

# Ch07. 데이터베이스 언어 SQL

SQL의 역할을 이해하고, 이를 기능별로 분류해본다.

SQL의 데이터 정의 기능을 예제를 통해 익힌다.

SQL의 데이터 조작 기능을 예제를 통해 익힌다.

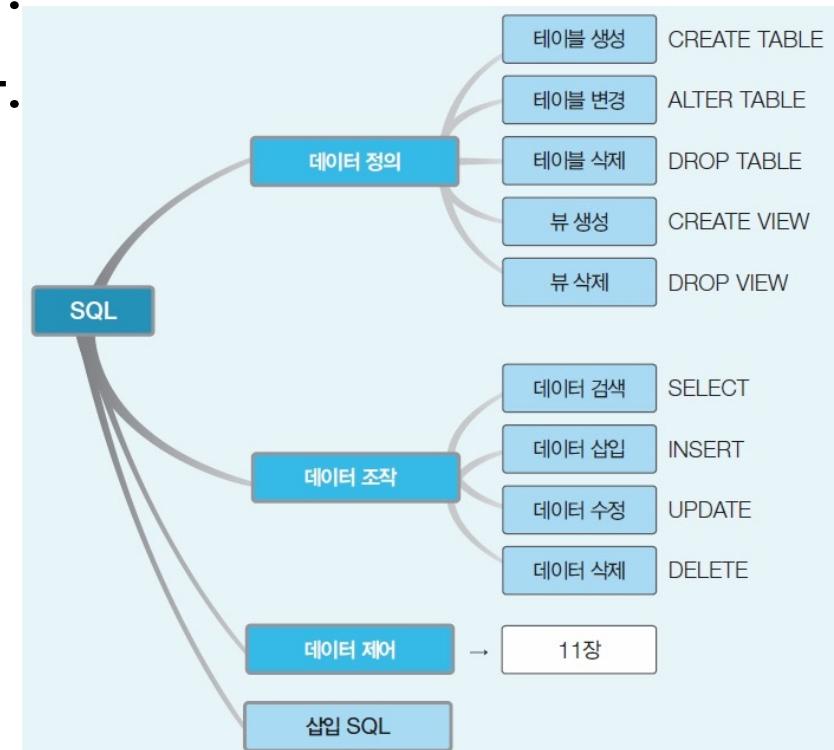
뷰의 개념과 장점을 이해한다.

삽입 SQL의 역할을 이해한다



# 학습목표

- SQL의 역할을 이해하고, 이를 기능별로 분류해본다.
- SQL의 데이터 정의 기능을 예제를 통해 익힌다.
- SQL의 데이터 조작 기능을 예제를 통해 익힌다.
- 뷰의 개념과 장점을 이해한다.
- 삽입 SQL의 역할을 이해한다.



# 01 SQL의 소개

## SQL(Structured Query Language)

- 의미
  - 관계 데이터베이스를 위한 표준 질의어
  - 비절차적 데이터 언어
- 발전 역사
  - SEQUEL(Structured English QUERy Language)에서 유래
    - › SEQUEL : 연구용 관계 데이터베이스 관리 시스템인 SYSTEM을 위한 언어
  - 미국 표준 연구소인 ANSI와 국제 표준화 기구인 ISO에서 표준화 작업을 진행
    - › 계속 수정 및 보완되고 있음
- 사용 방식
  - 대화식 SQL : 데이터베이스 관리 시스템에 직접 접근해 질의를 작성하여 실행
  - 삽입 SQL : 프로그래밍 언어로 작성된 응용 프로그램에 삽입

# 01 SQL의 소개

## SQL의 분류

Data Manipulation Language

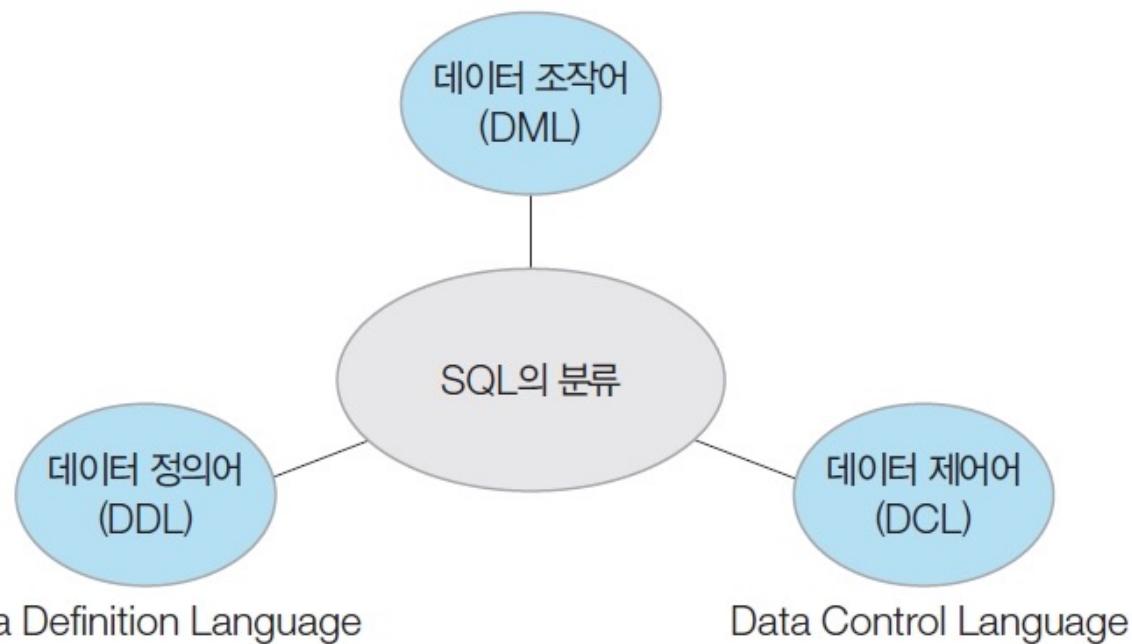


그림 7-1 SQL의 분류

# 01 SQL의 소개

## SQL의 분류

- 데이터 정의어(DDL)
  - 테이블을 생성하고 변경·제거하는 기능을 제공
- 데이터 조작어(DML)
  - 테이블에 새 데이터를 삽입하거나, 테이블에 저장된 데이터를 수정·삭제·검색하는 기능을 제공
- 데이터 제어어(DCL)
  - 보안을 위해 데이터에 대한 접근 및 사용 권한을 사용자별로 부여하거나 취소하는 기능을 제공

# 01 SQL의 소개

## 질의에 사용할 판매 데이터베이스 : 고객 테이블

고객 테이블

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500
orange	김용욱	22	silver	학생	0
melon	성원용	35	gold	회사원	5000
peach	오형준	NULL	silver	의사	300
pear	채광주	31	silver	회사원	500

# 01 SQL의 소개

## 질의에 사용할 판매 데이터베이스 : 제품 테이블

제품 테이블

제품번호	제품명	재고량	단가	제조업체
p01	그냥만두	5000	4500	대한식품
p02	매운쫄면	2500	5500	민국푸드
p03	쿵떡파이	3600	2600	한빛제과
p04	맛난초콜릿	1250	2500	한빛제과
p05	얼큰라면	2200	1200	대한식품
p06	통통우동	1000	1550	민국푸드
p07	달콤비스킷	1650	1500	한빛제과

# 01 SQL의 소개

## 질의에 사용할 판매 데이터베이스 : 주문 테이블

주문 테이블

주문번호	주문고객	주문제품	수량	배송지	주문일자
o01	apple	p03	10	서울시 마포구	2022-01-01
o02	melon	p01	5	인천시 계양구	2022-01-10
o03	banana	p06	45	경기도 부천시	2022-01-11
o04	carrot	p02	8	부산시 금정구	2022-02-01
o05	melon	p06	36	경기도 용인시	2022-02-20
o06	banana	p01	19	충청북도 보은군	2022-03-02
o07	apple	p03	22	서울시 영등포구	2022-03-15
o08	pear	p02	50	강원도 춘천시	2022-04-10
o09	banana	p04	15	전라남도 목포시	2022-04-11
o10	carrot	p03	20	경기도 안양시	2022-05-22

그림 7-2 질의에 사용할 판매 데이터베이스 : 고객, 제품, 주문 테이블

## 02 SQL을 이용한 데이터 정의

### SQL의 데이터 정의 기능

- 테이블 생성, 변경, 삭제



그림 7-3 SQL의 데이터 정의 기능

# 02 SQL을 이용한 데이터 정의

## ■ 테이블 생성 : CREATE TABLE 문

```
CREATE TABLE 테이블_이름 (
```

- ① 속성\_이름 데이터\_타입 [NOT NULL] [DEFAULT 기본\_값]
  - ② [PRIMARY KEY (속성\_리스트)]
  - ③ [UNIQUE (속성\_리스트)]
  - ④ [FOREIGN KEY (속성\_리스트) REFERENCES 테이블\_이름(속성\_리스트)]  
[ON DELETE 옵션] [ON UPDATE 옵션]
  - ⑤ [CONSTRAINT 이름] [CHECK(조건)]
- ```
);
```

- ❖ [ ]의 내용은 생략이 가능
- ❖ SQL 질의문은 세미콜론(:)으로 문장의 끝을 표시
- ❖ SQL 질의문은 대소문자를 구분하지 않음

## 02 SQL을 이용한 데이터 정의

### ■ 테이블 생성 : CREATE TABLE 문

- ① : 테이블을 구성하는 각 속성의 이름, 데이터 타입, 기본 제약 사항 정의
- ② : 기본키 정의
- ③ : 대체키 정의
- ④ : 외래키 정의
- ⑤ : 데이터 무결성을 위한 제약조건 정의

# 02 SQL을 이용한 데이터 정의

## ■ 테이블 생성 : CREATE TABLE 문

### ▪ 속성의 정의

- 테이블을 구성하는 각 속성의 데이터 타입을 선택한 다음 널 값 허용 여부와 기본 값 필요 여부를 결정
- NOT NULL
  - › 속성이 널 값을 허용하지 않음을 의미하는 키워드
  - › 예) 고객아이디 VARCHAR(20) NOT NULL
- DEFAULT
  - › 속성의 기본 값을 지정하는 키워드
  - › 예) 적립금 INT DEFAULT 0
  - › 예) 담당자 VARCHAR(10) DEFAULT '방경아'

문자열이나 날짜 데이터는  
작은 따옴표로 묶어서 표현  
(작은 따옴표로 묶여진 문자열은  
대소문자를 구분함)

# 02 SQL을 이용한 데이터 정의

## ■ 테이블 생성 : CREATE TABLE 문

표 7-1 속성의 데이터 타입

| 데이터 타입                                | 의미                                                    |
|---------------------------------------|-------------------------------------------------------|
| INT 또는 INTEGER                        | 정수                                                    |
| SMALLINT                              | INT보다 작은 정수                                           |
| CHAR(n) 또는 CHARACTER(n)               | 길이가 n인 고정 길이의 문자열                                     |
| VARCHAR(n) 또는<br>CHARACTER VARYING(n) | 최대 길이가 n인 가변 길이의 문자열                                  |
| NUMERIC(p, s) 또는 DECIMAL(p, s)        | 고정 소수점 실수<br>p는 소수점을 제외한 전체 숫자의 길이고, s는 소수점 이하 숫자의 길이 |
| FLOAT(n)                              | 길이가 n인 부동 소수점 실수                                      |
| REAL                                  | 부동 소수점 실수                                             |
| DATE                                  | 연, 월, 일로 표현되는 날짜                                      |
| TIME                                  | 시, 분, 초로 표현되는 시간                                      |
| DATETIME                              | 날짜와 시간                                                |

# 02 SQL을 이용한 데이터 정의

## ■ 테이블 생성 : CREATE TABLE 문

- 키의 정의
  - PRIMARY KEY
    - › 기본키를 지정하는 키워드
    - › 예) PRIMARY KEY(고객아이디)
    - › 예) PRIMARY KEY(주문고객, 주문제품)
  - UNIQUE
    - › 대체키를 지정하는 키워드
    - › 대체키로 지정되는 속성의 값은 유일성을 가지며 기본키와 달리 널 값이 허용됨
    - › 예) UNIQUE(고객이름)

# 02 SQL을 이용한 데이터 정의

## ■ 테이블 생성 : CREATE TABLE 문

### ▪ 키의 정의

무엇인가?

- FOREIGN KEY

- › 외래키를 지정하는 키워드
- › 외래키가 어떤 테이블의 무슨 속성을 참조하는지 REFERENCES 키워드 다음에 제시
- › 참조 무결성 제약조건 유지를 위해 참조되는 테이블에서 투플 삭제 시 처리 방법을 지정하는 옵션
  - ON DELETE NO ACTION: 투플을 삭제하지 못하게 함
  - ON DELETE CASCADE: 관련 투플을 함께 삭제함
  - ON DELETE SET NULL: 관련 투플의 외래키 값을 NULL로 변경함
  - ON DELETE SET DEFAULT: 관련 투플의 외래키 값을 미리 지정한 기본 값으로 변경함

$R \sqcup S$       R, S 도메인이 같아야 함

X 차수, 차원과 차이가 있어 합이 안됨

Y 연산이나 다른 것으로는 합이 안됨

Z ? 같다

동등한  
동일하게 작용하는 차원

자연연  
차원과 동일

세이연  
단위하는 곳 기준으로  
가져온다

X

X<sub>n</sub>

X

R 차원 + S 차원 = 합동

오른쪽은

X 없는 경우에서 Null값

DOL 정의

DML 쿼리

DCL 제한

;으로 끝나야지 문장이?      mysql 세이온은 괜찮

use 데이터베이스;

show databases;

# 02 SQL을 이용한 데이터 정의

## ■ 테이블 생성 : CREATE TABLE 문

- 키의 정의

- FOREIGN KEY

- › 참조 무결성 제약조건 유지를 위해 참조되는 테이블에서 투플 변경 시 처리 방법을 지정하는 옵션
      - ON UPDATE NO ACTION: 투플을 변경하지 못하게 함
      - ON UPDATE CASCADE : 관련 투플에서 외래키 값을 함께 변경함
      - ON UPDATE SET NULL : 관련 투플의 외래키 값을 NULL로 변경함
      - ON UPDATE SET DEFAULT : 관련 투플의 외래키 값을 미리 지정한 기본 값으로 변경함
    - › 예) FOREIGN KEY(소속부서) REFERENCES 부서(부서번호)
    - › 예) FOREIGN KEY(소속부서) REFERENCES 부서(부서번호)  
ON DELETE CASCADE ON UPDATE CASCADE

# 02 SQL을 이용한 데이터 정의

## 테이블 생성 : CREATE TABLE 문

[참조 무결성 제약조건 유지를 위한 투플 삭제 예]



그림 7-4 외래키를 통해 관계를 맺고 있는 2개의 테이블

- ON DELETE NO ACTION: 부서 테이블의 투플을 삭제하지 못하게 함
- ON DELETE CASCADE : 사원 테이블에서 홍보부에 근무하는 정소화 사원 투플도 함께 삭제
- ON DELETE SET NULL : 사원 테이블에서 정소화 사원의 소속부서 속성 값을 NULL로 변경
- ON DELETE SET DEFAULT : 사원 테이블에서 정소화 사원의 소속부서 속성 값을 기본 값으로 변경

# 02 SQL을 이용한 데이터 정의

## ■ 테이블 생성 : CREATE TABLE 문

- 데이터 무결성 제약조건의 정의
  - CHECK
    - › 테이블에 정확하고 유효한 데이터를 유지하기 위해 특정 속성에 대한 제약조건을 지정
    - › CONSTRAINT 키워드와 함께 고유의 이름을 부여할 수도 있음
    - › 예) CHECK(재고량 >= 0 AND 재고량 <= 10000)
    - › 예) CONSTRAINT CHK\_CPY CHECK(제조업체 = '한빛제과')

# 02 SQL을 이용한 데이터 정의

## 고객 테이블 생성을 위한 CREATE TABLE 문 작성 예

예제 7-1

고객 테이블은 고객아이디, 고객이름, 나이, 등급, 직업, 적립금 속성으로 구성되고, 고객아이디 속성이 기본키다. 고객이름과 등급 속성은 값을 반드시 입력해야 하고, 적립금 속성은 값을 입력하지 않으면 0이 기본으로 입력되도록 고객 테이블을 생성해보자.

*Customer*

▶▶ CREATE TABLE 고객 (  
*customer\_id* 고객아이디 VARCHAR(20) NOT NULL,  
*customer\_name* 고객이름 VARCHAR(10) NOT NULL,  
*grade* 나이 INT,  
등급 VARCHAR(10) NOT NULL,  
직업 VARCHAR(20),  
적립금 INT DEFAULT 0,  
PRIMARY KEY(고객아이디)  
);

고객 테이블

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| peach  | 오형준  | NULL | silver | 의사  | 300  |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |

# 02 SQL을 이용한 데이터 정의

## 제품 테이블 생성을 위한 CREATE TABLE 문 작성 예

### 예제 7-2

제품 테이블은 제품번호, 제품명, 재고량, 단가, 제조업체 속성으로 구성되고, 제품번호 속성이 기본키다. 재고량이 항상 0개 이상 10,000개 이하를 유지하도록 제품 테이블을 생성해보자.

▶▶ CREATE TABLE 제품 (

    제품번호 CHAR(3) NOT NULL,  
    제품명 VARCHAR(20),  
    재고량 INT,  
    단가 INT,  
    제조업체 VARCHAR(20),  
    PRIMARY KEY(제품번호),  
    CHECK (재고량 >= 0 AND 재고량 <=10000)  
);

product  
product\_num  
product\_name  
stock  
money  
pro\_company

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그느만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 02 SQL을 이용한 데이터 정의

## 주문 테이블 생성을 위한 CREATE TABLE 문 작성 예

### 예제 7-3

주문 테이블은 주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자 속성으로 구성되고, 주문번호 속성이 기본키다. 주문고객 속성이 고객 테이블의 고객아이디 속성을 참조하는 외래키이고, 주문제품 속성이 제품 테이블의 제품번호 속성을 참조하는 외래키가 되도록 주문 테이블을 생성해보자.

*request*

#### ▶▶ CREATE TABLE 주문 (

*order\_num* 주문번호 CHAR(3) NOT NULL,

*order\_customer* 주문고객 VARCHAR(20),

*order\_pro* 주문제품 CHAR(3),

*amount* 수량 INT,

*address* 배송지 VARCHAR(30),

*order\_date* 주문일자 DATE,

MS SQL에서는 주문일자 속성의 데이터 타입을 DATETIME로 지정함

PRIMARY KEY(주문번호),

FOREIGN KEY(주문고객) REFERENCES 고객(고객아이디),

FOREIGN KEY(주문제품) REFERENCES 제품(제품번호)

주문 테이블

| 주문번호 | 주문고객   | 주문제품 | 수량 | 배송지      | 주문일자       |
|------|--------|------|----|----------|------------|
| o01  | apple  | p03  | 10 | 서울시 마포구  | 2022-01-01 |
| o02  | melon  | p01  | 5  | 인천시 계양구  | 2022-01-10 |
| o03  | banana | p06  | 45 | 경기도 부천시  | 2022-01-11 |
| o04  | carrot | p02  | 8  | 부산시 금정구  | 2022-02-01 |
| o05  | melon  | p06  | 36 | 경기도 용인시  | 2022-02-20 |
| o06  | banana | p01  | 19 | 충청북도 보은군 | 2022-03-02 |
| o07  | apple  | p03  | 22 | 서울시 영등포구 | 2022-03-15 |
| o08  | pear   | p02  | 50 | 강원도 춘천시  | 2022-04-10 |
| o09  | banana | p04  | 15 | 전라남도 목포시 | 2022-04-11 |
| o10  | carrot | p03  | 20 | 경기도 안양시  | 2022-05-22 |

그림 7-2 질의에 사용할 판매 데이터베이스 : 고객, 제품, 주문 테이블

# 02 SQL을 이용한 데이터 정의

## 배송업체 테이블 생성을 위한 CREATE TABLE 문 작성 예

예제 7-4

배송업체 테이블은 업체번호, 업체명, 주소, 전화번호 속성으로 구성되고 업체번호 속성이 기본키다. 배송업체 테이블을 생성해보자.

```
▶▶ CREATE TABLE 배송업체 (
    업체번호 CHAR(3) NOT NULL,
    업체명 VARCHAR(20),
    주소 VARCHAR(100),
    전화번호 VARCHAR(20),
    PRIMARY KEY(업체번호)
);
```

company — num  
company — name  
address  
phone

Company

불어넣기 우편지

desc 테이블명;  
↳ type, size, null, key  
Default value

## 02 SQL을 이용한 데이터 정의

### 테이블 변경 : ALTER TABLE 문

- 새로운 속성 **추가**

```
ALTER TABLE 테이블_이름  
ADD 속성_이름 데이터_타입 [NOT NULL] [DEFAULT 기본_값];
```

↓옵션 **있어도 가능 없어도 가능**

#### 예제 7-5

[예제 7-1]에서 생성한 고객 테이블에 가입날짜 속성을 추가해보자.

```
▶▶ ALTER TABLE 고객 ADD 가입날짜 DATE;
```

## 02 SQL을 이용한 데이터 정의

### 테이블 변경 : ALTER TABLE 문

- 기존 속성 **삭제**

```
ALTER TABLE 테이블_이름 DROP COLUMN 속성_이름;
```

- 만약, 삭제할 속성과 관련된 제약조건이 존재한다면?
  - › 속성 삭제가 수행되지 않음
  - › 관련된 제약조건을 먼저 삭제해야 함

#### 예제 7-6

[예제 7-5]에서 추가한 고객 테이블의 가입날짜 속성을 삭제해보자.

▶▶ ALTER TABLE 고객 DROP COLUMN 가입날짜;

## 02 SQL을 이용한 데이터 정의

### ■ 테이블 변경 : ALTER TABLE 문

- 새로운 제약조건의 추가

```
ALTER TABLE 테이블_이름 ADD CONSTRAINT 제약조건_이름 제약조건_내용;
```

#### 예제 7-7

고객 테이블에 20세 이상의 고객만 가입할 수 있다는 데이터 무결성 제약조건을 추가해보자.

▶▶ ALTER TABLE 고객 ADD CONSTRAINT CHK\_AGE CHECK(나이 >= 20);

## 02 SQL을 이용한 데이터 정의

### ■ 테이블 변경 : ALTER TABLE 문

- 기존 제약조건의 삭제

```
ALTER TABLE 테이블_이름 DROP CONSTRAINT 제약조건_이름;
```

#### 예제 7-8

[예제 7-7]에서 추가한 고객 테이블에 20세 이상의 고객만 가입할 수 있다는 데이터 무결성 제약조건을 삭제해보자.

```
▶▶ ALTER TABLE 고객 DROP CONSTRAINT CHK_AGE;
```

## 02 SQL을 이용한 데이터 정의

### ■ 테이블 삭제 : DROP TABLE 문

```
DROP TABLE 테이블_이름;
```

- 만약, 삭제할 테이블을 참조하는 테이블이 있다면?
  - 테이블 삭제가 수행되지 않음
  - 관련된 외래키 제약조건을 먼저 삭제해야 함

#### 예제 7-9

배송업체 테이블을 삭제해보자.

▶▶ DROP TABLE 배송업체;

# 03 SQL을 이용한 데이터 조작

## SQL의 데이터 조작 기능

- 데이터 검색, 새로운 데이터 삽입, 데이터 수정, 데이터 삭제

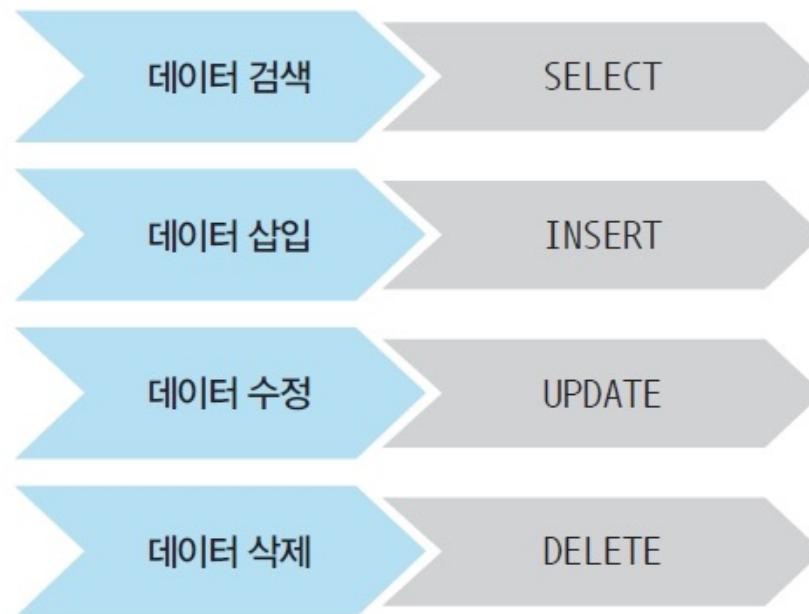


그림 7-5 SQL의 데이터 조작 기능

# 03 SQL을 이용한 데이터 조작

## 예제에서 사용할 판매 데이터베이스 : 고객 테이블

고객 테이블

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| peach  | 오형준  | NULL | silver | 의사  | 300  |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |

# 03 SQL을 이용한 데이터 조작

## 예제에서 사용할 판매 데이터베이스 : 제품 테이블

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 예제에서 사용할 판매 데이터베이스 : 주문 테이블

주문 테이블

| 주문번호 | 주문고객   | 주문제품 | 수량 | 배송지      | 주문일자       |
|------|--------|------|----|----------|------------|
| o01  | apple  | p03  | 10 | 서울시 마포구  | 2022-01-01 |
| o02  | melon  | p01  | 5  | 인천시 계양구  | 2022-01-10 |
| o03  | banana | p06  | 45 | 경기도 부천시  | 2022-01-11 |
| o04  | carrot | p02  | 8  | 부산시 금정구  | 2022-02-01 |
| o05  | melon  | p06  | 36 | 경기도 용인시  | 2022-02-20 |
| o06  | banana | p01  | 19 | 충청북도 보은군 | 2022-03-02 |
| o07  | apple  | p03  | 22 | 서울시 영등포구 | 2022-03-15 |
| o08  | pear   | p02  | 50 | 강원도 춘천시  | 2022-04-10 |
| o09  | banana | p04  | 15 | 전라남도 목포시 | 2022-04-11 |
| o10  | carrot | p03  | 20 | 경기도 안양시  | 2022-05-22 |

그림 7-6 데이터 조작 예제에서 사용하는 판매 데이터베이스 : 고객, 제품, 주문 테이블

# 03 SQL을 이용한 데이터 조작

## 데이터 삽입 : INSERT 문

- 데이터 직접 삽입

```
INSERT  
INTO 테이블_이름[(속성_리스트)]  
VALUES (속성값_리스트);
```

- INTO 키워드와 함께 튜플을 삽입할 테이블의 이름과 속성의 이름을 나열
  - › 속성 리스트를 생략하면 테이블을 정의할 때 지정한 속성의 순서대로 값이 삽입됨
- VALUES 키워드와 함께 삽입할 속성 값을 나열
- INTO 절의 속성 이름과 VALUES 절의 속성 값은 순서대로 일대일 대응되어야 함

# 03 SQL을 이용한 데이터 조작

## 데이터 삽입 : INSERT 문

### ■ 데이터 직접 삽입

예제 7-47

판매 데이터베이스의 고객 테이블에 고객아이디가 strawberry, 고객이름이 최유경, 나이가 30세, 등급이 vip, 직업이 공무원, 적립금이 100원인 새로운 고객의 정보를 삽입해보자. 그런 다음 고객 테이블에 있는 모든 내용을 검색하여 삽입된 새로운 투플을 확인해보자.

#### ▶▶ INSERT

```
INTO      고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
VALUES    ('strawberry', '최유경', 30, 'vip', '공무원', 100);
```

SELECT \* FROM 고객;

결과 테이블

| 고객아이디        | 고객이름  | 나이 | 등급    | 직업    | 적립금 |
|--------------|-------|----|-------|-------|-----|
| 'strawberry' | '최유경' | 30 | 'vip' | '공무원' | 100 |

|   | 고객아이디      | 고객이름 | 나이     | 등급     | 직업  | 적립금  |
|---|------------|------|--------|--------|-----|------|
| 1 | apple      | 정소화  | 20     | gold   | 학생  | 1000 |
| 2 | banana     | 김선우  | 25     | vip    | 간호사 | 2500 |
| 3 | carrot     | 고명석  | 28     | gold   | 교사  | 4500 |
| 4 | orange     | 김용욱  | 22     | silver | 학생  | 0    |
| 5 | melon      | 성원용  | 35     | gold   | 회사원 | 5000 |
| 6 | peach      | 오형준  | (null) | silver | 의사  | 300  |
| 7 | pear       | 채광주  | 31     | silver | 회사원 | 500  |
| 8 | strawberry | 최유경  | 30     | vip    | 공무원 | 100  |

# 03 SQL을 이용한 데이터 조작

## 데이터 삽입 : INSERT 문

- 데이터 직접 삽입

```
INSERT  
INTO    고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)  
VALUES ('strawberry', '최유경', 30, 'vip', '공무원', 100);
```

=

```
INSERT  
INTO    고객  
VALUES ('strawberry', '최유경', 30, 'vip', '공무원', 100);
```

# 03 SQL을 이용한 데이터 조작

## 데이터 삽입 : INSERT 문

### ■ 데이터 직접 삽입

예제 7-48

판매 데이터베이스의 고객 테이블에 고객아이디가 tomato, 고객이름이 정은심, 나이가 36 세, 등급이 gold, 적립금은 4,000원, 직업은 아직 모르는 새로운 고객의 정보를 삽입해보자. 그런 다음 고객 테이블에 있는 모든 내용을 검색하여, 삽입된 정은심 고객의 직업 속성이 널 값인지 확인해보자.

#### ▶▶ INSERT

```
INTO      고객(고객아이디, 고객이름, 나이, 등급, 적립금)  
VALUES    ('tomato', '정은심', 36, 'gold', 4000);
```

```
SELECT * FROM 고객;
```

결과 테이블

|   | 고객아이디      | 고객이름 | 나이     | 등급     | 직업     | 적립금  |
|---|------------|------|--------|--------|--------|------|
| 1 | apple      | 정소화  | 20     | gold   | 학생     | 1000 |
| 2 | banana     | 김선우  | 25     | vip    | 간호사    | 2500 |
| 3 | carrot     | 고명석  | 28     | gold   | 교사     | 4500 |
| 4 | orange     | 김용욱  | 22     | silver | 학생     | 0    |
| 5 | melon      | 성원용  | 35     | gold   | 회사원    | 5000 |
| 6 | peach      | 오형준  | (null) | silver | 의사     | 300  |
| 7 | pear       | 채광주  | 31     | silver | 회사원    | 500  |
| 8 | strawberry | 최유경  | 30     | vip    | 공무원    | 100  |
| 9 | tomato     | 정은심  | 36     | gold   | (null) | 4000 |

직업 속성에 널 값을 삽입

# 03 SQL을 이용한 데이터 조작

## 데이터 삽입 : INSERT 문

- 데이터 직접 삽입

```
INSERT
```

```
INTO 고객(고객아이디, 고객이름, 나이, 등급, 적립금)  
VALUES ('tomato', '정은심', 36, 'gold', 4000);
```

```
==
```

```
INSERT
```

```
INTO 고객
```

```
VALUES ('tomato', '정은심', 36, 'gold', NULL, 4000);
```

# 03 SQL을 이용한 데이터 조작

## 데이터 삽입 : INSERT 문

- 부속 질의문을 이용한 데이터 삽입
  - SELECT 문을 이용해 다른 테이블에서 검색한 데이터를 삽입

```
INSERT  
INTO 테이블_이름[(속성_리스트)]  
SELECT 문;
```

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그늘반두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

### 예) INSERT

```
INTO    한빛제품(제품명, 재고량, 단가)  
SELECT  제품명, 재고량, 단가  
FROM    제품  
WHERE   제조업체 = '한빛제과';
```

한빛제과에서 제조한 제품의 제품명,  
재고량, 단가를 제품 테이블에서 검색하여  
한빛제품 테이블에 삽입

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### ■ 기본 검색

- SELECT 키워드와 함께 검색하고 싶은 속성의 이름을 나열
- FROM 키워드와 함께 검색하고 싶은 속성이 있는 테이블의 이름을 나열
- 검색 결과는 테이블 형태로 반환됨

```
SELECT [ ALL | DISTINCT ] 속성_리스트  
FROM    테이블_리스트;
```

- ALL
  - › 결과 테이블이 투플의 중복을 허용하도록 지정, 생략 가능
- DISTINCT
  - › 결과 테이블이 투플의 중복을 허용하지 않도록 지정

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 기본 검색

### 예제 7-10

고객 테이블에서 고객아이디, 고객이름, 등급 속성을 검색해보자.

```
▶▶ SELECT 고객아이디, 고객이름, 등급  
      FROM 고객;
```

결과 테이블

|   | 고객아이디  | 고객이름 | 등급     |
|---|--------|------|--------|
| 1 | apple  | 정소화  | gold   |
| 2 | banana | 김선우  | vip    |
| 3 | carrot | 고명석  | gold   |
| 4 | orange | 김용욱  | silver |
| 5 | melon  | 성원용  | gold   |
| 6 | peach  | 오형준  | silver |
| 7 | pear   | 채광주  | silver |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### ▪ 기본 검색

예제 7-11

고객 테이블에 존재하는 모든 속성을 검색해보자.

▶▶ SELECT 고객아이디, 고객이름, 나이, 등급, 직업, 적립금  
FROM 고객;

고객 테이블

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| peach  | 오형준  | NULL | silver | 의사  | 300  |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |

결과 테이블

|   | 고객아이디  | 고객이름 | 나이     | 등급     | 직업  | 적립금  |
|---|--------|------|--------|--------|-----|------|
| 1 | apple  | 정소화  | 20     | gold   | 학생  | 1000 |
| 2 | banana | 김선우  | 25     | vip    | 간호사 | 2500 |
| 3 | carrot | 고명석  | 28     | gold   | 교사  | 4500 |
| 4 | orange | 김용욱  | 22     | silver | 학생  | 0    |
| 5 | melon  | 성원용  | 35     | gold   | 회사원 | 5000 |
| 6 | peach  | 오형준  | (null) | silver | 의사  | 300  |
| 7 | pear   | 채광주  | 31     | silver | 회사원 | 500  |

# 03 SQL을 이용한 데이터 조작

모든 속성을 검색할 때는  
모든 속성의 이름을 나열하지 않고  
\* 사용 가능

## 데이터 검색 : SELECT 문

- 기본 검색

### 예제 7-12

고객 테이블에 존재하는 모든 속성을 검색해보자.

▶▶ SELECT \*  
FROM 고객;

결과 테이블

|   | 고객아이디  | 고객이름 | 나이     | 등급     | 직업  | 적립금  |
|---|--------|------|--------|--------|-----|------|
| 1 | apple  | 정소화  | 20     | gold   | 학생  | 1000 |
| 2 | banana | 김선우  | 25     | vip    | 간호사 | 2500 |
| 3 | carrot | 고명석  | 28     | gold   | 교사  | 4500 |
| 4 | orange | 김용욱  | 22     | silver | 학생  | 0    |
| 5 | melon  | 성원용  | 35     | gold   | 회사원 | 5000 |
| 6 | peach  | 오형준  | (null) | silver | 의사  | 300  |
| 7 | pear   | 채광주  | 31     | silver | 회사원 | 500  |

고객 테이블

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| peach  | 오형준  | NULL | silver | 의사  | 300  |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 기본 검색

### 예제 7-13

제품 테이블에서 제조업체를 검색해보자.

```
▶▶ SELECT 제조업체  
      FROM 제품;
```

결과 테이블

|   | 제조업체 |
|---|------|
| 1 | 대한식품 |
| 2 | 민국푸드 |
| 3 | 한빛제과 |
| 4 | 한빛제과 |
| 5 | 대한식품 |
| 6 | 민국푸드 |
| 7 | 한빛제과 |

결과 테이블에서 제조업체가 중복됨

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 기본 검색

### 예제 7-14

제품 테이블에서 제조업체를 검색하되, ALL 키워드를 사용해보자.

▶▶ SELECT ALL 제조업체  
FROM 제품;

결과 테이블에서 제조업체가 중복됨

결과 테이블

|   | 제조업체 |
|---|------|
| 1 | 대한식품 |
| 2 | 민국푸드 |
| 3 | 한빛제과 |
| 4 | 한빛제과 |
| 5 | 대한식품 |
| 6 | 민국푸드 |
| 7 | 한빛제과 |

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### 예제 7-15

제품 테이블에서 제조업체 속성을 중복 없이 검색해보자.

```
▶▶ SELECT DISTINCT 제조업체  
      FROM 제품;
```

결과 테이블

|   | 제조업체 |
|---|------|
| 1 | 대한식품 |
| 2 | 민국푸드 |
| 3 | 한빛제과 |

결과 테이블에서 제조업체가  
한 번씩만 나타남

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### ■ 기본 검색

- AS 키워드를 이용해 결과 테이블에서 속성의 이름을 바꾸어 출력 가능
  - › 새로운 이름에 공백이 포함되어 있으면 큰따옴표나 작은따옴표로 묶어야 함
    - 오라클에서는 큰따옴표, MS SQL 서버에서는 작은따옴표 사용
  - › AS 키워드는 생략 가능

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### 예제 7-16

제품 테이블에서 제품명과 단가를 검색하되, 단가를 가격이라는 새 이름으로 출력해보자.

▶▶ SELECT 제품명, 단가 AS 가격  
FROM 제품;

결과 테이블

|   | 제품명   | 가격   |
|---|-------|------|
| 1 | 그냥만두  | 4500 |
| 2 | 매운쫄면  | 5500 |
| 3 | 쿵떡파이  | 2600 |
| 4 | 맛난초콜릿 | 2500 |
| 5 | 얼큰라면  | 1200 |
| 6 | 통통우동  | 1550 |
| 7 | 달콤비스킷 | 1500 |

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 산술식을 이용한 검색
  - SELECT 키워드와 함께 산술식 제시
    - › 산술식: 속성의 이름과 +, -, \*, / 등의 산술 연산자와 상수로 구성
  - 속성의 값이 실제로 변경되는 것은 아니고 결과 테이블에서만 계산된 결과 값이 출력됨

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 산술식을 이용한 검색

### 예제 7-17

제품 테이블에서 제품명과 단가 속성을 검색하되, 단가에 500원을 더해 ‘조정 단가’라는 새 이름으로 출력해보자.

▶▶ SELECT      제품명, 단가 + 500 AS "조정 단가"  
                FROM      제품;

결과 테이블

|   | 제품명   | 조정 단가 |
|---|-------|-------|
| 1 | 그냥만두  | 5000  |
| 2 | 매운쫄면  | 6000  |
| 3 | 쿵떡파이  | 3100  |
| 4 | 맛난초콜릿 | 3000  |
| 5 | 얼큰라면  | 1700  |
| 6 | 통통우동  | 2050  |
| 7 | 달콤비스킷 | 2000  |

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 조건 검색
  - 조건을 만족하는 데이터만 검색

```
SELECT [ ALL | DISTINCT ] 속성_리스트  
FROM   테이블_리스트  
[ WHERE 조건 ];
```

- WHERE 키워드와 함께 비교 연산자와 논리 연산자를 이용한 검색 조건 제시
  - › 숫자뿐만 아니라 문자나 날짜 값을 비교하는 것도 가능
    - 예) 'A' < 'C'
    - 예) '2019-12-01' < '2019-12-02'
  - › 조건에서 문자나 날짜 같은 작은따옴표로 묶어서 표현

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

표 7-2 비교 연산자

| 연산자 | 의미      |
|-----|---------|
| =   | 같다.     |
| < > | 다르다.    |
| <   | 작다.     |
| >   | 크다.     |
| <=  | 작거나 같다. |
| >=  | 크거나 같다. |

표 7-3 논리 연산자

| 연산자 | 의미                       |
|-----|--------------------------|
| AND | 모든 조건을 만족해야 검색한다.        |
| OR  | 여러 조건 중 한 가지만 만족해도 검색한다. |
| NOT | 조건을 만족하지 않는 것만 검색한다.     |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 조건 검색

### 예제 7-18

제품 테이블에서 한빛제과가 제조한 제품의 제품명, 재고량, 단가를 검색해보자.

```
▶▶ SELECT 제품명, 재고량, 단가  
      FROM 제품  
     WHERE 제조업체 = '한빛제과';
```

결과 테이블

|   | 제품명   | 재고량  | 단가   |
|---|-------|------|------|
| 1 | 쿵떡파이  | 3600 | 2600 |
| 2 | 만난초콜릿 | 1250 | 2500 |
| 3 | 달콤비스킷 | 1650 | 1500 |

SELECT = '한빛제과' (제조업체)  
제조업체 = '한빛제과' (제조업체)

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 만국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 만난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 만국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 조건 검색

예제 7-19

$\sum_{\text{고객} = \text{'apple'}} \wedge \text{수량} \geq 15 \left( \prod_{\text{제품}, \text{수량}, \text{주문일자}} (\text{주문}) \right)$

주문 테이블에서 apple 고객이 15개 이상 주문한 주문제품, 수량, 주문일자를 검색해보자.

▶▶ SELECT 주문제품, 수량, 주문일자  
FROM 주문  
WHERE 주문고객 = 'apple' AND 수량 >= 15;

결과 테이블

|   | 주문제품 | 수량 | 주문일자     |
|---|------|----|----------|
| 1 | p03  | 22 | 22/03/15 |

|     | 배송지      | 주문일자       |
|-----|----------|------------|
| 007 | 서울시 마포구  | 2019-01-01 |
| 008 | 인천시 계양구  | 2019-01-10 |
| 009 | 경기도 부천시  | 2019-01-11 |
| 010 | 부산시 금정구  | 2019-02-01 |
| 011 | 경기도 용인시  | 2019-02-20 |
| 012 | 충청북도 보은군 | 2019-03-02 |
| 013 | 서울시 영등포구 | 2019-03-15 |
| 014 | 강원도 춘천시  | 2019-04-10 |
| 015 | 전라남도 목포시 | 2019-04-11 |
| 016 | 경기도 안양시  | 2019-05-22 |

그림 7-6 데이터 조작 예제에서 사용하는 판매 데이터베이스 : 고객, 제품, 주문 테이블

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### ■ 조건 검색

#### 예제 7-20

주문 테이블에서 apple 고객이 주문했거나 15개 이상 주문된 제품의 주문제품, 수량, 주문일자, 주문고객을 검색해보자.

```
▶▶ SELECT 주문제품, 수량, 주문일자, 주문고객  
      FROM 주문  
     WHERE 주문고객 = 'apple' OR 수량 >= 15;
```

결과 테이블

|   | 주문제품 | 수량 | 주문일자     | 주문고객   |
|---|------|----|----------|--------|
| 1 | p03  | 10 | 22/01/01 | apple  |
| 2 | p06  | 45 | 22/01/11 | banana |
| 3 | p06  | 36 | 22/02/20 | melon  |
| 4 | p01  | 19 | 22/03/02 | banana |
| 5 | p03  | 22 | 22/03/15 | apple  |
| 6 | p02  | 50 | 22/04/10 | pear   |
| 7 | p04  | 15 | 22/04/11 | banana |
| 8 | p03  | 20 | 22/05/22 | carrot |

| 배송지      | 주문일자       |
|----------|------------|
| 서울시 마포구  | 2019-01-01 |
| 인천시 계양구  | 2019-01-10 |
| 경기도 부천시  | 2019-01-11 |
| 부산시 금정구  | 2019-02-01 |
| 경기도 용인시  | 2019-02-20 |
| 충청북도 보은군 | 2019-03-02 |
| 서울시 영등포구 | 2019-03-15 |
| 강원도 춘천시  | 2019-04-10 |
| 전라남도 목포시 | 2019-04-11 |
| 경기도 안양시  | 2019-05-22 |

!객, 제품, 주문 테이블

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 조건 검색

### 예제 7-21

제품 테이블에서 단가가 2,000원 이상이면서 3,000원 이하인 제품의 제품명, 단가, 제조업체를 검색해보자.

▶▶ SELECT 제품명, 단가, 제조업체  
FROM 제품  
WHERE 단가 >= 2000 AND 단가 <= 3000;

결과 테이블

|   | 제품명   | 단가   | 제조업체 |
|---|-------|------|------|
| 1 | 쿵떡파이  | 2600 | 한빛제과 |
| 2 | 맛난초콜릿 | 2500 | 한빛제과 |

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- LIKE를 이용한 검색
  - LIKE 키워드를 이용해 부분적으로 일치하는 데이터를 검색
  - 문자열을 이용하는 조건에만 LIKE 키워드 사용 가능

표 7-4 LIKE 키워드와 함께 사용할 수 있는 기호

| 기호 | 설명                            |
|----|-------------------------------|
| %  | 0개 이상의 문자 (문자의 내용과 개수는 상관 없음) |
| -  | 1개의 문자 (문자의 내용은 상관 없음)        |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- LIKE를 이용한 검색

표 7-5 LIKE 키워드의 사용 예

| 사용 예           | 설명                                     |
|----------------|----------------------------------------|
| LIKE '데이터%'    | 데이터로 시작하는 문자열(데이터로 시작하기만 하면 길이는 상관 없음) |
| LIKE '%데이터'    | 데이터로 끝나는 문자열(데이터로 끝나기만 하면 길이는 상관 없음)   |
| LIKE '%데이터%'   | 데이터가 포함된 문자열                           |
| LIKE '데이터 ___' | 데이터로 시작하는 6자 길이의 문자열                   |
| LIKE '__한%'    | 세 번째 글자가 '한'인 문자열                      |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 조건 검색

### 예제 7-22

고객 테이블에서 성이 김 씨인 고객의 고객이름, 나이, 등급, 적립금을 검색해보자.

▶▶ SELECT 고객이름, 나이, 등급, 적립금

FROM 고객

WHERE 고객이름 LIKE '김%';

결과 테이블

|   | 고객이름 | 나이 | 등급     | 적립금  |
|---|------|----|--------|------|
| 1 | 김선우  | 25 | vip    | 2500 |
| 2 | 김용욱  | 22 | silver | 0    |

고객 테이블

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| peach  | 오형준  | NULL | silver | 의사  | 300  |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 조건 검색

### 예제 7-23

고객 테이블에서 고객아이디가 5자인 고객의 고객아이디, 고객이름, 등급을 검색해보자.

▶▶ SELECT 고객아이디, 고객이름, 등급  
FROM 고객  
WHERE 고객아이디 LIKE '\_-----';

결과 테이블

|   | 고객아이디 | 고객이름 | 등급     |
|---|-------|------|--------|
| 1 | apple | 정소화  | gold   |
| 2 | melon | 성원용  | gold   |
| 3 | peach | 오형준  | silver |

고객 테이블

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| peach  | 오형준  | NULL | silver | 의사  | 300  |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### ▪ NULL을 이용한 검색

- IS NULL 키워드를 이용해 특정 속성의 값이 널 값인지를 비교
- IS NOT NULL 키워드를 이용해 특정 속성의 값이 널 값이 아닌지를 비교
- 검색 조건에서 널 값은 다른 값과 크기를 비교하면 결과가 모두 거짓이 됨

#### 예제 7-24

고객 테이블에서 나이가 아직 입력되지 않은 고객의 고객이름을 검색해보자.

```
▶▶ SELECT    고객이름  
      FROM     고객  
      WHERE    나이 IS NULL;
```

결과 테이블

|   | 고객이름 |
|---|------|
| 1 | 오형준  |

고객 테이블

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| peach  | 오형준  | NULL | silver | 의사  | 300  |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 조건 검색

### 예제 7-25

고객 테이블에서 나이가 이미 입력된 고객의 고객이름을 검색해보자.

▶▶ SELECT 고객이름  
FROM 고객  
WHERE 나이 IS NOT NULL;

결과 테이블

|   | 고객이름 |
|---|------|
| 1 | 정소화  |
| 2 | 김선우  |
| 3 | 고명석  |
| 4 | 김용욱  |
| 5 | 성원용  |
| 6 | 채광주  |

고객 테이블

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| peach  | 오형준  | NULL | silver | 의사  | 300  |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### ▪ 정렬 검색

- ORDER BY 키워드를 이용해 결과 테이블 내용을 사용자가 원하는 순서로 출력
- ORDER BY 키워드와 함께 정렬 기준이 되는 속성과 정렬 방식을 지정
  - › 오름차순(기본): ASC / 내림차순: DESC
  - › 널 값은 오름차순에서는 맨 마지막에 출력되고, 내림차순에서는 맨 먼저 출력됨
  - › 여러 기준에 따라 정렬하려면 정렬 기준이 되는 속성을 차례대로 제시

```
SELECT [ ALL | DISTINCT ] 속성_리스트  
FROM    테이블_리스트  
[ WHERE 조건 ]  
[ ORDER BY 속성_리스트 [ ASC | DESC ] ];
```

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### ▪ 조건 검색

#### 예제 7-26

고객 테이블에서 고객이름, 등급, 나이를 검색하되, 나이를 기준으로 내림차순 정렬해보자.

```
▶▶ SELECT 고객이름, 등급, 나이  
      FROM 고객  
     ORDER BY 나이 DESC;
```

결과 테이블

|   | 고객이름 | 등급     | 나이     |
|---|------|--------|--------|
| 1 | 오형준  | silver | (null) |
| 2 | 성원용  | gold   | 35     |
| 3 | 채광주  | silver | 31     |
| 4 | 고명석  | gold   | 28     |
| 5 | 김선우  | vip    | 25     |
| 6 | 김용욱  | silver | 22     |
| 7 | 정소화  | gold   | 20     |

고객 테이블

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| peach  | 오형준  | NULL | silver | 의사  | 300  |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### ■ 정렬 검색

#### 예제 7-27

주문 테이블에서 수량이 10개 이상인 주문의 주문고객, 주문제품, 수량, 주문일자를 검색해 보자. 단, 주문제품을 기준으로 오름차순 정렬하고, 동일 제품은 수량을 기준으로 내림차순 정렬해보자.

```
▶▶ SELECT 주문고객, 주문제품, 수량, 주문일자  
      FROM 주문  
     WHERE 수량 >= 10  
   ORDER BY 주문제품 ASC, 수량 DESC;
```

결과 테이블

|   | 주문고객   | 주문제품 | 수량 | 주문일자     |
|---|--------|------|----|----------|
| 1 | banana | p01  | 19 | 22/03/02 |
| 2 | pear   | p02  | 50 | 22/04/10 |
| 3 | apple  | p03  | 22 | 22/03/15 |
| 4 | carrot | p03  | 20 | 22/05/22 |
| 5 | apple  | p03  | 10 | 22/01/01 |
| 6 | banana | p04  | 15 | 22/04/11 |
| 7 | banana | p06  | 45 | 22/01/11 |
| 8 | melon  | p06  | 36 | 22/02/20 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

### ■ 집계 함수를 이용한 검색

- 특정 속성 값을 통계적으로 계산한 결과를 검색하기 위해 집계 함수를 이용
  - › 집계 함수(aggregate function)
    - 열 함수(column function)라고도 함
    - 개수, 합계, 평균, 최댓값, 최솟값의 계산 기능을 제공
- 집계 함수 사용 시 주의 사항
  - › 집계 함수는 널인 속성 값은 제외하고 계산함
  - › 집계 함수는 WHERE 절에서는 사용할 수 없고, SELECT 절이나 HAVING 절에서만 사용 가능

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

표 7-6 집계 함수

| 함수    | 의미        | 사용 가능한 속성의 타입 |
|-------|-----------|---------------|
| COUNT | 속성 값의 개수  | 모든 데이터        |
| MAX   | 속성 값의 최댓값 |               |
| MIN   | 속성 값의 최솟값 |               |
| SUM   | 속성 값의 합계  | 숫자 데이터        |
| AVG   | 속성 값의 평균  |               |

## 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- ## ■ 조건 검색

### 예제 7-28

제품 테이블에서 모든 제품의 단가 평균을 검색해보자.

▶ SELECT AVG(단가)  
FROM 제품; ex) where 단가 >= 2000

## 결과 테이블

|   |                                           |
|---|-------------------------------------------|
|   | AVG(단가)                                   |
| 1 | 2764.285714285714285714285714285714285714 |

## 제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 만국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 만국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

AVG(단가)

2764

그림 7-7 모든 제품의 평균 단가를 계산하는 과정 : 제품 테이블

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

예제 7-29

한빛제과에서 제조한 제품의 재고량 합계를 제품 테이블에서 검색해보자.

select이 있다

```
▶▶ SELECT SUM(재고량) AS "재고량 합계"  
      FROM 제품  
     WHERE 제조업체 = '한빛제과';
```

결과 테이블

|   | 재고량 합계 |
|---|--------|
| 1 | 6500   |

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 맛난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

예제 7-30

고객 테이블에 고객이 몇 명 등록되어 있는지 검색해보자.

▶▶ ① 고객아이디 속성을 이용해 계산하는 경우

```
SELECT COUNT(고객아이디) AS 고객수  
FROM 고객;
```

결과 테이블

|   | 고객수 |
|---|-----|
| 1 | 7   |

② 나이 속성을 이용해 계산하는 경우

```
SSELECT COUNT(나이) AS 고객수  
FROM 고객;
```

결과 테이블

|   | 고객수 |
|---|-----|
| 1 | 6   |

③ \*를 이용해 계산하는 경우

```
SELECT COUNT(*) AS 고객수  
FROM 고객;
```

결과 테이블

|   | 고객수 |
|---|-----|
| 1 | 7   |

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

널인 속성 값은 제외하고 개수 계산

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |
| peach  | 오형준  | NULL | silver | 의사  | 300  |

COUNT  
(고객아이디)

7

COUNT  
(나이)

6

그림 7-8 고객의 수를 계산하는 과정 : 고객 테이블

# 03 SQL을 이용한 데이터 조작

정확한 개수를 계산하기 위해서는  
보통 기본키 속성이나 \*를 주로 이용

## 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

| 고객아이디  | 고객이름 | 나이   | 등급     | 직업  | 적립금  |
|--------|------|------|--------|-----|------|
| apple  | 정소화  | 20   | gold   | 학생  | 1000 |
| banana | 김선우  | 25   | vip    | 간호사 | 2500 |
| carrot | 고명석  | 28   | gold   | 교사  | 4500 |
| orange | 김용욱  | 22   | silver | 학생  | 0    |
| melon  | 성원용  | 35   | gold   | 회사원 | 5000 |
| pear   | 채광주  | 31   | silver | 회사원 | 500  |
| peach  | 오형준  | NULL | silver | 의사  | 300  |

그림 7-9 COUNT(\*)로 개수를 계산하는 과정 : 고객 테이블

# 03 SQL을 이용한 데이터 조작

## 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

예제 7-31

제품 테이블에서 제조업체의 수를 검색해보자.

select  
from  
where

▶▶ SELECT COUNT(DISTINCT 제조업체) AS "제조업체 수"  
FROM 제품;

제품 테이블

| 제품번호 | 제품명   | 재고량  | 단가   | 제조업체 |
|------|-------|------|------|------|
| p01  | 그냥만두  | 5000 | 4500 | 대한식품 |
| p02  | 매운쫄면  | 2500 | 5500 | 민국푸드 |
| p03  | 쿵떡파이  | 3600 | 2600 | 한빛제과 |
| p04  | 만난초콜릿 | 1250 | 2500 | 한빛제과 |
| p05  | 얼큰라면  | 2200 | 1200 | 대한식품 |
| p06  | 통통우동  | 1000 | 1550 | 민국푸드 |
| p07  | 달콤비스킷 | 1650 | 1500 | 한빛제과 |

결과 테이블

|   | 제조업체 수 |
|---|--------|
| 1 | 3      |

DISTINCT 키워드를 이용해 중복을 없애고 서로 다른 제조업체의 개수만 계산

select B.C

from R

where A > 100

ORDER BY B Asc  
Dsc

R

| A   | B   | C   | D   |
|-----|-----|-----|-----|
| 100 | 100 | 100 | 100 |
| 100 | 100 | 100 | 100 |

A > 100