6.1 INTRODUCTION

As we know that in systems which support only classes, a class can be involved in 'is-a', 'class-attribute' and 'part-of' hierarchies. This leads to many difficulties in defining a query model for object-oriented databases. Different approaches have been taken by various systems. For example, IRIS [Fishman 1987] uses OSQL (Object Structured Query Language) which is an extension to SQL [Date 1987]. POSTGRES [Stonebraker 1990] uses POSTQUEL which is an extension to QUEL [Date 1987]. The query model developed for ORION [Banerjee 1987] is the first based on object-oriented paradigm.

The query model developed for ORION [Banerjee 1988] has some limitations. First, the result of an object-oriented query against a single class must return either all attributes or only one attribute of that class. This limitation is due to the fact that the result of a query has to be an instance of one of the existing classes. Second, a query against multiple classes in an object-oriented schema is disallowed. These limitations prohibit operations like relational project, join, and set-theoretic operations.
In this chapter, following the approach of separating types and classes, we extend the query model of ORION [Banerjee 1988] by defining relational algebraic operations (select and project) and set-theoretic operations (union, intersection, difference and cartesian product) on complex objects, and allowing new classes to be defined by applying these operations. That is, we define relational algebra operations and set-theoretic operations on classes having "is-part-of" relationship between classes [Bhal 1991c]. This extended query model allows a query against multiple target classes, i.e., a query that corresponds to relational join and set operations. It also enables us to retrieve complex objects and extract attributes from complex object hierarchy equivalent to relational select and project operations respectively.

A different approach is taken in [Osborn 1989] and in this paper, new objects created by 'combine' and 'partition' operations as a result of a query are grouped as subclasses of system defined class named 'CreatedAggregate'. We refer to [Rundensteiner 1989] for set-theoretic operations on classes having "is-a" relationship. For further related work on complex objects we refer to [Ketabchi 1988], [Kim 1988].
DEFINITIONS

Before giving some definitions, we recall that if the value of an attribute \( p \) of an object \( o \) of type \( T \) is an identifier of an object \( o' \) of type \( T' \), i.e., \( T.p.o=o'.oid \), then we say that \( o \) refers to \( o' \). If the value of an attribute of an object is such as integer, string or boolean then the attribute is simple whereas if the value of an attribute of an object is such that \( T.p.o=o'.oid \) and \( o' \) owns \( o' \) then we say that it is a complex attribute. In this case, \( o' \) is a dependent object, i.e., the existence of object \( o' \) depends on the existence of object \( o \). Note that, one object may simply refer to another object but may not own it. We now give some definitions.

Definition 6.2.1: Component-type Hierarchy

We define \( G^T_3 \) as a directed graph on \( T \) as follows: For \( S, T \in T \), a component-type edge joining from \( S \) to \( T \) exists if:

(a) a complex attribute \( p \) of \( S \) is such that domain of \( p \) is type \( T \), i.e., \((p,T) \in A(S)\);

(b) for all objects \( o_t \) of type \( T \), \( \exists \) a unique object \( o_s \) of type \( S \) such that \( o_s \) owns \( o_t \). We say that \( o_t \) is "is-part-of" \( o_s \) or \( o_t \) is a component of \( o_s \) and \( o_s \) is a parent of \( o_t \).

Also, in this case we say that \( T \) is a component-type of \( S \).

Let \( T_o \) be a type in \( T \) such that for any \( T \in T \), \( \exists \) a
component-type edge in $G^T_3$ from $T$ to $T_0$. Define a sub-graph $G^T_3(T_0)$ in $G^T_3$ with root $T_0$ as follows:

$\{T \in T \mid \exists$ a path from $T_0$ to $T$ in $G^T_3\} \cup \{T_0\}$

as the nodes and for any $S,T$ in $G^T_3(T_0)$ an edge joining from $S$ to $T$ exists if and only if it is an edge in $G^T_3$. We define $G^T_3(T_0)$ as a component-type hierarchy with root $T_0$ and an instance of $T_0$ in $G^T_3(T_0)$ as a complex object of type $T_0$. A complex object is connected to its components through complex attributes.

In this chapter, we have assumed that a dependent object is owned by exactly one parent object. We have also assumed that a complex attribute is single-valued.

**Definition 6.2.2: Component-Class Hierarchy**

Let $T_0 \in T$ be a given type such that $G^T_3(T_0)$ is defined. Now, let $C_0 : T_0$ be a class of type $T_0$. We define a component-class hierarchy $G^C_3(C_0)$ as a rooted directed tree as follows:

$\{C \in C \mid C$ is of type $T$ and $T \in G^T_3(T_0)\}$

as the nodes of $G^C_3(C_0)$;

for classes $C_1 : T_1$, $C_2 : T_2$ in $G^C_3(C_0)$ an edge joining from $C_1$ to $C_2$ exists if

(a) $C_1, C_2$ are both in $C^*$ or ($C - C^*$);

(b) $T_1$ is a component of $T_2$;

(c) for all objects $o_j \in C_1$, $\exists$ a unique object $o_i \in C_2$ such that $o_i$ owns $o_j$. 
In this case we say that $C_1$ is a component-class of $C_2$.

In Figure 6.1, we illustrate a component-type hierarchy for 'Microcomputer'. The root type is the type Microcomputer. The attributes installation, chipno, addresslines and datalines are simple attributes. The attributes cpu, memory and iounit are complex attributes. The objects of type Microcomputer are linked to their component objects through these attributes to classes Processor, Storage and Iounit, which in turn, are connected to their component objects. The component-class hierarchy corresponding to types in Figure 6.1 is shown in Figure 6.2a. We have also given some instances of component-type hierarchy in Table 6.1. In this table, we use the notation $T piq$ to uniquely identify an instance $Iq$ of class $T p$.

In this chapter, we refer to the domain of an attribute of a type as $\text{dom}(\text{<type name>.<property name>})$. For example, we refer to the domain of an attribute 'datalines' of type Microcomputer by $\text{dom}(\text{Microcomputer.datalines})$.

We use $P(<\text{type name}>)$, $S(<\text{type name}>)$ and $X(<\text{type name}>)$ to denote the set of all attributes, set of simple attributes and set of complex attributes of a type respectively. For example,

$P(\text{Microcomputer})=$\{installation, chipno, addresslines, datalines, cpu, memory, iounit\}

$S(\text{Microcomputer})=$\{installation, chipno, addresslines, datalines\}

$X(\text{Microcomputer})=$\{cpu, memory, iounit\}
FIG. 6.1: COMPONENT-TYPE HIERARCHY FOR 'MICROCOMPUTER'.
FIG. 6.2(a): COMPONENT-CLASS HIERARCHY. FIG. 6.2(b) RESULT OF PARTITION OPERATION
(see example 6.4.2)
Microcomputer
T011: ('DU', 80286, 20, 16, T1I1, T2I1, T3I1)
T012: ('JNU', 68000, 24, 16, T1I2, T2I2, T3I2)
T013: ('NPL', 8088, 8, 16, T1I3, T2I3, T3I3)
T014: ('NPL', 80386, 32, 32, T1I4, T2I4, T3I4)
T015: ('JNU', 80486, 32, 32, T1I5, T2I5, T3I5)
T016: ('DU', 80386, 32, 32, T1I6, T2I6, T3I6)

Processor
T1I1: (10, 23, 16, T4I1) T1I2: (10, 18, 20, T4I2)
T1I3: (10, 13, 16, T4I3) T1I4: (20, 18, 32, T4I4)
T1I5: (25, 24, 64, T4I5) T1I6: (20, 18, 32, T4I6)

Storage
T2I1: (4, 20, T5I1, T6I1) T2I2: (4, 20, T5I2, T6I2)
T2I2: (1, 20, T5I3, T6I3) T2I4: (16, 20, T5I4, T6I4)
T2I5: (256, 20, T5I5, T6I5) T2I6: (16, 20, T5I6, T6I6)

Iounit
T3I1: (2, 2, 96, 20, 'CGA', T7I1, T8I1, T9I1)
T3I2: (2, 2, 64, 24, 'VGA', T7I2, T8I2, T9I2)
T3I3: (2, 2, 62, 8, 'CGA', T7I3, T8I3, T9I3)
T3I4: (2, 2, 96, 32, 'VGA', T7I4, T8I4, T9I4)
T3I5: (2, 2, 96, 32, 'VGA', T7I5, T8I5, T9I5)
T3I6: (2, 2, 128, 32, 'MON', T7I6, T8I6, T9I6)

CU
T4I4: ('MxMn', 2, 6) T4I5: ('MxMn', 2, 8) T4I6: ('MxMn', 2, 6)

PrMemory
T5I1: (640, 'CMOS') T5I2: (1024, 'MOS') T5I3: (256, 'MOS')
T5I4: (1024, 'CMOS') T5I5: (4096, 'CMOS') T5I6: (1024, 'CMOS')

ExEpMemory
T6I1: (0.625, 'MOS') T6I2: (3, 'MOS') T6I3: (0.25, 'MOS')
T6I4: (15, 'MOS') T6I5: (252, 'CMOS') T6I6: (15, 'MOS')

HDDrive
T7I1: (20, 'MAG') T7I2: (40, 'MAG') T7I3: (20, 'MAG')
T7I4: (80, 'OPT') T7I5: (80, 'OPT') T7I6: (240, 'OPT')

FDDrive
T8I1: (360, 'MAG') T8I2: (360, 'MAG') T8I3: (1.2, 'OPT')
T8I4: (1.2, 'OPT') T8I5: (1.2, 'OPT') T8I6: (760, 'MAG')

Printer

Table 6.1: Instances of types shown in Figure 6.1.
6.3 OPERATIONS ON COMPLEX OBJECTS

In this section, we define operations on complex objects. Each operation is defined on the root of one or two component-class hierarchies and results again in a component-class hierarchy.

6.3.1 UNION Operation

Let $G^C_3(C_1)$ and $G^C_3(C_2)$ be two component-class hierarchies with root classes $C_1$ and $C_2$ respectively where $C_1 = C_1(I_1, D_1) : T$ and $C_2 = C_2(I_2, D_2) : T$ are of same type $T$. The result of the union of two component-class hierarchies with root classes $C_1$ and $C_2$ is again a component class hierarchy with root class $C$ that contains all objects belonging either to $C_1$ or $C_2$.

The general form of union operation is:

$$ C = C_1 \cup^C C_2 $$

where $\cup^C$ denotes the UNION operator and we define the new component-class hierarchy $C = C(I, D) : T$ as follows:

$I \cup D = \{ i | i \in C_1 \text{ or } i \in C_2 \}$;

$I = I_1 \cup I_2$;

$D = D_1 \cup D_2$;

$C$ is a subclass of the unique class corresponding to the type $T$.

Let $C_0 = C_1$, $C_2 = C_2$, $C = C$ and $T_0 = T$.

For any integer $d > 0$, let $(C_1^d)^m_{j=1}$ and $(C_2^d)^m_{j=1}$ be the set of all classes of distance $d'$ in $G^C_3(C_1)$ and $G^C_3(C_2)$ from $C_1$ and $C_2$ respectively. Then for each $C_1^d$, there
exists a sequence of classes $C_{1j}$ of type $T_{1j}$ and $C_{1j}$ is of distance 'i' in $G_{3}^{C}(C_{1})$ from $C_{10}$ such that

(a) $C_{10} = C_{10}$

(b) $C_{1j}$ is a component-class of $C_{1j-1}$, $\forall i = 1,2,\ldots,d$

Similarly, for each $C_{2j}$, there exists a sequence of classes $C_{2j}$ of type $T_{2j}$ and $C_{2j}$ is of distance 'i' in $G_{3}^{C}(C_{2})$ from $C_{20}$ such that

(a) $C_{20} = C_{20}$

(b) $C_{2j}$ is a component-class of $C_{2j-1}$, $\forall i = 1,2,\ldots,d$

Now, for each $j=1,2,\ldots,m$, define a class $C_{d} = C_{d}(I_{d},D_{d})$ of type $T_{d}$ which is of distance 'd' from $C$ in $G_{3}^{C}(C)$ as follows:

$I_{d} = I_{d}^{1} U I_{d}^{2}$

$D_{d} = D_{d}^{1} U D_{d}^{2}$

$C_{d}$ is a subclass of the unique class corresponding to type $T_{d}$

$C_{d}$ is component-class of $C_{d-1}$

Remark 6.3.1: The set-theoretic operations are based on object identity.

Remark 6.3.2: Since the union operation is based on object identity and therefore equal objects are not eliminated.
6.3.2 INTERSECTION Operation

Let $G^C_3(C_1)$ and $G^C_3(C_2)$ be two component-class hierarchies with root classes $C_1$ and $C_2$ respectively where $C_1=C_1(I_1,D_1):T$ and $C_2=C_2(I_2,D_2):T$ are of same type $T$. The result of the intersection of two component-class hierarchies with root classes $C_1$ and $C_2$ is again a component class hierarchy with root class $C$ that contains all objects belonging to both $C_1$ and $C_2$.

The general form of union operation is:

$$C = C_1 \cap^C C_2$$

where $\cap^C$ denotes the INTERSECTION operator and we define the new component-class hierarchy $C=C(I,D):T$ as follows:

$I \cup D = \{ i \mid i \in C_1 \text{ and } i \in C_2 \}$;

$I = I_1 \cap I_2$;

$D = D_1 \cap D_2$;

$C$ is a subclass of the unique class corresponding to type $T$.

Let $C_0=C_1$, $C_2=C_2$, $C_0=C$ and $T_0=T$.

For any integer $d>0$, let $\{C_1^j\}_{j=1}^m$ and $\{C_2^j\}_{j=1}^m$ be classes as defined in section 6.3.1.

Then, for each $j=1,2,\ldots,m$, we define a class $C_d^j = C_d^j(I_d^j,D_d^j):T_d^j$ of type $T_d^j$ which is of distance 'd' from $C$ in $G^C_3(C)$ as follows:

$I_d^j \cup D_d^j = \{ i \mid i \in C_1^j \text{ and } i \in C_2^j \}$;

$I_d^j = I_1^j \cap I_2^j$;

$D_d^j = D_1^j \cap D_2^j$;

$C_d^j$ is a subclass of the unique class corresponding to type
6.3.3 DIFFERENCE Operation

Let $G^c_3(C_1)$ and $G^c_3(C_2)$ be two component-class hierarchies with root classes $C_1$ and $C_2$ respectively where $C_1=Cl(I_1,D_1):T$ and $C_2=C_2(I_2,D_2):T$ are of same type $T$. The result of the difference of two component-class hierarchies with root classes $C_1$ and $C_2$ is again a component class hierarchy with root class $C$ that contains all objects belonging to $C_1$ but not in $C_2$.

The general form of difference operation is:

$$C = C_1 - C_2$$

where $-$ denotes the DIFFERENCE operator and we define the new component-class hierarchy $C=C(I,D):T$ as follows:

$I \cup D = \{ i | i \in C_1 \text{ and } i \notin C_2 \} ;$

$I = I_1 - I_2 ;$

$D = D_1 - D_2 ;$

$C$ is a subclass of the unique class corresponding to type $T$.

Let $C_1_0=C_1$, $C_2_0=C_2$, $C_0=C$ and $T_0=T$.

For any integer $d>0$, let $\{C_1^d_j\}_{j=1}^m$ and $\{C_2^d_j\}_{j=1}^m$ be classes as defined in section 6.3.1.

Then, for each $j=1,2,\ldots,m$, we define a class $C_d^j = C_d^j(I_d^j,D_d^j):T_d^j$ of type $T_d^j$ which is of distance 'd' from $C$ in $G_3^c(C)$ as follows:

$I_d^j \cup D_d^j = \{ i | i \in C_1^j \text{ and } i \notin C_2^j \} ;$
\[ I_d^j = I_{1d}^j - I_{2d}^j ; \]
\[ D_d^j = D_{1d}^j - D_{2d}^j ; \]
\[ C_d^j \text{ is a subclass of the unique class corresponding to type } T_d^j ; \]
\[ C_d^j \text{ is a component-class of } C_{d-1}^j . \]

### 6.3.4 CARTESIAN PRODUCT

Let \( G_3^c(C_1) \) and \( G_3^c(C_2) \) be two component-class hierarchies with root classes \( C_1 \) and \( C_2 \) respectively where \( C_1 = C_1(I_1,D_1) : T_1 \) and \( C_2 = C_2(I_2,D_2) : T_2 \). The result of the cartesian product of two component-class hierarchies with root classes \( C_1 \) and \( C_2 \) is again a component class hierarchy with root class \( C = C(I,D) : T \) of type \( T \). The attributes of \( T \) are the sum of the attributes of types \( T_1 \) and \( T_2 \), and the members of \( C \) are obtained by combining each member of \( C_1 \) with all the members of \( C_2 \).

The general form of cartesian product is:

\[ C = C_1 \times^c C_2 \]

where \( \times^c \) denotes the CARTESIAN PRODUCT and we define the new component-class hierarchy \( C = C(I,D) : T \) as follows:

If \( T_1 = \{(a_i,A_i)\}_{i=1}^n \) and \( T_2 = \{(b_j,B_j)\}_{j=1}^m \) then

\[ T = \{(f_k,F_k)\}_{k=1}^{n+m} \]

and \( T \) is a subtype of the root type \( T^0 \) such that for \( k=1,2,\ldots,n; \ f_k = a_k, \text{ and } F_k = A_k \) and with \( k=n+j, \ f_k = b_j \text{ and } F_k = B_j \) for \( j=1,\ldots,m; \)

\( T_1 \) and \( T_2 \) are component-types of \( T \).
I U D = \{ i | i = (i_1, i_2), i_1 \in C_1, i_2 \in C_2 \}
\forall (f_k, F_k) \in A(T),
T.f_k.i = \begin{cases} 
T_1.f_k.i_1 & \text{if } f_k \in S(T_1) \\
T_2.f_k.i_2 & \text{if } f_k \in S(T_2),
\end{cases}
I = I U D;
D = \emptyset.

C is the unique class corresponding to type T and is a subclass of the root class \( C^0 \).
Let \( C_0 = C_1, C_2 = C_2, C_0 = C \) and \( T_0 = T \).
For any integer \( d > 0 \), let \( \{C^{1d}_{k}\}_{k=1}^{n_d} \) and \( \{C^{2d}_{j}\}_{j=1}^{m_d} \) be all the classes of distance \( d \) in \( G^C_1(C_1) \) and \( G^C_2(C_2) \) from \( C_1 \) and \( C_2 \) respectively as defined in section 6.3.1, where
\( C^{1d}_{d}(I^{1d}_{d}, D^{1d}_{d}) : T^{1d}_{d} \) and \( C^{2d}_{d}(I^{2d}_{d}, D^{2d}_{d}) : T^{2d}_{d} \). For each \( T^{1d}_{d} \) (or \( T^{2d}_{d} \)), \exists \ a type \( T^{1d}_{d-1} \) (or \( T^{2d}_{d-1} \)) such that \( T^{1d}_{d} \) is a component-type of \( T^{1d}_{d-1} \) (or \( T^{2d}_{d-1} \)).
We define \( C^{d}_{d} = C^{d}_{d}(I^{d}_{d}, D^{d}_{d}) : T^{d}_{d} \) as follows:
\( T^{d}_{d} = \begin{cases} 
T^{1d}_{d} & \text{if } l = k = 1, 2, \ldots, n_d \\
T^{2d}_{d} & \text{if } l = n_d + j, j = 1, 2, \ldots, m_d
\end{cases} \)
For \( l = k = 1, 2, \ldots, n_d \);
\( I^{d}_{d} = \{ i | i \in I^{1d}_{d} \} \)
\( D^{d}_{d} = \{ i | i \in D^{1d}_{d} \} \)
\( \forall q \in S(T^{d}_{d}), T^{d}_{d}.q.i = T^{1d}_{d}.q.i \) and
for \( l = n_d + j \) and \( j = 1, 2, \ldots, m_d \);
\( I^{d}_{d} = \{ i | i \in I^{2d}_{d} \} \)
\( D^{d}_{d} = \{ i | i \in D^{2d}_{d} \} \)
\( \forall q \in S(T^{d}_{d}), T^{d}_{d}.q.i = T^{2d}_{d}.q.i \)
For each \( T^{d}_{d} \), \( \exists \ a type T^{d-1}_{d} \) and a property \( p \in X(T^{d-1}_{d}) \) such
that \( \text{dom}(T_{d-1}^l.p) = T_d^l \) and \( T_d^l \) is a component-type of \( T_{d-1}^l \).

\( \forall i \in C_d^l, \exists i' \in C_{d-1}^l \) such that

\( T_{d-1}^l.p.i' = i.\text{oid} \); \( i' \) owns \( i \).

i.e., \( C_d^l \) is a component-class of \( C_{d-1}^l \).

Remark 6.3.3: The cartesian product creates new objects. However, members of all the classes except the root class of the new component-class hierarchy have objects which are equal to the members of the corresponding given class but not identical.

Remark 6.3.4: The cartesian product is applied only on the root classes.

Remark 6.3.5: In the above definition the number of component classes of the new class at \( i \)th level \((i>0)\) is \( n_d + m_d \).

6.3.5 CHOOSE Operation

The choose operation defined on a component-class hierarchy is again a component-class hierarchy whose members are subset of members of the given class, satisfying the choose condition.

Let \( C = C(I,D):T \) be the root of the given component-class hierarchy \( G^c_3(C) \). Let \( M = \{ C_i \}_{i=1}^n \) be a main branch of \( C \) in \( G^c_3(C) \) where \( C_i = C_i(I_i,D_i):T_i \) for \( i=0,1,\ldots,n \) and \( C = C_0 \).

The general form of choose operation is

\[ C' = \sigma^c_B(C) \]

where \( \sigma^c \) denotes the CHOOSE operator and \( B \) is a boolean
expression defined as in section 5.4.5. We define the new class C' as:

\[ I' \cup D' = \left\{ i_0 \in C_0 | \exists \text{ a set of objects } \{ i_1, i_2, \ldots, i_n \}, \right. \]
\[ i_j \in C_j, \quad j=1,2,\ldots,n \text{ such that} \]
\[ (a) \text{ Boolean expression } B \text{ is true.} \]
\[ (b) (i_1.oid=T_0.p_0.i_0)\text{AND}(i_2.oid=T_1.p_1.i_1)\text{AND} \]
\[ \ldots \text{AND}(i_n.oid=T_{n-1}.p_{n-1}.i_{n-1}) \text{ is true; and} \]
\[ i_0 \text{ owns } i_1; \quad i_1 \text{ owns } i_2; \ldots; \quad i_{n-1} \text{ owns } i_n. \]

\[ I' = (I' \cup D') \cap I_r; \]
\[ D' = \text{Complement of } I' \text{ in } (I' \cup D'); \]

\[ C' = C'(I', D'): T \text{ is a subclass of } R = R(I_r, D_r): T \text{ where } R \text{ is the unique class corresponding to type } T. \]

Let \( E_0' = C' \).

For \( d > 0 \), let \( \{ E^j_d \}_{j=1}^n \) be the set of all classes of distance \( d \) from the root class \( C_0 \) in \( C_0^G(C) \).

For each \( T^j_d \), a type \( T^j_{d-1} \) and a property \( q^j \in X(T^j_{d-1}) \) exist such that \( \text{dom}(T^j_{d-1}.q^j) = T^j_d \) and \( T^j_d \) is a component-type of \( T^j_{d-1} \).

Define \( E^j_d(I^j_d, D^j_d): T^j_d \) as:

\[ I^j_d \cup D^j_d = \left\{ i \in E^j_d | \exists i_0 \in E^j_{d-1} \text{ such that} \right. \]
\[ i.oid=T^j_d . q^j . i_0 \text{ and} \]
\[ i_0 \text{ owns } i. \]

\[ I^j_d = (I^j_d \cup D^j_d) \cap I_r^j; \]
\[ D^j_d = \text{Complement of } I^j_d \text{ in } (I^j_d \cup D^j_d); \] and

\( E^j_d \) is a subclass of the unique class \( R^j_d = R^j_d(I_r^j, D_r^j): T^j_d \) corresponding to type \( T^j_d \);

\( E^j_d \) is a component-class of \( E^j_{d-1} \).

Remark 6.3.6: If \( i \in C' \) then \( i \in C \), i.e., new objects are
not created.

Remark 6.3.7: If the choose condition involves two or more main branches of C connected by logical operations then the result hierarchy can be obtained by applying the choose conditions on each of the branches separately and finally applying appropriate set-theoretic operations.

6.3.6 PARTITION Operation

The partition operation defined on a component-class hierarchy is again a component-class hierarchy where the attributes of the type corresponding to the root class is subset of the attributes corresponding to the root of the given component class hierarchy.

Let C=C(I,D):T be the root of the given component-class hierarchy. The general form of PARTITION operation is:

\[ C' = \pi^C_{A(I)}(C) \]

where \( \pi^C \) denotes the PARTITION operator and \( A(I) \) is the attribute list. We define the new component class hierarchy with root class \( C'(I',D'):T' \) as follows:

Let \( P_o = \{q_o(i)\}_{i=1}^k \) be the properties of T to be projected.
Define \( P(T') = \{q_o(i)\}_{i=1}^k \)

T' is a subtype of the root type T'.

\( I' \cup D' = \{ i \mid i \in C \} \)

\( \forall p \in P_o, T'.p.i = T.p.i \)

C' is the unique class corresponding to type T' and is a subclass of C'.

Let \( E'_o = C'_o \).
For $d > 0$, let $\{E^j_d\}^m_{j=1}$ be the set of all classes of distance $d$ from the root class $C$.

For each $T^j_d$, $j=0,1,\ldots,m$; we assume that $\exists$ a type $T^j_{d-1}$ and a property $p' \in X(T^j_{d-1})$ such that $\text{dom}(T^j_{d-1}.p') = T^j_d$.

Now, for each $j=0,1,\ldots,m$; let $P^j_d = \{q^j_d(i)\}^p_{i=1}$ be the properties of $T^j_d$ to be projected.

We define $E^j_j = \mathbb{P}^j_d (I^j_d, D^j_d) : T^j_d$ as:

\[
P(T^j_d) = P^j_d
\]

\[
\text{dom}(T^j_{d-1}.p') = T^j_d
\]

$T^j_d$ is a subtype of the root type $T^0$.

\[
I^j_d \cup D^j_d = \{ i \mid i \in E^j_d \}
\]

\[
I^j_d = I^j_d \cup D^j_d
\]

\[
D^j_d = \emptyset
\]

$\forall p \in S(T^j_d)$, $\forall i' \in E^j_d$, $T^j_d.p.i' = T^j_d.p'i$ and $\forall i \in E^j_d$, $T^j_d.p'.i = i'.\text{oid}$ for some $i' \in E^j_d$.

$C^j_d$ is the unique class corresponding to type $T^j_d$ and is a subclass of $C^0$.

Remark 6.3.8: Partition operation creates new objects.

Remark 6.3.9: If a complex attribute of an object does not belong to the set of attributes to be projected, then the branch beginning with the domain of this complex attribute is not included in the partitioned object.

The operations defined in Section 6.3 are extensions of the relational and set-theoretic operations on relations. This is evident from the fact that the operations defined on complex object hierarchies reduce to equivalent operations on relations, if the root class of
the complex object hierarchy has simple attributes only.

6.4 EXAMPLES OF QUERIES

In this section we give some examples of queries to illustrate the use of the operations defined above.

Example 6.4.1: Consider the following query:

'Choose all Microcomputers installed at "JNU" with chipno "80386" having "ECA" videoadaptor and a line printer with speed > 600.'

Microcomputer-JNU = σ^C_B (Microcomputer)

where the condition B is:

Microcomputer.installation="JNU" AND
Microcomputer.chipno="80386" AND
Microcomputer.iounit.videoadaptor="ECA" AND
Microcomputer.iounit.printer.speed>600.

Example 6.4.2: Consider the following query:

'Choose all Microcomputers installed at "JNU" and having chipno "80386". Extract attributes installation, chipno and iounit of class Microcomputer, attributes videoadaptor, hdd and fdd of class Iounit, and attribute capacity for classes HDDrive and FDDrive.

Step 1:

Microcomputer-JNU-80386 = σ^C_B (Microcomputer)

where the condition B is:

Microcomputer.installation="JNU" AND
Microcomputer.chipno="80386"

Step 2:

Microcomputer' = π^C_{A(l)} (Microcomputer-JNU-80386)

where
A(1) = Microcomputer(installation, chipno, iounit),
      Iounit(videoadaptor, hdd, fdd),
      HDDrive(capacity),
      PDDrive(capacity).

The complex object hierarchy tree obtained as a result of applying this operation on the tree given in Figure 6.2a is shown in Figure 6.2b.

Example 6.4.3: Let there be two types, Microprocessor and Peripheral with properties as follows:

P(Microprocessor) = {mpno, installation, chipno, addresslines,
                        datalines, cpu, memory}

and

P(Peripheral) = {pno, installation, iounit}

Consider the following query:

'List all Microprocessors installed at "JNU", along with their peripherals such that Microprocessor.mpno is equal to Peripheral.pno.'

Step 1:

Microprocessor-JNU = \sigma^C_B (Microprocessor)

where the condition B is

Microprocessor.installation = "JNU"

Step 2:

Temp = Microprocessor-JNU \times_C Peripheral

Step 3:

Microprocessor' = \sigma^C_B (Temp)

where the condition B is

Microprocessor.mpno = Peripheral.pno

The complex object hierarchy tree obtained as result of applying this operation on the given classes is shown in
Figure 6.3.

**Example 6.4.4:** Consider the following query:
'Choose all Microcomputers that are either installed at "JNU" having "ECA" videoadaptor and line printer with speed equal to 600 or installed at "NPL" with memory greater than 4MB.

Step 1:
\[ \text{Microcomputer-JNU} = \sigma_B^c(\text{Microcomputer}) \]
where the condition B is
\[
\begin{align*}
\text{Microcomputer}.\text{installation} &= "JNU" \text{ AND} \\
\text{Microcomputer}.\text{iounit}.\text{videoadaptor} &= "ECA" \text{ AND} \\
\text{Microcomputer}.\text{iounit}.\text{printer}.\text{speed} &= 600.
\end{align*}
\]

Step 2:
\[ \text{Microcomputer-NPL} = \sigma_B^c(\text{Microcomputer}) \]
where the condition B is
\[
\begin{align*}
\text{Microcomputer}.\text{installation} &= "NPL" \text{ AND} \\
\text{Microcomputer}.\text{memory}.\text{size} &> 4.
\end{align*}
\]

Step 3:
\[ \text{Microcomputer-JNU-NPL} = \text{Microcomputer-JNU} \cup^c \text{Microcomputer-NPL} \]

**Example 6.4.5:** Consider the following query:
'Choose all Microcomputers that are installed at "JNU" having memory greater than 4MB and all microcomputers with chipno "80386" having "ECA" videoadaptor.

Step 1:
\[ \text{Microcomputer-JNU} = \sigma_B^c(\text{Microcomputer}) \]
where the condition B is:
FIG. 6.3: RESULT OF CARTESIAN PRODUCT. (see example 6.4.3)
Microcomputer.installation = "JNU" AND
Microcomputer.memory.size > 4.

Step 2:

Microcomputer-80386 = $\sigma^c_B$ (Microcomputer)

where the condition $B$ is:

Microcomputer.chipno = "80386" AND
Microcomputer.iounit.videoadaptor = "ECA"

Step 3:

Microcomputer-JNU-NPL =

Microcomputer-JNU $\cap^c$ Microcomputer-NPL

**Example 6.5.6:** Consider the following query:

'Choose all microcomputers with chipno "80386" with "VGA" videoadaptor but not installed at "JNU" having memory less than 4MB.

Step 1:

Microcomputer-80386 = $\sigma^c_B$ (Microcomputer)

where the condition $B$ is:

Microcomputer.chipno = "80386" AND
Microcomputer.iounit.videoadaptor = "VGA"

Step 2:

Microcomputer-JNU = $\sigma^c_B$ (Microcomputer)

where the condition $B$ is:

Microcomputer.installation = "JNU" AND
Microcomputer.memory.size > 4.

Step 3:

Microcomputer-80386-not-JNU =

Microcomputer-80386 $\cap^c$ Microcomputer-JNU