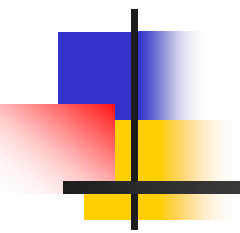
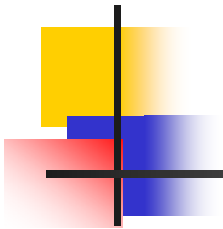
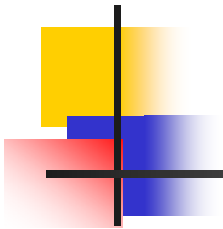


# Relational Calculus



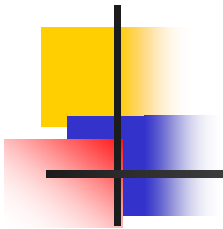
# Objectives

- Introduction +
- Tuple Relational Calculus +
- Domain relational Calculus +
- QBE +



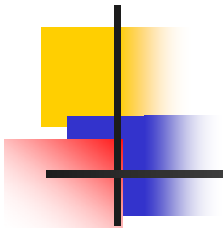
# - Introduction

- Relational calculus is a formal query language where we write one declarative expression to specify a retrieval request.
- A calculus expression specifies *what* is to be retrieved rather than *how* to retrieve it. Therefore, relational calculus is considered to be a **nonprocedural** language.
- There are two types of relational calculus:
  - Tuple relational calculus
  - Domain relational calculus.



# - Tuple Relational Calculus

- Definition +
- Relational Calculus Expression +
- Relational Calculus Atoms +
- Relational Calculus Formulas +
- Existential and Universal Qualifiers +
- Transformation of Universal and Existential Qualifiers +
- Safe Expression +
- Examples +



# -- Definition

- The **tuple relational calculus** is based on specifying a number of tuple variables. Each tuple variable usually ranges over a particular database relation.
- A tuple expression is written as *{t | f(t) }*

  Where t is a tuple variable

  *f(t)* is a conditional expression involving *t.*
- Example: Find all employees whose salary > 50,000.

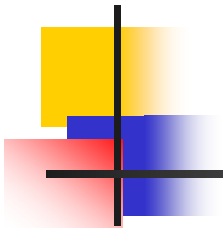
  **{t | employee(t) AND t.salary > 50000}**

  Note: The condition *employee(t)* specifies that the **range relation** of tuple variable *t* is *employee.*

# --Tuple Relational Calculus Expressions

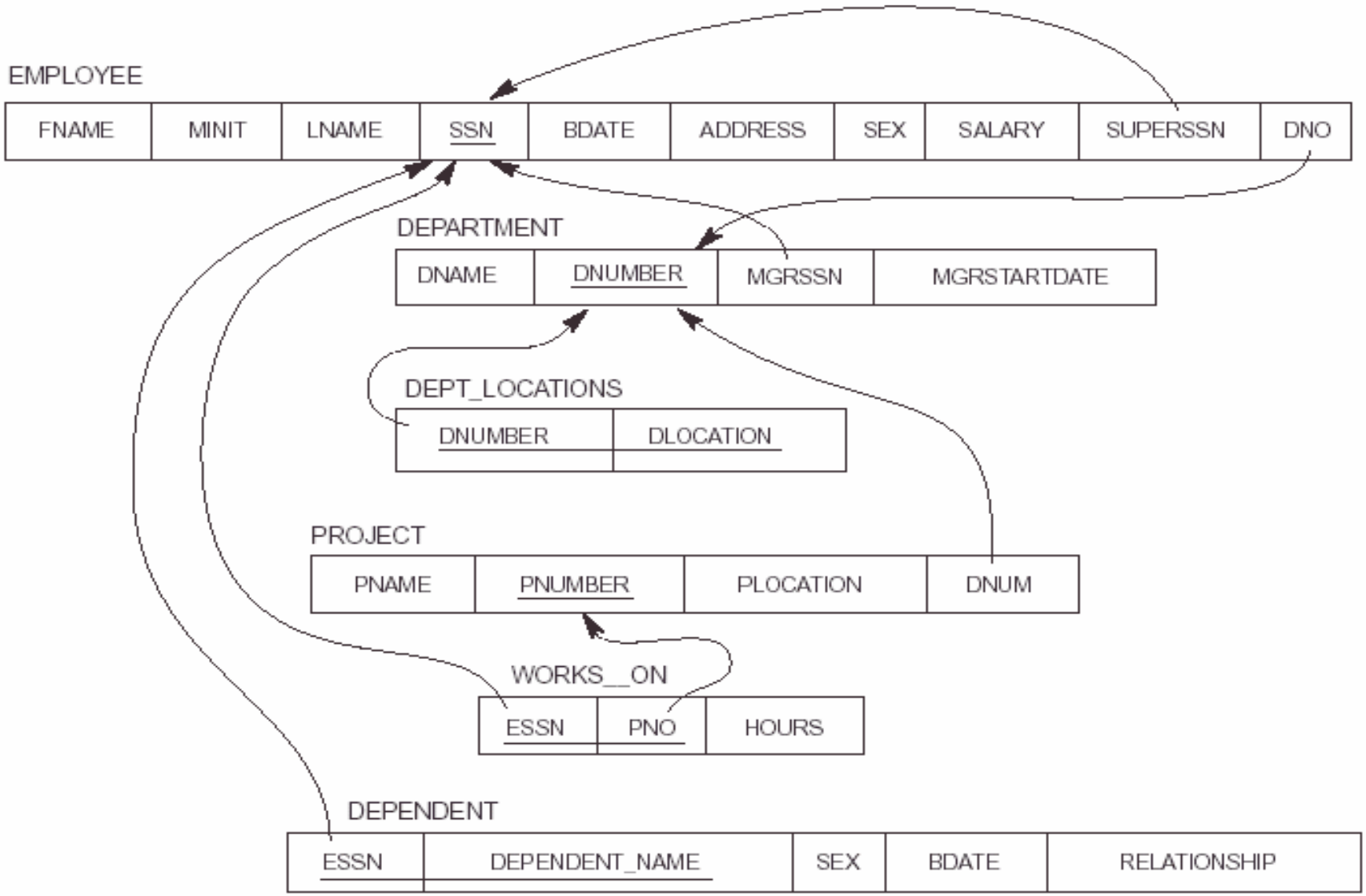
- A general expression of a tuple relational calculus is of the form:
  $\{t_i.A, t_j.B, \ldots \mid f(t_i, t_j, \ldots)\}$

  Where:
  - $t_i$, $t_j$, ... are tuple variables
  - $A$, $B$, ... is an attribute of the corresponding relation on which $t_i$, $t_j$, ... ranges.
  - $f$ is a condition or a **formula** of the tuple relational calculus.
- In Relational calculus a **safe expression** is the one guaranteed to yield a finite number of tuples otherwise the expression is **unsafe**.
- Example: $\{t \mid NOT(employee(t))\}$ is unsafe expression.



# -- Tuple Relational Calculus Atoms

- An **atom** is a building block of a relational calculus expression.
- An atom can have in one of the following forms:
  - **R($t_i$):** where R is a relation name. This atom specifies the range of tuple variable ti.
  - **$t_i$.*A* op $t_j$.B**: where op is one of the comparison operators.
  - **$t_i$.*A* op *c*** or ***c* op $t_i$.B**: where op is one of the comparison operators and c is a constant value.
- Each atom evaluates to either true or false for a specific value of tuples – called **the truth value** of an atom.


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

# --- Examples ...

- Retrieve all employees.

```
{
    e
    | employee(e)
}
```

- Retrieve the names of all employees.

```
{
    e.fname, e.lname
    | employee(e)
}
```

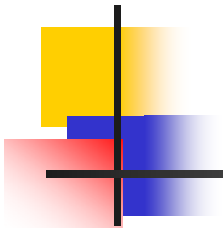
# … --- Examples

- Retrieve employees with salary greater than 5000.

```
{
    e
    |        employee(e)
     AND     e.salary > 5000
}
```

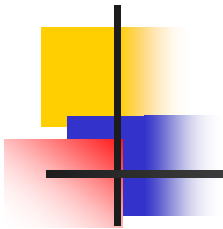
- Retrieve the names and salary of all employees who work in department 1 and whose salary > 5000

```
{
    e.fname, e.lname
    |        employee(e)
     AND dno = 1
     AND salary > 5000
}
```



# -- Tuple Relational Calculus Formulas

- A formula (condition) is made up of one or more atoms connected via the logical operators: *AND*, *OR*, and *NOT*.
- A formula can be recursively defined as:
  - Every atom is a formula
  - If *F* and *G* are formulas, then so are the following:
    - *F AND G*
    - *F OR G*
    - *NOT F*
    - *NOT G*



# -- Universal and Existential Quantifiers

- Two quantifiers symbols may appear in a formula:
  - The existential quantifier (∃)
  - The universal quantifier (∀)
- The truth values of formula with quantifiers is based on the concept of free and bound tuple variables in the formula.

# --- Free and Bound Tuple variables

- An occurrence of a tuple variable *t* in a formula *F* that is an atom is free in *F*.
- An occurrence of a tuple variable *t* is free or bound in a formula made up of logical connectives - (*F AND G*), (*F OR G*), (NOT F), and (*NOT G*) – depending whether it is free or bound in *F* or *G*.
- In the formula of the form *F = (G and H) or F = (G OR H),* a tuple variable may be free in *G* and bound in *H*, or vise versa. In this case, one occurrence of the tuple variable is bound and the other is free in *F*
- All free occurrences of a tuple variable *t* in *F* are bound in a formula *F= (∃t)(G) or F = (∀t)(G).* The tuple variable is bound to the quantifier specified in *F*.

# --- Truth Value of a Formula With Quantifier

- 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 occurrence 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 occurrence of t in F, otherwise *(∀t)(F)* is false.

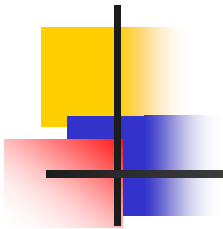
# --- Transforming the Universal and Existential Quantifiers

- (∀x)(F(x)) == NOT (∃x) (NOT (F(x)))
- (∀x)(F(x)) == NOT (∃x)(NOT(F(x)))
- (∀x)((F(x) AND G(x)) == NOT (∃x)(NOT(F(x)) OR NOT (G(x)))
- (∀x)((F(x) OR G(x)) == NOT (∃x)(NOT(F(x)) AND NOT (G(x)))
- (∃x)((F(x) OR G(x)) == NOT (∀x)(NOT(F(x)) AND NOT (G(x)))
- (∃x)((F(x) AND G(x)) == NOT (∀x)(NOT(F(x)) OR NOT (G(x)))
- (∀x)(F(x)) ⇨ (∃x)(F(x))
- NOT (∀x)(F(x)) ⇨ NOT (∃x)(F(x))
- Note:
  - The symbol == means equivalent
  - The symbol ⇨ means implies

# --- Examples ...

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

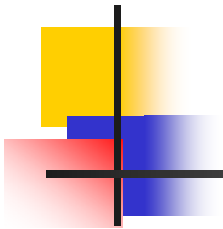
```
{
  t.fname, t.lname, t.address
  | employee(t)
  AND (  d)
              (
                  department(d)
                  AND d.dname = 'Research'
                  AND d.dnumber = t.dno
               )
}
```



# … ---- Examples …

- Find the names of employees who have no dependents.

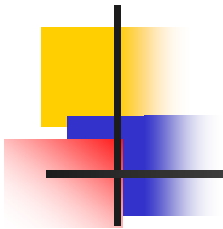
```
{
  e.fname, e.lname
  | employee(e)
   AND  NOT (  d)
                (
                  dependent(d)
                  AND e.ssn = d.essn
                )
}
```



# … ---- Examples

- List the names of managers who have at least one dependent.

```
{
   e.fname, e.lname
   | employee(e)
    AND  (  d) (  p)
                        (
                            department(d) AND dependent(p)
                            AND e.ssn = d.mgrssn
                            AND p.essn = e.ssn
                        )
}
```



# - Domain Relational Calculus

- The domain relational calculus uses variables that range over single values from domains of attributes.
- In this section we will cover:
  - Domain Relational Calculus Expression
  - Domain Relational Calculus Atom
  - Examples

# -- Domain relational Calculus Expression

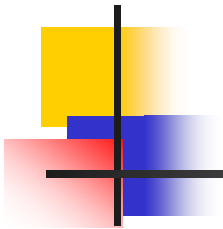
- Domain relational calculus expression can be written as:

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

Where $x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+m}$ are domain variables that range over domain of attributes. $f$ is a condition or formula of domain relational calculus.

# -- Domain relational Calculus Atom

- An atom is a building block of relational calculus expression.
- An atom can be in one of the following forms:
  - An atom of the form $R(x_1, x_2, ..., x_j)$, where R is a name of a relation of degree $j$ and each $x_i$, for $1 <= i <= j$, is a domain variable.
  - An atom of the form $x_i$ *op* $x_j$, where op is one of the comparison operators (except ≠) and $x_i$ and $x_j$ are domain variables.
  - An atom of the form $x_i$ op $c$ or $c$ op $x_j$, where op is one of the comparison operators (except ≠) and $x_i$ and $x_j$ are domain variables and $c$ is a constant value.



# -- Examples ...

- Retrieve the birth date and address of the employee

```
{
   uv
   | ( q) ( r) ( s) ( t) ( w) ( x) ( y) ( z)
   (
       employee(qrstuvwxyz)
    )
}
```

# -- Examples ...

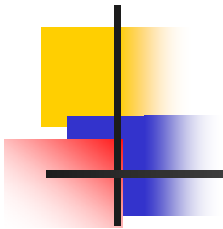
- Retrieve the birth date and address of the employee whose name is 'Adel M. Ali'.

```
{
   uv
   | ( q) ( r) ( s) ( t) ( w) ( x) ( y) ( z)
   (
       employee(qrstuvwxyz)
       AND q = 'Adel'
       AND r = 'M'
       AND s = 'Ali'
   )
}
```

# … -- Examples …

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

```
{
   qsv
   | ( z) ( k) ( m)
                    (
                        employee(qrstuvwxyz) AND department(kmno)
                        AND k  = 'Research'
                        AND m = z
                    )
}
```



# … -- Examples …

- Retrieve the name of employees who have no dependents.

```
{
   qs
     | (  t)
            (
                employee(qrstuvwxyz)
                AND NOT(  k)
                                (
                                   dependent(kmnop)
                                   AND t = k
                                )
            )
}
```

# … -- Examples …

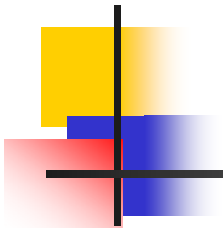
- Retrieve the name of employees who have no dependents.

```
{
   qs
     | (  t)
           (
               employee(qrstuvwxyz)
               AND (  k)
                       (
                           NOT(dependent(kmnop))
                           OR NOT ( t = k)
                       )
           )
}
```

# … ---- Examples

- List the names of managers who have at least one dependent.

```
{
    sq
    | (  t) (  j) (  a)
                    (
                        employee(qrstuvwxyz)
                        AND department(hijk)
                        AND dependents(abcde)
                        AND t = j
                        AND a = t
                    )
}
```



# - QBE

- Query-By-Example (QBE) language is a graphical query language with minimum syntax developed for database systems.
- In QBE, a query is formulated by filling in templates of relations that display on a monitor screen.
- Constants or example elements (a QBE term) can be filled in the columns of the template of that relation.

# -- Example of QBE Queries

- Retrieve the birth date and address of Adel M. Ali

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Superssn | Dno |
|---|---|---|---|---|---|---|---|---|---|
| Adel | M | Ali | | ✓ | ✓ | | | | |

**Employee**

- Retrieve the birth date and address of Adel M. Ali

**Works_on**

| Essn | Pno | hours |
|---|---|---|
| ✓ | 1 | >20 |
| | 2 | >20 |

# -- Example of QBE Queries

- Retrieve the birth date and address of Adel M. Ali

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Superssn | Dno |
|---|---|---|---|---|---|---|---|---|---|
| Adel | M | Ali | | ✓ | ✓ | | | | |

**Employee**