SQL 개발자 D 이론

모델링

- 데이터베이스의 모델링은 현실세계를 단순화하여 표현하는 기법이다.

모델링 특징

추상화

- 현실 세계를 일정한 형식으로 표현하는 것이다. 즉 아이디어나 개념을 간략하게 표현하는 과정

단순화

- 복잡한 현실 세계를 정해진 표기법으로 단순하고 쉽게 표현한다는 의미

명확화

- 불분명함을 제거.명확하게 해석할 수 있도록 기술

> 💡 **모델링 이해: 현실세계를 추상화,단순화,명확화하기 위해 일정한 표기법에 의해 표현하는 기법**

모델링 세가지 관점

데이터 관점

- **데이터 위주의 모델링. 어떤 데이터들이 업무와 얽혀있는지, 데이터간에 어떤 관계가 있는지에 대해서 모델링**

프로세스 관점

- **프로세스 위주의 모델링. 업무가 실제로 처리하고 있는일이 무엇인지, 앞으로 처리해야하는 일은 무엇인지 모델링**

데이터와 프로세스의 상관 관점

- 데이터와 프로세스의 관계 위주 모델링. 프로세스의 흐름에 따라 데이터가 어떤 영향을 받는지 모델링

💡 데이터 품질 보장을위해 데이터 모델링시 유의사항

중복 : 같은 데이터가 여러 엔터티(테이블)에 중복으로 저장되는현상 지양해야함
비유연성 : 사소한 변경에도 데이터 모델이 수리로 변경해야하는 상황이 올 수 있음
유지보수에 어려움 증가. 데이터모델과 프로세스 분리 → 유연성 높임
비일관성 : 너무 중복이 없는 경우에도 비일관성 발생
다른 데이터와 연관 고려
데이터 간의 연관 관계 명확하게 정의할것.

모델링의 세가지 단계

개념적 데이터 모델링

- 전체적 데이터 모델링 수행시 행해짐 레벨이 가장 높은 모델링
  업무중심적,포괄적인 수준 모델링 진행

논리적 데이터 모델링

- 재사용이 가장 높은 모델링. Key,속성,관계 등을 모두 표현하는 단계

물리적 데이터 모델링

- 실제 데이터베이스로 구현할 수 있도록 성능,가용성등 물리적 성격 고려하여 모델을 표현하는 단계

데이터의 독립성

3단계 스키마 구조

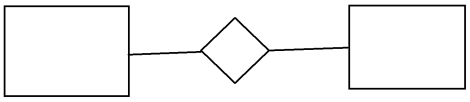
- 외부 스키마
  - 사용자의 관점 : 각 사용자가 보는 데이터베이스의 스키마를 정의
- 개념 스키마
  - 통합된 관점 :데이터베이스에 저장되는 데이터들을 표현하고 데이터들 간의 관계를 나타낸다.
- 내부 스키마
  - 물리적인관점 : 실직적인 데이터의 저장구조나 컬럼 정의,인덱스 등이 포함된다.

💡 3단계 스키마구조로 나누는 이유
데이터베이스에 대한 사용자들의 관점과 실제로 표현되는 물리적인 방식을 분리 독립석으 보장하기 위한것
논리적 독립성 : 개념스키마가 변경되어도 외부스키마는 영향을 받지 않음
물리적 독립성 : 내부스키마가 변경되어도 외부/개념 스키마는 영향을 받지 않음

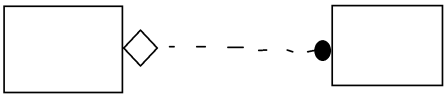
ERD

ERD 표기방식

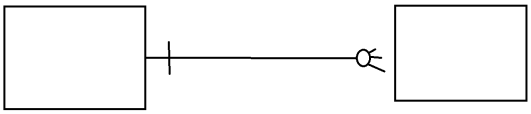
Peter chen: 실무에서 사용하는 경우는 드뭄



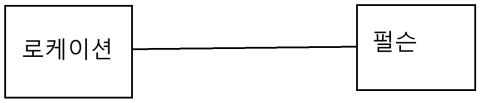
IDEF1X: 실무에서 사용하는 경우도 있음 ERWin에 사용되는 모델



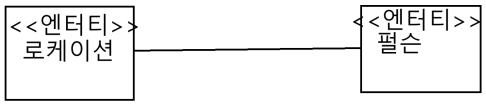
IE/Crows'Foot: 까마귀발 표기법 가장많이 사용 ERWin,ERStudio에서 사용되는모델



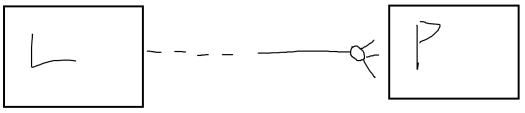
Min-Max/ISO : 각 엔터티의 참여도를 좀더 상세하게 나타내는 표현법



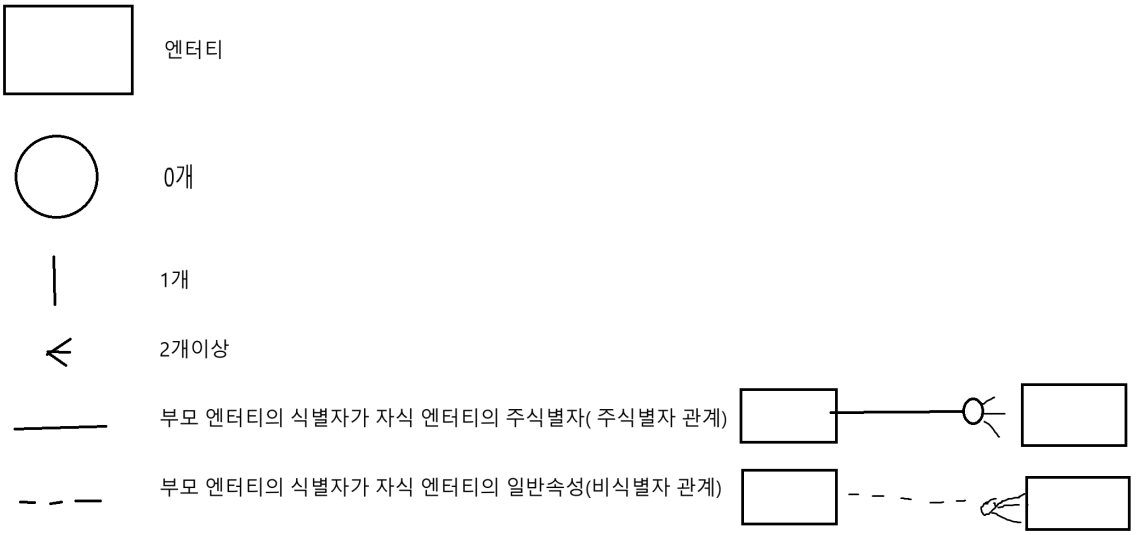
UML : 소프트 웨어 공학에서 주로 사용되는 모델



Case Method/ Barker : Oracle에서 사용되는 모델 CrowsFoo과 비슷



IE/Crows'Foot



ERD 작성 순서

1. **엔터티를 도출하고 그린다**
2. **엔터티를 적절하게 배치한다**
3. **엔터티 간의 관계를 설정한다**
4. **관계명을 기입한다**
5. **관계의 참여도를 기입한다**
6. **관계의 필수/선택 여부를 기입한다.**

엔터티

- **엔터티 : 식별이 가능한 객체 ex) 주문(주문번호,회원번호,주문일자) 주문상품(주문번호FK), 상품번호, 주문수량)**

> 💡 **엔터티 : 테이블 인스턴스: ROW(행) 속성: Column(열)**

엔터티 특징

- **업무에 쓰이는 정보여야함**
- **유니크함을 보장할 수 있는 식별자가 있어야함 식별이 모호하면 안됨.**
- **2개 이상의 인스턴스(행)을 가지고 있어야함**
- **속성을 가져야함 상세하게 나타낼 수 있는 속성 필요**
- **다른 엔터티와 1개이상의 관계를 가지고 있어야함 연관성이 있어야함**

엔터티 분류

유형 vs 무형

유형 엔터티	물리적인 형태존재 안정적 지속적 ex) 상품 회원 등
개념 엔터티	물리적인 형태 없음 개념적 ex) 부서, 학과
사건 엔터티	행위를 함으로 써 발생, 빈번함 통계자료로 이용가능 ex) 주문, 이벤트응모

발생시점

기본 엔터티	업무에 원래 존재하는정보 독립적으로 생성되며, 자식 엔터티를 가질 수 있음 ex 상품,사원,부서등
중심 엔터티	기본 엔터티로부터 파생되고 행위 엔터티 생성 업무에 있어서 중심적인 역할을 하며 데이터의 양이 많이 발생 ex 주문,매출계약
행위 엔터티	2개 이상의 엔터티로부터 파생 데이터가 자주 변경되거나 증가할 수있음 ex 주문내역 이벤트응모이력등

💡 **엔터티 이름 정할때 주의할 점**

- **업무에서 실제로 쓰이는 용어 사용**
- **한글은 약어 사용 금지 영문은 대문자로 표기**
- **단수 명사로 표현 띄어쓰기 안함**
- **다른엔터티와 의미상중복X**
- **해당엔터티가 갖고있는 데이터가 무엇인지 명확하게 표현**

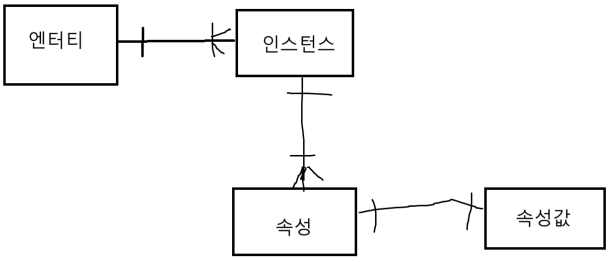
속성 (Attribute 열)

- **엔터티의 특징을 나타내는 최소의 데이터 단위**

속성값

- **인스턴스를 구체적으로 나타내주는 데이터 (여러개 속성값을 갖는경우 별도의 엔터티로 분리)**

엔터티(테이블),인스턴스(행),속성(열),속성값의 관계



- **한개의 엔터티 = 두개이상의 인스턴스**
- **한개의 인스턴스 = 두개 이상의 속성**
- **한 개의 속성= 하나의 속성값**

분류

- **특성에 따른 분류**
  - 기본속성
    - 업무 프로세스 분석 바로 정의 가능한 속성

    - 설계속성
        - 설계하다보니 필요하다고 판단되어 도출해낸 속성 유니크함부여
    - 파생속성
        - 특정한 규칙으로 변형하여 생성한 속성 계산값 가공된 값
- **구성방식에 따른 분류**
    - PK 속성
        - 엔터티의 인스턴스들을 식별할 수 있는 속성
    - FK 속성
        - 다른 엔터티의 속성에서 가져온 속성
    - 일반속성
        - PK,FK를 제외한 속성

도메인

- **속성이 가질 수 있는 속성값의 범위**

> 💡 **용어 사전 : 어떤 시스템이든 속성명은 업무와 직결되는 항목. 속성이름 정확,직관적으로 부여 용어 혼란 없앰**
> **시스템 카탈로그 : 시스템 자체 데이터 관련 데이터베이스 조회만 가능**

관계

- **엔터티와 엔터티와의 관계를 의미. 연관성 타입분류 존재관계 행위관계**
- 존재관계
    - 존재자체로 연관성 학생,학과
- 행위관계
    - 특정 행위를 하으로써 연관성이 생기는 관계 학생, 출석부 출석

표기법

- 관계명
    - 관계의 이름. 어떠한 관계인지 각엔터티 관점 관계명 하나씩 명확한 문장
- 관계차수
    - 관계에 참여하는 수 1:1 1:M M:N
- 관계선택사양
    - 필수인지 선택인지의 여부 필수적관계(반드시 존재해야하는 관계), 선택적 관계(없을 수도 있는 관계)

식별자

- **인스턴스를 구분가능하게 만들어주는 대표격인 속성**
- 주식별자
    - 기본키 PK 해당하는 속성
        - 유일성
            - 유니크함 부여 식별

        - 최소성
            - 유일성 보장 최소 개수의 속성
        - 불변성
            - 속성값 변경 X
        - 존재성
            - 속성값 NULL X

분류

- 대표성여부
    - 주식별자 : 다른 엔터티와 참조관계로 연결
    - 보조식별자 :다른 엔터티와 참조 관계로 연결되지 않음
- 스스로 생성되었는지 여부
    - 내부 식별자
        - 엔터티 내부에서 스스로 생성된 식별자
    - 외부식별자
        - 다른 엔터티에서 온 식별자. 다른 엔터티와의 연결고리 역할
- 단일 속성의 여부
    - 단일식별자
        - 하나의 속성구성된 식별자
    - 복합 식별자
        - 두 개 이상의 속성으로구성된 식별자
- 대체 여부
    - 원조 식별자
        - 업무프로세스에 존재하는 식별자 가공되지않은 식별자(본질식별자)
    - 대리 식별자
        - 주식별자의 속성이 두 개 이상인 경우 하나로 묶어서 사용 (인조식별자)

식별자 관계 vs 비식별자 관계

- 식별자 관계
    - 부모 엔터티의 식별자가 자식 엔터티의 주식별자가 되는 관계
- 비식별자 관계
    - 부모 엔터티의 식별자가 자식 엔터티의 주식별자가 아닌 일반속성이 되는 관계

챕터 1 끝

정규화

- 데이터 정합성(데이터 정확성,일관성 유지보장)을 위해 엔터티를 작은 단위로 분리하는 과정이다.

제1 정규형

- 모든 속성은 반드시 하나의 값만 가져야 한다.

제 2정규형

- 모든 일반속성은 반드시 모든 주식별자에 종속되어야 한다.

제 3정규형

- 주식별자가 아닌 모든 속성 간에는 서로 종속될 수 없다.

> 💡 **주의사항 !!! 지나친 정규화는 오히려 성능을 저하 시킬 수 있다.**

반 정규화

- 데이터의 조회 성능을 향상시키기 위해 데이터의 중복을 허용하거나 데이터를 그룹핑하는 과정이다.

> 💡 **입력, 수정, 삭제 성능은 저하 될 수 있음. 정규화가 끝난 후 실행**

테이블 반 정규화

테이블 병합	**1:1 관계 테이블 병합** **1:M 관계 테이블 병합** **슈퍼 서브타입 테이블 병합**
테이블 분할	**테이블 수직 분할(속성)** **테이블 수평 분할(인스턴스 분할, 파티셔닝)**
테이블 추가	**중복 테이블 추가** **통계 테이블 추가** **이력 테이블 추가** **부분 테이블 추가**

- 테이블 병합
  - Join이 필요한 경우가 많아 테이블을 통합하는것이 성능 측면에서 유리할 때 고민한다.
- 테이블 수직 분할
  - 엔터티 일부 속성을 별도의 엔터티로 분할(1:1관계성립)한다.
- 테이블 수평 분할
  - 엔터티의 인스턴스를 특정기준으로 별도의 엔터티로 분할 한다.
- 테이블 추가
  - 중복 테이블 추가 : 성능상 필요시 엔터티 추가
  - 통계 테이블 추가
  - 이력 테이블 추가
  - 부분 테이블 추가

컬럼 반정규화

- 중복 컬럼 추가
  - 업무 프로세스상 Join이 필요한 경우
- 파생 컬럼 추가
  - 프로세스 수행시 부하가 염려되는 계산값을 미리 컬럼으로 추가
- 이력 테이블 컬럼 추가
  - 조회 기준이 될 것으로 판돤되는 컬럼을 미리 추가

관계 반정규화

- 업무 프로세스상 Join이 필요한 경우가 많아 중복관계를 추가하는것 성능측면 유리

트랜잭션(Transaction)

- 데이터를 조작하기 위한 하나의 논리적인 작업 단위이다.

NULL

- NULL은 존재하지 않음. 즉 값이 없음을 의미한다.

SQL 기본 및 활용

관계형 데이터베이스개요

데이터베이스

- 데이터를 저장하는 공간

관계형 데이터베이스

- **RDB(Relational Database) 관계형 데이터 모델에 기초를 둔 데이터베이스**

테이블

- 모든 데이터를 2차원 테이블 형태로 표현

  세로 : 열 Column

  가로 : 행 Row

SQL(Structured Query Language)

- 관계형 데이터 베이스를 다루기 위한 언어

SELECT문

- 저장되어 있는 데이터를 조회하고자 할 때 사용하는 명령어이다.

  SELECT 컬럼1, 컬럼2 FROM 테이블 WHER 컬럼1 = 조건;

산술 연산자

- 수학에서 사용하는 사칙연산의 기능을 가진 연산자 NUMBER DATE유형 데이터에 사용

연산자	의미	우선순위
()	괄호로 우선순위 조정	1
*	곱하기	2
/	나누기 (0으로 나눌경우 에러)	2
+	더하기	3
-	빼기	3
%(SQL Server)	나머지(0으로나눌경우 null 반환	3

문자함수

- CHR(ASCII 코드)

LOWER(문자열)

- 문자열을 소문자로 변환해주는 함수이다.

```
SELECT LOWER('JENNIE') FROM DUAL;
```

UPPER(문자열)

- 문자열을 대문자로 변환해주는 함수

```
SELECT UPPER('JENNIE') FROM DUAL;
```

LTRIM(문자열[,특정문자]) []는 옵션 공백제거 왼쪽 또는 특정문자제거 왼쪽 기준

- 특정문자를 명시해주지 않으면 문자열의 왼쪽 공백 제거
  명시시 문자열 왼쪽부터 특정문자와 비교하여 특정문자 포함시 제거 포함되지 않으면 멈춤

```
SELECT LTRIM(' JENNIE') FROM DUAL;
```

RTRIM (문자열[,특정문자]) * 공백제거 오른쪽 또는 특정문자제거 오른쪽 기준

- 특정문자를 따로 명시 안할시 오른쪽 공백 제거
  오른쪽부터 한글자씩 특정문자 비교 특정문자 포함 제거

TRIM([위치] [특정문자] [FROM] 문자열)

- 오른쪽 왼쪽 문자열 공백 제거 LEADING OR TRAILING OR BOTH로 지정 특정문자 제거

SUBSTR(문자열, 시작점 [길이])

- 문자열 원하는 부분만 잘라서 반환해주는 함수이다.

LENGTH(문자열)

- 문자열의 길이를 반환해주는 함수이다.

REPLACE(문자열, 변경 전 문자열 [변경후 문자열]) * [ ] 옵션

- 문자열에서 변경 전 문자열을 찾아 변경 후 문자열로 바꿔주는 함수이다.
  변경 후 문자열을 명시해 주지 않으면 문자열에서 변경 전 문자열을 제거한다.

LPAD(문자열, 길이, 문자)

- 문자열이 설정한 길이가 될 때까지 왼쪽을 특정문자로 채우는 함수이다.

숫자 함수

ABS(수)

- 수의 절댓값을 반환해주는 함수이다.

SIGN(수)

- 수의 부호를 반환해주는 함수이다. 양수이면 1 , 음수이면 -1 , 0이면 0반환

ROUND(수[,자릿수] *[ ]는 옵션

- 수를 지정된 소수점 자릿수까지 반올림하여 반환해주는 함수이다.
  기본값은 0이며 반올림된 정수 반환, 음수일 경우 지정된 정수부를 반올림하여 반환

TRUNC(수 [,자릿수]) * [ ] 옵션

- 수를 지정된 소수점 자릿수까지 버림하여 반환해주는 함수이다.
  기본값은 0이며 버림된 정수로 반환 음수일 경우 정수부에서 버림 반환

CEIL(수)

- 소수점 이하의 수를 올림한 정수를 반환해주는 함수이다.

FLOOR(수)

- 소수점 이하의 수를 버림한 정수를 반환해주는 함수이다.

MOD(수1,수2)

- 수1을 수2로 나눈 나머지를 반환해주는 함수이다.
  단 수2가 0일경우 수1을반환한다

날짜 함수

SYSDATE

- 현재의 연, 월, 일, 시, 분 초를 반환해주는 함수이다.

EXTRACT(특정단위 FROM 날짜 데이터)

- 날짜데이터에서 특정단위(YEAR, MONTH, DAY, HOUR, MINUTE, SECOND)만을 출력해서 반환해주는 함수

ADD_MONTHS(날짜 데이터, 특정개월수)

- 날짜 데이터에서 특정 개월수를 더한 날짜를 반환해주는 함수이다.

변환함수

명시적 형변환과 암식적 형변환

- 명시적 형변환 : 변환 함수를 사용하여 데이터 유형 변환을 명시적으로 나타냄
  - 명시적 형변환에 쓰이는 함수
    - TO_NUMBEER(문자열)
      - 문자열을 숫자형으로 변환해주는 함수
    - TO_CHAR(수_날짜, [, 포맷]) * [ ]는 옵션
      - 수나 날짜형의 데이터를 포맷 형식의 문자형으로 변환해주는 함수이다.
    - TO_DATE(문자열,포맷)
    - 포맷 형식의 문자형의 데이터를 날짜 형으로 변환해주는 함수이다.
- 암시적 형변환 : 데이터베이스가 내부적으로 알아서 데이터 유형을 변환함

NULL관련 함수

NVL(인수1,인수2)

- 인수1의 값이 NULL일 경우 인수2를 반환하고 NULL이 아닐 경우 인수1을 반환해주는 함수이다.

NULLIF(인수1,인수2)

- 인수1과 인수2가 같으면 NULL을 반환하고 같지 않으면 인수1을 반환해주는 함수이다.

COALESCE(인수1,인수2,인수3...)

- NULL이 아닌 최초의 인수를 반환해주는 함수이다.

NVL(인수1,인수2,인수3)

- 인수1이 NULL이 아닌 경우 인수2를 반환하고 NULL인 경우 인수3을 반환하는 함수이다.

CASE

함수의 성격과 같기는 하지만 표현방식이 함수라기보다는 구문에 가까움

```
CASE 컬럼
    WHEN 조건1 THEN 값1
    WHEN 조건2 THEN 값2
    ELSE 값3
    END
```

WHERE 절

INSERT를 제외한 DML문을 수행할 때 원하는 데이터만 골라 수행할 수 있도록 해주는 구문이다.

SELECT, UPDATE나 DELETE도 마찬가지이다.

비교 연산자

연산자	의미	예시		부정비교연산자	의미	예시
=	같음	WHERE 컬럼 = 10		!=	같지않음	where 컬럼 != 10
<	작음	WHERE 컬럼 < 10		^=	같지 않음	where 컬럼 ^= 10
<=	작거나 같음	WHERE 컬럼 <=10		<>	같지 않음	where 컬럼 <> 10
>	큼	WHERE 컬럼 > 10		not 컬럼명 =	같지 않음	where not 컬럼 = 10
>=	크거나 같음	WHERE 컬럼 >= 10		not 컬럼명 >	크지 않음	where not 컬럼 = 10

SQL 연산자

```
BETWEEN A AND B   : A와 B 사이 (A,B 포함)                                    WHERE 컬럼
LIKE'비교 문자열' : 비교문자열을 포함 %는 문자열을 의미, '_'는 하나의 문자를 의미 WHERE 컬럼 LIKE '
                                    '_'혹은'%' 기호가 포함된 문자 검색시 ESCAPE 지정
IN(LIST)          : LSIT 중 하나와 일치                                      WHERE 컬럼
IS NULL           : NULL 값                                                  WHERE 컬럼

부정 연산자
```

```
NOT BETWEEN A AND B : A와 B 사이 아님 (A,B 포함 미포함)                    WHERE 컬럼 NO
NOT IN(LIST)        : LSIT 중 일치하는 것이 없음                          WHERE 컬럼 NO
IS NOT NULL         : NULL 값이 아님                                      WHERE 컬럼 I

논리 연산자

AND : 모든 조건이 TRUE여야 함
OR  : 하나 이상의 조건이 TRUE여야 함
NOT : TRUE면 FLASE이고 FALSE이면 TRUE
```