엔지니어링 19기 BASE session 5주차

DB 기초 이론

2022.08.17

엔지니어링 18기 이현진

Contents

1. DataBase

데이터 vs 정보

- 데이터 : 현실 세계에서 단순히 관찰하거나 측정하여 수집한 값
- 정보 : 데이터를 의사 결정에 유용하게 할용할 수 있도록 처리하여 체계적으로 저장한 결과물

데이터의 종류

- 정형 데이터 : 구조화된 데이터
- 반정형 데이터 : 구조에 따라 저장되었지만 데이터 내용 안에 구조에 대한 설명도 함께 존재

  ex) HTML, JSON
- 비정형 데이터 : 소셜데이터나 멀티미디어 데이터

1. DataBase

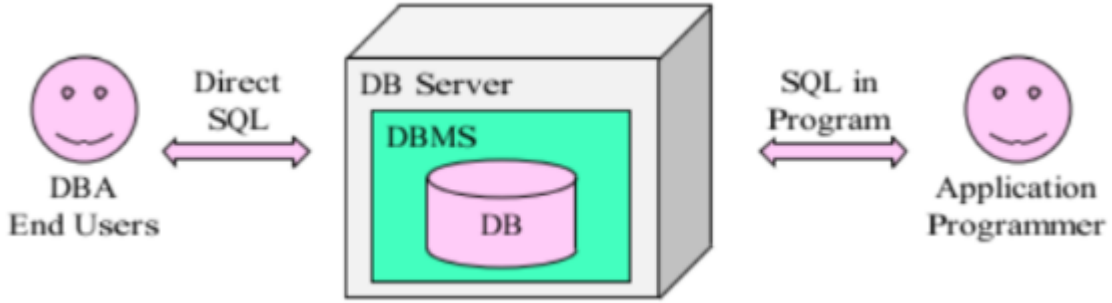
데이터베이스의 정의

특정 조직의 여러 사용자가 **공유**하여 사용할 수 있도록 여러 응용 시스템들의 정보를 **통합**해서 **저장**한 **운영데이터**의 집합

- 공유 데이터 : 한 조직의 여러 응용 프로그램이 공동으로 소유, 유지 사용하는 데이터
- 통합 데이터 : 최소의 중복과 통제 가능한 중복만 허용
- 저장 데이터 : 컴퓨터가 접근 가능하도록 매체에 저장된 데이터
- 운영 데이터 : 한 조직의 고유 기능을 수행하기 위해 필요한 데이터

논리적으로 연관된 하나 이상의 자료 모음이고, 데이터를 고도로 구조화함으로써 검색/갱신 등의 데이터 관리를 효율화

데이터베이스의 특징



- 실시간 접근성(Real-Time Accessibility)
- 계속적인 변화(Continuous Evolution)
- 동시 사용(Concurrent Sharing)
- 내용에 의한 참조(Content Reference)

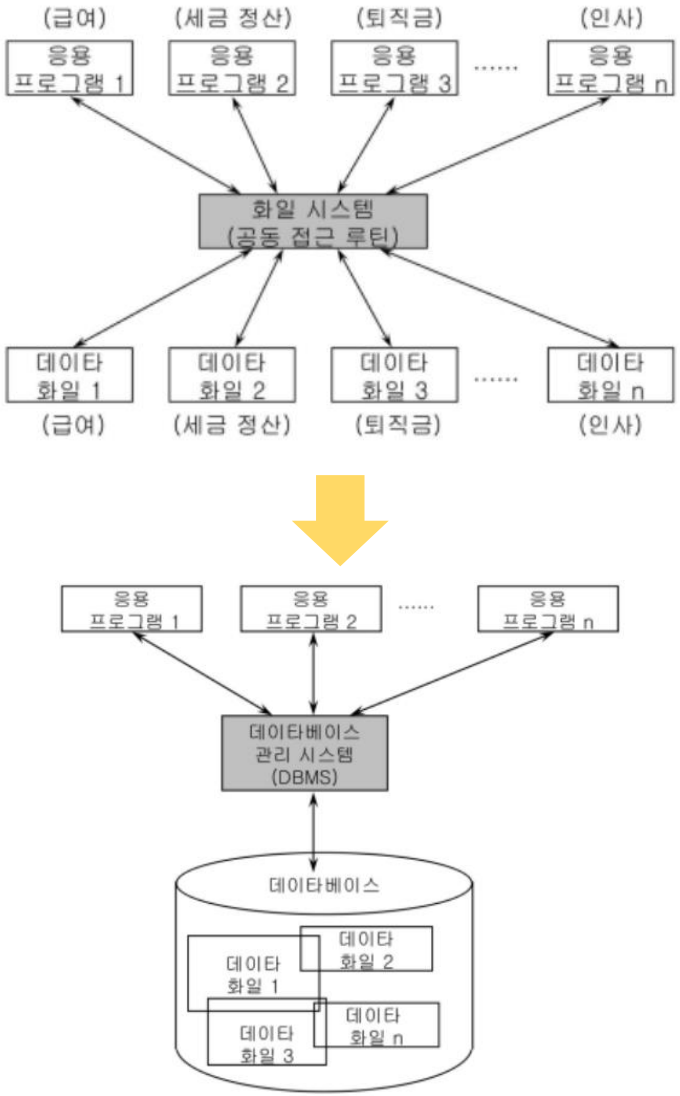
2. DBMS

파일 시스템

데이터의 입출력을 전담하는 별도의 시스템

파일 시스템의 문제점

- 데이터 종속성 : 응용 프로그램과 데이터 사이에 의존적
- 데이터 중복성 : 한 시스템 내에 내용이 같은 데이터가 중복되어 저장/관리
  - 데이터의 무결성과 일관성 유지에 어려움
  - 저장공간 낭비
- 동시 공유, 보안, 회복 기능 부족
- 응용 프로그램 개발 어려움



DBMS

데이터의 종속성과 중복성 문제 해결

관련 데이터를 통합하여 데이터를 공유할 수 있도록 관리하는 소프트웨어

응용프로그램과 데이터의 중재자

DB구성, 접근 방법, 유지 관리에 대한 모든 책임 담당

DB에 대한 접근 요청인 SQL request 처리

2. DBMS

DBMS 역할

1. Data Definition: DB의 구조를 정의하는 기능
2. Data Manipulation:데이터의 검색, 삽입, 삭제, 갱신 기능

→ CRUD(Create, Retrieve, Update, Delete)

3. Optimization & Execution:연산의 최적화 및 실행
4. Performance:최상의 성능을 보장
5. Data Protection: 데이터 보안
6. System Catalog(Data Dictionary or Metadata) : 데이터 분류 및 카테고리화

데이터 독립성

하위 스키마를 변경해도 상위 스키마가 영향을 받지 않는 특성

- 논리적 데이터 독립성

  개념 스키마가 변경되어도 외부 스키마가 영향 받지 않음

  관련 외부/개념 내용만 수정

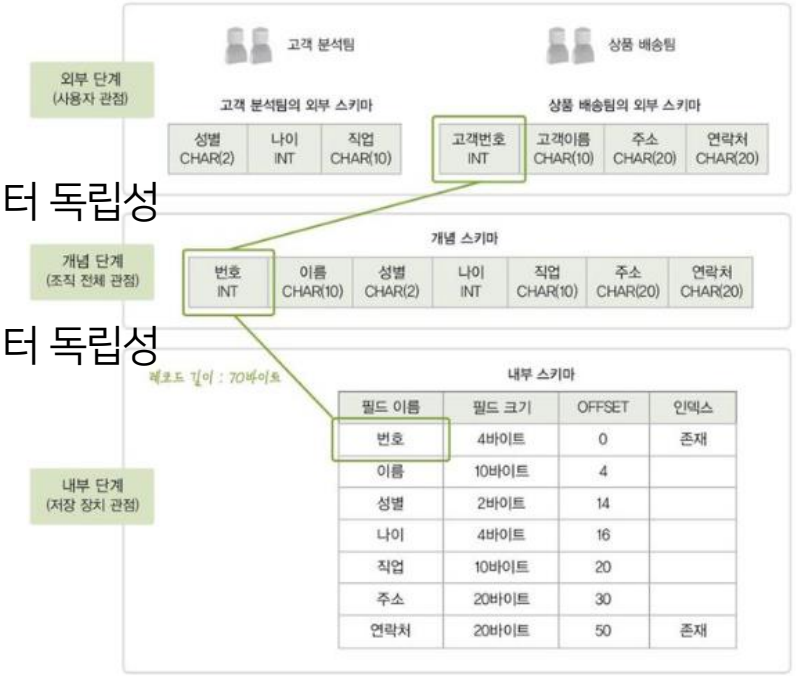
- 물리적 데이터 독립성

  내부 스키마가 변경되어도 개념스키마가 영향 받지 않음

  관련 개념/내부 내용만 수정

논리적 데이터 독립성

물리적 데이터 독립성



2. DBMS

DBMS 발전 과정

구분	모델	설명	DBMS
0세대	파일시스템	컴퓨터에서 파일이나 자료를 쉽게 발견 및 접근할 수 있도록 보관 또는 조직하는 체제	- ISAM - VSAM
1세대	계층형(Hierachical) HDBMS	계층적인 형태의 DBMS. 초기 세팅이 변화하면 이에 대처하기가 힘들어 잘 쓰이지 않음. 초기 DBMS의 형태.	- IMS - System2000
1세대	네트워크형(Network) NDBMS	구성과 설계가 복잡하고 데이터 종속성을 해결하지 못함.	- IDS - TOTAL - IDMS
2세대	관계형(Relatioinal) RDBMS	테이블과 테이블의 관계를 기반으로 하는 가장 범용적인 데이터베이스 관리 시스템. 데이터의 일관성(Consistency)를 보장함.	- Oracle - My-SQL - DB2 - SQL Server - Sybase
3세대	객체지향(Object Oriented) ODBMS	정보를 객체의 형태로 표현하는 DBMS.	- Object Store - UniSQL
4세대	NOSQL	데이터 간의 관계를 설정하지 않고 유연한 테이블 스키마를 가짐. 대용량 데이터/분산 처리에 적합하다는 장점이 있지만 데이터 일관성(Consistency)이 항상 보장되지 않음.	- MongoDM - HBase - Cassandra - Redis

3. ERD

존재하고 있는 것(Entity)들의 관계(Relationship)을 나타낸 도표(Diagram). E-R 다이어그램 혹은 ERD 라고 부름

사용되는 곳

- 관계형DB

  엔티티와 속성들을 테이블과 칼럼들로 변환 가능

  테이블들과 관계들 시각화 -> 설계 문제점 파악 용이

- 소프트웨어 계획 단계

  서로 다른 시스템 요소와 서로 간의 관계 식별에 도움

  Data flow diagram의 기초로 사용됨

특징

- 추상화

  데이터에 대한 개념적 표현을 제공

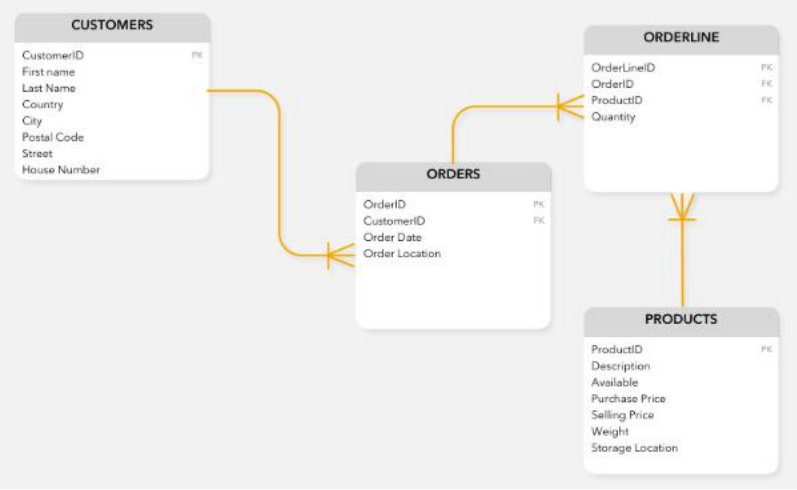
- 단순화

  현실세계를 약속된 규약이나 제한된 표기법과 언어로 표현

- 명확화

  누구나 이해하기 쉽게 애매한 점 없이 표현

예제



소비자가 상품을 주문하는 현실 세계의 일 나타내기

-> 주문이라는 테이블 기준

-> 주문을 한 소비자의 테이블

-> 상품의 테이블

-> 주문 관련 내용 테이블

프로젝트에서 사용되는 DB의 구조를 한눈에 파악할 수 있다.

3. ERD

도식화 기호 살펴보기

Entity - 개체

사물 또는 사건으로 정의

사각형 표기 -> 개체명 단수형으로 명명

가능한 대문자 개체명 사용

유일한 단어로 사용

Weak Entity

Entity 중 존재하는 다른 Entity에 의존적인 Entity

Entity 분류

구분	내용
유형 엔티티	물리적인 형태 존재 Ex_) 고객, 상품, 거래처, 학생, 교수 등
무형 엔티티	물리적인 형태가 없고 개념적으로만 존재 Ex_) 인터넷 장바구니, 부서 조직 등
문서 엔티티	업무 절차상에서 사용되는 문서나 장부, 전표에 대한 엔티티 Ex_) 거래명세서, 주문서 등
이력 엔티티	업무상 반복적으로 이루어지는 행위나 사건의 내용을 일자별, 시간별로 저장하기 위한 엔티티 Ex_) 입고 이력, 출고 이력, 구매 이력 등
코드 엔티티	무형 엔티티의 일종으로 각종 코드를 관리하기 위한 엔티티 Ex_) 국가코드, 각종 분류 코드

3. ERD

도식화 기호 살펴보기

Attribute - 속성

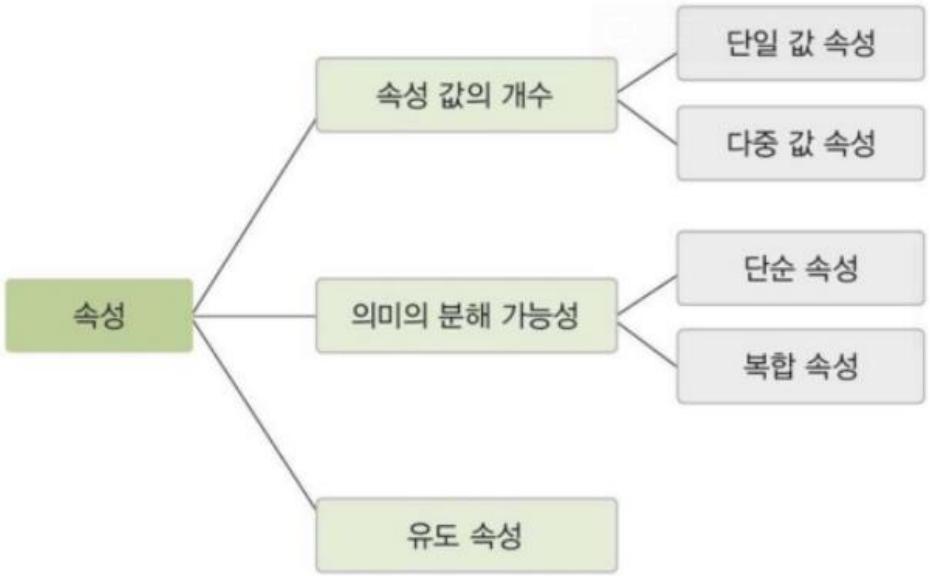
개체가 가지고 있는 요소 또는 성질

선으로 연결된 동그라미로 표기

속성명은 단수형이고 개체명과 동일한 명칭 사용X

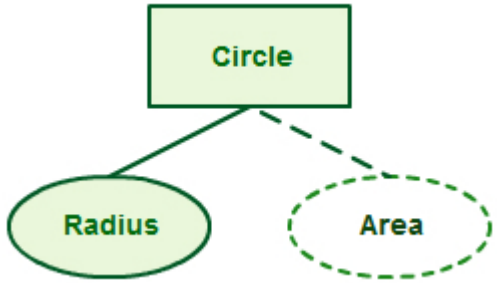
Null값 고려

Attribute 종류



Multivalued Attribute : 한 개 이상의 값 가짐

Derived Attribute : 다른 속성에 기초한 속성



유도 속성

기본 속성 값에서 유도되어 결정

3. ERD

도식화 기호 살펴보기

Relationthip - 관계

대부분 마름모형으로 표기하지만 표기법에 따라 다양하게 표현 가능

모든 엔티티에 대해 사각형을 만들고 이름을 적절하게 지정

매핑 카디널리티

관계를 맺는 두 개체 집합에서 각 개체 인스턴스가 연관성을 맺고 있는 상대 개체 집합의 인스턴스 개수

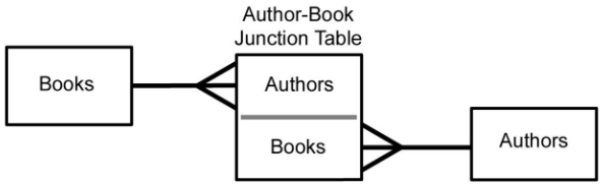
One-to-One 관계 - 학생과 키. 학생 한 명은 하나의 신체정보를 가지기 때문

One-to-Many 관계 - 저자 테이블과 댓글 테이블

Many-to-Many 관계 - 저자는 여러 글을 작성할 수 있고, 각 글도 여러 저자의 의해 공동으로 작성될 수 있음

두 테이블만 가지고 현실의 모델을 표현할 수 없기 때문에 이 둘을 매개해주는 연결 테이블을 생성한 뒤,

서로 1대 다 관계를 맺으면서 표현



필수/선택 기호 "|" 표시는 필수, "O" 표시는 선택

관계	선택성	IE 표기법	Barker 표기법
1 : 1	필수		
1 : 1	선택		
1 : N	필수		
1 : N	선택		

3. ERD

제약조건

테이블에 부적절한 자료가 입력되는 것을 방지하기 위해서 여러가지 규칙을 적용해 놓는 것. 테이블 안에서 데이터의 성격을 정의하는 것.

NOT NULL

컬럼을 필수 필드와 시킬 때 사용. NOT NULL 설정 시 해당 컬럼에는 꼭 데이터 입력해야 함

UNIQUE

데이터의 유일성을 보장(= 중복되는 데이터가 존재할 수 없음)하고, 자동으로 인덱스가 생성.
NULL값을 허용하고 하나의 테이블에 여러 개 올 수 있음

CHECK

컬럼의 값을 어떤 특정 범위로 제한

DEFAULT(컬럼 기본값) 지정

데이터 입력하지 않아도 지정된 값이 기본으로 입력되고 열이름이 명시되지 않으면 자동 기본값이고 값이 직접 명시되면 기본값은 무시됨

PRIMARY KEY

기본키 : UNIQUE + NOT NULL의 결합
데이터 행 대표 컬럼으로서의 역할 수행. 다른 테이블의 외래키들이 참조할 수 있는 키 -> 참조 무결성
자동 인덱스 생성
테이블의 각 필드값에 유니크한 값이 없으면 필드 두 개를 묶어서 primary key 지정 가능

FOREIGN KEY

기본키를 참조하는 컬럼 or 컬럼들의 집합
외래키를 가지는 컬럼의 데이터 형은 외래키가 참조하는 기본키의 컬럼과 데이터 형이 일치해야 한다(참조 무결성)
외래키에 의해 참조되고 있는 기본키는 **삭제 불가 !!**

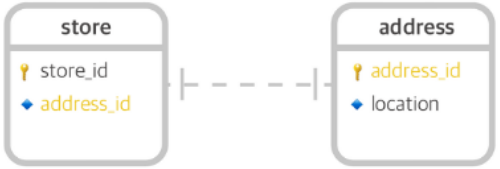
3. ERD

두 개체의 관계 고려

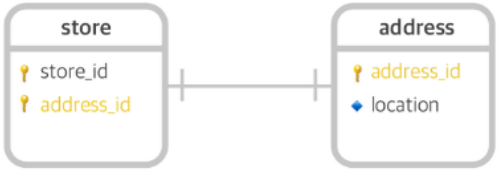
두 관계중 부모의 키를 PK로 받는지 안받는지에 따라서 점선, 실선 표기가 다르게 된다.

- 부모와 자식이 아래사진과 다음과 같을때
  - 부모 : address
  - 자식 : store
- 실선 : 식별 관계
  - 부모 자식 관계에서 자식이 부모의 키를 외래키로 참조해서 자신의 키로 설정.
- 점선 : 비식별 관계
  - 부모 자식 관계에서 자식이 부모의 키를 일반 속성으로 참조.

비식별자 관계
- 부모의 키가 일반 속성으로 포함



식별자 관계
- 부모의 키가 PK로 포함



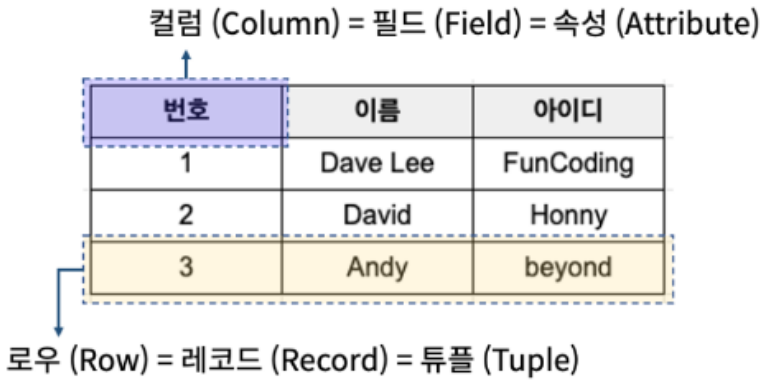
현실 세계를 개념적 데이터 모델을 통해 논리적 데이터 모델로 만드는 것

4. RDBMS

관계형(Relatioinal) 데이터베이스

RDB는 관계형 데이터 모델을 기초로 두고 모든 데이터를 2차원 테이블 형태로 표현하는 DB

RDBMS는 다른 테이블들과 관계를 맺고 모여 있는 집합체



- 가장 많이 사용되는 데이터베이스의 한 종류
- 역사가 오래되어 가장 신뢰성이 높고, 데이터 분류, 정렬, 탐색 속도 빠름
- 관계형 데이터베이스 = 테이블!
- 2차원 테이블 형식을 이용하여 데이터 정의하고 설명하는 데이터 모델
- 데이터를 속성(Attribute)과 값(Attribute Value)로 구조화
- 데이터의 구조화는 속성과 데이터 값 사이의 관계를 찾아내고 테이블 모양의 구조 도식화하는 것

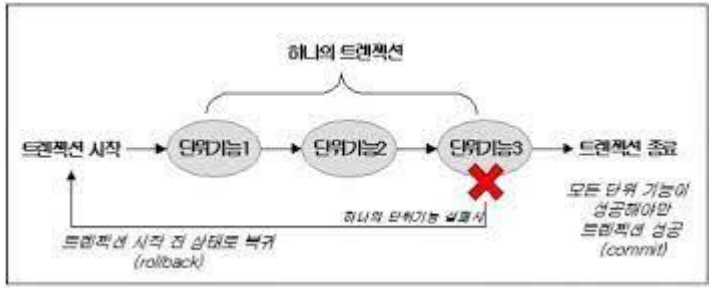
4. RDBMS

특성

트랜잭션 ACID

트랜잭션(Transaction) : 여러 개의 작업을 하나로 묶은 실행 유닛

- 각 트랜잭션은 하나의 특정 작업으로 시작을 해서 묶여 있는 모든 작업들을 다 완료해야 정상적으로 종료한다.
- 만약 하나의 트랜잭션에 속해 있는 여러 작업 중에서 단 하나의 작업이라도 실패하면, 이 트랜잭션에 속한 모든 작업을 실패한 것으로 판단한다.
- 작업이 하나라도 실패를 하게 되면 트랜잭션도 실패이고, 모든 작업이 성공적이면 트랜잭션 또한 성공이다.
- 성공 또는 실패 라는 두 개의 결과만 존재하는 트랜잭션은, 미완료된 작업없이 모든 작업을 성공해야 한다.



데이터베이스의 상태를 변환시키는 기능을 수행하기 위한 하나 이상의 쿼리를 모아 놓은 하나의 작업 단위

4. RDBMS

특성

트랜잭션 ACID

A – Atomicity(원자성)

C – Consistency(일관성)

I – Isolation(격리성, 고립성)

D – Durability(지속성

데이터베이스 내에서 일어나는 하나의

트랜잭션(transaction)의 안전성을 보장하기 위해

필요한 성질.

RDBMS를 사용하면 데이터베이스와 상호 작용하는

방식을 정확하게 규정할 수 있기 때문에,

데이터베이스에서 데이터를 처리할 때 발생할 수

있는 예외적인 상황을 줄이고,

데이터베이스의 무결성을 보호할 수 있다.

Atomicity(원자성)

시스템에서 한 트랜잭션의 연산들이 **모두 성공**하거나, 반대로 **전부 실패**되는 성질

ex) 계좌이체

1. A 계좌에서 출금한다.
2. B 계좌에 입금한다.

원자성을 지켰다면 1번과 2번, 두 작업이 모두 성공적으로 완료되어야 함.

그렇지 않으면(둘 중 하나의 작업이라도 실패한다면), 하나의 단위로 묶여

있는 모든 작업이 실패하게 만들어(롤백) 기존 데이터를 보호.

- 계좌이체를 하려는데 A 계좌에서는 출금이 이뤄지고, B 계좌에 입금되지 않았다고 가정
- 어디서 문제가 발생했는지 파악할 수 없다면, A 계좌에서 출금된 돈은 세상에서 사라지는 돈
- 만약 은행에서 이런 일이 발생한다면, 은행은 더이상 제 기능을 할 수 없는 것
- A 계좌에서 출금하는 일에 성공했지만, B 계좌에 입금하는 작업에 실패한다면 계좌 A에서 출금하는 작업을 포함하여 모든 작업이 실패로 돌아가야 한다는 것이 Atomicity(원자성)

4. RDBMS

특성

트랜잭션 ACID

A – Atomicity(원자성)

C – Consistency(일관성)

I – Isolation(격리성, 고립성)

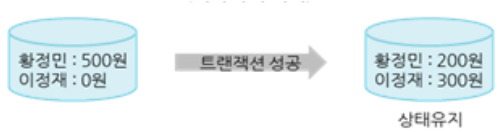
D – Durability(지속성

Consistency(일관성)

하나의 트랜잭션 이전과 이후, 데이터베이스 상태는 이전과 같이 유효해야 한다

-> 트랜잭션이 일어난 이후의 DB는 제약이나 규칙 만족해야 함.

ex) '모든 고객은 반드시 이름을 가지고 있어야 한다' 는 제약



1. 이름 없는 새로운 고객 추가
2. 기존 고객의 이름 삭제

) 제약 조건 위반

Isolation(격리성, 고립성)

모든 트랜잭션은 다른 트랜잭션으로부터 독립되어야 한다

-> 여러 트랜잭션 수행 시 각 트랜젝션은 고립(격리)되어 연속으로 실행된 것과 동일

ex) 계좌 B로 6천 원을, 계좌 C로 6천 원을 동시에 계좌 이체하는 경우,
계좌 B에 먼저 송금한 뒤 계좌 C에 보내는 결과와 동일

ex) 실행 중인 트랜잭션의 중간에 다른 트랜잭션 접근 X

A 사용자 트랜잭션	B 사용자 트랜잭션
① 황정민 계좌 : -300 ② 이정재 계좌 : +300	③ 이정재 계좌 : +100

4. RDBMS

특성

트랜잭션 ACID

A – Atomicity(원자성)

C – Consistency(일관성)

I – Isolation(격리성, 고립성)

D – Durability(지속성

Duration(지속성)

하나의 트랜잭션이 성공적으로 수행되었다면, 해당 **트랜잭션에 대한 로그가 남아야 하는 성질**

만약 런타임 오류나 시스템 오류가 발생하더라도, 해당 기록은 영구적

ex) 은행에서 계좌이체를 성공적으로 실행한 뒤에, 해당 은행 데이터베이스에 오류가 발생 가정

-> 계좌이체 내역은 기록으로 남아야 함.

장단점

장점	정해진 스키마에 따라 데이터 저장해서 명확한 데이터 구조 보장 각 데이터 중복 없이 한 번만 저장 가능
단점	테이블 간 관계 맺고 있어 시스템 커지면 JOIN문이 많은 복잡한 쿼리 생성 성능 향상 위해 Scale-up만을 지원 -> 비용 기하급수적으로 증가 스키마로 인해 데이터 유연X -> 스키마 변경될 경우 번거롭고 어렵다

4. RDBMS

Key 종류 설명

- 슈퍼 키(Super Key): 유일성을 만족하는 키.
- 복합 키(Composite Key): 2개 이상의 속성(attribute)를 사용한 키.
- 후보 키(Candidate key): 유일성과 최소성을 만족하는 키. 기본키가 될 수 있는 후보.
- 기본 키(Primary key): 후보 키에서 선택된 키. NULL값 X, 기본키로 선택된 속성(Attribute)은 동일한 값이 들어갈 수가 없음.
- 대체 키(Surrogate key): 후보 키 중에 기본 키로 선택되지 않은 키.
- 외래 키(Foreign Key): 어떤 테이블(Relation) 간의 기본 키(Primary key)를 참조하는 속성. 테이블(Relation)들 간의 관계를 나타냄.

Super Key

7조 테이블

학번	주민번호	이름	나이
1	928888-8888888	김기범씨	27
2	929999-7777777	추정범씨	27
3	007777-6666666	이은빈씨	19
4	986666-5555555	지혜리씨	21

어떤 속성끼리 묶던 중복값이 안나오고 서로 구별할 수 있으면 됨.

학번 / 주민번호
이름 + 나이
학번 + 주민번호
학번 + 주민번호 + 이름
학번 + 주민번호 + 이름 + 나이
주민번호 + 이름 + 나이

4. RDBMS

Key 종류 설명

Candidate Key

7조 테이블

학번	주민번호	이름	나이
1	928888-8888888	김기범씨	27
2	929999-7777777	추정범씨	27
3	007777-6666666	이은빈씨	19
4	986666-5555555	지혜리씨	21

슈퍼키들 중에서 속성을 최소한의 개수로 하여 구분할 수 있도록 하는 것

학번
주민번호

Primary Key

7조 테이블

학번이 기본키

학번	주민번호	이름	나이
1	928888-8888888	김기범씨	27
2	929999-7777777	추정범씨	27
3	007777-6666666	이은빈씨	19
4	986666-5555555	지혜리씨	21

후보키들 중 하나를 선택한 키. 최소성과 유일성 만족

테이블 내 기본키는 오직 1개

NULL 값 X, 중복된 값 X

학번

4. RDBMS

Key 종류 설명

Alternate Key

7조 테이블

학번	주민번호	이름	나이
1	928888-8888888	김기범씨	27
2	929999-7777777	추정범씨	27
3	007777-6666666	이은빈씨	19
4	986666-5555555	지혜리씨	21

후보키가 두 개 이상일 경우 기본키로 지정하고 남은 후보키

학번 기본키가 없어지면 주민번호가 기본키 대체

주민번호

Foreign Key

<학생 Table>

학번	주민등록번호	성명
A11	111-123	홍길동
A12	112-789	김철수
A13	119-753	박영희
A14	115-951	홍길동

<수강 Table>

학번	과목명
A11	영어
A11	수학
A13	영어
A13	수학

- 부모 테이블은 학생 테이블이고, 자식 테이블은 수강 테이블
- 학생테이블은 학번이 기본키이자 참조되는 참조키
- 수강테이블은 학번이 참조하는 키이자 외래키

4. RDBMS

RDBMS 소프트웨어 종류

Oracle	- 오라클에서 만들어 판매중인 상업용 데이터베이스 - 윈도우, 리눅스, 유닉스 등 다양한 운영체제에서 설치 가능 - MySQL, MSSQL보다 대량의 데이터 처리 용이 - 대기업 주로 사용. 글로벌 DB 시장 점유율 1위 - 가장 널리 사용되는 RDBMS
MySQL	- MySQL사에서 개발. 썬마이크로시스템즈를 거쳐 현재 오라클에 인수합병 - 윈도우, 리눅스, 유닉스 등 다양한 운영체제에서 설치 가능 - 오픈소스로 이루어져있는 무료 프로그램(상업적 사용 시 비용 발생) - 가격 증의 장점을 앞세워 다수의 중소기업에서 사용 중
MSSQL	- 마이크로소프트사에서 개발한 상업용 데이터베이스 - 다른 운영체제에서도 사용 가능하지만 윈도우 특화 - 비공개 소스로 폐쇄적인 운영(리눅스 버전은 오픈소스) - 중소기업에서 주로 사용
MariaDB	- MySQL이 오라클에 인수합병된 후 불확실한 라이선스 문제를 해결하려고 나온 오픈소스 RDBMS - C++로 구현 - MySQL과 동일한 소스 코드 기반 - MySQL과 비교해 애플리케이션 부분 속도가 약 4~5천 배 정도 빠름

4. RDBMS

RDBMS 구조의 존재 이유

ID	구분	이름	대여자	학번	저자
1	도서대출	정의란무엇인가	김강진	20090765	마이클샌델
2	학생정보	김강진		20090765	
3	도서대출	묘림대연구	이경택	20090123	KJ KIM
4	학생정보	이경택		20090123	

모든 자료를 하나의 표로 정리한 것
-> 구분하기 위한 칼럼의 추가, 빈칸 데이터 존재, 중복데이터 증가 등
-> 관리면에서나 시스템리소스 면에서나 매우 비효율적

도서대출정보

대출코드	이름	대여자학번	저자
A1	정의란무엇인가	20090765	마이클샌델
A2	묘림대연구	20090123	KJ KIM

도서에 관한 내용만 따로 추려서 만든 도서대출정보 테이블
PK는 대출코드

학생정보

학번	이름
20090765	김강진
20090123	이경택

학번은 학생마다 유일하므로 따로 key를 하나 더 가질 필요 X

학번을 통해 책 빌린 사람의 이름을 알 수 있고, 반대로 학생 이름을 통해 무슨 책을 빌렸는지를 두 표를 오고가며 알 수 있음

용어	설명	예시
엔터티	표이름	도서대출정보
인스턴스	표가 실현된 하나의 관측치, 행 1개	대출 1회 (A1 대출)
속성	칼럼	도서명, 저자
관계	두 표의 관계, 외래키가 중요한 역할을 함	학생이 도서를 빌림
주식별자	표의 각행을 유일하게 구분케하는 것	대출코드
외래식별자	표끼리 연결해주는 다른 표의 주식별자	대여자학번

4. RDBMS

https://gitmind.com/app/flowchart/3c22cc18eb2aa813d36d7524eef143e3

실습 – UML로 나타내보기

Customers

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Categories

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads

Employees

EmployeeID	LastName	FirstName	BirthDate	Photo	Notes
1	Davolio	Nancy	1968-12-08	EmpID1.pic	Education includes a BA in psychology from Colorado State University. She also completed (The Art of the Cold Call). Nancy is a member of 'Toastmasters International'.
2	Fuller	Andrew	1952-02-19	EmpID2.pic	Andrew received his BTS commercial and a Ph.D. in international marketing from the University of Dallas. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager and was then named vice president of sales. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.

OrderDetails

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5

Orders

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	1996-07-04	3
10249	81	6	1996-07-05	1
10250	34	4	1996-07-08	2

Products

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10

Shippers

ShipperID	ShipperName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

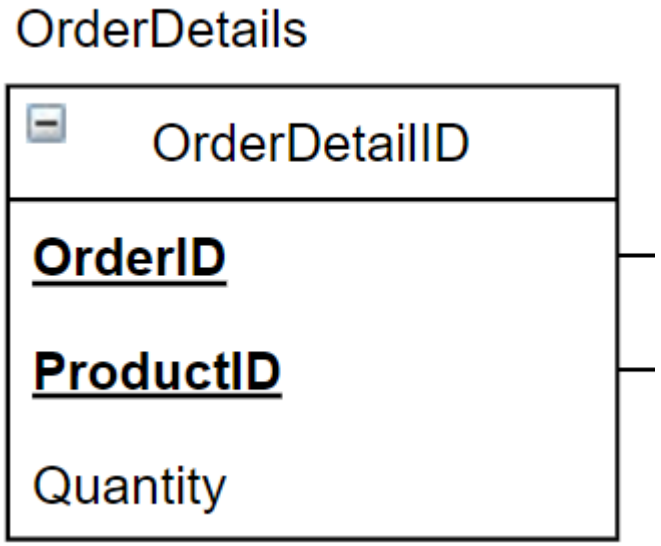
Suppliers

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	Londona	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA

4. RDBMS

https://gitmind.com/app/flowchart/3c22cc18eb2aa813d36d7524eef143e3

실습 - UML로 나타내보기

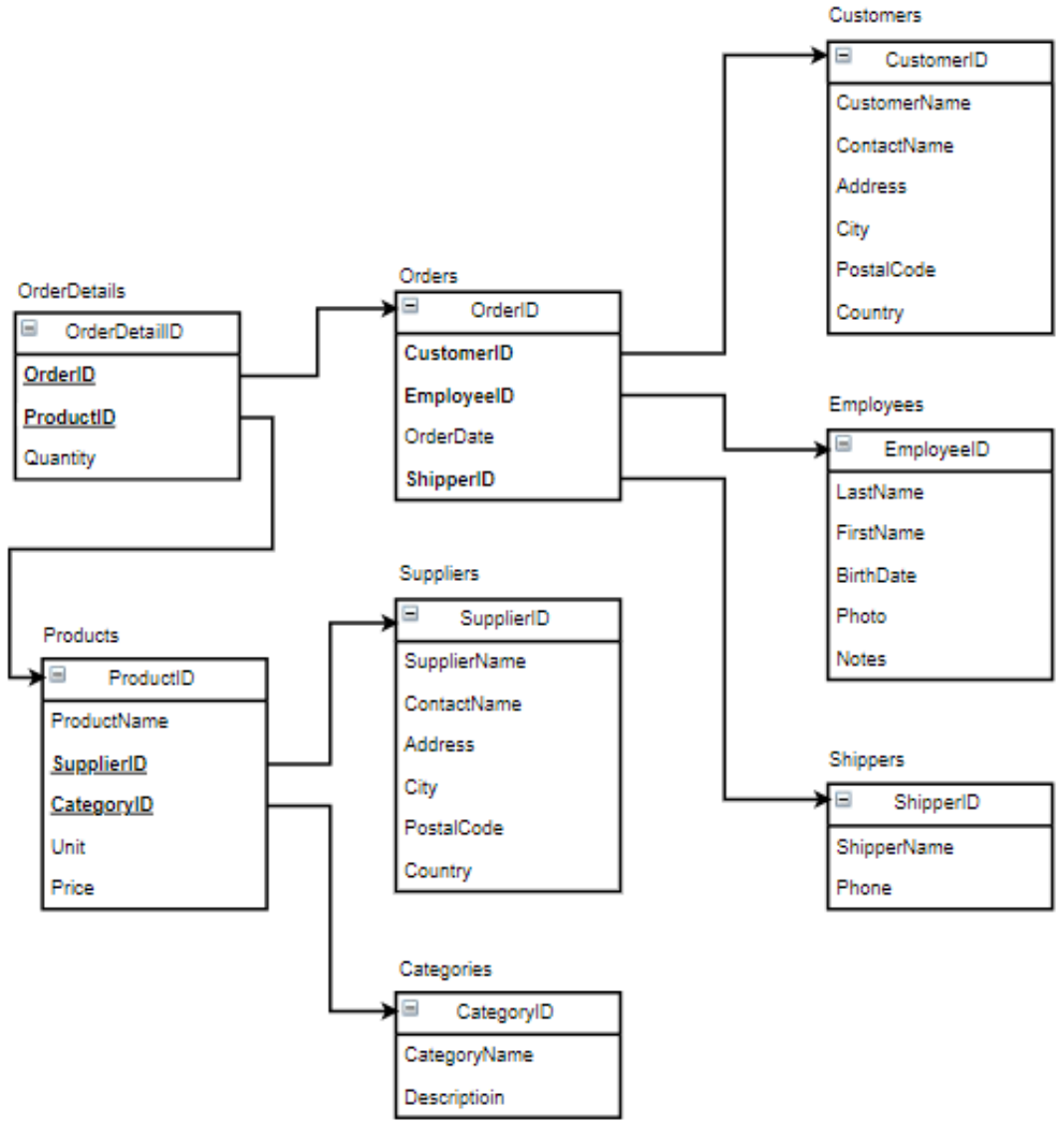


- Entity와 Attribute 나타내기
- Primary Key 표시
- Foreign Key 표시
- Foreign Key가 참조하는 Primary Key 표시
- 식별/비식별 표시

4. RDBMS

https://gitmind.com/app/flowchart/3c22cc18eb2aa813d36d7524eef143e3

실습 – UML로 나타내보기



5. SQL Query

명령어 종류	명령어	설명
데이터 조작어 (DML : Data Manipulation Language)	SELECT	데이터베이스에 들어 있는 데이터를 조회하거나 검색하기 위한 명령어(RETRIEVE)
	INSERT UPDATE DELETE	데이터베이스 테이블에 들어 있는 데이터에 변형을 가하는 종류(삽입,수정, 삭제)
데이터 정의어 (DDL : Data Definition Language)	CREATE ALTER RENAME TRUNCATE	테이블과 같은 데이터 구조를 정의하는 데 사용(생성, 변경, 삭제, 이름변경)
데이터 제어어 (DCL : Data Control Language)	GRANT REVOKE	데이터베이스에 접근하고 객체들을 사용하도록 권한 주고 회수하는 명령어
트랜잭션 제어어 (TCL: Transaction Control Lagnuage)	COMMIT ROLLBACK SAVEPOINT	논리적인 작업의 단위를 묶어서 DML에 의해 조작된 결과를 작업단위(트랜잭션)별로 제어하는 명령어

5. SQL Query

데이터 조작어 (DML : Data Manipulation Language)

테이블 예제(users)	name	age	department
1	Kim	22	CyberSecurity
2	Park	23	Computer Science
3	Choi	22	Media
4	Nam	23	Economics

INSERT

```
# 1번째 방법
INSERT INTO 테이블이름(필드이름1, 필드이름2, 필드이름3, ...)
VALUES (데이터값1, 데이터값2, 데이터값3, ...);
# 2번째 방법
INSERT INTO 테이블이름
VALUES (데이터값1, 데이터값2, 데이터값3, ...);
```

```
insert into Users (name, age, department)
values ('Kim', 22, 'CyberSecurity');
insert into Users
values ('Kim', 22, 'CyberSecurity');
```

SELECT

```
SELECT 필드이름 FROM 테이블이름 [WHERE 조건];
```

```
select name from users; select name, age from users
where department='CyberSecurity';
```

UPDATE

```
UPDATE 테이블이름
SET 필드이름1=데이터값1, 필드이름2=데이터값2, ...
WHERE 필드이름=데이터값;
```

```
update Users set age=23 where name = 'Kim';
```

DELETE

```
DELETE FROM 테이블이름 WHERE 필드이름=데이터값;
```

```
delete from Users where name = 'Kim';
```

5. SQL Query

데이터 정의어 (DDL : Data Definition Language)

CREATE

CREATE로 먼저 대학교 데이터베이스 만들기

```
CREATE DATABASE 데이터베이스 이름;
```

```
create database University;
```

대학교 데이터베이스를 사용한다고 선언하자

```
USE 데이터베이스 이름
```

```
use University
```

데이터베이스 안에 테이블을 생성

```
CREATE TABLE 테이블이름 (
 필드이름1 필드타입1,
 필드이름2 필드타입2,
...
)
```

```
create table employees(
name varchar(20) not null primary key,
age int,
department varchar(20),
);
```

ALTER

```
ALTER 개체형식 개체명 [매개변수]
ALTER DATABASE 데이터베이스이름 CHARACTER SET=문자집합이름
ALTER DATABASE 데이터베이스이름 COLLATE=콜레이션이름
```

```
alter table users add address VARCHAR(20); # 컬럼 추가
alter table users drop column age; #컬럼 삭제
alter table users modify column department int; # 컬럼 타입 수정
```

DROP / TRUNCATE

```
DROP DATABASE 데이터베이스이름  # 데이터베이스를 삭제
DROP TABLE 테이블이름   #테이블을 삭제
TRUNCATE TABLE 테이블이름 # 테이블은 그대로 두고, 데이터만 삭제
```

RENAME

```
RENAME TABLE 기존 테이블 이름 TO 바꿀 테이블 이름;
```

1.NOT NULL : 해당 필드는 NULL 값을 저장할 수 없게 됩니다.
2.UNIQUE : 해당 필드는 서로 다른 값을 가져야만 합니다.
3.PRIMARY KEY : 해당 필드가 NOT NULL과 UNIQUE 제약 조건의 특징을 모두 가지게 됩니다.
4.FOREIGN KEY : 하나의 테이블을 다른 테이블에 의존하게 만듭니다.
5.DEFAULT : 해당 필드의 기본값을 설정합니다.

5. SQL Query

데이터 제어어 (DCL : Data Control Language)

사용자 생성

```
CREATE USER '유저 이름'@'접속장소(IP)' identified by '비밀번호';
```

사용자 삭제

```
DROP USER '유저이름'@'접속장소'
```

사용자 권한 부여

```
GRANT '권한들' ON 'DB이름', '테이블' TO '유저 이름'@'접속장소'
# DB나 Table 자리에 * 을 넣어주게 되면, '모든'이라는 뜻
```

사용자 권한 확인

```
SHOW GRANTS FOR '유저 이름'@'접속장소'
```

사용자 권한 삭제

```
REVOKE ALL ON *.* FROM '유저 이름'@'접속장소'
```

트랜잭션 제어어 (TCL: Transaction Control Lagnuage)

COMMIT

정상적으로 종료되어 트랜잭션이 수행한 변경 내용을
데이터베이스에 반영하는 연산

ROLLBACK

하나의 트랜잭션 처리가 비정상적으로 종료되어
데이터베이스의 일관성이 깨졌을 때,
트랜잭션이 행한 모든 변경 작업을 취소하고
이전 상태로 되돌리는 연산

LOCK

트랜잭션이 수행되는 동안 특정 데이터에 대해서 다른
트랜잭션이 동시에 접근하지 못하도록 제한하는 기법

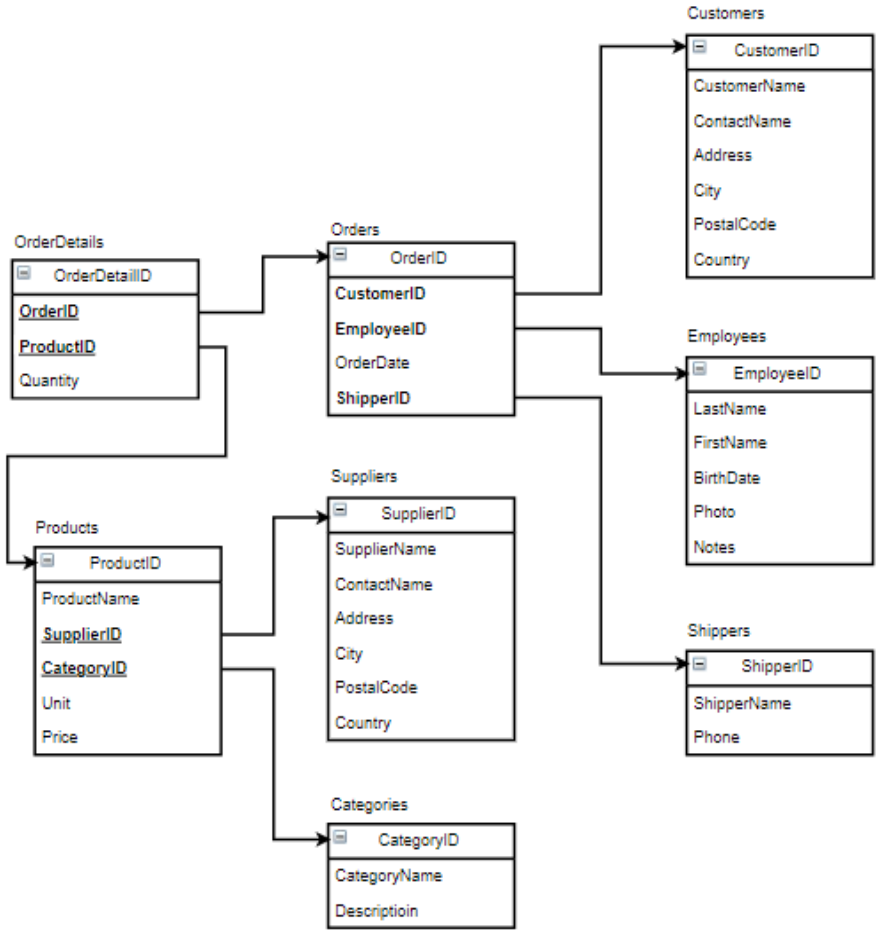
SAVEPOINT

현재의 트랜잭션을 작게 분할하는 명령어
ROLLBACK TO SAVEPOINT 문을 사용하여
지정한 곳까지 ROLLBACK 가.

5. SQL Query

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

실습



- 'Blauer See Delikatessen' 이름을 가진 Cutomer의 국적 출력

```
SELECT Country
from Customers
where CustomerName = 'Blauer See Delikatessen';
```

- OrderID가 '10248'인 주문건의 개수 출력

```
SELECT count(*)
FROM OrderDetails
where OrderId='10248';
```

- OrderID가 '10269'인 주문을 처리한 ShipperName출력

```
select ShipperName
from Orders as a, Shippers as b
where a.OrderID='10269'
        and a.ShipperID = b.ShipperID;
```

```
select ShipperName
from Orders as a
join Shippers as b
on a.ShipperID = b.ShipperID
where a.OrderID='10269';
```

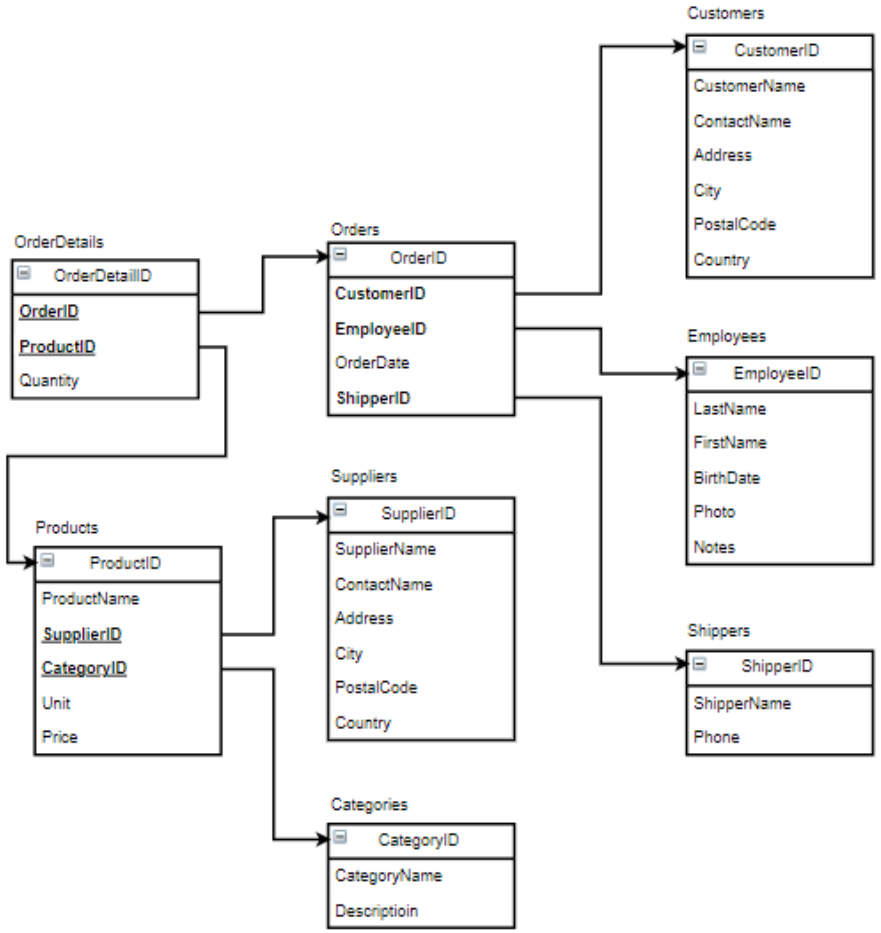
* 별칭(Alias)

- 컬럼의 소유주가 누구인지 명확하게 인식할 수 있음
- 중복된 컬럼명을 갖는 테이블을 JOIN 하여도 오류가 발생하지 않음
- 복잡한 SQL문에 대해 각각의 alias까지 숙지해야 하는 번거로움이 있음 (→ 별칭을 잘 지어줘야 함)
- 별칭이 바뀌면 각각의 컬럼에 대해 붙은 별칭을 바꿔줘야 함

5. SQL Query

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

실습



- 상품 카테고리별 상품의 개수 출력

```
SELECT CategoryID, count(*) as ProductQuantity
from Products
group by CategoryID;
```

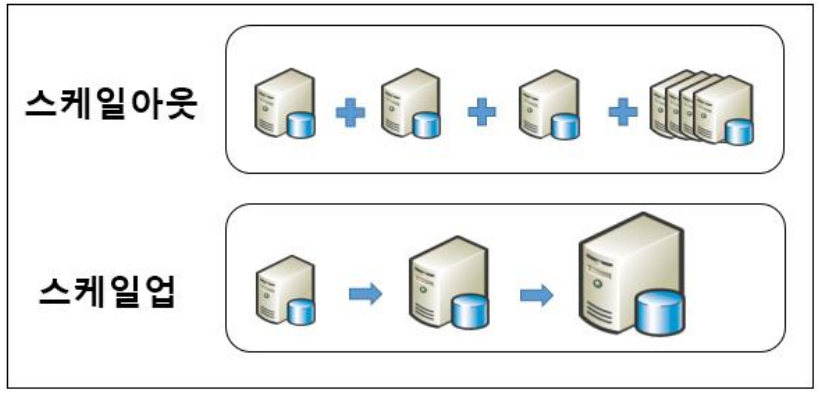
- 총 판매 금액이 10000 이상인 CustomerName 출력

```
SELECT CustomerName, sum(Price*Quantity) as SalesPrice
FROM Customers C
join Orders O
on C.CustomerID = O.CustomerID
join OrderDetails D
on O.OrderId = D.OrderID
join Products P
on D.ProductID = P.ProductID
group by C.CustomerID
having sum(Price*Quantity)>= 10000 ;
```

6. NoSQL

Not Onlys SQL(비-관계형 데이터베이스)로 고정적인 형태의 SQL 뿐만 아니라 새로운 형태도 가능하다는 의미

등장 배경



- 수직적 확장(스케일 업) :

더 큰 장비에 더 많은 프로세서와 디스크 스토리지, 메모리를 장착하는 방법

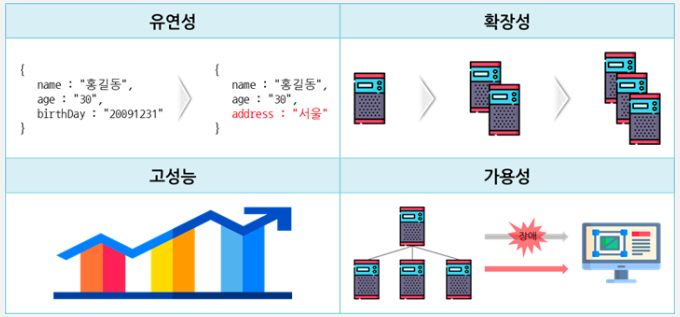
- 비용을 포함하여 여러가지 측면에서 한계 존재

- 수평적 확장(스케일 아웃) :

작은 장비를 많이 모아 클러스터 를 구성하는 방법 (분산 처리)

- 비용이 감소하며, 장비가 고장나도 중단이 발생하지 않 는 높은 가용성

대용량 데이터 관리 작업에서 중요한 관리 시스템의 특징



- **유연성** : 스키마 선언 없이 필드의 추가 및 삭제가 자유로운 Schema-less 구조
- **확장성** : 스케일 아웃에 의한 서버 확장이 용이
- **고성능** : 대용량 데이터를 처리하는 성능이 뛰어나다
- **가용성** : 여러 대의 백업 서버 구성이 가능하여 장애 발생 시에도 무중단 서비스가 가능

=> 점차 분산 처리 방식의 클러스터가 활성화되고, RDBMS는 클러스터에 적합한 설계가 아니어서 NoSQL이 등장

6. NoSQL

특징

- 관계형 모델을 사용하지 않으며 테이블 간의 조인 기능 없음
- 비SQL 인터페이스를 통한 데이터 접근
- 대부분 여러 대의 데이터베이스 서버를 클러스터링하여 하나의 데이터베이스를 구성
- 관계형 데이터베이스에서 지원하는 Data처리 완결성(ACID) 미보장
- 데이터의 스키마와 속성들을 다양하게 수용 및 동적 정의 (Schema-less)
- 데이터베이스의 중단 없는 서비스와 자동 복구 기능지원
- 다수가 오픈 소스로 제공
- 확장성, 가용성, 높은 성능
- BASE

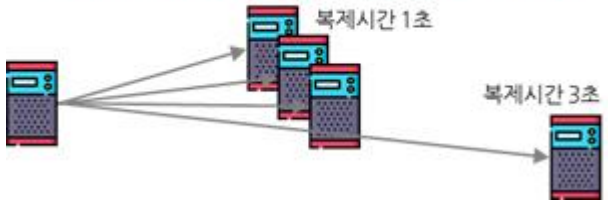
Basically Available(가용성)

Soft State(소프트 상태)

〈계좌이체 예시〉

A 사용자 트랜잭션	B 사용자 트랜잭션
① 황정민 계좌 : -300 ③ 이정재 계좌 : +300	② 이정재 계좌 : +100

Eventually Consistent(결과적 일관성)



6. NoSQL

장점

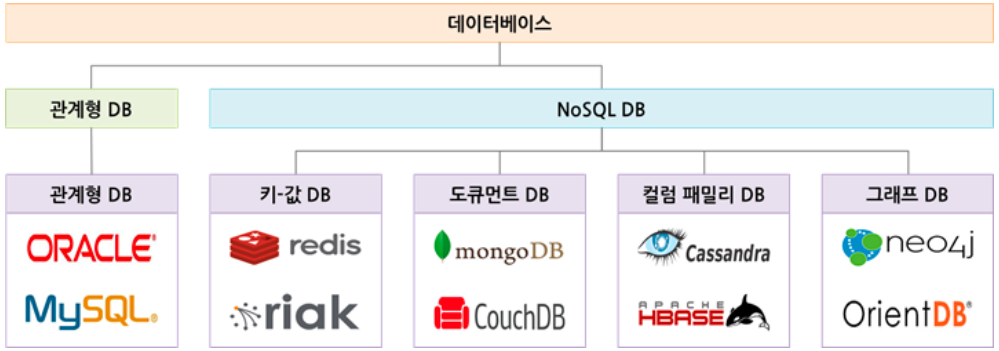
- 스키마가 없기 때문에 유연 -> 언제든지 데이터를 조정하고 새로운 필드 추가 가능
- 수직 및 수평적 확장이 가능하므로 데이터베이스가 애플리케이션에서 발생시키는 모든 읽기 / 쓰기 요청 처리가 가능
- 데이터는 애플리케이션이 필요로 하는 형식으로 저장 -> 데이터를 읽어오는 속도가 빨라짐

단점

- 데이터베이스 일관성에 약하다. 이 일관성을 가용성, 분할 용인, 속도와 맞바꾸었다
  - 분할 용인 : 네트워크 실패로 인하여 임의의 분할이 발생해도 시스템은 계속적으로 동작해야 한다.
    임의의 메시지들의 손실 또는 시스템의 부분 실패에도 불구하고, 시스템은 지속적으로 동작
- key값에 대한 입출력만 지원
- 스키마가 정해져 있지 않아, 데이터에 대한 규격화가 되어 있지 않다.
- 데이터가 여러 컬렉션에 중복되어 있어서 데이터를 UPDATE 하는 경우 모든 컬렉션에서 수행해야하기 때문에 느리다.
- 데이터 중복으로 인한 수정 작업의 번거로움

6. NoSQL

NOSQL 구조 4가지



Key-Value Store

- 데이터가 Key와 Value로 구성된 배열구조의 DB로 가장 단순
- 속도가 빠르고 분산 저장에 용이
- 값에 모든 데이터 타입 허용 가능
- 데이터 조회 시 Key로 접근 -> Key 중복 허용X

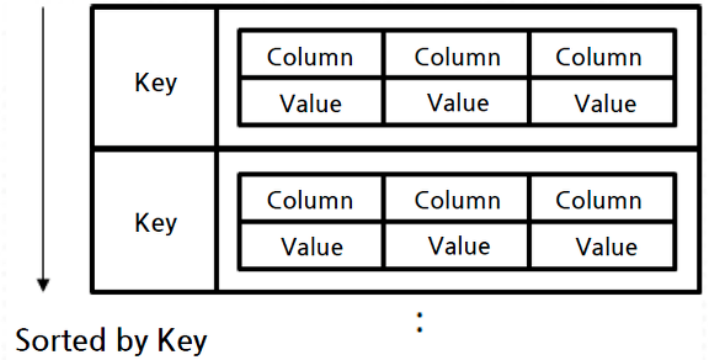
Document Database Store

- 데이터가 Key와 Document로 구성
- 복잡한 계층구조 표현 용이
- Sorting, Join, Grouping 등 RDB와 같은 추가 기능 지원되기도 함(ex_MongoDB)
- NoSQL중 가장 인기있고 JSON 기반 통신을 하는 HTTP 웹 서버의 경우 편리하게 이용 가능

6. NoSQL

NOSQL 구조 4가지

Wide Column Database Store



- Column-family Model 기반 DB
- Key에서 필드를 결정
- Key 기준으로 Sorting되어 저장

  Order by 제공하지 않는 NoSQL에서 장점
- RDBMS에서 Attribute가 계층적인 셈

Graphh Database Store

- 데이터를 Node와 Edge, Property와 함께

  그래프 구조를 사용하여 데이터 표현하고 저장
- 개체와 관계를 그래프 형태로 표현 -> 관계형 모델
- 데이터 간 관계가 탐색의 키일 경우 적합
- Ex) 소셜 네트워크, 추천 엔진, 패턴 인식

7. RDBMS vs NOSQL

CAP 이론

데이터베이스 시스템은 일관성(Consistency), 가용성(Availability), 분할 내구성(Partition Tolerance)의 3가지 특성 중 2가지 특성만을 충족 할 수 있고 3가지 모두 충족할 수 없다는 이론

- Consistency(일관성): 동시성, 동일성. 다중 클라이언트에서 같은 시간에 조회하는 데이터는 항상 동일한 데이터임을 보증해야 함
- Availability(가용성): 모든 클라이언트의 읽기와 쓰기 요청에 대 하여 항상 응답이 가능해야 함을 보증,
  클러스터 내 몇 개의 노드 가 망가지더라도 다른 노드가 클라이언트 요청 수행
- Tolerance to Partitions(파티션 허용): 지역적으로 분할된 네트 워크 환경에서 동작하는 시스템이 있다고 할 때,
  두 지역 간의 네트워크가 단절되거나 네트워크 데이터의 유실이 일어나더라도, 각 지역 내의 시스템은 정상적으로 동작.

Mysql

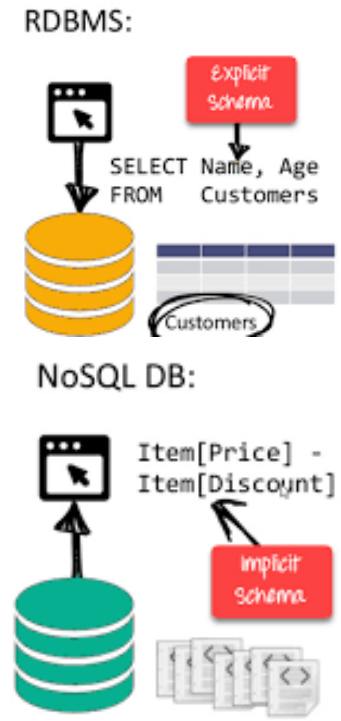
위의 삼각형 그래프에서 CA에 속하지만 cluster설정을 지원하여 cluster 설정을 하면 CP 만족

MynamoDB

aws에서 지원하는 key/Value의 NoSQL로 AP에 속함
strongly consistent 설정을 하게 CP
가장 최신의 데이터를 반드시 리턴해서 모든 노드를 찾기 때문에 가용성은 조금 떨어지는 것

7. RDBMS vs NOSQL

구분	RDBMS	NOSQL
처리 데이터	정형 데이터	정형 데이터, 비정형(반정형 포함) 데이터
대용량 데이터	대용량 처리 시 성능 저하	대용량 데이터 처리 지원
스키마	미리 정해진 스키마 존재. 수직적	스키마가 없거나 변경 자유로움. 수평적
트랜잭션	트랜잭션으로 일관성 유지 보장(ACID)	트랜잭션 지원하지 않아 일관성 유지 보장 어려움(BASE)
검색 기능	조인 등의 복잡한 검색 기능 제공	단순 데이터 검색 기능 제공
확장성	클러스터 환경에 적합하지 않음	클러스터 환경에 적합
라이선스	고가의 라이선스 비용	오픈 소스
대표 사례	Oracle, MySQL, MS SQL 서버 등	카산드라, 몽고DB, H베이스



언제 사용해야할까?

RDBMS

- 데이터 구조가 명확하며 변경될 여지가 없고 명확한 스키마가 중요한 경우
- 중복 데이터가 없어 변경이 용이하기 때문에 관계를 맺고 있는 데이터가 자주 변경이 이루어지는 시스템에 적합

NOSQL

- 정확한 데이터 구조 알 수 없고 데이터가 변경/확장이 될 수 있는 경우
- Update 가 많이 이루어지지 않는 시스템.
- 막대한 데이터를 저장해야 되는 시스템

엔지니어링 19기 BASE session 5주차

감사합니다.

2022.08.17

엔지니어링 18기 이현진