Mapping 기반 오픈 소스 접속 프레임워크이다
- JDBC로 데이터베이스에 접속하려면 다양한 메소드를 호출하고 해제해야 하는데, MyBatis는 이를 간소화 했고 접속 기능을 더욱 강화함
- MyBatis는 SQL 문장을 분리하여 XML 파일을 만들고, Mapping을 통해 SQL을 실행한다
- MyBatis는 SQL을 거의 그대로 사용할 수 있어 SQL 친화적인 국내환경에 적합

동적 SQL(Dynamic SQL) : 동적 SQL은 개발 언어에 삽입되는 SQL 코드를 문자열 변수에 넣어 처리하는 것으로, 조건에 따라 SQL 구문을 동적으로 변경하여 처리할 수 있다.
- 동적 SQL은 사용자로부터 SQL문의 일부 또는 전부를 입력받아 실행할 수 있음
- 동적 SQL은 값이 입력되지 않을 경우 사용하는 NVL 함수를 사용할 필요가 없음
- 동적 SQL은 응용 프로그램 수행 시 SQL이 변형될 수 있으므로 프리컴파일 할 때 구문 분석, 접근 권한 확인 등을 할 수 없다
- 동적 SQL은 정적 SQL에 비해 속도가 느리지만, 상황에 따라 다양한 조건을 첨가하는 유연한 개발이 가능하다

정적 SQL V/S 동적 SQL

	정적 SQL(static)	동적 SQL(dynamic)
SQL 구성	커서(Cursor)를 통한 정적 처리	문자열 변수에 담아 동적 처리
개발 패턴	커서의 범위 안에서 반복문을 활용하여 SQL 작성	NVL 함수 없이 로직을 통해 SQL 작성
실행 속도	빠름	느림
사전 검사	가능	불가능

섹션 118 SQL 테스트 (C등급)

: SQL 테스트는 SQL이 작성 의도에 맞게 원하는 기능을 수행하는지 검증하는 과정이다
- 단문 SQL은 SQL 코드를 직접 실행한 후 결과를 확인하는 것으로 간단히 테스트가 가능하다
- 절차형 SQL은 테스트 전에 생성을 통해 구문 오류(Syntax Error)나 참조 오류의 존재 여부를 확인한다
- 정상적으로 생성된 절차형 SQL은 디버깅을 통해 로직을 검증하고, 결과를 통해 최종적을 확인한다

※TCL : 트랜잭션 제어 언어

단문 SQL 테스트 : DDL, DML, DCL이 포함되어 있는 SQL과 TCL을 테스트 하는 것으로 , 직접 실행하여 결과물을 확인한다

- 실행 시 오류나 경고가 발생할 경우 메시지를 참조하여 문제를 해결한다
- **DESCRIBE** 명령어를 이용하면 DDL로 작성된 테이블이나 뷰의 속성, 자료형, 옵션들을 바로 확인할 수 있다
- DESC [개체명] ※정렬의 DESC에 유의
- DML로 변경한 데이터는 SELECT문으로 데ㅐ이터의 정상적인 변경 여부를 확인할 수 있다
- DCL로 설정된 사용자 권한은 사용자 권한 정보가 저장된 테이블을 SELECT로 조회하거나, SHOW 명령어로 확인할 수 있다

○ Oracle : SELECT * FROM DBA_ROLE_PRIVES WHERE GRANTEE = 사용자;

○ MySQL : SHOW GRANTS FOR 사용자@호스트(데이터베이스 주소);

절차형 SQL 테스트 : 프로시저, 사용자 정의 함수, 트리거 등의 절차형 SQL은 디버깅을 통해 기능의 적합성 여부를 검증하고, 실행을 통해 결과를 확인하는 테스트를 수행한다

- 많은 코드로 구성된 절차형 SQL의 특성상 오류 및 경고 메시지가 상세히 출력되지 않으므로, SHOW 명령어를 통해 오류 내용을 확인하고 문제를 수정한다

○형식 : SHOW ERRORS;

- 데이터베이스에 변화를 줄 수 있는 SQL문은 주석으로 처리하고, 출력문을 이용하여 화면에 출력하여 확인한다

○오라클 출력 형식 ※ 두 줄 모두 필요

: DBMS_OUTPUT.ENABLE; : 화면에 출력하기 위해 DBMS_OUTPUT 패키지를 볼러온다

: DBMS_OUTPUT.PUT_LINE(데이터); : '데이터'에 넣은 변수나 값을 화면에 출력한다

○MySQL 출력 형식

: SELECT 데이터; : '데이터'에 넣은 변수나 값을 화면에 출력한다

- 디버깅이 완료되면 출력문을 삭제하고, 주석 기호를 삭제한 후 절차형 SQL을 실행하여 결과를 검토한다

섹션 119 ORM(Object-Relational Mapping) (C등급)

: ORM은 객체지향 프로그래밍의 객체와 관계형 데이터베이스의 데이터를 연결(매핑)하는 기술을 의미한다

- ORM은 객체지향 프로그래밍에서 사용할 수 있는 가상의 객체지향 데이터베이스를 만들어 프로그래밍 코드와 데이터를 연결한다
- ORM으로 생성된 가상의 객체지향 데이터베이스는 프로그래밍 코드 또는 데이터베이스와 독립적이므로 재사용 및 유지보수가 용이하다
- ORM SQL코드를 직접 입력하지 않고 선언문이나 할당 같은 부수적인 코드가 생략되기 때문에 직관적이고 간단하게 데이터를 조작할 수 있다

ORM 프레임워크 : ORM을 구현하기 위한 구조와 구현을 위해 필요한 여러 기능들을 제하는 소프트웨어를 의미

- 종류

JAVA	JPA, Hibernate, EclipseLink, DataNucleus, Ebean 등
C++	ODB, QxOrm 등
Python	Django, SQLAlchemy, Storm 등
iOS	DatabaseObject, Core Data 등
.NET	NHibernate, DatabaseObjects, Dapper 등
PHP	Doctrine, Prope, RedBean 등

ORM의 한계

- ORM은 프레임워크가 자동을 SQL을 작성하기 때문에 의도대로 SQL이 작성되었는지 확인할 필요가 있다
- 객체지향적인 사용을 고려하고 설계된 데이터베이스가 아닌 경우 프로젝트가 크고 복잡해질수록 ORM 기술을 적용하기 어려워진다
- 기존의 기업들은 ORM을 고려하지 않은 데이터베이스를 사용하기 있기 때문에 ORM에 적합하게 변환하려면 많은 시간과 노력이 필요하다

섹션 120 쿼리 성능 최적화 (C등급)

: 쿼리 성능 최적화는 데이터 입·출력 애플리케이션의 성능 향상을 위해 SQL 코드를 최적화하는 것

- 쿼리 성능을 최적화하기 전에 성능 측정 도구인 APM을 사용하여 최적화할 쿼리를 선정해야함

- 최적화 할 쿼리에 대해 옵티마이저가 수립한 실행 계획을 검토하고 SQL코드와 인덱스를 재구성한다

※옵티마이저 : 작성된 SQL이 가장 효율적으로 수행되도록 최적의 경로를 찾아주는 모듈로, RBO와 CBO 두 종류가 있음

RBO(Rule Based Optimizer) v/s CBO (Cost Based Optimizer)

	RBO	CBO
최적화 기준	규칙에 정의된 우선순위	액세스 비용에 따른 순위
성능 기준	개발자의 SQL 숙련도	옵티마이저의 예측 성능
특징	실행 계획 예측이 쉬움	성능 통계치 정보활용, 예측이 복잡
고려사항	개발자의 규칙 이해도, 규칙의 효율성	비용 산출 공식의 정확성

실행 계획 (Execution Plan) : DBMS의 옵티마이저가 수립한 SQL 코드의 실행 절차와 방법을 의미한다 p533

- 실행 계획은 EXPLAIN 명령어를 통해 확인할 수 있으며, 그래픽이나 텍스트로 표현된다
- 실행 계획에는 요구사항들을 처리하기 위한 연산 순서가 적혀있으며, 연산에는 조인, 테이블 검색, 필터 정렬 등이 있다

쿼리 성능 최적화 : 실행 계획에 표시된 연산 순서, 조인 방식, 테이블 조회 방법 등을 참고하여 Sql문이 더 빠르고 효율적으로 작동하도록 SQL 코드와 인덱스를 재구성하는 것을 의미한다

1. SQL 코드 재구성
- WHERE 절을 추가하여 일부 레코드만 조회하게 함으로써 조회에 들어가는 비용을 줄임
- WHERE 절에 연산자가 포함되면 INDEX를 활용하지 못하므로 가능한 한 연산자 사용을 자제한다
- 서브 쿼리에 특정 데이터가 존재하는지 확인할 때는 IN보다 EXISTS를 활용한다
- 옵티마이저의 실행 계획이 잘못되었다고 판단될 경우 힌트를 활용하여 실행계획의 액세스 경로 및 조인 순서를 변경한다

2. 인덱스 재구성
- SQL 코드에서 조회하는 속성과 조건들을 고려하여 인덱스를 구성
- 실행 계획을 참고하여 인덱스를 추가ㅏ거나 기존 인덱스의 열 순서를 변경
- 인덱스의 추가 및 변경은 해당 테이블을 참조하는 다른 SQL문에도 영향을 줄 수 있으므로 신중히 결정
- 단일 인덱스로 쓰기나 수정 없이 읽기로만 사용되는 테이블의 경우
  IOT(Index- Organized Table)로 구성하는 것을 고려한다
- 불필요한 인덱스를 제거한다

5장 데이터 전환 (크게 중요하지 않음)

섹션 121 데이터 전환 (C등급)

: 운영 중인 기존 정보 시스템이 축적되어 있는 데이터를 추출하여 새로 개발할 정보 시스템에서 운영 가능하도록 변환한 후, 적재 하는 일련의 과정을 말함

- 데이터 전환은 ETL(Extraction, Transformation, load), 즉, 추출, 변환, 적재 과정이라고 한다
- 데이터 전환을 데이터 이행(Data Migration) 또는 데이터 이관이라고함

데이터 전환 계획서를 통해 데이터 전환이 필요한 대상을 분석하여 데이터 전환 작업에 필요한 모든 계획을 기록하는 문서로 주요 항목은 p542 참고

섹션 122 데이터 전환 계획서 작성(D등급)

: 데이터 전환 개요 항에는 데이터 전환의 목표, 성공적인 데이터 전환을 위한 주요 요인, 데이터 전환 작업을 위한 전제 조건 및 제약 사항을 기술

데이터 전환 대상 및 범위 : 데이터 전환 대상 및 범위에는 단위 업무별로 데이터 전환 대상 정보, 해당 업무에 사용되는 테이블 수, 데이터 크기등을 기술함

데이터 전환 환경 구성 : 원천 시스템과 목적 시스템의 구성도, 전환 단계별 DISK 사용량을 기술함

- 원천 시스템 구성도 : 원천 시스템의 서버, 스토리지, 네트워크를 포함한 구성도
- 목적 시스템 구성도 : 목적 시스템의 서버, 스토리지, 네트워크를 포함한 구성도
- 전환 단계별 DISK 용량 산정 : 전환 검증, 시험 단계, 본 전환 단계별로 요구

되는 파일 공간과 DB공간을 산정하여 기술한다

데이터 전환 조직 및 역할 작성 : 데이터 전환 조직 및 역할에는 데이터 전환을 수행하고 결과를 검증할 작업자와 작업자별 역할을 최대한 상세히 기술한다

데이터 전환 일정 작성 : 데이터 전환 및 검증 작업별로 상세하게 일정 수립하여 작성한다. 이때 도식화된 일정도 작성하여 포함한다

섹션 123 데이터 전환 방안 (D등급)

: 데이터 전환 방안 항목에는 데이터 전환 규칙, 데이터 전환 절차, 데이터 전환 방법, 데이터 전환 설계, 전환 프로글매 개발 및 테스트 계획, 데이터 전환 계획, 데이터 검증 방안이 있다

1. 데이터 전환 규칙 : 데이터 전환 괒엉에서 공통적으로 적용해야할 규칙들을 기술
2. 데이터 전환 절차 : 전환 준비, 전환 설계/개발, 전환 테스트, 실데이터 전환, 최종 전환 및 검증의 데이터 전환 절차를 체계적이고 상세히 기술. 데이터 흐름도를 작성해 작업의 이해를 높임
3. 데이터 전환 방법 : 단위 업무별로 데이터 전환 방법을 기술하되, 데이터 전환 시 업무별로 요구되는 전제 조건도 함께 기술한다.
4. 데이터 전환 설계 : 업무별로 전환 대상과 전환 제외 대상을 기술하고 천 시스템 테이블과 목적 시스템 테이블의 정의서를 작성한다
5. 전환 프로그램 개발 및 테스트 계획 : 테스트 계획 수립 후 관련 내용 기술
6. 데이터 전환 계획 : 데이터 전환 시간을 단축하기 위해 선 전환, 본 전환, 후 전환으로 분리하기 위해 계획을 수립한 후 관련 내용 기술
7. 데이터 검증 방안 : 데이터 전환 이후 전환 데이터의 정합성을 검증하고 전환 과정에서 발생할 수 있는 문제에 대응할 수 있도록 단계별 데이터 전환 검증 방안을 수립하고 관련 내용을 기술

섹션 124 데이터 검증 (C)

: 원천 시스템의 데이터를 목적 시스템이 데이터로 전환하는 과정이 정상적으로 수행되었는지 여부를 확인하는 과정

- 데이터 검증은 검증 방법과 검증 단계에 따라 분류할 수 있음

검증 방법에 따른 분류 : 로그 검증, 기본 항목 검증, 응용 프로그램 검증, 응용 데이터 검증, 값 검증이 있음

검증 단계에 따른 분류 : 원천 데이터를 추출하는 시점부터 전환 시점, DB 적재 시점, DB 적재 후 시점, 전환 완료 후 시점별로 그 목적과 검증 방법을 달리하여 데이터 전호나의 정합성을 검증한다

섹션 125 오류 데이터 측정 및 정제 (C등급)

: 고품질 데이터를 운영 및 관리하기 위해 수행됨

- 오류 데이터 측정 및 정제는 '데이터 품질 분석 -〉 오류 데이터 측정 -〉 오류 데이터 정제' 순으로 진행된다.

1. 데이터 품질 분석 : 오류 데이터를 찾기 위해 원천 및 목적 시스템 데이터의 정합성 여부를 확인하는 작업이다
2. 오류 데이터 측정 : 데이터 품질 분석을 기반으로 정상 데이터와 오류 데이터의 수를 측정하여 오류 관리 항목을 작성

- 정상 데이터는 전환 대상 범위의 데이터를 업무 영역별, 테이블별로 구분하여 수량을 측정 및 기재한다.
- 오류 데이터는 업무별로 오류 위치 및 유형을 확인하여 수량을 측정 및 기재한다.

3. 오류 데이터 정제 : 오류 관리 항목을 분석하여 원천 데이터를 정제하거나 전환 프로그램을 수정함

- 오류 데이터 분석을 한후 오류 데이터를 정제한다.

섹션 126 데이터 정제요청서 및 정제 보고서 (C등급)

: 원천 데이터의 정제와 전환 프르고름의 수정을 위해 요청사항 및 조치사항 등 데이터 정제와 관련된 전반적인 내용을 문서로 작성한 것

- 오류 관리 목록을 기반으로 데이터 정제 요건 목록을 작성하고 이 목록의 항목별로 데이터 정제요청서를 작성한다

1. 데이터 정제 요건 목록 작성 : 오류 관리 목록의 각 항목에 대해 정제 유형을 분류하고 현재의 정제 상태를 정의 - 정제 유형, 정제 방법, 상태를 기술

2. 데이터 정제요청서 작성 : 데이터 전환 시 발생한 오류의 수정을 위한 정제 요청의 전반적인 내용들을 작성하며, 데이터 정제 검토 시 신속한 의사 결정을 위해 오류사항의 해결 방안도 포함시킨다

3. 데이터 정제보고서의 개요 및 작성 : 데이터 정제 요청서를 통해 정제된 원천 데이터가 정상적으로 정제되었는지 확인한 결과를 작성한 것이다
 - 정제 요청서와 정제된 데이터 항목을 직접 육안으로 비교하여 확인한다