**1주차. 데이터베이스와 DBMS**

DIKW(데이터 생명주기)

-객관적 실재를 체계화한 것이 데이터(Data)

-데이터를 내 목적에 맞게 가공한 것이 정보(Information)

-정보를 집적하고 체계화하여 장래의 일반적 사용에 대비한 보편성 확보이 지식(Knowledge).

-지식을 가지고 추론을 하는 것이 지혜(Wisdom)

데이터베이스 시스템 주요 작업- 검색과 변경(삽입, 삭제, 수정) :검색을 잘 하기 위해 정리하는 것

데이터베이스 주요 개념 - **통합된** 데이터, **저장된** 데이터, 운영 데이터, 공용 데이터

데이터베이스의 주요 특징 - 실시간 접근성, 계속적인 변화, 동시 공유, 내용에 따른 참조

**2주차. DBMS의 역사와 주요 요소, 파일시스템과 DBMS의 차이**

1970년대 pc없음, 부정확 🡪 1980년대 네트워크 연결도 안됨. 엑셀느낌 🡪 1990년대 웹을 통해서 데이터베이스를 이용하여 통합관리 🡪 2000년대 인터넷을 이용해 주문, 불특정 다수의 고객유치 🡪 2010년대 각 목적에 따라 데이터베이스를 여러 개로 나눠 구축하고 서버로 관리

데이터처리 관점에서 바라본 정보 시스템의 발전

* 파일시스템(엑셀)

문제점: 독립적으로 파일을 다루기 때문에 중복 저장될 가능성이 있다.

동시에 파일을 다루기 때문에 데이터의 일관성 훼손 가능함.

* 데이터베이스 시스템

DBMS를 도입하여 통합관리(서버: 데이터를 가진 쪽, 클라이언트: 외부에서 데이터를 요청)

데이터를 위한 관리자가 한 명 있는 것 (일관성 유지, 복구, 동시 접근 제어 등)

* 웹 데이터베이스 시스템

외부의 클라이언트들이 정보를 요청할 수 있음 ex) 학교 데이터 관리자와 학생

* 분산 데이터베이스 시스템

대규모의 응용 시스템에 사용

DBMS가 필요한 이유 - 동시성, 무결성에 대한 문제를 해결하기 위해

파일시스템은 데이터를 파일 단위로 저장한다.

데이터 중복 - DBMS는 데이터를 공유하기 때문에 하나의 데이터만 관리한다. 중복 없음.

데이터 일관성 - 데이터를 넣어줄 때 파일시스템은 열의 이름 같은 것을 정하지 않고 만든다. 그래서 일관성이 떨어진다. DBMS 는 이런 것들이 정의되기 때문에 일관성 부분에서 좋다.

데이터 독립성 – 데이터와 데이터를 다루는 프로그램 간의 독립성을 말한다. 한마디로 한 컬럼을 지운다면 파일시스템에서는 오류가 뜨는데 DBMS는 그 컬럼이 없다고 뜨는 느낌

데이터 관리 관점 - 파일시스템은 보통이다. 휴지통에서 부르고 정도까지 가능하지만, DBMS는 복구, 보완, 동시성 제어 확실히 좋음

파일시스템의 장점은 소프트웨어 설치가 필요 없다. DBMS는 소프트웨어 설치해야 한다. 따로 관리도 해줘야 한다. 그러나 DBMS 장점은 무결성 유지 표준(숫자, 문자)을 만들어 이렇게 만들어야 한다는 것에는 용이하다.

DBMS는 데이터 관리를 따로 하기 때문에 프로그래밍속도는 느리다. 이건 파일시스템이 좋다. 근데 그럴 경우 무결성, 중복성이 있어 정확성은 떨어질 수 있다. 빠른 걸 원하면 파일시스템, 정확성을 원하면 DBMS, 현재 대부분 DBMS이 비중 높게 사용

데이터베이스 사용자에는 일반사용자(특별한 지식이 없어도 사용 가능), 응용프로그래머(프로그램을 만드는 사람들, sql을 사용하여 응용로직을 개발), sql 사용자(데이터를 모니터링하는 것), DBA(전 과정 담당, 운영)

**3주차. 관계데이터모델, 무결성 제약조건과 관계대수**

계층 데이터 모델- 계층적으로 관계가 잇는 모델

네트워크 데이터 모델- 단지 연결만 되어있는 데이터

관계 데이터 모델- 네트워크와 비슷하지만 좀 더 자세히 설명(이 부분은 이 테이블에서 참고해)

객체 데이터 모델- 관계 데이터 모델에서 좀 더 디테일한(이 부분은 테이블, 컬럼 등 설명 가능)

객체 관계 데이터 모델- 위에 두개를 합친 것

**데이터 모델의 구분**

포인터 사용 - 직접 찾아갈 수 있어 속도는 빠른데 개발하는데 오래 걸림

**속성 값 사용(관계 데이터 모델) - 1테이블의 속성이 2테이블의 키 값(primary key)인 것**

객체식별자 사용(객체데이터모델)- 객체가 테이블, record, column 다 가능하다. 상속 캡슐화 등의 개념을 사용할 수 있다. 장점은 관계데이터 모델보다 테이블 간의 관계 같은 것을 캡슐화를 통해 만들 수 있다. 단점은 개발하기 조금 더 어렵다. 그래서 잘 안쓴다.

**3단계 데이터베이스 구조**

스키마는 테이블이 어떻게 구성되어 있고 어떤 관계를 맺고 있다라는 것을 설명한 것이다. 데이터베이스는 먼저 어떤 테이블이 있고 어떤 관계가 있고 이런 것을 스키마로 설계한다. 이것이 개념스키마이다. 그리고 이 데이터를 어디에 저장할 것인지, 열들이 어떤 것만 되는지(문자, 숫자)를 정해주는 것이 내부 스키마이다. 이렇게 되면 전체 DBMS 가 구성이 된 것이다.

**외부스키마**(사용자가 보는 데이터)-**개념스키마**(전체데이터)-**내부스키마**(DBMS가 보는 데이터)

어디에 어떤 형식으로 저장이 되어야 한다는 것(개념/내부 매핑)

어떤 것과 관련 있다(외부/개념 매핑)

외부스키마- 일반사용자, SQL 사용자가 사용. 여러 개의 외부스키마가 있을 수 있다.

뷰(view)의 개념- 보는 용 이지만 내용을 고치면 개념스키마도 고쳐진다(매핑)

개념스키마 - 전체 데이터베이스를 의미, 어떤 열을 가지고 있고 등을 알 수 있다.

내부 스키마 - 실제로 저장되는 방법의 표현, 인덱스 데이터레코드의 배치방법 데이터압축 사항

우리는 주로 문자 숫자 어떤 것으로 지정되고 제약하는 정도만 한다.

DBMS는 매핑을 통해 스키마 간의 대응관계를 정의 - 외부/개념: 1대N대응, 개념/내부: 1대1 대응

3단계 데이터베이스 구조의 특징은 **데이터 독립성** 이다.

논리적 데이터 독립성 - 외부단계와 개념단계 사이의 독립성

개념스키마가 변경되어도 외부스키마에 영향을 미치지 않도록 지원

개념스키마가 하는 일은 외부스키마가 몰라도 된다.

물리적 데이터 독립성 - 개념단계와 내부단계 사이의 독립성

내부스키마가 하는 일은 외부스키마가 몰라도 된다.

논리적 독립성에 비해 훨씬 구현이 쉽다.

**데이터베이스의 개념적 구조 최종 정리** - 데이터들이 어떻게 저장되어야 한다는 데이터 모델이 있고 이에 의해 스키마를 설계하는데 개념스키마을 먼저 설계하고 물리적으로 어떻게 설계하는지 내부스키마를 만들면 DBMS가 완성이 되고, 각 사용자들은 논리적 데이터 독립성 때문에 외부스키마만 보게 되고 물리적 데이터 독립성 때문에 어떻게 저장되는지 모르게 설계한다.

관계데이터모델이 적용된 SQL언어는 비절차적인 언어로 원하는 데이터를 쉽게 표현하는 장점(지금까지 사용하는 이유)

**관계형데이터모델의 단어들**

**릴레이션**- 행과 열로 구성된 **테이블** 릴레이션의 시각적 표현(표) -> 테이블

릴레이션 간에 생성되는 관계: 다른 데이터모델과 구분하는 특징(**키** 값으로 관계를 맺음)

**스키마**- 어떻게 구성되어 있는지 어떤 정보를 담고 있는지에 대한 기본적 구조(첫 행인 헤더)

**속성**- 속성들이 모여 스키마가 된다.

논리모델은 속성을 신경 쓴 것

**도메인** - 속성이 가질 수 있는 값의 집합(int, char)

물리모델은 속성과 도메인을 신경 쓴 것

**차수**- 속성의 개수 ex) 차수가 3차입니다

**인스턴스**- 헤더에 따라 정의된 실제 데이터들

**튜플**- 튜플이 가지는 속성의 개수는 릴레이션 스키마의 차수와 동일– 행

**카디날리티**- 릴레이션에 저장된 튜플의 수 – 행의 수

릴레이션의 특징

속성은 단일값을 가진다.

**속성은 서로 다른 이름을 가진다.**

한 속성의 값은 모두 같은 도메인 값을 가진다.

속성의 순서는 상관없다.

결측값도 하나의 차수로 본다. 결측값이 있다고 차수가 다른 것이 아니다.

**릴레이션 내의 중복된 튜플은 허용하지 않는다.**

튜플의 순서는 상관없다.

관계 데이터 모델의 구성- 데이터를 2차원 테이블 형태의 릴레이션, 릴레이션에 대한 제약조건(도메인, 사용자조건), 관계 연산을 위한 관계대수

**4주차. SQL 문법 기초**

무결성(integrity)- 데이터베이스에 저장된 데이터는 결함이 없어야 한다. 질의에 대해서 신뢰성 있는 답을 제공해야 한다. 일관성 유지 및 중복 제거 등 신뢰도 유지를 위한 제약조건이 필요

키(key)- 무언가를 유일하게 식별하는 의미,

릴레이션에서 특정 튜플을 식별할 때 사용하는 속성 혹은 속성의 집합

릴레이션 간의 관계를 맺는 데도 사용

슈퍼키- 튜플을 유일하게 식별할 수 있는 하나의 속성 혹은 속성의 집합

Ex) 고객번호, 주민번호, (이름, 전화번호) 등

후보키- 슈퍼키에서 딱 필요한 친구(우리가 주로 얘기하는 키)

Ex)고객번호

복합키- 후보키가 두개 이상의 속성인 것, 웬만하면 안쓴다.

기본키(Primary Key)- 여러 후보키 중 하나를 선정하여 대표로 삼는 키 딱 하나!

기본키 제약조건- 릴레이션 내 튜플을 식별할 수 있는 고유값이어야한다.

null값 없어야 한다( 키가 null이면 안된다)

키 값의 변동이 일어나면 안 된다(만약 키 값을 바꾸려면 튜플 자체를 없애고 다시 생성해야 한다)

최대한 적은 수의 속성을 가진 것이어야 한다(복합키 사용 지향한다)

향후 키를 사용하는 데 있어서 문제 발생 소지가 없어야 한다(주민번호 개인정보 때문에 문제 생길 수 있어 기본키로 사용하는 것을 지향한다)

Ex) 고객(고객번호, 이름, 주민번호, 주소, 핸드폰)

대리키(=인조키)- 기본키가 보안을 요하거나, 여러 개의 속성으로 구성되어 복잡할 때 사용

새로운 열을 만들어 번호를 지정하는 것이다.

대체키- 기본키로 선정되지 않은 후보키

외래키(Foreign key)- 다른 릴레이션의 기본키를 참조하는 속성(관계형 데이터에 핵심)

Ex) 주문 릴레이션에 외래키로 고객번호와 도서번호가 있다.

특징- 참조하고 참조되는 양쪽 릴레이션의 도메인이 서로 같아야함

참조되는(PK) 값이 변경되면 참조하는(FK) 값도 변경- 기본키 삭제 떠올리기

자기 자신의 기본키를 참조하는 외래키도 가능 ex) emp 데이터

무결성 제약 조건- 데이터베이스에 저장된 데이터의 일관성과 정확성을 지키는 것

1. 도메인 무결성 제약조건

-릴레이션 내의 튜플들이 각 속성의 도메인에 지정된 값 만을 가져야 한다.

1. 개체 무결성 제약조건

-기본키 제약(중복, 일관성)

3. 참조 무결성 제약 조건

-외래키 제약, 참조되는 릴레이션(도서, 고객)(부모), 참조하는 릴레이션(주문)(자식)

--------------------------------------------------------------------------------------------------------------

**7강**

select name from customer where address like '%대한민국%'

union

select name from customer where custid in (select custid from orders);

select name from customer c, orders o where c.custid=o.custid;

select name from customer where adress like '%대한민국%'

and custid not in (select custid from orders);

select name from customer where adress like '%대한민국%'

and custid in (select custid from orders)

select name, adress from customer c where exists

(select \* from orders o where o.custid=c.custid)

DDL - CREATE : 생성, ALTER : 구조 변경, DROP : 삭제

데이터 타입 종류

INTEGER, INT : 4바이트 정수형

NUMERIC(m,d) : 전체자리수 m, 소수점이하 자리수 d를 가진 숫자형

CHAR(n) : 문자형 고정길이, 문자를 저장하고 남은 공간은 공백으로 채운다.

VARCHAR(n) : 문자형 가변길이

DATE : 날짜형, 연도, 월, 날, 시간을 저장한다

속성 제약 조건

NOT NULL : NULL을 허용하지 않음

UNIQUE : 유일한 값에 대한 제약

DEFAULT : 기본값 설정

CHECK : 값에 대한 조건을 부여할 때 사용

ON DELETE : 레코드 삭제시 외래키 속성에 대한 동작

RESTRICT(NO ACTION)이 기본 : 레코드 삭제 거부

CASCADE : 자식 테이블의 관련 레코드를 같이 삭제

SET NULL : 자식 테이블의 관련 레코드를 NULL값으로 설정

TABLE 생성 예제

CREATE TABLE NewBook(

bookid INTEGER,

bookname VARCHAR(20),

publisher VARCHAR(20),

price INTEGER);

CHAR(n) : n바이트를 가지는 문자형 타입

저장되는 문자의 길이가 n보다 작으면 나머지는 공백으로 채워 저장

VARCHAR(n) : 저장되는 문자의 길이만큼 기억장소를 차지하는 가변형

CHAR(n)은 공백을 채운 문자열이기 때문에 VARCHAR(n)과 동등 비교시 실패 가능

- 기본키 지정

CREATE TABLE NewBook(

bookid INTEGER,

bookname VARCHAR(20),

publisher VARCHAR(20),

price INTEGER,

PRIMARY KEY (bookid));

CREATE TABLE NewBook(

bookid INTEGER PRIMARY KEY,

bookname VARCHAR(20),

publisher VARCHAR(20),

price INTEGER);

- 기본키 중 복합키 지정

CREATE TABLE NewBook(

bookname VARCHAR(20),

publisher VARCHAR(20),

price INTEGER,

PRIMARY KEY (bookname, publisher));

EX)

CREATE TABLE NewCustomer(

custid INTEGER PRIMARY KEY,

name VARCHAR(40),

address VARCHAR(40),

phone VARCHAR(30));

제약조건 추가 예제

CREATE TABLE NewBook(

bookname VARCHAR(20) NOT NULL,

publisher VARCHAR(20) UNIQUE,

price INTEGER DEFAULT 10000 CHECK(price >1000),

PRIMARY KEY (bookname, publisher));

외래키 생성

외래키를 명시할 때 주의사항

반드시 참조되는 테이블이 존재해야하며 참조되는 테이블의 기본키를 외래키로 지정

ON DELETE옵션으로 참조되는 테이블의 레코드가 삭제될 때 취할 수 있는 동작 지정

EX)

CREATE TABLE NewOrders(

orderid INTEGER,

custid INTEGER NOT NULL,

bookid INTEGER NOT NULL,

saleprice INTEGER,

orderdate DATE,

PRIMARY KEY (orderid),

FOREIGN KEY(custid) REFERENCES NewCustomer(custid) ON DELETE CASCADE);

ALTER : 테이블의 속성과 속성에 관한 제약 변경(기본키 외래키 변경도 포함)

ADD : 제약사항 추가

DROP : 제약사항 삭제

MODIFY : 속성 변경

ALTER TABLE NewBook ADD isbn VARCHAR(13);

ALTER TABLE NewBook MODIFY isbn INTEGER;

ALTER TABLE NewBook DROP COLUMN isbn;

ALTER TABLE NewBook MODIFY bookid INTEGER NOT NULL;

ALTER TABLE NewBook ADD PRIMARY KEY (bookid);

DROP TABLE NewBook;

INSERT INTO Book(bookid, bookname, publisher, price)

VALUES(11,'스포츠의학','한솔의학서적',90000);

속성 이름은 생략 가능- 데이터 입력순서는 속성의 순서와 일치해야함

속성 이름을 사용할 경우에는 순서가 중요하지 않음

INSERT INTO Book(bookid,bookname,publisher)

VALUES(12,'스포츠의학','한솔의학서적);

INSERT INTO Book(bookid,bookname,price,publisher)

SELECT bookid,bookname,price,publisher

FROM Imported\_book;

UPDATE Customer

SET address='대한민국 부산'

WHEREc custid=5;

UPDATE Book

SET publisher=(SELECT publisher FROM imported\_book WHERE bookid=21)

WHERE bookid=14;

UPDATE 문 사용시 주의사항

UPDATE문에서 여러 속성값을 한번에 수정하는 작업은 가능하지만 위험하다

잘못 수정된 데이터를 원래대로 돌리는 방법이 있기는 하지만 경우에 따라 문제가 발생될 수 있어 주의

DELETE Book

WHERE bookid=11;

DELETE FROM Customer;

**8강**

함수의 정의 : 일정한 계산이나 처리를 수행하는 명령, 수학의 함수와 유사

내장함수와 사용자정의함수가 있다.

내장함수 : SELECT절, WHERE절, UPDATE절 등에서 모두 사용 가능

단일행 함수- 숫자, 문자, 날짜/시간, 변환, 정보, NULL관련 함수, 집계함수, 분석함수(윈도우함수)

숫자함수 – ABS(숫자) : 숫자의 절댓값을 계산

CEIL(숫자) : 숫자보다 크거나 같은 최소의 정수 – 올림

FLOOR(숫자) : 숫자보다 작거나 같은 최소의 정수 – 내림

ROUND(숫자, m) : 숫자의 반올림, m은 반올림 기준 자릿수

LOG(n,숫자) : 숫자의 자연로그 값을 반환

POWER(숫자,n) : 숫자의 n제곱 값을 계산

SQRT(숫자) : 숫자의 제곱근 값을 게산

SIGN(숫자) : 숫자가 음수면 -1, 0이면 0, 양수면 1

문자함수 – CHAR나 VARCHAR의 데이터 타입을 대상으로 단일문자나 문자열을 가공한 결과 반환

1. 대표적인 숫자값 반환함수

ASCII(c) : 대상 알파벳 문자의 아스키 코드 값을 반환

LENGTH(s) : 대상 문자열의 Byte 반환, 알파벳 1byte, 한글 3byte

CHAR\_LENGTH(s) : 문자열의 문자 수를 반환

한 문자가 차지하는 바이트의 기준은 사용하는 데이터베이스의 환경변수 설정에 따라 상이

1. 대표적인 문자값 반환함수

-CONCAT(s1,s2) : 두 문자열을 연결 ex) CONCAT('마당', ' 서점') => '마당 서점'

-LOWER(s) : 문자열을 소문자로 변환 ex) LOWER('MR. SCOTT') => 'mr. scott'

-UPPER(s) : 대상 문자열을 모두 대문자로 변환 ex) UPPER('mr. scott') => 'MR. SCOTT'

-LPAD(s,n,c) : 문자열의 왼쪽부터 지정한 자리수까지 지정한 문자로 채움

ex) LPAD('Page 1', 10, '\*') => '\*\*\*\*Page 1'

-RPAD(s,n,c) : 대상 문자열의 오른쪽부터 지정한 자리수까지 지정한 문자로 채움

ex) RPAD('AbC', 5, '\*') => 'AbC\*\*'

-REPLACE(s1,s2,s3) : 문자열의 지정한 문자를 원하는 문자로 변경

ex) REPLACE('JACK & JUE', 'J', 'BL') => 'BLACK & BLUE'

-SUBSTR(s,n,k) : 대상 문자열의 지정된 자리에서부터 지정된 길이만큼 잘라서 반환

ex) SUBSTR('ABCDEFG', 3, 4) => 'CDEF'

-TRIM(c FROM s) : 양쪽에서 지정된 문자를 삭제(문자열만 넣으면 기본값으로 공백 제거)

ex) TRIM('=' FROM '==BROWNING==') => 'BROWNING'

날짜/시간 함수 - 날짜/시간은 문자열로 저장/관리보다 날짜형 데이터로 저장/관리가 필요

날짜만의 연산을 손쉽게 처리할 수 있기 때문

1. 날짜/시간 표현을 위한 Format의 주요 지정자

%w(요일 번호), %W(요일), %a(요일의 약자), %d(1달 중 날짜)

%j(1년 중 날짜), %h(12시간), %H(24시간), %i(분), %m(월 번호)

%b(월 이름 약어), %M(월 이름), %s(초), %Y(4자리 연도)

%y(4자리 연도의 마지막 2자리)

2. 주요 날짜/시간 함수

-STR\_TO\_DATE(string, format)) : 문자열(STRING) 데이터를 날짜형(DATE)으로 반환

ex) STR\_TO\_DATE('2019-02-14', '%Y-%m-%d') => 2019-02-14

-DATE\_FORMAT(data, format) : 날짜형(DATE) 데이터를 문자열(VARCHAR)로 변환

ex) DATE\_FORMAT('2019-02-14', '%Y-%m-%d') => '2019-02-14'

-ADDDATE(date, interval) : DATE 형의 날짜에서 INTERVAL 지정한 시간만큼 더함

ex) ADDDATE('2019-02-14', INTERVAL 10 DAY) => 2019-02-24

-DATE(date) : DATE 형의 날짜 부분을 반환

ex) SELECT DATE('2003-12-31 01:02:03'); => 2003-12-31

-DATEDIFF(date1, date2) : DATE 형의 date1 – date2 날짜 차이를 반환

ex) SELECT DATEDIFF('2019-02-14', '2019-02-04') => 10

-SYSDATE : DBMS 시스템상의 오늘 날짜를 반환하는 함수

ex) SYSDATE() => 2018-06-30 21:47:01

NULL 값 처리

NULL값은 비교 연산자로 비교가 불가능함

NULL값의 연산을 수행하면 결과 역시 NULL값으로 반환됨

집계함수를 사용할 때 주의할 점

NULL + 숫자 연산의 결과는 NULL

집계함수 계산 시 NULL이 포함된 행은 집계에서 빠짐

해당되는 행이 하나도 없을 경우 SUM, AVG 함수의 결과는 NULL이 되며, COUNT 함수의 결과는 0이다.

NULL 관련 주요 함수

IS NULL, IS NOT NULL : NULL값 확인 함수

IFNULL : NULL값을 다른 값으로 대체하여 연산하거나 다른 값으로 출력

행번호 출력 : SQL문 결과로 나오는 행에 번호를 붙이거나 행번호에 따라 결과의 개수를 조절

변수를 사용하여 처리

MYSQL에는 변수의 이름 앞에 @기호 사용, 치환문에는 :=기호 사용

--------------------------------------------------------------------------------------------------------------

**9강**

1. 부속질의 정의 : 하나의 SQL문 안에 다른 SQL문이 중첩된 질의

다른 테이블에서 가져온 데이터로 현재 테이블에 있는 정보를 찾거나 가공할 때 사용

보통 데이터가 대량일 때 데이터를 모두 합쳐서 연산하는 조인보다 필요한 데이터만 찾아서 공급해주는 부속질의가 성능이 더 좋음

주질의와 부속질의로 구성됨

2. 부속질의의 분류

위치와 역할에 따른 구분

|  |  |  |  |
| --- | --- | --- | --- |
| 명칭 | 위치 | 영문 및 동의어 | 설명 |
| 스칼라 부속질의 | SELECT 절 | scalar subquery | SELECT 절에서 사용되며 단일 값을 반환하기 때문에 스칼라 부속질의라고 함 |
| 인라인 뷰 | FROM 절 | inline view, table subquery | FROM 절에서 결과를 뷰(view) 형태로 반환하기 때문에 인라인 뷰라고 함 |
| 중첩질의 | WHERE 절 | nested subquery, predicate subquery | WHERE 절에 술어와 같이 사용되며 결과를 한정시키기 위해 사용됨. 상관 혹은 비상관 형태. |

동작 방식에 따른 구분

상관 부속질의 : 주질의의 특정 열 값을 부속질의가 상속받아 부속질의의 질의에 사용하는 형태

비상관 부속질의(일반 부속질의) : 독립된 질의를 수행해서 결과 값을 가져오는 형태

반환하는 결과의 형태에 따른 구분

단일행 부속질의 : 부속질의의 결과 하나의 행을 반환하여 주질의에 전달, 비교 연산자의 수행이나 스칼라 부속질의 등에서 나타남

다중행 부속질의 : 부속질의의 결과 여러 개의 행을 반환, IN연산자 사용

3. 스칼라 부속질의

SELECT 절에서 사용되는 부속질의

부속질의의 결과값을 단일 행, 단일 열의 스칼라 값으로 반환

원칙적으로 스칼라 값이 들어갈 수 있는 모든 곳에 사용 가능하며, 일반적으로 SELECT 문과 UPDATE SET 절에 사용

주질의와 부속질의와의 관계는 상관/비상관 모두 가능

4. 인라인 뷰

FROM 절에서 사용되는 부속질의

테이블 이름 대신 인라인 뷰 부속질의를 사용하면 보통의 테이블과 같은 형태로 사용 가능

부속질의 결과 반환되는 데이터는 다중 행, 다중 열이어도 상관없음

다만 가상의 테이블인 뷰 형태로 제공되어 상관 부속질의로 사용될 수는 없음

5. 중첩질의

WHERE절에서 사용되는 부속질의

주질의에 사용된 자료 집합의 조건을 WHERE절에 서술

WHERE절은 보통 데이터를 선택하는 조건 혹은 술어와 같이 사용

중첩질의를 술어 부속질의라고도 함

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 술어 | 연산자 | 반환 행 | 반환 열 | 상관 |
| 비교 | =, ＞, ＜, ＞=, ＜=, ＜＞ | 단일 | 단일 | 가능 |
| 집합 | IN, NOT IN | 다중 | 단일 | 가능 |
| 한정 | ALL, SOME(ANY) | 다중 | 단일 | 가능 |
| 존재 | EXISTS, NOT EXISTS | 다중 | 다중 | 필수 |

비교 연산자의 사용 – 부속질의가 반드시 단일행, 단일열을 반환

IN, NOT IN – 주질의 속성 값이 부속질의에서 제공한 결과 집합에 있는지 확인하는 역할

부속질의의 결과 다중 행을 가질 수 있음

IN 연산자는 주질의와 WHERE 절에 사용되는 속성 값을 부속질의의 결과 집합과 비교해 하나라도 있으면 참, NOT IN은 반대로 존재하지 않으면 참

ALL, SOME(ANY) – ALL은 모두 SOME(ANY)는 어떠한(최소한 하나라도)라는 의미

ALL은 결과 집합의 최대값, SOME은 결과 집합의 최소값

EXISTS, NOT EXISTS – 데이터의 존재 유무를 확인하는 연산자

주질의에서 부속질의로 제공된 속성의 값을 가지고 부속질의에 조건을 만족하여 값이 존재하면 참이 되고, 주질의는 해당 행의 데이터를 출력

**반드시 부속질의에 주질의의 열이름이 필요**

6. 뷰의 정의 - 하나 이상의 테이블을 합하여 만든 가상의 테이블

조인이나 부속질의를 매번 하지 않도록 가상의 테이블 생성

합한다 = SELECT문을 통해 얻은 최종 결과

뷰는 이러한 결과를 가상의 테이블로 정의하여 실제 테이블처럼 사용할 수 있도록 만든 데이터베이스 개체

- 뷰의 장점

편리성 및 재사용성 – 자주 사용되는 복잡한 질의를 뷰로 미리 정의해 놓을 수 있음

복잡한 질의를 간단히 작성

보안성 – 사용자별로 필요한 데이터만 선별하여 보여줄 수 있고, 중요한 질의의 경우 질의 내용을 암호화

개인정보(주민번호)나 급여, 건강 같은 민감한 정보를 제외한 테이블을 만들어 사용

독립성 – 미리 정의된 뷰를 일반 테이블처럼 사용할 수 있기 때문에 편리

사용자가 필요한 정보만 요구에 맞게 가공하여 뷰로 만들어 쓸 수 있음

원본 테이블의 구조가 변해도 응용에 영향을 주지 않도록 하는 논리적 독립성 제공

- 뷰의 특징

원본 데이터 값에 따라 같이 변함

독립적인 인덱스 생성이 어려움

삽입, 삭제 갱신 연산에 많은 제약이 따름

SELECT문을 제외한 갱신작업을 수행하는데 제약

기본키를 포함하지 않는 수정 요청이나 베이스 테이블 두 개 이상에서 속성을 포함하는 수정 요청은 제약을 위반할 가능성이 있기 때문에 금지

- 뷰의 사용 : SELECT문을 활용하여 먼저 테스트 후 뷰로 정의

Create view, create or replace view, drop view

시스템 뷰

시스템 뷰 = 데이터 딕셔너리 뷰 = 시스템 카탈로그

DBMS관련 정보를 테이블 형태로 만들어 제공

MySQL을 포함한 모든 DBMS에서는 데이터베이스 개체나 시스템 통계 정보등을 사용자가 직접 확인할 수 있도록 제공

MySQL시스템 뷰는 INFORMATION\_SCHEMA 데이터베이스에 저장

--------------------------------------------------------------------------------------------------------------**10 강**

1. 데이터베이스의 물리적 저장

DBMS와 데이터파일

PDF 그림 참고

실제 데이터가 저장되는 곳은 보조기억장치

가장 많이 사용되는 장치는 하드디스크

하드디스크에 저장된 데이터를 읽어 오는데 걸리는 시간은 모터에 의해 분당 회전하는 속도, 데이터를 읽을 때 액세스 암이 이동하는 시간, 주기억장치로 읽어오는 시간에 영향을 받음

액세스 시간 = 탐색시간(액세스 헤드를 트랙에 이동시키는 시간) + 회전지연시간(섹터가 액세스 헤드에 접근하는 시간) + 데이터 전송시간(데이터를 주기억장치로 읽어오는 시간)

속도 문제 줄이기 위해 주기억장치에 DBMS가 사용하는 공간 중 일부를 버퍼 풀로 만들어 사용

버퍼에 자주 사용하는 데이터를 저장해두며 LRU 알고리즘을 이용하여 사용빈도가 높은 데이터 위주로 저장하고 관리

MySQL의 저장장치 엔진은 플러그인 방식으로 선택

InnoDB엔진이 기본

텍스트이(가) 표시된 사진

자동 생성된 설명 사용하는 데이터베이스별로 별도의 파일을 생성

2. 인덱스(index, 색인)

자료를 쉽고 빠르게 찾을 수 있도록 만든 데이터 구조

일반적인 RDBMS의 인덱스는 대부분 B-tree(Balanced-tree)구조

루트 노드, 내부 노드, 리프 노드로 구성

리프 노드가 모두 같은 레벨에 존재하는 균형 트리

모든 검색을 루트 노드에서부터 시작하여 내부 노드를 지나 리프 노드까지 내려가면서 이루어짐

인덱스의 특징

인덱스는 테이블에서 한 개 이상의 속성을 이용하여 생성함

빠른 검색과 함께 효율적인 레코드 접근이 가능함

순서대로 정렬된 속성과 데이터의 위치만 보유하므로 테이블보다 작은 공간을 차지함

저장된 값들은 테이블의 부분집합이 됨

일반적으로 B-tree 형태의 구조를 가짐

데이터의 수정, 삭제 등의 변경이 발생하면 인덱스의 재구성이 필요함

3. MySQL 인덱스

클러스터 인덱스

클러스터 인덱스는 테이블 당 하나만 생성

인덱스 페이지가 단순해져 인덱스 저장 시 차지하는 공간도 작음

기본적인 인덱스로 테이블 생성 시 기본키를 지정하면 기본키에 대하여 클러스터 인덱스를 생성

기본키를 지정하지 않으면 먼저 나오는 UNIQUE 속성에 대하여 클러스터 인덱스를 생성

기본키나 UNIQUE속성이 없는 테이블은 MySQL이 자체 생성한 행본호를 이용하여 클러스터 인덱스를 생성

보조 인덱스

보조 인덱스는 테이블 당 여러 개 생성 가능

클러스터 인덱스가 아닌 모든 인덱스는 보조 인덱스이며 보조 인덱스의 각 레코드는 보조 인덱스 속성과 기본키 속성 값을 갖고 있음

보조 인덱스를 검색하여 기본키 속성 값을 찾은 다음 클러스터 인덱스로 가서 해당 레코드를 탐색

4. 인덱스의 생성

인덱스는 검색을 빨리하기 위해 사용

데이터의 양이 없거나 데이터 값이 몇 종류 안되어 선택도가 높은 경우 인덱스가 없는 것이 더 빠름

인덱스 생성 시 고려사항

인덱스는 WHERE 절에 자주 사용되는 속성이어야 함

인덱스는 조인에 자주 사용되는 속성이어야 함

단일 테이블에 인덱스가 많으면 속도가 느려질 수 있음(테이블 당 4~5개 정도 권장)

속성이 가공되는 경우 사용하지 않음

속성의 선택도가 낮을 때 유리함(속성의 모든 값이 다른 경우)

Create index \_\_\_\_\_ on \_\_\_\_(\_\_\_\_);

ANALYZE TABLE \_\_\_\_;(최적화 하는 것)

DROP INDEX \_\_\_\_\_\_ ON \_\_\_\_\_\_\_\_;

5. 데이터 모델링

현실 세계에 존재하는 데이터를 컴퓨터 세계의 데이터베이스로 옮기는 변환 과정

데이터베이스 설계의 핵심 과정

6. 2단계 데이터 모델링

개념적 데이터 모델링 – 현실 세계의 중요 데이터를 추출하여 개념 세계로 옮기는 작업

논리적 데이터 모델링 – 개념 세계의 데이터를 데이터베이스에 저장하는 구조로 표현하는 작업

7. 데이터 모델

데이터 모델링의 결과물을 표현하는 도구

데이터 모델은 연산, 데이터 구조, 제약조건으로 구성되어 있다.

8. 데이터베이스 생명주기 - 데이터베이스의 생성과 운영에 관련된 특징

요구사항 수집 및 분석

사용자들의 요구사항을 듣고 분석하여 데이터베이스 구축의 범위를 정함

설계

분석된 요구사항을 기초로 주요 개념과 업무 프로세스 등을 식별(개념적 설계)

사용하는 DBMS의 종류에 맞게 변환(논리적 설계)

데이터베이스 스키마를 도출(물리적 설계)

구현

설계 단계에서 생성한 스키마를 실제 DBMS에 적용하여 테이블 및 관련 객체(뷰, 인덱스 등)을 만듦

운영

구현된 데이터베이스를 기반으로 소프트웨어를 구축하여 서비스를 제공함

감시 및 개선

데이터베이스 운영에 따른 시스템의 문제를 관찰하고 데이터베이스 자체의 문제점을 파악하여 개선함

9. 데이터 모델링 과정

1. 요구사항 수집 및 분석 – 현실세계의 대상 및 사용자의 요구 등을 정리 및 분석

사용자 식별, 데이터 베이스 용도 식별, 사용자 요구 사항 수집 및 명세

2. 설계

2-1) 개념적 모델링 – 중요 개념을 구분

핵심 Entity(독립개체) 도출, ERD작성

2-2) 논리적 모델링 – 각 개념을 구체화

ERD-RDB모델 사상, 상세 속성의 정의, 정규화 등

2-3) 물리적 모델링 – 데이터베이스 생성 계획에 따라 개체, 인덱스 등을 생성

DB 개체 정의, 테이블 및 인덱스 등 설계

3. 데이터베이스 구현

10. 개념적 모델링

요구사항을 수집하고 분석한 결과를 토대로 업무의 핵심적인 개념을 구분

전체적인 뼈대를 만드는 과정

개체(entity)를 추출하고 각 개체들 간의 관계를 정의

ER다이어그램을 만드는 과정까지 포함

11. 논리적 모델링

개념적 모델링에서 만든 ER다이어그램을 사용하려는 DBMS에 맞게 사상

실제 데이터베이스로 구현하기 위한 모델을 만드는 과정

개념적 모델링에서 추출하지 않았던 상세 속성들을 모두 추출

정규화 수행

데이터 표준화 수행

12. 물리적 모델링

작성된 논리적 모델을 실제 컴퓨터의 저장 장치에 저장하기 위한 물리적 구조를 정의하고 구현하는 과정

DBMS 특성에 맞게 저장 구조를 정의해야 데이터베이스가 최적의 성능을 낼 수 있음

--------------------------------------------------------------------------------------------------------------**11강**

1. ER 모델

피터 첸이 제안한 개념적 모델

개체(Entity) : 독립적인 의미를 지니고 있는 유무형의 사람 또는 사물

개체 간의 관계를 이용해 현실 세계를 개념적 구조로 표현

핵심요소 : 개체, 속성, 관계

2. ER 다이어그램

개체-관계모델을 이용해 현실세계를 개념적으로 모델링한 결과물을 표준화된 그림으로 표현한 것

3. 개체 : 저장할 가치가 있는 중요 데이터를 가지고 있는 사람이나 사물, 개념, 사건 등

사람, 사물, 장소, 개념, 사건과 같이 유무형의 정보를 가지고 있는 독립적인 실체

데이터베이스에서 주로 다루는 개체는 낱개로 구성된 것, 낱개가 각각 데이터 값을 가지는 것, 데이터 값이 변하는 것 등

다른 개체와 구별되는 이름을 가지고 있고, 각 개체만의 고유한 특성이나 상태, 즉 속성을 하나 이상 가지고 있음

비슷한 속성의 개체 타입을 구성

개체를 고유의 이름과 속성들로 정의한 것으로 파일 구조의 레코드 타입에 대응

개체 인스턴스

개체를 구성하고 있는 속성이 실제 값을 가짐으로써 실제화된 개체

레코드 인스턴스에 대응

개체 집합

특정 개체 타입에 대한 개체 인스턴스들을 모아 놓은 것

4. 개체 타입의 유형

강한 개체 – 다른 개체의 도움 없이 독자적으로 존재할 수 있는 개체

보통 개체 타입은 강한 개체 타입을 의미

약한 개체 – 독자적으로는 존재할 수 없고 반드시 상위 개체 타입을 가짐

개체 타입의 ER다이어그램 표현

ER다이어그램에서 개체 타입은 직사각형으로 표현하고 사각형 안에 이름을 표기

텍스트이(가) 표시된 사진

자동 생성된 설명

5. 속성

개체나 관계가 가지고 있는 고유의 특성

의미 있는 데이터의 가장 작은 논리적 단위

파일 구조의 필드와 대응됨

E-R 다이어그램에서는 타원으로 표현하고 타원 안에 이름을 표기

속성이 개체를 유일하게 식별할 수 있는 키일 경우 속성 이름에 밑줄 표기

6. 속성의 유형

단일 값 속성 – 값을 하나만 가질 수 있는 속성 ex) 고객 개체의 이름, 적립금 속성

다중 값 속성 – 값을 여러 개 가질 수 있는 속성 ex) 고객 개체의 연락처 속성

단순 속성 – 의미를 더는 분해할 수 없는 속성 ex)고객 개체의 적립금 속성

복합 속성 – 의미를 분해할 수 있는 속성 ex) 고객 개체의 주소 속성

유도 속성 – 기존의 다른 속성의 값에서 유도되어 결정되는 속성

값이 별도로 저장되지 않음

ex) 책 개체의 가격과 할인율 속성으로 계산되는 판매가격 속성

ex) 고객 개체의 출생연도 속성으로 계산되는 나이 속성

키 속성 – 각 개체 인스턴스를 식별하는데 사용되는 속성

모든 개체 인스턴스의 키 속성 값이 다름

둘 이상의 속성들로 구성되기도 함

ex) 고객 개체의 고객아이디 속성

속성의 ER 다이어그램 표현

테이블이(가) 표시된 사진

자동 생성된 설명

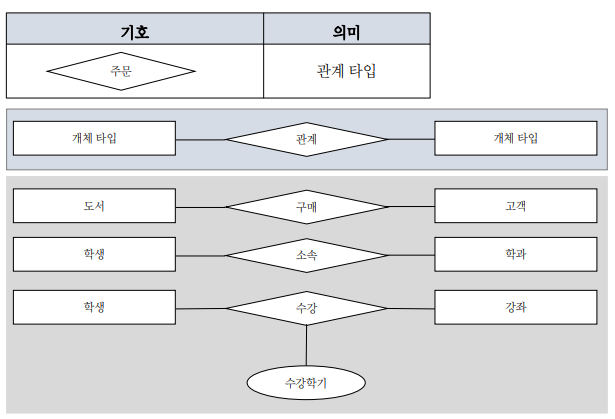
7. 관계 : 개체 사이의 연관성을 나타내는 개념

개체 집합들 사이의 대응관계, 매핑을 의미 ex)고객 개체와 책 개체 간의 구매 관계

관계 타입 : 개체 타입과 개체 타입 간의 연결 가능한 관계를 정의한 것

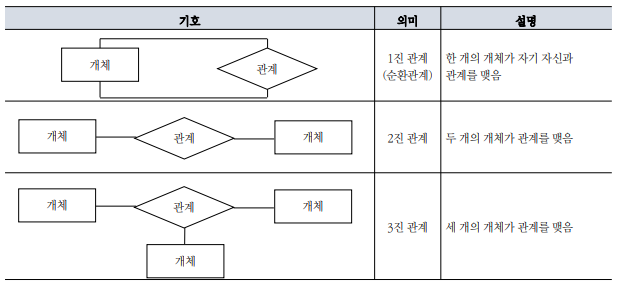
관계 집합 : 관계로 연결된 집합

관계 타입의 ER 다이어그램 표현



8. 관계 타입의 유형

차수에 따른 유형\_관계 타입의 차수 : 관계 집합에 참가하는 개체 타입의 수



관계 대응 수에 따른 유형\_관계 대응수 : 두 개체 타입의 관계에 실제로 참여하는 개별 개체 수

테이블이(가) 표시된 사진

자동 생성된 설명

관계 대응 수에 따른 유형 종류

1. 일대일(1:1) 관계 : 좌측 개체 타입에 포함된 개체가 우측 개체 타입에 포함된 개체와 일대일로 대응하는 관계 ex) 결혼, 사원별 컴퓨터
2. 일대다(1:N), 다대일(N:1) 관계 : 실제 일상생활에서 가장 많이 볼 수 있는 관계로, 한 쪽 개체 타입의 개체 하나가 다른 쪽 개체 타입의 여러 개체와 관계를 맺음

ex)학과와 학생(학생입장에서 학과 하나, 학과입장에서 학생 다수)

ex)부모와 자식(부모입장에서 자식 다수, 자식입장에서 부모 하나)

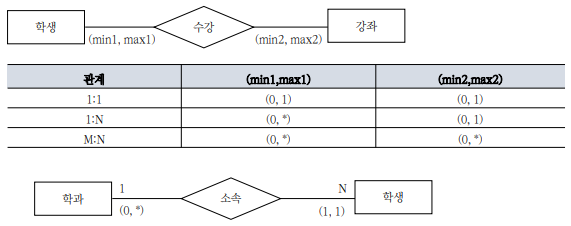
1. 다대다(N:M) 관계 : 각 개체 타입의 개체들이 서로 임의의 개수의 개체들과 서로 복합적인 관계를 맺고 있는 관계

ex) 학생과 강좌(학생입장에서 강좌 여러개, 강좌입장에서 학생 여러명)

관계 대응수의 최소값과 최대값

관계 대응수 1:1, 1:N, M:N에서 1, N, M은 각 개체가 관계에 참여하는 최댓값을 의미

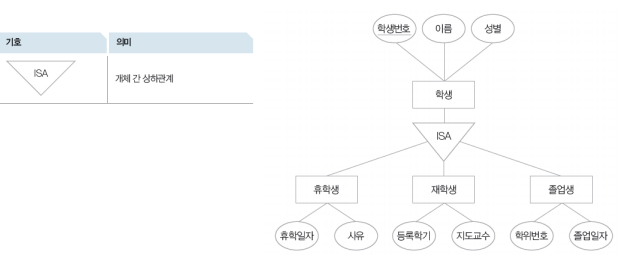
관계에 참여하는 개체의 최솟값을 표시하지 않는다는 단점을 보완하기 위해 다이어그램에서는 대응수 외에 최솟값과 최댓값을 관계실선 위에 표기

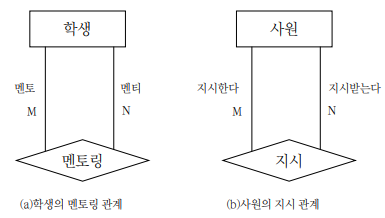


🡪 학과입장에서 학생은 0명에서 \* 까지, 학생입장에서 학과 1개에서 1개까지

• ISA(is-a) 관계

상위 개체 타입(슈퍼클래스)의 특성에 따라 하위 개체 타입(서브클래스)이 결정



• 순환적 관계(recursive relationship)

하나의 개체 타입이 동일한 개체 타입(자기 자신)과 순환적으로 관계를 가지는 형태

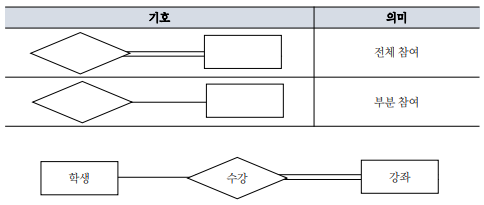
관계선을 개체 타입과 관계 타입 양쪽에 두개를 표기

9. 참여 제약 조건

• 개체 집합 내 모든 개체가 관계에 참여하는지 유무에 따라 구분

전체 참여는 개체 집합의 모든 개체가, 부분 참여는 일부만 참여함

전체 참여를 (최솟값, 최댓값)으로 표현할 경우 최솟값이 1 이상으로 모두 참여한다는 뜻이고, 부분 참여는 최솟값이 0 이상임



역할(role)

• 개체 타입 간의 관계를 표현할 때 각 개체들은 고유한 역할(role) 담당

역할은 관계만으로 알 수 있으면 생략

관계에서 역할이 명확하지 않을 경우에는 반드시 표기



약한 개체 타입과 식별자

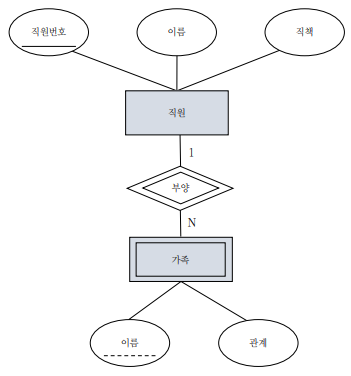
• 약한 개체(weak entity) 타입

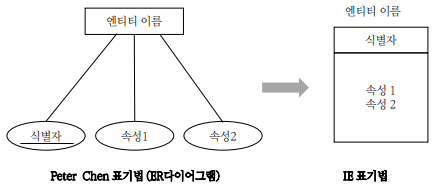
상위 개체 타입이 결정되지 않으면 개별 개체를 식별할 수 없는 종속된 개체 타입

테이블이(가) 표시된 사진

자동 생성된 설명 식별자(discriminator) 또는 부분키(partial key) : 약한 개체 타입은 독립적인 키로는 존재할 수 없지만 상위 개체 타입의 키와 결합하여 약한 개체 타입의 개별 개체를 고유 하게 식별하는 속성

• 약한 개체(weak entity) 타입과 식별자 예제





IE(Information Engineering) 표기법

• ER다이어그램을 더 축약하여 쉽게 표현한 방식

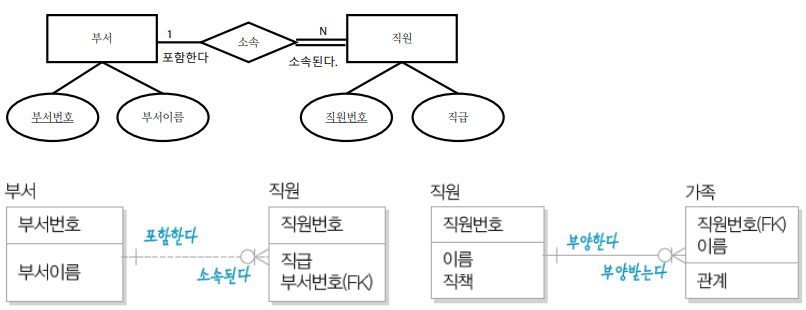
ERWin 등의 소프트웨어에서 사용

새발 표기법(crow-feet)

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명• IE표기법에서 관계는 실선 혹은 점선으로 표기

• IE표기법 예시



10. ER모델을 관계 데이터 모델로

매핑(mapping, 사상)

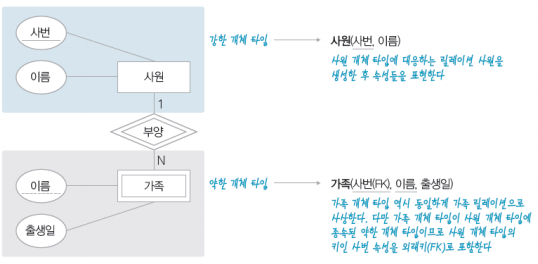
• 완성된 ER 모델을 실제 데이터베이스로 구축하기 위해 논리적 모델링



개체 타입 매핑

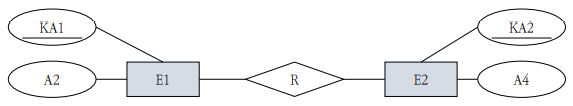
• [1단계] 강한(정규) 개체 타입 : 정규 개체 타입 E의 경우 대응하는 릴레이션 R을 생성

• [2단계] 약한 개체 타입 : 약한 개체 타입에서 생성된 릴레이션은 자신의 키와 함께 강한 개체 타입의 키를 외래키로 사상하여 자신의 기본키를 구성



관계 타입의 매핑

• 관계 타입 매핑 방법



[방법1] 오른쪽 개체 타입 E2를 기준으로 관계 R을 표현 E1(KA1, A2), E2(KA2, A4, KA1)

[방법2] 왼쪽 개체 타입 E1을 기준으로 관계 R을 표현 E1(KA1, A2, KA2), E2(KA2, A4)

[방법3] 단일 릴레이션 ER로 모두 통합하여 관계 R을 표현 ER(KA1, A2, KA2, A4)

[방법4] 개체 타입 E1, E2와 관계 타입 R을 모두 독립된 릴레이션으로 표현

E1(KA1, A2), R(KA1, KA2), E2(KA2, A4)

• [3단계] 이진 1:1 관계 타입

이진 1:1 관계 타입의 경우 [방법1]~[방법4]까지 모든 유형으로 사상 가능, 개체가 가진 정보 유형에 따라 판단

[방법4]의 경우 불필요한 개체를 생성함에 따라 잘 사용하지 않음

[방법1]또는 [방법2] 중 외래키(FK)에 NULL값이 덜 발생하는 방법 사용

예) 사원수가 컴퓨터 대수보다 많다면 [방법1] 반대는 [방법2]

• [4단계] 이진 1:N 관계 타입

이진 1:N 관계 타입의 경우 N의 위치에 따라 [방법1] 또는 [방법2]의 유형으로 매핑

N이 있는 곳에 관계를 표현

• [5단계] 이진 M:N 관계 타입

이진 M:N 관계 타입은 [방법4]의 유형으로 매핑

양쪽 모두 다수의 대응 수를 가지므로 새로운 릴레이션을 반드시 생성

이 때 생성되는 새로운 릴레이션을 교차 테이블이라고 함

• [6단계] N진 관계 타입

ER 모델의 차수가 3 이상인 다진 관계 타입의 경우 [방법4]의 유형으로 매핑

다중 값 속성의 사상

• 여러 값을 가질 수 있는 다중 속성은 직접 매핑이 불가능

[방법1] 다중값 속성의 개수를 알 수 없는 경우 : 새로운 릴레이션 생성

[방법2] 다중값 속성의 개수가 적고 제한 가능한 경우 : 별도의 속성으로 구성

• [7단계] 다중값 속성

[방법1]은 새로 생성된 릴레이션에 값이 계속 저장될 수 있어 효율적이지만 새로운 릴레이션이 생기게 되어 모델이 복잡