0.1 2.4 An Algebraic Query language

Definition. ***Set operation***

1. ***Union*** *$R \cup S$, the union of R and S, is set of elements that are in R or S or both, element only appear once*
2. ***Intersection*** *$R \cap S$, the intersection of R and S, the set of elements that are in both R and S*
3. ***Difference*** *$R - S$, difference of R and S, is the set of elements that are in R but not in S*

Note R and S must have schemas with identical set of attributes, the types (domains) for each attribute must be same. Also, R and S must be ordered so that the order of attributes is same for both relations

Definition. ***subset operation***

1. ***Projection*** *produce from a relation R a new relation that has some of R's columns.*

$$\pi_{A_1, A_2, \cdots, A_n}(R)$$

is a relation that has only the columns for attributes $A_1, A_2, \cdots, A_n$ of R

2. ***Selection*** *produce from a relation R a new relation with a subset of R's tuples satisfying some condition C that invovles the attributes of R*

$$\sigma_C(R)$$

Definition. ***Renaming*** *applies to a relation R, change the name of relation to S and its attribute name to $A_1, \cdots, A_n$*

$$\rho_{S(A_1, \cdots, A_n)}(R)$$

Definition. ***Relation between operations***

1. *intersection can be expressed in terms of set difference*

$$R \cap S = R - (R - S)$$

2. *theta joins can be expressed by taking selection of a product*

$$R \bowtie_C S = \sigma_C(R \times S)$$

3. *natural join can be expressed by starting with the product then apply selection with a condition C of the form*

$$R.A_1 = S.A_1 \wedge \cdots \wedge R.A_n = S.A_n$$

where $A_1, \cdots, A_n$ are attributes appearing in schemas of both R and S. Finally have to project out one copy of each of the equatedf attributes. Let L be thet list of attributes in shcema of R followed by those attributres in schema of S that are not also in the schema of R

$$R \bowtie S = \pi_L(\sigma_C(R \times S))$$

4. *union, difference, selection, projection, product, renaming forms a set where none can be written in terms of the other five*

Definition. ***linear notation***

1. ***assignment***

$$R(A_1, \cdots, A_n) := < expr >$$

0.2 2.5 Constraint on Relations

Definition. ***Relational algebra as a constriant language***

1. *If R is expression of relational algebra,*

$$R = \emptyset \quad (R \subseteq \emptyset)$$

is a constraint that says no tuples in result of R

2. *if R and S are expressions of relational algebra, then*

$$R \subseteq S \quad (R - S = \emptyset)$$

is a constriant that says every tuple in result of R must also be in result of S

Definition. ***Referential Integrity Constriants*** *asserts one value appearing in one context also appears in another, related context. In general, if we have any value v (maybe be represented by ¿1 attribute) as the component in attribute A of some tuple in relation R, then because of design intentions we may expect that v appear in a particular component (for attribute B) of some tuple of another relation S. We can express this integrity constraint as*

$$\pi_A(R) \subseteq \pi_B(S) \iff \pi_A(R) - \pi_B(S) = \emptyset$$

Definition. ***Key Constraint*** *constraints that a set of attributes is a key for a relation (i.e. no two tuple agree on the key)*

$\rho_{MS1}(name, address, gender, birthdate)(MovieStar) \quad \rho_{MS2}(name, address, gender, birthdate)(MovieStar)$

$$\rho_{MS1.name=MS2.name \wedge MS1.address!=MS2.address}(MS1 \times MS2) = \emptyset$$

represents $(name, address)$ is the key for $MovieStar$

Definition. ***Domain constriant*** *Want to enforce a type constrinat on values of a particular attribute, i.e. integer only.*

$$\sigma_{gender!=F \wedge gender!='M'}(MovieStar) = \emptyset$$