**- Chapter 6: Formal Relational Query Languages**

**- Relational Algebra**

**- Select Operation**

기호: . P는 selection predicate라고 부름.

Join is a boy. 에서 boy(J)라고 나타낼 때 boy가 selection predicate.

다음과 같이 정의됨: . T는 r의 tuple이면서 p 조건을 만족하는 값 이라고 풀어 쓸 수 있음.

각각의 term은: <attribute> op <attribute> or <constant> 로 나타낼 수 있음. 이 때 op는 =, /=, >, >=, <, <= 중 하나이고, 전체의 formula는 이 term들을 and, or, not으로 연결한 것임.

예시: branch-name이 “Perryridge”인 tuple들.

**- Project Operation**

기호: . A\_k는 attribute의 이름들이고 r은 relation의 이름.

결과는 명시되어 있지 않은 column들을 제외한 k개의 column으로 이루어짐. 그리고 중복된 row는 제거해야 함. => **relation이 set이기 때문**

예시:  account에서 branch-name attribute를 제거.

**- Union Operation**

기호: 

다음과 같이 정의됨: 

r U s 가 valid하기 위해서는 (Union compatible)

1. r, s가 같은 arity를 가져야 한다. (attribute의 개수가 같아야 함)
2. attribute domain들은 compatible 해야 함.

Ex) r의 2번째 column은 s의 2번째 column과 같은 형태의 값을 다루어야 함.

예시:  project한 각각의 결과를 union.

Depositor relation에서 customer-name column과 borrower relation의 customer-name column을 relation한 값. == account나 loan에서의 모든 고객들을 찾는 것.

**- Set Difference Operation**

기호: 

다음과 같이 정의됨: 

Set difference는 반드시 compatible한 relation들만을 받아야 함.

- r과 s는 반드시 같은 arity를 가져야 함.

- r과 s의 attribute domain들은 반드시 compatible 해야 함.

**- Cartesian-Product operation**

Notation: 

다음과 같이 정의됨: 

Attribute들 r(R)과 s(S)는 disjoint 하다고 가정한다 

그렇지 않으면 rename이 필요하다.

**- Composition of Operations**

하나의 연산의 결과로 또 다른 연산을 수행. 예를 들어 

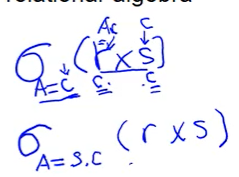
**- Rename Operation**

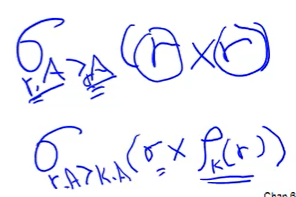
r이나 r X s 같은 relation 혹은 operation의 결과 relation을 k 같은 다른 이름으로 부르고 싶을 때 사용.

Notation: 

만약 relational algebra 표현 E가 n개의 arity를 가지고 있으면,

와 같이 나타내면 expression E의 결과를 N이라는 이름으로 바꾸고, 또 attribute들 이름도 A1, A2, …, An 으로 바꿔서 리턴한다.

 <- cartesian product 후 같은 record가 있으면 이렇게 s.c 같은 식으로 테이블 이름을 붙이면 됨.

Rename은  이렇게 같은 테이블에서 두 개를 join하면 테이블 이름을 붙여도 ambiguity가 사라지지 않으므로 rename operation이 필요함.

**- Example Queries**

**Schema Diagram for University Database는 pdf 참조.**

Find all students in the CS department



Find the name of each student in the CS department



Find the names of all persons who are either an instructor or a student



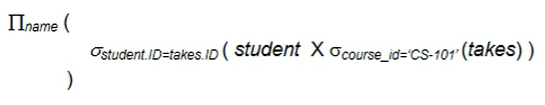
Find the names of all persons who are both an instructor and a student (assuming names are unique)



Find the names of all students who takes/took course CS-101.

Query 1 

이건 먼저 cartesian product를 한 뒤 select, project.

Query 2  이건 먼저 takes에서 select를 한 뒤 cartesian product를 하고 project.

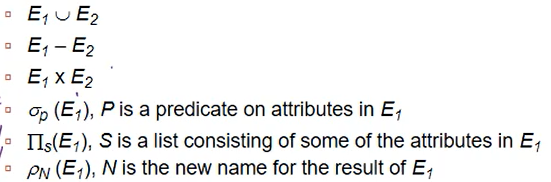
Relational Algebra는Procedural language. Query를 할 때 순서까지 중요하기 때문.

**- Formal Definition**

Basic expression in the relational algebra는 다음 중 하나로 이루어진다.



E1과 E2를 relational-algebra expressions 들이라고 하면 다음은 모두 relational-algebra expression이다.



relational algebra는 formal definition은 6개의 fundamental operation이 있음.

additional operations는 위의 6개로 표현이 가능. (**Set intersection, natural join, assignment)**

**- Set-Intersection Operation**

Notation: 

다음과 같이 정의됨: 

Union compatibility를 가정함.

1. r, s는 같은 arity를 가짐
2. r과 s의 attribute들은 compatible 함.

와 같음.

**- Natural-Join Operation**

Cartesian product를 수행하면서 공유하는 attribute들이 같은 경우에만 결과에 붙여주는 연산.

r(R)과 s(S) 가 있다고 할 때, 로 나타냄.

결과는 r로부터 나온 튜플 t\_r과 s로부터 나온 t\_s의 튜플들의 각각의 쌍으로 얻어진 R U S 스키마의 relation인데, 다음 조건을 만족해야 함.:

t\_r과 t\_s가 R 교집합 S에서 같은 값을 가지고 있으면, tuple t는 결과에 추가됨.



Example



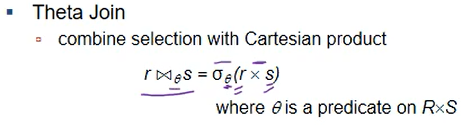
Natural join으로 나타내면 

**- Properties**

 <- A1부터 Ak가 공유 attribute라고 할 때 project후 교집합을 한 것은 natural join 후 project 한 것과 같다.

 <- 교환 법칙, 결합 법칙 성립. Attribute의 순서가 상관이 없기 때문.

,  <- 이 두 개는 당연함.



Natural join을 일반화시켜서 cartesian product 후 selection한 한게 theta join.

세타 조인을 하면 natural join의 형질들은 무시됨.

r 세타조인 c<e (s) 라고 하면 select\_(c<e){r X s} 와 같음.

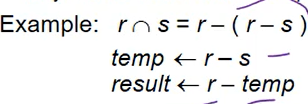
즉 그냥 Cartesian product로 두 relation을 엮은 뒤에 조건을 줘서 원하는 record들만 뽑는 것!

**- Assignment Operation**

복잡한 query들을 편리하게 표현할 수 있게 해줌. Query를 연속된 assignment들로 나타낼 수 있음.

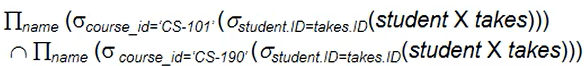
assignment들은 임시의 relation variable로 만들어져야 한다. 그래서 오른쪽의 결과는 왼쪽에 assign되고, 향후 재사용 할 수 있음.

주로 긴 expression을 간단하게 나타내기 위해 사용.

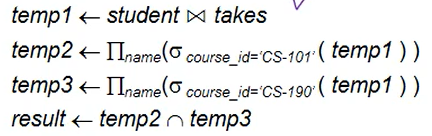


**- Example Queries**

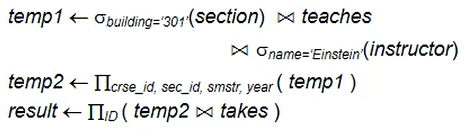
Find the names of all students who take/took both courses CS-101 and CS-190.

Query 1: 

처음에 왜 select에서 and를 안 쓰는지 궁금했는데 생각해보니 동시에 CS-101과 CS-190을 들을 수 없기 때문에 따로 뽑고 교집합을 써준 듯.

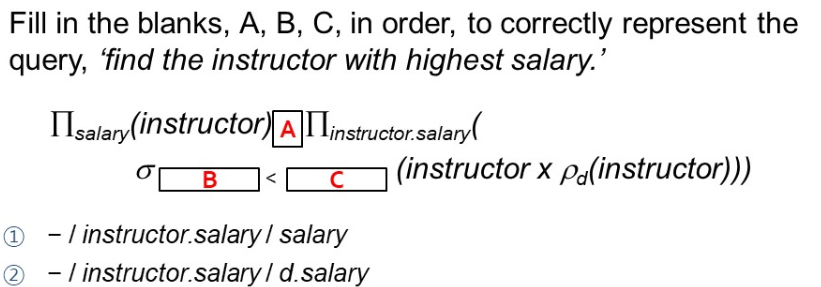
Query 2: 

Find the IDs of all students who were taught by an instructor named Einstein in building 301.



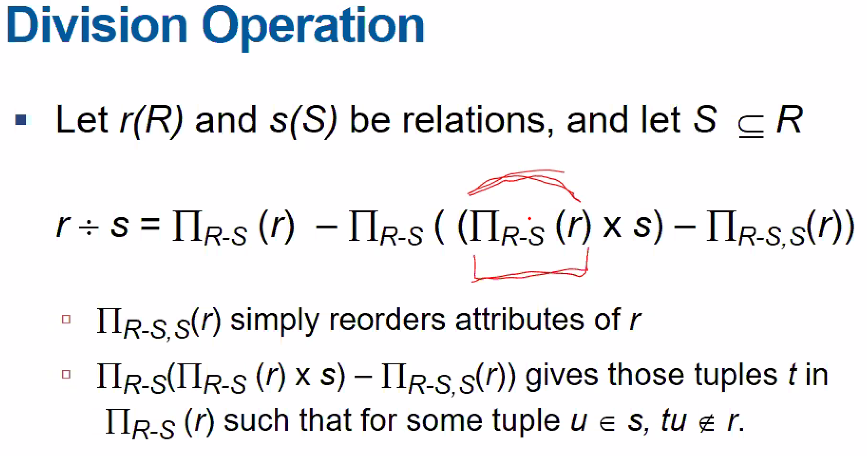
Result를 뽑을 때 temp2를 따로 만든 이유는, natural join은 공유하고 있는 attribute들은 모두 join에 참여하기 때문에, relation 전체를 주면 원하지 않는 효과가 생길 수 있음. 얘를 들어 takes와 temp1 사이에 같은 이름을 가진 attribute가 있거나 하는 경우.

따라서 원하는 attribute들만 project해서 넣어줌.



2번. 전체 salary에서 오른쪽을 set difference하면, 오른쪽은 제일 큰 값을 제외하곤 모두 뽑히게 됨. 그래서 제일 큰 값만 남게 된다.

이거 생각할 땐 연산이 row by row인걸 계속 고려하면 편함!



Division은 r 나누기 s 하면 s는 r의 subset이어야 하고, s를 모두 포함하는 r을 나타냄.



즉 나누기는 큰 relation이 작은 relation이 가진 attribute를 제외하고 다른 attribute를 뽑는데, 뽑히는 기준은 작은 relation의 attribute의 모든 값을 가지고 있어야 함.

Operator 우선순위

