PURDUE UNIVERSITY

CS 542: Distributed Database Systems

Relational Database Overview

Prof. Chris Clifton

14 January 2009

Indiana Center for Database Systems

Based on 3 Key Features

- Simple data structures: 2-dimensional tables – physical data independence
- Solid foundation for consistency – normalization, integrity rules
- Set-oriented manipulation of relations – relational algebra, calculus, and SQL

Basic Concepts

- A database is a structured collection of data related to some real-life enterprise.
- A relational database structures the data in the form of tables.
- A relation R, defined over n sets $D_1, \ldots D_n$, is a set of tuples $<d_1, d_2, \ldots, d_n>$ such that d_i belongs to D_i

Example

- Employees relation:
  - EMP(ENO, ENAME, TITLE, SAL, PNO, RESP, DUR)
  - PROJ(PNO, PNAME, BUDGET)

RDBMS terms

- Tables or relations
- Schema
- Attributes
- Tuples
- Null Value

 5

Keys

- A key is a minimal nonempty subset of attributes of a relation that uniquely identify each tuple of the relation.
- A superkey is a superset of a key
- A primary key is a specially designated key of the relation.
- All others keys are called candidate keys.

 6

Normalization

- Possible anomalies in the schema include:
  - Repetition anomaly: e.g. name, title, and salary of every employee repeated for each project – wastage of space.
  - Update anomaly: due to repetition, multiple rows need to be changed
  - Insertion anomaly: it may not be possible to add new information to the database, e.g. a new employee can't be added unless assigned to a project
  - Deletion anomaly: unnecessary deletion of data, e.g. if an employee works only on one project, and it is cancelled, then we lose information about the employee.

 7

Normalization

- Normalization transforms arbitrary schemas into ones without these problems.
- Achieved through decomposition, and functional dependencies.
- Several normal forms are defined, e.g 3NF, BCNF, etc.
- Want to have a lossless-join, and dependency preserving decomposition.

 8

Normalization

- We say that A functionally determines B if (where A and B are sets of attributes of relation R), for each value of A in R, there is only one value of B in R, or A → B.
  - PNO → (PNAME, BUDGET)
  - (ENO, PNO) → (ENAME, TITLE, SAL, RESP,DUR)
- Decomposition attempts to eliminate dependencies within relations by breaking them up into multiple relations.

(C) Ozsu & Valduriez 9

Integrity Rules

- Specified through keys
- Every relation has a primary key – no duplicate combinations of these attributes
- Key attributes cannot be NULL
- Foreign keys ensure referential integrity
- Domain constraints.
- Explicit constraints can also be specified
  - e.g. CHECK (SAL < 200000)

(C) Ozsu & Valduriez 10

Relational Data Languages

- Relational Algebra
  - Operational: describes *how* to get the answer
- Relational Calculus
  - Declarative: describes *what answer is desired*, not how to obtain it
- SQL
  - Practical query language supported by RDBMS products.

(C) Ozsu & Valduriez 11

Example Instances

- "Sailors" and "Reserves" relations for our examples.
- We'll use positional or named field notation, assume that names of fields in query results are 'inherited' from names of fields in query input relations.

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

(C) Ramakrishnan and Gehrke 12

Relational Algebra

σ π

- Basic operations:
  - *Selection* (σ) Selects a subset of rows from relation.
  - *Projection* (π) Deletes unwanted columns from relation.
  - *Cross-product* ($\times$) Allows us to combine two relations.
  - *Set-difference* ($-$) Tuples in reln. 1, but not in reln. 2.
  - *Union* ($\cup$) Tuples in reln. 1 and in reln. 2.
- Additional operations:
  - Intersection, *join*, division, renaming: Not essential, but (very!) useful.
- Since each operation returns a relation, operations can be *composed* (Algebra is "closed".)

Projection

- Deletes attributes that are not in *projection list*.
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation.
- Projection operator has to eliminate *duplicates*! (Why??)
  - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$\pi_{sname,rating}(S2)$

age
35.0
55.5

$\pi_{age}(S2)$

Selection

- Selects rows that satisfy *selection condition*.
- No duplicates in result! (Why?)
- *Schema* of result identical to schema of (single) input relation.
- *Result* relation can be the *input* for another relational algebra operation! (*Operator composition.*)

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

$$\sigma_{rating>8}(S2)$$

sname	rating
yuppy	9
rusty	10

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be *union-compatible*:
  - Same number of fields.
  - 'Corresponding' fields have the same type.
- What is the *schema* of the result?

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup$ 2

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap$ 2

sid	sname	rating	age
22	dustin	7	45.0

$S1 -$ 2

Cross-Product

- Each row of S1 is paired with each row of R1.
- *Result schema* has one field per field of S1 and R1, with field names 'inherited' if possible.
  - *Conflict*: Both S1 and R1 have a field called *sid*.

(C) Ramakrishnan & Gehrke 17

Cross-Product

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Renaming operator: $\rho\,(C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1)$

(C) Ramakrishnan & Gehrke 18

Joins

- *Condition Join*: $R \bowtie_c S = \sigma_c(R \times S)$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

- *Result schema* same as that of cross-product.
- Fewer tuples than cross-product, might be able to compute more efficiently
- Sometimes called a *theta-join*.

Joins

- *Equi-Join*: A special case of condition join where the condition *c* contains only *equalities*.

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

$$S1 \bowtie_{sid} R1$$

- *Result schema* similar to cross-product, but only one copy of fields for which equality is specified.
- *Natural Join*: Equijoin on *all* common fields.

Relational Calculus

- Comes in two flavours: *Tuple relational calculus* (TRC) and *Domain relational calculus* (DRC).
- Calculus has *variables, constants, comparison ops*, *logical connectives* and *quantifiers*.
  - *TRC*: Variables range over (i.e., get bound to) *tuples*.
  - *DRC*: Variables range over *domain elements* (= field values).
  - Both TRC and DRC are simple subsets of first-order logic.
- Expressions in the calculus are called *formulas*. An answer tuple is essentially an assignment of constants to variables that make the formula evaluate to *true*.

Domain Relational Calculus

- *Query* has the form:

$$\left\{ \langle x1, x2, ..., xn \rangle \mid p\left(\langle x1, x2, ..., xn \rangle\right) \right\}$$

  - *Answer* includes all tuples $\langle x1, x2, ..., xn \rangle$ that make the *formula* $p\left(\langle x1, x2, ..., xn \rangle\right)$ *true*.
  - *Formula is recursively defined, starting with* simple *atomic formulae* (getting tuples from relations or making comparisons of values), and building bigger and better formulas using *logical connectives*.

DRC Formulas

- *Atomic formula:*
  - $\langle x1, x2, ..., xn \rangle \in Rname$, or *X op Y*, or *X op* constant
  - *op* is one of $<, >, =, \leq, \geq, \neq$
- *Formula:*
  - an atomic formula, or
  - $\neg p, p \wedge q, p \vee q$, where *p* and *q* are formulae, or
  - $\exists X(p(X))$, where variable *X* is *free* in *p(X)*, or
  - $\forall X(p(X))$, where variable *X* is *free* in *p(X)*
- The use of quantifiers $\exists X$ and $\forall X$ is said to *bind* *X*.
  - A variable that is not bound is free.

Free and Bound Variables

- The use of quantifiers $\exists X$ and $\forall X$ in a formula is said to *bind* *X*.
  - A variable that is not bound is free.
- Let us revisit the definition of a query:

$$\{\langle x1, x2, ..., xn \rangle \mid p(\langle x1, x2, ..., xn \rangle)\}$$

- There is an important restriction: the variables *x1, ..., xn* that appear to the left of '|' must be the **only** free variables in the formula *p(...)*.

Find all sailors with a rating above 7

$$\left\{ \langle I,N,T,A \rangle \mid \langle I,N,T,A \rangle \in Sailors \wedge T > 7 \right\}$$

- The condition $\langle I,N,T,A \rangle \in Sailors$ ensures that the domain variables *I, N, T* and *A* are bound to fields of the same Sailors tuple.
- The term $\langle I,N,T,A \rangle$ to the left of '|' (which should be read as *such that*) says that every tuple $\langle I,N,T,A \rangle$ that satisfies *T*>7 is in the answer.
- Modify this query to answer:
  - Find sailors who are older than 18 or have a rating under 9, and are called 'Joe'.

Find sailors who've reserved all boats

$$\left\{ \langle I,N,T,A \rangle \mid \langle I,N,T,A \rangle \in Sailors \wedge \right.$$
$$\forall B,BN,C \left(\neg \left(\langle B,BN,C \rangle \in Boats \right) \vee \right.$$
$$\left. \left. \left(\exists Ir,Br,D \left(\langle Ir,Br,D \rangle \in Reserves \wedge I = Ir \wedge Br = B \right) \right) \right) \right\}$$

- Find all sailors *I* such that for each 3-tuple $\langle B,BN,C \rangle$ either it is not a tuple in Boats or there is a tuple in Reserves showing that sailor *I* has reserved it.

Unsafe Queries, Expressive Power

- It is possible to write syntactically correct calculus queries that have an infinite number of answers! Such queries are called *unsafe*.
  - e.g., $\{ S \mid \neg (S \in Sailors) \}$
- It is known that every query that can be expressed in relational algebra can be expressed as a safe query in DRC / TRC; the converse is also true.
- *Relational Completeness*: Query language (e.g., SQL) can express every query that is expressible in relational algebra/calculus.

(C) Ramakrishnan & Gehrke 30

Basic SQL Query

SELECT [DISTINCT] *target-list*
FROM *relation-list*
WHERE *qualification*

- ***relation-list*** **A list of relation names (possibly with a *range-variable* after each name).**
- ***target-list*** **A list of attributes of relations in *relation-list***
- ***qualification*** **Comparisons (Attr *op* const or Attr1 *op* Attr2, where *op* is one of** $<, >, =, \leq, \geq, \neq$ **) combined using AND, OR and NOT.**
- **DISTINCT is an optional keyword indicating that the answer should not contain duplicates. Default is that duplicates are *not* eliminated!**

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of *relation-list.*
  - Discard resulting tuples if they fail *qualifications.*
  - Delete attributes that are not in *target-list.*
  - If DISTINCT is specified, eliminate duplicate rows.
- This strategy is probably the least efficient way to compute a query! An optimizer will find more efficient strategies to compute *the same answers.*

(C) Ramakrishnan & Gehrke 32

Other Operations

- UNION, INTERSECT, EXCEPT
- Nested Queries
- ANY, ALL, IN
- Aggregate operators: SUM, COUNT, MAX, MIN, AVG
- Grouping: GROUP BY, HAVING
- Embedded SQL

(C) Ramakrishnan & Gehrke 33