데이터 형식

**데이터 타입 (숫자 데이터 형식)**

테이블이(가) 표시된 사진

자동 생성된 설명

**​**

**데이터 타입 (문자 데이터 형식)**

테이블이(가) 표시된 사진

자동 생성된 설명

\* char 와 varchar의 차이

char(6)에 대한민국을 저장하면

char는 공백까지 대한민국OO 총 6의 데이터를 저장한다면

varchar는 대한민국의 4개의 데이터만 저장한다.

​

**left(str,n)**

**right (str,n)**

**mid(str,a,b)**

**데이터 타입(날짜 데이터 형식)**

테이블이(가) 표시된 사진

자동 생성된 설명

DATE : 년월일, TIME : 시분초, DATETIME : 년월일시분초

CREATE TABLE Tbl\_Car (

ID int PRIMARY KEY IDENTITY(1,1),

Color nvarchar(30));

CREATE TABLE Tbl\_Person (

ID int PRIMARY KEY IDENTITY(1,1),

CarID int REFERENCES Tbl\_Car(ID),

Name nvarchar(20));

INSERT INTO Tbl\_Person (CarID, Name) VALUES (NULL, 'Timmy');

01 SQL 시작하기

변수

SELECT문에서 대입할 때는 := 이렇게만 대입이 가능

SET @HOUR := 8

\*SELECT 문 순서\*

FROM -> WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY

\*is null

Is not null

SELECT if ( ISNULL(user\_email), 'exist!', 'it is null' ) FROM user\_mstr;

ifnull(컬럼,초기값)

SELECT (

CASE

WHEN 조건1 THEN '조건1 반환값'

WHEN 조건2 THEN '조건2 반환값'

ELSE '충족되는 조건 없을 때 반환값'

END)

SELECT if([조건],[참일 때 값],[거짓일 때 값])

DESC [table];

컬럼, 레코드

SELECT+ [컬럼] or \*(데이터 너무 많으면 렉걸림) or DISTINCT(뚜렷한) [컬럼]: 중복 제거

FROM + [테이블]

Where+ [레코드]: 조건

A AND B

A && B

A OR B

A || B

NOT A

!A

BETWEEN A AND B(이상 이하)

[컬럼] IN (A,B)

NOT IN

[DATE , MONTH , DAY , HOUR , MINUTE , SECOND 사용해 날짜, 월, 일, 시, 분, 초 확인 실시]

1. DATE() 함수는 전달받은 값에 해당하는 날짜 정보를 반환합니다

2. MONTH() 함수는 전달받은 값에서 월에 해당하는 값만을 반환하며, 0부터 12 사이의 값을 가집니다

3. DAY() 함수는 전달받은 값에서 일자에 해당하는 값만을 반환하며, 0부터 31 사이의 값을 가집니다

4. HOUR() 함수는 전달받은 값에서 시간에 해당하는 값만을 반환하며, 0부터 23 사이의 값을 가집니다

5. MINUTE() 함수는 전달받은 값에서 분에 해당하는 값만을 반환하며, 0부터 59 사이의 값을 가집니다

6. SECOND() 함수는 전달받은 값에서 초에 해당하는 값만을 반환하며, 0부터 59 사이의 값을 가집니다

CAST (A as B): A를 B로 형식을 바꿈



02 데이터를 제어하는 DML

SQL 명령어 종류

DML(Data Manipulation Language):

데이터에서 유사한 값 찾기:

Where [컬럼] LIKE ‘어린왕자’

Where [컬럼] LIKE ‘%왕자’

Where [컬럼] LIKE ‘%린왕%’

데이터를 정렬하고 싶을 떄

Order by [컬럼] asc/desc

\*컬럼 각각에 대해 따로 선언 가능(ex) order by A asc, B desc)

데이터를 삽입

Insert into [테이블]([컬럼A],[컬럼B])

Values([데이터1],[데이터2]);

\*컬럼을 명시하지 않으면 순서대로 값을 삽입

\*여러 개 삽입시 values(a),(b);

데이터 수정

Update [테이블]

Set [컬럼]=’a’(변경할 값)

Where 조건;(현재 값)

데이터행 삭제

Delete

From [테이블]

Where 조건;

\*where 조건 없을 시 모든 데이터 삭제

DDL

DCL

TCL

03 SQL과 함수

Count: 검색한 결과의 데이터의 개수를 가져오는 내장 함수(NULL 제외)

Select count([컬럼]) from [테이블]; ->컴럼의 개수

Select count([\*]) from [테이블]; -> 모든 컬럼의 개수(가장 많은 데이터를 가진 컬럼 기준?)

LIMIT: 테이블에서 출력하고자 하는 데이터의 개수를 제한하는 명령

Select \* from book limit a: a개 가져오기

Select \* from book limit a,b: a부터, b까지 가져오기

SUM & AVG: 지정한 컬럼들의 합이나 평균값을 구해주는 내장함수

Select sum([컬럼]) from [테이블];

Select avg([컬럼]) from [테이블];

MAX & MIN: 존재하는 데이터에서 최대값, 최솟값을 가져오는 내장함수(문자형도 가능(사전형))

Select min([컬럼]) from [테이블];

함수:

1. 데이터 값을 계산하거나 조작: 행 함수

2. 행의 그룹을 계산하거나 요약: 그룹 함수

3. 열의 데이터 타입을 변환

04 다수의 테이블 제어하기

데이터 그룸 짓기

select [컬럼], count(\*) from [테이블] group by [컬럼];

select [컬럼], sum([컬럼2]) from [테이블] group by [컬럼];

select [컬럼], max([컬럼2]) from [테이블] group by [컬럼];

select [컬럼], min([컬럼2]) from [테이블] group by [컬럼];

데이터 그룹에 조건 적용하기

Select [컬럼]

From [테이블]

Group by [컬럽]

Having 컬럼의 조건;

\*join

1. inner join (=join)

2. left join

3. right join

두개의 테이블에서 한번에 조회하기

\*우리나라 데이터 관리법상 한 사람을 특정할 수 있는 모든 데이터를 하나의 테이블에 넣으면 안됨

Select \*

From [테이블1]

Inner join [테이블2]

On [테이블2].[컬럼] = [테이블2].[컬럼];

두 테이블중 겹치는 부분만 출력

Left join(null이 있을 때도 사용 가능)

Select \*

From [테이블1]

left join [테이블2]

On [테이블2].[컬럼] = [테이블2].[컬럼];

(테이블 1의 모든 데이터가 출력됨) 테이블1>테이블2 : 왼쪽 데이터, 겹치는 부분을 출력

Left join(null이 있을 때도 사용 가능)

Select \*

From [테이블1]

right join [테이블2]

On [테이블2].[컬럼] = [테이블2].[컬럼];

(테이블 2의 모든 데이터가 출력됨) 테이블1<테이블2 : 오른쪽 데이터, 겹치는 부분을 출력

서브쿼리: 하나의 쿼리 안에 포함된 또 하나의 쿼리

알려지지 않은 기준을 이용한 검색에 유용

메인 쿼리가 실행되기 이전에 한 번만 실행

한 문장에서 여러 번 사용 가능(재귀)

Select \* from employee

Where 급여 > (select 급여 from employee where 이름=’elice’);

괄호와 함께 사용해야함

서브쿼리 안에서는 order by 사용 금지

연산자의 오른쪽에 사용해야함

select문으로만 작성 가능

반환에 따른 분류

단일 행 서브쿼리 정의: 결과가 한 행만 나오는 서브 쿼리, 서브쿼리가 결과를 1개의 값만 반환하고 이 결과를 메인 쿼리로 전달하는 쿼리

다중 행 서브쿼리 정의: 서브쿼리가 결과를 2개 이상 반환하고, 이 결과를 메인쿼리에 전달하는 쿼리

select \* from employee

where 급여 in (

select max(급여) from employee group by 부서번호);

in

any: 중에 하나라도~

all: 중에 전부~

<any: 최대값 >any: 최소값

>all: 최대값 <all: 최소값

날짜데이터는 무조건 ‘xxxx-xx-xx’

위치에 따른 분류

1. 일반적인 서브쿼리의 형태(where절에서 사용)

2. 스칼라 서브쿼리: select절에 사용, 오로지 한 행만 반환, 마치 join을 사용하는 것과 비슷

왜 join말고 스칼라 서브쿼리 사용?->대용량 데이터 취급할 때 계산속도가 더 빠르니까

select students.name, (

select math

from middle\_test as m

where m.student\_id = students.student\_id

) as middle\_avg

from students;

A (as) B : A를 B라고 별명 짓는다.

(CAST(REPLACE(CURRENT\_DATE,’-‘,’’) AS UNSIGNED) -  
CAST(REPLACE(birthdate,’-‘,’’) AS UNSIGNED)) / 10000 ) as age

SELECT DISTINCT emp\_no, (

SELECT AVG(salary)

FROM salaries s1

WHERE s1.emp\_no = s2.emp\_no

) AS avg\_salary

FROM salaries s2

SET @HOUR := -1;

SELECT (@HOUR := @HOUR + 1) AS HOUR,

(SELECT COUNT(\*) FROM ANIMAL\_OUTS WHERE HOUR(DATETIME) = @HOUR)

AS COUNT

FROM ANIMAL\_OUTS WHERE @HOUR < 23;