Chapter 2 Relational Model

Tian Tang
United International College

Relational Model

- Used by all major commercial database systems
- Very simple model
- Query with high-level languages: simple yet expressive
- Efficient implementations

Outline

- Structural Concepts & Terminology of Relational Databases
- Relational Algebra Operations
  - Fundamental Relational-Algebra-Operations
  - Additional Relational-Algebra-Operations
  - Extended Relational-Algebra-Operations
- Null Values
- Modification of the Database

Structural Concepts

- Database: a set of named relations (or table)
- Each relation has a set of named attributes (or columns), $A_1, A_2, ..., A_m$.
- Each tuple (or row) has a value for each attribute, $t_1, t_2, ..., t_m$. The order of tuples is irrelevant.

Table: Student

ID	Name	Gender	GPA
123	Richard	M	3.9
234	Grace	F	3.4
...	...	...	...

Table: College

Name	Location	Enrollment
UIC	Zhuhai	7,500
HKUST	Hong Kong	14,208
HKBU	Hong Kong	8,266
...	...	...

Structural Concepts

- Each attribute has a type (or domain).
  - Integer, string, float, enumerate, $\cdots$
  - Attribute values are (normally) required to be atomic, which means the value stored is indivisible, e.g., cannot be a set of account numbers.
  - Domain is atomic if all its members are atomic.

Table: Student

ID	Name	Gender	GPA
123	Richard	M	3.9
234	Grace	F	3.4
$\cdots$	$\cdots$	$\cdots$	$\cdots$

Table: College

Name	Location	Enrollment
UIC	Zhuhai	7,500
HKUST	Hong Kong	14,208
HKBU	Hong Kong	8,266
$\cdots$	$\cdots$	$\cdots$

Structural Concepts

- Schema - structural description of relations in database, including name of relations, attributes and types, denoted as $R = (A_1, A_2, ..., A_m)$. $r(R)$ denotes a relation on the relation schema R.

Table: Student

ID	Name	Gender	GPA
123	Richard	M	3.9
234	Grace	F	3.4
...	...	...	...

Table: College

Name	Location	Enrollment
UIC	Zhuhai	7,500
HKUST	Hong Kong	14,208
HKBU	Hong Kong	8,266
...	...	...

Structural Concepts

- Instance - actual contents of the table at a given point of time, consists of the tuples in the relation

Table: Student

ID	Name	Gender	GPA
123	Richard	M	3.9
234	Grace	F	3.4
...	...	...	...

Table: College

Name	Location	Enrollment
UIC	Zhuhai	7,500
HKUST	Hong Kong	14,208
HKBU	Hong Kong	8,266
...	...	...

Structural Concepts

- NULL - special value for *unknown* or *undefined*

Table: Student

ID	Name	Gender	GPA
123	Richard	M	3.9
234	Grace	F	3.4
345	Bob	M	Null

Table: College

Name	Location	Enrollment
UIC	Zhuhai	7,500
HKUST	Hong Kong	14,208
HKBU	Hong Kong	8,266
...	...	...

Student with GPA$>$ 3.5? Student with GPA$\geq$ 3.5? All Student?

Structural Concepts

- Key - attribute whose value is unique in each tuple, or set of attributes whose combined values are unique.
  - Identify specific tuples
  - Database system build special index structures to access tuple by key
  - Refer tuples in another relation

Table: Student

ID	Name	Gender	GPA
123	Richard	M	3.9
234	Grace	F	3.4
345	Bob	M	Null

Table: College

Name	Location	Enrollment
UIC	Zhuhai	7,500
HKUST	Hong Kong	14,208
HKBU	Hong Kong	8,266
...	...	...

Structural Concepts

- Superkey - A set of attributes that uniquely identifies each tuple in a relation
- Candidate key - A “minimal superkey”
- Primary Key - A candidate key that is most appropriate to become the main key of the relation. A key uniquely identify each tuple in a relation

Table: Student

ID	Name	Gender	GPA	Nationality
123	Richard	M	3.9	Chinese
234	Grace	F	3.4	Chinese
345	Bob	M	Null	USA

Superkey-{ID,Name},{ID}, {Name}, {GPA}, {Name,Gender,GPA},...

Candidate key-{ID},{Name},{GPA},...

Primary Key-{ID}

Structural Concepts

- Secondary key-The candidate key which are not selected for primary key
- Foreign key-An attribute or combination of attributes in one table whose value must either match the primary key in another table or be null.

Table: Student

ID	Name	Gender	GPA	Nationality
123	Richard	M	3.9	Chinese
234	Grace	F	3.4	Chinese
345	Bob	M	Null	USA

Secondary key-{Name,Gender,GPA}

Foreign key-{Nationality}

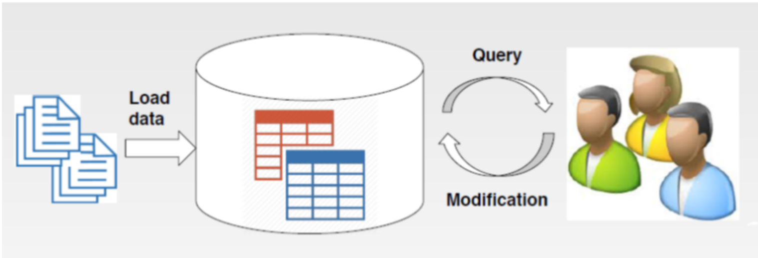
Class Exercise

- 1. Given a relation r defined over the schema R, which of the following can always uniquely identify the tuples in r?
  A. any non-null attributes of R
  B. super key of R
  C. the first attribute in R
  D. R itself
- 2. Given the following relation, list all candidate keys and superkeys.

A	B	C	D
A1	B1	C1	D1
A1	B2	C2	D1
A2	B1	C2	D1

Querying Relational Databases

- Steps in creating and using a (relational) database
  - Design schema; create using Data Definition Language (DDL)
  - "Bulk load" initial data
  - Repeat: execute queries and modifications



Query Languages

- Computer languages used to make queries in database
- Categories of programming languages
  - Procedural, specifies a series of well-structured steps and procedures to compose a program
  - Non-procedural, or declarative, expresses the logic of a computation without describing its control flow
- Two formal query languages of relational model, form the base for real languages:
  - Relational algebra: procedural, very useful for representing execution plans, foundation for SQL
  - Relational calculus: declarative, lets users describe what they want, rather than how to compute it, e.g, QBE and Datalog

Relational Algebra

- Algebra in general is a pair (s, o), where
  - s is a set of operands, and
  - o is a set of operators (unary or multiary).
- For example, linear algebra.
- Queries in relational algebra are composed into a sequence of operands and operators.
  - Operators in this course is limited to unary (take one operand) or binary (two arguments).
- Every operator take one or two relations as operands and produce a new relation as the result.
- Categories:
  - Fundamental operations of relational algebra
  - Additional operations of relational algebra
  - Extended operations of relational algebra

Select Operation

- Select Operation (σ) selects tuples that satisfy the given predicate from a relation, denoted as $\sigma_p(r)$,
  - p is called the selection predicate, user defined conditions.
- Notation:

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

  Where p is a formula in propositional logic formula consisting of term connected by:$\wedge$(**and**), $\vee$(**or**), $\neg$(**not**)
  Each term is in the form of:

$$< attribute > \textbf{op} < attribute > \text{or} < constant >$$

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

$$\sigma_{GPA>3.5}(Student)$$

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

Project Operation

- Project operation $\prod$ deletes unwanted columns from relation.
- Notation:

$$\prod_{A_1,A_2,...,A_m}(r)$$

  where $A_1, A_2, ..., A_m$ are attribute names of relation r.
- The result is defined as the relation of k columns, obtained by erasing the columns that are not listed.
- Duplicate rows are removed from the result automatically.
- Example: To eliminate the gender attribute of Students:

$$\prod_{ID,Name,GPA}(r)$$

- Intuitively, this operator is same as projecting a 3-D object to a 2-D plain.

Project Operation-Example

- Relation r

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

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

A	C
α	1
α	1
β	1
β	2

$=$

A	C
α	1
β	1
β	2

Union Operation

- Union operation ($\cup$) performs binary union between two given relations, denoted as $r \cup s$
- Notation:

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

  where r and s are either database relations or relation result set
- For $r \cup s$ to be valid:
  1. r, s must have the same number of attributes
  2. The attribute domains must be compatible
- Example: To find all customers with either an account or a loan:

$$\prod_{customer\_name}(depositor) \cup \prod_{customer\_name}(borrower)$$

Union Operation-Example

- Relation r, s

r:

A	B
α	1
α	2
β	1

s:

A	B
α	2
β	3

- $r \cup s$:

A	B
α	1
α	2
β	1
β	3

Set Difference Operation

- Result of set difference ($-$) operation is tuples, which are present in the first relation but not in the second relation, denoted as $r - s$
- Notation:

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

- Set differences must be taken between compatible relations.
  - r and s must have the same number of attributes
  - attribute domains of r and s must be compatible
- Example: to find all customers with an account but no loan:

$$\prod_{customer\_name}(depositor) - \prod_{customer\_name}(borrower)$$

Set Difference Operation-Example

- Relation r, s

r:

A	B
α	1
α	2
β	1

s:

A	B
α	2
β	3

- $r - s$:

A	B
α	1
β	1

Cartesian-Product Operation

- Combines information of two different relations into one, denoted as $r \times s$
- Notation:

$$r \times s = \{< t, q > | t \in r \text{ and } t \in s\}$$

  return a relation whose schema contains all the fields of r followed by all the fields of s.
- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.
  - Naming conflict might appear

Cartesian-Product Operation-Example

- Relation r, s

r:

A	B
α	1
β	2

s:

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

- $r \times s$:

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

Rename Operation

- Allows us to rename the output relation, denoted as ρ.
- Notation:

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

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

- Rename the expression name: $\rho_X(E)$
- Rename the given attributes of the expression:

$$\rho_{X(B_1 \to A_1, B_2 \to A_2, ..., B_k \to A_k)}(E)$$

- Example: $\rho_{UICStudent(ID \to Student\_ID, Name \to Student\_Name)}(Student)$

Rename Operation-Example

- Relation r:

A	B
α	1
β	2

- $\rho_{A \to C, B \to D}(E)$

C	D
α	1
β	2

Composition of Operations

- Can build expressions using multiple operations
- Since the result of any relational algebra operation is a relation, again, this intermediate result may be the input for a subsequent operation
- Example: $\sigma_{A=C}(r \times s)$

r:

A	B
α	1
β	2

s:

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

Composition of Operations

- Can build expressions using multiple operations
- Since the result of any relational algebra operation is a relation, again, this intermediate result may be the input for a subsequent operation
- Example:$\sigma_{A=C}(r \times s)$

$q = t \times s$:

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

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

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

Banking Example

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

Example Queries

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Find all loans of over $1200

$$\sigma_{amount>1200}(loan)$$

- Find the loan number for each loan of an amount greater than $1200

$$\prod_{loan\_number}(\sigma_{amount>1200}(loan))$$

Example Queries

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Find the names of all customers who have a loan, an account, or both, from the bank

$$\prod_{customer\_name}(borrower) \cup \prod_{customer\_name}(depositor)$$

Example Queries

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Find the names of all customers who have a loan at the Perryridge branch
  - Query 1

$$\prod_{customer\_name}(\sigma_{branch\_name="Perryridge"} \\ (\sigma_{borrower.loan\_number=loan.loan\_number}(borrower \times loan)))$$

  - Query 2

$$\prod_{customer\_name}(\sigma_{loan.loan\_number=borrower.loan\_number} \\ (\sigma_{branch\_name="Perryridge"}(loan \times borrower)))$$

Example Queries

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank

$$\prod_{customer\_name}(\sigma_{branch\_name=\text{``}Perryridge\text{''}}$$
$$(\sigma_{borrower.loan\_number=loan.loan\_number}(borrower \times loan)))$$
$$- \prod_{customer\_name}(depositor)$$

Example Queries

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Find the largest account balance
  - strategy:
    - Find those balances that are not the largest
      −Rename account relation as d so that we can compare each account balance with all others
    - Use set difference to find those account balances that were not found in the earlier step.

$$\prod_{balance}(account) - \prod_{account.balance} (\sigma_{account.balance < d.balance}(acount \times \rho_d(account)))$$

Relational Model

- Structural Concepts & Terminology of Relational Databases
- Relational Algebra Operations
  - Fundamental Relational-Algebra-Operations
  - Additional Relational-Algebra-Operations
  - Extended Relational-Algebra-Operations
- Null Values
- Modification of the Database

Additional Operations

- Additional algebra are defined in terms of the fundamental operations. They do not add power to the algebra, but are useful to simplify common queries
    - Set Intersection
    - Natural join
    - Division
    - Assignment

Set-Intersection Operation

- Set intersection is denoted by $\cap$, returns a relation that contains tuples that are in both of its argument relations
- Notation:

$$r \cap s = \{t | t \in r \text{ and } t \in s\}$$

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

Set-Intersection Operation-Example

- Relation r, s:

r:

A	**B**
α	1
α	2
β	1

s:

A	**B**
α	2
β	3

- $r \cap s$

A	**B**
α	2

Natural-Join Operation

- Natural join combines a Cartesian product and a selection into one operation, denoted as $\bowtie$
- Notation:
  $r \bowtie s$, where r and s be relations on schemas R and S. Then, the result is a relation on Schema $R \cup S$, obtained as follows:
  - Consider each of of tuples t_r from r and t_s from s
  - If t_r and t_s have same value on each of attributes of $R \cap S$, add a tuple t to the result, where
    1.t has same value as t_r on R
    2.t has same value as t_s on S
- Example: $R = (A, B, C, D)$ and $S = (E, B, D)$
  - Result schema = (A,B,C,D,E)
  - $r \bowtie s$ is defined as:

$$\prod_{R \cup S}(\sigma_{r.B=s.B \wedge r.D=s.D}(r \times s))$$

Natural Join Operation-Example

- Relation r, s:

r:

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

s:

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

- $r \bowtie s$

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

Division Operation

- Suited to queries that include the phrase "for all", denoted as $r \div s$.
- Let r and s be relations on schemas R and S, where
  - $R = (A_1, A_2, \cdots, A_m, B_1, B_2, ..., B_n)$
  - $S = (B_1, B_2, \cdots, B_n)$
  - The result of $r \div s$ is a relation on schema of $R - S = (A_1, A_2, ..., A_m)$
  - The tuple t is in $r \div s$ if for every tuple t_s in s, there is a tuple t_r in r satisfying both of the following:
    1. $t_r[S] = t_s[S]$
    2. $t_r[R - S] = t$
  - $r \div s$ can be expressed using fundamental operations
    $r \div s = \{t | t \in \prod_{R-S}(r)$ and $\forall u \in s(tu \in r)\}$, where tu means the concatenation of tuples t and u to produce a single tuple

Division Operation-Example

- Relation r, s:

r:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

s:

B
1
2

- $r \div s$:

A
α
β

Another Division Example

- Relation r, s:

r:

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

s:

D	E
a	1
b	1

- $r \div s$:

A	B	C
α	a	γ
γ	a	γ

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
    - followed by an expression whose value is displayed as a result of the query
  - Assignment must always be made to a temporary relation variable
- Example: Write $r \div s$ as
  $temp1 \leftarrow \prod_{R-S}(r)$
  $temp2 \leftarrow \prod_{R-S}((temp1 \times s) - r)$
  $result = temp1 - temp2$
  - The result to the right of the $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$
  - May use variable in subsequent expressions

Bank Example Queries

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Find the names of all customers who have a loan and an account at bank

$$\prod_{customer\_name}(borrower) \cap \prod_{customer\_name}(depositor)$$

- Find the names of all customers who have a loan at the bank and the loan amount

$$\prod_{customer\_name, amount}(borrower \bowtie loan)$$

Bank Example Queries

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Find the names of all customers who have an account from at least the "Downtown" and the "Uptown" branches
  - Query 1

$$\prod_{customer\_name}(\sigma_{branch\_name=\text{``}Downtown''}(depositor \bowtie account)) \cap \prod_{customer\_name}(\sigma_{branch\_name=\text{``}Uptown''}(depositor \bowtie account))$$

  - Query 2

$$\prod_{customer\_name}((depositor \bowtie account) \div \rho_{temp(branch\_name)}(\{(\text{``}Downtown''), (\text{``}Uptown'')\}))$$

Note that Query 2 use a constant relation.

Bank Example Queries

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Find the names of all customers who have an account at all branches located in Brooklyn city

$$\prod_{customer\_name}((depositor \bowtie account) \div \prod_{branch\_name}(\sigma_{branch\_city = \text{``}Brooklyn\text{''}}(branch)))$$

Relational Model

- Structural Concepts & Terminology of Relational Databases
- Relational Algebra Operations
  - Fundamental Relational-Algebra-Operations
  - Additional Relational-Algebra-Operations
  - Extended Relational-Algebra-Operations
- Null Values
- Modification of the Database

Extended Relational-Algebra-Operations

- Relational algebra operations have been extended in various ways
  - More generalised
  - More useful
- Three major extensions:
  - Generalized projection
  - Aggregate functions
  - Additional join operations
- All of these appear in SQL standards

Generalized Projection

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

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

- E is any relational-algebra expression.
- Each of $F_1, F_2, \ldots, F_n$ is an arithmetic expressions involving constants and attributes in the schema of E
- Example:
    - Given relation *credit_info(customer_name, limit, credit_balance)*, find how much more each person can spend:

$$\prod_{customer\_name, limit - credit\_balance}(credit\_info)$$

- Apply rename:

$$\prod_{customer\_name, limit - credit\_balance \rightarrow credit\_available}(credit\_info)$$

Aggregate Functions and Operations

- Aggregate function takes a collection of values and returns a single value as a result
  - Duplicates are not eliminated
- Common aggregate functions:
  - avg: average value
  - min: minimum value
  - max: maximum value
  - sum: sum of values
  - count: number of values
- Notation:

$$g_{F_1(A_1),\ldots,F_n(A_n)}(E)$$

  - E is any relational-algebra expression
  - Each F_i is an aggregate function applied to attribute A_i of E

Aggregate Functions and Operations

- Grouping. Sometimes need to compute aggregates on a per-item basis
- Back to College relation
  - What is the average enrolment of colleges in each location?
  - How many colleges in each location?
- Steps:
  - Input relation College is grouped by unique values of Location
  - average(enrolment) and count(name) are applied to each group
- Notation:

$$_{G_1,G_2,\ldots,G_k}g_{F_1(A_1),\ldots,F_n(A_n)}(E)$$

  - E is any relational-algebra expression.
  - Each F_i is an aggregate function applied to attribute A_i.
  - $G_1, G_2, \ldots, G_k$ is a list of attributes on which to group.

Aggregate Operation Example

- Relation r:

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

- $g_{sum(c)}(r)$:

sum(C)
27

- $g_{count(c)}(r)$:

count(C)
4

Aggregate Operation Example

- Relation account grouped by branch-name:

branch_name	account_name	balance
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

$$_{branch\_name}g_{\mathbf{sum}(balance)}(account)$$

branch_name	sum(balance)
Perryridge	1300
Brighton	1500
Redwood	700

Aggregate Functions(Cont.)

- 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

$$_{branch\_name}\mathcal{G}_{\textbf{sum}(balance)\ \textbf{as}\ \textbf{sum\_balance}}(account)$$

branch_name	**sum_balance**
Perryridge	1300
Brighton	1500
Redwood	700

Outer Join

- An extension of the join operation that avoids loss of information
- Natural join requires that both left and right tables have a matching tuple
- Computes the join and then adds tuples from one relation that does not match tuples in the other relation to the result
- Missing information is represented by NULL values
- Categories:
  - Left outer join
  - Right outer join
  - Full outer join

Outer Join

- Left outer join

$$r ⟕ s$$

1.If a tuple t_r in r does not match any tuple in s, result contains $\{t_r, null, ..., null\}$
2.If a tuple t_s in s doe not match any tuple in r, it is excluded.

- Right outer join

$$r ⟖ s$$

1.If a tuple t_r does not match any tuple in s, it is excluded
2.If a tuple t_s does not match any tuple in r, result contains $\{t_s, null, ..., null\}$

- Full outer join

$$r ⟗ s$$

Includes tuples from t_r that does not match s as well as t_s that does not match r

Outer Join Example

- Relation *loan*

loan_number	branch_name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation *borrower*

customer_name	loan_number
Jones	L-170
Smith	L-230
Hayes	L-155

Outer Join Example

- Join

$$loan \bowtie borrower$$

loan_number	branch_name	amount	customer_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- Left Outer Join

$$loan ⟕ borrower$$

loan_number	branch_name	amount	customer_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

Outer Join Example

- Right Outer Join

$$loan ⟖ borrower$$

loan_number	branch_name	amount	customer_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

- Full Outer Join

$$loan ⟗ borrower$$

loan_number	branch_name	amount	customer_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes

Relational Model

- Structural Concepts & Terminology of Relational Databases
- Relational Algebra Operations
  - Fundamental Relational-Algebra-Operations
  - Additional Relational-Algebra-Operations
  - Extended Relational-Algebra-Operations
- Null Values
- Modification of the Database

Null Values

- It is possible for tuples to have a null value, denoted by *null*, for some of their attributes
- *null* signifies an unknown value or that a value does not exist
- The result of any arithmetic expression involving *null* is *null*
- Aggregate functions simply ignore null values (as in SQL)
- For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same (as in SQL)

Null Values

- Comparisons with null values return the special truth value: *unknown*
- Three-valued logic using the truth value *unknown*:
  - OR:(unknown **or** true)=true,
    (unknown **or** false)=unknown,
    (unknown **or** unknown)=unknown
  - AND: (true **and** unknown)=unknown,
    (false **and** unknown)=false,
    (unknown **and** unknown)=unknown
  - NOT:(**not** unknown)=unknown
  - In SQL *P* is **unknown** evaluates to true if predicate *P* evaluates to *unknown*
- For each relational operation, need to specify behaviour w.r.t. null and unknown

Null Values

- Result of select predicate is treated as false if it evaluates unknown
  - $\sigma_P(r)$-if P evaluates to unknown for a tuple, the tuple is excluded from the result.
- Natural join. Tuples are excluded if common attribute has a null value
- Project. Null value is treated like any other value.
- Grouping, Union, Intersection and Difference, null value is treated like any other value
- Aggregation
  - Null value is removed from the input multiset before the function is applied
  - If aggregate function gets an empty multiset for input, the result is null (except for count, count returns 0)

Null Values-Example

loan_number	branch_name	amount	customer_name
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes

- $g_{\textbf{sum}(amount)}(r)$

Sum(amount)
10,000

- $g_{\textbf{count}(amount)}(r)$

Count(amount)
3

Relational Model

- Structural Concepts & Terminology of Relational Databases
- Relational Algebra Operations
  - Fundamental Relational-Algebra-Operations
  - Additional Relational-Algebra-Operations
  - Extended Relational-Algebra-Operations
- Null Values
- Modification of the Database

Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations are expressed using the assignment operator ($\leftarrow$)

Deletion

- A delete request is expressed similarly to a query, except instead of displaying tuples to the user, the selected tuples are removed from the database
- Can delete only whole tuples; cannot delete values on only particular attributes
- A deletion is expressed in relational algebra by:

$$r \leftarrow r - E$$

where r is a relation and E is a relational algebra query

Deletion Examples

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Delete all account records in the Perryridge branch

$$account \leftarrow account - \sigma_{branch\_name = \text{“}perryridge\text{''}}(account)$$

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

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

Deletion Examples

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- Delete all accounts at branches located in Needham (reference key)

  $r_1 \leftarrow \sigma_{branch\_city = \text{``}Needham\text{''}}(account \bowtie branch)$
  $r_2 \leftarrow \prod_{account\_number, branch\_name, balance}(r_1)$
  $r_3 \leftarrow \prod_{customer\_name, account\_number}(r_2 \bowtie depositor)$
  $account \leftarrow account - r_2$
  $depositor \leftarrow depositor - r_3$

Insertion

- To insert data into a relation, we either:
  - specify a tuple to be inserted
  - 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

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

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

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

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

Insertion Examples

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
loan (loan_number, branch_name, amount)
depositor (customer_name, account_number)
borrower (customer_name, loan_number)

- 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

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

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

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

Updating

- A mechanism to change a value in a tuple without charging *all* values in the tuple
- Use the generalized projection operator to do this task

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

- Each F_i is either
  - the I^{th} attribute of r, if the I^{th} attribute is not updated, or,
  - if the attribute is to be updated F_i is an expression, involving only constants and the attributes of r, which gives the new value for the attribute

Update Examples

- Make interest payments by increasing all balances by 5 percent

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

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

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