

# 데이터베이스 설계 (Database Design) Lecture 06: SQL II

담당교수: 전강욱(컴퓨터공학부)  
kw.chon@koreatech.ac.kr

# 지난시간 복습

## ■ SQL 발전역사 확인

- IBM에서 개발된 SYSTEM R을 위한 언어로부터 유래
- 이후 미국 표준 연구소인 ANSI와 국제 표준화 기구인 ISO에서 표준화 작업을 진행
- 표준화가 진행될 수록 다양한 기능이 추가

## ■ SQL은 선언적 언어이며, 원하는 데이터만을 명시하여 데이터 추출

# 지난시간 복습 (계속)

## ■ CREATE TABLE명령

- PRIMARY KEY절은 릴레이션의 기본키를 구성하는 하나 이상의 속성을 명시
- UNIQUE 절은 대체키를 명시
- FOREIGN KEY절은 참조 무결성을 지정

## ■ 속성 데이터 타입과 도메인 확인

- CREATE DOMAIN명령을 통해 도메인은 새로 생성 가능

## ■ 제약조건 및 디폴트 값 명시

- NOT NULL, DEFAULT 등

# 지난시간 복습 (계속)

- **DROP 명령어: DB나 테이블 삭제**
  - **DROP SCHEMA DB명 [CASCADE|RESTRICT];**
  - **DROP TABLE 테이블명 [CASCADE|RESTRICT];**
- **ALTER 명령어: 기본 테이블의 정의를 변경**
  - 속성의 추가/제거, 열 정의 변경, 테이블 제약 조건 추가/제거
- **SELECT 문: DB에서 정보를 검색하는 명령어**
  - **SELECT 속성목록 FROM 테이블명 WHERE 조건;**

# 지난시간 복습 (계속)

- **INSERT 명령어: 릴레이션에 하나 또는 이상의 튜플들을 삽입하는 명령어**
  - **INSERT INTO 테이블명 VALUES** (추가할 레코드 값);
- **DELETE 명령어: 릴레이션에서 하나 또는 이상의 튜플들을 제거하는 명령어**
  - **DELETE FROM 테이블명 WHERE** 삭제조건;
- **UPDATE 명령어: 튜플의 속성 값을 변경하는 명령어**
  - **UPDATE 테이블명 SET 속성 WHERE** 변경조건;

# 지난시간 복습 (계속)

- 뷰(View)는 다른 테이블들에서 유도된 가상 테이블
  - 실제로 저장되지는 않음
  - 뷰에 대한 질의는 제한을 받지 않지 않음(갱신 연산 제외)
- 뷰 관련 명령어
  - 뷰 정의: CREATE VIEW 뷰이름;
  - 뷰 삭제: DROP VIEW 뷰이름;

# 복습 문제

- VIEW는 물리적으로 구현되어 있는 테이블이다 ( )
- SELECT \* WHERE DEGREE="PHD" ( )
- 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정의하는 데 사용되는 언어는 DML이다 ( )
- 기본키는 NOT NULL 속성을 반드시 포함한다 ( )
- GROUP BY 절의 조건으로 일반적으로 WHERE 절을 사용한다 ( )
- 테이블을 정의할 때 ALTER TABLE 명령어를 이용한다 ( )
- 중복된 레코드가 한 번만 검색되도록 하는 명령어는 ( ) 이다

# 복습 문제 (계속)

- 뷰는 삽입, 삭제, 갱신 연산에 제약 사항이 따른다 ( )
- 뷰는 데이터 접근 제어로 보안을 제공한다 ( )
- 뷰는 물리적으로 구현되는 테이블이다 ( )
- 뷰는 데이터의 논리적 독립성을 제공한다 ( )
- 관계 데이터베이스에서 main table의 데이터를 삭제 시 외래키에 대해 부합되는 모든 데이터를 삭제하는 참조 무결성의 법칙은 ( ) 이다



# 복습 문제 (계속)

- DEPENDENT 테이블에서 ESSN 속성을 중복을 제거하고 검색하시오
- PROJECT 테이블에서 PLOCATION이 "Houston"인 PNAME, PNUMBER, DNUM을 검색하시오

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

# 복습 문제 (계속)

- EMPLOYEE 테이블에서 직원 이름(FNAME, MINT, LNAME)과 SALARY를 출력하되, SALARY에 50000달러를 더하여 UPDATED\_SALARY라는 새 이름으로 검색하시오

EMPLOYEE	FNAME	MINT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	09-Jan-55	Houston	M	30000	333445555	5
	Franklin	T	Wong	333445555	08-Dec-45	Houston	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	19-Jul-58	Spring	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	20-Jun-31	Bellaire	F	43000	888665555	4
	Ramesh	K	Narayn	666884444	15-Sep-52	Oak	M	38000	333445555	5
	Joyce	A	English	453453453	31-Jul-62	Houston	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	29-Mar-59	Houston	M	25000	987654321	4
	James	E	Borg	888665555	10-Nov-27	Houston	M	55000	NULL	1

# 복습 문제 (계속)

- EMPLOYEE 테이블에서 생년월일 기준으로 내림차순이 되도록 모든 열을 검색하시오
- EMPLOYEE 테이블에서 ADDRESS별 평균 연봉을 검색하시오

EMPLOYEE	FNAME	MINT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	09-Jan-55	Houston	M	30000	333445555	5
	Franklin	T	Wong	333445555	08-Dec-45	Houston	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	19-Jul-58	Spring	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	20-Jun-31	Bellaire	F	43000	888665555	4
	Ramesh	K	Narayn	666884444	15-Sep-52	Oak	M	38000	333445555	5
	Joyce	A	English	453453453	31-Jul-62	Houston	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	29-Mar-59	Houston	M	25000	987654321	4
	James	E	Borg	888665555	10-Nov-27	Houston	M	55000	NULL	1

# 복습 문제 (계속)

- EMPLOYEE 테이블에서 SUPERSSN이 333445555 인 직원의 FNAME, SSN, BDATE를 나타내는 VIEW를 SUPERVISED\_BY\_333445555 라는 이름으로 만드시오

EMPLOYEE	FNAME	MINT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	09-Jan-55	Houston	M	30000	333445555	5
	Franklin	T	Wong	333445555	08-Dec-45	Houston	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	19-Jul-58	Spring	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	20-Jun-31	Bellaire	F	43000	888665555	4
	Ramesh	K	Narayn	666884444	15-Sep-52	Oak	M	38000	333445555	5
	Joyce	A	English	453453453	31-Jul-62	Houston	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	29-Mar-59	Houston	M	25000	987654321	4
	James	E	Borg	888665555	10-Nov-27	Houston	M	55000	NULL	1

# NCS 정보

- 능력 단위명 : 물리 데이터베이스 설계, 데이터베이스 구현
- 능력 단위요소 : 물리 E-R 다이어그램 작성하기, 데이터베이스 오브젝트 생성
- 학습목표(수행 준거) :
  - 3.1 논리 데이터베이스 설계에서 엔티티, 속성, 주식별자, 외래 식별자를 각각 테이블, 컬럼, 기본 키, 외래 키로 변환하여 표현할 수 있다.
  - 3.2 물리 데이터베이스 설계에 따라 데이터베이스 오브젝트를 생성하기 위한 DDL(Data Definition Language)을 작성할 수 있고 생성된 오브젝트에 대한 유효성 여부를 검사할 수 있다.
  - 3.3 엔티티 명, 속성 명에 대한 data dictionary(용어사전)를 정의하고, 테이블 명, 컬럼 명, 키 종류, NULL값 허용여부 정보를 기준으로 테이블 기술서를 작성할 수 있다.

# NCS 정보 (계속)

## ■ 지식

- DDL과 DML 문법 이해 및 사용법
- 테이블 정의서(내부 스키마)
- 무결성 제약조건 이해

## ■ 기술

- DDL로 테이블을 생성/삭제하는 능력
- 테이블 정의서(내부 스키마) 기술 능력

## ■ 태도

- SQL문을 이해하고 활용하려는 노력
- 테이블 정의서(내부 스키마)를 정확히 기술하려는 태도
- 테이블 정의서를 준수하려는 태도
- 생성된 테이블이 활용도를 다각도로 테스트하려는 적극적인 태도

# 세부 학습목표

- 1. 물리적 모델링의 주요 업무를 나열할 수 있다.
- 2. 릴레이션 스키마를 내부 스키마로 변환할 수 있다.
- 3. 성능 향상을 위해 물리적 구조를 변경하는 방법에 대해 설명할 수 있다.

COMPANY 데이터베이스	EMPLOYEE	FNAME	MINT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
		John	B	Smith	123456789	09-Jan-55	Houston	M	30000	333445555	5
		Franklin	T	Wong	333445555	08-Dec-45	Houston	M	40000	888665555	5
		Alicia	J	Zelaya	999887777	19-Jul-58	Spring	F	25000	987654321	4
		Jennifer	S	Wallace	987654321	20-Jun-31	Bellaire	F	43000	888665555	4
		Ramesh	K	Narayn	666884444	15-Sep-52	Oak	M	38000	333445555	5
		Joyce	A	English	453453453	31-Jul-62	Houston	F	25000	333445555	5
		Ahmad	V	Jabbar	987987987	29-Mar-59	Houston	M	25000	987654321	4
		James	E	Borg	888665555	10-Nov-27	Houston	M	55000	NULL	1

WORKS_ON	ESSN	PNO	HOURS	DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	123456789	1	32.5		Research	5	333445555	22-May-78
	123456789	2	7.5		Administration	4	987654321	01-Jan-85
	666884444	3	40		Headquarters	1	888665555	19-Jun-71
	453453453	1	20					
	453453453	2	20					
	333445555	2	10					
	333445555	3	10					
	333445555	10	10					
	333445555	20	10					
	999887777	30	30					
	999887777	10	10					
	987987987	10	35					
	987987987	30	5					
	987654321	30	20					
	987654321	20	15					
	888665555	20	null					

DEPT_LOCATION	DNUMBER	DLOCATION
	1	서울
	4	천안
	5	서울
	5	부산
	5	대전

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPARTMENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	05-Apr-76	DAUGHTER
	333445555	Theodore	M	25-Oct-73	SON
	333445555	Joy	F	03-May-48	SPOUSE
	987654321	Abner	M	29-Feb-32	SPOUSE
	123456789	Michael	M	01-Jan-78	SON
	123456789	Allice	F	32-DEC-78	DAUGHTER
	123456789	Elizabeth	F	05-May-57	SPOUS



# 널 값을 포함한 비교

## ■ 널 값의 의미

- 알려지지 않은 값 (존재하지만 알지 못하는)
- 이용할 수 없거나 오류해둔 값 (존재하지만 의도적으로 오류한)
- 적용할 수 없는 속성 (이 튜플에는 정의되지 않는)

## ■ 속성 값이 NULL인지 검사

- IS NULL
- IS NOT NULL

## ■ 질의 18

- 상사가 없는 모든 종업원들의 이름을 검색하시오.  

```
SELECT  FNAME, LNAME  
FROM    EMPLOYEE  
WHERE   SUPERSSN IS NULL ;
```

# 중첩 질의(nested query)와 집합 비교

## ■ 중첩 질의

- 다른 질의의 WHERE절 내에 완전한 SELECT 질의가 나타나는 형태
- 외부 질의와 내부 질의로 구분

## ■ 비교연산자 IN

- 외부 질의의 한 튜플에 대하여, 이 튜플이 임의의 튜플 집합의 원소가 되는지 비교하는 연산

# 중첩 질의(nested query)와 집합 비교 (계속)

## ■ 질의 4A

- 성이 'Smith'인 종업원(일반 직원 혹은 프로젝트를 담당하는 부서의 관리자)이 참여하는 프로젝트의 프로젝트 번호 목록을 작성하시오.

```
SELECT  DISTINCT PNUMBER
FROM    PROJECT
WHERE   PNUMBER IN ( SELECT  PNUMBER
                      FROM    PROJECT, DEPARTMENT, EMPLOYEE
                      WHERE   DNUM=DNUMBER AND MGRSSN=SSN
                      AND LNAME='Smith')

OR

PNUMBER IN ( SELECT  PNO
              FROM    WORKS_ON, EMPLOYEE
              WHERE   ESSN=SSN AND LNAME='Smith') ;
```

# 중첩 질의(nested query)와 집합 비교 (계속)

## ■ 질의 4A

- 성이 'Smith'인 종업원(일반 직원 혹은 프로젝트를 담당하는 부서의 관리자)이 참여하는 프로젝트의 프로젝트 번호 목록을 작성하시오.

```
SELECT  DISTINCT PNUMBER
FROM    PROJECT
WHERE   PNUMBER IN ( SELECT  PNUMBER
                      FROM    PROJECT, DEPARTMENT, EMPLOYEE
                      WHERE   DNUM=DNUMBER AND MGRSSN=SSN
                      AND LNAME='Smith')

OR

PNUMBER IN ( SELECT  PNO
              FROM    WORKS_ON, EMPLOYEE
              WHERE   ESSN=SSN AND LNAME='Smith') ;
```

# 중첩 질의(nested query)와 집합 비교 (계속)

## ■ 질의

- SSN이 123456789인 사원이 일하는 프로젝트와 일한 시간의 조합이 동일한 사원의 SSN을 검색하라.

```
SELECT DISTINCT ESSN
FROM   WORKS_ON
WHERE  (PNO, HOURS) IN ( SELECT      PNO, HOURS
                        FROM        WORKS_ON
                        WHERE         SSN='123456789');
```

# 중첩 질의(nested query)와 집합 비교 (계속)

## ■ = ALL 연산자

- 하나의 값 v가 집합 V내의 모든 값들과 같으면 참이 됨
- ALL 앞에 >, >=, <, <=, <를 사용할 수도 있음

## ■ = ANY(= SOME) 연산자

- 하나의 값 v가 집합 V내의 어떤 하나의 값과 같으면 참이 됨
- ANY(SOME) 앞에 >, >=, <, <=, <를 사용할 수도 있음

## ■ 질의

- 5번 부서에 근무하는 모든 사원보다 급여가 많은 사원을 검색하라.

```
SELECT  LNAME, FNAME
FROM    EMPLOYEE
WHERE   SALARY > ALL ( SELECT  SALARY
                        FROM    EMPLOYEE
                        WHERE   DNO=5) ;
```

# 중첩 질의(nested query)와 집합 비교 (계속)

## ■ 중첩 질의에서 속성 이름의 모호성

- 만약 외부 질의문의 FROM 절에 있는 릴레이션과 내부 질의문의 FROM 절에 있는 다른 릴레이션에 동일한 속성명이 있다면 속성 이름의 모호성이 발생
- 애매한 속성에 대한 참조규칙은 항상 **가장 안쪽 가까운 질의문에 선언된 릴레이션을 먼저 참조**하는 것임
- 내부 질의에서 외부 질의에 명시된 릴레이션의 속성을 참조하려면 별명을 사용해야 함

## ■ 질의 16

- 자신의 부양가족과 이름, 성별이 같은 종업원들의 이름을 검색하시오.

```
SELECT  E.FNAME, E.LNAME
FROM    EMPLOYEE AS E
WHERE   E.SSN IN ( SELECT  ESSN
                   FROM    DEPENDENT
                   WHERE    E.FNAME=DEPENDENT_NAME AND
                           E.SEX=SEX) ;
```

# 상관 중첩 질의

## ■ 상관된 질의 (Correlated Query)

- 내부 질의의 WHERE 절에 있는 조건에서 외부질의에 선언된 릴레이션의 일부 애트리뷰트를 참조하는 경우에 두 질의를 상관된 질의라고 함

## ■ 비중첩 질의로의 변환

- 중첩된 SELECT ... FROM ... WHERE... 블록과 =과 IN 비교 연산자를 이용해서 작성한 질의는 항상 단일 블록 질의로 변환할 수 있음

## ■ 질의 16A

- 자신의 부양가족과 이름, 성별이 같은 종업원들의 이름을 검색하시오.

```
SELECT  E.FNAME, E.LNAME
FROM    EMPLOYEE AS E
WHERE   E.SSN IN (SELECT ESSN
                  FROM DEPENDENT AS D
                  WHERE E.FNAME=D.DEPENDENT_NAME
                  AND E.SEX=D.SEX);
```

상관  
중첩질의

```
SELECT  E.FNAME, E.LNAME
FROM    EMPLOYEE AS E, DEPENDENT AS D
WHERE   E.SSN=D.ESSN AND E.FNAME=DEPENDENT_NAME AND
        E.SEX=D.SEX ;
```

비중첩질의



# EXISTS 함수

## ■ EXISTS 함수

- 상관된 중첩질의에서 내부 질의의 결과가 **공집합인가를 검사함**
- EXISTS(Q)
  - 질의 Q의 결과에 **최소한 한 개의 튜플이 있다면 참을 반환**

## ■ 질의 16B

- 자신의 부양가족과 이름, 성별이 같은 종업원들의 이름을 검색하시오.

```
SELECT  E.FNAME, E.LNAME
FROM    EMPLOYEE E
WHERE   EXISTS ( SELECT  *
                  FROM    DEPENDENT
                  WHERE    E.SSN=ESSN AND SEX=E.SEX AND
                           E.FNAME=DEPENDENT_NAME) ;
```

# EXISTS 함수 (계속)

## ■ NOT EXIST 함수

- 상관된 중첩질의에서 내부 질의의 결과가 **공집합인가를 검사함**
- NOT EXISTS(Q)
  - 질의 Q의 결과에 **튜플이 없다면 참을 반환**

## ■ 질의 6

- 부양가족이 없는 종업원들의 이름을 검색하시오.

```
SELECT  FNAME, LNAME
FROM    EMPLOYEE
WHERE   NOT EXISTS ( SELECT  *
                      FROM    DEPENDENT
                      WHERE    SSN=ESSN) ;
```

## ■ UNIQUE 함수

- 질의 Q의 결과에 **중복된 튜플이 없다면 TRUE를 반환**

# EXISTS 함수 (계속)

## ■ 질의 7

- 부양가족이 적어도 한명 이상 있는 관리자의 이름을 검색하라.

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE EXISTS (SELECT *
               FROM DEPENDENT
               WHERE SSN=ESSN)
AND
EXISTS (SELECT *
        FROM DEPARTMENT
        WHERE SSN=MGRSSN) ;
```

# EXISTS 함수 (계속)

## ■ 질의

- 5번 부서가 담당하는 모든 프로젝트에 근무하는 직원들의 이름을 검색하라. (각 직원에 대하여 그 직원이 근무하지 않는 5번 부서가 관리하는 프로젝트가 존재하지 않는 경우에 그 직원을 검색하라)

```
SELECT      FNAME, LNAME
FROM        EMPLOYEE
WHERE       NOT EXISTS (
    SELECT    *
    FROM      WORKS_ON AS B
    WHERE     (B.PNO IN (
        SELECT PNUMBER
        FROM   PROJECT
        WHERE  DNUM=5))

    AND
    NOT EXISTS (
        SELECT *
        FROM   WORKS_ON AS C
        WHERE  C.PNO=B.PNO)) ;
```

# 명시적 집합과 속성의 재명명

- WHERE 절에 값들의 명시적 집합 사용 가능

- 질의 17

- 프로젝트 번호 1, 2, 3에서 일하는 모든 종업원들의 SSN을 검색하시오.

```
SELECT DISTINCT ESSN  
FROM   WORKS_ON  
WHERE  PNO IN (1, 2, 3) ;
```

# 명시적 집합과 속성의 재명명 (계속)

## ■ 질의 결과 애트리뷰트의 재명명

- 결과에 나타나는 애트리뷰트의 이름은 키워드 **AS**를 사용하여 원하는 새 이름으로 재명명할 수 있음
- AS를 사용하여 애트리뷰트와 릴레이션에 별명을 붙일 수 있음

## ■ 질의 8A

- 종업원에 대해, 종업원의 성과 이름, 직속 감독자의 성과 이름을 검색하시오.

```
SELECT  E.LNAME AS EMPLOYEE_NAME,  
        S.LNAME AS SUPERVISOR_NAME  
FROM    EMPLOYEE AS E, EMPLOYEE AS S  
WHERE   E.SUPERSSN=S.SSN ;
```

# 조인된 테이블

- FROM 절에 조인 연산의 결과를 지정

- SQL2에서는 질의의 FROM 절에 조인연산의 결과를 지정할 수 있음

- 질의 1A

- ‘Research’ 부서에서 일하는 모든 종업원들의 이름과 주소를 검색하시오.

```
SELECT  FNAME, LNAME, ADDRESS
FROM    (EMPLOYEE JOIN DEPARTMENT ON DNO=DNUMBER)
WHERE   DNAME='Research' ;
```

# 집단 함수

## ■ 집단함수

- SQL에서는 COUNT, SUM, MAX, MIN, AVG 등의 집단 (or 내장) 함수를 제공함
- COUNT 함수는 질의에서 튜플이나 값의 개수를 반환함
- SUM, MAX, MIN, AVG 함수는 수치 값들의 다중집합에 적용되며, 각각 합, 최대값, 최소값, 평균값을 반환함

## ■ 질의 19

- 종업원의 급여의 합, 최고 급여, 최저 급여, 평균 급여를 구하시오.  

```
SELECT    SUM (SALARY), MAX (SALARY), MIN (SALARY), AVG (SALARY)
FROM      EMPLOYEE ;
```



# 집단 함수 (계속)

## ■ 집단함수

- 조건을 만족하는 튜플들을 대상으로 집단 함수 값들을 얻으려면 , WHERE절에서 튜플의 조건을 제시할 수 있음

## ■ 질의 20

- ‘Research’ 부서에 있는 모든 종업원들의 급여의 합과 최고 급여 , 최소 급여, 평균 급여를 구하시오.

```
SELECT  SUM (SALARY), MAX (SALARY), MIN (SALARY), AVG  
        (SALARY)  
FROM    EMPLOYEE, DEPARTMENT  
WHERE   DNO=DNUMBER AND DNAME='Research' ;
```

# 집단 함수 (계속)

- COUNT(\*)

- 튜플의 수를 반환

- 질의 21

- 회사내의 총 종업원의 수를 검색하시오.

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

- 질의 22

- 'Research' 부서에 속해 있는 종업원의 수를 검색하시오.

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

# 집단 함수 (계속)

- 특정 튜플을 선택하기 위해 집단함수 이용

- 중첩질의 이용

- 질의 5

- **둘 이상**의 부양가족이 있는 모든 사원의 이름을 검색하시오.

```
SELECT  LNAME, FNAME
FROM    EMPLOYEE
WHERE ( SELECT  COUNT (*)
        FROM    DEPENDENT
        WHERE   SSN=ESSN ) >= 2 ;
```

# 그룹핑: Group by와 Having 절

## ■ 그룹화 (grouping)

- 특정 속성(들)의 값이 같은 튜플들을 모아서 그룹을 생성하고, 이들 그룹에 대하여 집단함수를 적용함
- 이 때, 특정 애트리뷰트들을 **그룹화 애트리뷰트** 라고 하며, SQL의 **GROUP BY**절에 지정함
- 대부분의 경우, SELECT절에 그룹화 애트리뷰트(들)를 지정하여 그 값과 그 값에 해당하는 튜플 그룹에 **집단함수**를 적용한 결과를 동시에 반환함

# 그룹핑: Group by와 Having 절 (계속)


## ■ 질의 24

- 각 부서에 대해서 부서 번호, 부서 내에 있는 종업원의 수, 평균 봉급은?

```
SELECT      DNO, COUNT (*), AVG (SALARY)
FROM        EMPLOYEE
GROUP BY    DNO ;
```

- EMPLOYEE 튜플들을 DNO 값을 기준으로 분할하여 그룹들을 생성함
- 그 다음에, 각 그룹의 튜플들에 대하여 COUNT와 AVG함수를 적용함

EMPLOYEE	FNAME	MINT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	09-Jan-55	Houston	M	30000	333445555	5
	Franklin	T	Wong	333445555	08-Dec-45	Houston	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	19-Jul-58	Spring	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	20-Jun-31	Bellaire	F	43000	888665555	4
	Ramesh	K	Narayn	666884444	15-Sep-52	Oak	M	38000	333445555	5
	Joyce	A	English	453453453	31-Jul-62	Houston	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	29-Mar-59	Houston	M	25000	987654321	4
	James	E	Borg	888665555	10-Nov-27	Houston	M	55000	NULL	1



DNO	COUNT(*)	AVG(SALARY)
5	4	33250
4	3	31000
1	1	55000

# 그룹핑: Group by와 Having 절 (계속)

## ■ 질의 25

- 각 프로젝트에 대해서 프로젝트 번호, 프로젝트 이름, 그 프로젝트에서 근무하는 사원들의 수를 검색하라.

```
SELECT    PNUMBER, PNAME, COUNT (*)  
FROM      PROJECT, WORKS_ON  
WHERE     PNUMBER=PNO  
GROUP BY  PNUMBER ;
```

# 그룹핑: Group by와 Having 절 (계속)

## ■ 질의 26

- 두 명 이상의 사원이 근무하는 각 프로젝트에 대해서 프로젝트 번호, 프로젝트 이름, 프로젝트에서 근무하는 사원의 수를 검색하라.

```
SELECT      PNUMBER, PNAME, COUNT (*)  
FROM        PROJECT, WORKS_ON  
WHERE       PNUMBER=PNO  
GROUP BY    PNUMBER ;  
HAVING      COUNT (*) >= 2
```

# 그룹핑: Group by와 Having 절 (계속)

## ■ 질의 27

- 각 프로젝트에 대해서 프로젝트 번호, 프로젝트 이름, 5번 부서에 속하면서 프로젝트에서 근무하는 사원의 수를 검색하라.

```
SELECT      PNUMBER, PNAME, COUNT (*)  
FROM        PROJECT, WORKS_ON, EMPLOYEE  
WHERE       PNUMBER=PNO AND SSN=ESSN AND DNO=5  
GROUP BY   PNUMBER ;
```



# 그룹핑: Group by와 Having 절 (계속)

## ■ 질의 28

- 6명 이상의 사원이 근무하는 각 부서에 대해서 부서번호와 40,000 달러가 넘는 급여를 받는 사원의 수를 검색하라.

```
SELECT      DNUMBER, COUNT (*)
FROM        DEPARTMENT, EMPLOYEE
WHERE       DNUMBER=DNO AND SALARY > 40000 AND
           DNO IN ( SELECT      DNO
                   FROM        EMPLOYEE
                   GROUP BY     DNO
                   HAVING      COUNT (*) >= 6)
GROUP BY    DNUMBER ;
```

# SQL에 대한 논의와 요약

- SQL 질의는 6개의 절로 구성되지만, 필수사항은 처음의 두 개 뿐임
  - 질의의 평가 순서
    - FROM → WHERE절 → GROUP BY → HAVING → SELECT → ORDER BY
  - SELECT <애트리뷰트 목록>
    - SELECT 절은 질의 결과에 포함될 애트리뷰트들이나 함수를 나열함
  - FROM <테이블 목록>
    - FROM 절은 질의의 대상을 명시하는 곳으로 조인된 릴레이션이나 릴레이션(들)을 지정함
  - [WHERE <조건>]
    - WHERE 절은 튜플들에 대한 조건을 명시함
  - [GROUP BY <집단화 애트리뷰트>]
    - GROUP BY절은 그룹화 애트리뷰트들을 지정함
  - [HAVING <집단 조건>]
    - HAVING 절은 그룹들에 대한 조건을 지정함
  - [ORDER BY <애트리뷰트 목록>]
    - ORDER BY 절은 정렬 기준이 되는 애트리뷰트(들)을 지정함

# 뷰(view)의 개념

- 뷰는 다른 테이블들 또는 이전에 정의한 뷰에서 유도된 가상 테이블
  - SELECT 문의 결과를 뷰로 지정 가능
- 뷰는 가상의 테이블이므로, 물리적인 형태로 저장되지 않음
  - 뷰에 적용할 수 있는 갱신 연산들은 제한
  - 뷰에 대한 질의는 특별한 제한이 없음

# 뷰의 명시

## ■ 뷰의 생성

- CREATE VIEW 문을 사용하여 생성
- 뷰이름, 속성명 리스트, 뷰의 내용을 지정하는 질의로 구성
  - 뷰의 속성명을 생략할 경우, 기본 테이블의 속성명과 동일하게 지정됨
  - e.g., `CREATE VIEW WORKS_ON1` 속성명을 지정하지 않아, **WORKS\_ON1**의 속성명은 **FNAME, LNAME, PNAME, HOURS**가 됨

```
AS SELECT FNAME, LNAME, PNAME, HOURS
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE SSN=ESSN AND PNO=PNUMBER ;
```

```
CREATE VIEW DEPT_INFO ( DEPT_NAME, NO_OF_EMPS, TOTAL_SAL)
AS SELECT DNAME, COUNT (*), SUM (SALARY) 속성명 명시적으로 지정
FROM DEPARTMENT, EMPLOYEE
WHERE DNUMBER=DNO
GROUP BY DNAME ;
```

# 뷰의 명시(계속)

## ■ 뷰의 사용을 통해 질의 작성을 간단하게 할 수 있음

### □ 뷰는 하나의 테이블처럼 활용

- e.g., WORKS\_ON1 뷰를 이용하여 "ProjectY" 에 참여하는 직원들의 성과 이름을 검색하시오.

```
SELECT PNAME, FNAME, LNAME  
FROM WORKS_ON1  
WHERE PNAME = "ProjectY";
```

```
CREATE VIEW WORKS_ON1  
AS SELECT FNAME, LNAME, PNAME, HOURS  
FROM EMPLOYEE, PROJECT, WORKS_ON  
WHERE SSN=ESSN AND PNO=PNUMBER ;
```

뷰를 사용하지 않는 경우, 두 개 이상의 릴레이션들의 조인이 필요

### □ 뷰의 최신성

- 뷰는 SQL 질의에서 참조될 때마다 기본 테이블을 활용하여 계산되므로 항상 최신 정보 제공
- 기본 테이블의 튜플들의 갱신이 뷰에 자동적으로 반영

# 뷰의 명시(계속)

## ■ 뷰의 특성

- 뷰를 최신 정보로 유지하는 것은 사용자 관여 없이 DBMS가 관리
- 뷰는 보안기법의 일종으로도 사용
  - 사용자에게 관련 데이터만 제공 가능

## ■ 뷰의 삭제

- 뷰가 필요하지 않는 경우, DROP VIEW 명령으로 제거
- e.g., DROP VIEW WORKS\_ON1

# 뷰의 구현

## ■ 질의수정(query modification) 방식

- 뷰에 대한 질의를 기본 테이블들에 대한 질의로 변환하여 처리
- 단점: 복잡한 질의로 정의된 뷰들은 비효율적
  - 특히 짧은 시간 내에 뷰에 많은 질의가 적용될 때

## ■ 뷰의 실체화(view materialization)

- 임의의 뷰 테이블을 물리적으로 생성하고 유지하는 방식
- 가정: 뷰에 다른 질의들이 사용됨
- 문제점: 기본 테이블이 갱신되면 뷰 테이블도 변경해야 함
- 해결방법: 오버헤드가 적은 점진적 갱신(incremental update)기법 필요

# 뷰의 갱신

## ■ 집단함수를 사용하지 않은 단일 뷰의 갱신

- 하나의 기본 릴레이션을 사용해서 정의된 뷰가 기본 릴레이션의 기본 키를 포함하면, 뷰의 각 튜플을 기본 릴레이션의 한 튜플과 정확하게 대응하므로 뷰의 갱신이 가능함

## ■ 조인을 포함하는 뷰의 갱신

- 기본 릴레이션들에 대한 갱신 동작으로 사상 가능

## ■ 갱신할 수 없는 뷰

- 그룹화와 집단함수를 사용하여 정의된 뷰는 베이스 테이블에 대한 갱신으로 매핑하는데 모호성이 있으므로 갱신할 수 없음
- 일반적으로 다수의 릴레이션들을 조인하여 생성한 뷰는 갱신 불가능함



# 주장(Assertion)으로 제약조건 명시

## ■ 선언적 주장으로 확장된 제약조건 명시

- DDL의 CREATE ASSERTION 문을 활용
- 각 주장문은 **제약조건 이름**을 가지며, 다음에 키워드 **CHECK**가 오며, 데이터베이스 상태가 주장을 만족하는 여부에 따라 **조건**이 뒤에 옴
  - 제약조건 이름은 해당 제약조건을 참조/수정/삭제하기 위해 사용
  - 조건의 경우 어떠한 WHERE절도 사용될 수 있지만, 대부분의 경우 EXISTS나 NOT EXISTS 사용
- e.g., 사원의 급여가 자신이 근무하는 부서의 관리자의 급여보다 많을 수 없다.

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK (NOT EXISTS (
    SELECT *
    FROM EMPLOYEE E, EMPLOYEE M, DEPARTMENT D
    WHERE      E.SALARY > M.SALARY AND
              E.DNO=D.NUMBER AND
              D.MGRSSN=M.SSN));
```

# SQL 트리거(Trigger)

- SQL 트리거는 조건이 발생할 때 데이터베이스를 모니터하기 위해 행동의 유형을 명시함
- 트리거는 주장과 유사한 구문으로 표기되며 다음 사항을 포함함
  - 사건 (e.g., 갱신 연산): 어떤 사건이 있으면 검사가 시작
  - 조건: 동작을 수행할 조건을 명시
  - 동작: 주어진 조건이 만족되면 수행
- **주장문**은 주장조건을 위반하는 **갱신을 금지**시키는 반면에, **트리거 문**은 갱신을 수행하고 나서 트리거 조건이 발생하면 **정의된 동작을 실행**함
- e.g., 어떤 부서에 있는 직원이 그 부서의 관리자보다 월급이 많다면 지정된 함수를 실행하는 트리거는 다음과 같이 지정

```
DEFINE TRIGGER SALARY_TRIGGER
ON EMPLOYEE E, EMPLOYEE M, DEPARTMENT D :
    E.SALARY > M.SALARY AND E.DNO = D.DNUMBER AND D.MGRSSN=M.SSN
ACTION_PROCEDURE INFORM_MANAGER (D.MGRSSN) ;
```

# SQL의 추가적인 기능들

## ■ 권한 기능

- SQL은 데이터베이스 사용자에게 권한을 부여하고 취소하는 기능을 제공함
- i.e., GRANT/REVOKE

## ■ 호스트 언어와 결합되어 사용

- SQL은 C, C++, COBOL, JAVA, PASCAL 등과 같은 범용 프로그래밍 언어 내에서 사용될 수 있음
- Embedded SQL/C, C++, COBOL, JAVA, PASCAL

## ■ 트랜잭션 기능

- SQL은 트랜잭션 제어 명령문을 가짐 (begin transaction / end transaction)

## ■ 기타 유용한 명령어

- 상용 DBMS는 SQL 명령 이외에도 물리적 데이터베이스 설계 매개변수와 릴레이션들을 위한 파일 구조, 그리고 인덱스와 같은 접근경로를 명시하기 위한 명령어의 집합을 가지고 있음

# 요약

- 더 복잡한 SQL 검색 질의
- SQL에서의 뷰
- 주장으로 제약조건 및 트리거로 동작 명시

**감사합니다!**

**kw.chon@koreatech.ac.kr**