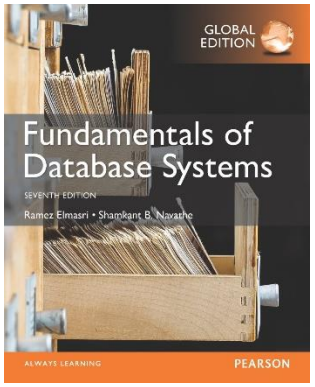
데이터베이스 시스템

충북대학교 컴퓨터과학과

이종윤 교수님

제7장:

더 많은 SQL: 복잡한 쿼리, 트리거, 뷰 및 스키마 수정



7장 개요

- 더 복잡한 SQL 검색 쿼리
- 지정하기의미론적 기음제약 주장과 행동이 트리거로 사용됨

- 조회수 (가상 테이블) SQL에서
- SQL의 스키마 수정

더 복잡한 SQL 검색 쿼리

더 복잡한 SQL 검색 쿼리

- **추가 기능을 사용하면 사용자는 데이터베이스에서 더 복잡한 검색을 지정할 수 있습니다.**
  - 중첩된 쿼리 ,조인된 테이블 , 그리고외부 조인 (FROM절에서) 집계 함수 , 그리고그룹화

NULL과 3값 논리를 포함하는 비교

- 의 의미널(NULL)
  - 알 수 없는 값
  - 사용할 수 없거나 보류된 가치
  - 해당되지 않는 속성

- 각 개인널(NULL)다른 모든 것과 다르다고 여겨지는 가치널(NULL)값

- SQL은 다음을 사용합니다.3-밸 유ed 논리 :
  - 참, 거짓,그리고알려지지 않은 ((아마도)

- NULL = NULL 비교 영형n은 피한다

NULL과 3값 논리를 포함한 비교(계속)

Table 7.1 Logical Connectives in Three-Valued Logic

(a)

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

(b)

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

(c)

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

NULL과 3값 논리를 포함한 비교(계속)

- SQL은 속성 값이 맞는지 확인하는 쿼리를 허용합니다. 널(NULL)
  - 이다또는NULL이 아닙니다

Query 18. Retrieve the names of all employees who do not have supervisors.

```
Q18:    SELECT    Fname, Lname
        FROM      EMPLOYEE
        WHERE     Super_ssn IS NULL;
```

NULL과 3값 논리를 포함한 비교(계속)

```
mysql> 현재_부서_직원 설명;
+-------------+---------+------+--------+---------+| 필
드                  | 유형 | Null | 키 | 기본값 | 추가 |
+-------------+----------+-------+--------+---------+|
직원 번호          | int(11) | 아니요        |          | 널(NULL)       |           |
| 부서번호         | char(4) | 아니요        |          | 널(NULL)       |           |
| 시작_날짜 | 날짜 | 종료_        | 예        |          | 널(NULL)       |           |
날짜 | 날짜                          | 예        |          | 널(NULL)       |           |
+-------------+----------+-------+--------+---------+ 세
트당 4개 행(0.00초)

mysql> SELECT emp_no from current_dept_emp WHERE from_date 이다
널(NULL)
공집합(0.33초)

MySQL>
```

중첩 쿼리, 튜플 및 세트/멀티세트 비교

- **중첩된 쿼리**
  - 완벽한블록 내에서 select-from-where어디 다른 쿼리의 절
  - **외부 쿼리와 중첩된 하위 쿼리**
- **비교 연산자안에**
  - 가치를 비교합니다*다섯*값의 집합(또는 다중 집합)을 사용하여*다섯*
  - 평가합니다진실만약에*다섯*요소 중 하나입니다*다섯*

중첩된 쿼리(계속)

```
Q4A:    SELECT    DISTINCT Pnumber
        FROM      PROJECT
        WHERE     Pnumber IN
                  ( SELECT     Pnumber
                    FROM       PROJECT, DEPARTMENT, EMPLOYEE
                    WHERE      Dnum = Dnumber AND
                               Mgr_ssn = Ssn AND Lname = 'Smith' )
                  OR
                  Pnumber IN
                  ( SELECT     Pno
                    FROM       WORKS_ON, EMPLOYEE
                    WHERE      Essn = Ssn AND Lname = 'Smith' );
```

중첩된 쿼리(계속)

```
mysql> SELECT emp_no FROM dept_emp WHERE emp_no안에
(SELECT emp_no FROM current_dept_emp WHERE dept_no =
'd001') 제한 5;
+ - - - - - - - - -
+ | 직원 번호 |
+ - - - - - - - - - +
|     10017 |
|     10055 |
|     10058 |
|     10108 |
|     10108 |
+ - - - - - - - - - +
세트당 5개 행(0.29초)

MySQL>
```

중첩된 쿼리(계속)

- **튜플을 사용하세요비교에서의 값**
  - 괄호 안에 넣으세요.

```
SELECT    DISTINCT Essn
FROM      WORKS_ON
WHERE     (Pno, Hours) IN    ( SELECT    Pno, Hours
                               FROM      WORKS_ON
                               WHERE     Essn = '123456789' );
```

mysql> SELECT emp_no FROM dept_emp WHERE(사원번호, 부서번호)IN
(SELECT emp_no, dept_no FROM dept_emp WHERE dept_no = 'd001') limit
3;

```
+---------
+ | 직원 번호 |
+---------+
|    10017 |
|    10055 |
|    10058 |
+---------+
```

세트당 3개 행(0.00초)

MySQL>

중첩된 쿼리(계속)

- **다른 비교 연산자를 사용하여 단일 값을 비교합니다.*다섯***
  - =어느 (또는 =일부)연산자
    - 반품진실만약 값이*다섯*집합 내의 어떤 값과 같다*다섯*따라서 다음과 같습니다.안에
  - 결합 가능한 다른 연산자어느 (또는일부): >, >=, <, <=,그리고 <>
  - 모두:값은 중첩된 쿼리의 모든 값을 초과해야 합니다.

```
SELECT    Lname, Fname
FROM      EMPLOYEE
WHERE     Salary > ALL    ( SELECT    Salary
                            FROM      EMPLOYEE
                            WHERE     Dno = 5 );
```

중첩된 쿼리(계속)

```
mysql> 현재_부서_직원 설명;
+-------------+----------+------+---------+---------+| 필
드                | 유형 | Null | 키 | 기본값 | 추가 |
+-------------+----------+-------+---------+---------+| 직
원 번호           | int(11) | 아니요       |          | 널(NULL)       |           |
| 부서번호        | char(4) | 아니요       |          | 널(NULL)       |           |
| 시작_날짜 | 날짜 | 종료_       | 예       |          | 널(NULL)       |           |
날짜 | 날짜                       | 예       |          | 널(NULL)       |           |
+-------------+----------+-------+---------+---------+ 세
트당 4개 행(0.00초)

mysql> desc 급여;
+-------------+----------+------+---------+---------+| 필
드                | 유형 | Null | 키 | 기본값 | 추가 |
+-------------+----------+-------+---------+---------+| 직
원 번호           | int(11) | 아니요       | 기본 | 널              |           |
| 급여            | int(11) | 아니요       |          | 널(NULL)       |           |
| 시작_날짜 | 날짜 | 종료_       | 아니요   | 기본 | 널              |           |
날짜 | 날짜                       | 아니요   |          | 널(NULL)       |           |
+-------------+----------+-------+---------+---------+ 세
트당 4개 행(0.00초)

mysql> SELECT 급여 FROM 급여 WHERE급여 > 전체(SELECT S.salary from salaries S, current_dept_emp C
WHERE S.emp_no = C.emp_no AND C.dept_no = 'd001') limit 3;
+---------
+ | 급여 |
+---------+
| 145732 |
| 145215 |
| 148820 |
+---------+
```

중첩된 쿼리(계속)

- **잠재적인 오류를 방지하고 디모호함**
  - 튜플 변수 생성 (별칭) SQL 쿼리에서 참조되는 모든 테이블에 대해

Query 16. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

```
Q16:  SELECT   E.Fname, E.Lname
      FROM     EMPLOYEE AS E
      WHERE    E.Ssn IN    ( SELECT   D.Essn
                             FROM     DEPENDENT AS D
                             WHERE    E.Fname = D.Dependent_name
                             AND E.Sex = D.Sex );
```

중첩된 쿼리(계속)

mysql> SELECT 급여 FROM 급여 WHERE 급여 > ALL (SELECT S.급여 from급여 AS S, 현재 부서 직원 AS CWHERE S.emp_no = C.emp_no AND C.dept_no = 'd001') limit 5;

```
+ - - - - - - - - -
+ | 급여 |
+ - - - - - - - - +
| 145732 |
| 145215 |
| 148820 |
| 145300 |
| 149440 |
+ - - - - - - - - - +
세트당 5개 행(0.84초)

MySQL>
```

상관관계가 있는 중첩 쿼리

- ar에 대한 질의 이자형t를 사용하여 중첩 시간e = 또는 IN 비교 연산자는 colla가 될 수 있습니다. 피sed를 하나의 블록으로 통합 : 예를 들어, Q16은 다음과 같이 작성될 수 있습니다.

```
Q16:  SELECT   E.Fname, E.Lname
      FROM     EMPLOYEE AS E
      WHERE    E.Ssn IN   ( SELECT   D.Essn
                            FROM     DEPENDENT AS D
                            WHERE    E.Fname = D.Dependent_name
                                     AND E.Sex = D.Sex );
```

Q16A:	선택하다	E.Fname, E.Lname
	에서	직원은 E이고, 부양가족은 D입니다.
	어디	E.Ssn=D.Essn이고 E.Sex=D.Sex이고
		E.Fname=D.Dependent_name입니다.

- 상관관계가 있는 중첩 쿼리
  - 외부 쿼리의 각 튜플에 대해 한 번씩 평가됨

SQL에서 쿼리 상관관계를 위한 EXISTS 및 UNIQUE 함수

- **존재한다기능**
  - 상관관계가 있는 중첩 쿼리의 결과를 확인합니다.비어있거나 비어있지 않음 . 이들은 다음을 반환하는 부울 함수입니다.진실또는거짓결과.

- **존재한다그리고존재하지 않음**
  - 일반적으로상관관계가 있는 중첩 쿼리와 함께 사용됨

- **SQL 함수유니크(Q)**
  - 반품진실만약 있다면쿼리 Q의 결과에 중복된 튜플이 없습니다.

EXISTS의 사용

Q7: 부양가족이 한 명 이상 있는 관리자의 이름을 나열하세요.

Fname, Lname을 선택하세요
직원으로부터
어디**존재한다**(선택하다 *
종속에서
여기서 Ssn = Essn)그리고
존재한다(선택하다 *
부서에서
WHERE Ssn= Mgr_Ssn)

EXISTS의 사용

mysql> SELECT 급여 FROM 급여 WHERE존재한다(SELECT S.salary from salaries AS S, current_dept_emp AS C WHERE S.emp_no = C.emp_no AND C.dept_no = 'd001') limit 5;

```
+---------
+|급여|
+---------+|
    60117|
|   62102|
|   66074|
|   66596|
|   66961|
+---------+
5개 행 세트(0.30초)

MySQL>
```

존재하지 않음의 사용

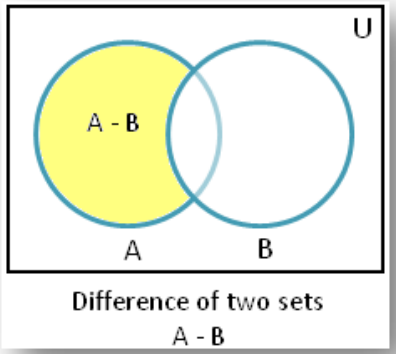
"을 달성하려면모두를 위해" (범용 양화사- 8장 참조) 효과, 우리는 SQL에서 이런 방식으로 이중 부정을 사용합니다:

질의: 근무하는 직원의 이름과 성을 나열하세요.모두 프로젝트 제어 영형Dno=5로 채워짐.

Fname, Lname을 선택하세요
직원으로부터
어디**존재하지 않음**((P번호 선택
프로젝트에서
WHERE Dno=5)

공집합

제외하고(선택 Pno
WORKS_ON에서
WHERE Ssn= ESsn));



위는 이중 부정과 동일합니다.: 부서 번호 5에서 관리하고 있는 프로젝트가 존재하지 않는 직원의 이름을 나열하세요.

존재하지 않음의 사용

mysql> SELECT 급여 FROM 급여 WHERE존재하지 않음(SELECT S.salary from salaries AS S, current_dept_emp AS C WHERE S.emp_no = C.emp_no AND C.dept_no = 'd001') limit 5;
공집합(0.30초)

MySQL>

mysql> SELECT 급여 FROM 급여 WHERE 급여안에 없음(SELECT S.salary from salaries AS S, current_dept_emp AS C WHERE S.emp_no = C.emp_no 및 C.dept_no = 'd001') 제한 5;

```
+---------
+| 급여 |
+---------+|
     81025 |
|    85112 |
|    69366 |
|    40006 |
|    43636 |
+---------+
```

MySQL에 주의하세요EXCEPT, MINUS 연산자를 지원하지 않습니다..

SQL에서 "모두를 위해"를 달성하기 위한 이중 부정

- **Q3B:**

선택하다 성, 성 이름
에서 직원
어디 **존재하지 않음**(**선택하다** *
에서 B에서 작동
어디 (B.Pno IN (P번호 선택
프로젝트에서
WHERE Dnum=5)
그리고
존재하지 않음(**선택하다** *
에서 C에 대한 작업
어디 C.Essn=Ssn
그리고 C.Pno=B.Pno)));

선택작업중(B)튜플의프노 5부서에서 관리하는 프로젝트입니다

만약 없다면작업중(C)같은 것을 가진 튜플프노그리고같은주민등록번호

위는 다음의 직접적인 렌더링입니다.:부서번호 5에서 관리하고 있는 프로젝트가 존재하지 않는 직원의 이름을 나열하세요.

SQL에서 속성의 명시적 집합 및 이름 변경

- **WHERE 절에서 명시적인 값 집합을 사용할 수 있습니다.**

질문 17:

선택하다	**DISTINCT 에센스**
에서	**작동중**
어디	**Pno IN (1, 2, 3);**

MySQL>선택별개의현재 부서 직원의 부서 번호

```
+ - - - - - - - - - - -
+ | 부서번호 |
+ - - - - - - - - - - -
+ |d005        |
| d007         |
| d004         |
| d003         |
| d008         |
| d006         |
| d009         |
| d001         |
| d002         |
+ - - - - - - - - - - +
```

SQL에서 속성의 명시적 집합 및 이름 변경

- **원하는 새 이름 뒤에 AS 한정자를 사용합니다.**
  - 쿼리 결과에 나타나는 모든 속성의 이름을 바꿉니다.

```
Q8A:    SELECT    E.Lname AS Employee_name, S.Lname AS Supervisor_name
        FROM      EMPLOYEE AS E, EMPLOYEE AS S
        WHERE     E.Super_ssn = S.Ssn;
```

mysql> SELECT C.dept_no처럼부서 번호 FROM 현재 부서 직원처럼C, 직원처럼E는 C.emp_no = E.emp_no limit 5인 경우입니다.

```
+ - - - - - - - - - - - - - - - - - - -
- + | 부서번호 |
+ - - - - - - - - - - - - - - - - - - -+
| d005                                 |
| d007                                 |
| d004                                 |
| d004                                 |
| d003                                 |
+ - - - - - - - - - - - - - - - - - - - +
5개 행 세트(0.29초)
```

SQL의 FROM 절에서 조인된 테이블 지정

- **조인된 테이블**
  - 사용자가 쿼리의 FROM 절에서 조인 작업의 결과로 생성된 테이블을 지정할 수 있도록 허용합니다.

- **Q1A의 FROM 절**
  - 단일 조인된 테이블을 포함합니다. JOIN이라고도 불릴 수 있습니다.내부 조인

```
Q1A:    SELECT    Fname, Lname, Address
        FROM      (EMPLOYEE JOIN DEPARTMENT ON Dno = Dnumber)
        WHERE     Dname = 'Research';
```

SQL에서 JOIN된 테이블의 다양한 유형

- **다양한 유형의 조인을 지정하세요**
  - 자연 조인
  - 다양한 유형의외부 조인 (왼쪽, 오른쪽, 전체)

- **두 관계 R과 S에 대한 NATURAL JOIN**
  - 조인 조건이 지정되지 않았습니다
  - 와 동일합니다암묵적 EQUIJOIN 각 쌍의 조건

    R과 S에서 같은 이름을 가진 속성

자연 조인

- NATURAL JOIN을 사용하여 한 관계의 속성 이름을 바꿔 다른 관계와 조인할 수 있습니다.

Q1B: 선택하다 성, 이름, 주소
에서 (직원 자연 가입
(부서를 부서로(이름, 날짜, 연락처,
미스데이트)))
어디 Dname='연구';

위의 작업은 다음과 같습니다.EMPLOYEE.Dno = DEPT.Dno를 암묵적 조인 조건으로 사용

자연 조인

mysql> SELECT C.dept_no AS department_number FROM current_dept_emp AS C 자연 조인직원 AS E 제한 5명

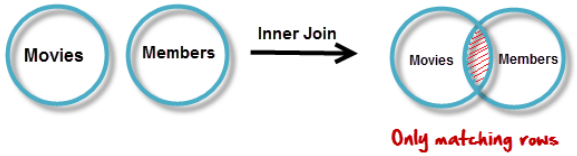
```
+ - - - - - - - - - - - - - - - - - - -
- + | 부서번호 |
+ - - - - - - - - - - - - - - - - - - +
| d005                    |
| d007                    |
| d004                    |
| d004                    |
| d003                    |
+ - - - - - - - - - - - - - - - - - - +
```

mysql> SELECT C.dept_no AS department_number FROM current_dept_emp AS C 자연 조인직원 AS E WHEREC.emp_no = E.emp_no제한 5;

```
+ - - - - - - - - - - - - - - - - - - -
- + | 부서번호 |
+ - - - - - - - - - - - - - - - - - - +
| d005                    |
| d007                    |
| d004                    |
| d004                    |
| d003                    |
+ - - - - - - - - - - - - - - - - - - +
```

INNER 및 OUTER 조인

- **INNER JOIN (OUTER JOIN 대비)**
  - 기본 유형 조인된 테이블의 조인
  - 튜플은 결과에 포함됩니다.일치하는 튜플이 존재하는 경우에만 다른 관계에서



- **왼쪽 외부 조인**
  - 왼쪽 테이블의 모든 튜플은 결과에 나타나야 합니다.
  - 일치하는 튜플이 없는 경우
    - NULL 값으로 패딩됨 오른쪽 테이블의 속성에 대해

- **오른쪽 외부 조인**
  - 오른쪽 테이블의 모든 튜플은 결과에 나타나야 합니다.
  - 일치하는 튜플이 없는 경우
    - 왼쪽 테이블의 속성에 대해 NULL 값으로 패딩됨

예: LEFT OUTER JOIN

E.Lname을 선택하세요**처럼**직원 이름
일.성 이름**처럼**감독자 이름
직원으로부터**처럼**이자형**왼쪽 외부 조인**직원**처럼**에스
ON E.Super_ssn = S.Ssn)

대체 구문:

E.Lname을 선택하세요**,**일.성 이름 **직원**
E에서 직원 S가 있는 곳E.슈퍼_SSN +=
S.SSN

예: LEFT OUTER JOIN

mysql> SELECT C.* FROM 현재_부서_직원 AS C왼쪽 조인 직원 AS E에
C.dept_no = 'd001' 제한 5;

```
+---------+-----------+-------------+---------
---+ | 직원 번호 | 부서 번호 | 시작일 | 종료일 |
+--------+---------+------------+------------+ |
    10017 | d001        | 1993-08-03 | 9999-01-01 | |
|   10055 | d001        | 1992-04-27 | 1995-07-22 | |
|   10058 | d001        | 1988-04-25 | 9999-01-01 | |
|   10108 | d001        | 1999-12-06 | 2001-10-20 | |
|   10140 | d001        | 1991-03-14 | 9999-01-01 |
+---------+-----------+-------------+ 세트당 5개
행(0.33초)
```

MySQL>

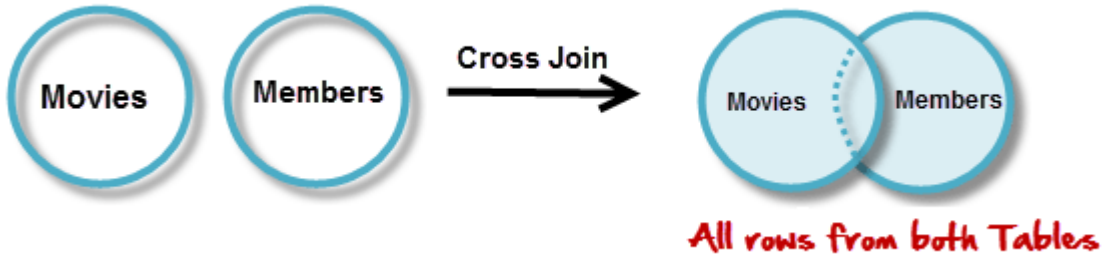
예: 오른쪽 외부 조인

```
mysql> SELECT E.* FROM 현재_부서_직원 AS C오른쪽 조인직원 AS E 에C.dept_no = 'd001'
제한 5;
+ - - - - - - - - - - + - - - - - - - - - - - - - - + - - - - - - - - - - - + - - - - - - - - - - + - - - - - - - - - - - + | 사
원 번호 | 생년월일 | 이름 | 성 | 성별 | 입사일 |
+ - - - - - - - - + - - - - - - - - - - + - - - - - - - - - - - + - - - - - - - - - - + - - - - - - - - + - - - - - - - - - - - + |
      10001 | 1953-09-02 | 조지 10001 |           | 파셀로          | 엠          | 1986-06-26 |
|     1953-09-02 | 조지 10001 |                  | 파셀로          | 엠          | 1986-06-26 |
|     1953-09-02 | 조지 10001 |                  | 파셀로          | 엠          | 1986-06-26 |
|     1953-09-02 | 조지 10001 |                  | 파셀로          | 엠          | 1986-06-26 |
|     1953-09-02 | 조지                           | 파셀로          | 엠          | 1986-06-26 |
+ - - - - - - - - - + - - - - - - - - - - - - - - + - - - - - - - - - - - + - - - - - - - - - - + - - - - - - - - - - - + 세트
당 5개 행(0.29초)

MySQL>
```

예:크로스 조인(데카르트 곱)

mysql> SELECT E.* FROM 현재_부서_직원 AS C크로스 조인직원 AS E 제한 5명

```
+----------+---------------+------------+-----------+------------+| 사
원 번호 | 생년월일 | 이름 | 성 | 성별 | 입사일 |
+--------+------------+------------+-----------+--------+------------+|
     10001 | 1953-09-02 | 조지 10002 |           | 파셀로          | 엠          | 1986-06-26 |
|    1964-06-02 | 베잘렐 10003 |           | 짐멜            | 여          | 1985-11-21 |
|    1959-12-03 | 파르토 10004 |           | 뱀포드          | 엠          | 1986-08-28 |
|    1954-05-01 | 크리스티안 10005 |        | 코블릭          | 엠          | 1986-12-01 |
|    1955-01-21 | 교이치                          | 말리니악        | 엠          | 1989-09-12 |
+----------+---------------+------------+-----------+------------+ 세트
당 5개 행(0.30초)

MySQL>
```



FROM 절의 Multiway JOIN

- FULL OUTER JOIN – LEFT 및 RIGHT OUTER JOIN의 경우 결과를 결합합니다.

- 다중 방식 조인에 대한 JOIN 사양을 중첩할 수 있습니다.

Q2A: 선택 Pnumber, Dnum, Lname, 주소, Bdate
에서 ((프로젝트가입하다부서ON Dnum=Dnumber)
JOIN직원ON Mgr_ssn=Ssn)

어디 위치='스태포드';

SQL의 집계 함수

- **요약하는 데 사용됨 아르 자형정보를 ize하다 여러 튜플에서 단일 튜플 요약으로**

- **내장된 집계 함수**
  - **세다,합집합,맥스,최소,** 그리고**평균**

- **그룹화**
  - 요약하기 전에 튜플의 하위 그룹을 만듭니다.

- **전체 그룹을 선택하려면가지고 있다절이 사용됩니다**

- **집계 함수는 다음에서 사용할 수 있습니다.선택하다절 또는 가지고 있다절**

집계 결과 이름 바꾸기

- 다음 쿼리는 계산된 값의 단일 행을 반환합니다. 직원테이블:

```
질문 19:    선택하다    SUM(급여), MAX(급여), MIN(급여), AVG(급여)
            에서        직원;
```

- 결과는 새로운 이름으로 표시될 수 있습니다.

```
Q19A:       선택하다    SUM(급여) AS Total_Sal, MAX(급여) AS
                        Highest_Sal, MIN(급여) AS Lowest_Sal, AVG(급
                        여) AS Average_Sal
            에서        직원;
```

집계 결과 이름 바꾸기

```
MySQL>선택SUM(급여), AVG(급여), MIN(급여), MAX(급여) 급여에서;

+----------------+---------------+----------------+ | SUM(급
여) | AVG(급여) | MIN(급여) | MAX(급여) |
+--------------+-------------+-------------+-------------+
| 181480757419 | 63810.7448 |                38623 |           158220 |
+----------------+---------------+----------------+ 세트당 1줄
(0.72초)

MySQL>
```

SQL의 집계 함수(계속)

- NULL 값 에이집계 함수가 삭제되면 다시 삭제됨 ~이다 적용된 특정 열에

Query 20. Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
Q20:    SELECT    SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
        FROM      (EMPLOYEE JOIN DEPARTMENT ON Dno = Dnumber)
        WHERE     Dname = 'Research';
```

Queries 21 and 22. Retrieve the total number of employees in the company (Q21) and the number of employees in the 'Research' department (Q22).

```
Q21:    SELECT    COUNT (*)
        FROM      EMPLOYEE;

Q22:    SELECT    COUNT (*)
        FROM      EMPLOYEE, DEPARTMENT
        WHERE     DNO = DNUMBER AND DNAME = 'Research';
```

SQL의 집계 함수(계속)

```
MySQL>선택세다(*)직원으로부터;
+ - - - - - - - - - - -
+ | 개수(*) |
+ - - - - - - - - - - +
| 300024 |
+ - - - - - - - - - - +
1세트당 1행(0.03초)

MySQL>
```

부울에 대한 집계 함수

- 일부그리고모두부울 값에 대한 함수로 적용될 수 있습니다.

- 일부true를 반환합니다.적어도 하나의 요소 컬렉션에 TRUE가 있습니다 (OR과 유사)

- 모두true를 반환합니다.모든 이자형요소 컬렉션에는 TRUE(AND와 유사)가 있습니다.

그룹화:그룹별로절

- **튜플의 하위 집합으로 관계 분할**
  - 기반으로**그룹화 속성(들)**
  - 각 그룹에 독립적으로 기능을 적용합니다.

- **그룹별로절**
  - 그룹화 속성을 지정합니다.

- **세다 (*)그룹의 행 수를 계산합니다**

GROUP BY의 예

- 그룹화 속성은 SELECT 절에 나타나야 합니다. Q24:

  SELECT Dno, COUNT (*), AVG (급여)
  FROM 직원
  그룹별로 아니요;

(a)

Fname	Minit	Lname	Ssn	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

mysql> SELECT 부서_번호, COUNT(*)
FROM 현재_부서_직원그룹별로 부서번호;

```
+----------+-----------+
| 부서 번호 | COUNT(*) |
+----------+-----------+
| d001     |     18426 |
| d002     |     15579 |
| d003     |     16071 |
| d004     |     66675 |
| d005     |     76958 |
| d006     |     18295 |
| d007     |     46922 |
| d008     |     19285 |
| d009     |     21813 |
+----------+-----------+
```

GROUP BY의 예

- 그룹화 속성에 NULL이 가능한 값으로 있는 경우 null 값에 대한 별도 그룹이 생성됩니다(예: 위 쿼리의 null Dno)

- GROUP BY는 JOIN의 결과에 적용될 수 있습니다.

질문 25:	선택하다	Pnumber, Pname, COUNT (*)
	에서	프로젝트, WORKS_ON
	어디	P번호=Pno
	그룹별로	번호, 이름;

GROUP BY의 예

```
mysql> SELECT 부서_번호, 성별, COUNT(*) FROM 현재_부서_직원 C, 직원 E WHERE
C.직원_번호 = E.직원_번호                                    그룹별로부서번호, 성별;
+----------+---------+--------
--+ | 부서번호 | 성별 | COUNT(*) |
+----------+--------+----------+
| d001     | 엠     |    11111 |
| d001     | 여     |     7315 |
| d002     | 엠     |     9273 |
| d002     | 여     |     6306 |
| d003     | 엠     |     9701 |
| d003     | 여     |     6370 |
| d004     | 엠     |    39885 |
| d004     | 여     |    26790 |
| d005     | 엠     |    46218 |
| d005     | 여     |    30740 |
| d006     | 엠     |    10921 |
| d006     | 여     |     7374 |
| d007     | 엠     |    28176 |
| d007     | 여     |    18746 |
| d008     | 엠     |    11587 |
| d008     | 여     |     7698 |
| d009     | 엠     |    13101 |
| d009     | 여     |     8712 |
+----------+--------+----------+
```

그룹화: GROUP BY와 HAVING 절(계속)

- **가지고 있다절**
  - 전체 그룹을 선택하거나 거부하기 위한 조건을 제공합니다.

- **질의 26. 각 프로젝트에 대해*2명 이상의 직원이 근무하는 경우*프로젝트 번호, 프로젝트 이름, 프로젝트에 참여한 직원 수를 검색합니다.**

질문 26:	선택하다	Pnumber, Pname, COUNT (*)
	에서	프로젝트, WORKS_ON
	어디	P번호=Pno
	그룹별로	P번호, P이름
	가지고 있다	개수(*) > 2;

그룹화: GROUP BY와 HAVING 절(계속)

(b)

Pname	Pnumber	...	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	...	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

These groups are not selected by the HAVING condition of Q26.

After applying the WHERE clause but before applying HAVING

Pname	Pnumber	...	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10		333445555	10	10.0
Computerization	10	...	999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the HAVING clause condition

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Result of Q26
(Pnumber not shown)

그룹화: GROUP BY와 HAVING 절(계속)

mysql> SELECT 부서_번호, 성별, COUNT(*) FROM 현재_부서_직원 C, 직원 E WHERE C.직원_번호 = E.직원_번호 부서번호, 성별로 그룹화
COUNT(*) > 10000인 경우

```
+-----------+---------+--------
-- + | 부서번호 | 성별 | COUNT(*) |
+---------+--------+----------+|
d001       | 엠       |    11111 |
| d004     | 엠       |    39885 |
| d004     | 여       |    26790 |
| d005     | 엠       |    46218 |
| d005     | 여       |    30740 |
| d006     | 엠       |    10921 |
| d007     | 엠       |    28176 |
| d007     | 여       |    18746 |
| d008     | 엠       |    11587 |
| d009     | 엠       |    13101 |
+-----------+---------+--------
--- + 10개 행 세트(1.07초)

MySQL>
```

WHERE 절과 HAVING 절 결합

- **다음 쿼리를 고려하십시오. 우리는 다음을 계산하고 싶습니다.*총*각 부서에서 급여가 4만 달러를 초과하는 직원 수(단, 직원이 5명 이상인 부서에만 해당).**

- **잘못된 쿼리:**

```
선택하다      아니,세다(*)
에서          직원
어디          급여 > 40000
그룹별로      디노
가지고 있다   세다(*) >5;
```

그것은 오직 다음과 같은 부서만을 선택할 것입니다.
각자 4만 달러 이상을 벌어들이는 직원이 5명 이상입니다.

규칙은 다음과 같습니다어디절은 개별 튜플 또는 결합된 튜플을 선택하기 위해 먼저 실행됩니다.가지고 있다이 절은 나중에 개별 튜플 그룹을 선택하는 데 적용됩니다.

WHERE 절과 HAVING 절 결합(계속)

쿼리의 정확한 사양:

- **참고사항:어디절은 튜플별로 적용되지만가지고 있다 튜플 그룹 전체에 적용됨**

```
Q28:    SELECT      Dno, COUNT (*)
        FROM        EMPLOYEE
        WHERE       Salary>40000 AND Dno IN
                    ( SELECT      Dno
                      FROM        EMPLOYEE
        GROUP BY    Dno
                      HAVING      COUNT (*) > 5)
        GROUP BY    Dno;
```

사용와 함께

- **그만큼와 함께절은 사용자에게 허용합니다특정 쿼리에서만 사용될 테이블을 정의하려면 (모든 SQL 구현에서 사용할 수 있는 것은 아닙니다.)**

- **편의를 위해 사용됨임시 "보기"를 생성하려면 그리고 그것을 쿼리에서 바로 사용합니다.**

- **단계별 쿼리를 더 직관적으로 볼 수 있는 방법을 제공합니다.**

에이공통 테이블 표현식(CTE)~이다이름이 지정된 임시 결과 세트 저것단일 진술의 범위 내에 존재합니다 그리고 그것은 나중에 해당 진술에서 여러 번 언급될 수도 있습니다.

WITH의 예

- **Q28을 수행하는 대체 접근 방식을 참조하세요.**

Q28': **와 함께**BIGBEPTS (Dno)**처럼**
(**선택하다**디노
에서직원
그룹별로디노
가지고 있다 **세다**(*) >5)

선택하다 아니,**세다**(*)
에서 직원
어디 급여>40000**그리고**나도 몰라**N**빅딥
그룹별로아니요;

CASE의 사용

- SQL에도 CASE 구조가 있습니다
- 특정 조건에 따라 값이 달라질 수 있을 때 사용됩니다.
- 값이 예상되는 SQL 쿼리의 모든 부분에서 사용할 수 있습니다.
- 튜플을 쿼리, 삽입 또는 업데이트할 때 적용 가능

CASE 사용 예

- 다음 예에서는 직원들이 부서마다 다른 인상을 받고 있음을 보여줍니다(업데이트 U6의 변형)

U6':

업데이트	직원		
세트	급여 =		
사례	**언제** Dno = 5	그 다음에	급여 + 2000
	언제 Dno = 4	그 다음에	급여 + 1500
	언제 Dno = 1	그 다음에	급여 + 3000

SQL 쿼리의 확장된 블록 구조

SELECT <attribute and function list>
FROM <table list>
[**WHERE** <condition>]
[**GROUP BY** <grouping attribute(s)>]
[**HAVING** <group condition>]
[**ORDER BY** <attribute list>];

제약 조건을 어설션으로 지정
그리고 트리거로서의 액션

제약 조건을 어설션으로 지정하고 작업을 트리거로 지정

- **의미적 제약: 다음은 EER 및 관계형 모델의 범위를 벗어납니다.**

- **주장 만들기**
  - 내장 관계형 모델 제약 조건의 범위를 벗어나는 추가 유형의 제약 조건 지정

- **트리거 생성**
  - 특정 이벤트 및 조건이 발생할 때 데이터베이스 시스템이 수행할 자동 작업을 지정합니다.

SQL에서 일반 제약 조건을 어설션으로 지정

- **주장 만들기**
  - 원하는 조건을 위반하는 튜플을 선택하는 쿼리를 지정하세요.
  - 간단한 범위를 넘어서는 경우에만 사용하세요.확인하다개별 속성 및 도메인에 적용됩니다.

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK ( NOT EXISTS ( SELECT     *
                     FROM       EMPLOYEE E, EMPLOYEE M,
                                DEPARTMENT D
                     WHERE      E.Salary>M.Salary
                                AND     E.Dno = D.Dnumber
                                AND     D.Mgr_ssn = M.Ssn ) );
```

SQL의 트리거 소개

- **트리거 생성성명**
  - 데이터베이스 모니터링에 사용됨

- **일반적인 트리거에는 "활성 데이터베이스"에 대한 규칙을 만드는 세 가지 구성 요소가 있습니다.**
  - **이벤트(들)**
  - **상태**
  - **행동**

트리거 사용

- **표준 구문을 사용한 예입니다.(참고: PostgreSQL과 같은 다른 SQL 구현에서는 다른 구문을 사용합니다.)**

R5:
트리거 생성급여 위반
삽입 또는 업데이트 전OF 급여, 관리자_ssn**에** 직원

각 행에 대해
언제 (새로운.급여 >(직원의 급여 선택
WHERE Ssn = NEW.Supervisor_Ssn))
INFORM_SUPERVISOR(신규.감독자.신규.신규.신규)

SQL의 뷰(가상 테이블)

- **SQL의 뷰 개념**
  - 다른 테이블에서 파생된 단일 테이블이라고 합니다.**테이블 정의**
  - ~로 간주됨반드시 채워지지 않은 가상 테이블

SQL의 뷰 사양

- **뷰 생성명령**
  - 테이블 이름, 속성 이름 목록 및 뷰의 내용을 지정하기 위한 쿼리를 제공합니다.
  - V1에서 속성은 기본 테이블의 이름을 유지합니다. V2에서 속성은 이름이 지정됩니다.

```
V1:   CREATE VIEW   WORKS_ON1
      AS SELECT     Fname, Lname, Pname, Hours
         FROM       EMPLOYEE, PROJECT, WORKS_ON
         WHERE      Ssn = Essn AND Pno = Pnumber;

V2:   CREATE VIEW   DEPT_INFO(Dept_name, No_of_emps, Total_sal)
      AS SELECT     Dname, COUNT (*), SUM (Salary)
         FROM       DEPARTMENT, EMPLOYEE
         WHERE      Dnumber = Dno
         GROUP BY   Dname;
```

SQL의 뷰 사양

```
V1:   CREATE VIEW     WORKS_ON1
      AS SELECT       Fname, Lname, Pname, Hours
          FROM        EMPLOYEE, PROJECT, WORKS_ON
          WHERE       Ssn = Essn AND Pno = Pnumber;

V2:   CREATE VIEW     DEPT_INFO(Dept_name, No_of_emps, Total_sal)
      AS SELECT       Dname, COUNT (*), SUM (Salary)
          FROM        DEPARTMENT, EMPLOYEE
          WHERE       Dnumber = Dno
          GROUP BY    Dname;
```

WORKS_ON1

Fname	Lname	Pname	Hours

DEPT_INFO

Dept_name	No_of_emps	Total_sal

SQL에서의 뷰 사양(계속)

- **뷰가 정의되면 SQL 쿼리는 FROM 절에서 뷰 관계를 사용할 수 있습니다.**

- **뷰는 항상 최신 상태로 유지됩니다**
  - DBMS의 책임이지 사용자의 책임이 아님

- **드롭 뷰명령**
  - 뷰를 처리하다

구현 보기, 업데이트 보기 및 인라인 보기

- **쿼리를 위한 뷰를 효율적으로 구현하는 복잡한 문제**

- **전략1: 쿼리 수정 접근 방식**
  - 필요에 따라 뷰를 계산합니다. 영구적으로 저장하지 마십시오.
  - 기본 테이블에 대한 쿼리로 뷰 쿼리 수정
  - 단점: 실행하는 데 시간이 많이 걸리는 복잡한 쿼리를 통해 정의된 뷰에는 비효율적입니다.

실체화 보기

- **전략 2: 구체화 보기**
  - 물리적으로 임시 뷰 테이블을 생성합니다. 뷰가 처음 쿼리될 때
  - 뷰에 대한 다른 쿼리가 뒤따를 것이라는 가정 하에 해당 테이블을 유지합니다.
  - 기본 테이블이 업데이트될 때 뷰 테이블을 자동으로 업데이트하기 위한 효율적인 전략이 필요합니다.

- **실체화된 뷰에 대한 증분 업데이트 전략**
  - DBMS는 실체화된 뷰 테이블에 삽입, 삭제 또는 수정해야 하는 새로운 튜플을 결정합니다.

실체화 보기(계속)

- **실체화를 처리하는 다양한 방법:**
  - **즉시 업데이트**전략은 기본 테이블이 변경되는 즉시 뷰를 업데이트합니다.
  - **게으른 업데이트**전략은 뷰 쿼리에 의해 필요할 때 뷰를 업데이트합니다.
  - **주기적 업데이트**전략은 뷰를 주기적으로 업데이트합니다(후자의 전략에서 뷰 쿼리는 최신이 아닌 결과를 얻을 수 있음).
    일반적으로 은행, 소매점 운영 등에서 사용됩니다.

업데이트 보기

- **집계 함수 없이 단일 테이블에 정의된 뷰에 대한 업데이트**

  – 기본 테이블에 대한 업데이트에 매핑할 수 있음 - 기본 키가 뷰에 보존된 경우 가능

- **집계 뷰에서는 업데이트가 허용되지 않습니다. 예:**

UV2:

업데이트	**부서 정보**
세트	**총_샐=100000**
어디	**Dname='연구';**

처리할 수 없습니다총_판매뷰 정의의 계산된 값입니다.

업데이트 보기

- **MySQL에서는 뷰를 쿼리할 수 있을 뿐만 아니라 업데이트할 수도 있습니다. 즉, INSERT 또는 UPDATE 문을 사용하여 업데이트 가능한 뷰를 통해 기본 테이블의 행을 삽입하거나 업데이트할 수 있습니다. 또한,DELETE 문을 사용하면 뷰를 통해 기본 테이블의 행을 제거할 수 있습니다.**

- **그러나 업데이트 가능한 뷰를 생성하려면뷰를 정의하는 SELECT 문에는 다음 요소가 포함되어서는 안 됩니다.**
  - MIN, MAX, SUM, AVG, COUNT와 같은 집계 함수.
  - 별개의
  - GROUP BY 절.
  - HAVING절.
  - UNION 또는 UNION ALL 절.
  - 왼쪽 조인 또는 외부 조인.
  - FROM 절에 나타난 테이블을 참조하는 SELECT 절이나 WHERE 절의 하위 쿼리입니다.
  - FROM 절에서 업데이트할 수 없는 뷰에 대한 참조입니다.
  - 리터럴 값만 참조합니다.
  - 기본 테이블의 모든 열에 대한 다중 참조.

업데이트 및 인라인 뷰 보기

- **조인을 포함하는 보기**
  - DBMS에서 어떤 업데이트가 의도된 것인지 판단하는 것이 불가능한 경우가 많습니다.

- **절체크 옵션 포함**
  - 뷰를 업데이트하려면 뷰 정의의 끝에 추가해야 하며 업데이트되는 튜플이 뷰에 남아 있도록 해야 합니다.

- **인라인 뷰**
  - 정의됨에서SQL 쿼리의 절(예: WITH 예제에서 사용된 것을 보았습니다)

권한 부여 메커니즘으로서의 뷰

- SQL 쿼리 권한 부여 문(GRANT 및 REVOKE)은 30장에서 자세히 설명합니다.

- 뷰는 권한이 없는 사용자로부터 특정 속성이나 튜플을 숨기는 데 사용할 수 있습니다.

- 예를 들어, 부서 5에 근무하는 직원의 정보만 볼 수 있는 사용자의 경우

  뷰에 접근하다부서5EMP:

뷰 생성	부서5EMP**처럼**
선택하다	*
에서	직원
어디	Dno = 5;

권한 부여 메커니즘으로서의 뷰

```
mysql> CREATE VIEW TEST AS SELECT * FROM dept_emp WHERE
dept_no = 'd001';
쿼리 확인, 영향을 받은 행 0개(0.69초)

MySQL>
mysql> SELECT * FROM TEST 제한 5;
+ - - - - - - - - - + - - - - - - - - - - - + - - - - - - - - - - - - - + - - - - - - - - -
- - - + | 직원 번호 | 부서 번호 | 시작일 | 종료일 |
+ - - - - - - - - + - - - - - - - - - + - - - - - - - - - - - + - - - - - - - - - - - + |
      10017 | d001           | 1993-08-03 | 9999-01-01 | |
|     10055 | d001           | 1992-04-27 | 1995-07-22 | |
|     10058 | d001           | 1988-04-25 | 9999-01-01 | |
|     10108 | d001           | 1999-12-06 | 2001-10-20 | |
|     10140 | d001           | 1991-03-14 | 9999-01-01 |
+ - - - - - - - - - + - - - - - - - - - - - + - - - - - - - - - - - - - + 세트당 5개
행(0.00초)

mysql> DROP VIEW TEST;
쿼리 확인, 영향을 받은 행 0개(0.00초)
```

SQL의 스키마 변경 문

SQL의 스키마 변경 문

- **스키마 진화 명령**
  - DBA는 데이터베이스가 작동하는 동안 스키마를 변경하려고 할 수 있습니다.
  - 데이터베이스 스키마를 다시 컴파일할 필요가 없습니다.

DROP 명령

- **떨어지다명령**
  - 테이블, 도메인 또는 제약 조건과 같은 명명된 스키마 요소를 삭제하는 데 사용됩니다.

- **드롭 동작 옵션:**
  - 종속그리고얽매다

- **예:**
  - DROP SCHEMA 회사 캐스케이드;
  - 이렇게 하면 스키마와 테이블, 뷰, 제약 조건 등을 포함한 모든 요소가 제거됩니다.

ALTER 테이블 명령

- **테이블 변경 작업은 다음과 같습니다.**
  - 열(속성) 추가 또는 삭제
  - 열 정의 변경
  - 테이블 제약 조건 추가 또는 삭제

- **예:**
  - ALTER TABLE COMPANY.EMPLOYEE ADD COLUMN Job VARCHAR(12);

제약 조건 추가 및 삭제

- **테이블에 지정된 제약 조건 변경**
  - 명명된 제약 조건 추가 또는 삭제

```
ALTER TABLE COMPANY.EMPLOYEE
DROP CONSTRAINT EMPSUPERFK CASCADE;
```

열 삭제, 기본값

- **열을 삭제하려면**
  - 다음 중 하나를 선택하세요종속또는얽매다
  - 종속뷰 등에서 해당 열을 삭제합니다.얽매다 만약 그것을 언급하는 견 가능하다 해가 없다면.

```
ALTER TABLE COMPANY.EMPLOYEE
                    DROP COLUMN 주소 CASCADE;
```

- **기본값은 삭제 및 변경이 가능합니다.**

```
테이블 변경회사.부서열 변경관리자_SSN
                    기본값 삭제;

테이블 변경회사.부서
                    열 변경관리자_SSN기본값으로 설정'333445555';
```

표 7.2 SQL 구문 요약

Table 7.2 Summary of SQL Syntax

```
CREATE TABLE <table name> ( <column name> <column type> [ <attribute constraint> ]
                                { , <column name> <column type> [ <attribute constraint> ] }
                                [ <table constraint> { , <table constraint> } ] )

DROP TABLE <table name>
ALTER TABLE <table name> ADD <column name> <column type>

SELECT [ DISTINCT ] <attribute list>
FROM ( <table name> { <alias> } | <joined table> ) { , ( <table name> { <alias> } | <joined table> ) }
[ WHERE <condition> ]
[ GROUP BY <grouping attributes> [ HAVING <group selection condition> ] ]
[ ORDER BY <column name> [ <order> ] { , <column name> [ <order> ] } ]

<attribute list> ::= ( * | ( <column name> | <function> ( ( [ DISTINCT ] <column name> | * ) ) )
                          { , ( <column name> | <function> ( ( [ DISTINCT] <column name> | * ) ) } ) )

<grouping attributes> ::= <column name> { , <column name> }

<order> ::= ( ASC | DESC )

INSERT INTO <table name> [ ( <column name> { , <column name> } ) ]
( VALUES ( <constant value> , { <constant value> } ) { , ( <constant value> { , <constant value> } ) }
| <select statement> )
```

다음 슬라이드로 계속

표 7.2 (계속) SQL 구문 요약

Table 7.2 Summary of SQL Syntax

DELETE FROM <table name> [WHERE <selection condition>]
UPDATE <table name> SET <column name> = <value expression> { , <column name> = <value expression> } [WHERE <selection condition>]
CREATE [UNIQUE] INDEX <index name> ON <table name> (<column name> [<order>] { , <column name> [<order>] }) [CLUSTER]
DROP INDEX <index name>
CREATE VIEW <view name> [(<column name> { , <column name> })] AS <select statement>
DROP VIEW <view name>

NOTE: The commands for creating and dropping indexes are not part of standard SQL.

요약

- **복잡한 SQL:**
  - 중첩 쿼리, 조인된 테이블(FROM 절), 외부 조인, 집계 함수, 그룹화

- **의미적 제약을 처리하기주장 만들기그리고 트리거 생성**

- **뷰 생성진술 및 구체화 전략**

- **DBA를 위한 스키마 수정ALTER TABLE, ADD 및 DROP COLUMN, ALTER CONSTRAINT등.**