데이터베이스

관계 연산자

학습목표

- 일반 집합 연산자의 특징을 설명할 수 있다.
- 순수 관계 연산자의 특징을 설명할 수 있다.

학습내용

- 일반 집합 연산자
- 순수 관계 연산자

관계 대수 연산자

일반 집합 연산자

- 형식적 관계형 데이터 모델에서 릴레이션은 튜플들의 집합
- 튜플의 집합이기 때문에 유래된 연산자
- 합집합, 교집합, 차집합, 카티션 곱

순수 관계 연산자

- 관계형 데이터 모델을 위해 특별히 제안된 연산자들

추가적인 연산자

- 기본적인 관계 연산자들 이외에 통계치와 같은 사용의 편리성을 위하여 추가된 연산자들

일반 집합 연산자

합집합, 교집합, 차집합

01 합집합(Union)

- 집합 연산 기호 ∪로 표현
- 이항 연산자
  - 2개의 릴레이션을 피연산자로 가짐
- R ∪ S
  - 두 릴레이션 R과 S의 합집합
  - R ∪ S는 릴레이션 R 또는 S에 속하는 튜플들의 집합

$$R \cup S = \{ t \mid t \in R \vee t \in S \}$$
$$\max(|R|, |S|) \leq |R \cup S| \leq |R| + |S|$$

일반 집합 연산자

합집합, 교집합, 차집합

02 교집합(Intersect)

- 집합 연산 기호 ∩로 표현
- 이항 연산자
  - 2개의 릴레이션을 피연산자로 가짐
- R ∪ S
  - 두 릴레이션 R과 S의 교집합
  - R ∩ S는 릴레이션 R과 S에 공통으로 존재하는 튜플들의 집합

$$R \cap S = \{ t \mid t \in R \wedge t \in S \}$$
$$0 \le |R \cap S| \le \min\{ |R|, |S| \}$$

일반 집합 연산자

합집합, 교집합, 차집합

03 차집합(Difference)

- 집합 연산 기호 -로 표현
- 이항 연산자
  - 2개의 릴레이션을 피연산자로 가짐
- R - S
  - 두 릴레이션 R과 S의 차집합
  - R - S는 릴레이션 R에는 존재하지만 S에는 존재하지 않는 튜플들의 집합

$$R - S = \{ t \mid t \in R \wedge t \notin S \}$$
$$0 \leq |R - S| \leq |R|$$

일반 집합 연산자

합집합, 교집합, 차집합

04 합집합과 교집합

- 연산의 결과 또한 릴레이션(튜플들의 집합)임으로 중복이 제거됨
- 교환 법칙(Commutative Operation) 성립
  - $R \cup S = S \cup R$
  - $R \cap S = S \cap R$
- 결합 법칙(Associative Operation) 성립
  - $R \cup (S \cup T) = (R \cup S) \cup T$
  - $R \cap (S \cap T) = (R \cap S) \cap T$

05 차집합과 교집합

- 차집합: 교환법칙이 성립하지 않음
  - $R - S \neq S - R$
- 교집합: 합집합과 차집합으로 표현 가능
  - $R \cap S = R \cup S - (R - S) - (S - R)$

일반 집합 연산자

합집합, 교집합, 차집합

06 합병 호환성(Union Compatibility)

- ∪, ∩, - 연산의 피연산자(릴레이션)들이 지켜야 할 제약 조건
  - 릴레이션은 튜플의 집합
  - 집합은 동종의 원소(Homogeneous Elements)들로 구성
    - ∪, ∩, - 과 같은 연산의 동종의 원소들로 구성된 릴레이션 쌍을 피연산자로 가져야 함
  - 폐쇄 성질: 연산의 결과 또한 릴레이션이어야 함

- 피연산자의 제약조건

01	차수(Degree: 속성의 수)가 같아야 함
02	대응되는 애트리 뷰트 쌍별로 타입(또는 도메인)이 같아야 함
03	대응되는 애트리 뷰트 쌍별로 의미(Semantic)가 같아야 함

  - 1,2 조건: 시스템에서 확인해줌
  - 3 조건: 사람이 해야 함

    예 | 몸무게(실수) U 키(실수)
    - 1, 2 조건은 만족, 그러나 몸무게와 키를 합친 결과는 의미 없음

일반 집합 연산자

카티션 곱

01 카티션 곱(Cartesian Product)

- 집합 연산 기호 ×로 표현
- 크로스 조인(Cross Join)이라고도 불림
- 이항 연산자
  - 2개의 릴레이션을 피연산자로 가짐
  - 다른 일반 집합 연산자와 달리 합병 호환성이 필요하지 않음
- R × S
  - R에 속한 모든 튜플들과 S에 속한 모든 튜플들을 결합하여 새로운 튜플을 생성

02 카티션 곱의 적용

- 단독으로 사용하는 경우에는 별 의미가 없지만 이 결과에 대하여 다른 연산자를 적용하면 유용함
- $R(a_1,..., a_n)$ 와 $S(b_1,..., b_m)$일 때
  - R × S의 결과 릴레이션의 스키마는$(a_1,..., a_n, b_1,...,b_m)$임

$$R \times S = \{ r \cdot s \mid r \in R \wedge s \in S \}$$

여기서, ·: 접속(Concatenation)을 의미

$$|R \times S| = |R| \times |S|$$

차수(Degree) = R의 차수 + S의 차수

순수 관계 연산자

순수 관계 연산자

일반 집합 연산자들과 달리
릴레이션이 2차원 테이블 구조로 제안된 연산자

- 셀렉트 (Select)
- 프로젝트 (Project)
- 조인 (Join)
- 디비전 (Division)
- 개명 연산자 (Rename)

※ 집단 연산자(Aggregation Operator)
초기 관계 연산자에는 속하지 않았으나 추후 필요성에 따라 추가됨

순수 관계 연산자

셀렉트와 프로젝트

01 셀렉트(Select)

- 일항 연산자
  - 피연산자로 릴레이션 하나를 가짐
- 릴레이션에서 선택조건을 만족하는 튜플들의 집합을 선택
  - 주어진 조건을 만족하는 튜플들만 걸러내는 연산
- 표기법

$\sigma_{선택조건}(R)$

- σ: 셀렉트 연산자
- 선택조건: 릴레이션 R의 속성들에 대한 논리식
  ※ 선택조건은 릴레이션 R의 각 튜플에 개별적으로 적용

- $\sigma_{선택조건}(R)$의 차수는 R과 같음
- $0 \le |\sigma_{선택조건}(R)| \le |R|$

» 예 | 학생 중 학번이 100인 학생 튜플을 구하시오.

$\sigma_{학번=100}(학생)$

순수 관계 연산자

셀렉트와 프로젝트

01 셀렉트(Select)

- $0 \le |\sigma_{\text{선택조건}}(R)| \le |R|$
  - 선택율(Selectivity): 릴레이션 R에서 선택조건에 의하여 선택된 튜플 수의 비율
- 셀렉트 연산은 교환적(Commutative)임
  - $\sigma_{\text{조건1}}(\sigma_{\text{조건2}}(R)) = \sigma_{\text{조건2}}(\sigma_{\text{조건1}}(R)) = \sigma_{\text{조건1} \wedge \text{조건2}}(R)$

» 예 | 학생 중 학과가 컴공이고 학년이 4인 학생 튜플들을 구하시오.

$$\sigma_{\text{학과} = \text{'컴공'}}(\sigma_{\text{학년} = 4}(\text{학생})) = \sigma_{\text{학년} = 4}(\sigma_{\text{학과} = \text{'컴공'}}(\text{학생})) = \sigma_{\text{학과} = \text{'컴공'} \wedge \text{학년} = 4}(\text{학생})$$

- 선택율이 작은 셀렉트 연산을 먼저 수행하는 것이 좋음

순수 관계 연산자

셀렉트와 프로젝트

02 프로젝트(Project)

- 일항 연산자
  - 피연산자로 릴레이션 하나를 가짐
- 릴레이션에서 일부 속성(즉, 열)을 선택하고 나머지는 버리는 연산
  - 릴레이션에서 관심있는 속성들만을 골라내는 연산
- 표기법

$\Pi_{속성리스트}(R)$

- Π: 프로젝트 연산자
- 속성리스트: 릴레이션 R에서 선택하고자 하는 속성들의 리스트

순수 관계 연산자

셀렉트와 프로젝트

02 프로젝트(Project)

- $\Pi_{속성리스트}(R)$의 차수는 속성리스트의 차수와 같음
- $|\Pi_{속성리스트}(R)| \leq |R|$
  - 왜 ≤ 일까?
    즉 = 이어야 하지 않나?
    - 릴레이션은 튜플의 집합
    - 모든 관계 연산자는 폐쇄 성질을 만족
      - ➡ 결과도 릴레이션임
      - ➡ 결과에 속한 중복 튜플은 모두 제거됨
- $|\Pi_{속성리스트}(R)| \leq |R|$

» 예 | 학생테이블에서 성별을 추출하시오.

$$\Pi_{성별}(학생) = \{'남', '여'\}$$

- 학생 튜플 수가 100개이어도 결과는 2개

- 프로젝트 연산에서 교환법칙이 성립하지 않음
- $\Pi_{리스트1}(\Pi_{리스트2}(R)) = \Pi_{리스트1}(R)$

» 예 | 학생테이블에서 이름을 추출하시오.

$$\Pi_{이름}(\Pi_{학번,이름,주소}(R)) = \Pi_{이름}(R)$$

순수 관계 연산자

셀렉트와 프로젝트

03 셀렉트와 프로젝트

- σ과 Π만 있으면 단일 테이블에서 원하는 정보를 충분히 추출할 수 있음
  - σ를 이용하여 원하는 튜플들의 (행) 선택
  - Π를 이용하여 선택된 튜플들의 속성(열)들 추출

예 | 학생 중 학번이 300인 학생의 이름과 성적은?

$\Pi_{\text{이름,성적}}(\sigma_{\text{학번=300}}(\text{학생}))$

학생

이름	학번	학과	성적
김홍연	100	컴공	2.8
강성민	200	컴공	3.9
빈준길	300	전자	2.3
이석주	400	기계	3.5

순수 관계 연산자

셀렉트와 프로젝트

03 셀렉트와 프로젝트

예 | 학생 중 컴공학과 학생이고, 성적이 3.0 이상인 학생의 이름을 조회하시오.

$\Pi_{\text{이름}}(\sigma_{\text{성적}>=3.0}(\sigma_{\text{학과}='\text{컴공}'}(\text{학생}))$
$=\Pi_{\text{이름}}(\sigma_{\text{학과}='\text{컴공}'}(\sigma_{\text{성적}>=3.0}(\text{학생}))$

학생

이름	학번	학과	성적
김홍연	100	컴공	2.8
강성민	200	컴공	3.9
빈준길	300	전자	2.3
이석주	400	기계	3.5

학생

이름	학번	학과	성적
김홍연	100	컴공	2.8
강성민	200	컴공	3.9
빈준길	300	전자	2.3
이석주	400	기계	3.5

순수 관계 연산자

디비전

01 디비전(Division)

- 이항 연산자: 피연산자로 릴레이션 2개를 가짐
- 표기법: R ÷ S

제약사항	- 두 릴레이션 $R(a_1,..., a_n)$, $S(b_1,..., b_m)$에 대하여, $\{b_1,..., b_m\} \subseteq \{a_1,..., a_n\}$
의미	- S에 속한 모든 튜플과 연관이 있는 R 튜플을 찾는 것 - 주로 "모든 ~에 대한"이라는 질의에 적합
수학적 정의	- $R \div S = \{t \mid t \in \Pi_C(R) \wedge t \cdot s \in R \text{ for all } s \in S\}$ - where, $c = \{a_1,..., a_n\} - \{b_1,..., b_m\}$

순수 관계 연산자

디비전

02 연산 (예)

동일한 이름(개똥이)을 지니는 학생들의 학번을 선별해 내는 연산

학생

학번	이름
100	개똥이
200	소똥이
300	말똥이
400	개똥이

÷

학생 이름

이름
개똥이

=

학번
100
400

- 관계형 데이터베이스 표준 질의문인 SQL에서는 직접적으로 지원하지 않음

순수 관계 연산자

디비전

03 ÷ 연산

- 프로그래밍 언어의 정수형 나누기와 유사
  - 5/2 = 2(정수형 나누기이기 때문에 소수점 이하는 버림)
  - (5/2) × 2 = 4 ≠ 5

- (R ÷ S) × S ⊆ R
  - R을 S로 나누고 다시 S로 카티션 프로젝트를 하면 R로 복원되지 않을 수 있음

학생

학번	이름
100	개똥이
200	소똥이
300	말똥이
400	개똥이

÷

학생 이름

이름
개똥이

=

학번
100
400

학번
100
400

×

학생 이름

이름
개똥이

=

학생

학번	이름
100	개똥이
400	개똥이

순수 관계 연산자

개명 연산자

01 개명 연산자(Rename Operator)

- 관계 대수 연산 결과인 릴레이션은 이름을 가지지 않아서 결과 릴레이션을 참조하기 어려운 경우가 있음
- 중간 결과물에 대한 이름을 지정하고 싶을 때 개명 연산자를 사용
- 표기법

$\rho_s(E)$	- 관계 연산식 E의 결과 릴레이션 이름을 S로 지정
$\rho_{s(B1,B2,...,Bn)}(E)$	- 관계 연산식 E의 결과 릴레이션 이름을 S로 지정하고 각 속성을 $B_1,...,B_n$이라는 이름으로 지정
$\rho_{(B1,B2,...,Bn)}(E)$	- 관계 연산식 E의 결과 릴레이션의 속성들만 $B_1,...,B_n$이라는 이름으로 지정

순수 관계 연산자

개명 연산자

02 연산 (예)

- 아래 연산 결과를 RESULT라 하자.

$$RESULT \leftarrow \pi_{FNAME, LNAME, SALARY}(EMPLOYEE)$$

  - RESULT는 FNAME, LNAME, SALARY를 속성으로 가짐

- 이는 다음과 같이 간단히 할 수 있다.

$$\rho_{RESULT}(\pi_{FNAME, LNAME, SALARY}(EMPLOYEE))$$

- 속성 이름도 A,B,C라고 하고 싶다.

$$\rho_{RESULT(A,B,C)}(\pi_{FNAME, LNAME, SALARY}(EMPLOYEE))$$

순수 관계 연산자

집단 연산자

01 집단 연산 (예1)

- 기본적인 관계 연산만으로는 원하는 릴레이션을 적용할 수 없어 추가된 연산 중 하나로 릴레이션에 저장된 튜플들의 통계정보를 추출하고 싶을 경우
- 기본 문법

$F_{\text{집단 함수리스트}}(R)$

- F: 집단 연산자
- 집단 함수리스트: F(속성),..., F(속성)
- F( )는 SUM, MAX, MIN, AVG, COUNT 등과 같은 집단 함수

예 | 학생 테이블의 성적 속성의 평균을 출력하시오.

$$F_{AVG(score)}(STUDENT)$$

순수 관계 연산자

집단 연산자

01 집단 연산 (예2)

- 테이블 전체에 대한 통계 정보보다 테이블에 속한 튜플들을 작은 그룹들로 나누고 각 그룹의 통계정보를 파악하고 싶은 경우
- 표기법

그룹화 속성리스트 F 집단 함수리스트 (R)

- F: 집단 연산자
- 집단 함수리스트: F(속성),..., F(속성)

예 | 학년별 최고 성적 및 평균 성적

$$_{year}F_{MAX(score)AVG(score)}(STUDENT)$$

1 일반 집합 연산자

- ✓ 합집합, 교집합, 차집합, 카티션 곱
- ✓ ∪, ∩, - 연산의 피연산자(릴레이션)들은 합병 호환성을 만족해야 함

  1.차수(Degree: 속성의 수)가 같아야 함

  2. 대응되는 애트리 뷰트 쌍별로 타입(또는 도메인)이 같아야 함

  3. 대응되는 애트리 뷰트 쌍별로 의미(Semantic)이 같아야 함

2 순수 관계 연산자

- ✓ 셀렉트: 릴레이션의 수평적 부분집합
- ✓ 프로젝트: 릴레이션의 수직적 부분집합
- ✓ 디비전: 릴레이션 S에 속한 모든 튜플들과 연관이 있는 릴레이션 R의 튜플을 찾는 것
- ✓ 개명 연산: 연산 중간 결과물에 대한 이름 지정
- ✓ 집단 연산자: 릴레이션에 저장된 튜플들의 통계 정보 추출