3. SQL

Duplicates, Set Operations

Generated by 이유림

참고 강의 : https://www.edwith.org/database/lecture/27855/

DUPLICATES

- **Duplicates** : 중복된 값이 릴레이션에 존재하는 것
- **Multiset** : Duplicates를 허용하는 집합
- 관계대수 연산의 멀티셋 버전의 결과에는 그 멀티셋이 그대로 유지됨.

DUPLICATES – SELECTION

예) 릴레이션 r1과 r2가 멀티셋일 때

1. $\sigma_\theta(r_1)$: 릴레이션 r1에 t1이라는 튜플이 c1개 존재하고,

   튜플 t1이 셀렉션 연산의 조건을 만족한다면, 셀렉션 연산의 결과에 튜플 t1이 c1개 들어있게 됨.

   (결과에 튜플 t1 하나만 갖는 게 아니라, 모든 t1의 사본이 존재)

DUPLICATES – PROJECTION

예) 릴레이션 r1과 r2가 멀티셋일 때

2. $\Pi_A(r)$: 릴레이션 r1에 튜플 t1에 대한 각각의 복사본에 대하여 **ΠA(t1)**을 하나의 튜플에 대한 프로젝션이라고 했을 때,
전체 릴레이션 프로젝션 연산 결과에는 하나의 튜플 t1을 프로젝션한 결과(**ΠA(t1)**)에 대한 사본이 들어있게 됨.

DUPLICATES – CARTESIAN PRODUCT

예) 릴레이션 r1과 r2가 멀티셋일 때

3. r1 X r2 : 릴레이션 r1에 튜플 t1 (c1개)

릴레이션 r2에 튜플 t2 (c2개)

r1 X r2의 결과에는 t1-t2 사본이 c1*c2개 있음

SQL DUPLICATES

select A1, A2, …, An

from r1, r2, …, rm

where *P*

위의 select, from, where 절은 아래의 관계 대수 표현의 멀티셋 버전과 동일함.

$$\prod_{A_1, A_2, \ldots, A_n} (\sigma_P (r_1 \times r_2 \times \ldots \times r_m))$$

[SQL 쿼리 – 관계대수 연산 매핑]

= from 절 : Cartesian product / **where 절 :** selection / **select** 절 : projection

(멀티셋 버전이기 때문에 모든 연산의 결과는 duplicates를 가질 수 있음)

SQL DUPLICATES

select A1, A2, ..., An

from r1, r2, ..., rm

where *P*

$$\prod_{A_1, A_2, \ldots, A_n} (\sigma_P (r_1 \times r_2 \times \ldots \times r_m))$$

1. **select** 절 : select 절에 A1, ..., An까지 원하는 attribute들의 이름을 적는 것
   = 각각의 attribut들을 가져오는 projection 연산 $\prod_{A_1, A_2, \ldots, A_n}$
2. **from** 절 : from 절에 r1, r2, ..., rm까지 사용할 릴레이션을 적는 것
   = 릴레이션들을 cartesian product 하는 것 $r_1 \times r_2 \times \ldots \times r_m$
3. **where** 절 : where 절에 조건 P를 적는 것
   = 만들어진 결과 릴레이션에 조건 P를 만족하는 selection 연산 σ_P

SET OPERATIONS (1)

- **Set Operations** : 집합 연산
- **Union**(합집합), **intersect**(교집합), **except**(차집합)

  : 자동으로 중복 제거
- **Union all, intersect all, except all** (멀티셋 버전)

  : 멀티셋 버전, 중복된 값을 유지

SET OPERATIONS (2)

- 멀티셋 버전의 집합연산을 통해 중복된 튜플의 개수를 예측할 수 있음

예) 릴레이션 r에 m번 중복, s에 n번 중복된 튜플이 있는 경우

1. r **union all** s : 결과에 m+n개의 중복된 튜플을 가짐. **m+n**

(두 릴레이션의 튜플들을 모두 합치는 연산 수행)

2. r **intersect all** s : m, n 중 최솟값만큼 중복 (중복 x라면, 1의 값을 가짐) **min(m, n)**

(릴레이션에 공통으로 들어있는 튜플을 구하는 연산)

3. r **except all** s : m-n개의 중복된 튜플 가짐. **max(0, m-n)**

m-n<0 라면 0개의 중복된 튜플을 가짐. (중복된 튜플 없음)

(릴레이션에 공통으로 들어있는 튜플을 제외한 튜플들을 구함)

SET OPERATIONS – 연산 수행 예시

아래의 *Section* 릴레이션에 대한 연산 수행 예시 (다음장부터 계속)

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

SET OPERATIONS – 연산 수행 예시 1

Relation c1

select course_id

from *section*

where semester='Fall' **and** year=2009

Relation c2

select course_id

from *section*

where semester='Spring' **and** year=2010

course_id
CS-101
CS-347
PHY-101

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

SET OPERATIONS – 연산 수행 예시 1

예) 2010년 봄과 **(or)** 2009년 가을에 개설된 과목을 찾기 위한 집합 연산

= union (합집합) 연산

(**select** course_id **from** *section* **where** semester='Fall' **and** year=2009)

union

(**select** course_id **from** *section* **where** semester='Spring' **and** year=2010)

↑

위의 세 줄 : 2010년 봄과 2009년 가을에 개설된 과목을 찾기 위한 쿼리

참고) 멀티셋 버전이 아니므로 중복이 제거됨(멀티셋버전: **union all**)

SET OPERATIONS – 연산 수행 예시 2

예) 2009년 가을과 2010년 봄 둘 다(and)에 개설된 과목을 찾기 위한 연산

= intersect (교집합) 연산

(**select** course_id **from** *section* **where** semester='Fall' **and** year=2009)

intersect

(**select** course_id **from** *section* **where** semester='Spring' **and** year=2010)

↑

위의 세 줄 : 2010년 봄과 2009년 가을에 개설된 과목을 찾기 위한 쿼리

쿼리 결과

course_id
CS-101

참고) 멀티셋 버전이 아니므로 중복이 제거됨(여기에서는 중복 x, 제거는 안함)

SET OPERATIONS – 연산 수행 예시 3

예) 2009년 가을학기에는 있지만(not but) 2010년 봄학기에는 개설되지 않는 수업 찾기

2009년 가을학기 수업 중 2010년 봄학기 수업을 모두 빼면 됨 = except (차집합) 연산

(**select** course_id **from** *section* **where** semester='Fall' **and** year=2009)

except

(**select** course_id **from** *section* **where** semester='Spring' **and** year=2010)

쿼리 결과

course_id
CS-347
PHY-101

[쿼리의 결과]

= 첫 번째 예시에서 원하는 결과 중 두 번째 예시의 결과에 나타나지 않은 작업들임.

[차집합 연산의 특징]

= 연산을 수행하기 전 중복을 제거함 (제거하지 않으면 올바른 값 X)