관계 대수

Chapter 4, Part A

관계 질의어

- *질의어*: DB내의 데이타를 조작 및 검색
- 관계 모델은 단순하면서도 막강한 질의어들을 제공:
  - 논리학에 기반한 강력한 정형화
  - 다양한 최적화 가능
- 질의어 != 프로그래밍 언어
  - 질의어는 "튜어링식 완전성"을 고려하지 않음.
  - 질의어는 복잡한 계산용이 아님.
  - 질의어는 대규모 데이타 집단에 대한 쉽고 효율적인 접근 제공.

정형 관계 질의어

"실제의" 언어(SQL등)와, 구현 방법의 기반이 되는 수학적인 질의어 형태 2가지:

1. *관계 대수(Relational Algebra)*: 더 절차적. 수행 계획 표현에 적절.
2. *관계 해석(Relational Calculus)*: 원하는 것(what)만 명세하고, 그 방법(how)은 생략. 비절차적, 명세형.

- 대수와 해석의 이해가 곧 SQL과 질의 수행법 이해의 관건!

시작하기 전에

- 질의는 하나 이상의 릴레이션 인스턴스에 적용되어, 하나의 릴레이션 인스턴스를 낸다.
  - 입력 릴레이션의 스키마는 고정(인스턴스는 임의).
  - 결과 릴레이션의 스키마도 고정! 질의상의 정의에 따라 결정됨.
- 위치나 이름으로 표기:
  - 위치 표기법은 정형화가 쉽고, 이름 표기법은 이해하기 쉽다.
  - SQL은 두 방식 모두 사용.

예제 인스턴스

- 앞으로 사용할 "뱃사람"과 "에약"의 인스턴스.
- 위치 표기법과 이름 표기법을 병용할 것이며, 질의 결과의 필드명은 입력 테이블을 본딴다고 가정.

R1

뱃사람 번호	배번호	일자
22	101	10/10/96
58	103	11/12/96

S1

뱃사람 번호	뱃사람 이름	등급	나이
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

뱃사람 번호	뱃사람 이름	등급	나이
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

관계 대수

- 기본 연산자:
  - *셀렉션(Selection)* (σ) 릴레이션에서 투플들을 선별.
  - *프로젝션(Projection)* (π) 필요없는 필드들을 제거.
  - *카티션 프로덕트(Cartesian Product)* ($\times$) 두 릴레이션을 조합.
  - *차집합(Set-difference)* ($-$) 릴1에는 있고 릴2에는 없는 투플.
  - *합집합(Union)* ($\cup$) 릴1과 릴2에 있는 투플들을 모두 모음.
- 기타 연산자:
  - 교집합(intersection), 조인(join), 디비전(division), 개명(renaming): 필수는 아니지만 유용함.
- 연산의 결과가 다시 릴레이션이므로, 연산의 조합이 가능! (대수의 "닫힌" 성질)

프로젝션

- 프로젝션 리스트에 없는 필드들을 삭제.
- 결과 테이블의 스키마는 프로젝션 리스트의 필드들로만 구성되며, 그 이름은 입력 테이블로부터 따온다.
- 프로젝션 연산자는 중복 투플을 없애야 한다!(왜??)
  - Note: 실제 시스템들은, 사용자가 명시하지 않는 한, 중복을 제거하지 않음(왜?)

뱃사람번호	등급
yuppy	9
lubber	8
guppy	5
rusty	10

$\pi_{뱃사람이름, 등급}(S2)$

나이
35.0
55.5

$\pi_{나이}(S2)$

셀렉션

- 셀렉션 조건에 부합되는 투플들만 선별.
- 결과에는 중복이 없다!(왜?)
- 결과의 스키마는 입력 테이블과 동일함.
- 결과 릴레이션도 다른 관계 대수 연산자의입력이 될 수 있다!(연산자 조합)

뱃사람 번호	뱃사람이름	등급	나이
28	yuppy	9	35.0
58	rusty	10	35.0

$\sigma_{등급>8}(S2)$

뱃사람이름	등급
yuppy	9
rusty	10

$\pi_{뱃사람이름, 등급}(\sigma_{등급>8}(S2))$

합집합, 교집합, 차집합

- 모두 입력 테이블은 2개이며, 합병가능(*union-compatible*) 이어야함.
  - 필드의 수가 같고,
  - 대응 필드의 타입이 같다.
- 결과의 스키마는?

뱃사람 번호	뱃사람 이름	등급	나이
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

뱃사람 번호	뱃사람 이름	등급	나이
22	dustin	7	45.0

$S1 - S2$

뱃사람 번호	뱃사람 이름	등급	나이
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

카티션 프로덕트 : $S1 \times R1$

- S1의 각 투플에, R1의 각 투플을 조합.
- *결과 스키마* 는 S1과 R1의 필드들을 모두 한번씩 따온 형태.
  - 충돌: S1과 R1에 모두 뱃사람번호라는 필드존재.

뱃사람번호	뱃사람 이름	등급	나이	뱃사람번호	배번호	일자
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

☛ *개명 연산자*: $\rho(C(1 \rightarrow \text{뱃사람번호}1, 5 \rightarrow \text{뱃사람번호}2), S1 \times R1)$

죠인

- *조건 죠인*: $R \bowtie_c S = \sigma_c (R \times S)$

뱃사람 번호	뱃사람 이름	등급	나이	뱃사람 번호	배번호	일자
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

$$S1 \bowtie_{S1.\text{뱃사람번호} < R1.\text{뱃사람번호}} R1$$

- *결과 스키마* 는 카티션 프로덕트와 동일.
- 카티션 프로덕트보다 결과의 양이 적으므로, 더 효율적인 계산이 가능.
- 세타죠인*(theta-join)* 이라고도 함.

죠인(계속)

- *동등죠인(Equi-Join)*: 조건 죠인 중에서 조건이 '='인 경우.

뱃사람 번호	뱃사람 이름	등급	나이	뱃사람 번호	배번호	일자
22	dustin	7	45.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

$$S1 \bowtie_{\text{뱃사람번호}} R1$$

- *결과 스키마*는 카티션 프로덕트와 비슷하지만, 비교 필드는 한번씩만 나온다.
- *자연 죠인(Natural Join)*: 모든 공통 필드에 대한 동등 죠인.

디비젼

- 기본 연산자는 아니지만, 다음과 같은 경우에 효율적:
  모든 배를 예약한 뱃사람들을 구하라.
- A는 x와 y, B는 y로 구성된다면:
  - $A/B = \{\langle x\rangle \mid \exists\langle x,y\rangle \in A \ \forall\langle y\rangle \in B\}$
  - 즉, ***A/B***는 ***B***의 *모든* ***y*** 투플(배)에 대하여, ***xy*** 투플이 ***A***내에 존재하는, 그러한 ***x***투플(뱃사람)들의 모임.
  - *또는*: A에서 한 x값(뱃사람)에 대응하고 있는 y값(배)집합이 B의 모든 y값으로 구성되면, 그 x는 A/B에 포함.
- 여기에서 x와 y는 어떤 필드(집합)이 되어도 무관함. y는 B의 필드집합이며, $x \cup y$ 는 A의 필드집합이다.

디비젼 A/B의 예

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

sno
s1
s2
s3
s4

A/B1

pno
p2
p4

B2

sno
s1
s4

A/B2

pno
p1
p2
p4

B3

sno
s1

A/B3

A/B를 기본 연산자들로 분해하면

- 디비전은 필수 연산자가 아님. 표기에 편리할 뿐.
  - (죠인도 마찬가지이지만, 대부분 별도로 구현하고 있음.)
- *착상*: 먼저, B의 y중에 부합되지 않는 것이 나타나는 x값들을 구한다.
  - B의 y들과 조합한 xy중에 A에 나타나지 않는 쌍이 발견되면, 그 x는 '부합되지 않는' 것이다.

부합되지 않는 x : $\pi_x((\pi_x(A)\times B)-A)$

A/B: $\pi_x(A)$ – 부합되지 않는 투플

배번호 103을 예약한 뱃사람의 이름을 구하라

- 방법 1: $\pi_{\text{뱃사람이름}}((\sigma_{\text{배번호}=103}{}^{\text{예약}}) \rhd\lhd \text{뱃사람})$
- 방법 2 : $\rho(Temp1, \sigma_{\text{배번호}=103}{}^{\text{예약}})$

  $\rho(Temp2, Temp1 \rhd\lhd \text{뱃사람})$

  $\pi_{\text{뱃사람이름}}(Temp2)$
- 방법 3 : $\pi_{\text{뱃사람이름}}(\sigma_{\text{배번호}=103}(\text{예약} \rhd\lhd \text{뱃사람}))$

적색 배를 예약한 뱃사람의 이름을 구하라.

- 배의 색상정보는 배 테이블에 있으므로, 죠인이 추가적으로 필요함:

  $\pi_{\text{뱃사람이름}}((\sigma_{\text{색상}='red'}{}^{\text{배}}) \rhd\lhd^{\text{예약}} \rhd\lhd \text{뱃사람})$

- 더 효율적인 방법:

$\pi_{\text{뱃사람이름}}(\pi^{\text{뱃사람번호}}((\pi_{\text{배번호}}\sigma_{\text{색상}='red'}{}^{\text{배}}) \rhd\lhd \text{예약}) \rhd\lhd \text{뱃사람})$

☛윗 형태가 입력되면 최적화기가 두번째 형태를 구할 수 있다!

적색 또는 녹색 배를 예약한 뱃사람의 이름을 구하라.

- 적색 배나 녹색 배를 모두 구한 후, 이중에 한 척을 예약한 사람들을 구한다.

  $\rho(Tempboats, (\sigma_{\text{색상}='red' \vee \text{색상}='green'}{}^{\text{배}}))$

  $\pi_{\text{뱃사람이름}}(Tempboats \rhd\lhd \text{예약} \rhd\lhd \text{뱃사람})$
- 합집합 연산으로 Tempboats를 구할 수도있다!(어떻게?)
- $\vee$를 $\wedge$로 바꾸면?

적색 배와 녹색 배를 예약한 뱃사람들을 구하라.

- 앞의 방식으로는 안됨! 적색 배를 예약한 뱃사람들을 구하고, 녹색 배를 예약한 뱃사람들을 구한 후, 교집합한다. (뱃사람의 키는 *뱃사람 번호*임을 이용):

$$\rho(Temped, \pi_{\text{뱃사람번호}}((\sigma_{\text{색상}='red'}{}^{\text{배}}) \rhd\lhd {}^{\text{예약}}))$$

$$\rho(Tempgreen, \pi_{\text{뱃사람번호}}((\sigma_{\text{색상}='green'}{}^{\text{배}}) \rhd\lhd {}^{\text{예약}}))$$

$$\pi_{\text{뱃사람이름}}((Tempred \cap Tempgreen) \rhd\lhd {}^{\text{뱃사람}})$$

모든 배를 예약한 뱃사람의 이름을 구하라.

- 디비전을 이용한다. 그러러면 입력 릴레이션의 스키마를 잘 만들어야 한다.

$$\rho(Tempsids, (\pi_{\text{뱃사람번호}, \text{배번호}}{}^{\text{예약}}) / (\pi_{\text{배번호}}{}^{\text{배}}))$$

$$\pi_{\text{뱃사람이름}}(Tempsids \rhd\lhd {}^{\text{뱃사람}})$$

- 'Interlake'라는 배를 모두 예약한 사람을 구하려면?

..... $/\pi_{\text{배번호}}(\sigma_{\text{배이름}=Interlake}{}^{\text{배}})$

요약

- 관계 모델에서는 질의어가 단순하면서도 막강함, 그 정의는 엄밀하다.
- 관계 대수는 더 절차적이며, 질의 수행 전략 표현에 적절
- 같은 질의를 여러가지로 표현 가능. 그 중에서 가장 효율적인 형태를 최적화기가 찾는다.