데이터베이스

데이터 - 관찰 측정을 통해 수집된 사실 또는 값이 특정 기준에 의해 정리된것

정보 - 어떤 목적에 의해 데이터를 해석하거나 가공한 형태

지식 - 연결된 정보들의 패턴을 바탕으로 경험 학습 이해 추론을 통해 처리된 결과물

데이터베이스 할 수 있는 역할? 데이터들의 집합. 여기서 정보를 얻을 수 있다.

데이터를 바탕으로 정보를 만들어내는 주체는 사람이다.(아직까지는) 앞으로는 ai에게 시키겠다는게 4차혁명

데이터에 대한 이해가 있어야 정보를 추출할 수 있다

데이터베이스란?

조직에 필요한 정보를 얻기 위해 논리적으로 연관된 데이터를 모아 구조적으로 통합해놓은것

가장 쉽게 db구축하기, 정보 추출하기?

유사한 것을 살펴본다.

모든 db는 구조가 다르다

간단한 거래도 많은 데이터 포함

ex)영수증, 승차권

예약에 관한 정보, 등등 db에서 가져와서 종이에 출력하는 것

데이터베이스의 정의

1. 통합된 데이터

데이터를 통합하는 개념. 중복 최소화하여 중복으로 인한 데이터 불일치 현상 제거

2. 저장된 데이터

문서로 보관된 데이터가 아니라 디스크, 테이프 같은 저장장치에 저장된 데이터

3. 운영 데이터

조직의 목적을 위해 사용되는 데이터. 업무를 위한 검색을 할 목적으로 저장된 데이터

4. 공용 데이터

공동으로 사용되는 데이터

db특징 -1. 검색과 변경(삽입 삭제 수정)이 잦다

2. 실시간 접근이 가능 - 실시간으로 데이터 처리 못하면 문제 발생(예매를 위한 좌석수 검색하는 데 몇 분걸리면 곤란)

3. 끊임없이 동적으로 변화 - 삽입 삭제 수정을 통해 바뀐 데이터 저장(주식 시세가 실시간으로 업데이트)

4. 여러 사용자가 동시에 공유 - db에 접근하는 프로그램이 여러개 있다는 의미

5. 내용에 의한 참조 가능 - 데이터의 물리적 위치가 아닌 데이터 값에 따라 참조(예를 들어 학생이름으로 검색)

dbms - 사용자와 db를 연결시켜주는 sw(유형의 개념)

db - 데이터를 모아둔 토대(무형의 개념)

데이터 모델 = 데이터가 저장되는 기법에 관한 내용

문제해결을 위해선 우선 문제를 분해해야한다

그리고 하나씩 해결

db안쓰면 데이터 불일치 해결하기 힘들다. 다시말해 db를 이용하면 데이터를 관리하기가 편하다. 일관성을 유지할 수 있다

db는 원칙적으로 remote 가 맞지만 실습할 때는 local로 실습(내 컴퓨터에 구축하여 실습)

웹개발하려면 일단 서버단(백엔드) 구축. 그 다음 프론트 구축

db는 백엔드에 포함. 서버단에서 돌아감

데이터베이스 시스템 - dbms 도입하여 데이터 통합관리. 데이터 중복줄이고 무결성유지

데이터를 가진 쪽을 서버, 데이터를 요청하는 쪽을 클라이언트

응용프로그래머 : 응용프로그램 개발 - dbms접속 - db접근

db를 구축할 때 1차적으로 데이터를 가공하여 집어넣는다(이게 모델링)

SQL(Structured Query Language) - db생성 변경 삭제, 레코드 삽입 조회 수정, db백업 롤백 보안 관리 등에 의한 처리에 사용되는 표준언어

- 새로운 db 생성, 삭제 가능

- db에 자료 삽입 삭제 가능

- db안에 새로운 테이블 생성, 삭제 가능

- db에서 자료 검색, 추출 가능

SQL은 관계형 db와 비관계형 db(noSQL)로 나뉘는데 주로 관계형 db다룸.

SQL의 종류 - 데이터 정의어(DDL) - (DB생성 삭제),

조작어(DML) - (레코드 관리),

제어어((DCL) - DB백업 롤백 보안 관리)

개발자는 정의어와 조작어 다룸

검색할 때는 keyword 와 식별자로 검색

예약어(keyword) - 컴퓨터 프로그래밍 언어에서 이미 문법적인 용도로 사용되고 있기 때문에 식별자로 사용할 수 없는 단어들 (SELECT, FROM, Where)은 예약어

식별자(identifier) - 다른 것들과 구분하기 위해서 사용되는 변수의 이름, 상수의 이름, 함수의 이름, 사용자 정의 타입의 이름 등 '이름'을 일반화 해서 지칭하는 용어이다.

select 찾고자 하는 정보

from 테이블

모든 정보 검색하려면? select \* from 테이블명

가격이 10,000원 이상(검색조건)인 도서이름과 출판사 검색

SELECT bookname, publisher

FROM Book

Where price >= 10000; (조건절 + (필드명 + 연산자 + 값))

여기서 price는 식별자인 동시에 테이블의 레이블이다. 컬럼(열, 필드)명이라고도 한다.

레코드=row=행

데이터를 입출력하는 기본 단위가 레코드(행을 집어넣는다?)

IT의 철칙 - 숫자와 문자 구분.

'문자'

SQL은 대소문자 구분 안 함

스키마 = 구조

db를 구축할 때 설계도가 필요하다.

key(keyword)가 되는 조건 - 유일해야함, null이 돼선 안 됨

한 교수는 여러과목을 강의한다.

한 과목을 여러 교수가 강의한다.

교수와 과목 간의 관계 설정(강의로 엮임)

교수 - 강의 - 과목

관계 테이블을 만든다. 다대다는 테이블로 못 만듦. 일대다 구조로 만듦

erd(개체 관계도) - Entity Relation Diagram

엔티티는 테이블 의미

attibute는 컬럼

PK는 primary key(기본키)

FK는 외래키 - 상대방 참조 테이블의 기본키

db를 설계한다 - 개념 단계(조직 전체 관점)를 확실히 구성해야 프로그래밍까지 가능

외부단계(사용자 관점) 개념단계(조직 전체 관점) 내부단계(저장 장치 관점)

sql만들면 erd 만들어줌 반대로 erd 만들면 sql만들어주기도 함

dbms 든 자바든 구현하기 위한 도구일 뿐이다. 도구가 주가 돼선 안 됨

데이터베이스 모델링 - 그림으로 표현

노력하면 시간이 해결해준다

구글링 열심히 하고

창업은 과제라고 생각하는 사람이 한다

문제라고 인식해서 포기해버리면 창업못함

127.0.0.1 루프백주소 자기자신 주소

스키마는 테이블의 구조

show databases ; - 현재 만들어진 db 보여줌

drop database testdb; - db 제거

drop table testTable - 테이블제거

신텍스에러(문법오류), 런타임에러(실행중 오류), warning.

(

SNO VARCHAR(5) PRIMARY KEY, // sno 필드명는 varchar - 가변형 타입 primary는 기본키

SNAME VARCHAR(15), // 필드명 + 타입

STATUS INTEGER(2),

CITY VARCHAR(10)

);

CREATE TABLE SPJ

(

SNO VARCHAR(5) REFERENCES S(SNO), sno는 s테이블의 sno 참조

PNO VARCHAR(5) REFERENCES P(PNO),

JNO VARCHAR(5) REFERENCES J(JNO),

QTY INTEGER(5),

PRIMARY KEY(SNO,PNO,JNO) // PK가 복합키다. pk의 조건은 not null, 유일해야함. 여기서 not null이 되려면 3개가 필요함

순서는 참조테이블 만들고 그 후 관계 테이블 만든다

차례는 s p j, 만들고 spj테이블 만든다

테이블을 만들었으면 데이터를 집어넣는다

reverse engineering 하면 erd의 형태 만들어준다

왜 리버스? 정방향은 erd 만들고 db만드는것

리버스는 db만들고 erd 만드는 것

INSERT INTO S (SNO,SNAME,STATUS,CITY) VALUES

('S1','KRUNAL',10,'LONDON');

INSERT INTO S (SNO,SNAME,STATUS,CITY) VALUES

('S2','RAMESH',5,'INDIA');

INSERT INTO S (SNO,SNAME,STATUS,CITY) VALUES

('S3','VIVEK',4,'LONDON');

INSERT INTO S (SNO,SNAME,STATUS,CITY) VALUES

('S4','VIMAL',3,'JAPAN');

INSERT INTO S (SNO,SNAME,STATUS,CITY) VALUES

('S5','HEMAL',10,'KORIA');

INSERT INTO S (SNO,SNAME,STATUS,CITY) VALUES

('S6','RAJU',2,'CHINA');

INSERT INTO S (SNO,SNAME,STATUS,CITY) VALUES

('S7','VINU',3,'CHINA');

insert into 테이블명 컬럼(필드)명 values (넣을값);

SELECT \* FROM warehouse.spj; // db명.테이블명 - 테이블의 모든 데이터 가져온다

외부 스크립트 열어서 db 만들어 저장할 수 있다. 이때 create database warehouse; 하고 use warehouse; 해야함

중복을 줄이기 위해 데이터를 쪼갠다(정규화) 쪼개기 이전 상태의 정보를 알 수 있어야한다. 쪼갤 때 테이블의 pk를 관계 되는 테이블의 FK 로 넣어준다.

관계 데이터 모델

릴레이션(relation) – 행과 열로 구성된 테이블

릴레이션십 – 릴레이션 내에서 생성되는 관계 : 릴레이션 내 데이터들의 관계

릴레이션 간에 생성되는 관계 : 릴레이션 간의 관계

Key는 중복이 안될 것 같은 필드로 설정

키가 될 수 있는 키들을 후보키라고 함

데이터를 보고 어떻게 설계할 지 구상. 어떻게 중복줄일지

애트리뷰트(필드(파일시스템에서), 속성, 컬럼, 열). 차수? 컬럼의 수

투플(레코드(파일시스템에서), 행, row) 카디날리티(관계에서 씀. 독립된 테이블에서는 안씀) – 관계된 행의 수

스키마 – 테이블의미(관계 데이터에서)

인스턴스 - 행의 데이터, 연관된 데이터의 집합

도메인 – 속성이 가질 수 있는 값의 집합 (도서이름 ={축구의 역사, 축구아는 여자, 축구의 이해})

스키마의 표현

릴레이션 이름(속성1:도메인1. 속성2:도메인2, 속성3:도메인3….)

도서(도서이름, 도서번호, 출판사, 가격…)

릴레이션의 특징

1. 속성은 단일 값을 가진다 – 각 속성의 값은 도메인에 정의된 값만을 가지며 그값은 모두 단일 값이어야함.
2. 속성은 서로 다른 이름을 가진다 – 속성은 한 릴레이션에서 서로 다른 이름을 가져야만 함
3. 한 속성의 값은 모두 같은 도메인 값을 가진다 – 한 속성에 속한 열은 모두 그 속성에서 정의한 도메인 값만 가질 수 있음
4. 속성의 순서는 상관없다 – 속성의 순서가 달라도 릴레이션 스키마는 같음

예)릴레이션 스키마에서 (이름, 주소) 순으로 속성을 표시하거나 (주소, 이름) 순으로 표시해도 노상관

5. 릴레이션 내의 중복된 투플은 허용 안함 – 하나의 릴레이션 인스턴스 내에서는 서로 중복된 값을 가질 수 없다. 즉 모든 투플은 서로 값이 달라야함

6. 투플의 순서는 상관없다

관계 데이터 모델(릴레이션, 제약조건, 관계대수) ->

컴퓨터 시스템에 구현 관계 데이터베이스 시스템(릴레이션(sql로 생성 및 관리), 제약조건(sql로 제약선언), 관계대수(sql로 연산))

무결성 제약조건

1. 키 – 특정 투플을 식별할 때 사용하는 속성 혹은 속성의 집합. 릴레이션 간의 관계를 맺는 데도 사용

데이터를 보고 키를 찾거나 설정한다

기본키 선정 시 고려사향

릴레이션 내 투플을 식별할 수 있는 고유한 값 가져야함,

null값은 허용하지 않음.

변동이 일어나지 않아야함. (폰번호가 키가될 수 없는 이유? 폰 번호 없는 사람도 있어서, 변동이 있으므로)

릴레이션 스키마를 표현할 때 기본키는 밑줄을 그어 표시함

릴레이션 이름(속성1, 속성2, …속성n)

고객(고객번호, 이름, 전화번호..)

외래키 – 다른 릴레이션의 기본키를 참조하는 속성

다른 릴레이션의 기본키를 참조하여 관계 데이터 모델의 특징인 릴레이션 간의 **관계를 표현**

관계 릴레이션 – 평소에는 릴레이션 아니었는데 관계를 표현하려다 보니 릴레이션이 됨?

외래키 특징

관계 데이터 모델의 릴레이션 간의 관계 표현

다른 릴레이션의 기본키를 참조하는 속성임

참조하고(외래키) 참조하는(기본키) 양쪽 릴레이션의 도메인은 서로 같아야함

참조되는 값(기본키)이 변경되면 참조하는 값(외래키)도 변경됨.

자기 자신의 기본키를 참조하는 외래키도 가능 -

Null, 중복 허용

외래키가 기본키의 일부가 될 수 잇다

프로그래밍에서 함수는 라이브러리니까 필요할 때마다 꺼내본다

SELECT MIN(Price) AS SmallestPrice //as 뒤에는 출력되길 원하는 단어  
FROM Products;

SELECT ANIMAL\_ID, NAME

from ANIMAL\_INS

where INTAKE\_CONDITION='sick'; // 아픈 순으로 정렬

SELECT ANIMAL\_ID, NAME

from ANIMAL\_INS

where INTAKE\_CONDITION != 'Aged' //어린 순으로 정렬

order by ANIMAL\_ID;

연산자 =, like 차이?

=은 정확히 일치하는 값

Like는 유사한 값

% - 없거나, 하나있거나, 여러 개 있거나 (a% : a로 시작하는, %a : a로 끝나는 데이터 찾음, %or% or이 아무데나 있을 때) --- 본문 검색할 때 사용한다

‘\_’ – 하나 일치할 때 ( \_r : 반드시 r 포함하는 데이터 찾음)

mileage decimal(7, 0) unsigned default 0 (7자리, 소수점 0번째 까지) 즉 정수형. Null이 안되도록 기본값 0부여

stat enum('Y','N') default 'Y', -- 들어갈 수 있는 게 y n중 하나. 기본값으로 y부여

select \*

from db0215.members; (db . 테이블명) – 테이블에 있는 모든 데이터 출력

데이터 무결성 – db에 저장된 데이터의 일관성과 정확성을 지키는 것

도메인 무결성 제약조건 – 도메인 제약이라고 함. 릴레이션 내의 투플들이 각 속성의 도메인에 지정된 값만을 가져야한다는 조건 sql 문에서 type, null, 기본값, 체크 등을 사용하여 지정 가능

개체 무결성 제약조건 – 기본키 제약이라고 함. 기본키를 지정하고 그에 따른 무결성 원칙, 중복 안되고 null 안되고.

삽입 – 기본키 값이 같으면 삽입이 금지됨

수정 – 기본키 값이 같거나 null로도 수정 금지

삭제 – 특별한 확인 필요 없고 즉시 수행

참조 무결성 제약 조건

외래키 제약이라고 함. 릴레이션 간의 참조 관계를 선언하는 제약조건. 자식 릴레이션 외래키는 부모 릴레이션의 기본키와 **도메인(행, 투플)이 동일해야 하며** 자식 릴레이션 값이 변경될 떄 부모 릴레이션의 제약을 받는다.

이름은 다를 수 있지만 데이터 (타입)은 다를 수 없다

create table orders (

orderid int not null,

ordernumber int not null,

personid int,

primary key(orderid),

foreign key(personid) references persons(id) -- w3school 에 personid 돼있는데 오타임. persons테이블에 personid 컬럼이 없다

); -- 외래키가 persons 테이블에서 id컬럼 참조

insert into orders value(1,77895,4); // 참조 무결성에 의해 데이터 삽입불가(person id)에서 투플이 3까지 밖에 없으므로)

insert into orders value(2,44678,3);

insert into orders value(3,22456,2);

insert into orders value(4,24562,1);

부모 테이블의 기본키를 자식 테이블에 그냥 넣으면 관계가 맺어진다

참조 무결성 제약조건

**삭제 – 같이 지우든지 못지우게 하든지(기본 세팅은 못지우게 돼있다), 수정도 마찬가지**

학과 – 부모릴레이션 – 참조하는 테이블을 같이 삭제할 수 있어서 금지하거나 다른 추가작업이 필요함

학생 – 바로 삭제가능

Restricted – 자식 릴레이션에서 참조하고 있을 경우 부모 릴레이션의 삭제작업 거부

예) 학과 릴레이션의 투플 삭제 거부

관계대수

릴레이션의 수학적 개념 – 릴레이션 역시 집합이므로 합집합 교집합 카티전프로덕트 연산 가능

관계대수 연산자

셀렉션 - 행 단위로 출력

릴레이션에서 조건에 맞는 투플 추출

Ex) Select \* from R1

Where A=a1 or A=a2;

프로젝션 - 열 단위로 출력

릴레이션에서 조건에 맞는 속성(컬럼 추출)

Ex) Select A,B from R2

프로젝션과 셀력션 복합 사용

Select 도서이름, 출판사 from 테이블

Where 가격 >=9000

Where 조건절에는 and or 등이 들어가야함 ‘,’ 안됨

Erd 만들어서 포워드 엔지니어링 하면 만들어짐

db설계 단계

1단계 : 요구사항분석 - db용도 파악.

결과물 : 요구 사항 명세서-

2단계 : 개념적 설계 = dbms에 독립적인 개념적 구조 설계.

결과물 : 개념적스키마(E-R-다이어그램) 즉 설계도. ERD를 손수 그려보는게 도움됨

3단계 : 논리적 설계 – DBMS에 적합한 논리적 구조 설계

결과물 : 논리적 스키마(릴레이션 스키마)

4단계 : 물리적설계 – DBMS로 구현 가능한 물리적 구조 설계

결과물 : 물리적 스키마

5단계 : 구현 - SQL문을 작성한 후 이를 DBMS에서 실행하여 DB생성

Having 은 group by에 대한 조건절

Join은 두개의 테이플 있을 때 사용

SELECT Customers.customer\_id, Customers.first\_name, Count(Orders.order\_id) AS order\_count

FROM Customers

LEFT JOIN Orders // custormers 테이블과 orders 테이블 조인 left는 왼쪽 테이블에 있는거 다 출력. 그냥 join은 일치하는 것만

ON Customers.customer\_id = Orders.customer\_id // 각 id가 일치되는 것 끼리. Customer\_id가 기본키. Orders 테이블에서 외래키

GROUP BY Customers.customer\_id;

I am working on MySQL. I want to get the name of city which has the largest population among the cities which have the countrycode is ‘kor’. Plus, I want to get the number of population

Use max function

Program please.

부모 자식 관계 형성할 때 항상 부모 테이블 먼저 만듦 일대다 구조에서 일이 부모, 다가 자식이됨.

foreign key (custid) references customer(custid) on delete cascade – 외래키 만드는법

drop database if exsists madang;

drop user if exists madang@localhost; // 아이디 / 호스트

create user madang@localhost identified with mysql\_native\_password by ‘madang’ create database madang – 비밀번호 부여

create database madaing;

grant all privileges on madang.\* to madang@localhost with grant option; - 권한부여, DCL, madang은 db. \*은 table임

commit; - 적용

d d l / d m l /d c l ?

데이터 정의어 d d l

Create, alter, drop…

Select \*

From book

Where publisher in(‘굿스포츠’,대한미디어’’)

Select \*

From book

Where publisher not in(‘굿스포츠’, ‘대한미디어’);

이걸 or로 표현하면?

축구의 역사를 출간한 출판사를 검색

select bookname, publisher

from book

where bookname like '축구의 역사'; or

select bookname, publisher

from book

where bookname = '축구의 역사';

or

select bookname, publisher

from book

where bookname in ('축구의 역사');

도서이름에 출구가 포함된 출판사 검색

select bookname, publisher

from book

where bookname like '%축구%';

도서이름의 왼쪽 두 번째 위치에 ‘구’라는 문자열을 갖는 도서 검색

select \*

from book

where bookname like '%구%';

or

select \*

from book

where bookname like '\_구%';

복합조건

축구에 관한 도서 중 가격이 20,000 원 이상인 도서 검색

select \*

from book

where bookname like '%축구%' and price >= 20000;

출판사가 굿스포츠 혹은 대한미디어인 도서 검색

select \*

from book

where publisher = '굿스포츠' or publisher= '대한미디어';

// 중복제거 하여 이름 개수 카운트. Null은 제외

SELECT count(distinct NAME)

from ANIMAL\_INS

where name is not null;

-- 도서를 이름순으로 검색

select \*

from book

order by bookname;

-- 도서를 가격순으로 검색, 가격 같으면 이름순으로 검색

select \*

from book

order by price, bookname;

-- 도서를 갸격의 내림차순으로 검색. 만약 가격 같다면 출판사의 오름차순으로 검색

select \*

from book

order by price desc, publisher asc;

-- 집게 함수

-- 고객이 주문한 도서의 총 판매액을 구하라

select sum(salesprice) as revenue

from orders;

-- 2번 김연아 고객이 주문한 도서의 총 판매액

select sum(salesprice) as 김연아구매액

from orders

where custid = 2;

-- 고객이 주문한 도서의 총 판매액, 평균값, 최저가, 최고가 구하기

select sum(salesprice) as total, round(avg(salesprice), 0) as average, min(salesprice) as minimum, max(salesprice) as maximum

from orders ;

-- 마당서점의 도서 판매 건수 구하기

select count(\*)

from orders;

-- 고객별로 주문한 도서의 총 수량과 총 판매액 구하기 + 소계(roll up)

select custid, count(\*) as 도서수량, sum(salesprice) as 총판매액

from orders

group by custid with rollup;

-- 가격이 8,000원 이상인 도서를 구매한 고객에 대하여 고객별 주문 도서의 총수량을 구하라(단, 두권 이상 구매한 고객만 구함)

select custid, count(\*) as 구매수량

from orders

where salesprice >= 8000

group by custid

having count(\*) >= 2;

\*\*having 절은 반드시 group by절과 같이 작성하고 where 절보다 뒤에 나와야함

그리고 having 절에는 집계함수가 와야함(sum avg max min count)

-- 조인 : 고객과 고객의 주문에 관한 데이터를 모두 보여라. 일반적인 조인 - 같은 것 끼리만 합침

select \*

from customer, orders

where customer.custid = orders.custid

order by customer.custid;

-- 고객의 이름과 고객이 주문한 도서의 판매가격을 검색

select name, sum(salesprice)

from customer, orders

where customer.custid = orders.custid

group by customer.name

order by customer.name;

-- 고객의 이름과 고객이 주문한 도서의 이름 구하기

select customer.name, book.bookname

from customer, orders, book

where customer.custid = orders.custid and orders.bookid = book.bookid -- 일단 다 연결해놓고 select에 있는거 두개만 끄집어내라

조인 : 동등, 자연, 외부 이 세개 많이 씀

동등 – 두 릴레이션 간의 같은 값을 가진 집합

자연 – 동등 조인에서 중복 속성 제거

외부 – 자연 조인 후 각각 왼쪽(left) 오른쪽(right) 양쪽(full)의 모든 값을 결과로 추출, 조인이 실패(또는 값이 없을 경우)한 쪽의 값을 null로 채움

조인 하는 이유? 데이터 정규화하는 과정에서 릴레이션이 분해되기 때문에 조인 필요

일반조인 informal

Select <속성들>

From 테이블1, 테이블2 // 조인 명령문 쓰지 않고도 이렇게 사용가능 informal한 방법

Where <조인조건> and <검색조건>

일반조인 - formal

Select <속성들>

From 테이블1 inner join 테이블2 on <조인조건>

Where <검색조건> - where절에서는 =이런거 안되고 is 쓰라고 !!!! (=는 됨)

동등 조인 – 중복 존재

자연 조인 – 중복제거

레프트 조인 – 왼쪽에 있는 모든 투플 포함(일치 하지 않는 투플도). 오른쪽 투플은 일치하는 것만 출력

라이트 조인 – 마찬가지

Join, inner join 키워드 없이 where 조건절에 쓰나 같은 결과가 나온다(제일 간단한 방법)

Inner 조인 – 동등조인. 자료형은 불변이나 컬럼명은 다를 수 있다

프로그래머스 lv3 없어진 기록 찾기

SELECT ANIMAL\_OUTS.ANIMAL\_ID, ANIMAL\_OUTS.NAME

from ANIMAL\_INS right outer join ANIMAL\_OUTS on ANIMAL\_INS.ANIMAL\_ID = ANIMAL\_OUTS.ANIMAL\_ID

where ANIMAL\_INS.ANIMAL\_ID is null

order by ANIMAL\_OUTS.ANIMAL\_ID;

조인에서 null 처리하는 방법

SELECT Customers.customer\_id, Customers.first\_name, ifnull(Orders.amount, "no")

FROM Customers

LEFT JOIN Orders

ON Customers.customer\_id = Orders.customer;

부속질의는 가장 밑에서부터 먼저 연산한다고 생각(릴레이션이 다 다르다)

상관부속질의 – 릴레이션이 같다. 중첩부속질의라고 할 수도 있다. 상위부속질의와 하위부속질의가 독립적이지 않다. 서로 관련있다. 연산 순서는 위에서부터 먼저 연산

정의어

Unique – 값은 유일한데 키는 아닐 때

Check – 제약사항. 무결성 제약조건

On delete cascade – 고객의 정보 삭제하면 고객이 주문한 정보까지 연쇄적으로 삭제

즉 부모릴레이션에서 삭제하면 자식릴레이션에서도 삭제?

Alter문

Add, drop, modify

Add<제약이름>, drop<제약이름> : 제약사항 추가 or 삭제 시 사용

Drop문

레코드 삭제 - delete 레코드명 from 테이블명

조작어

Insert update delete slelct

Update 문

기본 문법

Update 테이블명

Set 속성이름1=갑1[, 속성이름2=값2…] // 값 1은 새로운 값

[where <검색조건>];

서브쿼리 = 서브쿼리 먼저 수행, 그 다음 메인쿼리

서브쿼리, 조인쿼리 편한거 쓰면됨

기본키 추가

Alter table newbook add primary key(bookid)

기본키 제거

Alter table newbook drom primary key

SQL 함수 – 내장함수와 사용자 정의 함수로 나뉨

행번호 출력 – 내장함수는 아니지만 자주 사용되는 문법

부속질의 – 조인보다 성능이 더 좋음

스칼라 부속질의 - select절에 사용. 컬럼 값으로 사용

인라인 뷰 - from절에서 사용되는 부속질의. 일종의 가상 테이블. 상관 부속질의로 사용 불가

뷰는 수정한다는 개념이 아니다.

Primary key, unique key 차이? 유니크 키는 null 가능

인덱스

기본키도 인덱스의 하나다

프로젝트할 때 프로시져 만들어서 써라