Ramakrishnan - Database Management Systems 3rd Edition

Part I FOUNDATIONS

- Chapter 1 — OVERVIEW OF DATABASE SYSTEMS
- Chapter 2 — INTRODUCTION TO DATABASE DESIGN
- Chapter 3 — THE RELATIONAL MODEL
- Chapter 4 — RELATIONAL ALGEBRA AND CALCULUS
  - Questions
    - What is the foundation for relational query languages like SQL? What is the difference between procedural and declarative languages?
      - In a procedural language, you define the whole process and provide the steps how to do it. You just provide orders and define how the process will be served. In a declarative language, you just set the command or order, and let it be on the system ho
    - What is relational algebra, and why is it important?
      - 4.2
    - What are the basic algebra operators, and how are they combined to write complex queries?
      - 4.2
    - What subset of mathematical logic is used in relational calculus, and how is it used to write queries?
      - 4.3
  - 4.1 PRELIMINARIES
    - **Relational Algebra** [ASCII: Sailors, Sailors.age, sal, rank, Reserves, Reserves.sailor, Reserves.bid, Reserves.day, sailor, bid, day]
    - 既可以用field name表示field，也可以用order表示
  - 4.2 RELATIONAL ALGEBRA
    - What is relational algebra, and why is it important?
    - What are the basic algebra operators, and how are they combined to write complex queries?
    - Selection and Projection
      - Relational algebra includes operators to select rows from a relation (σ) and to project columns from a relation (π). These operations allow us to manipulate data in a single relation. Consider, for instance, the table Sailors shown in Figure 4.1.
      - $\sigma_{rating>8}(S2)$
sid	sname	rating	age
28	yuppy	9	35.0
58	Rusty	10	35.0
        - The subscript rating> 8 specifies the selection criterion to be applied while retrieving tuples
    - Set Operations
      - The usual set operations—union, intersection, and cross-product—also apply to sets of tuples in a relation, and can be used to combine two relations.
      - Union
        - **Union:** R ∪ S returns a relation instance containing all tuples that occur in either relation instance R or relation instance S. Union is also defined for relation instances.
        - Two relation instances are said to be **union-compatible** if the following conditions hold
          - they have the same number of the fields, and
          - corresponding fields, taken in order from left to right, have the same domains.
      - Intersection
        - **Intersection:** R ∩ S returns a relation instance containing all tuples that occur in both relation instances R and relation instance S. Intersection is also defined for relation instances.
      - Set difference
        - Set Difference: R − S returns a relation instance containing all tuples that occur in relation instance R but not in relation instance S. Set difference is also defined for relation instances.
      - Cross product
        - Cross product: R × S returns a relation instance whose schema contains all the fields of R (in the same order as they appear in R) followed by all the fields of S (in the same order as they appear in S). The result of R × S contains one tuple r, q for each pair of tuples r ∈ R, q ∈ S. When R × S is computed in a query, we must rename at least one field of R and one field of S. The cross product operation is sometimes called Cartesian product.
    - Renaming
      - The expression ρ(R(F), E) takes an arbitrary relational algebra expression E and returns an instance of a (new) relation called R.
        - R contains the same tuples as the result of E, and has the same schema as E, but some fields are renamed
        - Renaming list F
          - oldname -> newname or position -> newname
    - Joins
      - a join can be defined as a cross-product followed by selec- tions and projections, joins arise much more frequently in practice than plain cross products
        - the result of a cross-product is typically much larger than the result of a join, and it is very important to recognize joins and implement them without materializing the underlying cross-product by applying the selections and projections 'on-the-fly'.
      - Condition Joins
        - a join condition c and a pair of relation instances as arguments and returns a relation instance
          - $R \bowtie_c S = \sigma_c(R \times S)$
      - Equijoin
        - join condition consists solely of equalities
      - Natural Join
        - on all fields having the same name
    - Division
      - the set of all x values (in the form of unary tuples) such that for every y value in (a tuple of) B, there is a tuple (x,y) in A.
      - disqualified tuples
        - $\pi_x((\pi_x(A) \times B) - A)$
      - **or can define** C as
        - $\pi_x(A) - \pi_x((\pi_x(A) \times B) - A)$
    - More Examples of Algebra Queries
  - 4.3 RELATIONAL CALCULUS
    - What subset of mathematical logic is used in relational calculus, and how is it used to write queries?
    - declarative language, in contrast to alge- bra, which is a procedural language. Explain the distinction.
    - In context to the algebra, which is procedural, the calculus is nonprocedural, or declarative,
      - Relational calculus has had a big influence on the design of commercial query languages such a.s SQL and, especially, Query- by-Example (QBE).
      - The variant of the calculus we present in detail is called the **tuple relational calculus (TRC)**.
    - Tuple Relational Calculus
      - Tuple variables ranging over relations can be used in a variety of ways. For example, we can require that every tuple assigned to a tuple variable be in a specified relation, for instance, Reserves or Sailors. This is done by simply stating this relation after the tuple variable in the query. We can also ask that every tuple assigned to a tuple variable be in some relation that is stored in the database; we do not specify the relation in this case. We can also ask that every tuple assigned to a tuple variable be in some relation that is the result of an expression over stored relations. We can ask that every tuple assigned to a tuple variable be in some relation that is the result of an expression over the tuples in other relations. We can ask that every tuple assigned to a tuple variable be in some relation that is the result of an expression over the tuples in the same relation. We can ask that every tuple assigned to a tuple variable be in some relation that is the result of an expression over some arbitrary tuples.
      - Syntax of TRC Queries
        - p(R) denotes a formula in which the variable R appears
          - any atomic formula
          - ¬p, p∧q, p∨q, or p⇒q
          - ∃X(p(X)), where X is a tuple variable
          - ∀X(p(X)), where X is a tuple variable
            - As quantifiers ∃ and ∀ can range over arbitrarily many tuples, it
              - A variable is said to be free in a formula or subformula
        - each variable in a TRC formula has a well-defined domain from which values for the variable are drawn. That is, each variable has a well-defined **type**.
        - A TRC query is defined to be expression of the form {T | p(T)}, where T is the only free variable in the formula p.
      - Semantics of TRC Queries
        - The answer to a TRC query {T | p(T)} is the set of all tuples t for which the formula p(T) evaluates to true with variable T assigned the tuple value t.
        - the given database instance, F evaluates to (or simply 'is') true if one of the following holds
          - F is a prime. Either F is R(t) or F is equal to t.R op u.S, where R and S are relations mentioned in the database, t is a tuple variable, and u is a tuple variable.
          - F is a conjunction G∧H, G∨H, or ¬G, where G and H are simpler formulas and G and H evaluate to true when the given database is used.
          - F is of the form ∃R(G), where R is a tuple variable and G is a formula that evaluates to true when the given database is used and R is assigned a tuple that is in the relation R.
          - F is of the form ∀R(G), where R is a tuple variable and G is a formula that evaluates to true when the given database is used and R is assigned any tuple that is in the relation R.
          - F is a formula of the form R op S, where R and S are tuples and op is a comparison operator (such as, =, ≤, or ≠) defined on the values in the fields of R and S.
      - Examples of TRC Queries
        - (Q11) Find the names and ages of sailors with a rating above 7.
          - {P | ∃S ∈ Sailors(S.rating > 7 ∧ P.name = S.sname ∧ P.age = S.age)}
        - (Q12) Find the names and ages of sailors with a rating above 7.
          - {P | ∃S ∈ Sailors(S.rating > 7 ∧ P.name = S.sname ∧ P.age = S.age)}
        - (Q13) Find the sailor_id, boat_id, and day for each reservation that satisfies the following two conditions: The sailor is older than 21 and the boat is not green. Or the sailor is younger than 21 and the boat is red. Ordinary arithmetic comparisons are used to evaluate the conditions in the query.

sailor	bid	day
T.sid1a	103	10/10/96
T.sid2a	102	10/10/96
T.sid3a	103	10/8/96
T.sid4a	104	10/7/96
T.sid5a	102	11/10/96
T.sid6a	103	11/6/96
T.sid7a	104	11/12/96
T.sid1b	101	9/1/96
T.sid2b	102	9/4/96
T.sid3b	103	9/9/96

        - (Q14) Find the names of sailors who reserved all boats.
          - {P | ∃S ∈ Sailors ∀B ∈ Boats (∃R ∈ Reserves(R.sid = S.sid ∧ R.bid = B.bid ∧ R.day = 'Aug 1, 1996'))}
        - (Q15) Find the names of sailors who reserved all boats called Interlake.
          - {P | ∃S ∈ Sailors ∀B ∈ Boats (B.bname = 'Interlake' ⇒ ∃R ∈ Reserves(R.sid = S.sid ∧ R.bid = B.bid ∧ R.day = 'Aug 1, 1996'))}
    - Domain Relational Calculus
      - A **domain variable** is a variable that ranges over the values in the domain of some attribute
        - e.g., the variable can be assigned an integer if it appears in an attribute whose domain is the set of integers
      - A DRC query has the form {[X1,X2, ..., Xn] | P(X1,X2, ..., Xn)}, where each Xi is either a domain variable or a constant and p(X1, X2, ..., Xn) denotes a **DRC formula**.
      - ∃ ∀R(R.a op S.b ∧ R.a op constant) ∨ ∃ ∀R(R.a op S.b ∧ R.a op constant)
        - R ∈ Rel
        - R.a op S.b
        - R.a op constant, or constant op R.a
      - A **formula** is recursively defined to be one of the following, where P and q are themselves formulas and p(X) denotes a formula in which the variable X appears:
        - any atomic formula
        - ¬p, p∧q, p∨q, or p⇒q
        - ∃X(p(X)), where X is a domain variable
        - ∀X(p(X)), where X is a domain variable
      - Examples of DRC Queries
        - (Q11) Find all sailors with a rating above 7.
          - {[I, N, R, A] | ∃ I, N, R, A (Sailors(I, N, R, A) ∧ R > 7)}
        - (Q12) Find the names and ages of sailors with a rating above 7.
          - {[N, A] | ∃ I, N, R, A (Sailors(I, N, R, A) ∧ R > 7)}
        - (Q13) Find the sailor id, boat id, and day for each reservation that satisfies the following two conditions: The sailor is older than 21 and the boat is not green. Or the sailor is younger than 21 and the boat is red. Ordinary arithmetic comparisons are used to evaluate the conditions in the query.
          - {[I, B, D] | ∃ N, R, A (Sailors(I, N, R, A) ∧ Reserves(I, B, D) ∧ ((A > 21 ∧ ¬∃ C (∃ BN, CL (Boats(B, BN, CL) ∧ C = 'green'))) ∨ (A < 21 ∧ ∃ C (∃ BN, CL (Boats(B, BN, CL) ∧ C = 'red')))))}
        - (Q14) Find the names of sailors who reserved all boats.
          - {[N] | ∃ I, R, A (Sailors(I, N, R, A) ∧ ∀ B, BN, CL (∃ D (Boats(B, BN, CL) ∧ Reserves(I, B, D))))}
        - (Q15) Find the names of sailors who reserved all boats called Interlake.
          - {[N] | ∃ I, R, A (Sailors(I, N, R, A) ∧ ∀ B, BN, CL (Boats(B, BN, 'Interlake') ∧ ∃ D (Reserves(I, B, D))))}
    - EXPRESSIVE POWER OF ALGEBRA AND CALCULUS
      - We therefore define a safe TRC formula Q to be a formula such that:
        - 1. For any given I, the set of answers for Q contains only values that are in Dom(Q, I).
        - 2. For each subexpression of the form ∃R(p(R)) in Q, if a tuple r (assigned to variable R) makes the formula true, then r contains only constants in Dom(Q, I).
        - 3. For each subexpression of the form ∀R(p(R)) in Q, if a tuple r (assigned to variable R) contains a constant that is not in Dom(Q, I), then r must make the formula true.
      - Dom(Q, I) be the set of all constants that appear in these relation instances I or in the formulation of the query Q itself.
      - If a query language can express all the queries that we can express in relational algebra, it is said to be relationally complete
  - 4.4 EXPRESSIVE POWER OF ALGEBRA AND CALCULUS
- Chapter 5 — SQL: QUERIES, CONSTRAINTS, TRIGGERS

Part II APPLICATION DEVELOPMENT

Part III STORAGE AND INDEXING

Part IV QUERY EVALUATION

Part V TRANSACTION MANAGEMENT

Part VI DATABASE DESIGN AND TUNING

- Chapter 19 — SCHEMA REFINEMENT AND NORMAL FORMS
- Chapter 20 — PHYSICAL DATABASE DESIGN AND TUNING
- Chapter 21 — SECURITY AND AUTHORIZATION

Part VII ADDITIONAL TOPICS

- Chapter 22 — PARALLEL AND DISTRIBUTED DATABASES
- Chapter 23 — OBJECT-DATABASE SYSTEMS
- Chapter 24 — DEDUCTIVE DATABASES
- Chapter 25 — DATA WAREHOUSING AND DECISION SUPPORT
- Chapter 26 — DATA MINING
- Chapter 27 — INFORMATION RETRIEVAL AND XML DATA
- Chapter 28 — SPATIAL DATA MANAGEMENT
- Chapter 29 — FURTHER READING
- Chapter 30 — THE MINIBASE SOFTWARE