Database Systems

LESSON 08: RELATIONAL ALGEBRA

September 2019

Contents

Relational Algebra (RA) and Relational Calculus (RC)

Formally equivalent languages in relational model

RA

- Specifies how data can be retrieved from a relation
- High-level procedural language (requires sequence of operations to be specified)

RC

- Specifies what is to be retrieved from a relation
- Non-procedural language (doesn’t require sequence of operations to be specified)
- Not covered

Part 1

BASIC OPERATIONS

Relational Algebra

Six basic operators

- Selection: σ
- Projection: Π
- Union: $\cup$
- Set difference: –
- Cartesian product: x
- Rename: ρ

The operators take one or two relations as inputs and produce a new relation as a result.

Selection 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

Selection Operation

Notation: $\sigma_p(r)$

- *p* is called the **selection predicate**

Defined as:

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

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 of selection:

$$\sigma_{dept\_name=\text{“}Physics\text{”}}(instructor)$$

Projection Operation (Example)

Relation R

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

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Projection Operation

Notation:

$$\Pi_{A_1, A_2, \ldots, A_k}(r)$$

- 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 *dept_name* attribute of *instructor*

$$\Pi_{ID,\ name,\ salary}\ (instructor)$$

Union 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
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.

1. *r, s* must have the *same* **arity** (same number of attributes)
2. The attribute domains must be **compatible** (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \wedge year=2009} (section)) \cup \Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \wedge year=2010} (section))$$

Set Difference of Two Relations (Example)

Relation r, s

$r - s$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
β	1

Set Difference Operation

Notation $r - s$

Defined as:

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

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

Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \wedge year=2009}(section)) - \Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \wedge year=2010}(section))$$

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	10	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 x s

Defined as:

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

Assume that attributes of r(R) and s(S) are disjoint.

- That is, $R \cap S = \varnothing$

If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

Composition of Operations

Can build expressions using multiple operations

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

r x s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	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	10	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.

Example:

$$\rho_X(E)$$

returns the expression *E* under the name *X*

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

$$\rho_{X(A_1, A_2, \ldots, A_n)}(E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

Example Query 01

Find the names of all instructors in the Physics department, along with the course_id of all courses they have taught.

Instructor relation

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Teaches relation

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Example Query 01 (contd.)

Q1

- instructor x teaches
- $\sigma_{\text{instructor.ID=teaches.ID}}$ (instructor x teaches)
- $\sigma_{\text{dept\_name=“Physics”}}$ ($\sigma_{\text{instructor.ID=teaches.ID}}$ (instructor x teaches))
- $\Pi_{\text{instructor.ID,course\_id}}$ ($\sigma_{\text{dept\_name=“Physics”}}$ (
  $\sigma_{\text{instructor.ID=teaches.ID}}$ (instructor x teaches)))

Q2

- $\Pi_{\text{instructor.ID,course\_id}}$ ($\sigma_{\text{instructor.ID=teaches.ID}}$ (
  $\sigma_{\text{dept\_name=“Physics”}}$ (instructor) x teaches))

Example Query 01 (contd.)

Instructor x teaches

inst.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
22222	Einstein	Physics	95000	10101	CS-437	1	Fall	2009
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2010
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2010
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2009
22222	Einstein	Physics	95000	32343	HIS-351	1	Spring	2010
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
33456	Gold	Physics	87000	10101	CS-437	1	Fall	2009
33456	Gold	Physics	87000	10101	CS-315	1	Spring	2010
33456	Gold	Physics	87000	12121	FIN-201	1	Spring	2010
33456	Gold	Physics	87000	15151	MU-199	1	Spring	2010
33456	Gold	Physics	87000	22222	PHY-101	1	Fall	2009
33456	Gold	Physics	87000	32343	HIS-351	1	Spring	2010
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…

Outcome

name	*course_id*
Einstein	PHY-101

Example Query 02

Fine the largest salary in the university

Instructor relation:

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Example Query 02(contd.)

Step 1: find instructor salaries that are less than some other instructor salary (i.e. not maximum)

using a copy of *instructor* under a new name *d*

$$\Pi_{instructor.salary}\,(\sigma_{\,instructor.salary < d.salary}\,(instructor \times \rho_{d}\,(instructor)))$$

Step 2: Find the largest salary

$$\Pi_{salary}\,(instructor) - \Pi_{instructor.salary}\,(\sigma_{\,instructor.salary < d.salary}\,(instructor \times \rho_{d}\,(instructor)))$$

salary
65000
90000
40000
60000
87000
75000
62000
72000
80000
92000

salary
95000

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
- $\rho_x (E_1)$, x is the new name for the result of E_1

Part 2

ADDITIONAL OPERATIONS

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

Set-Intersection Operation

Notation: $r \cap s$

Defined as:

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

Assume:

- r, s have the *same arity*
- attributes of r and s are compatible

Note: $r \cap s = r - (r - s)$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	2

$r \cap s$

Natural-Join Operation

Notation r ⋈ s

Let *r* and *s* be relations on schemas *R* and *S* respectively.
Then, r ⋈ s is a relation on schema $R \cup S$ obtained as follows:

- Consider 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 $R \cap S$, add a tuple *t* to the result, where
  - *t* has the same value as t_r on *r*
  - *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* ⋈ *s* is defined as:

$$\Pi_{r.A,\ r.B,\ r.C,\ r.D,\ s.E} (\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s))$$

Natural Join Example

Relations r, s

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

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

Natural Join Examples (instructor ⋈ teaches)

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

Natural Join Example (contd.)

Find the names of all instructors in the Comp. Sci. department together with the course titles of all the courses that the instructors teach

- $\Pi_{\text{name, title}} (\sigma_{\text{dept\_name=“Comp. Sci.”}} (\text{instructor} \bowtie \text{teaches} \bowtie \text{course}))$

Natural join is associative

- (instructor $\bowtie$ teaches) $\bowtie$ course is equivalent to instructor $\bowtie$ (teaches $\bowtie$ course)

Natural join is commutative

- instruct $\bowtie$ teaches is equivalent to teaches $\bowtie$ instructor

Theta Join

A (*general* or *theta*) *join* of R and S is the expression

$R \bowtie_c S$

- where *join-condition c* is a *conjunction* of terms:
  - A_i *oper* B_i
  - A_i is an attribute of R; B_i is an attribute of S;
  - *oper* is one of =, <, >, $\geq \neq \leq$.

Meaning

$\sigma_c (R \times S)$

- join-condition c becomes a select condition c
- except for possible renamings of attributes

Theta Join Example

Relations:

- Employee(*Name,Id,MngrId,Salary*)
- Manager(*Name,Id,Salary*)

Find the names of all employees that earn more than their managers.

$$\pi_{\text{Employee.}Name}\,(\text{Employee} \bowtie_{MngrId=Id \text{ AND } Employee.Salary>Manager.Salary} \text{Manager})$$

Equijoin

Equijoin: Join condition is a conjunction of *equalities*.

$\pi_{Name,CrsCode}(\text{Student} \bowtie_{Id=StudId} \sigma_{Grade='A'}(\text{Transcript}))$

Student

Id	*Name*	*Addr*	*Status*
111	John	…..	…..
222	Mary	…..	…..
333	Bill	…..	…..
444	Joe	…..	…..

Transcript

StudId	*CrsCode*	*Sem*	*Grade*
111	CSE305	S00	B
222	CSE306	S99	A
333	CSE304	F99	A

Mary	CSE306
Bill	CSE304

Natural join is a special case of equijoin

Assignment Operation

The assignment operation (←) provides a convenient way to express complex queries.

- Write query as a sequential program consisting of
  - a series of assignments
  - followed by an expression whose value is displayed as a result of the query.
- Assignment must always be made to a temporary relation variable.

Outer Join

An extension of the join operation that avoids loss of information.

Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.

Uses null values:

- null signifies that the value is unknown or does not exist
- All comparisons involving null are (roughly speaking) **false** by definition.

Outer Join Examples

Relation instructor

ID	*name*	*dept_name*
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music

Relation teaches

ID	*course_id*
10101	CS-101
12121	FIN-201
76766	BIO-101

Join

instructor ⋈ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201

Left Outer Join

instructor ⟕ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	***null***

Outer Join Examples (contd.)

Relation instructor

ID	*name*	*dept_name*
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music

Relation teaches

ID	*course_id*
10101	CS-101
12121	FIN-201
76766	BIO-101

Right Outer Join

instructor ⟖ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
76766	***null***	***null***	BIO-101

Full Outer Join

instructor ⟗ *teaches*

ID	*name*	*dept_name*	*course_id*
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	***null***
76766	***null***	***null***	BIO-101

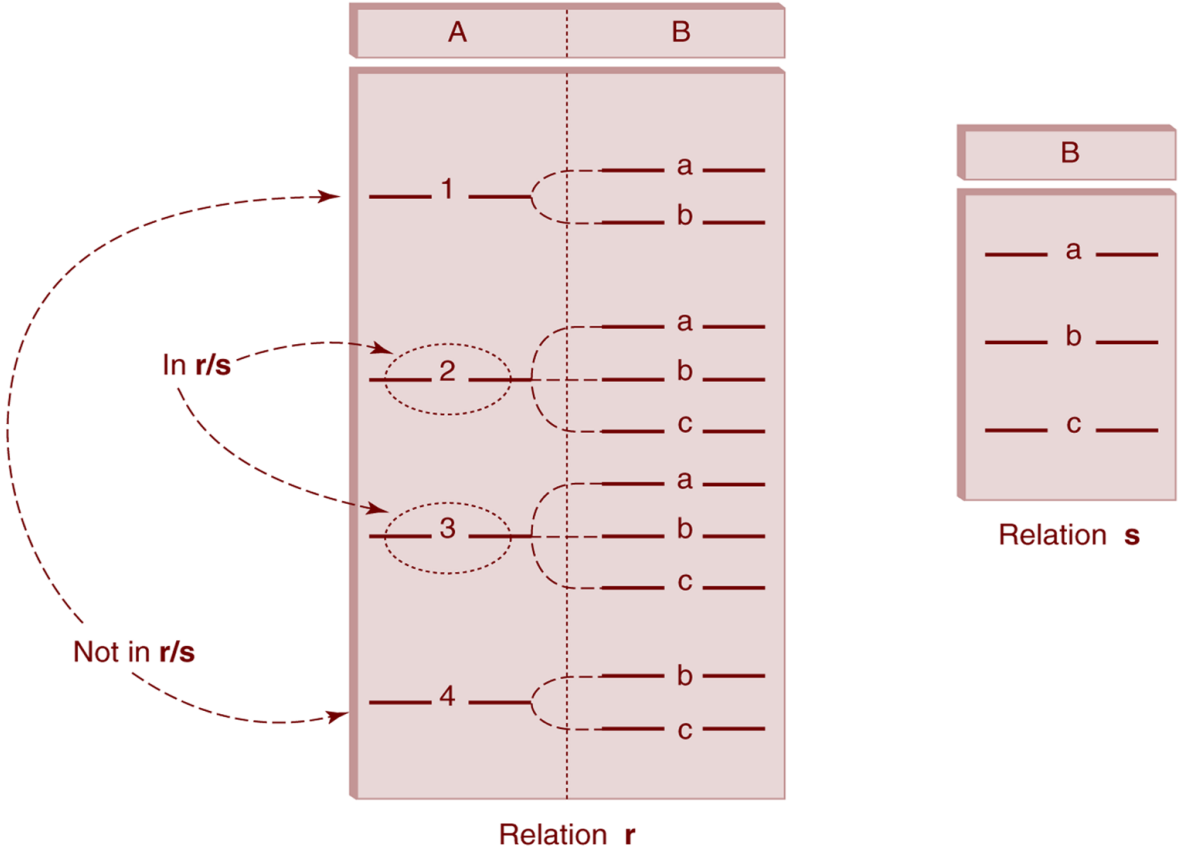
Division Operation

Division identifies the attribute values from a relation that are found to be paired with **all** of the values from another relation.

Example queries:

- Find all instructors who have taught **all** courses in the CS department.
- Find the students who have taken all courses in the CS department.

Division Operation Example



Division Operation (contd.)

Given relations r(R) and s(S), such that $S \subset R$, $r \div s$ is the largest relation t(R-S) such that

$$t \times s \subseteq r$$

E.g. let $r(ID, course\_id) = \prod_{ID,\ course\_id}(takes)$ and

$$s(course\_id) = \prod_{course\_id}(\sigma_{dept\_name=\text{“Biology”}}(course))$$

then $r \div s$ gives us students who have taken all courses in the Biology department

Can write $r \div s$ as

$$temp1 \leftarrow \prod_{R-S}(r)$$

$$temp2 \leftarrow \prod_{R-S}((temp1 \times s) - \prod_{R-S,S}(r))$$

$$result = temp1 - temp2$$

Part 3

EXTENDED RELATIONAL-ALGEBRA-OPERATIONS

Extended Relational-Algebra-Operations

Generalized Projection

Aggregate Functions

Generalized Projection

Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\Pi_{F_1, F_2, \ldots, F_n}(E)$$

- E is any relational-algebra expression
- Each of F_1, F_2, …, F_n are are arithmetic expressions involving constants and attributes in the schema of E.

Example: given relation *instructor(ID, name, dept_name,* salary) where salary is annual salary, get the same information but with monthly salary

$$\Pi_{ID,\ name,\ dept\_name,\ salary/12}\ (instructor)$$

Aggregate Functions and Operations

Aggregation function takes a collection of values and returns a single value as a result.

- **avg**: average value
- **min**: minimum value
- **max**: maximum value
- **sum**: sum of values
- **count**: number of values

Aggregate operation in relational algebra

$$G_1, G_2, \ldots, G_n \; \mathcal{G} \; F_1(A_1), F_2(A_2), \ldots, F_n(A_n) (E)$$

- *E* is any relational-algebra expression
- G_1, G_2 …, G_n is a list of attributes on which to group (can be empty)
- Each F_i is an aggregate function
- Each A_i is an attribute name

Note: Some books/articles use γ instead of $\mathcal{G}$ (Calligraphic G)

Aggregate Operation Examples

Relation *r*:

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

$$\mathcal{G}_{\mathbf{sum(c)}}(\mathrm{r})$$

sum(*c*)
27

Aggregate Operation Examples (contd.)

Find the average salary in each department

$$dept\_name\ \mathcal{G}\ \mathbf{avg}(salary)\ (instructor)$$

dept_name	*salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

Aggregate Functions (Contd.)

Result of aggregation does not have a name

- Can use rename operation to give it a name
- For convenience, we permit renaming as part of aggregate operation

$$dept\_name\ \mathcal{G}\ \textbf{avg}(salary)\ \textbf{as}\ avg\_sal\ (instructor)$$

Part 4

DATABASE MODIFICATION

Modification of the Database

Database contents may be modified using the following operations:

- Deletion
- Insertion
- Updating

All these operations can be expressed using the assignment operator

Deletion

Similarly to a query, except that selected tuples are removed from the database instead.

- Applied to whole tuples only, not values on particular attributes.

Expressed by

$$r \leftarrow r - E$$

- r is a relation and
- E is a relational algebra query.

Deletion Examples

Relation schemas:
- *BRANCH*(*branch name*, *branch city, assets*)
- *LOAN* (*loan number*, *branch name, amount*)
- *ACCOUNT* (*account number*, *branch name, balance*)
- *DEPOSITOR* (*customer name*, *account number*)

Delete all account records in the Perryridge branch:

$$account \leftarrow account - \sigma_{branch\_name = \text{“Perryridge”}} (account)$$

Delete all loan records with amount in the range of 0 to 50:

$$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 50} (loan)$$

Delete all accounts at branches located in Needham city:

$$r_1 \leftarrow \sigma_{branch\_city = \text{“Needham”}} (account \bowtie branch)$$

$$r_2 \leftarrow \prod_{account\_number,\ branch\_name,\ balance} (r_1)$$

$$r_3 \leftarrow \prod_{customer\_name,\ account\_number} (r_2 \bowtie \text{depositor})$$

$$account \leftarrow \text{account} - r_2$$

$$depositor \leftarrow \text{depositor} - r_3$$

Insertion

To insert data into a relation, we either:

- specify a tuple to be inserted or
- write a query whose result is a set of tuples to be inserted

in relational algebra, an insertion is expressed by:

$$r \leftarrow r \cup E$$

where r is a relation and E is a relational algebra expression.

The insertion of a single tuple is expressed by letting E be a constant relation containing one tuple.

Insertion Examples

Relation schemas:

- *BORROWER (customer name, loan number)*
- *LOAN (loan number, branch name, amount)*
- *ACCOUNT (account number, branch name, balance)*
- *DEPOSITOR (customer name, account number)*

Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch:

$$account \leftarrow account \cup \{(\text{“A-973”}, \text{“Perryridge”}, 1200)\}$$

$$depositor \leftarrow depositor \cup \{(\text{“Smith”}, \text{“A-973”})\}$$

Provide as a gift for all loan customers in the Perryridge branch, a $200 savings account. Let the loan number serve as the account number for the new savings account:

$$r_1 \leftarrow (\sigma_{branch\_name = \text{“Perryridge”}} (borrower \bowtie loan))$$

$$account \leftarrow account \cup \prod_{loan\_number,\ branch\_name,\ 200} (r_1)$$

$$depositor \leftarrow depositor \cup \prod_{customer\_name,\ loan\_number} (r_1)$$

Updating

Change a value in a tuple without charging all values in the tuple

Use the generalized projection operator to do this task:

$$r \leftarrow \prod_{F_1, F_2, \ldots, F_l,} (r)$$

Update Examples

Relation schemas:

- *BORROWER (customer name, loan number)*
- *LOAN (loan number, branch name, amount)*
- *ACCOUNT (account number, branch name, balance)*
- *DEPOSITOR (customer name, account number)*

Make interest payments by increasing all balances by 5 percent

$$account \leftarrow \prod_{account\_number,\ branch\_name,\ balance\ *\ 1.05} (account)$$

Pay all accounts with balances over $10,000 6 percent interest and pay all others 5 percent

$$account \leftarrow \prod_{account\_number,\ branch\_name,\ balance\ *\ 1.06} (\sigma_{BAL > 10000} (account)) \cup \prod_{account\_number,\ branch\_name,\ balance\ *\ 1.05} (\sigma_{BAL \leq 10000} (account))$$

Part 5

MULTISET RELATIONAL ALGEBRA

Multisets

Multiset X

Tuple
(1, a)
(1, a)
(1, b)
(2, c)
(2, c)
(2, c)
(1, d)
(1, d)

Equivalent Representations of a **Multiset**

$\lambda(\mathbf{X})$= “Count of tuple in X” (Items not listed have implicit count 0)

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	1
(2, c)	3
(1, d)	2

In a set all counts are {0,1}.

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

$\cap$

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

$=$

Multiset Z

Tuple	$\lambda(Z)$
(1, a)	2
(1, b)	0
(2, c)	2
(1, d)	0

$$\lambda(Z) = min(\lambda(X), \lambda(Y))$$

For sets, this is **intersection**

Generalizing Set Operations to Multiset Operations

Multiset X

Tuple	$\lambda(X)$
(1, a)	2
(1, b)	0
(2, c)	3
(1, d)	0

$\cup$

Multiset Y

Tuple	$\lambda(Y)$
(1, a)	5
(1, b)	1
(2, c)	2
(1, d)	2

$=$

Multiset Z

Tuple	$\lambda(Z)$
(1, a)	7
(1, b)	1
(2, c)	5
(1, d)	2

$$\lambda(Z) = \lambda(X) + \lambda(Y)$$

For sets, this is **union**

Multiset Relational Algebra

Pure relational algebra removes all duplicates

- e.g. after projection

Multiset relational algebra retains duplicates, to match SQL semantics

- SQL duplicate retention was initially for efficiency, but is now a feature

Multiset relational algebra defined as follows

- selection: preserve the number of occurrences
- projection: no duplicate elimination
- cross product: no duplicate elimination
  - If there are *m* copies of *t1* in *r*, and *n* copies of *t2* in *s*, there are *m* x *n* copies of *t1.t2* in *r* x *s*
- Other operators similarly defined
  - E.g. union: *m* + *n copies,* intersection: min(*m, n*) copies