Database Management Systems

ICT1212

Relational Calculus

Department of ICT

Faculty of Technology

University of Ruhuna

Lecture 7

Part 02

What we Discuss Today

- Example Database Application (COMPANY)
- Relational Calculus
  - Tuple Relational Calculus
  - Domain Relational Calculus
- Overview of the QBE language

Summery of Relational Algebra

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

Example : COMPANY

Schema diagram for the COMPANY relational database schema; the primary keys are underlined.

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

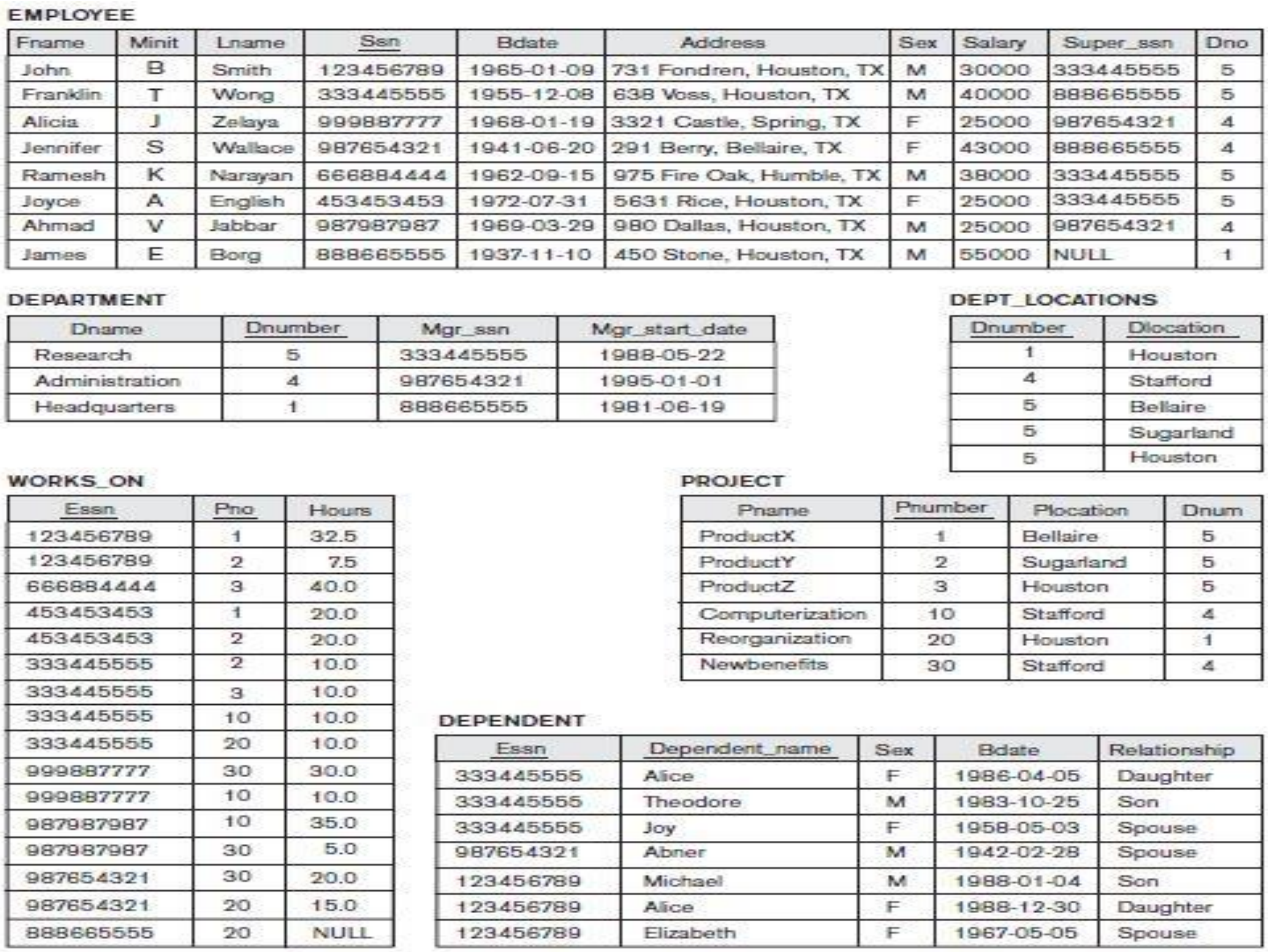
WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Example : COMPANY



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

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 is considered to be a **nonprocedural** 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

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 $\pi_{\text{FNAME, LNAME}}$) of each EMPLOYEE tuple t that satisfies the condition

t.SALARY>50000 (SELECTION $\sigma_{\text{SALARY} > 50000}$) 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 is (∃ 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.**
- If F is a formula, then so is (∀ t)(F), where t is a tuple variable. 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.**

  It 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.

Example Query Using Existential Quantifier

- Retrieve the name and address of all employees who work for the 'Research' department.

  Query :

  { 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.

  Query :
  {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)

Example Query Using Existential/ Universal Quantifier

- List the names of employees who have no dependents.

Query with Existential quantifier :

{e.Fname, e.Lname | EMPLOYEE(e) and (NOT (∃d)(DEPENDENT(d) and e.Ssn=d.Essn))}

Using the general transformation rule, we can rephrase above query as follows:

Same Query with Universal quantifier :

{e.Fname, e.Lname | EMPLOYEE(e) and ((∀d)(NOT(DEPENDENT(d)) or NOT(e.Ssn=d.Essn)))}

Languages Based on Tuple Relational Calculus

- The language **SQL** is based on tuple calculus. It uses the basic

  SELECT <list of attributes>

  FROM <list of relations>

  WHERE <conditions>

  block structure to express the queries in tuple calculus where the 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.

- 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.

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.

- 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.

- An expression of the domain calculus is of the form

  {x1, x2, . . ., xn | COND(x1, x2, . . ., xn, xn+1, xn+2, . . .,xn+m)}

  where x1, x2, . . ., xn, xn+1, xn+2, . . ., xn+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’)}

- 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.

Example Query Using Domain Calculus

- Retrieve the name and address of all employees who work for the ‘Research’ department.

  Query :

  {q, s, v | (∃z) (∃l) (∃m) (EMPLOYEE(qrstuvwxyz) and
  DEPARTMENT(lmno) and
  l=‘Research’ AND m=z)}

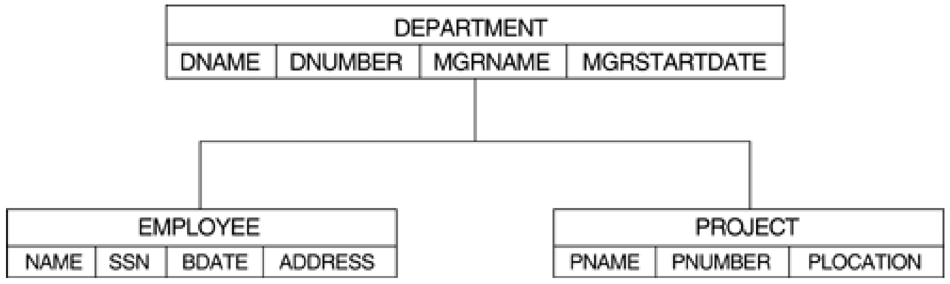
- A condition relating two domain variables that range over attributes from two relations,
- such as m = z in Q1, is a join condition, whereas a condition that relates a
- domain variable to a constant, such as l = ‘Research’, is a selection condition..

QBE: A Query Language Based on Domain Calculus

- 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.
- This language is based on the idea of giving an example of a query using **example elements**.
- An example element stands for a domain variable and is specified as an example value preceded by the underscore character.
- P. (called **P dot**) operator (for "print") is placed in those columns which are requested for the result of the query.
- A user may initially start giving actual values as examples, but later can get used to providing a minimum number of variables as example elements.
- The language is very user-friendly, because it uses minimal syntax.
- QBE was fully developed further with facilities for grouping, aggregation, updating etc. and is shown to be equivalent to SQL.
- The language is available under QMF (Query Management Facility) of DB2 of IBM and has been used in various ways by other products like ACCESS of Microsoft, PARADOX.

QBE Examples

- QBE initially presents a relational schema as a “blank schema” in which the user fills in the query as an example:

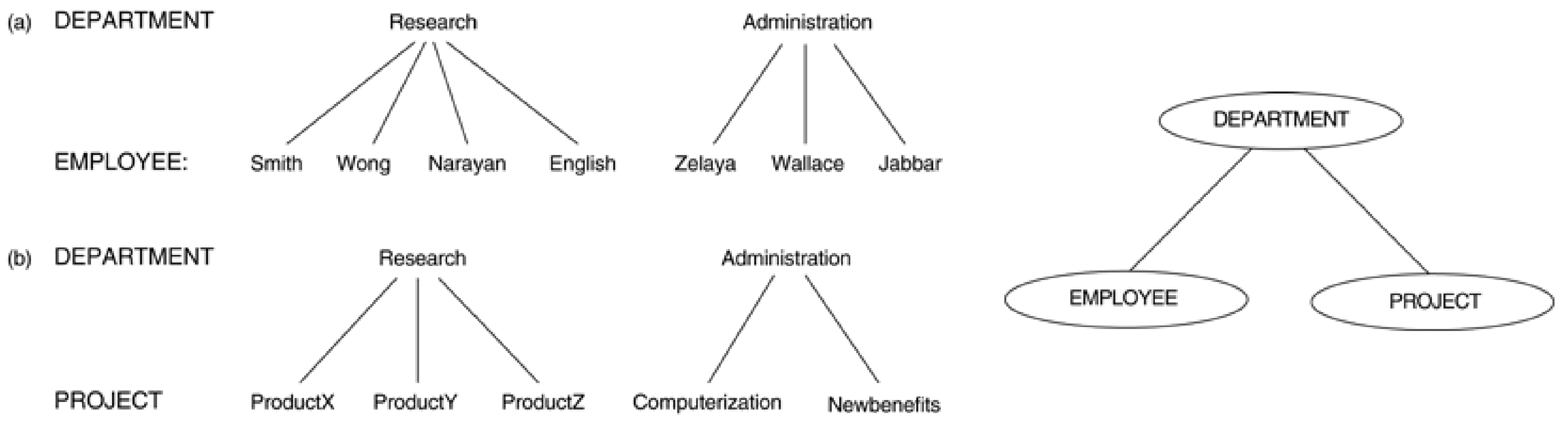


QBE Examples

- The following domain calculus query can be successively minimized by the user as shown:

Query :

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



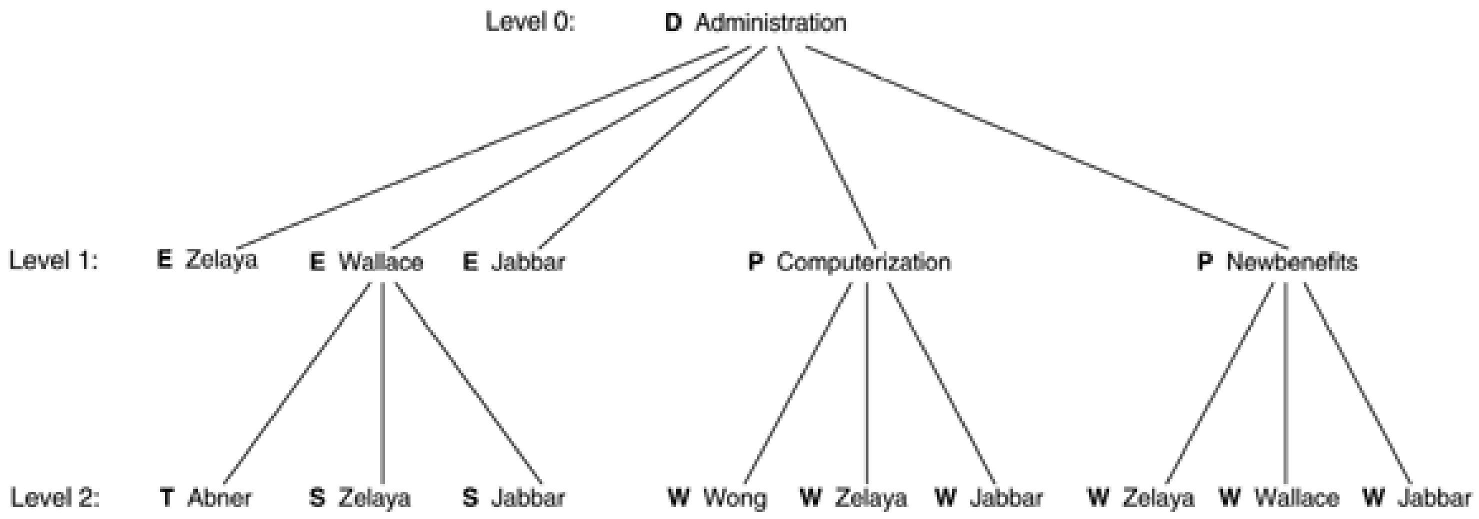
QBE Examples

Specifying complex cinditions in QBE:

- A technique called the “condition box” is used in QBE to state more involved Boolean expressions as conditions.
- The D.4(a) gives employees who work on either project 1 or 2, whereas the query in D.4(b) gives those who work on both the projects.

QBE Examples

- Illustrating join in QBE. The join is simple accomplished by using the same example element in the columns being joined. Note that the Result is set us as an independent table.



Reference

- *Chapter 6 - Fundamentals of Database Systems*

(6th Edition) By Remez Elmasri & Shamkant B. Navathe

Questions ???

Thank You