

DE LA SALLE UNIVERSITY

FORMALISMS OF OBJECT-ORIENTED
DATABASE

8570480

A Thesis
Presented to the
Faculty of the Computer Science Graduate Program
College of Computer Studies
De La Salle University

In Partial Fulfillment of the
Requirements for the Degree
Master Of Science in Computer Science

by

MARY ANN G. NAVALTA

May, 1990

THE DLSU-EAC LIBRARY



DE LA SALLE UNIVERSITY

ABSTRACT

The database can be viewed as a collection of tables (relational model), as a collection of record types and sets (network model) or as a forest of tree structures (hierarchical model). The logical model of the database is the correct level for database users to focus on. Considering the performance of these databases is highly dependent upon the database design, many of them have been over designed to ensure adequate performance. The performance of the database depends primarily on the efficiency of the data structures used to represent the data in the database and on how efficiently the system is able to operate on these data structure.

The database models mentioned above came into existence because of a need for a tool to make data processing professionals and end users of their services more productive. To date, most of the research work is focus on the other database model which is object-oriented model.

Object-oriented modelling represents a successful unifying paradigm in various areas of computing specifically databases. In an object-oriented environment the world is viewed as a collection of active objects. Each object has a set of variables which determine the state of the object. Each object can understand a set of predefined messages. Objects communicate with each other by sending messages. Sending a message to inquire about the state of an object is like invoking a function and sending a message to change the state of an object is like an update operation. Updates can be



DE LA SALLE UNIVERSITY

generated because sending a message to an object can cause more messages to be sent to other objects. Similar objects are defined by classes. Classes are organized as class/subclass hierarchies. Subclasses inherit the messages of their superclasses. This inheritance and the class/subclass relationships support generalization abstraction.

This research is directed towards formulating the formal formalisms of object-oriented database. This study shows that object-oriented increases the productivity of the design systems by providing modelling facilities which closely mirror the proofs that object modelling improves the structure of the data in promoting the availability of the data. Thus, the study substantiates its significance in providing a better set of constructs for modelling database. Moreover, the study proves that object paradigm helps the semantic gap between databases and applications.



DE LA SALLE UNIVERSITY

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vii
LISTINGS	ix
CHAPTER 1 INTRODUCTION	1
1.1 Relational Database Concepts	3
1.2 Object-Oriented Concepts	6
1.3 Motivation	9
1.4 Statement of the Problem	10
1.4.1 General Objective	12
1.4.2 Specific Objectives	12
1.4.3 Significance of the Study	13
1.4.4 Scope and Limitations	14
1.4.5 Methodology	15
1.5 Organization of the Paper	17



DE LA SALLE UNIVERSITY

	Page
CHAPTER 2 STRUCTURAL CONCEPTS OF RELATIONAL AND OBJECT-ORIENTED DATABASES	20
2.1 Relational Data Model	21
2.2 Object-Based Logical Models	26
2.2.1 Chen's Entity-Relationship Model (ER Model-Original Classes of Objects)	28
2.2.2 Teorey's Logical Relational Design Methodology (LRDM-Extended Classes of Objects)	29
2.3 Blaha's Object-Oriented Model	31
2.3.1 High Level Representation	33
2.3.2 Middle Level Representation	40
2.3.3 Low Level Representation	47
2.4 Comparison of Data Representation Between Chen's Entity-Relationship Model and Blaha's Object-Oriented Model	47
2.5 Performance Measures	59
CHAPTER 3 PROPERTIES OF AN OBJECT-ORIENTED PARADIGM	62
3.1 Objects, Messages and Encapsulation	63
3.2 Classes, Inheritance and Class Categories	70
3.3 Flow Process/Data Structures	74



DE LA SALLE UNIVERSITY

	Page
3.4 Phases in Developing and Establishing the Formalisms of Object-Oriented Model	79
3.5 Prototype of Object-Oriented Primitive Operations	81
CHAPTER 4 FORMALISMS OF OBJECT-ORIENTED MODEL	95
4.1 Definition Presentation of Object- Oriented Concepts	96
4.2 Model for Object-Oriented Databases	104
4.3 Diagram Model for Object-Oriented Databases	115
4.4 Data Security Protocol	122
4.5 Application Model for Object-Oriented Database	135
4.6 Assessment of Object-Oriented Model and Other Models	140
CHAPTER 5 PERFORMANCE	148
CHAPTER 6 CONCLUSION AND FUTURE DIRECTIONS	170
REFERENCES	175



DE LA SALLE UNIVERSITY

LIST OF FIGURES

	Page
Fig. 2.1 Three Levels of Representation	32
Fig. 2.2 Generalization Relationship	34
Fig. 2.3 Aggregation Relationships	35
Fig. 2.4 Association Relationships	36
Fig. 2.5 Qualification Aggregation Relationship	39
Fig. 2.6 Middle Level For An Object	42
Fig. 2.7 Middle Level For A Generalization Relationship	44
Fig. 2.8 Middle Level For An Existence-Dependence Aggregation	45
Fig. 2.9 Middle Level For A Free-Standing Aggregation	46
Fig. 2.10 Low Level Representation	48
Fig. 2.11 Levels of Data Abstraction	52
Fig. 2.12 Comparison on Generalization Relationships	54
Fig. 2.13 Substructure of Object Customer	55
Fig. 2.14 Comparison Of Aggregation Relationship	57
Fig. 2.15 Comparison on Association Relationship	58



DE LA SALLE UNIVERSITY

	Page
Fig. 2.16 Comparison Between Object-Oriented Model and ER Model/Relational Model	60
Fig. 3.1 Data Structures	75
Fig. 3.2 Class Hierarchy of Equipment	83
Fig. 3.3 Memory Organization of Centrifugal Class Properties	88
Fig. 4.1 Object Identifier	102
Fig. 4.2 Definition of Nodes and Arcs	116
Fig. 4.3 Representation of Memo Class and Memo Instance	136
Fig. 4.4 Logical Data Structure which Represents the Memo Class	141
Fig. 4.5 Logical Data Structures which Represent Memo Instance	142



DE LA SALLE UNIVERSITY

LISTINGS

	Page
Listing 1 Centrifugal Pump Example Program	85
Listing 2 Chain of Inheritance for the Pump Class	90

