Relational Calculus

- A **relational calculus** expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in **tuple calculus**) or over columns of the stored relations (in **domain calculus**).
- In a calculus expression, there is *no order of operations* to specify how to retrieve the query result—a calculus expression specifies only what information the result should contain.
  - This is the main distinguishing feature between relational algebra and relational calculus.

Relational Calculus (continued)

- Relational calculus is considered to be a **nonprocedural** or **declarative** language.
- This differs from relational algebra, where we must write a *sequence of operations* to specify a retrieval request; hence relational algebra can be considered as a **procedural** way of stating a query.

Tuple Relational Calculus

- The tuple relational calculus is based on specifying a number of tuple variables.
- Each tuple variable usually ranges over a particular database relation, meaning that the variable may take as its value any individual tuple from that relation.
- A simple tuple relational calculus query is of the form

  {t | COND(t)}

  - where t is a tuple variable and COND (t) is a conditional expression involving t.
  - The result of such a query is the set of all tuples t that satisfy COND (t).

Tuple Relational Calculus (continued)

- Example: To find the first and last names of all employees whose salary is above $50,000, we can write the following tuple calculus expression:

{t.FNAME, t.LNAME | EMPLOYEE(t) AND t.SALARY>50000}

- The condition EMPLOYEE(t) specifies that the **range relation** of tuple variable t is EMPLOYEE.
- The first and last name (PROJECTION $\pi_{FNAME, LNAME}$) of each EMPLOYEE tuple t that satisfies the condition t.SALARY>50000 (SELECTION $\sigma_{SALARY >50000}$) will be retrieved.

The Existential and Universal Quantifiers

- Two special symbols called quantifiers can appear in formulas; these are the universal quantifier (∀) and the existential quantifier (∃).
- Informally, a tuple variable t is bound if it is quantified, meaning that it appears in an (∀ t) or (∃ t) clause; otherwise, it is free.
- If F is a formula, then so are (∃ t)(F) and (∀ t)(F), where t is a tuple variable.
  - The formula (∃ t)(F) is true if the formula F evaluates to true for some (at least one) tuple assigned to free occurrences of t in F; otherwise (∃ t)(F) is false.
  - The formula (∀ t)(F) is true if the formula F evaluates to true for every tuple (in the universe) assigned to free occurrences of t in F; otherwise (∀ t)(F) is false.

The Existential and Universal Quantifiers (continued)

- ∀ is called the universal or “for all” quantifier because every tuple in “the universe of” tuples must make F true to make the quantified formula true.
- ∃ is called the existential or “there exists” quantifier because any tuple that exists in “the universe of” tuples may make F true to make the quantified formula true.

Example Query Using Existential Quantifier

- Retrieve the name and address of all employees who work for the 'Research' department. The query can be expressed as :

{t.FNAME, t.LNAME, t.ADDRESS | EMPLOYEE(t) and (∃ d) (DEPARTMENT(d) and d.DNAME='Research' and d.DNUMBER=t.DNO) }

- The only *free tuple variables* in a relational calculus expression should be those that appear to the left of the bar (|).
  - In above query, t is the only free variable; it is then *bound successively* to each tuple.
- If a tuple *satisfies the conditions* specified in the query, the attributes FNAME, LNAME, and ADDRESS are retrieved for each such tuple.
  - The conditions EMPLOYEE (t) and DEPARTMENT(d) specify the range relations for t and d.
  - The condition d.DNAME = 'Research' is a selection condition and corresponds to a SELECT operation in the relational algebra, whereas the condition d.DNUMBER = t.DNO is a JOIN condition.

Example Query Using Universal Quantifier

- Find the names of employees who work on *all* the projects controlled by department number 5. The query can be:

{e.LNAME, e.FNAME | EMPLOYEE(e) and ((∀ x)(not(PROJECT(x)) or not(x.DNUM=5)

OR ((∃ w)(WORKS_ON(w) and w.ESSN=e.SSN and x.PNUMBER=w.PNO))))}

- Exclude from the universal quantification all tuples that we are not interested in by making the condition true *for all such tuples*.
  - The first tuples to exclude (by making them evaluate automatically to true) are those that are not in the relation R of interest.
- In query above, using the expression **not(PROJECT(x))** inside the universally quantified formula evaluates to true all tuples x that are not in the PROJECT relation.
  - Then we exclude the tuples we are not interested in from R itself. The expression not(x.DNUM=5) evaluates to true all tuples x that are in the project relation but are not controlled by department 5.
- Finally, we specify a condition that must hold on all the remaining tuples in R.

((∃ w)(WORKS_ON(w) and w.ESSN=e.SSN and x.PNUMBER=w.PNO)

Languages Based on Tuple Relational Calculus

- The language **SQL** is based on tuple calculus. It uses the basic block structure to express the queries in tuple calculus:
  - SELECT <list of attributes>
  - FROM <list of relations>
  - WHERE <conditions>
- SELECT clause mentions the attributes being projected, the FROM clause mentions the relations needed in the query, and the WHERE clause mentions the selection as well as the join conditions.
  - SQL syntax is expanded further to accommodate other operations. (See Chapter 8).

Languages Based on Tuple Relational Calculus (continued)

- Another language which is based on tuple calculus is **QUEL** which actually uses the range variables as in tuple calculus. Its syntax includes:
  - RANGE OF <variable name> IS <relation name>
- Then it uses
  - RETRIEVE <list of attributes from range variables>
  - WHERE <conditions>
- This language was proposed in the relational DBMS INGRES. (system is currently still supported by Computer Associates – but the QUEL language is no longer there).

The Domain Relational Calculus

- Another variation of relational calculus called the domain relational calculus, or simply, domain calculus is equivalent to tuple calculus and to relational algebra.
- The language called QBE (Query-By-Example) that is related to domain calculus was developed almost concurrently to SQL at IBM Research, Yorktown Heights, New York.
  - Domain calculus was thought of as a way to explain what QBE does.
- Domain calculus differs from tuple calculus in the type of variables used in formulas:
  - Rather than having variables range over tuples, the variables range over single values from domains of attributes.
- To form a relation of degree n for a query result, we must have n of these domain variables— one for each attribute.

The Domain Relational Calculus (continued)

- An expression of the domain calculus is of the form

$$\{ x_1, x_2, \ldots, x_n \mid \mathbf{COND}(x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m})\}$$

  - where $x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ are domain variables that range over domains (of attributes)
  - and COND is a condition or formula of the domain relational calculus.

Example Query Using Domain Calculus

Retrieve the birthdate and address of the employee whose name is ‘John B. Smith’.

- Query :

{uv | (∃ q) (∃ r) (∃ s) (∃ t) (∃ w) (∃ x) (∃ y) (∃ z)
(EMPLOYEE(qrstuvwxyz) and q=’John’ and r=’B’ and s=’Smith’)}

- **Abbreviated notation EMPLOYEE(qrstuvwxyz)** uses the variables without the separating commas: **EMPLOYEE(q,r,s,t,u,v,w,x,y,z)**

- Ten variables for the employee relation are needed, one to range over the domain of each attribute in order.
  - Of the ten variables q, r, s, . . ., z, only u and v are free.
- Specify the *requested attributes*, BDATE and ADDRESS, by the free domain variables u for BDATE and v for ADDRESS.
- Specify the condition for selecting a tuple following the bar (|)—
  - namely, that the sequence of values assigned to the variables qrstuvwxyz be a tuple of the employee relation and that the values for q (FNAME), r (MINIT), and s (LNAME) be ‘John’, ‘B’, and ‘Smith’, respectively.

Chapter Summary

- Relational Algebra
  - Unary Relational Operations
  - Relational Algebra Operations From Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
  - Examples of Queries in Relational Algebra
- Relational Calculus
  - Tuple Relational Calculus
  - Domain Relational Calculus