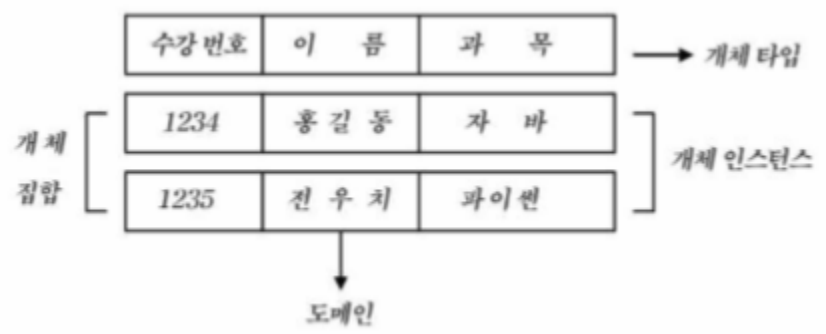
데이터베이스

데이터베이스는 조직의 여러 응용 시스템들이 공동으로 사용하기 위해 통합하고 저장한 데이터들로써 중복을 최소화하여 컴퓨터 기억 장치에 모아 놓은 집합체이다.

- 데이터베이스는 유용한 데이터의 집합으로 다양한 데이터들을 관리하기 위해서 데이터를 저장하고 사용자가 원하는 정보를 쉽게 찾을 수 있어야 한다.
- 데이터베이스는 조직에 필요한 정보를 얻기 위해 논리적으로 연관된 데이터를 모아 구조적으로 통합해 놓은 것이다.
- 데이터베이스가 유용한 데이터가 되기 위해서는 원하는 정보를 쉽게 얻거나 수정 삭제가 가능해야 한다.

데이터베이스의 구성



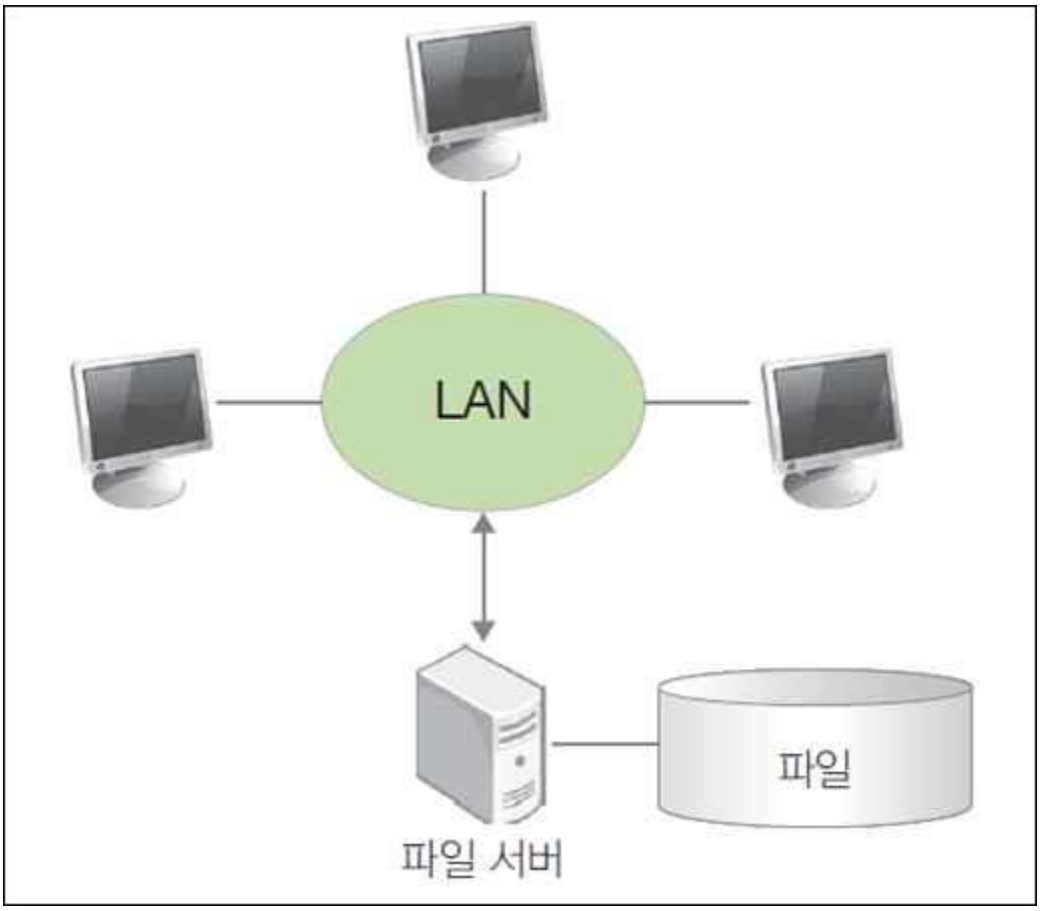
데이터베이스는 개별 객체인 개체(entity)를 가지고 있으며 개체의 형태는 다음과 같다.

- 도메인: 하나의 속성이 취할 수 있는 모든 값이다.
- 개체 인스턴스: 하나의 개체에 개체 값을 갖는 실제 레코드이다.
- 객체 집합: 개체 인스턴스의 집합이다.

데이터베이스는 개체 간에 여러 가지 상호 관계를 의미한 것으로 속성 관계, 객체 관계 등이 있으며 데이터베이스의 가장 단위인 속성(attribute)으로 관계들을 설명한다.
데이터베이스는 데이터에 대한 중복 최소화, 공유화, 독립성, 보안, 무결성을 목적으로 한다.

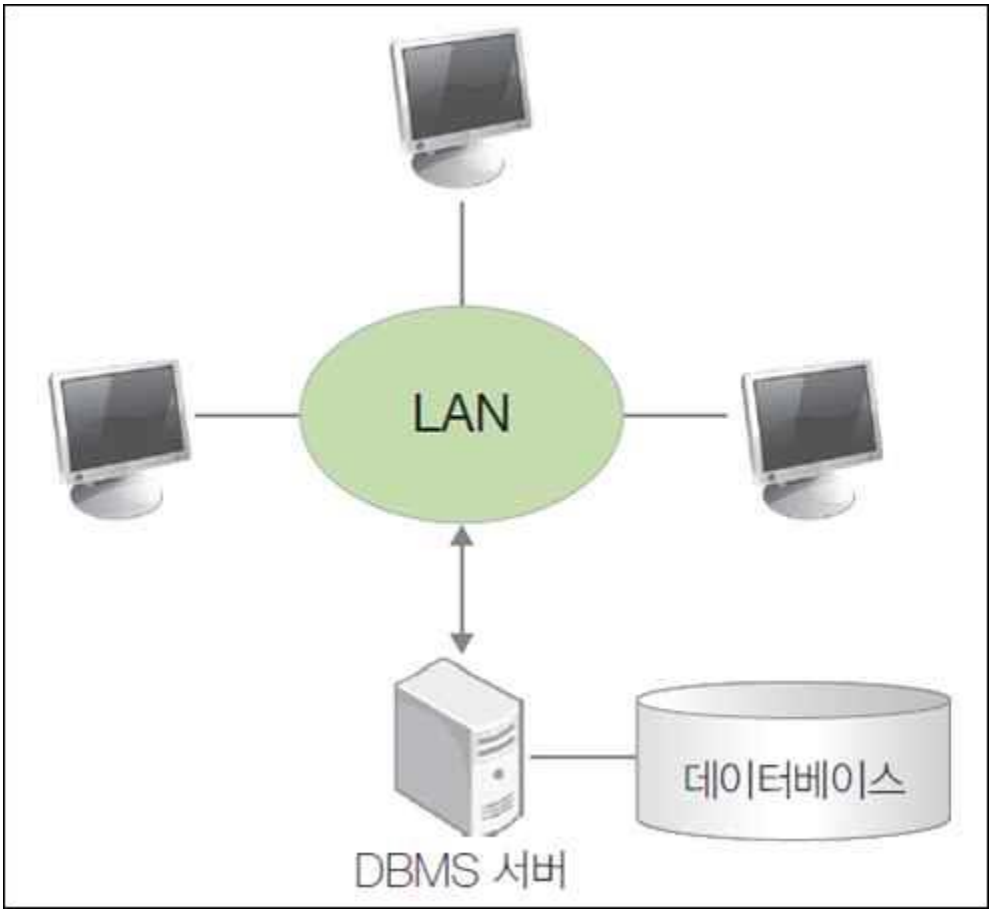
데이터베이스 시스템의 형태

1. 파일시스템



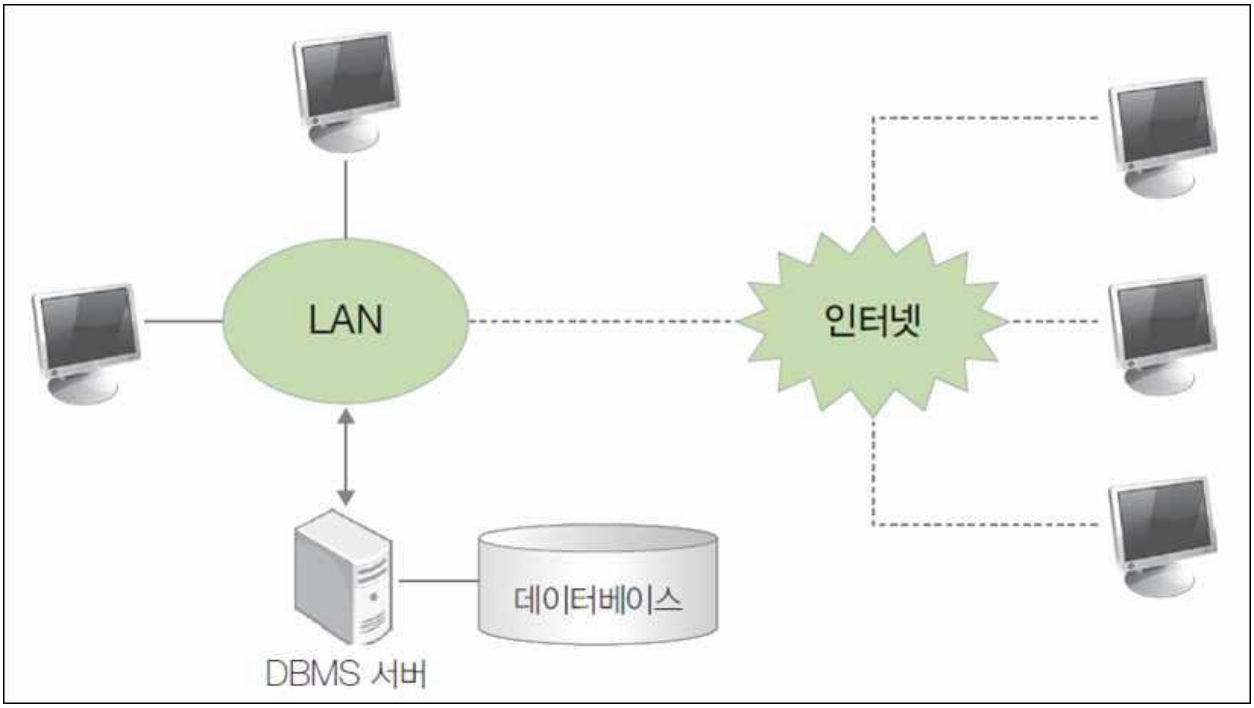
파일 시스템은 데이터를 파일 단위로 파일 서버에 저장한다.
개별 컴퓨터는 LAN(Local Area Network)을 통하여 파일 서버에 연결되어 있고 파일 서버에 저장된 데이터를 사용하기 위해 개별 컴퓨터의 응용 프로그램에서 요청한다.
응용 프로그램이 독립적으로 파일을 다루기 때문에 데이터가 중복으로 저장될 가능성이 있고 동시에 파일을 다루기 때문에 데이터의 일관성이 훼손될 수 있다.
소기업에서는 파일 시스템 사용하여 업무를 처리한다.

2. 데이터베이스 관리시스템



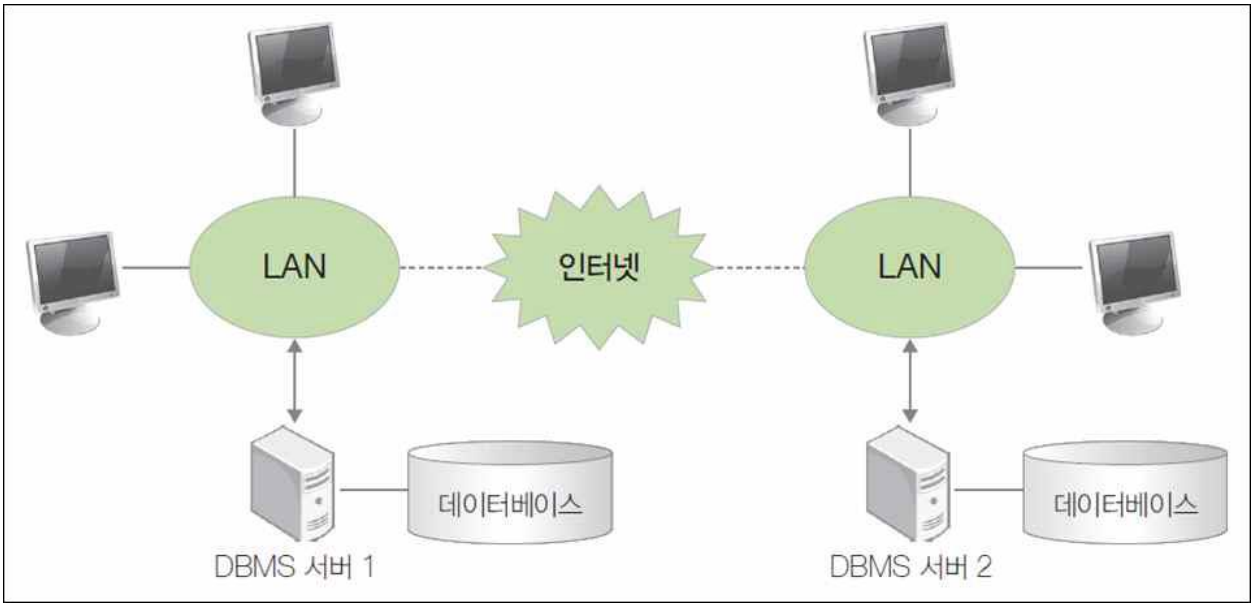
데이터베이스 관리시스템은 DBMS(DataBase Management System)라고 한다.
DBMS 서버를 도입하여 데이터를 통합 관리하는 시스템이다.
DBMS 서버는 파일을 관리하고 데이터의 일관성 유지, 복구, 동시 접근 제어 등의 기능을 수행한다.
DBMS 서버는 데이터를 저장하기 전 설계 과정을 거치기 때문에 데이터의 중복을 줄이고 데이터를 표준화하여 무결성을 유지한다.
중소기업에서는 데이터베이스 관리시스템 사용하여 업무를 처리한다.

3. 웹 데이터베이스 관리시스템



웹 데이터베이스 관리시스템은 웹 브라우저를 사용하여 데이터베이스를 관리하는 시스템이다.
불특정 다수 고객을 상대로 하는 온라인 상거래나 공공 민원 서비스 등에 사용된다.
대기업에서는 인터넷을 사용하여 실시간 서비스로 업무를 처리한다.

4. 분산 데이터베이스 관리시스템



분산 데이터베이스 관리시스템은 여러 곳에 분산된 DBMS 서버를 연결하여 운영하는 시스템이다.
분산 데이터베이스 관리시스템은 대규모의 응용 시스템에서 사용된다.
대기업에서 인터넷을 사용하여 고객 서비스와 업무를 처리한다.

관계형 데이터베이스 관리시스템 (Relational DataBase Management System, RDBMS)

RDBMS 소프트웨어는 많은 양의 데이터베이스를 편리하게 효율적으로 관리한다.
RDBMS 는 다양한 DBMS 중에서 가장 잘 알려진 소프트웨어로 일반적으로 RDBMS 를 줄여서 DBMS 라고도 한다.
RDBMS 는 자바와 같은 프로그램 언어로 만든 응용 프로그램과 데이터베이스의 중재자로서 모든 응용 프로그램들이 데이터베이스를 공유해서 사용할 수 있도록 관리해 주는 소프트웨어다.
RDBMS 는 데이터 중복을 최소화하고 데이터를 공유하므로 몇 개의 레코드를 동시에 수정하는 경우라도 오류의 발생률이 매우 낮다.
RDBMS 는 데이터의 무결성과 일관성을 가지고 있으며 데이터의 보안 보장성과 데이터의 표준화를 보장한다.
RDBMS 는 변화에 대한 준비성을 기반으로 한 데이터의 독립성을 보장한다.
RDBMS 는 데이터 처리의 복잡성으로 인해 복잡한 자료처리 방법이 요구되며 회복의 복잡성과 시스템의 취약성을 가지고 있다.
RDBMS 는 정형화된 데이터 항목들의 집합체로서 확장이 쉽다는 장점이 있는데 데이터베이스를 만든 후 관련되는 응용 프로그램들을 변경하지 않고도 새로운 데이터 항목을 데이터베이스에 추가할 수 있다.

RDBMS 는 2 차원 테이블 구조로 데이터를 관리하며 정보를 저장한다.
2 차원 테이블의 열을 컬럼(column)이라고 하고 행을 레코드(record)라고 한다.

RDBMS 의 대표적인 소프트웨어는 다음과 같다.
① Oracle ② MySQL ③ MS-SQL

RDBMS 의 기능

데이터 정의(Definition):
데이터의 구조를 정의하고 데이터 구조에 대한 삭제와 변경 기능을 수행한다.
다양한 응용 프로그램과 DB 가 서로 상호 접속하는 방법을 제공한다.

데이터 조작(Manipulation)
데이터를 조작하는 소프트웨어가 요청하는 데이터의 삽입, 수정, 삭제 작업을 지원한다.
사용자와 데이터베이스 간에 상호작용의 수단을 제공한다.

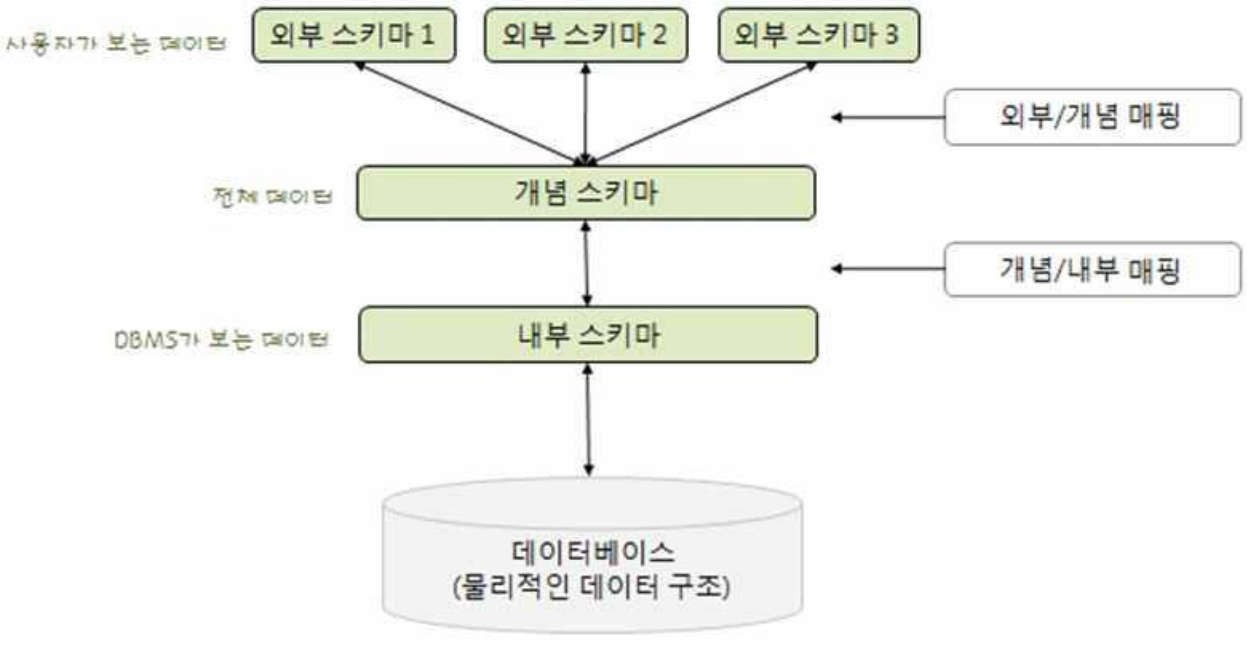
데이터 추출(Retrieval)
데이터를 조작하는 소프트웨어가 요청하는 데이터나 사용자가 조회하는 데이터를 추출한다.

데이터 제어(Control)
데이터베이스 사용자를 생성하고 접근을 제어하여 백업과 회복, 동시성 제어 등의 기능을 지원한다.
데이터의 정확성과 안전성을 유지한다.

데이터베이스 시스템의 구성

1. 스키마

스키마(schema)는 데이터베이스를 구성하는 파일, 레코드, 항목 등의 상호 관계 전체를 말한다.



외부 스키마

외부 스키마(external schema)는 외부 단계로 사용자가 직접 접근할 수 있는 바깥쪽의 스키마이다.

사용자의 관점에 따라 여러 개의 부분집합으로 나누어지는 것을 말하며 뷰의 개념이다.

사용자가 접근하는 계층으로 전체 데이터베이스 중에서 하나의 논리적인 부분을 의미한다.

개념 스키마

개념 스키마(conceptual schema)는 개념 단계로 논리적인 데이터베이스 전체의 구조를 말한다.

개념 스키마는 전체 데이터베이스의 정의로 통합 조직별로 하나만 존재한다.

개념 스키마는 데이터베이스 관리자(Database Administrator)가 관리하며 데이터베이스 관리자를 DBA 라고 한다.

개념 스키마는 저장장치에 독립적으로 기술되며 데이터와 관계, 제약사항, 무결성에 관한 내용이 포함된 데이터베이스 파일에 저장된 데이터의 형태를 나타낸다.

내부 스키마

내부 스키마(internal schema)는 내부 단계로 물리적인 데이터의 구조를 말한다.
내부 스키마는 기억 장치 내에 데이터가 저장된 데이터의 물리적인 설계도이다.
내부 스키마는 물리적 저장장치에 데이터베이스가 실제로 저장되는 방법의 표현이다.
내부 스키마는 하나며 인덱스, 레코드의 배치 방법, 데이터 압축 등에 관한 사항이 포함된다.

2. 매핑

매핑(mapping)은 키(key) 역할을 하는 데이터와 값(value) 역할을 하는 데이터를 하나씩 짝지어 저장하는 데이터 구조다.

외부/개념 매핑

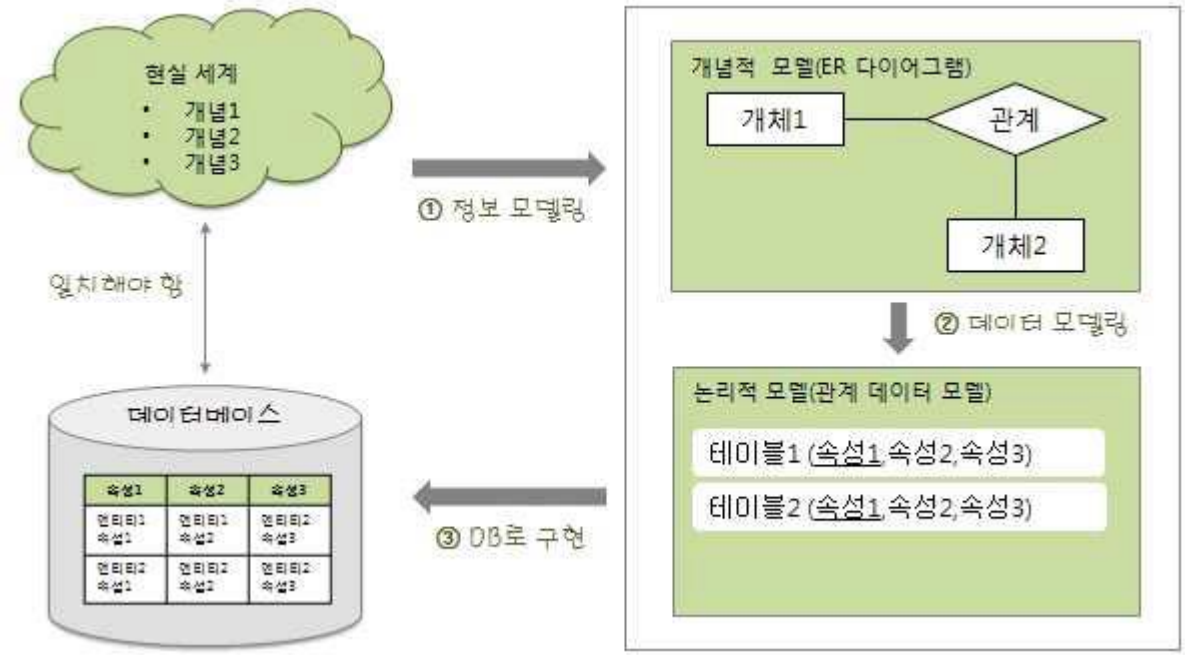
사용자의 외부 스키마와 개념 스키마 간의 매핑으로 외부 스키마의 데이터가 개념 스키마의 어느 부분에 해당하는지 대응시킨다.

개념/내부 매핑

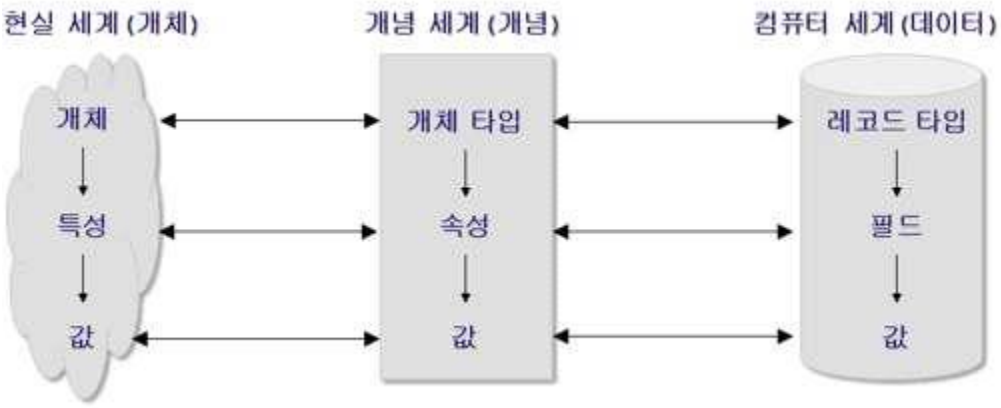
개념 스키마의 데이터가 내부 스키마의 물리적 장치 어디에 어떤 방법으로 저장되는지 대응시킨다.

데이터베이스의 모델링

데이터 모델링: 데이터베이스를 구성하는 데이터의 형태로 여러 가지 연산을 하는 것을 의미한다.



데이터 독립



데이터 독립은 논리적 데이터 독립과 물리적 데이터 독립으로 나누어진다.

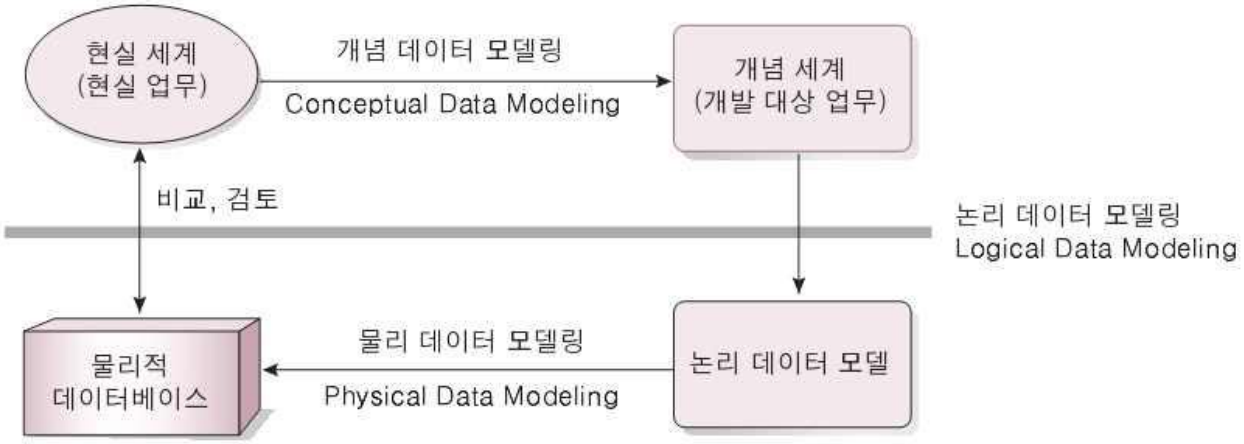
논리적 데이터 독립(logical data independence)
외부 단계와 개념 단계 사이의 독립성으로 개념 스키마가 변경되어도 외부 스키마에는 영향을 미치지 않도록 지원한다.
논리적 구조가 변경되어도 응용 프로그램에는 영향이 없도록 하는 개념이다.

물리적 데이터 독립(physical data independence)
개념 단계와 내부 단계 사이의 독립성으로 저장장치 구조 변경과 같이 내부 스키마가 변경되어도 개념 스키마에 영향을 미치지 않도록 지원한다.

물리적 독립성은 논리적 독립성보다 구현하기 쉽다.

데이터 모델

데이터 모델은 데이터베이스 내에 존재하는 데이터들 사이의 관계를 규정한다.
데이터 모델은 데이터의 의미와 데이터의 제약조건을 명시하기 위해 사용하는 개념적인 도구다.
데이터 모델은 데이터베이스에서 검색과 갱신을 수행하는 기본 연산들의 집합 뿐 아니라 점차 데이터베이스 응용의 동적 측면이나 행동도 데이터 모델에 포함하고 있다.



1. 개념 데이터 모델(Conceptual Data Model)

개념 데이터 모델은 속성들로 기술된 개체 타입과 이 개체 타입 간의 관계를 이용한다.
개념 데이터 모델은 데이터 표현의 논리적 성격에 초점을 맞추어 데이터를 개념적 단계와 뷰 단계에서 기술하는 것으로 융통성 있게 데이터를 구조화하고 데이터 제약조건을 분명히 지정할 수 있다.

1) E-R 모델(Entity-Relationship Model)

E-R 모델은 데이터베이스 구조의 논리적 구조와 물리적 구조를 표현하기 위해 제안된 모델이다.
데이터 구조와 데이터 간의 관계를 개체 집합(Entity Set)과 관계 집합(Relationship Set)으로 도형화 하여 개념적으로 표현한다.

2) 시멘틱 객체 모델(Semantic Object Model)

시멘틱 객체 모델은 E-R 모델과 마찬가지로 데이터 모델의 한 형식이다.
시멘틱 객체는 식별 가능한 어떤 사물을 표현하는 것으로 고유한 개체를 충분히 기술해 주는 속성의 이름을 가진 집합을 의미한다.
시멘틱 객체는 사용자에 의해 독립적이고 분리된 것으로 인식되며 사용자가 추적하고 기록하려는 것을 나타낸다.
시멘틱 객체 모델의 구성 요소는 시멘틱 객체, 속성, 시멘틱 객체 뷰 등이 있다.
시멘틱 객체 모델은 고유한 개체를 나타낼 수 있는 속성들의 집합으로 E-R 모델의 관계는 의미 객체 내부의 속성으로 표현된다.

3) 시멘틱 네트워크 데이터 모델(Semantic Network Data Model)

시멘틱 네트워크 데이터 모델은 인공지능 분야에서 지식을 표현하기 위해 개발된 방법으로 노드(Node)와 아크(Arc)로 구조를 표현한다.
노드는 사물을 개체 또는 인스턴스값으로 표현하고 아크는 사물 간의 관계 또는 단정(Assertion)을 표현한다.

E-R 모델이 단순히 개체 간의 연관 관계만을 표현한다면은 시멘틱 네트워크 데이터 모델은 일반화, 부품, 인스턴스, 단정 관계 등 더 많은 관계를 표현할 수 있다.

4) OMT 모델(Object Modeling Technique Model)

OMT 모델은 객체지향 소프트웨어 공학의 객체 모델링 기법이다.
OMT 모델은 객체 관계 데이터베이스에 적합한 E-R 모델을 확장한 형태다.

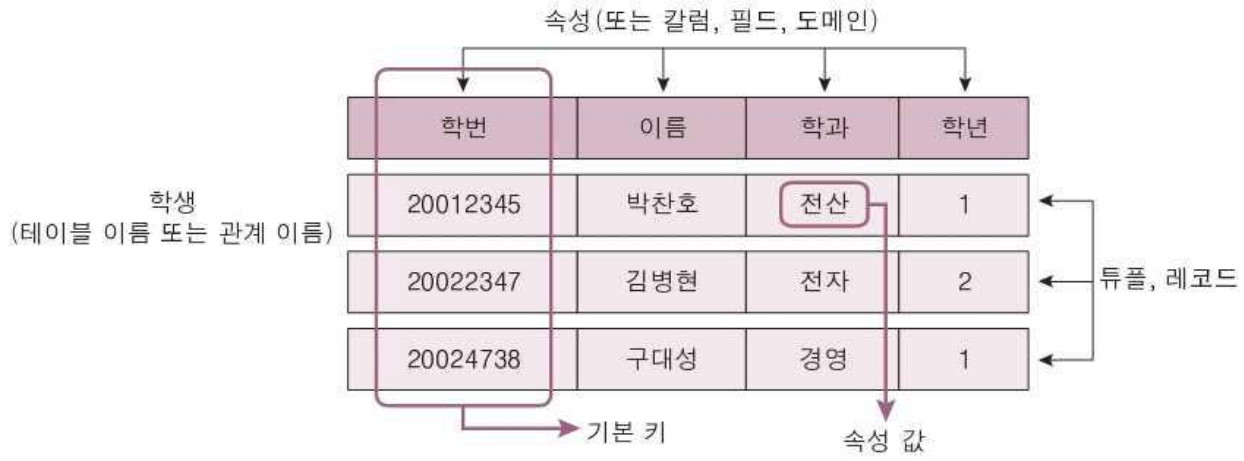
2. 논리 데이터 모델(Logical Data Model)

논리 데이터 모델은 논리적 데이터로 기술된 데이터 타입 간의 관계를 이용한다.
논리 데이터 모델은 개념적 단계와 뷰 단계에서 데이터를 기술하는 데 사용한다.
논리 데이터 모델은 데이터베이스의 전체적인 논리적 구조를 기술하는 데 사용한다.

1) 객체 데이터 모델(Object Data Model)

객체 데이터 모델은 개체의 관계를 통해서 모델링하므로 관계 데이터 모델이라고도 한다.
객체 데이터 모델은 단순하고 균일한 데이터 구조로 되어있어 많이 사용되고 있는 모델이다.
객체 데이터 모델은 개체의 모든 데이터와 데이터 사이의 관계를 2 차원의 테이블 형태로 기술한다.
객체 데이터 모델의 테이블에서 행은 하나의 개체를 나타내고 테이블의 열은 개체의 속성을 나타낸다.

(1) 객체 데이터 모델의 구성 요소



객체 데이터 모델의 구성 요소에 대한 특징은 다음과 같다.

- 릴레이션: 릴레이션은 테이블을 지칭한다.
- 속성: 속성은 관계 테이블의 열로 파일 처리의 필드와 같다.
- 튜플: 튜플은 관계 테이블의 행으로 파일 처리의 레코드와 같다.
- 도메인: 도메인은 개개의 속성값 집합으로 취하는 값의 범위를 의미한다.
- 인스턴스: 인스턴스는 행의 실제값으로 실제 레코드를 의미한다.

(2) 객체 데이터 모델의 특성

객체 데이터 모델의 사용자는 데이터를 테이블 형식으로 인식한다.
객체 데이터 모델의 테이블은 행과 열로 구분하며 행의 중복은 허락되지 않는다.
객체 데이터 모델의 행 순서는 임의로 설정하고 조회나 변경이 자유롭다.
객체 데이터 모델의 각 열은 하나의 값만 가지고 각 열의 속성은 하나이며 열의 순서는 무관하다.
객체 데이터 모델은 null 값을 허용하며 모든 속성의 값은 원자값이다.
객체 데이터 모델의 기본키인 주 식별자는 유일한 것이며 null 값을 가지지 않는다.
객체 데이터 모델은 참조 무결성과 영역 무결성을 가진다.
객체 데이터 모델은 높은 성능의 시스템 설비가 요구되며 설계 미숙 시 문제점이 발생할 확률이 매우 높다.

(3) 객체 데이터 모델의 장점

업무 변화에 따른 적응력이 탁월하며 시스템 설계가 단순하다.
구조가 탄력적이어서 필요할 때 테이블 사이의 연결을 통해 데이터를 생성할 수 있다.
데이터 간의 복잡한 관계를 개념적으로 간단하게 표현하며 강력한 데이터 조작 능력을 제공한다.
데이터의 첨가, 삭제, 수정이 쉽고 미래의 정보 요구에 신축성 있게 대응할 수 있다.

2) 계층 데이터 모델(Hierarchical Data Model)

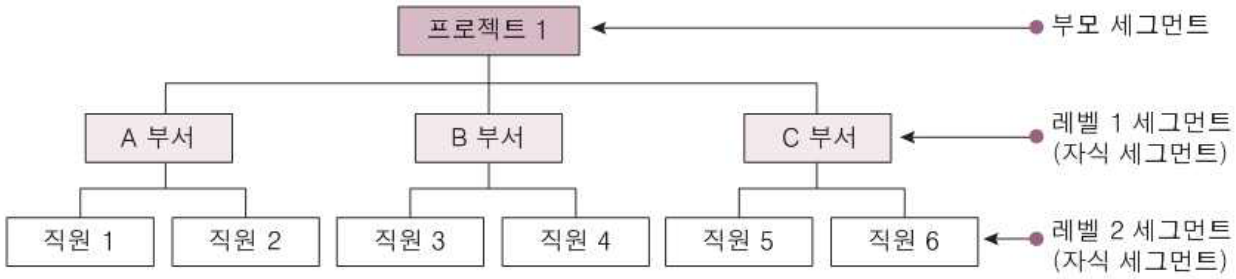
계층 데이터 모델은 대표적인 초기 데이터 모델이라고 할 수 있으며 논리적 구조가 계층 성질을 갖는 트리 형태의 자료구조로 표현된다.

계층 데이터 모델은 E-R 모델의 개체를 레코드 타입으로 표현한다.

계층 데이터 모델은 E-R 모델의 1:N 관계로 레코드 타입을 부모-자식 관계로 표현한다.

계층 데이터 모델에서 필드의 집합을 세그먼트(segment)라고 한다.

계층 데이터 모델의 계층 구조는 부모 세그먼트에서 시작하고 부모 요소 밑에는 여러 개의 자식 세그먼트가 있다.



계층 데이터 모델은 데이터의 처리를 신속하게 할 수 있으며 데이터 성능에 대한 예측이 쉽다.

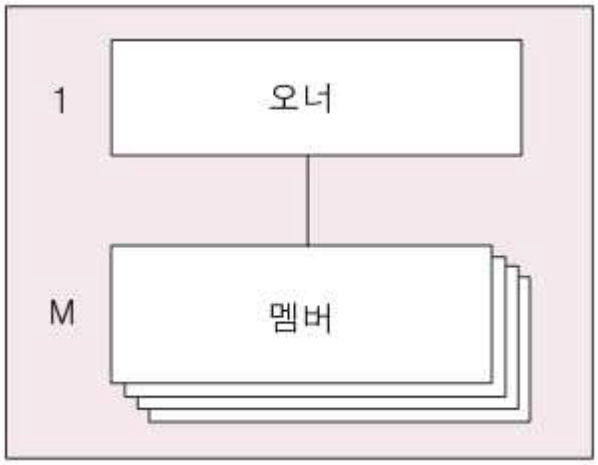
계층 데이터 모델은 업무 변화에 따른 대응력이 부족하며 단순한 작업에 능통한 전산 요원이 필요하다.

계층 데이터 모델은 일대일 관계가 매우 복잡하고 삽입과 삭제 등의 운영 면에서도 복잡성을 가지고 있으므로 삭제 시 주의가 필요하다.

3) 네트워크 데이터 모델(Network Data Model)

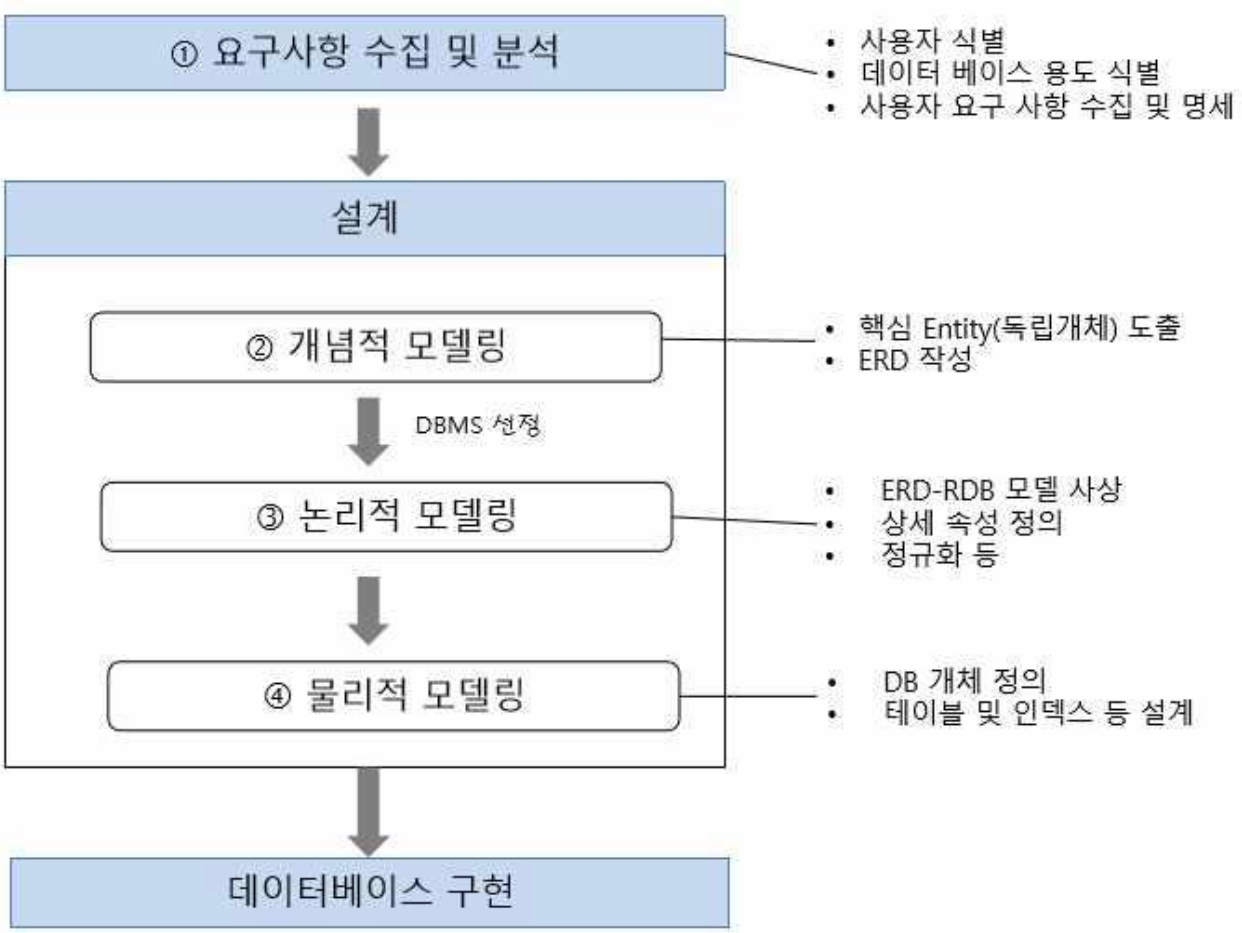
네트워크 데이터 모델은 데이터 간의 가능한 연결을 허용함으로써 계층 데이터 모델의 단점을 보완한 모델이다.

네트워크 데이터 모델은 데이터를 표현하는 데 있어 제한적인 트리 구조가 아닌 그래프 형태를 기반으로 한다.



네트워크 데이터 모델은 그래프 형태이므로 1:M 관계, M:M 관계 등의 표현이 가능하지만, 데이터의 종속성이 필요하다.

데이터 모델링의 흐름

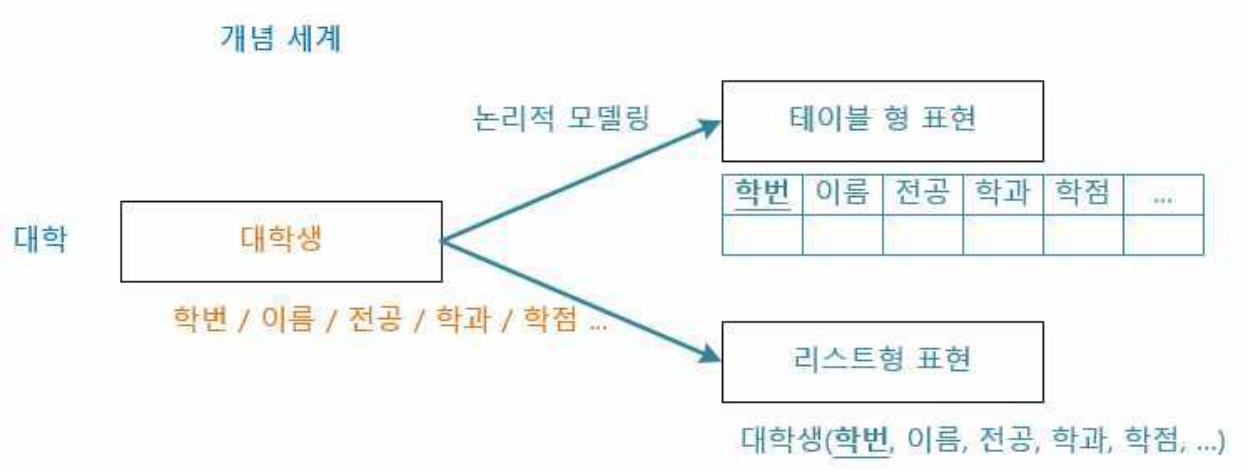

① 요구사항 수집 및 분석
• 사용자 식별
• 데이터 베이스 용도 식별
• 사용자 요구 사항 수집 및 명세
설계
② 개념적 모델링
• 핵심 Entity(독립개체) 도출
• ERD 작성
DBMS 선정
③ 논리적 모델링
• ERD-RDB 모델 사상
• 상세 속성 정의
• 정규화 등
④ 물리적 모델링
• DB 개체 정의
• 테이블 및 인덱스 등 설계
데이터베이스 구현

1. 논리적 모델링(Logical Modeling)

논리적 모델링은 사용하는 DBMS 의 종류에 맞게 변환한다.
논리적 모델링은 개념적 모델링에서 만든 ER 다이어그램을 사용하고자 하는 DBMS 에 맞게 매핑하여 실제 데이터베이스로 구현하기 위한 모델을 만드는 과정이다.
ER 다이어그램은 DBMS 의 특성에 따라 객체 데이터 모델, 네트워크 데이터 모델 등으로 매핑이 이루어진다.

논리적 모델링은 다음과 같은 과정을 수행한다.

- 개념적 모델링에서 추출하지 않았던 상세 속성들을 모두 추출한다.
- 정규화를 수행한다.
- 데이터의 표준화를 수행한다.

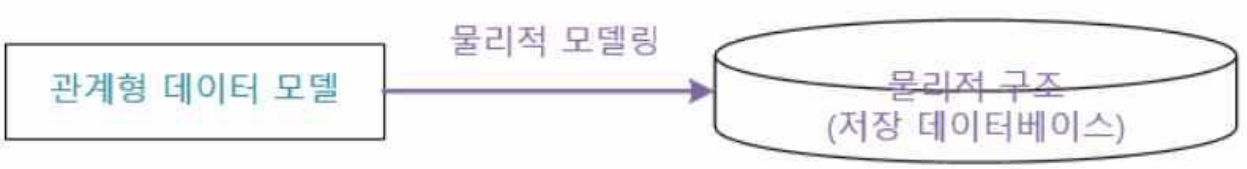


2. 물리적 모델링(Physical Modeling)

물리적 모델링은 데이터베이스 스키마를 도출한다.
물리적 모델링은 작성된 논리적 모델을 실제 컴퓨터의 저장장치에 저장하기 위한 물리적 구조를 정의하고 구현하는 과정이다.
물리적 모델링은 DBMS 의 특성에 맞게 저장구조를 정의해야 데이터베이스가 최적의 성능을 낼 수 있다.

물리적 모델링 시 트랜잭션, 저장 공간 설계 측면에서 고려할 사항은 다음과 같다.

- 응답시간을 최소화해야 한다.
- 얼마나 많은 트랜잭션을 동시에 발생시킬 수 있는지 검토해야 한다.
- 데이터가 저장될 공간을 효율적으로 배치해야 한다.



3. 데이터베이스 구현

설계 단계에서 생성한 스키마를 실제 DBMS 에 적용하여 테이블 및 관련 객체를 만든다.
관련 소프트웨어에 설계한 데이터베이스를 적용하여 서비스를 제공할 수 있도록 프로그램을 완성한다.
구현된 데이터베이스를 기반으로 소프트웨어를 구축하여 서비스를 제공한다.
데이터베이스 운영에 따른 시스템의 문제를 관찰하고 데이터베이스 자체의 문제점을 파악하여 개선한다.
데이터베이스가 지속해서 운영될 수 있도록 유지 보수한다.

JDK 설치

JDK 8 버전 설치
http://www.oracle.com/

윈도우 환경 변수 설정

환경 변수는 시스템의 실행 파일이 있는 디렉터리를 지정한다.
환경 변수는 운영체제에서 동작하는 응용 소프트웨어가 참조하기 위한 설정이 기록되는 곳이다.
자바의 JDK 를 구동하기 위해서는 환경 변수에서 설정을 해주어야 한다.

1. PC 아이콘 마우스 우클릭
2. 고급 시스템 설정
3. 고급 탭
4. 환경변수 버튼
5. 시스템 변수 에서 '새로 만들기'
6. 변수이름: JAVA_HOME
7. 디렉터리 찾아보기
8. 변수 값: jdk1.8.0_xxx 폴더 선택
9. 확인
10. Path 설정

Path 변수는 실행 디렉터리의 위치를 설정하는 곳이다.
Path 변수를 클릭하여 자바의 실행 디렉터리를 Path 에 등록한다.
일반적으로 프로그램의 실행 디렉터리의 이름은 binary 의 약자인 bin 으로 되어있다.

11. 시스템 변수 창에서 Path 클릭 후 편집 버튼을 누른다.
12. %JAVA_HOME%\bin; 추가

Windows 시스템
명령 프롬프트
Javac –version 으로 자바 컴파일러 버전 확인

데이터베이스의 프로그래밍

데이터베이스 프로그래밍은 RDBMS 에 데이터를 정의하고 저장된 데이터를 읽어와 데이터를 변경하는 프로그램을 작성하는 과정

데이터 정의어(DDL)

데이터 정의어는 데이터베이스를 구축하는데 필요한 매크로와 특수 명령어 군으로 이루어진 독립 언어로 논리적인 데이터의 정의, 기술을 위하여 사용된다.
데이터 정의어는 데이터 항목, 레코드, 세그먼트, 그룹 데이터 항목 등을 규정한다.
데이터 정의어는 키가 되는 데이터 항목을 규정하고 그 키가 유일한 것인지 중복된 것인지를 명시한다.
데이터 정의어는 세그먼트 간의 관계를 규정하고 이름을 부여한다.
데이터의 정의어는 스키마, 도메인, 테이블, 뷰, 인덱스를 정의하거나 제거하는 문으로 구성된다.
데이터의 정의어로 정의된 내용은 메타데이터가 되고 시스템 카탈로그에 저장된다.

데이터 조작어(DML)

데이터 조작어는 응용 프로그램이 데이터베이스에 명령하여 원하는 데이터를 사용자에게 제공할 수 있도록 한다.
데이터 조작어는 데이터의 전송, 편집, 연산 등의 제어를 담당한다.

데이터 질의어(DQL)

데이터 질의어는 데이터베이스에 대한 지식이 없는 일반 사용자들이 이용하는 언어이다.
데이터 질의어는 비절차어의 일종으로 자연어로 되어있어 대화식으로 데이터베이스를 쉽게 이용하게 되어있다.

SQL(Structured Query Language)

1. SQL 의 개요

SQL 은 데이터베이스에서 데이터의 검색, 데이터의 삽입, 데이터의 수정, 데이터의 삭제를 하기 위한 데이터베이스 질의 언어로 관계 대수나 관계 해석을 기초로 한 고급 데이터 언어이다.

SQL 은 ANSI (American National Standards Institute) 와 ISO (International Standards Organization) 에서 관계형데이터베이스의 표준언어로 채택되어 거의 모든 상용 관계 DBMS 에서 지원되고 있다.

SQL 은 구조화 질의이 뿐만 아니라 데이터의 정의, 데이터의 조작, 데이터의 제어 기능 등을 수행하고 관계형 DB 의 서브 언어로 데이터 정의어, 데이터 질의어, 데이터 조작어를 포함한다.

SQL 은 데이터 처리 작업의 단순화, 데이터 질의, 보고서 작성, 데이터 조작 기능을 제공한다.

SQL 을 잘 활용하면 데이터베이스의 데이터 관리를 최대한 효율적으로 이용할 수 있다.

데이터베이스 프로그래밍에서 중심적으로 다루는 SQL 은 데이터 질의어와 데이터 조작어이다.

SQL 로 접근 방법:
1) 단말기에서 대화식 처리: SQL 의 명령어로 검색, 삽입, 수정, 삭제 등의 작업 처리를 한다.
2) 응용 프로그램 이용 처리: 호스트 언어에서 SQL 언어를 삽입하여 사용한다.

2. SQL 의 필요한 구성

데이터베이스

데이터베이스는 나의 조직 안에서 다수의 사용자가 공동으로 사용하기 위하여 통합되고 저장된 운영 자료의 집합을 말하며 일반적으로 DB 라는 약자로 불린다.
데이터베이스는 원칙적으로 같은 내용의 자료가 중복되어 있지 않은 통합 자료를 관리한다.
데이터베이스는 컴퓨터에서 사용할 수 있는 보조 기억 장치에 자료를 저장하고 있다.

테이블

테이블은 실제 데이터를 저장하는 곳이다.
(1) 기본 테이블: 정의된 테이블로 독자적인 사용이 가능하다.
(2) 임시 테이블: 처리 과정에서 중간 결과로 만들어진 테이블이다.
(3) 가상 테이블: 어떤 테이블로 유도된 테이블로 뷰라고도 한다.

도메인

SQL 이 지원하는 데이터 타입으로 정의한다.
한번 정의된 도메인은 여러 테이블에서 공유해서 사용할 수 있다.
도메인을 정의하지 않고 테이블 정의 시 직접 데이터 타입을 정의해도 된다.

3. SQL 문

1) SQL 문의 개요

SQL 문은 관계형 데이터베이스에서 데이터베이스나 테이블을 생성하거나 데이터의 검색, 수정, 삭제, 입력 등을 하기 위해서 사용하는 언어를 의미한다.

2) SQL 문의 종류

(1) DDL (Data Definition Language)
DDL 은 데이터 정의어를 의미하며 사용하는 문은 다음과 같다.

- create 문은 데이터베이스나 테이블을 생성한다.
- alter 문은 데이터베이스나 테이블을 수정한다.
- drop 문은 데이터베이스나 테이블을 삭제한다.
- rename 문은 데이터베이스명이나 테이블명을 변경한다.
- truncate 문은 데이터베이스나 테이블의 저장 공간을 삭제한다.

(2) DQL (Data Query Language)
DQL 은 데이터 질의어를 의미하며 사용하는 문은 다음과 같다.
select 문은 데이터를 검색한다.

(3) DML (Data Manipulation Language)
DML 은 데이터 조작어를 의미하며 사용하는 문은 다음과 같다.

- insert 문은 데이터를 입력한다.
- update 문은 데이터를 수정한다.
- delete 문은 데이터를 삭제한다.

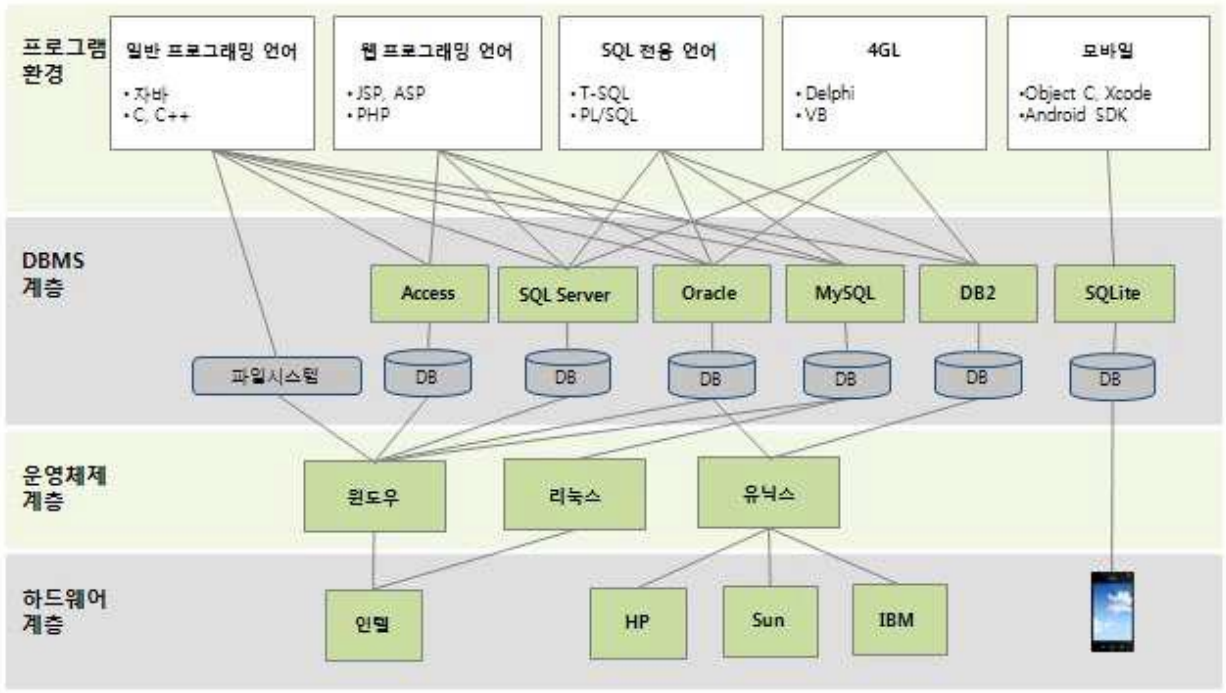
(4) TCL (Transaction Control Language)
TCL 은 데이터 처리어를 의미하며 데이터의 작업 처리인 트랜잭션에 적용한다.
TCL 에 사용하는 문은 다음과 같다.

- commit 명령어는 트랜잭션을 저장한다.
- rollback 명령어는 트랜잭션을 취소한다.
- savepoint 명령어는 트랜잭션 내의 책갈피 기능을 한다.

(5) DCL (Data Control Language)
DCL 은 데이터 제어어를 의미하며 사용하는 문은 다음과 같다.

- grant 명령어는 데이터베이스의 권한을 부여한다.
- revoke 명령어는 데이터베이스의 권한을 취소한다.

데이터베이스 프로그래밍의 방법



1) SQL 전용 언어를 사용하는 방법

SQL 자체의 기능을 확장하여 변수, 제어, 입출력 등의 기능을 추가한 새로운 언어를 사용하는 방법이다.

Oracle 은 PL/SQL 이라는 언어를 사용한다.

2) 프로그래밍 언어에 SQL 을 삽입하여 사용하는 방법

호스트 언어가 자바 등 프로그래밍 언어인 경우로 프로그래밍 언어로 작성된 응용 프로그램에서 데이터베이스에 저장된 데이터를 관리하고 조회한다.

삽입된 SQL 문은 RDBMS 의 컴파일러가 처리한다.

3) 웹 프로그래밍 언어에 SQL 을 삽입하여 사용하는 방법

호스트 언어가 JSP, ASP, PHP 등 웹 스크립트 언어인 경우다.

4) 4GL(4th Generation Language) 사용하는 방법

데이터베이스 관리 기능과 비주얼 프로그래밍 기능을 갖춘 GUI 기반 소프트웨어 개발 도구를 사용하여 프로그래밍하는 방법이다.

Delphi, Power Builder, Visual Basic 등이 있다.

오라클 데이터베이스

Oracle Express Edition

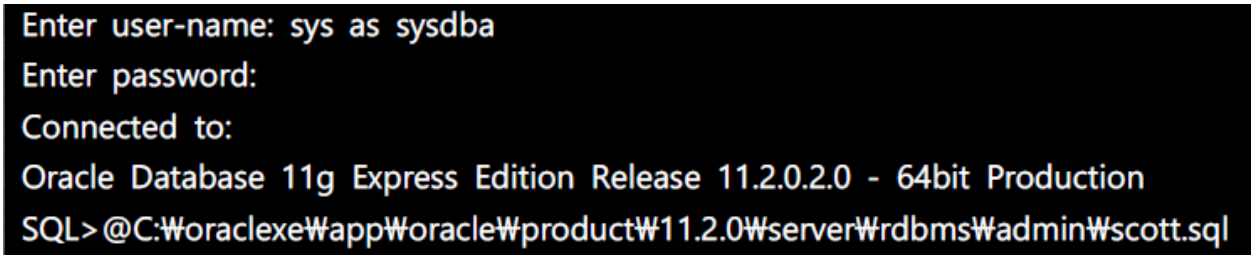
기능적 제한

- 서버당 하나의 인스턴스만 설치할 수 있다.
- CPU 가 여러 개가 있는 서버에 설치하더라도 하나의 CPU 만 사용하게 되어있다.
- 사용자 데이터가 최대 11GB 까지만 저장할 수 있다.
- 메모리는 최대 1GB 까지만 사용할 수 있다.

scott 계정은 Oracle Standard Edition 11g 에서는 제공하지만, Oracle Express Edition 11g 에서는 제공되지 않는다.

Oracle Express Edition 11g 에서 scott 계정을 사용하려면 수동으로 설치해야 한다.

C:\oraclexe\app\oracle\product\11.2.0\server\rdbms\admin 폴더에서 scott.sql 파일을 설치한다. 경로확인!!!!!!



SQL Developer
SQL Developer 에디터에서 최고 관리자인 sys 계정으로 접속하여 scott.sql 파일을 설치한다.

scott 계정에서 오라클을 실습할 수 있도록 실습 테이블을 제공

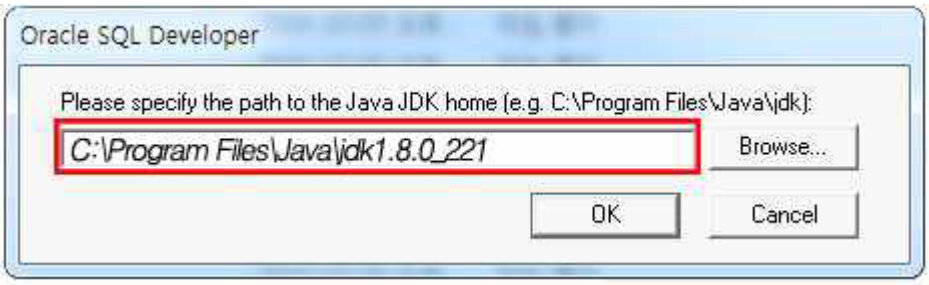
테이블	컬럼	널	데이터 타입
emp(사원)	사원 번호(empno)	not null	number(4)
	사원명(ename)		varchar2(10)
	직급(job)		varchar2(9)
	관리자 번호(mgr)		number(4)
	입사일(hiredate)		date
	급여(sal)		number(7, 2)
	커미션(comm)		number(7, 2)
	부서번호(deptno)		number(2)
dept(부서)	부서번호(deptno)	not null	number(2)
	부서명(dname)		varchar2(14)
	부서 위치(loc)		varchar2(13)
salgrade(급여등급)	등급(grade)		number
	최소월급(losal)		number
	최대월급(hisal)		number

SQL Developer

오라클 11g 의 통합 개발 환경은 Toad, SQLGate, SQL Developer 등이 있으며
오라클에서 무료로 배포하는 SQL Developer 를 사용

경로 추가

압축을 풀고 폴더 내의 sqldeveloper.exe( ) 파일을 관리자 권한으로 실행한다.
JDK 미포함 된 설치파일을 실행시켰음으로 JDK 설치한 곳의 경로를 입력한다.



사용자 접속
녹색의 십자가 표시 클릭

sys 계정 등록
sys 계정으로 접속하기 위해서는 접속에 필요한 내용을 입력하여 등록한다.
접속 유형에서 롤(L)을 SYSDBA 로 설정해야 한다.

scott 계정 등록
scott 계정으로 접속하기 위해서는 접속에 필요한 내용을 입력하여 등록한다.
접속 유형에서 롤(L)을 기본값으로 설정해야 한다.

오라클의 데이터 타입

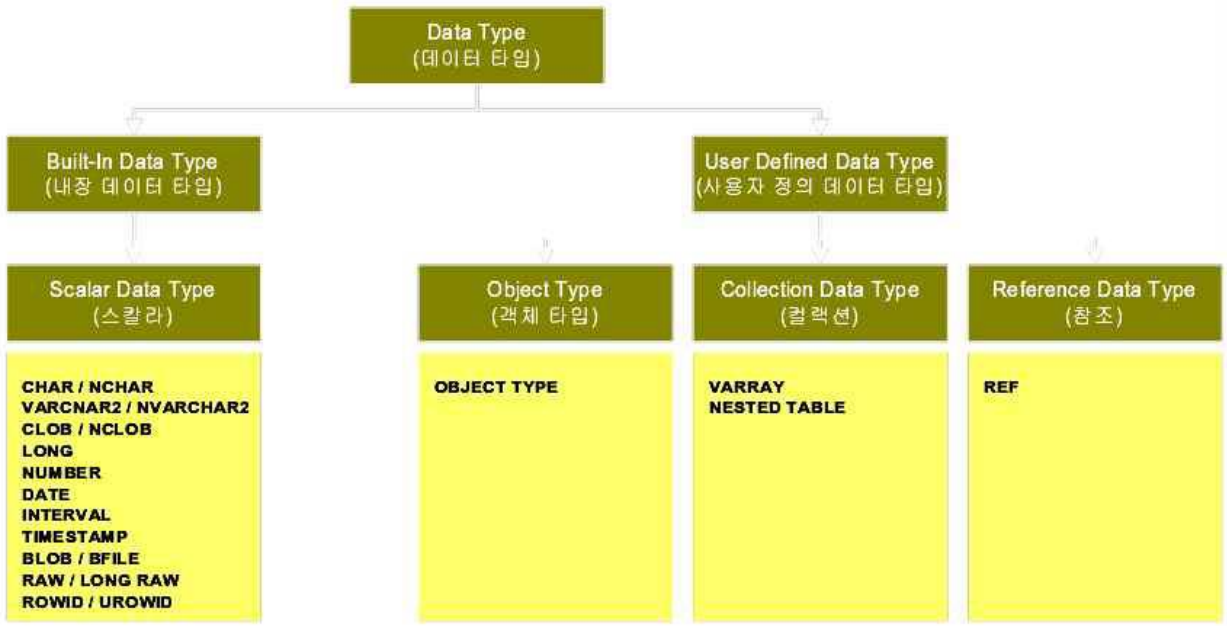
오라클의 데이터 타입은 테이블 컬럼을 속성을 정의한다.
오라클의 데이터 타입은 프로시저와 함수의 인자에 사용되는 값이 저장되는 방식을 결정한다.

오라클은 미국 국립 표준 협회(ANSI)에서 제공하는 데이터 타입을 지원하고 있다.

ANSI 타입	ORACLE 타입
character(n) / char(n)	char(n)
character varying(n) / char varying(n)	varchar2(n)
national characer(n) / national char(n) / nchar(n)	nchar(n)
national character varying(n) / national char varying(n) /nchar varying(n)	nvarchar2(n)
number(p, s) / decimal(p, s) (a)	number(p, s)
integer / int / smallint	number(38)
float(b) / double precision(c) / real(d)	number

데이터 타입의 형태

오라클의 데이터 타입은 내장 데이터 타입과 사용자 정의 데이터 타입으로 구분한다.



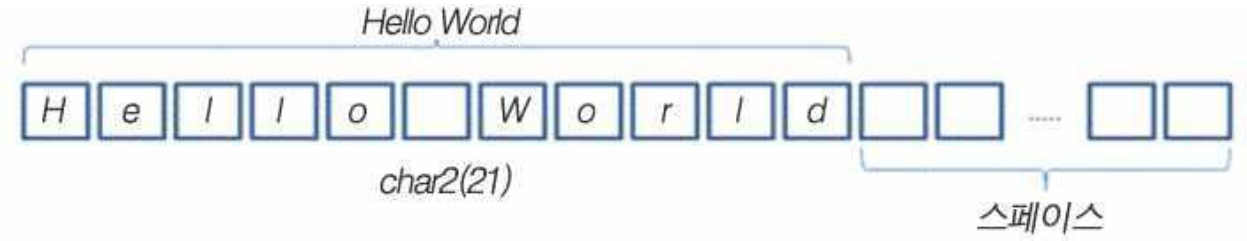
내장 데이터 타입(Built-In Data Type)

1. Character 데이터 타입

오라클 10g 버전 이하에서는 영문 이외의 문자는 2 바이트로 인식하지만, 오라클 11g 버전 이상에서는 영문 이외의 문자는 3 바이트로 인식한다.

1) char 데이터 타입

char 데이터 타입은 고정 길이 문자 데이터를 저장한다.
char 데이터 타입으로 문자 데이터를 정의할 때 지정된 길이만큼의 바이트로 저장한다.
문자 데이터를 문자 개수로 저장할 때는 char 데이터 타입이 아닌 nchar 데이터 타입으로 사용할 수 있다.
char 데이터 타입으로 문자 데이터를 저장할 때 공백 문자는 스페이스로 추가되어 저장된다.

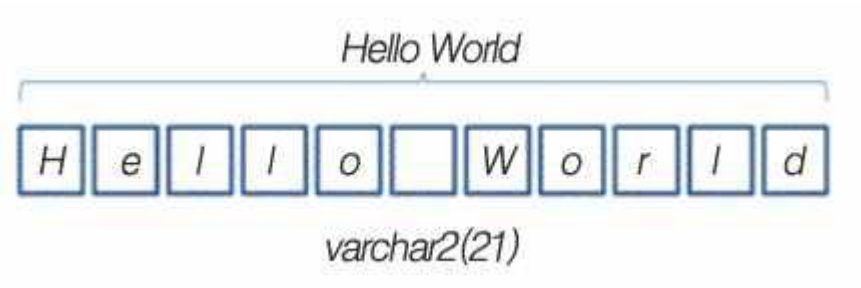


공백 문자를 스페이스로 대체하는 것은 정확한 비교를 하는 데 방해가 될 수 있다.
char 데이터 타입은 문자 개수와 상관없이 항상 지정된 바이트 수만큼 공간이 할당된다.
char 데이터 타입은 최대 2000 바이트까지 저장 가능하며 자릿수를 지정하지 않으면 기본 길이는 1 바이트이다.
char 데이터 타입은 공백 문자가 없는 고정 길이 문자 데이터일 경우에만 사용을 권장한다.

예) id char(6): id 컬럼은 영문은 6 글자를 저장할 수 있고 한글의 경우에는 2 글자를 저장할 수 있다.

2) varchar2 데이터 타입

varchar2 데이터 타입은 가변 길이 문자 데이터를 저장한다.
varchar2 데이터 타입으로 문자 데이터를 정의할 때 지정된 길이만큼의 바이트로 저장한다.
문자 데이터를 문자 개수로 저장할 때는 varchar2 데이터 타입이 아닌 nvarchar2 데이터 타입으로 사용할 수 있다.
char 데이터 타입으로 문자 데이터를 저장할 때 공백 문자는 스페이스로 추가되지 않고 실제 데이터만 저장된다.



varchar2 데이터 타입은 문자 개수와 상관없이 항상 지정된 바이트 수만큼 공간이 할당된다.
varchar2 데이터 타입은 최대 4000 바이트까지 저장 가능하며 자릿수는 반드시 지정해야 한다.
varchar2 데이터 타입은 입력되는 문자개수만큼만 공간이 할당되며 데이터 값이 유동적일 때 사용을 권장한다.

예) job varchar2(9): job 컬럼은 영문은 9 글자를 저장할 수 있고 한글의 경우에는 3 글자를 저장할 수 있다.

3) clob 데이터 타입

clob 데이터 타입은 변동 길이 문자 데이터를 최대 4 기가까지 저장한다.
clob 데이터 타입은 문자 데이터를 정의할 때 지정된 길이만큼의 바이트로 저장한다.
문자 데이터를 문자 개수로 저장할 때는 clob 데이터 타입이 아닌 nclob 데이터 타입으로 사용할 수 있다.
clob 데이터 타입은 기본적으로 insert 문에서는 수행한다.
clob 데이터 타입에서 select 문이나 update 문에서 수행하기 위해서는 별도의 PL/SQL 패키지이나 API 를 사용해야 한다.

4) long 데이터 타입

long 데이터 타입은 clob 데이터 타입 이전에 사용되던 변동 길이 문자 데이터로 최대 2 기가까지 저장한다.
long 데이터 타입은 하위 버전에 대한 호환성을 위해 존재하며 clob 데이터 타입을 사용하는 것을 권장한다.

2. Number 데이터 타입

형식: number([전체자릿수], [소수점자릿수])

number 데이터 타입은 수치 데이터를 저장하기 위한 데이터 타입이다.
number 데이터 타입은 변동 길이의 숫자 데이터를 저장하는 데 사용된다.
number 데이터 타입은 정수값과 실수값을 저장할 때 최대 21 바이트까지 사용할 수 있다.
number 데이터 타입은 지정된 자릿수만큼 공간이 할당된다.
number 데이터 타입에서 정수값을 저장하기 위해서는 소수점자릿수를 생략하면 된다.
number 데이터 타입에서 전체자릿수와 소수점자릿수 모두 생략하면 입력한 데이터 값만큼 공간이 할당된다.

3. Date 데이터 타입

1) date 데이터 타입

date 데이터 타입은 날짜와 시간을 고정 길이로 저장하기 위한 데이터 타입이다.
date 데이터 타입의 데이터 길이는 고정된 7 바이트이다.
date 데이터 타입은 세기, 년, 월, 일, 시간, 분, 초 등을 표기하는 데 사용한다.

2) timestamp 데이터 타입

timestamp 데이터 타입은 date 데이터 타입의 확장된 모델로 정밀도가 높다.
timestamp 데이터 타입의 데이터 길이는 고정된 11 바이트이다.
timestamp 데이터 타입에서 첫 7 바이트는 date 데이터 타입과 같고 마지막 4 바이트는 초 이하 단위로 저장한다.
timestamp 데이터 타입은 세기, 년, 월, 일, 시간, 분, 초 등을 표기하는 데 사용한다.
timestamp 데이터 타입에서 초 이하 단위는 9 자리까지 저장할 수 있다.
timestamp 데이터 타입은 특정 시점을 나타내는 데 사용되며 타임 존과 관련된 정보를 어떤 방식으로 저장하는지에 따라 달라진다.

3) interval 데이터 타입

interval 데이터 타입은 timestamp 데이터 타입 간의 차이를 나타내는데 사용되는 데이터 타입이다.
interval 데이터 타입은 년과 월의 필드의 차이를 나타내는데 두 timestamp 데이터 타입 간의 기간을 년, 개월, 일, 시간, 분, 초로 나타낸다.

4. Binary 데이터

1) 이미지 데이터 타입

이미지 데이터 타입은 이미지 데이터를 저장하는 데 사용된다.
raw 데이터 타입과 long raw 데이터 타입 대신에 blob 데이터 타입과 bfile 데이터 타입을 사용하는 것을 권장한다.

(1) blob 데이터 타입
blob 데이터 타입은 최대 4 기가까지의 바이너리 데이터를 저장한다.
blob 데이터 타입은 데이터 타입의 내부 저장 방식은 clob 데이터 타입과 유사하다.
blob 데이터 타입에 저장된 데이터는 트랜잭션에 의해 변경할 수 있다.

(2) bfile 데이터 타입
bfile 데이터 타입은 바이너리 데이터를 데이터베이스 외부인 운영체제상의 파일 시스템에 저장한다.
bfile 데이터 타입은 최대 30 바이트의 디렉터리 별칭과 최대 256 바이트의 파일 이름 표현이 가능하다.
bfile 데이터 타입에 저장된 데이터는 변경할 수 없다.
bfile 데이터 타입에 대한 무결성, 보안, 지속성은 운영체제에서 보장되어야 한다.

(3) raw 데이터 타입
raw 데이터 타입은 최대 2000 바이트까지 저장할 수 있다.

(4) long raw 데이터 타입
long raw 데이터 타입은 최대 2 기가 바이트까지 저장할 수 있다.

2) 컬럼 데이터 타입

컬럼 데이터 타입은 Binary 데이터 타입으로 테이블의 열에 대한 고유식별자이다.
컬럼 데이터 타입은 특정 열의 물리적으로 저장하지는 않지만, 특정 행을 찾는 가장 빠른 구조를 제공한다.

(1) rowid 컬럼
rowid 컬럼은 테이블의 행에 대한 고유식별자이며 명시적으로 컬럼값으로서 저장되지 않는다.

rowid 컬럼은 오라클에서 인덱스를 생성하기 위하여 내부적으로 사용하는 슈도(Pseudo) 컬럼이다.
rowid 컬럼은 데이터베이스에 보관되지 않으며 데이터베이스 자료도 아니다.
rowid 컬럼은 사용자가 임으로 변경하거나 삭제할 수 없다.
rowid 컬럼은 테이블의 컬럼처럼 참조만 가능하며 데이터베이스에 값이 저장되지는 않는다.
rowid 컬럼은 행의 물리적 주소를 직접 부여하지는 않지만, 행 위치를 지정하는 데 사용한다.
rowid 컬럼을 사용하면 가장 빠르게 테이블의 행을 액세스할 수 있다.
rowid 컬럼을 주어진 키 값의 집합을 가진 행을 지정하기 위해 인덱스에 저장된다.

(2) rownum 컬럼
rownum 컬럼은 슈도(Pseudo) 컬럼으로 rowid 컬럼과 거의 같은 기능을 가진다.
rownum 컬럼은 MySQL 데이터베이스의 limit 키워드와 같은 기능이다.
rownum 컬럼은 SQL 문의 결과에 차례대로 숫자 1 부터 순차적으로 번호를 부여하는 컬럼이다.

	ROWNUM	ENAME	SAL
1	1	SMITH	800
2	2	ALLEN	1600
3	3	WARD	1250
4	4	JONES	2975
5	5	MARTIN	1250
6	6	BLAKE	2850
7	7	CLARK	2450
8	8	KING	5000
9	9	TURNER	1500
10	10	JAMES	950
11	11	FORD	3000
12	12	MILLER	1300

rownum 컬럼은 게시판의 페이징 처리할 때 사용하며 다음 문에서 사용한다.

- update 문 사용 시 다중 건이 나올 때 첫 번째 건만 갱신할 때 where 문에서 사용한다.
- delete 문 사용 시 다중 건이 나올 때 첫 번째 건만 삭제할 때 where 문에서 사용한다.

사용자 정의 데이터 타입(User Defined Data Type)

사용자 정의 데이터 타입은 내장 데이터 타입을 기반으로 하거나 다른 사용자 정의 데이터 타입을 기반으로 하여 사용자가 정의하는 데이터 타입이다.

1. Object 데이터 타입

object 데이터 타입은 데이터들을 조작하기 위한 함수 혹은 프로시저를 묶어서 사용자가 정의한 새로운 데이터 타입이다.
object 데이터 타입은 사용자가 정의한 복합 데이터 타입이다.
object 데이터 타입은 테이블이나 컬럼으로 사용할 수 있다.
object 데이터 타입은 내부적으로는 일반 테이블 형태로 변환되어 저장되며 내장 데이터 타입이 될 때까지 확장된다.
object 데이터 타입은 사용자가 정의할 때마다 생성되므로 정해진 데이터 타입 코드는 없다.

object 데이터 타입은 3 가지 구성 요소로 이루어진다.

- name: object type 에 대한 고유식별자
- attribute: built-in 데이터타입 또는 다른 object 데이터타입
- method: PL/SQL, C, Java 와 같은 언어로 작성된 모듈

2. Collection 데이터 타입

1) varray 데이터 타입

varray 데이터 타입은 순서를 갖는 요소의 1 차원 배열로 저장하는 데 사용된다.
varray 데이터 타입의 컬럼 자체는 스칼라 타입의 컬럼처럼 단일값처럼 간주한다.
varray 데이터 타입은 데이터조작어에 의해 조작할 수 있다.
varray 데이터 타입의 각 요소에는 인덱스가 있으며 배열 상의 순서를 나타낸다.
varray 데이터 타입의 요소들은 배열의 맨 끝부터 삭제할 수 있다.

2) nested table 데이터 타입

nested table 데이터 타입은 순서를 지정하지 않은 레코드나 행의 집합이다.
nested table 데이터 타입은 테이블을 하나의 컬럼에 저장할 때 사용된다.

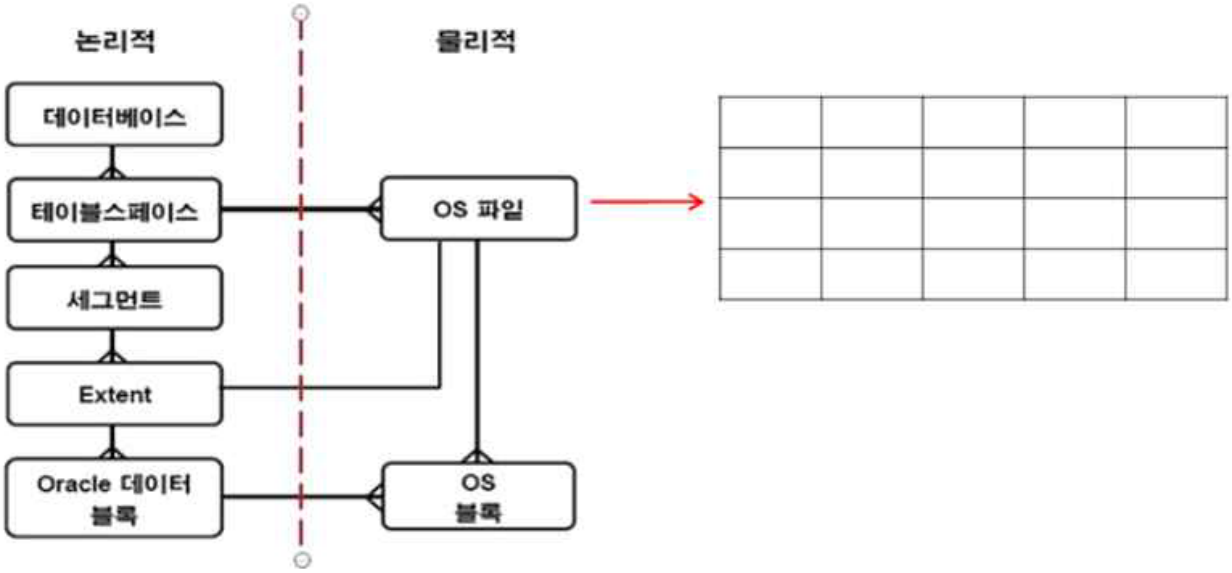
3. reference 데이터 타입

다른 객체를 참조하는 reference 데이터 타입도 사용자 정의 데이터 타입에 해당한다.
reference 데이터 타입은 테이블에 저장된 객체에 대한 참조이며 객체 그 자체는 아니다.

오라클의 테이블

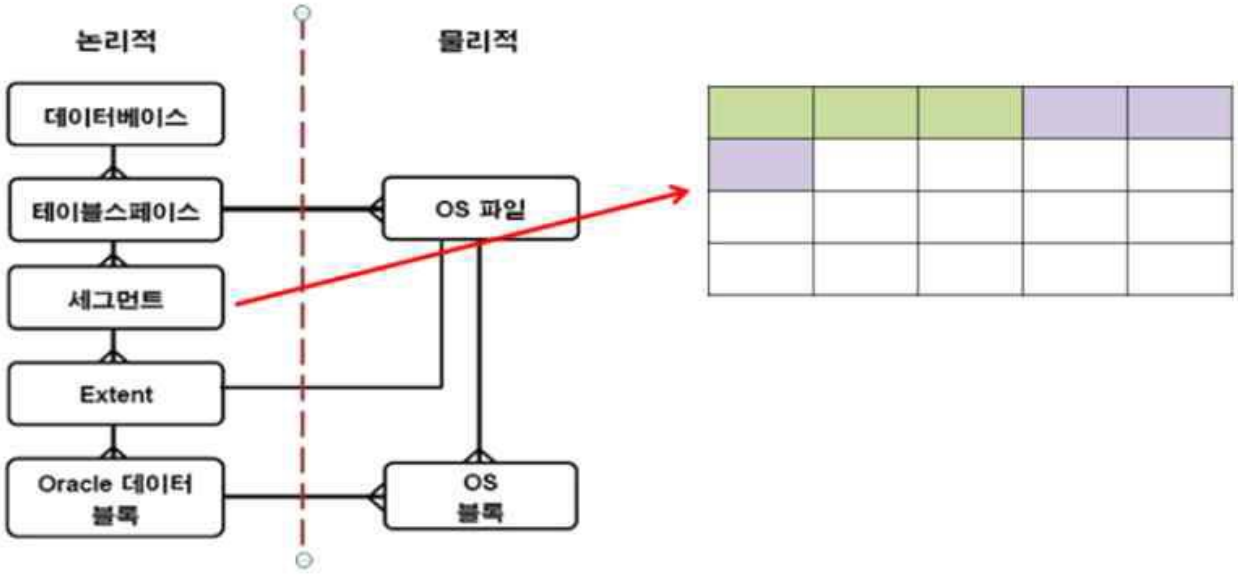
오라클은 Tablespace, Segment, Extent, Data Block 이라는 형태로 나누어 관리한다.

테이블 스페이스



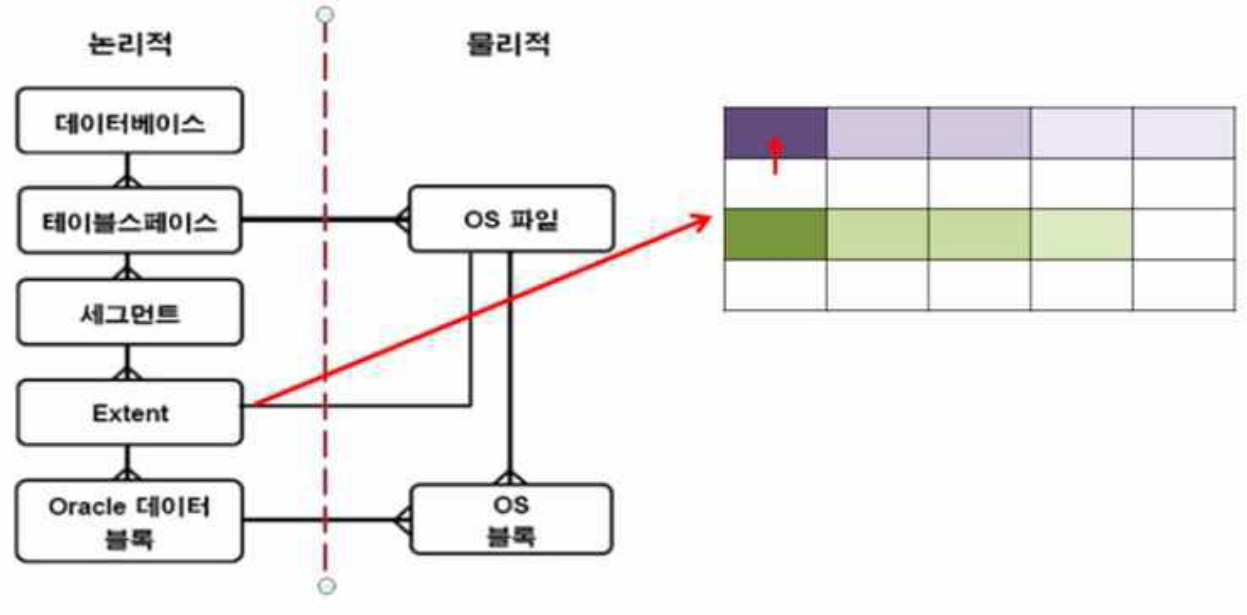
테이블 스페이스(Tablespace)는 논리적으로 서로 관련된 데이터가 저장된 파일들을 묶어놓은 단위이다.
테이블 스페이스는 데이터 파일 경로와는 관계가 없다.
테이블 스페이스는 실제 파일인 물리적인 파일과 논리적인 저장 단위를 서로 분리하는 역할을 한다.
테이블 스페이스는 데이터베이스의 논리적 저장 구조 중에 가장 포괄적이고 넓은 범위를 가진 형태다.
실제 데이터베이스 각각의 물리적 파일들은 테이블 스페이스에 N:1 형태로 연결되어 있다.
테이블 스페이스는 논리적으로 서로 관련된 데이터가 저장된 파일들을 묶어놓은 단위이다.

세그먼트



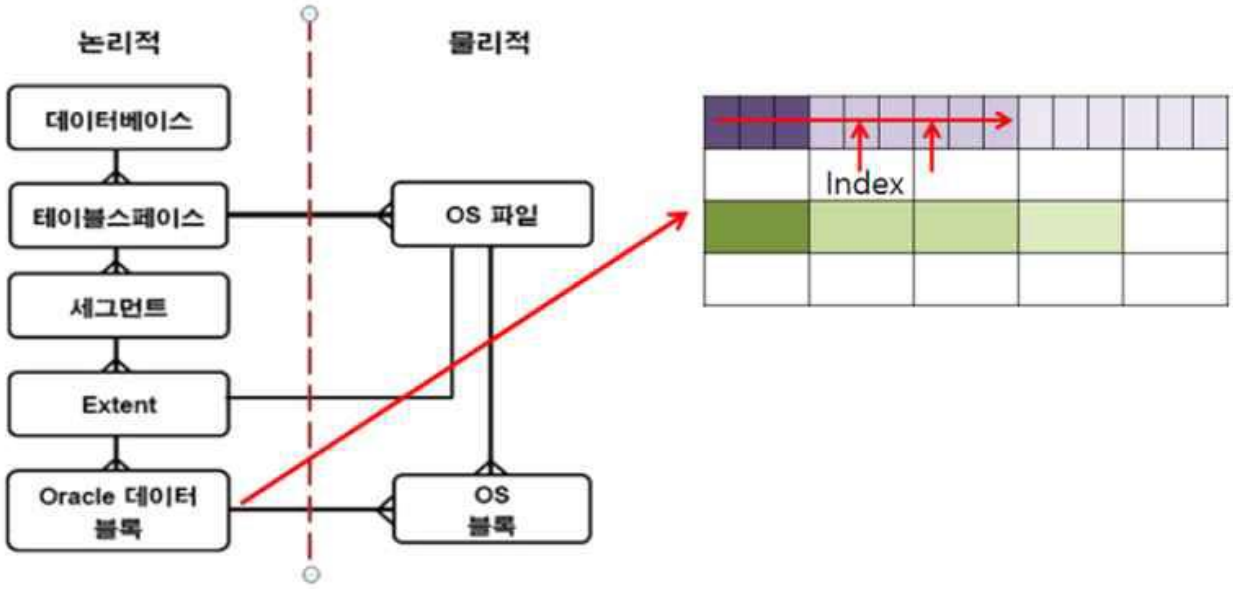
세그먼트(Segment)는 데이터베이스를 사용할 때 가장 많이 접근하게 되는 것은 테이블이다.
세그먼트는 테이블 스페이스 내에 특정 유형의 논리적 저장 구조로 할당된 영역이다.
세그먼트는 테이블, 인덱스 등이 세그먼트에 포함된다.
세그먼트는 저장 공간을 가지는 객체이며 각각의 테이블이다.

익스텐트

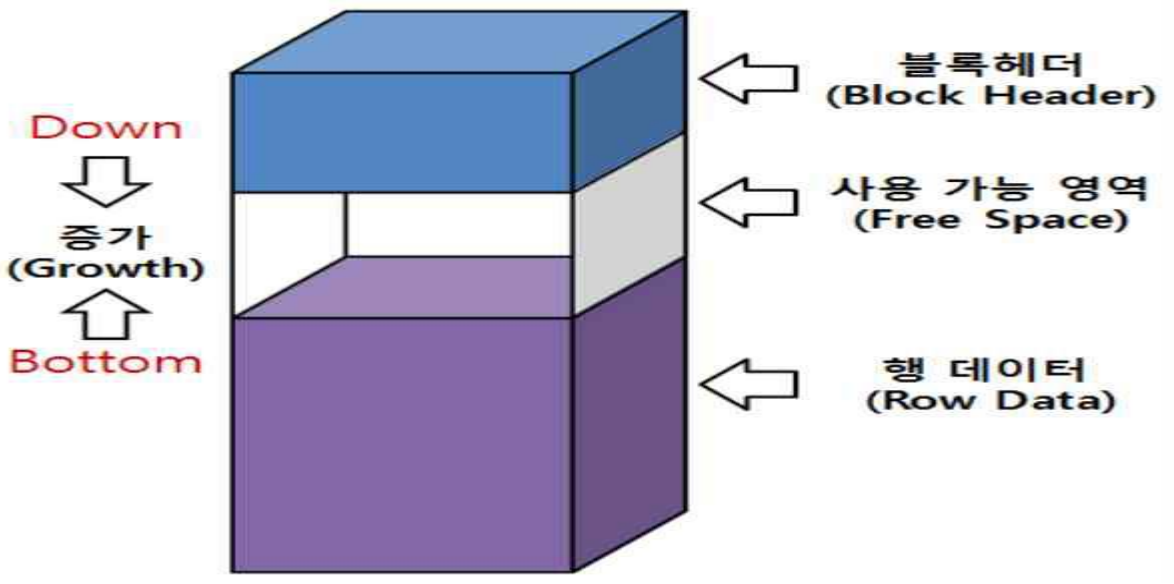


익스텐트(Extent)는 하나 이상의 연속된 데이터 블록의 모임이다.
익스텐트는 세그먼트에 대한 공간 할당 단위이기도 하다.
테이블 스페이스는 여러 개의 물리적 파일로 이루어지지만, 익스텐트는 하나의 데이터 파일에만 존재할 수 있다.

블록(block)

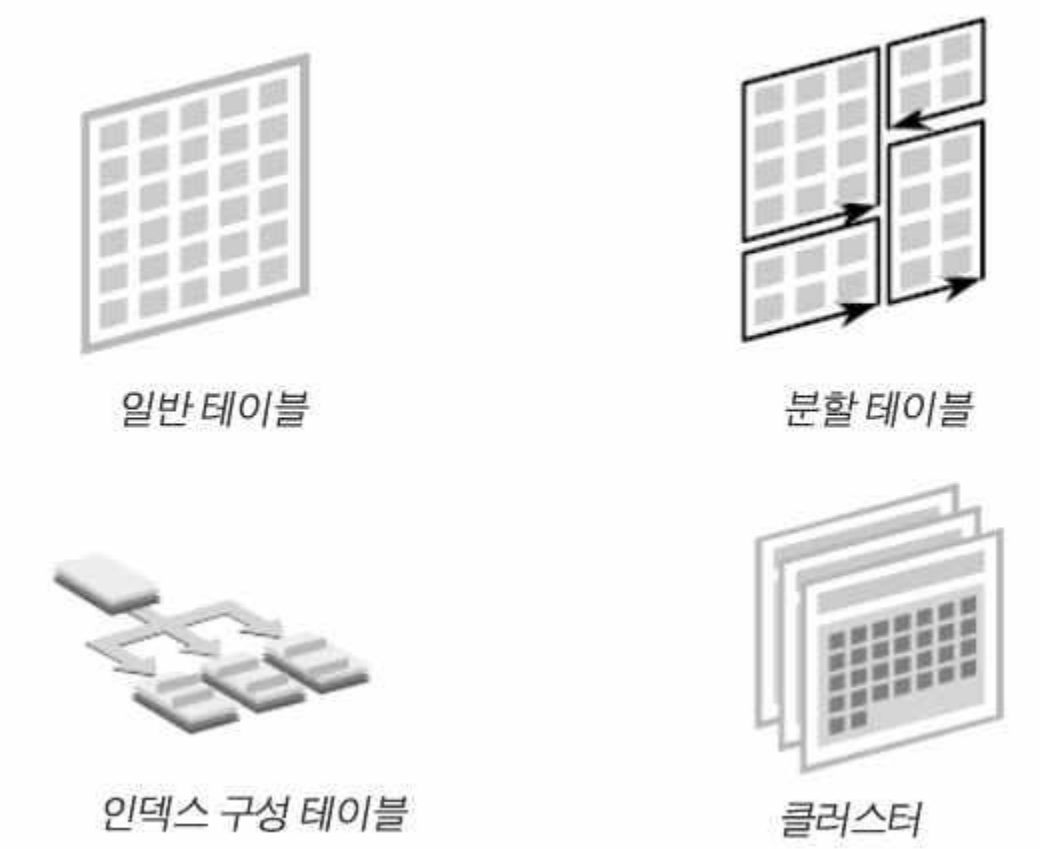


블록(block)은 오라클의 기본 입출력 단위로서 모든 데이터는 블록으로 이루어져 있다.
데이터베이스에서 데이터를 검색할 때 아주 작은 데이터 하나만을 찾으려고 할 때도 최소한 하나의 블록에 저장된 전체 데이터에 대해서는 스캔이 이루어진다.
운영체제에 따라 블록 크기의 시스템 기본설정이 다르며 최소는 2KB 이고 최대는 64KB 이다.
블록은 톱다운(Top-Down) 방식으로 헤더 정보가 증가하고 보텀업(Bottom-Up) 방식으로 데이터가 증가한다.



사용자 데이터 저장

오라클에서는 여러 가지 방법으로 사용자 데이터를 저장할 수 있다.



클러스터

클러스터는 테이블 데이터를 저장하기 위해 선택적인 방식을 제공한다.
클러스터는 같은 데이터 블록을 공유하는 테이블 또는 테이블 그룹으로 구성된다.

클러스터의 특성

- 클러스터는 함께 저장되어야 하는 행을 식별하는 데 사용되는 클러스터 키를 가진다.
- 클러스터 키는 하나 이상의 열로 구성할 수 있다.
- 클러스터의 테이블은 클러스터 키에 대응하는 열을 가진다.
- 클러스터는 테이블을 사용하는 응용 프로그램에 그대로 적용되는 방식이다.

인덱스 구성 테이블

인덱스 구성 테이블은 하나 이상의 열에 기본키 인덱스를 가진 힙 테이블과 유사하다.
인덱스 구성 테이블에서는 정확한 일치와 범위 검색을 포함한 질의에 대해 빠른 키 기반의 테이블 데이터 액세스를 제공한다.
인덱스 구성 테이블의 키 열이 테이블과 인덱스에서 중복되지 않으므로 필요한 저장 영역이 감소한다.

분할 테이블

분할 테이블에서는 크기를 조정할 수 있는 응용 프로그램을 구축할 수 있다.

분할 테이블의 특성:

- 분할 테이블에는 하나 이상의 분할 영역이 있으며 각 분할 영역은 범위 분할, 해시 분할, 조합 분할, 목록 분할을 사용하여 분할된 행을 저장한다.
- 분할 테이블의 각 분할 영역은 세그먼트이며 다른 테이블 스페이스에 있을 수 있다.
- 분할 테이블의 분할 영역은 여러 프로세스를 동시에 사용하여 질의하거나 조작할 수 있는 대형 테이블에 유용하다.
- 분할 테이블 내의 분할 영역 관리에 특수 명령을 사용할 수 있다.

일반 테이블

1. 일반 테이블의 개요

테이블이라고 하는 일반 테이블은 사용자 데이터를 저장하는 데 가장 일반적으로 사용되는 폼이며 기본 테이블이다.
데이터베이스 관리자는 일반 테이블의 행 분산에 대해 매우 제한된 제어를 수행한다.
행은 일반 테이블의 작업에 따라 임의의 순서로 저장된다.
테이블을 생성한 사용자가 테이블을 관리하며 테이블의 구조는 다음과 같다.

emp 테이블의 구조를 desc 명령어로 확인한다.
--(더블 하이픈)으로 한 줄 주석문을 사용한다.

실습
-- emp 테이블에서 desc 명령어로 구조를 확인

desc emp;

결과:

이름	널?	유형
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

2. 일반 테이블의 조회

형식
select *|columns from [as 별칭] tables;

테이블이나 명령문에서 컬럼을 select 문으로 조회한다.
컬럼 대신에 *(애스터리스크)를 사용하면 모든 컬럼을 조회할 수 있다.

as 명령어는 컬럼명의 별칭이나 가상 컬럼을 생성할 수 있으며 as 명령어는 생략할 수 있다.

형식
select *|columns from [as 별칭] tables
where 조건식;

실습
--scott 계정으로 접속하여 scott 계정의 소유 테이블을 tab 시스템 테이블로 확인한다.

```
select * from tab;
```

실습
--최고 관리자인 sys 계정으로 접속하여 오라클의 모든 계정을 all_users 시스템 테이블로 확인한다.

```
select username from all_users;
```

오라클의 함수

함수는 사용자가 원하는 데이터를 검색할 때 자주 사용한다.
함수는 구현이 복잡한 기능들을 미리 생성해 두고 손쉽게 사용할 수 있도록 하기 위한 기능이다.

제공되는 함수들은 기본적으로 SQL 문을 더욱 강력하게 해준다.
함수는 데이터값을 조작하는 데 사용된다.
데이터 타입 변환 SQL 함수는 때때로 인수를 받아들일 수 있으며 항상 값을 반환한다.

함수의 특성:

- 데이터에 계산을 수행할 수 있다.
- 개별적인 데이터 항목을 수정할 수 있다.
- 행의 그룹에 관한 결과를 조작할 수 있다.
- 출력을 위한 날짜와 숫자 형식을 조절할 수 있다.
- 열의 데이터 타입을 변환할 수 있다.

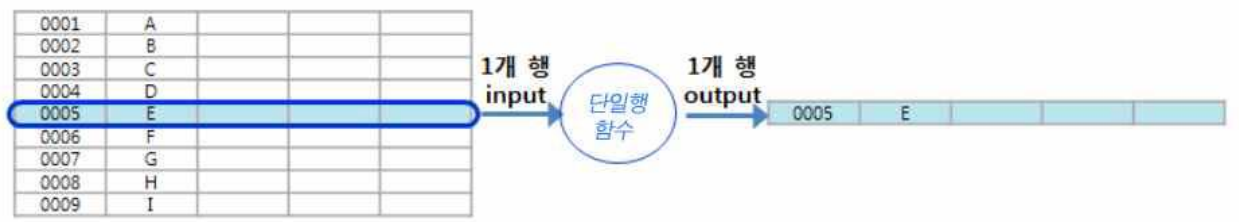
함수의 분류

함수의 유형은 크게 단일행 함수(single-row function)와 다중행(multi-row function) 함수로 이루어져 있다.

단일행 함수

1. 단일행 함수의 개요

단일행 함수는 단일 행에 대해서만 실행되며 행 당 하나의 결과를 반환한다.
단일행 함수는 데이터 항목을 조작하고 인수를 받아들이고 하나의 값을 반환한다.
단일행 함수는 반환되는 각 행에서 실행되고 행 당 하나의 결과를 반환한다.



단일행 함수는 데이터 유형을 수정할 수 있으며 중첩될 수 있다.
단일행 함수는 열이나 표현식을 인수로 받아들일 수 있다.
단일행 함수는 참조되는 유형이 아닌 다른 유형의 데이터 값을 반환할 수 있다.
단일행 함수는 하나 이상의 인수를 받아들일 수 있다.
단일행 함수는 select 문, where 문 order by 문에서 사용할 수 있고 중첩될 수 있다.

단일행 함수는 다음과 같은 경우에 사용된다.

- 데이터에 대해 계산을 수행할 경우
- 각각의 데이터 항목을 변경할 경우
- 출력할 날짜 형식을 변경할 경우
- 컬럼 데이터 타입을 변경할 경우

2. 단일행 함수의 종류

단일행 함수에는 문자, 숫자, 날짜, 변환, 일반 등의 종류가 있다.

1) 문자 함수

문자 함수는 입력으로 문자 데이터를 받아들이고 문자 변환하거나 문자를 조작하여 문자를 반환할 수 있다.

(1) lower 함수

형식 lower(column)

lower 함수는 인자인 column 의 인자값이 대소문자로 혼합되어 있거나 대문자로만 되어있는 문자열을 소문자 문자열로 변환한다.

실습
emp 테이블의 ename 컬럼값인 영문자를 소문자로 변환하여 조회한다.

```
-- emp 테이블의 empno 컬럼, ename 컬럼, hiredate 컬럼을 조회한다.

select empno, ename, hiredate from emp
where lower(ename) ='ford';
```

(2) upper 함수

형식 upper(column)

upper 함수는 인자인 column 의 인자값이 대소문자로 혼합되어 있거나 소문자로만 되어있는 문자열을 대문자 문자열로 변환한다.

실습
emp 테이블의 ename 컬럼값인 영문자를 대문자로 변환하여 조회한다.

```
select empno, ename, hiredate from emp
where ename =upper('ford');
```

(3) initcap 함수

형식 initcap(column)

initcap 함수는 인자인 column 의 인자값인 영문자의 첫 번째 문자를 대문자로 변환하고 나머지 문자는 소문자로 변환한다.

실습
dept 테이블의 dname 컬럼값인 영문자와 loc 컬럼값인 영문자에서 각 단어의 첫 번째 문자를 대문자로 나머지 문자는 소문자로 변환하여 조회한다.

```
select deptno, initcap(dname), initcap(loc) from dept
where deptno = 10;
```

(4) concat 함수

형식 concat(column1, column2)

concat 함수는 인자인 column1 컬럼값을 column2 컬럼값과 연결한다.
concat 함수는 ||(더블 버티컬바) 연결 연산자와 같은 기능을 가지지만 두 개의 인자만 사용할 수 있다.

실습
dept 테이블에서 dname 컬럼값과 loc 컬럼값을 연결한다.

```
select deptno, dname, concat(deptno, dname) from dept
where deptno = 10;
```

(5) substr 함수

형식 substr(column, m[,n])

substr 함수는 인자인 column 에서 인자인 m 위치에서 시작하는 문자열의 n 개 문자열 길이만큼 지정된 문자들을 반환하고 지정된 길이의 문자열을 추출한다.
substr 함수의 인자인 m 이 음수이면 문자 끝에서부터 카운트를 시작한다.
substr 함수의 인자인 n 이 생략된 경우 문자의 끝까지 모든 문자가 반환한다.

실습
emp 테이블의 ename 컬럼값인 영문자의 1 번째 문자에서 시작하여 2 번째 문자 까지만 조회한다.

```
select substr(ename, 1, 2) from emp;
```

(6) length 함수

형식 length(column)

length 함수는 인자인 column 의 문자열 개수를 반환하고 문자열 길이를 숫자로 표현한다.

실습
dept 테이블의 dname 컬럼값인 문자의 길이를 구한다.

```
select deptno, dname, length(dname) from dept
where deptno = 10;
```

(7) instr 함수

형식 instr(column, string, [m], [n])

instr 함수는 인자인 column 지정된 문자열을 찾아내서 0(없다) 또는 1(있다)로 값을 반환하고 위치를 숫자로 표기한다.
조회 시작 위치인 instr 함수의 인자인 m 과 문자열의 발생수인 인자인 n 을 제공할 수 있다.
instr 함수의 인자인 m 과 n 의 기본값은 1 이며 이 경우 문자열의 처음부터 조회를 시작하고 첫 번째로 찾은 결과를 반환한다.

실습 1
dept 테이블의 dname 컬럼값에서 명명된 문자의 위치를 구한다.

```
select deptno, dname, instr(dname, 'G') from dept
where deptno = 10;
```

실습 2

dept 테이블에서 loc 컬럼에서 특정 문자를 가진 행만 조회한다.

```
select deptno, dname, loc from dept
where instr(loc, 'DALLAS') > 0;
```

실습 3

dept 테이블에서 loc 컬럼에서 특정 문자열을 가진 행을 제외하고 조회한다.

```
select deptno, dname, loc from dept
where instr(loc, 'DALLAS') < 1;
```

(8) lpad 함수

형식 lpad(column, n, string)

lpad 함수는 인자인 column 의 길이가 인자인 n 이 되도록 왼쪽부터 문자로 채운 표현식을 반환하며 전체 길이 중 남는 부분 좌측에 주어진 문자로 채운다.

실습

dept 테이블의 dname 컬럼에 * 문자를 지정한 15 만큼 왼쪽 자리 채운다.

```
select deptno, dname, lpad(dname, 15, '*') from dept
where deptno = 10;
```

(9) rpad 함수

형식 rpad(column, n, string)

rpad 함수는 인자인 column 의 길이가 인자인 n 이 되도록 오른쪽부터 문자로 채운 표현식을 반환하며 전체 길이 중 남는 부분 우측에 주어진 문자로 채운다.

실습

dept 테이블의 dname 컬럼에 * 문자를 지정한 15 만큼 오른쪽 자리 채운다.

```
select deptno, dname, rpad(dname, 15, '*') from dept
```

where deptno = 10;

(10) trim 함수

형식 trim(leading|trailing|both, trim_character from column)

trim 함수의 인자인 leading 은 문자열에서 오른쪽 문자를 제거한다.
trim 함수의 인자인 trailing 은 문자열에서 왼쪽 문자를 제거한다.
trim 함수의 인자인 both 는 문자열에서 양 끝에 있는 문자를 제거한다.
trim 함수의 인자인 trim_characte 가 문자열 리터럴이면 단일 따옴표로 묶어야 한다.

실습 1
dept 테이블의 dname 컬럼값인 문자의 양 끝에 있는 공백 문자를 제거한다.

```
select deptno, dname, trim(both ' ' from dname) from dept
where deptno = 10;
```

실습 2
dept 테이블의 dname 컬럼값인 문자의 오른쪽에 있는 공백 문자를 지운다.

```
select deptno, dname, trim(leading ' ' from dname) from dept
where deptno = 10;
```

(11) translate 함수

형식 translate(column, string1, string2)

translate 함수는 인자인 column 에서 인자인 string1 문자열을 string2 문자열로 대체한다.
translate 함수는 지정 문자열과 대체 문자열을 각각 하나의 문자로 대체한다.

실습
dept 테이블의 dname 컬럼값인 문자열에서 NG 문자를 SO 문자로 대체한다.

```
select deptno, dname, translate(dname, 'NG', 'SO' ) from dept
where deptno = 10;
```

(12) replace 함수

형식 replace(text, search_string, replacement_string)

replace 함수는 인자인 text 에서 인자인 search_string 문자열을 replacement_string 문자열로 대체한다.
replace 함수는 지정된 정확한 문자열을 찾아서 대체 문자열로 변경한다.
정확한 문자열을 변경하려고 할 때는 replace 함수를 사용하고 문자 각각을 변경하려고 하면 translate 함수를 사용한다.

실습
dept 테이블의 dname 컬럼값에 공백 문자열을 빈 문자열로 대체한다.

```
select deptno, dname, replace(dname, ' ', '') from dept
where deptno = 10;
```

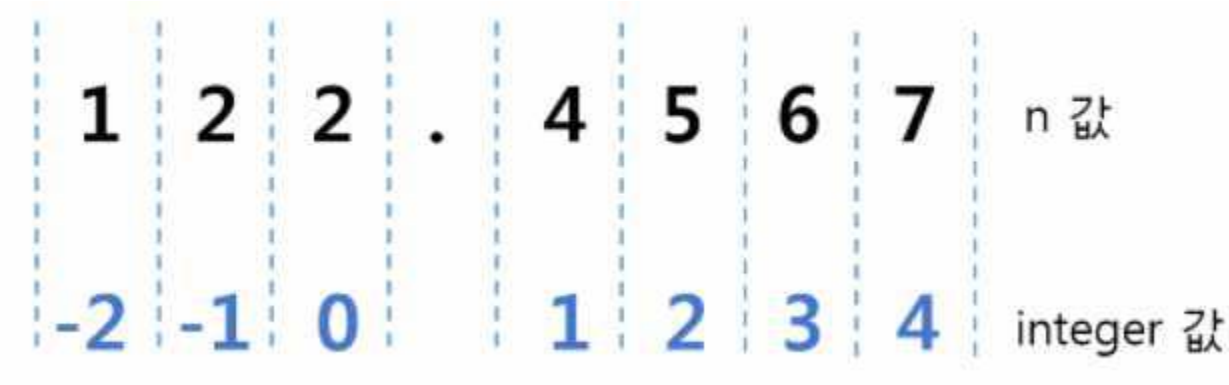
2) 숫자 함수

숫자 함수는 숫자를 입력을 받아들이고 숫자 값을 반환한다.

(1) round 함수

형식 round(column, n)

round 함수는 절반에 걸리면 무조건 반올림하는 사사오입 방식을 선택하고 있다.
round 함수는 인자인 column 값에 소수점 n 자릿수까지 반올림한다.
round 함수의 인자인 n 자릿수는 integer 위치까지 반올림하며 integer 값에 반올림하고자 하는 자릿수를 적어준다.



round 함수의 인자인 n 이 양수이면 소수 자리를 반올림하고 음수이면 정수 자리를 반올림한다.
round 함수의 인자인 n 은 생략할 수 있고 n 이 생략된 경우 소수점의 자릿수가 없다.
round 함수의 인자인 n 이 음수이면 소수점 왼쪽의 숫자가 반올림되고 기본값은 0 이다.
예를 들어 round(123.4567, 3)이라고 했을 때 123.4567 에서 소수점 넷째 자리인 7 에서 반올림해서 소수점 셋째 자리까지 표시해 주므로 123.457 이 된다.

실습
dual 테이블에서 45.926 을 소수점 두 자리에서 반올림한다.

```
select round(45.926, 2) from dual;
```

(2) trunc 함수

형식 trunc(column, n)

trunc 함수는 인자인 column 값에 소수점 n 자릿수까지 절삭한다.
trunc 함수의 인자인 n 이 양수이면 소수 자리를 절삭하고 음수이면 정수 자리를 절삭한다.
trunc 함수의 인자인 n 은 생략할 수 있으며 기본값은 0 이다.

실습
dual 테이블에서 45.926 을 소수점 두 자리에서 절삭한다.

```
select trunc(45.926, 2) from dual;
```

(3) mod 함수

형식 mod(m, n)

mod 함수의 인자인 m 을 n 으로 나눈 나머지를 반환한다.
mod 함수는 좌측에 있는 값을 우측에 있는 값으로 나눈 나머지를 반환하는 것이다.

실습
dual 테이블에서 1600 을 300 으로 나눈 나머지를 반환한다.

```
select mod(1600, 300) from dual;
```

3) 날짜 함수

오라클은 세기, 년, 월, 일, 시, 분, 초를 나타내는 내부 숫자인 7 바이트 형식으로 날짜를 저장한다.
기본 날짜 표시와 입력 형식은 DD-MM-YY 이다.
유효한 오라클 날짜는 기원전 4712 년 1 월 1 일부터 서기 9999 년 12 월 31 일까지이다.
날짜 데이터 타입의 값에 대해 실행되며 숫자를 반환하는 months_between 함수를 제외하고 모든 날짜 함수는 날짜 데이터 타입의 값을 반환한다.

오라클은 날짜를 숫자로 저장하므로 산술 연산자로 다음과 같이 산술 계산을 할 수 있다.

- Date + Number 날짜 연산: 일수를 날짜에 더하며 결과는 날짜
- Date + Number / 24 날짜 연산: 24 로 일수를 나누면 시간이 되며 시간을 날짜에 더하며 결과는 날짜
- Date – Number 날짜 연산: 날짜에서 일수를 빼며 결과는 날짜
- Date – Date 날짜 연산: 특정 날짜에서 다른 특정 날짜를 빼며 결과는 일수

(1) sysdate 함수
sysdate 함수는 시스템의 현재 날짜를 반환한다.
sysdate 함수는 현재의 오라클 서버 날짜와 시간을 반환한다.
sysdate 함수는 컬럼 이름을 사용하듯이 사용할 수 있다.

실습
dual 테이블에서 시스템의 현재 날짜를 구한다.

```
select sysdate from dual;
```

(2) months_between 함수

형식 months_between(date1, date2)

months_between 함수의 인자인 date1 과 date2 사이의 월수를 조회한다.
months_between 함수의 결과는 인자인 date1 이 date2 보다 늦은 날짜이면 결과는 양수로 표현하고 인자인 date1 이 date2 보다 앞선 날짜이면 결과는 음수로 표현한다.
months_between 함수의 결과에서 정수가 아닌 부분은 월 일부분을 나타낸다.

실습
emp 테이블에서 입사 날짜와 현재 날짜의 근무 개월 수를 조회한다.

```
select ename, hiredate, sysdate, months_between(sysdate, hiredate) from emp
where deptno = 10;
```

(3) add_months 함수

형식 add_months(date, n)

add_months 함수의 인자인 date 에 월 수 n 을 추가하고 인자인 n 값은 정수여야 하며 음수가 될 수도 있다.

실습
emp 테이블에서 입사 날짜로부터 5 개월이 지난 후의 날짜를 조회한다.

```
select ename, hiredate, add_months(hiredate, 5) from emp
where deptno = 10;
```

(4) next_day 함수

형식 next_day(date, char)

next_day 함수는 인자인 date 다음에 오는 지정된 요일인 char 의 날짜를 찾는다.
next_day 함수의 인자인 char 값은 요일을 나타내는 숫자나 문자열이 될 수 있다.

실습
emp 테이블에서 입사일로부터 돌아오는 금요일을 조회한다.

```
select ename, hiredate, next_day(hiredate, '금요일') as 월 from emp
where deptno = 10;
```

(5) last_day 함수

형식 last_day(date)

last_day 함수는 인자인 date 에 해당하는 날짜가 있는 월의 말일 날짜를 찾는다.
숫자 함수인 round 함수와 trunc 함수는 날짜 조작에도 사용할 수 있다.

실습 1
emp 테이블에서 해당 날짜에 대한 월의 말일 날짜를 조회한다.

```
select ename, hiredate, last_day(hiredate) as 월 from emp
where deptno = 10;
```

실습 2
emp 테이블에서 입사일과 현재 날짜의 근무 개월 수를 반올림하여 조회한다.

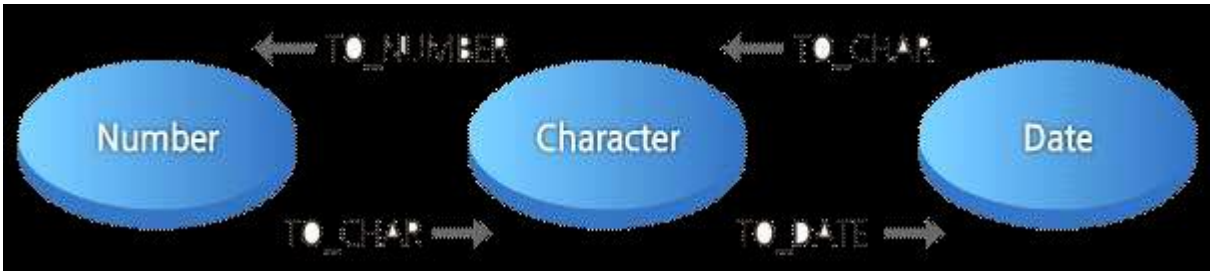
```
select ename, hiredate, sysdate, round(months_between(sysdate, hiredate)) 월수
from emp
where deptno = 10;
```

실습 3
emp 테이블에서 입사일과 현재 날짜의 근무 개월 수를 절삭하여 조회한다.

```
select ename, hiredate, sysdate, trunc(months_between(sysdate, hiredate)) 월수
from emp
where deptno = 10;
```

4) 변환 함수

데이터 타입 변환은 오라클 서버에 의해 암시적으로 수행되거나 사용자에 의해 명시적으로 수행될 수 있다.
명시적 데이터 타입 변환은 변환 함수를 사용하여 수행되며 변환 함수는 값의 데이터 타입을 변환한다.
일반적으로 함수 이름의 형식은 data type TO data type 규칙을 따르며 첫 번째 데이터 타입은 입력 데이터 타입이고 두 번째 데이터 타입은 출력 데이터 타입이다.
암시적 데이터 타입 변환은 오라클 서버는 자동으로 표현식에서 데이터 타입 변환을 수행한다.



변환 함수에는 to_char 함수, to_number 함수, to_date 함수가 있다.
to_char 함수는 to_number 함수의 기능과 to_date 함수의 기능을 포함하고 있으므로 일반적으로 to_char 함수의 사용을 권장한다.
일반적으로 오라클 서버에서는 데이터 타입 변환이 필요한 경우 표현식에 대한 규칙을 사용한다.

날짜 형식의 요소는 출력 포맷 서식으로 지정한다.

- YYYY : 년도를 4 자리로 표현하며 숫자로 된 전체 년도이다.
- YEAR : 영어 철자로 표기된 년도로 표현한다.
- MM : 월을 숫자로 표현하며 월의 2 자리 값이다.
- MONTH : 월의 전체 이름으로 표현한다.
- MON : 월의 약자로 표현한다.
- DAY : 요일의 전체 이름으로 표현한다.
- DY : 요일의 약자로 표현한다.
- DD : 숫자 형식의 월과 일을 표현한다.
- AM(PM) : 오전인 AM 과 오후인 PM 의 시각을 표현한다.
- A.M(P.M) : 오전인 A.M 과 오후인 P.M 의 시각을 표현한다.
- HH(HH12) : HH 는 전일 시간과 HH12 는 시간을 반일 시간을 표현한다.
- HH24 : 전일 시간인 0~23 으로 24 시간을 표현한다.

- MI : 분인 0~59 로 표현한다.
- SS : 초인 0~59 로 표현한다.
- 0(1111, '099999') : 앞부분을 0 을 숫자로 표현한다.
- $(1111, '$99999') : 달러 기호를 앞에 표현한다.

(1) to_char 함수

형식 to_char(number|date, fmt)

to_char 함수는 포맷 서식인 fmt 를 사용하여 숫자나 날짜를 varchar2 문자열로 변환한다.
to_char 함수의 포맷 서식 fmt 는 데이터 타입의 형식을 설명하는 문자열 리터럴이다.
to_char 함수는 다음의 규칙을 준수한다.
포맷 서식은 '(싱글 쿼터)로 묶어야 하며 대소문자를 구분한다.
포맷 서식에서 문자열을 표현하기 위해서는 '(싱글 쿼터) 안에 "(더블 쿼터)를 사용한다.
포맷 서식은 임의의 유효한 날짜 형식 요소를 포함할 수 있다.
,(콤마)를 사용하여 날짜를 포맷 서식과 구분해야 한다.
자동으로 출력의 날짜와 월 이름은 공백으로 채워진다.

실습 1
dual 테이블에서 현재 날짜를 년 월 일이라는 한글 형식을 적용하여 포맷 서식으로 조회한다.

```
select sysdate, to_char(sysdate, 'YYYY "년" MM "월" DD "일" ') from dual;
```

실습 2
dual 테이블에서 포맷 서식으로 현재 시각을 조회한다.

```
select sysdate, to_char(sysdate, 'hh24:mi:SS') from dual;
```

실습 3
emp 테이블에서 포맷 서식으로 입사 년도만 출력하여 조회한다.

```
select empno, ename, to_char(hiredate, 'yyyy') as 년도 from emp;
```

실습 4
emp 테이블에서 급여를 6 자리고 바꾸고 앞에 $ 숫자 포맷 서식으로 변환하여 조회한다.

```
select empno, ename, to_char(sal, '$999,999') as 급여 from emp;
```

(2) to_number 함수

형식 to_number(number, fmt)

to_number 함수는 숫자를 포함한 문자열을 포맷 서식인 fmt 를 사용하여 지정된 형식의 숫자로 변환한다.

실습
dual 테이블에서 문자열을 숫자 포맷 서식으로 변환하여 조회한다.

```
select to_number('1234', '9999') from dual;
```

5) 일반 함수

일반 함수는 임의의 데이터 타입을 사용한다.
일반 함수는 표현식 리스트에서 null 값의 사용과 관련이 있다.

(1) nvl 함수

형식 nvl(expr1, expr2)

nvl 함수는 null 값을 실제 값으로 변환한다.
nvl 함수의 인자인 expr1 은 null 을 포함할 수 있는 표현식이나 값이다.
nvl 함수의 인자인 expr2 는 null 을 변환하기 위한 대상값이다.

(2) nvl2 함수

형식 nvl2(expr1, expr2, expr3)

nvl2 함수는 인자인 expr1 을 검사하고 expr1 이 null 이 아니면 인자인 expr2 를 반환하고 인자인 expr1 이 null 이면 expr3 를 반환한다.
nvl2 함수의 인자인 expr1 은 null 을 포함할 수 있는 표현식이나 값으로 임의의 데이터 타입을 가질 수 있다.
nvl2 함수의 인자인 expr2 는 인자인 expr1 이 null 이 아니면 반환되는 값이다.
nvl2 함수의 인자인 expr3 는 인자인 expr1 이 null 이면 반환되는 값이다.

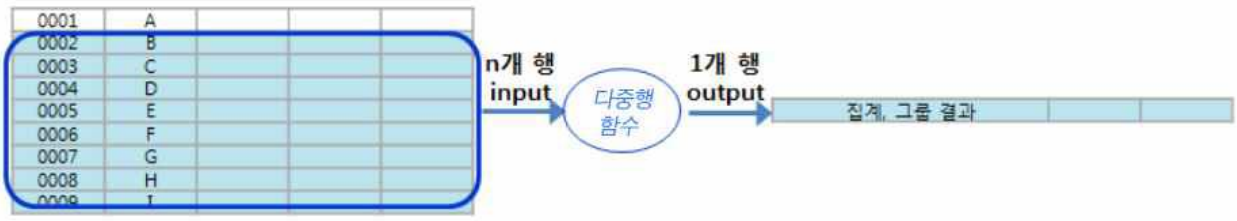
실습
emp 테이블에서 사원 테이블의 커미션을 추가한 연봉을 조회하고 null 이면 0 으로 조회한다.

```
select ename, sal, sal*12, comm, sal*12+nvl(comm, 0) from emp;
```

다중행 함수

1. 다중행 함수의 개요

다중행 함수는 데이터의 다중 행을 대상으로 조회하여 하나의 결과값을 반환한다.



단일행 함수와 달리 다중행 함수는 행 집합에 대해 실행되어 그룹 당 하나의 결과를 산출하고
행 집합은 전체 테이블이나 그룹으로 분할된 테이블로 구성될 수 있다.
다중행 함수는 select 문 뒤에 배치한다.
여러 다중행 함수는 select 문 뒤에 ,(콤마)로 구분하여 함께 사용할 수 있다.
다중행 함수의 집계 관련 함수는 where 문에서는 사용할 수 없다.
distinct 명령어를 사용하면 중복되지 않는 값만 사용한다.
다중행 함수의 집계 관련 함수는 group by 문을 사용하지 않으면 일반 컬럼과 사용할 순 없다.

2. 다중행 함수의 집계 관련 함수

(1) count 함수

형식 count(expr|*)

count 함수는 인자인 expr 의 행 개수이며 null 값은 무시한다.
count 함수의 인자인 expr 는 null 이 아닌 값을 평가한다.
count 함수는 테이블에서 컬럼의 만족하는 행 개수를 구한다.
count 함수에서 특정 컬럼을 기술하면 해당 컬럼값을 가진 행의 개수를 얻는다.
count 함수에 *(아스터리스크)를 기술하면 중복된 행과 null 값으로 된 행을 비롯하여 선택된 모든 행의 수를 구한다.

실습
emp 테이블에서 커미션을 받는 사원 수와 전체 행 개수를 구한다.

```
select count(*), count(comm) from emp;
```

(2) max 함수

형식 max(expr)

max 함수는 인자인 expr 의 최대값을 구하고 null 값은 무시한다.

실습 1
emp 테이블에서 급여의 최대값을 구한다.

```
select max(sal) from emp;
```

실습 2
emp 테이블에서 최근 입사일을 구한다.

```
select max(hiredate) from emp;
```

(3) min 함수

형식 min(expr)

min 함수는 인자인 expr 의 최소값을 구하고 null 값은 무시한다.

실습 1
emp 테이블에서 급여의 최소값을 구한다.

```
select min(sal) from emp;
```

실습 2
emp 테이블에서 최초 입사일을 구한다.

```
select min(hiredate) from emp;
```

(4) sum 함수

형식 sum(n)

sum 함수는 인자인 n 의 합을 구하고 null 값은 무시한다.
합은 관측값을 모두 합한 값이다.

실습
emp 테이블에서 급여의 합을 구한다.

```
select sum(sal) from emp;
```

(5) avg 함수

형식 avg(n)

avg 함수는 인자인 n 의 평균을 구하고 null 값은 무시한다.
평균은 관측값을 모두 합한 후에 관측도수로 나눈 값이다.

실습
emp 테이블에서 급여의 평균을 구한다.

select avg(sal) from emp;

(6) variance 함수

형식 variance(expr)

variance 함수는 인자인 expr 의 분산을 구하고 null 값은 무시한다.

실습
emp 테이블에서 급여의 분산을 구한다.

select variance(sal) from emp;

(7) stddev 함수

형식 stddev(expr)

stddev 함수는 인자인 expr 의 표준 편차를 구하고 null 값은 무시한다.
표준 편차는 각각의 자료들이 평균적으로 떨어져 있는 정도를 의미한다.
표준 편차는 분산의 제곱근으로 분산보다 많이 쓰인다.

실습
emp 테이블에서 급여의 실질적인 분포를 구한다.

select stddev(sal) from emp;