CHAPTER 5

A FORMAL MODEL FOR OBJECT-ORIENTED DATABASES

5.1 INTRODUCTION

The deficiencies of the relational model for supporting advanced applications are mentioned in Chapter 1. However unlike relational databases [Maier 1983] which have theoretical foundation, a universally agreed upon model to support object-oriented databases has not yet emerged.

Types and classes are the two primary concepts in object-oriented data models [Kim 1990], [Nierstrasz 1989]. Some models have only types [Fishman 1987], [Manola 1986] and some have only classes [Banerjee 1987], [Bretl 1989], [Weiser 1989]. For example, data model of IRIS [Fishman 1987] has only types, and it supports functions that take a type as argument and return the instances of the type. Also, when a new object of a type is created, it is added to the set of instances of that type and also to the set of instances of all its supertypes. However, it maintains no relationships between these sets.

In systems which have only classes [Banerjee 1987], [Bretl 1989], [Weiser 1989], a class serves two purposes. First, it defines the structure and behaviour of objects belonging to it which means the type. Second, it represents a collection of objects having the same type.
That is, in these systems, the concepts of types and collection of objects are overloaded in the definition of a class. This approach creates confusion and complicates the model. Also, it is not possible to have classes which share the same type. Further, it is not easy to define operations on classes that create new classes. In ORION [Banerjee 1988], [Bertino 1989], for example, the result of a query has to be an instance of an existing class and must return all or only one attribute of that class. Thus, it disallows operations which create new classes similar to relational project, join and set-theoretic operations.

In our model [Bhalla 1991b], we make a clear distinction between type and collection. This approach gives rise to clear semantics for defining them. Also, it enables us to separate the two meanings of 'is-a' relationship by distinguishing subtype/supertype and subclass/superclass relationships. Further, it is possible to have classes that share the same type. Also, with this approach it is easier to define operations on classes and create new classes along the line of relational databases. Operations such as cartesian product and partition (corresponding to relational project) creates new types in addition to new classes. Since types and classes cannot exist in isolation, we also identify the position of the new classes and types in the respective hierarchy. Another advantage of this approach
is to have queries against multiple target classes and operations like relational project and set-theoretic operations which are difficult in models which support only types or classes.

5.2 OBJECT-ORIENTED SCHEMA

This section presents a formal framework for object-oriented databases. We begin by defining types and classes and then analyse the relationships between them.

Definition 5.2.1: A type describes the structure (attributes) and operations that are used to access the object.

Let T be a type and A(T) be its attributes. We denote A(T) as \( \{(a_i, T_i)\}_{i=1}^{n} \) where \( T_i \) is the domain of the attribute \( a_i \). Let M(T) be the operations defined on T. We denote M(T) as \( \{m_1, m_2, \ldots, m_j\} \) and the type T as \((A(T), M(T))\).

For example, we define a type CIRCLE as:

\[
\text{CIRCLE} = (A(\text{CIRCLE}), M(\text{CIRCLE}))
\]

where \( A(\text{CIRCLE}) = \{(\text{centre}, \text{POINT}), (\text{radius}, \text{Real})\}\) and \( M(\text{CIRCLE}) = \{\text{perimeter, area}\} \).

A primitive type has no attribute but has a value only. For example, integer, real, string are primitive types.

Definition 5.2.2: Type-attribute graph. Let \( T \) be a set of types. Since the domain of an attribute of a type is again a type, this leads to a type-attribute graph. We
define this graph $G_1^*$ as a directed graph on $T$ as follows. For $S, T \in T$, with $A(S) = \{(a_i, S_i)\}_{i=1}^n$, there is an edge joining from $S$ to $T$ in $G_1^*$ if $T = S_i$ for some $i$. Let $T_0$ be a given type in $T$. We define a subgraph $G_1^*(T_0)$ of $G_1^*$ with root $T_0$ by:

$$\{ S \in T \mid \exists \text{ a path from } T_0 \text{ to } S \text{ in } G_1^* \} \cup \{ T_0 \}$$

as the nodes of $G_1^*(T_0)$ and for $S, T \in G_1^*(T_0)$ $\exists$ an edge joining from $S$ to $T$ in $G_1^*(T_0)$ if it an edge in $G_1^*$. Figure 5.1 is an example of type-attribute graph. In this figure, lines point to the domain type of the attribute. Also, single and double arrows indicate single and multi-valued attributes respectively.

Remark 5.2.1: $G_1^*$ can be a cyclic graph (Figure 5.2).

Remark 5.2.2: The leaf nodes in type-attribute graph will be the primitive types.

Definition 5.2.3: Primitive types of a type $T_0$. We define the leaf nodes of the graph $G_1^*(T_0)$ as the primitive types of the given type $T_0$.

Definition 5.2.4: Subtype. Let $S_0 = \{A(S_0), M(S_0)\}$ and $T_0 = \{A(T_0), M(T_0)\}$ be two given types. Let $P(S_0)$ and $P(T_0)$ be the set of all primitive types of $S_0$ and $T_0$ respectively. We say that $S_0$ is a subtype of $T_0$ if for any $p \in P(T_0)$ and for all sequence of attributes $\{(a_i, T_i)\}_{i=1}^n$ with $(a_i, T_i) \in A(T_{i-1}) \forall i = 1, \ldots, n$ and $p = T_n$, $\exists q \in P(S_0)$ and a sequence of attributes $\{(a_i, S_i)\}_{i=1}^n$ with $(a_i, S_i) \in (A(S_{i-1}) \forall i = 1, 2, \ldots, n$ and $q = S_n$ such that $\text{dom}(S_n) \subseteq \text{dom}(T_n)$.
FIG. 5.1: TYPE-ATTRIBUTE GRAPH.

FIG. 5.2: TYPE-ATTRIBUTE GRAPH WITH CYCLE.
Also in this case we say that $T_0$ is a supertype of $S_0$.

Figure 5.3 shows an example of subtype relationship. In this figure, science student SC-STUD is a subtype of STUDENT. Similarly, SC-DEPT and SC-PROJ are subtypes of DEPARTMENT and PROJECT respectively.

**Remark 5.2.3:** Let $S_0$ and $T_0$ be primitive types. Then $S_0$ is a subtype of $T_0$ if $\text{dom}(S_0) \subseteq \text{dom}(T_0)$.

**Definition 5.2.5:** **Equality of types.** If $S$ is a subtype of $T$ and $T$ is a subtype of $S$, then we say that $S$ is equal to $T$ and denote by $S=T$.

**Definition 5.2.6:** **Proper subtype.** We say that $S$ is a proper subtype of $T$ if $S$ is a subtype of $T$ and $S \neq T$.

**Theorem 5.2.1:** 'Subtype-of' is a partial order relation on the set of types.

**Proof:** The relation is obviously reflexive and anti-symmetric. To show that 'subtype-of' is a transitive relation, we consider $R_0, S_0$ and $T_0$ be types in $T$ such that $R_0$ is a subtype of $S_0$ and $S_0$ is a subtype of $T_0$.

For any $p \in P(T_0)$ and for all sequence of attributes $\{(a_i, T_i)\}_{i=1}^{n}$ with $(a_i, T_i) \in A(T_{i-1})$, for $i=1,2,\ldots,n$ and $p=T_n$, \exists q \in P(S_0) and sequences of attributes $(a_i, S_i) \in A(S_{i-1})$ for $i=1,2,\ldots,n$ and $q=S_n$ such that $\text{dom}(S_n) \subseteq \text{dom}(T_n)$ and $M(S_i) \subseteq M(T_i)$ for $i=0,1,\ldots,n$. 

\[ M(S_i) \subseteq M(T_i) \quad i=0,1,\ldots,n \]
FIG. 5.3: GRAPHICAL REPRESENTATION OF SUBTYPE.
Now, for \( q \in P(S_0) \), \( \{(a_i, S_i)\}_{i=1}^{n} \), and since \( R_0 \) is a subtype of \( S_0 \), \( \exists r \in P(R_0) \) and \( \{(a_i, R_i)\}_{i=1}^{n} \) such that

(i) \( \forall i=1, 2, \ldots, n, \ (a_i, R_i) \in A(R_{i-1}) \)

(ii) \( r=R_n \) with \( \text{dom}(R_n) \subseteq \text{dom}(S_n) \) and \( M(R_i) \subseteq M(S_i) \) for \( i=0, 1, \ldots, n; \)

for \( p \in P(T_0) \) and \( \{(a_i, T_i)\}_{i=1}^{n} \) with \( (a_i, T_i) \in A(T_{i-1}) \), for \( i=1, 2, \ldots, n \) and \( p=T_n, \exists r \in P(R_0) \) and \( \{(a_i, R_i)\}_{i=1}^{n} \) with \( (a_i, R_i) \in A(R_{i-1}) \) for \( i=1, 2, \ldots, n \) and \( r=R_n \) such that

\( \text{dom}(R_n) \subseteq \text{dom}(S_n) \subseteq \text{dom}(T_n) \) and

for \( i=0, 1, 2, \ldots, n \) \( M(R_i) \subseteq M(S_i) \subseteq M(T_i) \)

\( \Rightarrow R_0 \) is a subtype of \( T_0 \).

Since 'subtype-of' relation is a partial order, the set of types \( T \) with 'subtype-of' relation will form a directed acyclic graph which we denote as \( G^T_2 \). Throughout this chapter we assume that this directed acyclic graph is a rooted graph and we denote the root as \( T^0 \). Since a subtype inherits all attributes and operations from its supertype, type hierarchy is also called inheritance hierarchy.

**Definition 5.2.7: Class.** A class is a set of objects of a given type. We denote a class as

\[
\text{Class} \ <\text{classname}>:\text{<typename>}
\]

The members of a class may have different types such that they are related by subtype relation. More specifically, the members of a class of type \( T \) may be instances of type \( T \) or subtypes of \( T \).
FIG. 5.4: MULTIVALEUED FUNCTION FROM TYPES TO CLASSES.

FIG. 5.5: ONE-ONE AND ONTO FUNCTION FROM TYPES TO UNIQUE CLASSES.
We split the members of a class \(C:T\) into two disjoint sets, \(I\) and \(D\), such that \(I\) is the set of instances of type \(T\) and \(D\) is such that \((I \cup D)\) is the set of members of type \(T\). That is, for a class of type \(T\) two sets are associated, namely: the set \(I\) of instances of type \(T\) and set \(D\), the union of instances of subtypes of \(T\).

**Definition 5.2.8: Association between types and classes.**
Let \(c\) be a set of classes whose types are elements of \(T\). We assume that for each type \(T \in T\), there exists at least one class \(C \in c\) such that the members of \(C\) are of type \(T\). In fact, this association between types and classes is multi-valued (Figure 5.4).

\[T \rightarrow f(T) = \text{set of classes of type } T.\]

Moreover, we assume that for each type \(T \in T\), there exists a unique class \(C = C(I,D)\) whose members are of type \(T\) such that if \(C' = C'(I',D'):T\) is any class whose members are also of type \(T\) implies

\[I' \subseteq I \quad \text{and} \quad D' \subseteq D \quad \text{holds.}\]

Now, consider \(c^*\) a subset of \(c\) whose members are these \(C\)'s for each type \(T \in T\). Then the above association between types and classes is one-one and onto function (Figure 5.5).

**Definition 5.2.9: Equality of Classes.** Let \(C_1 = (I_1,D_1):T\) and \(C_2 = (I_2,D_2):T\) be two classes of the same type. We define \(C_1\) is equal to \(C_2\) if \(I_1 = I_2\) and \(D_1 = D_2\).
Theorem 5.2.2: The relation 'is equal to' is an equivalence relation on C.

Remark 5.2.4: Throughout in our discussion, a class C always means the equivalence class determined by C.

Definition 5.2.10: Subclass. We define the 'subclass' relation S on C by the following:

\[ S = \left\{ (C_1, C_2) \in C \mid \text{for } C_1(I_1, D_1):T_1 \text{, } C_2(I_2, D_2):T_2 \text{, } \right. \]
\[ \text{and } T_1 \text{ is a subtype of } T_2, \]
\[ I_1, D_1 \subseteq D_2 \text{ when } T_1 \text{ is a proper subtype of } T_2 ; \]
\[ I_1 \subseteq I_2 \text{ and } D_1 \subseteq D_2, \text{ when } T_1 = T_2. \]

where \( C = (c \times c^*) \cup V \subseteq c \times c \);
\[ V = \{ (C, C) \mid C \in c \} . \]

By definition if \((C_1, C_2) \in S\) then \(C_1\) is a subclass of \(C_2\).

Remark 5.2.5: \(V\) is a subset of \(S\), i.e., \((C, C) \in S\) which implies any class \(C\) in \(C\) is always a subclass of itself.

Theorem 5.2.3: The 'Subclass' relation on \(C\) is partial order.

Proof: The relation is obviously reflexive (see remark 5.2.5).

Let \(C_1 = (I_1, D_1):T_1\) and \(C_2 = (I_2, D_2):T_2\) be two classes such that \(C_1\) is a subclass of \(C_2\) and \(C_2\) is also a subclass of \(C_1\).

By the definition of subclass, \(T_1\) is a subtype of \(T_2\) and
T₂ is a subtype of T₁ and hence T₁ = T₂. Again, since T₁ = T₂ and by the assumption, I₁ ⊆ I₂, D₁ ⊆ D₂ and I₂ ⊆ I₁, D₂ ⊆ D₁, are true, i.e., I₁ = I₂, D₁ = D₂ and thus the classes C₁ and C₂ are equal. Therefore, subclass relation is anti-symmetric.

Let C₁ = (I₁, D₁): T₁, C₂ = (I₂, D₂): T₂ and C₃ = (I₃, D₃): T₃ be classes such that C₁ is a subclass of C₂ and C₂ is a subclass of C₃.

Then from the definition of subclass, T₁ is a subtype of T₂ and T₂ is a subtype of T₃. But since subtype relation is a partial order implies T₁ is a subtype of T₃.

**Case 1:** All the types are different, i.e., T₁ ≠ T₂ ≠ T₃.

From the definition of subclass, T₁ is a subtype of T₂ and T₂ is a subtype of T₃, and further I₁, D₁ ⊆ D₂ and I₆, D₆ ⊆ D₃.

Therefore I₁, D₁ ⊆ D₃ and hence C₁ is a subclass of C₃ because T₁ is a proper subtype of T₃.

**Case 2:** Assume that T₁ = T₂ and T₂ ≠ T₃.

In this case I₁ ⊆ I₂, D₁ ⊆ D₂ and I₂, D₂ ⊆ D₃

I₁, D₁ ⊆ D₃ and thus C₁ is a subclass of C₃ because T₁ ≠ T₃

**Case 3:** Assume that T₁ ≠ T₂ and T₂ = T₃.

This case is similar to Case 2.

**Case 4:** Assume that all the types are same, i.e., T₁ = T₂ = T₃

C₁ is a subclass of C₂ and T₁ = T₂ implies I₁ ⊆ I₂, D₁ ⊆ D₂.
FIG. 5.6: UNIQUE CLASSES CORRESPONDING TO TYPES AND THEIR SUBCLASSES.
Similarly, $C_2$ is a subclass of $C_3$ and $T_2 = T_3$ implies $I_2 \subseteq I_3$, $D_2 \subseteq D_3$.

Hence, $T_1 = T_3$ and $I_1 \subseteq I_3$, $D_1 \subseteq D_3$ implies $C_1$ is a subclass of $C_3$.

Thus, 'subclass' relation satisfies transitive property. Hence it is a partial order.

**Definition 5.2.11: Class Hierarchy.** The set of classes $c$ with 'subclass' relation will form a directed acyclic graph which we denote by $G^c_1$. The set of unique classes again with subclass relation will form a directed acyclic graph $G^{C*}_1$ and is a subgraph of $G^c_1$. We also assume that the class corresponding to the root type $T^0$ is $C^0$ and is the root of class hierarchy. Figure 5.6 shows examples of two graphs $G^T_2$ and $G^c_1$. In this figure a double circled node represents a unique class $C^* \in C^*$. It is clear from the figure that the graphs $G^T_2$ and $G^{C*}_1$ are isomorphic.

**Definition 5.2.12: Class-Attribute Graph.** We first introduce some notations. We use $<\text{object reference}\.oid$ to refer to the identifier of the object.

To refer to the values of a property of an object, we use the notation:

$<$type name$>.<\text{property name}>.<\text{object reference}>$

Let $i_1$ and $i_2$ are members of $C_1$ and $C_2$ respectively, and let $p$ be an attribute of $i_1$ such that $T_1.p.i_1 = i_2.oid$, then we say that $i_1$ refers to $i_2$. 
Let \( G^C \) be a directed graph defined on \( C \) such that there is an edge joining from \( C_1 \) to \( C_2 \) if \( \exists \) at least one member of \( C_1 \) that refers to a member of \( C_2 \).

### 5.3 OPERATIONS ON CLASSES

In this section, we use "\( i \in C \)" to denote that the object \( i \) belongs to the class \( C \). The predicate "\( i \in C \)" is based on the object identity.

#### 5.3.1 UNION Operation

Let \( C_1(I_1,D_1):T_1 \) and \( C_2(I_2,D_2):T_2 \) be two given classes. We first define the class \( C'(I',D'):T \) as;

\[
C' = \text{lub}(C_1,C_2)
\]

where \( \text{lub} \) is the least upper bound w.r.t. subclass hierarchy.

Now we define the union of two classes \( C_1 \) and \( C_2 \) as a class \( R(I,D):T \) that contains all objects belonging to either \( C_1 \) or \( C_2 \), i.e.,

\[
R = C_1 \cup C_2
\]

where

- \( R \) is a subclass of \( C' \);
- \( I \cup D = (I_1 \cup D_1) \cup (I_2 \cup D_2) \)
- \( I = (I \cup D) \cap I' \)
- \( D = \text{Complement of } I \text{ in } (I \cup D) \)

**Case 1:** We assume that \( T_1 = T_2 \) which implies \( T_1 = T_2 = T \). By the definition of \( C' \), we have \( I_1 \subseteq I' \), \( D_1 \subseteq D' \), \( I_2 \subseteq I' \), and \( D_2 \subseteq D' \).
Then

\[ I = I' \cap ( (I_1 \cup D_1) \cup (I_2 \cup D_2) ) \]
\[ = I' \cap ( (I_1 \cup I_2) \cup (D_1 \cup D_2) ) \]
\[ = (I' \cap (I_1 \cup I_2)) \cup (I' \cap (D_1 \cup D_2)) \]
\[ = (I_1 \cup I_2) \cup \emptyset \quad \text{(Figure 5.7a)} \]
\[ = I_1 \cup I_2 \]

and

\[ D = (I \cup D) - I \]
\[ = (I_1 \cup I_2) \cup (D_1 \cup D_2) - (I_1 \cup I_2) \]
\[ = D_1 \cup D_2, \quad \text{since } (I_1 \cup I_2) \cap (D_1 \cup D_2) = \emptyset. \]

So, when \( T_1 = T_2 = T \) then

\[ I = I_1 \cup I_2, \]
\[ D = D_1 \cup D_2 \]

**Case 2: \( T_1 \neq T_2 \) and \( T_2 = T \)**

Since \( T_2 = T \) and \( C_2 \) is a subclass of \( C' \) implies

\[ I_2 \subseteq I' \quad \text{and } D_2 \subseteq D'. \]

Again \( T_1 \) is a proper subtype of \( T \) and \( C_1 \) is a subclass of \( C' \) implies \( I_1, D_1 \subseteq D' \).

Then

\[ I = I' \cap ( (I_1 \cup D_1) \cup (I_2 \cup D_2) ) \]
\[ = (I' \cap (I_2 \cup (I_1 \cup D_1 \cup D_2)) \] \]
\[ = (I' \cap I_2) \cup (I' \cap (I_1 \cup D_1 \cup D_2)) \]
\[ = I_2 \cup \emptyset \quad \text{(Figure 5.7b)} \]
\[ = I_2 \]

and

\[ D = (I \cup D) - I \]
\[ = (I_1 \cup I_2 \cup D_1 \cup D_2) - I_2 \]
\[ = I_2 \cup (I_1 \cup D_1 \cup D_2) - I_2 \]
(a) \[ T_1 = T_2 = T \]

(b) \[ T_1 \neq T_2 \text{ but } T_2 = T \]

(c) \[ T_1 \neq T_2 \text{ and } T_2 \neq T \]

**FIG. 5.7: UNION OF TWO CLASSES**

\[ C_1 = (I_1, D_1) : T_1 \text{ and } C_2 = (I_2, D_2) : T_2 \]
\[ I_1 \cup D_1 \cup D_2; \]
since, \( I_2 \cap (I_1 \cup D_1 \cup D_2) = \emptyset \) \hspace{1em} (Figure 5.7b)

So, when \( T_1 \) is a proper subtype of \( T \) and \( T_2 = T \) then
\[
I = I_2 \quad \text{and} \quad D = I_1 \cup D_1 \cup D_2
\]

Case 3: \( T_1 \neq T_2 \) and \( T_1 = T \)
This is similar to Case 2 and in fact we can deduce the following:

When \( T_2 \) is a proper subtype of \( T \) and \( T_1 = T \) then
\[
I = I_1 \quad \text{and} \quad D = I_2 \cup D_1 \cup D_2
\]

Case 4: \( T_1 \) and \( T_2 \) are proper subtypes of \( T \).
Since \( C_1 \) and \( C_2 \) are subclasses of \( C' \) and \( T_1 \), \( T_2 \) are proper subtypes of \( T \) implies
\[
I_1, D_1 \subseteq D' \quad \text{and} \quad I_2, D_2 \subseteq D'
\]
Therefore, \((I_1 \cup D_1) \cup (I_2 \cup D_2) \subseteq D' \) \hspace{1em} (Figure 5.7c)

Thus,
\[
I = I' \cap ((I_1 \cup D_1) \cup (I_2 \cup D_2)) = \emptyset \quad \text{since} \ I' \cap D' = \emptyset
\]
and
\[
D = (I \cup D) - I = (I_1 \cup D_1) \cup (I_2 \cup D_2) - \emptyset
\]
\[
= I_1 \cup D_1 \cup I_2 \cup D_2.
\]

So, when \( T_1 \) and \( T_2 \) are proper subtypes of \( T \) then
\[
I = \emptyset \quad \text{and} \quad D = I_1 \cup D_1 \cup I_2 \cup D_2.
\]
Theorem 5.3.1: Let \( \{C_0, C_1, \ldots, C_n\} \) be the set of all classes of type \( T \) such that \( C_i = (I_i, D_i) \) \( \forall i=0,1,\ldots,n \) then \( C = \{ \bigcup_{i=0}^{n} I_i, \bigcup_{i=0}^{n} D_i \} \) is a class and is equal to the unique class \( C^* = (I, D) : T \) corresponds to the type \( T \).

Proof: Since \( \{C_i\}_{i=0}^{n} \) is the set of all classes of a given type \( T \), let us assume that \( C^* = C_0 \). Now, for each \( i=1,2,\ldots,n \), \( C_i (I_i, D_i) \) is a subclass of \( C_0 \) implies,

\[
I_i \subseteq I_0 \quad \text{and} \quad D_i \subseteq D_0.
\]

Thus,

\[
\bigcup_{i=1}^{n} I_i \subseteq I_0 \quad \text{and} \quad \bigcup_{i=1}^{n} D_i \subseteq D_0 \quad (5.3.1.1)
\]

Suppose, \( C' = \text{lub}\{C_0, C_1, \ldots, C_n\} \) and let \( C(I, D) \) be the union of \( \{C_i\}_{i=0}^{n} \). By definition of union operator (see case 1),

\[
I = I_0 \cup I_1 \cup \ldots \cup I_n \quad \text{and} \quad D = D_0 \cup D_1 \cup \ldots \cup D_n \quad (5.3.1.2)
\]

Using (5.3.1.1) and (5.3.1.2), we obtain \( I = I_0 \) and \( D = D_0 \) imply the classes \( C = C(I, D) \) and \( C_0(I_0, D_0) \) are equal.

Remark 5.3.1: The union operation and other set-theoretic operations are based on object identity and therefore equal objects are not eliminated.

Remark 5.3.2: Least upper bound always exists and if it is not unique then we define the lub as the class with highest priority.

5.3.2 INTERSECTION Operation

Let \( C_1(I_1, D_1) : T_1 \) and \( C_2(I_2, D_2) : T_2 \) be two given classes. We first define a class \( C'(I', D') : T \) as:

\[
C' = \text{glb}(C_1', C_2')
\]

where \( \text{glb} \) is the greatest lower bound w.r.t. subclass hierarchy; \( C_1' \) and \( C_2' \) are the unique classes corresponding
to the types $T_1$ and $T_2$ respectively.

We now define the intersection of two classes $C_1$ and $C_2$ as a class $R(I,D):T$ that contains objects belonging to both $C_1$ and $C_2$, i.e.,

$$R = C_1 \cap C_2$$

where

- $R$ is a subclass of $C'$;
- $I \cup D = (I_1 \cup D_1) \cap (I_2 \cup D_2)$;
- $I = (I \cup D) \cap I'$ and
- $D = \text{Complement of } I \text{ in } (I \cup D)$

**Case 1:** $T_1 = T_2$ implies $T_1 = T_2 = T$

By assumption, the classes $C_1'$, $C_2'$ and $C'$ are the unique classes corresponding to the type $T$ implies

$$I_1' = I_2' = I', \quad D_1' = D_2' = D'$$

Then,

$$I = I' \cap ((I_1 \cup D_1) \cap (I_2 \cup D_2))$$

and

$$D = (I \cup D) - I$$

(Figure 5.8a)
FIG. 5.8: INTERSECTION OF TWO CLASSES

\[ C_1 = (I_1, D_1): T_1 \quad \text{and} \quad C_2 = (I_2, D_2): T_2 \]
So, when $T_1 = T_2 = T$ then
\[ I = I_1 \cap I_2 \quad \text{and} \quad D = D_1 \cap D_2 \]

Case 2: Assume that $T_1 \neq T_2$ and $T = T_2$.

Since $C'_2$ and $C'$ are the unique classes corresponding to the types $T_2$ and $T$ respectively and since $T_2 = T$ implies $C'_2$ is equal to $C'$.

Thus, $I' = I'_2$, $D' = D'_2$.

Also $C'$ is a subclass of $C'_1$ and hence
\[ I'_1, D' \subseteq D'_1 \]

Further, $C_1$ and $C_2$ are subclasses of $C'_1$ and $C'_2$ respectively
\[ \Rightarrow I_1 \subseteq I'_1, \quad D_1 \subseteq D'_1 \quad \text{and} \quad I_2 \subseteq I'_2, \quad D_2 \subseteq D'_2. \]

Therefore, using the above relations:
\[ I \cup D = (I_1 \cup D_1) \cap (I_2 \cup D_2) \]
\[ = (I_2 \cap D_1) \cup (D_1 \cap D_2) \quad \text{(Figure 5.8b)} \]
\[ \Rightarrow I = I' \cap ((I_1 \cup D_1) \cap (I_2 \cup D_2)) \]
\[ = I' \cap ((I_2 \cap D_1) \cup (D_1 \cap D_2)) \]
\[ = I_2 \cap D_1 \quad \text{(Figure 5.8b)} \]

and
\[ D = (I \cup D) - I \]
\[ = (I_1 \cup D_1) \cap (I_2 \cup D_2) - I \]
\[ = (I_2 \cap D_1) \cup (D_1 \cap D_2) - (I_2 \cap D_1) \]
\[ = D_1 \cap D_2, \]

since $(I_2 \cap D_1) \cap (D_1 \cap D_2) = \emptyset$ (Figure 5.8b)
So, when $T_1 \neq T_2$ and $T = T_2$ then

$$I = I_2 \cap D_1$$
and

$$D = D_1 \cap D_2$$

Case 3: $T = T_1$ and $T_1 \neq T_2$

This is similar to Case 2 and in fact we can deduce the following:

So, when $T = T_1$ and $T_1 \neq T_2$ then

$$I = I_1 \cap D_2$$
and

$$D = D_1 \cap D_2$$

Case 4: $T$ is a proper subtype of $T_1$ and $T_2$.

$C'$ is a subclass of $C'_1$ and $C'_2$ implies by assumption,

$$I', D' \subseteq D'_1; \quad I', D' \subseteq D'_2$$

Therefore,

$$I = I' \cap ((I_1 \cup D_1) \cap (I_2 \cup D_2))$$

$$= (I' \cap (I_1 \cap I_2)) \cup (I' \cap ((I_1 \cap D_2) \cup (D_2 \cap I_2)) \cup (D_1 \cap D_2)))$$

$$= \emptyset \cup (I' \cap (I_1 \cap D_2)) \cup (I' \cap ((D_1 \cap I_2) \cup (D_1 \cap D_2)))$$

$$= \emptyset \cup (I' \cap (D_1 \cap I_2)) \cup (I' \cap (D_1 \cap D_2))$$

$$= I' \cap (D_1 \cap D_2)$$

and

$$D = (I \cup D) - I$$

$$= (I_1 \cup D_1) \cap (I_2 \cup D_2) - I$$

$$= (I_1 \cap I_2) \cup (I_1 \cap D_2) \cup (D_1 \cap I_2) \cup (D_1 \cap D_2) - (I' \cap (D_1 \cap D_2))$$

$$= (I_1 \cap I_2) \cup (I_1 \cap D_2) \cup (D_1 \cap I_2) \cup ((D_1 \cap D_2) - (I' \cap (D_1 \cap D_2)))$$
So, when $T$ is a proper subtype of $T_1$ and $T_2$ then

$I = I' \cap D_1 \cap D_2$ and

$$D = (I_1 \cap I_2) \cup (I_1 \cap D_2) \cup (D_1 \cap I_2) \cup ((D_1 \cap D_2) - (I' \cap D_1 \cap D_2))$$

Remark 5.3.3: If $\text{glb}(C'_1, C'_2)$ does not exist then class $R$ is not created.

Remark 5.3.4: If $\text{glb}$ is not unique then we define the $\text{glb}$ as the class with highest priority.

5.3.3 DIFFERENCE Operation

Let $C_1(I_1, D_1): T_1$, $C_2(I_2, D_2): T_2$ be two given classes and $C'_1(I'_1, D'_1): T_1$ be the unique class corresponding to type $T_1$.

Now we define the difference of two classes $C_1$ and $C_2$ as a class $R(I, D): T$ that contains all objects that are in $C_1$ but not in $C_2$, i.e.,

$$R = C_1 - C_2$$

where

$R$ is a subclass of $C'_1$;

$T = T_1$;

$I \cup D = (I_1 \cup D_1) - (I_2 \cup D_2)$

$I = (I \cup D) \cap I'_1$

$D = \text{Complement of } I \text{ in } (I \cup D)$

Case 1: $T_1 = T_2$ implies $T_1 = T_2 = T$

By assumption,

$I_1 \subseteq I'_1$, $D_1 \subseteq D'_1$ and $I_2 \subseteq I'_2$, $D_2 \subseteq D'_2$. Then

$I = I'_1 \cap (I \cup D)$
\[ I_1' \cap (I_1 \cup D_1) - (I_2 \cup D_2) \]
\[ = I_1' \cap (I_2 - I_2) \cup (D_1 - D_2) \]
\[ = I_1 - I_2 \quad \text{(Figure 5.9a)} \]
and

\[ D = (I \cup D) - I \]
\[ = ((I_1 - I_2) \cup (D_1 - D_2)) - (I_1 - I_2) \]
\[ = D_1 - D_2 \]

since \((I_1 - I_2) \cap (D_1 - D_2) = 0 \quad \text{(Figure 5.9a)}\)

So, when \(T_1 = T_2 = T\) then

\[ I = I_1 - I_2 \quad \text{and} \]
\[ D = D_1 - D_2 \]

Case 2: Assume that \(T_1 \neq T_2\)

Then, \(I_1 \subseteq I_1'\), \(D_1 \subseteq D_1'\) is satisfied.

Now,

\[ I = I_1' \cap (I \cup D) \]
\[ = I_1' \cap ((I_1 \cup D_1) - (I_2 \cup D_2)) \]
\[ = I_1' \cap ((I_1 - (I_2 \cup D_2)) \cup (D_1 - (I_2 \cup D_2))) \quad \text{(Figure 5.9b)} \]
\[ = (I_1' \cap (I_1 - (I_2 \cup D_2))) \cup (I_1' \cap (D_1 - (I_2 \cup D_2))) \]
\[ = (I_1 \cap (I_2 \cup D_2)) \cup \emptyset \quad \text{(Figure 5.9b)} \]
\[ = I_1 - (I_2 \cup D_2) \]

and

\[ D = (I \cup D) - I \]
\[ = (I_1 - (I_2 \cup D_2)) \cup (D_1 - (I_2 \cup D_2)) - (I_1 - (I_2 \cup D_2)) \]
\[ = D_1 - (I_2 \cup D_2); \]
FIG. 5.9: DIFFERENCE OF TWO CLASSES
\[ C_1(I_1, D_1) : T_1 \] and \[ C_2(I_2, D_2) : T_2 \]
since \((I_1 - (I_2 \cup D_2)) \cap (D_1 - (I_2 \cup D_2)) = \emptyset\)

So, when \(T_1 \neq T_2\) then
\[
I = I_1 - (I_2 \cup D_2) \quad \text{and} \quad D = D_1 - (I_2 \cup D_2)
\]

5.3.4 CARTESIAN PRODUCT

The cartesian product of two classes \(C_1(I_1,D_1):T_1\) and \(C_2(I_2,D_2):T_2\) is a class \(R(I,D):T\) whose members are obtained by combining objects from the given classes and is denoted by
\[
R = C_1 \times C_2
\]

We define the new type \(T\) and the class \(R\) as follows:
If \(T_1 = \{(a_i,A_i)\}_{i=1}^n\) and \(T_2 = \{(b_j,B_j)\}_{j=1}^m\) then \(T = \{(e_k,E_k)\}_{k=1}^{n+m}\) and \(T\) is a subtype of the root type \(T^0\) such that for \(k=1,2,\ldots,n\); \(e_k = a_k\), and \(E_k = A_k\) and with
\(k=n+j, e_k = b_j\) and \(E_k = B_j\) for \(j=1,\ldots,m\);

\(R\) is the unique class corresponding to \(T\) and is a subclass of \(C^0\);
\(I \cup D = \{i \mid i=(i_1, i_2), i_1 \in C_1, i_2 \in C_2 \text{ such that} \}
\[\forall (e_k,E_k) \in A(T), \]
\[T.e_k.i = \begin{cases} T_1.e_k.i_1 & \text{if } (e_k,E_k) \in A(T_1), \\ T_2.e_k.i_2 & \text{if } (e_k,E_k) \in A(T_2), \end{cases} \]

\(I = I \cup D\),
\(D = \emptyset\)
5.3.5 CHOOSE Operation

Consider the class-attribute graph \( G_2^c \). Then, for a sequence of classes \( \{C_i\}_{i=1}^n \), if a sequence of attributes \( p_j \) of \( i_j \) and for each member \( i_j \) in \( C_j \), a member \( i_{j+1} \in C_{j+1} \) exist such that \( T_j.p_j.i_j = i_{j+1}.oid \) for \( j=1, \ldots, n-1 \) and \( C_n \) is a leaf node in \( G_2^c \), we call this sequence of classes \( \{C_i\}_{i=1}^n \), as a main branch of \( C_1 \) in \( G_2^c \).

Now, the choose operation defined on a given class \( C_1(I_1,D_1):T_1 \) is a class \( R(I,D):T_1 \) whose members are subset of members of the given class satisfying the choose condition.

Let us assume that the choose condition \( B \) defined on a main branch \( \{C_i\}_{i=1}^n \) of \( C_1 \) is given by:

\[
B = B_1 \theta_1 B_2 \theta_2 \cdots \theta_{n-1} B_n \theta_n B_0^o \theta_{n+1} B_1^1 \cdots \theta_{n+1} B_l^l
\]

where each \( B_j \) is a boolean expression connecting a number of conditions \( B_j^k \) (\( k=1,2,\ldots,m \)) by means of boolean operators AND, OR and NOT; each \( B_j^k \) is of the form: For a given object \( i_j \in C_j \),

\[
B_j^k(i_j) = \begin{cases} 
<T_j.p.i_j><\text{relational operator}><\text{constant value}> & \text{or} \\
<T_j.p.i_j><\text{relational operator}><T_j.q.i_j> & 
\end{cases}
\]

for \( p, q \) are simple attributes of \( T_j \);

each \( \theta_j \) is a boolean operator;

for given objects \( i_o \in C_r, j_o \in C_s \),

\[
B^t(i_o,j_o) = <T_r.p.i_o><\text{relational operator}><T_s.q.j_o>,
\]

for \( p, q \) are simple attributes of \( T_r \) and \( T_s \) respectively.

The relational operator set \{\( =, <, \leq, >, \geq, = \)\} is
applicable to attributes having ordered values. In case the values of the attributes are unordered the only relational operators meaningful are "=" and "#". The general form of choose operation is

\[ R = \sigma_B (C_1) \]

where \( \sigma \) denotes the CHOOSE operator and we define the class \( R \) as:

\( R \) is a subclass of \( C'_1(I'_1,D'_1):T_1 \) where \( C'_1 \) is the unique class corresponding to type \( T_1 \);

\( I \cup D = \left\{ i_1 \in C_1 \mid \exists \text{ a set of objects } \{i_2, i_3, \ldots, i_n\}, \right. \)

\[ i_j \in C_j, \quad j=2,\ldots,n \text{ such that} \]

\( (a) \text{ Boolean expression } B \text{ is true.} \)

\( (b) \quad ((i_2.\text{oid}=T_1.p_1.i_1) \text{AND}(i_3.\text{oid}=T_2.p_2.i_2) \text{AND} \)

\[ \ldots \text{AND}(i_n.\text{oid}=T_{n-1}.p_{n-1}.i_{n-1}) \text{ is true;} \]

\( I = (I \cup D) \cap I' \); \n
\( D = \text{Complement of } I \text{ in } (I \cup D). \)

Remark 5.3.5: If an instance \( i \in R \) then \( i \in C_1 \), i.e., new objects are not created.

Remark 5.3.6: If the choose condition involves two or more main branches of \( C_1 \) with logical operation then one by one the choose operation can be applied. (See Remark 6.3.7).

**5.3.6 PARTITION Operation**

The partition operation applied on a class \( C'_1(I'_1,D'_1):T_1 \) is a class \( R(I,D):T \) such that the attributes of type \( T \) is a subset of attributes of type \( T_1 \).

The general form of PARTITION operation is:
\[ R = \pi_{A(I)}(C_1) \]

where \( \pi \) denotes the PARTITION operator and \( A(I) \) is an attribute list and is a subset of \( A(T_1) \). We define the type \( T \) and the class \( R \) as follows:

If \( A(T_1) = \{(a_i, A_i)\}_{i=1}^{n} \) then \( A(T) = \{(b_i, B_i)\}_{i=1}^{m} = A(I) \);

with \( b_j = a_i \); \( B_j = A_i \) for some \( i \).

\( T \) is a subtype of the root type \( T^0 \).

\( R \) is the unique class corresponding to type \( T \) and is a subclass of \( C^0 \).

\[ I \cup D = \{ i \mid i \in C_1 \} \]

\[ I = I \cup D \]

\[ D = \emptyset \]

Remark 5.3.7: Partition operation creates new objects whose set of attributes is a subset of attributes of the given class.

5.4. EXAMPLES OF QUERIES

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

Query 5.4.1:

Choose all students with status "PART-TIME".

\[ \text{STUDENT-PART-TIME} = \sigma_{B}(\text{STUDENT}) \]

where the condition \( B \) is:

\[ \text{STUDENT.status} = \text{"PART-TIME"} \]

The class \( \text{STUDENT-PART-TIME} \) is a subclass of \( \text{STUDENT} \).

Query 5.4.2:

Choose all science students with status "PART-TIME" or all arts students with status "FULL-TIME".
Step 1:

\[ \text{STUDENT-SCIENCEpt} = \sigma_B (\text{SC-STUD}) \]
where the condition \( B \) is:
\[ \text{SC-STUD.status} = '\text{PART-TIME}' \]
The class \( \text{STUDENT-SCIENCEpt} \) is a subclass of \( \text{SC-STUD} \).

Step 2:

\[ \text{STUDENT-ARTSft} = \sigma_B (\text{ARTS-STUD}) \]
where the condition \( B \) is:
\[ \text{ARTS-STUD.status} = '\text{FULL-TIME}' \]
The class \( \text{STUDENT-ARTSft} \) is a subclass of \( \text{ARTS-STUD} \).

Step 3:

\[ \text{STUDENT-SCpt-ARTSft} = \text{STUDENT-SCIENCEpt} \cup \text{STUDENT-ARTSft} \]
The class \( \text{STUDENT-SCpt-ARTSft} \) is a subclass of \( \text{STUDENT} \).

Query 5.4.3:

Choose all science students with status "PART-TIME" and all students with age greater than 30.

Step 1:

\[ \text{STUDENT-SCIENCEpt} = \sigma_B (\text{SC-STUD}) \]
where the condition \( B \) is:
\[ \text{SC-STUD.status} = '\text{PART-TIME}' \]
The class \( \text{STUDENT-SCIENCEpt} \) is a subclass of \( \text{SC-STUD} \).

Step 2:

\[ \text{STUDENT-AGEgt30} = \sigma_B (\text{STUDENT}) \]
where the condition \( B \) is:
\[ \text{STUDENT.age} > 30 \]
The class \( \text{STUDENT-AGEgt30} \) is a subclass of \( \text{STUDENT} \).

Step 3:

\[ \text{STUDENT-SCpt-gt30} = \text{STUDENT-SCIENCEpt} \cap \text{STUDENT-AGEgt30} \]
The class STUDENT-SCpt-gt30 is a subclass of STUDENT.

**Query 5.4.4:**
Choose all departments located at 'Delhi' or 'Bangalore' but not science departments located at 'Bangalore'.

**Step 1:**

DEPARTMENT-DB = σ _B_ (SC-DEPT)

where the condition _B_ is:

DEPARTMENT.location = 'DELHI' OR
DEPARTMENT.location = 'BANGALORE'

The class DEPARTMENT-DB is a subclass of SC-DEPT

**Step 2:**

DEPARTMENT-SCIENCEb = σ _B_ (SC-DEPT)

where the condition _B_ is:

SC-DEPT.location = 'BANGALORE'

The class DEPARTMENT-SCIENCEb is a subclass of SC-DEPT

**Step 3:**

DEPARTMENT-RESULT = DEPARTMENT-DB - DEPARTMENT-SCIENCEb

The class DEPARTMENT-RESULT is a subclass of DEPARTMENT.

**Query 5.4.5:**

Let the types CIRCLE, LINE and SQUARE be defined as follows:

CIRCLE = ( (centre,POINT), (radius,Real) )

LINE = ( (start,POINT), (end,POINT) )

SQUARE = ( (location,POINT), (length,Real) )

Now, consider the query:

Choose all circles and lines where the centre of circle is same as the start or end point of the line.
Step 1:

CIRCLE-LINE = CIRCLE X LINE

The class CIRCLE-LINE is a subclass of root class C^o.

Step 2:

CIRCLE-LINE-pt = \sigma_B(CIRCLE-LINE)

where the condition B is:
CIRCLE.centre = LINE.start OR
CIRCLE.centre = LINE.end

The class CIRCLE-LINE-pt is a subclass of CIRCLE-LINE.

Query 5.4.6:

Choose all circles and squares where the centre of circle
is same as location of the square. Select attributes
location and radius.

Step 1:

CIRCLE-SQUARE = CIRCLE X SQUARE

The class CIRCLE-SQUARE is a subclass of root class C^o.

Step 2:

CIRCLE-SQUARE-Loc = \sigma_B(CIRCLE-SQUARE)

where the condition B is:
CIRCLE.centre = SQUARE.location

The class CIRCLE-SQUARE-Loc is a subclass CIRCLE-SQUARE.

Step 3:

CIRCLE-SQUARE-p = \pi_{A()} CIRCLE-SQUARE-Loc

where the attribute list A() is as follows:
A() = {(location,POINT),(radius:real)}

The class CIRCLE-SQUARE-p is a subclass of root class C^o.