DATA BASE SYSTEMS

LECTURE 5

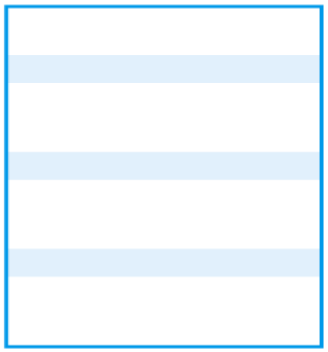
PROPOSED BY

DR: ALSHAIMAA MOSTAFA MOHAMMED

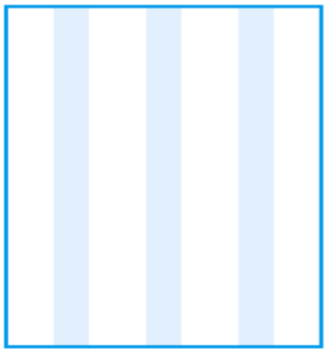
CHAPTER 5

RELATIONAL ALGEBRA AND CALCULUS

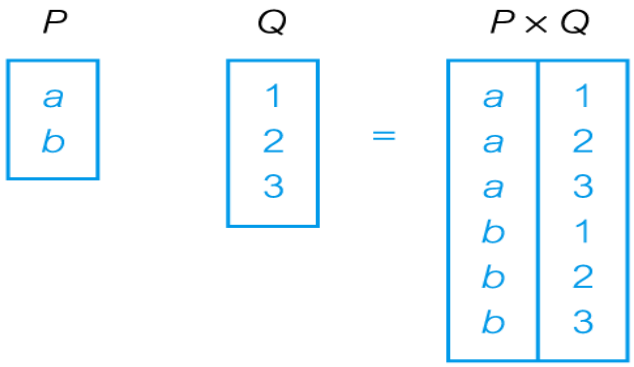
RELATIONAL ALGEBRA OPERATIONS



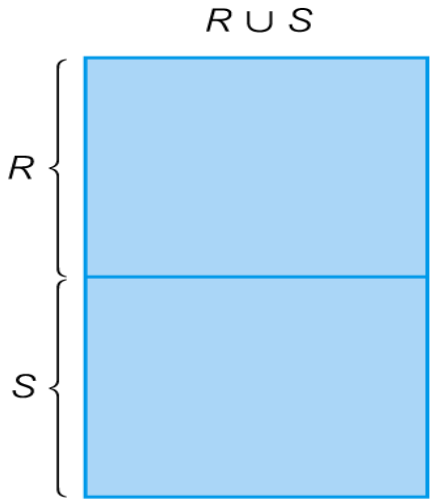
(a) Selection



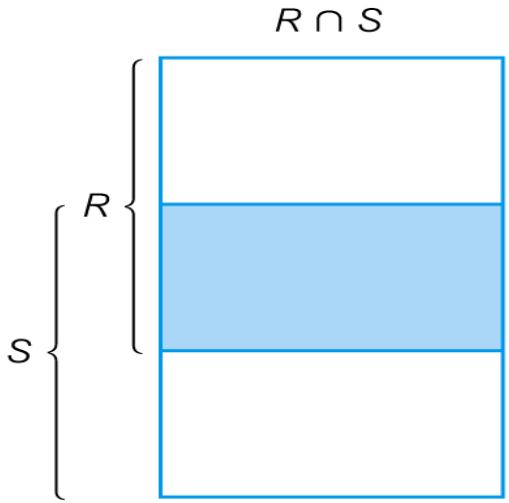
(b) Projection



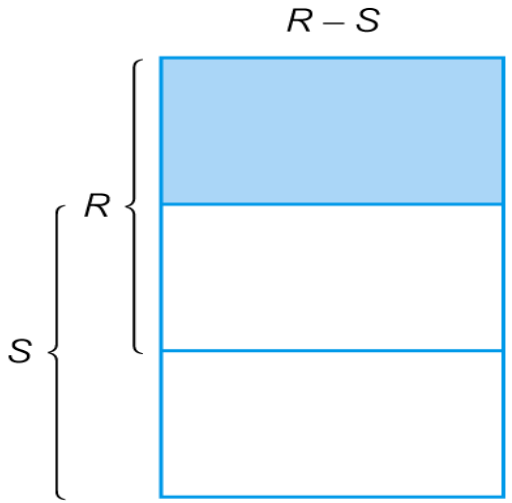
(c) Cartesian product



(d) Union

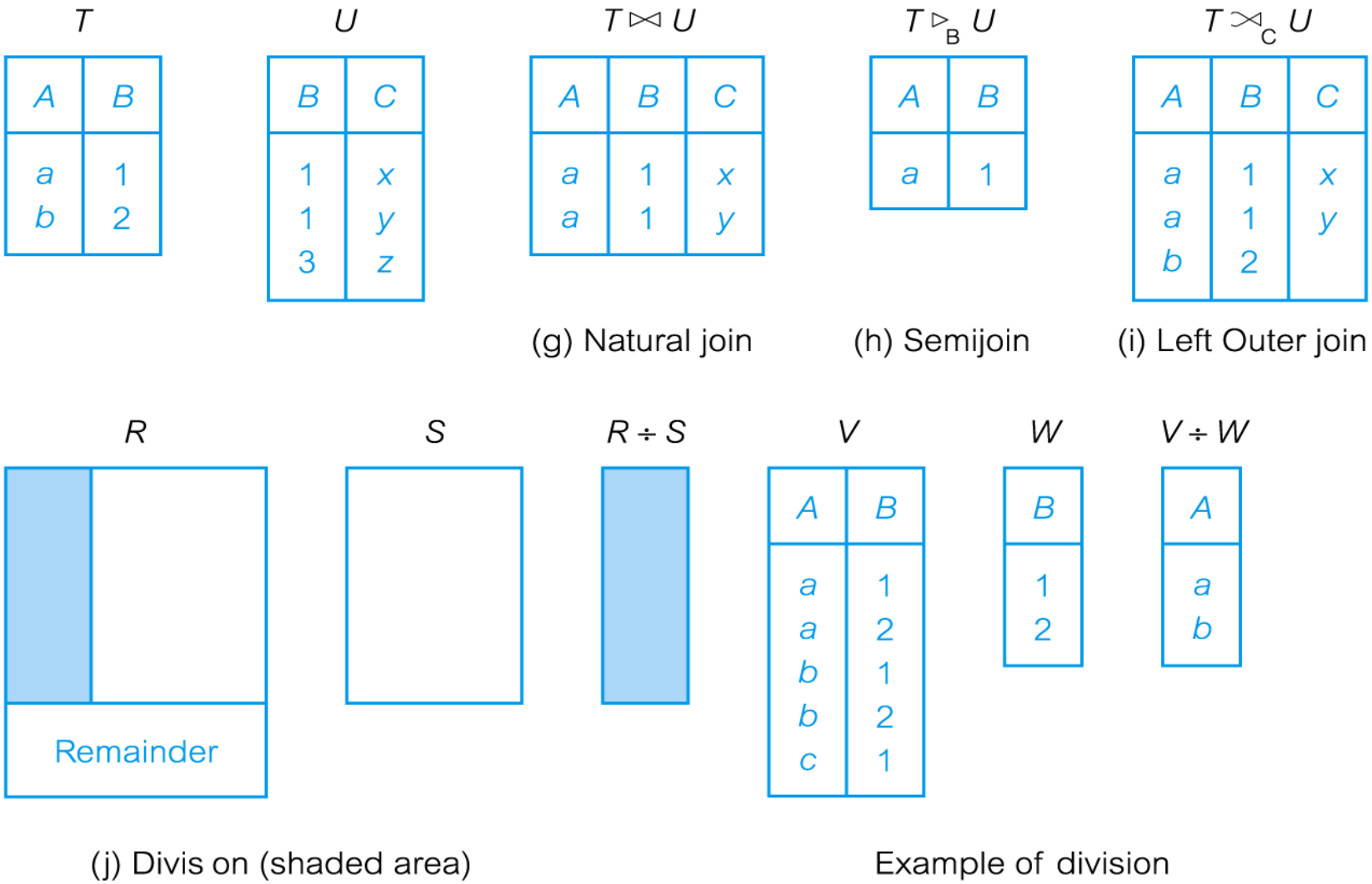


(e) Intersection



(f) Set difference

RELATIONAL ALGEBRA OPERATIONS



AGGREGATE OPERATIONS

- Aggregate or set functions are introduced to relational algebra to increase its expressive power.
- An aggregate function operates on a **set** of values (tuples) and computes one single value as output.
- These functions take a collection of values and return a single value as a result.
- The result of aggregation does not have a name. You can use rename operation to give it a name.

AGGREGATE OPERATIONS

- The various aggregate functions are

(*A*) **avg** (average value) : computes the average of all values in the (numeric) set

(*B*) **min**(minimum value): finds the minimum value of all values in the set

(*C*) **max**(maximum value): finds the maximum value of all values in the set

(*D*) **sum** (sum of values): computes the sum of all values in the (numeric) set

(*E*) **count** (number of values): returns the cardinality (number of elements) in the set

EXAMPLE

Example. Consider the relation r as shown below:

A	B	C
α	α	17
α	β	20
β	β	13
β	β	30

1. $\mathcal{F}_{\text{sum(c)}}(r) = 80$ (*i.e.*, sum(C)=80)
2. $\mathcal{F}_{\text{max(c)}}(r) = 30$ (*i.e.*, max(C)=30)
3. $\mathcal{F}_{\text{min(c)}}(r) = 13$ (*i.e.*, sum(C)=13)

RENAME OPERATION

- The rename operation is a unary operation which is used to give names to relational algebra expressions.
- It is denoted by rho (ρ).
- Suppose, you want to find cartesian product of a relation with itself then by using rename operator we give an alias name to that relation.
- Now, you can easily multiply that relation with its alias.

RENAME OPERATION

- **The second form of rename operator :**

You can also rename the attributes of any relation.

$$\rho x\,(a1, a2, \ldots, an)\,(E)$$

Where a1, a2, …, a*n* are new names for attributes of relation E.

RELATIONAL CALCULUS

- Relational calculus query specifies what is to be retrieved rather than how to retrieve it.
  - No description of how to evaluate a query.
- In first-order logic (or predicate calculus), predicate is a truth-valued function with arguments.
- When we substitute values for the arguments, function yields an expression, called a proposition, which can be either true or false.

RELATIONAL CALCULUS

- If predicate contains a variable (e.g. ‘x is a member of staff’), there must be a range for x.
- When we substitute some values of this range for x, proposition may be true; for other values, it may be false.
- When applied to databases, relational calculus has forms: tuple and domain.

RELATIONAL CALCULUS

- An alternative to relational algebra is relational calculus.
- It is a query system where queries are expressed as variables and formulas on these variables.
- It is a non-procedural or declarative by nature.

RELATIONAL CALCULUS

- In this, the formulas describe the properties of the required result relation without describing how to compute it *i.E.,* Query represents only results and hides the procedure to find the result. Relational calculus is based on predicate calculus, which is used to symbolize logical arguments in mathematics. Relational calculus has two variants.

RELATIONAL CALCULUS

- The first one is called *tuple relational calculus* in which variables take on tuples as values.
- The second one is called *domain relational calculus* in which variables range over the underlying domains.

TUPLE RELATIONAL CALCULUS

- Interested in finding tuples for which a predicate is true. Based on use of tuple variables.
- Tuple variable is a variable that 'ranges over' a named relation: i.e., variable whose only permitted values are tuples of the relation.
- Specify range of a tuple variable S as the Staff relation as:

  Staff(S)
- To find set of all tuples S such that P(S) is true:

  {S | P(S)}

TUPLE RELATIONAL CALCULUS - EXAMPLE

- To find details of all staff earning more than £10,000:

  $\{S \mid Staff(S) \wedge S.salary > 10000\}$

- To find a particular attribute, such as salary, write:

  $\{S.salary \mid Staff(S) \wedge S.salary > 10000\}$

TUPLE RELATIONAL CALCULUS

- Can use two quantifiers to tell how many instances the predicate applies to:
  - Existential quantifier $ (‘there exists’)
  - Universal quantifier " (‘for all’)

- Tuple variables qualified by " or $ are called bound variables, otherwise called free variables.

TUPLE RELATIONAL CALCULUS

- Existential quantifier used in formulae that must be true for at least one instance, such as:

  Staff(S) Ù ($B)(Branch(B) Ù
  (B.branchNo = S.branchNo) Ù B.city = ‘London’)

- Means ‘There exists a Branch tuple with same branchNo as the branchNo of the current Staff tuple, S, and is located in London’.

TUPLE RELATIONAL CALCULUS

- Universal quantifier is used in statements about every instance, such as:

  ("B) (B.city ≠ ‘Paris’)

- Means ‘For all Branch tuples, the address is not in Paris’.

- Can also use ~($B) (B.city = ‘Paris’) which means ‘There are no branches with an address in Paris’.

TUPLE RELATIONAL CALCULUS

- Formulae should be unambiguous and make sense.
- A (well-formed) formula is made out of atoms:
  - R(Si), where Si is a tuple variable and R is a relation
  - Si.a1 q Sj.a2
  - Si.a1 q c
- Can recursively build up formulae from atoms:
  - An atom is a formula
  - If F1 and F2 are formulae, so are their conjunction, F1 Ù F2; disjunction, F1 Ú F2; and negation, ~F1
  - If F is a formula with free variable X, then ($X)(F) and ("X)(F) are also formulae.

EXAMPLE - TUPLE RELATIONAL CALCULUS

- List the names of all managers who earn more than £25,000.

- {S.fName, S.lName | Staff(S) ∧
- S.position = ‘Manager’ ∧ S.salary > 25000}

- List the staff who manage properties for rent in Glasgow.

- {S | Staff(S) ∧ ($P) (PropertyForRent(P) ∧ (P.staffNo = S.staffNo) Ù P.city = ‘Glasgow’)}

EXAMPLE - TUPLE RELATIONAL CALCULUS

- List the names of staff who currently do not manage any properties.

$$\{S.fName, S.lName \mid Staff(S) \wedge (\sim(\$P)(PropertyForRent(P) \wedge (S.staffNo = P.staffNo)))\}$$

Or

$$\{S.fName, S.lName \mid Staff(S) \wedge ((\forall P)(\sim PropertyForRent(P) \vee \sim(S.staffNo = P.staffNo)))\}$$

EXAMPLE - TUPLE RELATIONAL CALCULUS

- List the names of clients who have viewed a property for rent in Glasgow.

{C.fName, C.lName | Client(C) Ù (($V)($P)
(Viewing(V) Ù PropertyForRent(P) Ù
(C.clientNo = V.clientNo) Ù
(V.propertyNo=P.propertyNo) Ù
P.city =‘Glasgow’))}

TUPLE RELATIONAL CALCULUS

- Expressions can generate an infinite set. For example:
  {S | ~Staff(S)}

- To avoid this, add restriction that all values in result must be values in the domain of the expression.

DOMAIN RELATIONAL CALCULUS

- Uses variables that take values from domains instead of tuples of relations.

- If F(d1, d2, . . . , dn) stands for a formula composed of atoms and d1, d2, . . . , dn represent domain variables, then:

  $\{d1, d2, \ldots, dn \mid F(d1, d2, \ldots, dn)\}$

  is a general domain relational calculus expression.

EXAMPLE - DOMAIN RELATIONAL CALCULUS

- Find the names of all managers who earn more than £25,000.

$$\{fN, lN \mid (\$sN, posn, sex, DOB, sal, bN)$$
$$(Staff\,(sN, fN, lN, posn, sex, DOB, sal, bN) \wedge$$
$$posn = \text{‘Manager’} \wedge sal > 25000)\}$$

EXAMPLE - DOMAIN RELATIONAL CALCULUS

- List the staff who manage properties for rent in Glasgow.

{sN, fN, lN, posn, sex, DOB, sal, bN |
($sN1,cty)(Staff(sN,fN,lN,posn,sex,DOB,sal,bN) ∧
PropertyForRent(pN, st, cty, pc, typ, rms,
rnt, oN, sN1, bN1) Ù
(sN=sN1) Ù cty='Glasgow')}

EXAMPLE - DOMAIN RELATIONAL CALCULUS

- List the names of staff who currently do not manage any properties for rent.

{fN, lN | ($sN)
(Staff(sN,fN,lN,posn,sex,DOB,sal,bN) ∧
(~($sN1) (PropertyForRent(pN, st, cty, pc, typ,
rms, rnt, oN, sN1, bN1) Ù (sN=sN1))))}

EXAMPLE - DOMAIN RELATIONAL CALCULUS

- List the names of clients who have viewed a property for rent in Glasgow.

{fN, lN | ($cN, cN1, pN, pN1, cty)
(Client(cN, fN, lN,tel, pT, mR) ∧
Viewing(cN1, pN1, dt, cmt) ∧
PropertyForRent(pN, st, cty, pc, typ,
rms, rnt,oN, sN, bN) Ù
(cN = cN1) Ù (pN = pN1) Ù cty = 'Glasgow')}

DOMAIN RELATIONAL CALCULUS

- When restricted to safe expressions, domain relational calculus is equivalent to tuple relational calculus restricted to safe expressions, which is equivalent to relational algebra.

- Means every relational algebra expression has an equivalent relational calculus expression, and vice versa.

S.No.	Domain Relational Calculus	Tuple Relational Calculus
1.	In domain relational calculus, the variables range over field values.	In tuple relational calculus, the variables take on tuples as values.
2.	It strongly influences SQL.	It strongly influences the QBE.
3.	In domain relational calculus, additional rules are needed to deal with the safety of expressions.	In tuple relational calculus, it is possible to restrict any existentially qualified variable to range over a specific relation for safe expressions.
4.	It is a non-procedural language.	It is also a non-procedural language.

S.No.	Relational Calculus	Relational Algebra
1.	It is non-procedural or declarative language.	It is a procedural language.
2.	It has a big influence on query languages like SQL and QBE.	It has big influence on almost all query languages based on relations.
3.	It describes the set of answers without being implicit about how they should be computed.	The relational algebra query describes a step by step procedure for computing the desired result.
4.	All relational algebra queries can be expressed in relational calculus.	It is restricted to safe queries on the calculus.
5.	The relational calculus expression can be written in any order and the order does not affect the strategy for evaluating the query.	A certain order among the operations is implicitly specified and the order influences the strategy for evaluating the query.
6.	The relational calculus languages are terse.	The relational algebra is more user friendly.