Chpater 2 Review
(Introduction to Relational Model)

Structure of Relational Database (Cont.)

Relation

Refer to a table

Attribute

Refer to a column of a table

Tuple (or record)

Refer to a row in a table

attributes
(or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(or rows)

Tuple

2.2 Database Schema

A database consists of multiple relations

Database schema

- Logical design of the database

Database instance

- A snapshot of the data in the database at a given instant in time

Relation schema

- Logical design of a table
- A list of attributes and their corresponding domains

Relation instance

- A snapshot of the data in a table at a given instant in time

2.3 Keys

Type of Keys: **Superkey, Candidate key, Primary key, Foreign key**

Superkey (슈퍼키): *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*

튜플을 식별할 수 있는 하나의 속성(Attributes) 또는 속성들의 집합

ID	이름	주민번호	주소	핸드폰
C1	박지성	810101-1111111	영국 맨체스터	000-5000-0001
C2	김연아	900101-2222222	한국 서울	000-6000-0001
C3	장미란	830101-2333333	한국 강원도	000-7000-0001
C4	추신수	820101-1444444	미국 클리브랜드	000-8000-0001

- **슈퍼키**
  - **{주민번호} , {주민번호, 이름}, {주민번호, 이름, 주소}, {주민번호, 이름, 핸드폰}**

  {ID}, {ID, 이름, 주소}, {ID, 이름, 주민번호, 주소, 핸드폰} 등….

2.3 Keys

Candidate key(후보키): Candidate Key is minimal superkey for which no proper subset is a superkey

튜플을 식별할 수 있는 슈퍼키들의 최소 집합

ID	이름	주민번호	주소	핸드폰
C1	박지성	810101-1111111	영국 맨체스터	000-5000-0001
C2	김연아	900101-2222222	한국 서울	000-6000-0001
C3	장미란	830101-2333333	한국 강원도	000-7000-0001
C4	추신수	820101-1444444	미국 클리브랜드	000-8000-0001

- **후보키**
  - **{ID}, {주민번호}**
- **슈퍼키**
  - **{주민번호} , {주민번호, 이름}, {주민번호, 이름, 주소}, {주민번호, 이름, 핸드폰}**

  {ID}, {ID, 이름, 주소}, {ID, 이름, 주민번호, 주소, 핸드폰} 등….

2.3 Keys

Primary key(주 키, 기본키): One of the candidate keys is selected to be the **primary key**.

여러 후보키들 중 하나를 선정하여 대표로 삼는 키

기본키 조건

- **튜플을 식별할 수 있는 고유의 값을 가져야함 (슈퍼키, 후보키)**
- **최대한 적은 수의 속성을 가져야 함 (후보키)**
- **키 값의 변동이 일어나면 안됨**
- **Null 값이 없어야함**

2.3 Keys

Foreign key (외래키): Value in one relation must appear in another

Referencing relation

Referenced relation

하나의 릴레이션에서 다른 릴레이션의 속성(attribute)을 참조하는 속성

(일반적으로 외래키는 다른 릴레이션의 기본키을 참조)

ID	이름	주민번호
C1	박지성	810101-1111111
C2	김연아	900101-2222222
C3	장미란	830101-2333333
C4	추신수	820101-1444444

기본키

주문번호	ID	도서
O1	C1	축구의역사
O2	C1	맨유의 전술
O3	C2	피겨 교본
O4	C3	역도의 이해

기본키 외래키

참조

2.5 Relational Query Languages

Procedural vs. non-procedural, or declarative

“Pure” languages:

- Relational algebra
- Tuple relational calculus
- Domain relational calculus

Relational operators (Symbol)

- Selection (σ)
- Project (Π)
- Union (∪)
- Difference (–)
- Intersection (∩)
- Join
  - Cartesian product (x)
  - Natural Join (⋈)

Joining two relations – Cartesian Product

Cartesian Product

Combines all rows in the two tables

Product code	Product name	Unit price
101	Melon	800G
102	Strawberry	150G
103	Apple	120G

X

Export dest. code	Export dest. name
12	The Kingdom of Minanmi
23	Alpha Empire
25	The Kingdom of Ritol

3 rows

Product code	Product name	Unit price	Export dest. code	Export dest. name
101	Melon	800G	12	The Kingdom of Minanmi
101	Melon	800G	23	Alpha Empire
101	Melon	800G	25	The Kingdom of Ritol
102	Strawberry	150G	12	The Kingdom of Minanmi
102	Strawberry	150G	23	Alpha Empire
102	Strawberry	150G	25	The Kingdom of Ritol
103	Apple	120G	12	The Kingdom of Minanmi
103	Apple	120G	23	Alpha Empire
103	Apple	120G	25	The Kingdom of Ritol

3 x 3 =
9 rows

Joining two relations – Cartesian Product

Relations *r, s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

r x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

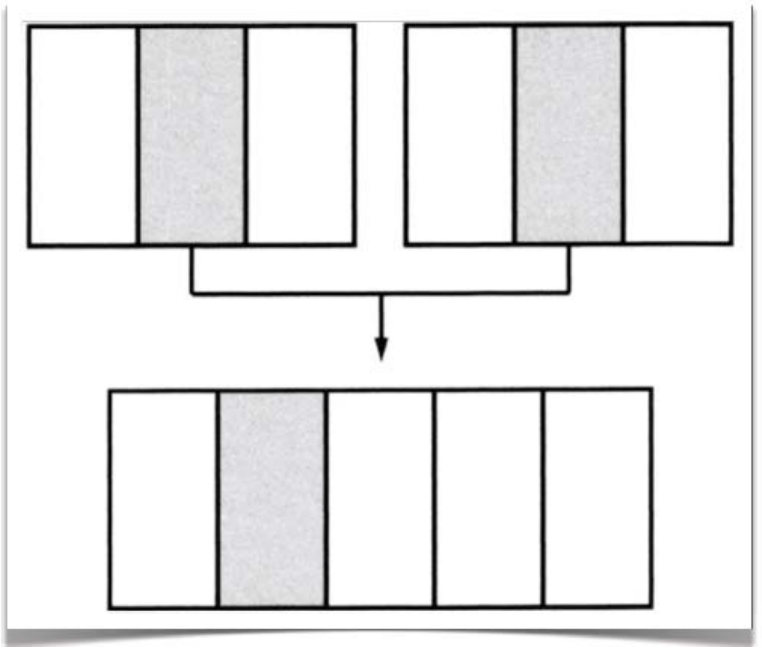
Joining two relations – Natural Join

Let r and s be relations on schemas R and S respectively. Then, the "natural join" of relations R and S is a relation on schema $R \cup S$ obtained as follows:

Consider each pair of tuples t_r from r and t_s from s.

If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where

- t has the same value as t_r on r
- t has the same value as t_s on s



자연조인이란 릴레이션 R과 S에 공통적으로 존재하는 속성들을 이용하여 공통 속성들의 값들이 서로 같은 투플들을 조인하는 것이다.

Natural Join Example

Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

Natural Join

r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Relational Algebra

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary}>=85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\ salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
⋈ (Natural Join)	$instructor \bowtie department$
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
× (Cartesian Product)	$instructor \times department$
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
∪ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the two input relations.

Chapter 3: Introduction to SQL (1)

Revision by Gun-Woo Kim

Dept. of Computer Science and Engineering

Hanyang University

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use

Contents

3.1 Overview of the SQL Query Language

SQL language has several parts

Data-definition language (DDL) provides commands for defining, deleting, modifying relation schemas

Data-manipulation language (DML) provides the ability to query information from the database

Integrity

- SQL DDL includes commands for specify integrity constraints that the data must satisfy

View definition

- SQL DDL includes commands for defining views

Transaction control

Embedded SQL

Authorization

3.1 Overview of the SQL Query Language

SQL language has several parts

Data-definition language (DDL) provides commands for defining, deleting, modifying relation schemas

Data-manipulation language (DML) provides the ability to query information from the database

Integrity

- SQL DDL includes commands for specify integrity constraints that the data must satisfy

View definition

- SQL DDL includes commands for defining views

Transaction control

Embedded SQL

Authorization

3.2 SQL Data Definition

The SQL **data-definition language (DDL)** allows the specification of information about relations, including:

- The schema for each relation
- The domain of values associated with each attribute
- Integrity constraints
- And as we will see later, also other information such as
  - The set of indices to be maintained for each relations
  - Security and authorization information for each relation
  - The physical storage structure of each relation on disk

Basic Types in SQL DDL

char(n). Fixed length character string, with user-specified length *n.*

varchar(n). Variable length character strings, with user-specified maximum length *n.*

int. Integer (a finite subset of the integers that is machine-dependent). 기본4바이트

smallint. Small integer (a machine-dependent subset of the integer domain type). 기본2바이트

numeric(p,d). Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.

numeric (3,1) : 44.5 → OK, 444.5 or 0.32 → ?

float(n). Floating point number, with user-specified precision of at least *n* digits.

More are covered in Chapter 4.

Basic Schema Definition – Create Table

An SQL relation is defined using the **create table** command:

create table r (A_1 D_1, A_2 D_2, ..., A_n D_n,
(integrity-constraint$_1$),
...,
(integrity-constraint$_k$))

r is the name of the relation

each A_i is an attribute name in the schema of relation r

D_i is the data type of values in the domain of attribute A_i

Example:

create table *instructor* (
ID **char**(5),
name **varchar**(20) **not null,**
dept_name **varchar**(20),
salary **numeric**(8,2))

insert into *instructor* **values** (‘10211’, ’Smith’, ’Biology’, 66000);

insert into *instructor* **values** (‘10211’, null, ’Biology’, 66000);

Integrity Constraint in Create Table

not null

primary key (A_1, ..., A_n)

foreign key (A_m, ..., A_n) **references** *r*

The attributes for any tuple in the relation must correspond to the primary key attributes of some tuple in relation *r*

create table *instructor* (

ID **char**(5),
name **varchar**(20) **not null,**
dept_name **varchar**(20),
salary **numeric**(8,2),
primary key (*ID*),
foreign key *(dept_name*) **references** *department)*

Note: Declare *dept_name* as the primary key for *department*

Note: **primary key** declaration on an attribute automatically ensures **not null**

Integrity Constraint in Create Table

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

Key

Product

PName	Price	Category	CName
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Foreign key

More Relation Definitions

create table *student* (
ID **varchar**(5),
name **varchar**(20) **not null**,
dept_name **varchar**(20),
tot_cred **numeric**(3,0),
primary key (*ID*),
foreign key *(dept_name*) **references** *department)*);

create table *takes* (
ID **varchar**(5),
course_id **varchar**(8),
sec_id **varchar**(8),
semester **varchar**(6),
year **numeric**(4,0),
grade **varchar**(2),
primary key *(ID, course_id, sec_id, semester, year),*
foreign key (*ID*) **references** *student,*
foreign key (*course_id, sec_id, semester, year*) **references** *section*);

Note: *sec_id* can be dropped from primary key above, to ensure a student cannot be registered for two sections of the same course in the same semester

And More Still

create table *course* (
course_id **varchar**(8) **primary key**,
title **varchar(**50),
dept_name **varchar**(20),
credits **numeric**(2,0),
foreign key *(dept_name*) **references** *department)*);

Primary key declaration can be combined with attribute declaration as shown above

Drop and Alter Table Constructs

drop table *student*

Deletes the table and its contents

delete from *student*

Deletes all contents of table, but retains table

alter table

alter table *r* **add** *A D*

- where *A* is the name of the attribute to be added to relation *r* and *D* is the domain of *A.*
- All tuples in the relation are assigned *null* as the value for the new attribute.

alter table *r* **drop** *A*

- where *A* is the name of an attribute of relation *r*
- Dropping of attributes not supported by many databases

3.3 Basic Structure of SQL Queries

SQL **data-manipulation language (DML)** provides the ability to query information, and insert, delete and update tuples

A typical SQL query has the form:

select $A_1, A_2, ..., A_n$
from $r_1, r_2, ..., r_m$
where P

```
SELECT  <attributes>
FROM    <one or more relations>
WHERE   <conditions>
```

A_i represents an attribute

R_i represents a relation

P is a predicate (condition).

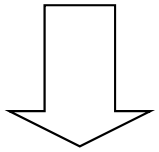
The result of an SQL query is a relation.

3.3 Basic Structure of SQL Queries

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   *
FROM     Product
WHERE    category='Gadgets'
```



PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

"selection"

The select Clause

The **select** clause list the attributes desired in the result of a query

corresponds to the projection operation of the relational algebra

Example: find the names of all instructors:

select *name*
from *instructor*

NOTE: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)

E.g. *Name* ≡ *NAME* ≡ *name*

Some people use upper case wherever we use bold font.

참고(2장): Selection of Columns (Attributes)

Projection

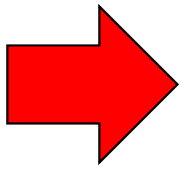
Extract data Vertically (i.e. as a column)

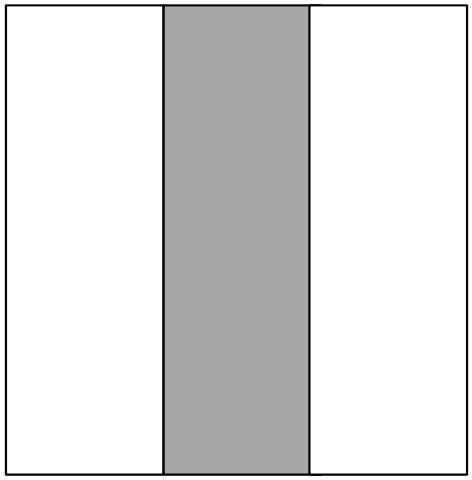
Chooses some of the attributes of the relation

(i.e. columns of the table)

Product Relation

Product Name	*Unit Price*
Melon	*5,000*
Strawberry	*3,000*
Apple	*2,000*
Lemon	*1,500*





Project Product Name
OR
Select Product Name
From Product

Melon
Strawberry
Apple
Lemon

The select Clause (Cont.)

SQL allows duplicates in relations as well as in query results.

To force the elimination of duplicates, insert the keyword **distinct** after select.

Find the names of all departments with instructor, and remove duplicates

select distinct *dept_name*
from *instructor*

The keyword **all** specifies that duplicates not be removed.

select all *dept_name*
from *instructor*

distinct 나 **all**을 쓰지 않으면? 즉, default는 ?

→ Oracle에서는 중복된 것이 있으면 모두 나온다. 즉, **all**이 default로 되어 있음

The select Clause (Cont.)

Student

ID	First	Last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

```
SELECT ALL Last
  FROM Student
```

Last
Smith
Jones
Brown
Jones
Brown

Sometimes you end up with duplicate entries

Using **`DISTINCT`** removes duplicates

Using **`ALL`** retains them - this is the default

```
SELECT DISTINCT Last
  FROM Student
```

Last
Smith
Jones
Brown

The select Clause (Cont.)

An asterisk in the select clause denotes “all attributes”

select *
from *instructor*

The **select** clause can contain arithmetic expressions involving the operation, +, –, ∗, and /, and operating on constants or attributes of tuples.

The query:

select *ID, name, salary/12*
from *instructor*

would return a relation that is the same as the *instructor* relation, except that the value of the attribute *salary* is divided by 12.

The from Clause

The **from** clause lists the relations involved in the query

Corresponds to the Cartesian product operation of the relational algebra.

Find the Cartesian product *instructor X teaches*

select ∗
from *instructor, teaches*

generates every possible instructor – teaches pair, with all attributes from both relations

Cartesian product not very useful directly, but useful combined with where-clause condition (selection operation in relational algebra)

The where Clause

The **where** clause specifies conditions that the result must satisfy

Corresponds to the selection predicate of the relational algebra.

To find all instructors in Comp. Sci. dept with salary > 80000

select *name*
from *instructor*
where *dept_name* = 'Comp. Sci.' **and** *salary* > 80000

Comparison results can be combined using the logical connectives **and, or,** and **not.**

Comparisons can be applied to results of arithmetic expressions.

The where Clause

select *name*
from *instructor*
where *dept_name* = 'Comp. Sci.' **and** *salary* > 80000

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

name
Brandt

The where Clause

Grade

ID	Code	Mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

```
SELECT * FROM Grade
WHERE Mark >= 60
```

ID	Code	Mark
S103	DBS	72
S104	PR1	68
S104	IAI	65
S107	PR1	76
S107	PR2	60

```
SELECT DISTINCT ID
FROM Grade
WHERE Mark >= 60
```

ID
S103
S104
S107