Select Operation – Example

- Relation *r* :

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D>5}(r)$:

A	B	C	D
α	α	1	7
β	β	23	10

Select Operation

- Notation: $\sigma_p(r)$
- p is called the selection predicate
- Defined as:

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

  - Where p is a formula in propositional calculus consisting of terms connected by : $\wedge$ (*and*), $\vee$ (*or*), $\neg$ (*not*)
  - Each term is one of:

    <attribute> *op* <attribute> or <constant>

    where *op* is one of: $=, \neq, >, \geq, <, \leq$

- Example:

$$\sigma_{\textit{branch-name}=\text{“Perryridge”}}(\textit{account})$$

Project Operation – Example

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation

- Notation:

$$\Pi_{A1,\ A2,\ \ldots,\ Ak}\ (r)$$

  where A_1, A_2 are attribute names and r is a relation name.
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result

  – since relations are sets
- Example: To eliminate the *branch-name* attribute of *account*

- $$\Pi_{account\text{-}number,\ balance}\ (account)$$

Union Operation – Example

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cup s$:

A	B
α	1
α	2
β	1
β	3

Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid (Union compatible)
  1. r, s must have the *same arity* (same number of attributes)
  2. The attribute domains must be *compatible*

  (e.g., 2nd column of r deals with the same type of values as does the 2^{nd} column of s)

  e.g.: to find all customers with either an account or a loan

  $\Pi_{customer\text{-}name}(depositor) \cup \Pi_{customer\text{-}name}(borrower)$

Set Difference Operation – Example

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r – s:*

A	B
α	1
β	1

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \text{ and } t \notin s\}$$

- Set differences must be taken between *compatible* relations.
  - *r* and *s* must have the *same arity*
  - attribute domains of *r* and *s* must be compatible

Cartesian-Product Operation – Example

- Relations *r, s:*

A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s:*

A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	19	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-Product Operation

- Notation: $r \times s$
- Defined as:

$$r \times s = \{ t\, q \mid t \in r \text{ and } q \in s \}$$

  - Assume that attributes of $r(R)$ and $s(S)$ are disjoint; i.e., $R \cap S = \varnothing$.
  - If not, renaming of attributes is needed.

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$
- $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	19	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	20	a
β	2	β	20	b

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.

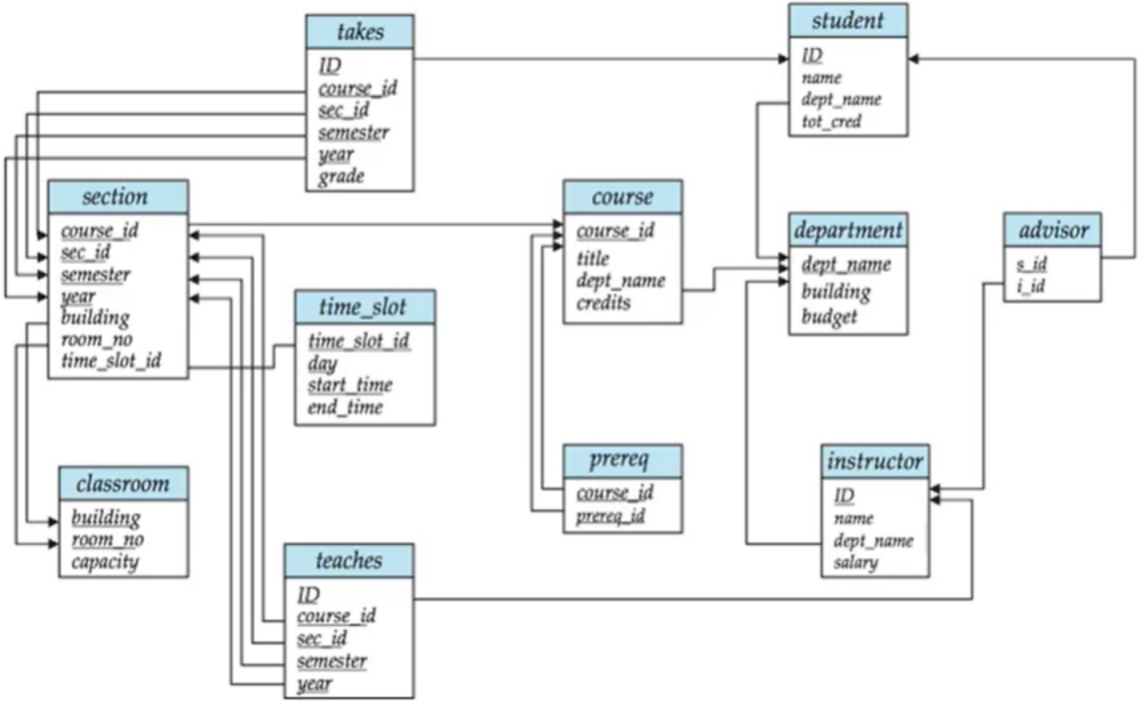
- $\rho_N(E)$ returns the expression E under the name N

- If a relational-algebra expression E has arity n, then

$$\rho_{N\,(A1,\,A2,\,\ldots,\,An)}(E)$$

returns the result of expression E under the name N, and with the attributes renamed to $A1, A2, \ldots, An$.

Schema Diagram for University Database



Example Queries

- Find all students in the CS department
- Find the name of each student in the CS department

Example Queries

- 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)

Example Queries

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

Query 1

Query 2

Formal Definition

- A *basic expression in the relational algebra* consists of either one of the following:
  - A relation in the database
  - A constant relation

- Let E_1 and E_2 be relational-algebra expressions; the following are all *relational-algebra expressions*:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\Pi_S(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_N (E_1)$, N is the new name for the result of E_1

Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Assignment
- Outer Join
- Generalized Projection
- Aggregation

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:

$$r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$$

- Assume union compatibility:
  - *r*, *s* have the *same arity*
  - attributes of *r* and *s* are compatible
- Note: $r \cap s = r - (r - s)$

Set-Intersection Operation – Example

- Relation *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Natural-Join Operation

- Let $r(R)$ and $s(S)$
- Notation: $r \bowtie s$ ✓
- The result is a relation on schema $R \cup S$ which is obtained by considering each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$,
    a tuple t is added to the result, where
    t has the same value as t_r on R, and t has the same value as t_s on S.
- Example:

  $R = (A, B, C, D)$ & $S = (E, B, D)$
  Result schema = (A, B, C, D, E)

  $r \bowtie s =$

Natural-Join Operation – Example

- Relations *r*, *s*:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

s

- $r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Properties

- $\Pi_{A1,\ldots,Ak}(r) \cap \Pi_{A1,\ldots,Ak}(s) = \Pi_{A1,\ldots,Ak}(r \bowtie s)$
- $(r \bowtie s) \bowtie t = r \bowtie (s \bowtie t)$
-
- If $R=S$ then $r \bowtie s = r \cap s$

- Theta Join
  - combine selection with Cartesian product

Assignment Operation

- The assignment operation ($\leftarrow$)
  - provides a convenient way to express complex queries
  - write query as a sequential program consisting of a series of assignments
- Assignment must always be made to a temporary relation variable.
  - The result to the right hand side is assigned to the relation variable on the left hand side.
  - May use the variable in subsequent expressions.
- Example: $r \cap s = r - (r - s)$

$$temp \leftarrow r - s$$
$$result \leftarrow r - temp$$

Example Queries

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

  - Query 1

  - Query 2

Example Queries

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