ORGANIZING INFORMATION
Principles of Data Base and Retrieval Systems

Dagobert Soergel

College of Library and Information Services
University of Maryland
College Park, Maryland

1985

ACADEMIC PRESS, INC.

(Harcourt Brace Jovanovich, Publishers)

Orlando San Diego New York London
Toronto Montreal Sydney Tokyo

c1c?72o!c)

ACADEMIC PRESS, INC.
Orlando, Florida 32887

United Kingdom Edition published by
ACADEMIC PRESS INC. (LONDON) LTD.
24-28 Oval Road, London NW1 7DX

LIBRARY OF CONGRESS CATALOGING-IN-PUBLICATION DATA

Soergel, Dagobert.
Organizing information.

(Library and information science)
Bibliography: p.
Includes index.
1. Information storage and retrieval systems.
I. Title. II. Series.
Z699.S539 1985 025'.04 83-15741
ISBN 0-12-654260-0 (alk. paper)
ISBN 0-12-654261-9 (paperback)

PRINTED IN THE UNITED STATES OF AMERICA

85 86 87 88 9 8 7 6 5 4 3 2 1

Contents

Preface xiii

I
THE SYSTEMS APPROACH TO INFORMATION TRANSFER

1 INFORMATION SYSTEMS FOR PROBLEM SOLVING 3

2 THE NATURE OF INFORMATION 9
Objectives 9
2.1 The Role of the Image 9
2.2 Image Formation by a Solitary Individual 11
2.3 Image Formation through Communication 12
2.4 A Model of Information Transfer and Use 14
2.5 Data, Information, and Knowledge 16
2.6 Classification of Information Systems by Services Delivered 18
2.7 Summary and Evolving Principles 20

3 THE STRUCTURE OF INFORMATION 21
Objective 21
Introduction 21
3.1 The University Data Base: An Example 21
3.2 Elements of Information Structure 23
3.3 Analyzing Reference Tools: An Example 32

4 THE INFORMATION TRANSFER NETWORK 33
Objectives 33
Introduction 33
4.1 Transactions in the Information Transfer Network 34
4.2 Configurations of Transactions 35
4.3 Characteristics of Transactions 37

5 THE STRUCTURE OF INFORMATION SYSTEMS 41
Objective 41
5.1 The Overall Structure of Information Systems 41
5.1.1 Identifying the Needs of Specific Users 44
5.1.2 Acquiring Entities or Information about Them 45
5.1.3 The ISAR System 45
5.1.4 Making Entities or Information Available to the User 45
5.1.5 Further Processing of Information 46
5.1.6 Identifying Needs in General: The Needs Directory 46
5.1.7 Public Relations 49
5.1.8 Functional versus Organizational Breakdown of a System 50
5.2 The Retrieval Problem: A View from Scratch 50
5.2.1 The Structure of Index Languages and Files: A Preview 56
5.3 The Structure of an ISAR System 57
5.3.1 The ISAR System as a Whole 59
5.3.2 System Rules and Conventions. The Conceptual Schema 59
5.3.3 Notes on the Other Components of an ISAR System 60
5.4 Definitions 62
5.4.1 Descriptor, Lead-in Term, Index Language,
Lead-in Vocabulary, Thesaurus 62
5.4.2 Search Request, Query, Query Statement, Query Formulation 62
5.4.3 Indexing, Cataloging, Coding 63

II
OBJECTIVES OF ISAR SYSTEMS

6 SYSTEMS ANALYSIS 69
Objectives 69
Introduction 69
6.1 Approaches to Decision Making 70
6.2 Functions in the Systems Analysis Process 71
6.3 Phases in Systems Analysis: System Life Cycle 79
6.4 Information and Data Collection in Systems Analysis 84
6.5 Selection Decisions 86
6.6 Resource-Oriented versus Procedure-Oriented Systems Analysis 89
6.7 Performance versus Impact of Information Services 90

7 ASSESSMENT OF USERS' PROBLEMS AND NEEDS 93
Objectives 93
Introduction 93
7.1 User Studies as a Basis for System Design 94
7.2 Principles for the Study of Needs 97
7.2.1 Setting Priorities 97
7.2.2 Shared Responsibility—User and Information Professional 98
7.2.3 Need, Want, Demand or Recognized Need, Use, and Impact 98
7.2.4 Unencumbered Assessment of Needs 100

7.3 Approaches to Studying Needs 100
7.3.1 Problem Analysis Based on Records 102
7.3.2 Analysis of Requests and Searching Behavior 103
7.3.3 Querying (Potential) Users about Their Needs 104

8 OBJECTIVES OF ISAR SYSTEMS 109
Objectives 109
Introduction 109
8.1 A Measure of Answer Quality (Local Performance) 114
8.1.1 Measures for Individual Aspects of Answer Quality 114
8.1.2 A Single Composite Measure of Answer Quality 123
8.2 A Measure of Global ISAR System Performance 125
8.3 Testing versus Evaluation 126
8.4 Relevance and Relevance Judgments 127
8.5 Implications for ISAR System Design and Operation 129
8.5.1 Implications for System Design: Innovative Features 129
8.5.2 Implications for System Operation 131
8.5.3 Relevance Judgments and Professionalism 132

III
DATA SCHEMAS AND DATA STRUCTURES

9 DATA SCHEMAS AND FORMATS 137
Objectives 137
Introduction 137
9.1 Designing a Conceptual Schema 138
9.1.1 Rules and Conventions for the Form of Entity Identifiers 142
9.1.2 General Rules for Establishing Relationships 143
9.2 Record Formats: General Considerations 144
9.2.1 Functions of Records in Data Base Processes 144
9.2.2 Structure as a Key Concept 146
9.2.3 Fixed Field and Variable Field Records 148
9.3 Criteria for the Design and Evaluation of Data Schemas 150
9.4 Input Formats 152
9.4.1 Design Considerations for Input Formats 155
9.5 Output Formats 157
Appendix 9.1 Examples of Record Formats 159

10 ELEMENTARY QUERY FORMULATION 165
Objectives 165
10.1 Logical *AND* 165
10.2 Logical *OR* 166
10.3 Logical *AND* with Logical *OR* 167
10.4 The Ambiguity of Natural Language “and” 168
10.5 Levels of Parentheses 169
10.6 Logical *NOT* and Its Pitfalls 170

11 DATA STRUCTURES AND ACCESS 173
, Objectives 173
11.1 Exploration of Data Structures 173
11.2 Functions and Characteristics of Data Structures 195
11.3 Main File and Index File(s) as a Data Structure 196
11.3.1 Notes on Terminology 196
11.3.2 Some Usage and Design Considerations 197
11.3.3 A Main File of Entities as Part of the Data Structure 198
11.4 The Concept of Order 199
11.5 The Two-Dimensional Continuuni of File Types 202
11.6 Trade-offs between Data Base Costs and Searching Costs 203
11.6.1 Amount of Information 203
11.6.2 Degree of Order 204
11.6.3 Design Considerations 204
11.7 Definitions 206

IV
INDEX LANGUAGE FUNCTIONS AND STRUCTURE

12 TERMINOLOGICAL CONTROL 213
Objectives 213
Introduction; The Problem of Terminological Control 213
12.1 Concepts versus Terms: The Synonym-Homonym Structure 217
12.2 Grouping Closely Related Concepts: The Equivalence Structure 219
12.3 Classificatory Structure 220
12.4 Index Language 221
12.5 Thesaurus 222

13 INDEX LANGUAGE FUNCTIONS 225
Objective 225
13.1 Review: The Information Retrieval Problem 225
13.2 The Role of the Index Language in Indexing 227
13.2.1 Disadvantages of Entity-Oriented indexing 227
13.2.2 Request-Oriented Indexing: General Approach 230
13.2.3 Request-Oriented Indexing: Implementation 233
13.2.4 Supplementary Entity-Oriented Indexing 236
13.3 The Role of the Index Language in Searching 239
13.3.1 The Checklist Technique Applied to Query Formulation 239
13.3.2 Compensating for the Lack of Request-Oriented Indexing 240
13.4 The Role of the Index Language in Data Base Organization 240
13.5 Choosing the Best indexing Approach 244
13.5.1 Cost of Indexing 244

13.5.2 Quality of Indexing 245
13.5.3 Cost and Quality of Searching 246
13.6 The Functions of Hierarchy: A Summary 246
13.7 A Philosophy of Indexing and Classification 247

14 INDEX LANGUAGE STRUCTURE 1: CONCEPTUAL 251
Objectives 251
introduction 251
14.1 Hierarchy 252
14.2 Concept Combination and Semantic Factoring. Facet Analysis 256
14.3 Interaction between Concept Combination and Hierarchy 261
14.4 Application and Illustration: Searching 272
14.5 Conceptual Analysis, Facet Analysis: Elaboration 278
14.5.1 Developing a Scheme of Facets 278
14.5.2 Recognizing General Concepts 278
14.5.3 Subfacets 280
14.5.4 Facet Analysis and Relationships among Precombined Descriptors 280
14.5.5 Advantages of Semantic Factoring and Facet Analysis 280
14.6 Hierarchy: Elaboration 281
14.6.1 Hierarchical versus Associative Relationships 281
14.6.2 Types of Hierarchical Relationships 282
14.6.3 Introducing New Broader Concepts 283
14.7 Concept Formation in Thesaurus Building 285

15 INDEX LANGUAGE STRUCTURE 2: DATA BASE ORGANIZATION 289
Objectives 289
Introduction 289
15.1 The Problem 290
15.2 Grouping Entities. Searching in Grouped Files 291
15.2.1 The Idea of Grouping and Precombined Descriptors 291
15.2.2 From Ideal to Reality: Limited Precombination 299
15.2.3 Access Advantages of Grouped Files 303
15.3 Grouping versus Description of Entities 303
15.4 Postcombination and Precombination 305
15.4.1 Posicombinauon versus Precombination as a Matter of Degree 305
15.4.2 Deciding on the Overall Degree of Precombination 307
15.4.3 Deciding on Individual Precombined Descriptors 308
15.4.4 Precombined Descriptors in Indexing and Searching 310
15.5 Organizing an Index Language for Access 312
15.5.1 Descriptor-Find Indexes 313
15.5.2 Arrangement and Designation of Descriptors 317
15.6 A Unified Index Language for Different Search Mechanisms 322

V
ISAR SYSTEMS OPERATION AND DESIGN

16 INDEXING SPECIFICITY AND EXHAUSTIVITY 327
Objectives 327
16.1 Importance of Indexing for System Performance 327
16.2 Definition of Exhaustivity and Specificity of Indexing 328
16.2.1 Definition of Exhaustivity 328
16.2.2 Definition of Specificity 330
16.3 Effects of Exhaustivity and Specificity of Indexing on Retrieval Performance 331
16.3.1 Effects of Exhaustivity 332
16.3.2 Effects of Specificity 336
16.3.3 Misconceptions about the Effects of Exhaustivity and Specificity 337
16.3.4 Summary 338
16.4 Designing Indexing Rules and Procedures 338
16.4.1 Factors Influencing Exhaustivity 338
16.4.2 Factors Influencing Specificity 339
16.4.3 Cost Considerations 340

17 SEARCHING 343
Objective 343
Introduction 343
17.1 Recognize and State the Need. State Search Requirements 346
17.1.1 Recognize the Existence of a Need 346
17.1.2 Develop the Query Statement 347
17.1.3 Determine Specific Search Requirements 350
17.2 Develop the Search Strategy 350
17.2.1 Formulate the Query Conceptually 351
17.2.2 Select Sources and Arrange Them in a Search Sequence 359
17.2.3 Translate the Conceptual Query Formulation into the Language of Each Source 362
17.2.4 Free-Text Searching 366
17.2.5 The Interplay between Conceptual and Source-Specific Query Formulation 368
17.3 Execute the Search Strategy 368
17.4 Review Search Results and Revise Search 369
17.5 Edit Search Results and Send Them to the User 370
17.6 Check Whether the Answer Was Helpful 371
17.7 Interaction 371
17.8 Monitor the Search Process and Assess Results 376

18 DESIGN AND EVALUATION OF INFORMATION SYSTEMS 379
Objective 379
Introduction 379

18.1 Determine User Requirements and Abilities 382
18.1.1 Determine the Scope of the Information System 382
18.1.2 Obtain Search Requests 383
18.2 Develop the Collection and Obtain Relevance Judgments 384
18.2.1 Develop the Collection of Entities 384
18.2.2 Obtain Presearch Relevance Judgments 384
18.3 Design and Construct the ISAR System 385
18.4 Operate the ISAR System 387
18.4.1 Index Entities 387
18.4.2 Formulate Queries 387
18.4.3 Retrieve Entities and Judge Their Relevance 389
18.5 Evaluate ISAR System Performance 389
18.5.1 Macroanalysis: Performance Measures 389
18.5.2 Microanalysis: Retrieval Successes and Failures 390
18.6 Retrieval Testing and System Design and Operation 390
18.7 Cost-Benefit Analysis as a Design Principle 392
18.8 Problem-Orientation as a Design Principle 395

Bibliography 399
Author Index 423
Subject Index 429

Preface

This book gives a theoretical base and a perspective for the analysis, de» sign, and operation of information systems, particularly their information storage and retrieval (ISAR) component, whether mechanized or manual. Information systems deal with many types of entities: events, persons, documents, business transactions, museum objects/research projects, and technical parts, to name a few. Among the purposes they serve are to inform the public, to support managers, researchers, and engineers, and to provide a knowledge base for an artificial intelligence program. The principles discussed in this book apply to all these contexts. The book achieves this generality by drawing on ideas from two conceptually overlapping areas—data base management and the organization and use of knowledge in libraries—and by integrating these ideas into a coherent framework. The principles discussed apply to the design of new systems and, more importantly, to the analysis of existing systems in order to exploit their capabilities better, to circumvent their shortcomings, and to introduce modifications where feasible.

This book is intended for use in an introductory course on organizing and retrieving information (called, for example, "Introduction to the Organization of Information," "Introduction to Information Storage and Retrieval," or "Introduction to Information Science") offered in a school of library and information science, a business school, or a more broadly based information studies or information management program. Beyond that, it is meant to inspire, a modernization and integration of the library/information science curriculum. The book can be used fora broadly based course that teaches the general principles of ISAR and treats cataloging and reference service as specific areas of application. Such a course not only overcomes the artificial separation of cataloging and reference but also gives students wide flexibility in choosing their first position and a sound base from which to strike out in many directions in the further development of their careers. It can be offered as ä package of designated sections of the cataloging and reference course, without changing any course numbers. Such a course can be extended to include students from business information systems, journalism, and cognate areas: The theoretical base is common to all, but the application areas are different. This book is also suitable for self-study by practitioners who are looking for a sounder theoretical base for their daily work, A workbook with exercises and discussion of additional examples is in preparation; a draft is available from the author.

Information studies is a nascent field. It shows considerable confusion in its terminology, partly due to the lack of a prevalent conceptual framework: The same term is used for different concepts; different terms are used for the same concept. This book follows the terminology of major writers in the field but sometimes introduces a new term for a new concept or to replace an existing term that reflects a faulty concept analysis.

Throughout the book the development of ideas proceeds from the point of view that information specialists are professionals who cooperate with the user in determining information needs and who use their knowledge to design systems or to do searches to meet these needs, as opposed to merely looking for what the user thinks is needed.

Organization of the Book. Part I places information systems in context; it discusses the nature and structure of information and lays out the overall structure of an information system. Part II provides the basis for considering the design and use of ISAR systems in light of the objectives to be achieved, allowing for a discussion of the merits of design alternatives in Parts III through V. Part III deals with the logical representation of data and with structures for providing access to these data. It deals on a general level with the rules and conventions necessary in an ISAR system. Part IV focuses attention on subject retrieval (but many of the principles have more general application). It discusses the nature of index languages—terminological control, basic functions, and conceptual structure. Part V discusses indexing and searching and, in conclusion, testing and design of the system.

Acknowledgments. This book was developed from lectures given at the University of Maryland; the students' many questions forced me to sharpen my thinking, and their comments on successive versions of the manuscript were extremely useful Norman Roberts, Harold Borko, and Raya Fidel all gave good advice, which, among other things, was instrumental in reducing this book to a manageable size. Jane Bergling and Marie Somers typed the manuscript many times over, somehow managing to interpret my scribbled revisions. My wife Lissa was always ready to examine and discuss ideas and to make suggestions concerning both content and form; she also spent hours editing and proofreading. I owe an intellectual debt to many in the field but above all to the pioneering spirit of Calvin Mooers.

PART I

The Systems Approach to Information Transfer

CHAPTER 12

Terminological Control

OBJECTIVES

The objectives of this chapter are to explain the problems of terminological control as applied in indexing and searching and to define *thesaurus* with its synonym-homonym structure (all terms), *classificatory structure* (concepts expressed by preferred terms), *index language* (subject descriptors), and also *lead-in vocabulary* (lead-in terms).

INTRODUCTION: THE PROBLEM OF TERMINOLOGICAL CONTROL

Controlling entity identifiers is essential for retrieval success. This chapter deals with subject retrieval and thus is concerned with control over subject descriptors, that is, identifiers for subjects or concepts. The principles discussed apply to controlling identifiers in general. In free-text entity representations—such as names of organizations or titles, abstracts, or full text of documents—concepts are identified by terms. But free terms used as concept identifiers present many problems. Consider searching in a machine-readable bibliographic data base using terms in titles, abstracts, or full text *ifree-text searching* or searching with an *uncontrolled vocabulary*.) The topic is *Harbors,* which may seem like a very simple query. Since any term is searchable, why not try the query formulation *Harbors.* It will find some relevant documents but by no means all (low recall). A look at the sample list of terms occurring in the data base in Fig. 12.1a shows *Harbor* (singular), *Harbour,* and *Port.* To achieve high recall one must look for all these terms. Thus the query formulation should be

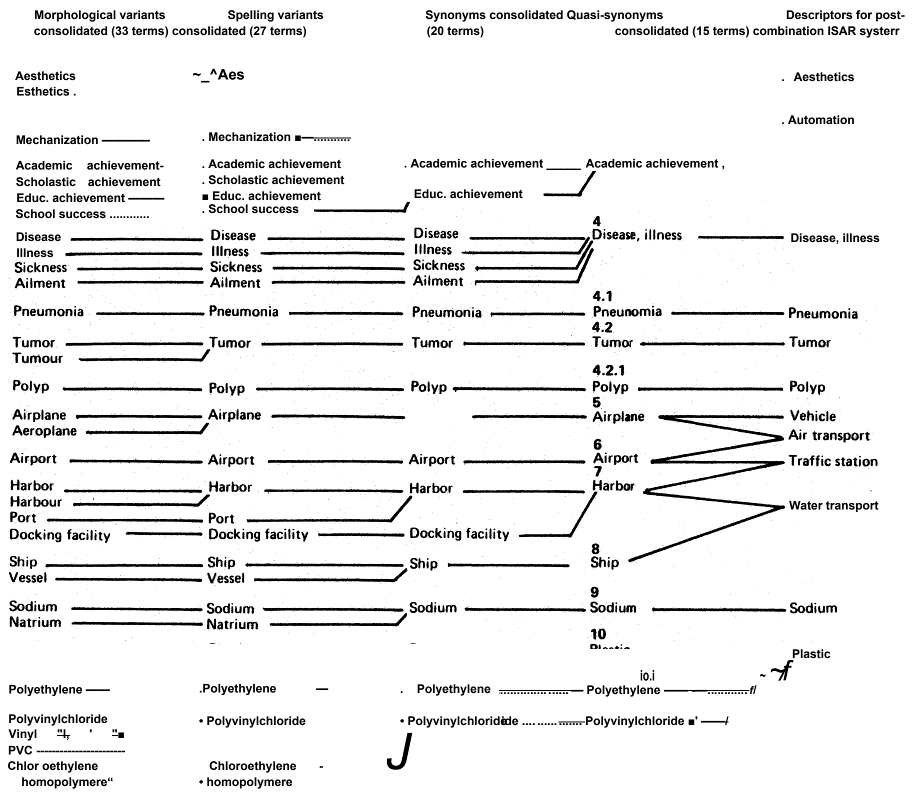
Harbor OR Harbors OR Harbour OR Harbours OR Port OR Ports.

Academic achievement 3	Mechanisation 2
Aeroplane 5	Mechanised 2
Aeroplanes 5	Mechanization 2
Aesthetics 1	Mechanized 2
Ailment 4	Natrium 9
Airplane 5	Plastic 10
Airplanes 5	Plastics 10
Airport 6	Pneumonia 4.1
Airports 6	Polyethylene 10.1
Automated 2	Polyp 4.2.1
Automatic 2	Polyvinylchloride 10.
Automation 2	Port 7
Chloroethylene	Ports 7
homopolymer 10.2	PVC 10.2
Disease 4	Scholastic achievement
Diseases 4	School success 3
Docking facilities 7	Ship 8
Docking facility 7	Ships 8
Educational achievement 3	Sick 4
Esthetics 1	Sickness 4
Harbor 7	Sodium 9
Harbors 7	Tumor 4.2
Harbour 7	Tumors 4.2
Harbours 7	Tumour 4.2
ill 4	Tumours 4.2
Illness 4	Vessel 8
Illnesses 4	Vinyl 10.2

Fig. 12.1 Terminological control and its uses: (a) Terms from an uncontrolled vocabulary Also index to Fig. 12.1b (53 terms).

The very richness and variety of the terminology used in natural language causes problems in retrieval. Overcoming these problems requires *terminological control* either in indexing or in searching. A data base using an *uncontrolled vocabulary* requires *terminological control in searching* through query term expansion; a query term is replaced by an OR-combination of the query term itself with any morphological and spelling variants, synonyms, and quasi-synonyms. This is a challenge for the searcher: When searching for *Aesthetics,* would you always remember to include "OR *Esthetics*^{99}*! When searching for *Illness,* would you always remember to include

Illness OR Disease OR Sickness OR Ailment?



¹ Reduction would be greater if, for example, all types of vehicles and traffic facilities were listed in column 4.

Fig. 12.1b Stepwise reduction of a set of terms.

Applying terminological control effectively requires a search aid (Fig. 12.1). An alphabetical index (Fig. 12.1a) leads from *Illness* to *Disease, illness* (column 4 of Fig. 12.1b). *(Disease, illness* is one term to designate a concept that includes anything that might be called *Disease, Illness, Sickness,* or *Ailment.)* Following the lines from right to left, the searcher finds in column 1 all the terms and spelling variants to use.

Terminological control in searching is feasible in a computerized search system, particularly if query term expansion is done through the search program, thus taking the burden off the searcher. (Whether free-text searching is desirable is quite another matter; see Chapter 13). In a manual search system, such as a card catalog or printed index, looking under many synonyms and quasi-synonyms is time consuming. Entities or references to entities related to one concept should be collocated in one place in the file, not scattered in many places. Thus they must be filed under *one* term or other concept identifier. This unique concept identifier may be created from scratch or selected from the spelling variants, synonyms, and quasi-synonyms referring to the concept. This is *terminological control in indexing* by using a *controlled vocabulary.* From any term in column 1 the indexer is led to the corresponding descriptor in column 4. The same is true for the searcher. Thus the structure of Fig. 12.1b ensures that indexers always use the same terms for a given concept and that searchers use that same term.

To sum up: with terminological control in searching, the structure of Fig. 12.1b is used from right to left for query term expansion. With terminological control in indexing, the structure is used from left to right for index term consolidation; the one query term found is sufficient to retrieve all entities relevant to a given concept.

Homonyms present a further—and less tractable—problem. The term *Port* used in free-text searching may return also documents on *Telecommunication ports* and on *Port wine.* The following example made the news: Somebody was searching a newspaper data base for reports on *Bats in Texas.* Using the query formulation *Bat* * AND *Texas* (the * signifies any term starting with the three-letter sequence *Bat)*$_r$ he found articles on the *Batting averages of Texas baseball teams.* In a system with an uncontrolled vocabulary a search program with access to a sophisticated search thesaurus can use context to determine the applicable meaning of a homonym. A controlled vocabulary contains a unique term for each meaning of a homonym.

Finally, hierarchy is very important for high recall; hierarchical relationships between concepts are shown in Fig. 12.1b, columns 4 and 5. When looking for *Plastic,* a searcher should also consider the narrower terms *Polyethylene* and *Polyvinylchloride,* including, if the vocabulary is uncontrolled, also its spelling variants *Vinyl* and *PVC* and its synonym *Chloroethylene homopolymer.*

As a last example consider a search for *Disease* AND *Employment* in an ISAR system with an uncontrolled vocabulary. The *Disease* component must be expressed broadly by combining the following terms with OR:

> *Disease* and its plural *Diseases,* its equivalent terms *Illness, Sickness, Ailment* (each with its plural), and *Sick* (a morphological variant of *Sickness),* and all the narrower terms of *Disease,* such as *Pneumonia* and *Tumor* (with *Tumour)* and, under that, *Polyp* (including plurals).

The *Employment* component should be expressed as

Employment OK Employed OR *Employee OK Occupation* OR *Occupational* OR *Job* OR *Work,*

including plurals and such forms as *Job-related*.

A computerized ISAR system using a controlled vocabulary requires only elemental descriptors because it can search for descriptor combinations; for example, to search for documents on *Harbor,* one would use the query formulation

Traffic station AND Water transport.

Sometimes it is useful to avoid very specific concepts—such as *Polyethylene*—as descriptors and use a broader concept—such as *Plastic*—instead in order to keep the index language simple.

So far this discussion has concentrated on the use of a vocabulary structure in searching and indexing. But such a structure must be created before it can be used. Such an effort starts from a set of terms collected from query statements; document titles, abstracts, or full texts; position descriptions; and similar free-text entity representations; such a set is shown in Fig. 12.1a. The initial set of terms—5,000 for a small-to-medium size thesaurus, 50,000 for a large one—is rather unwieldy and does not allow for the kind of easy overview that is essential for developing a logical conceptual structure. Fortunately, the large set of terms can be reduced to a much smaller and more manageable set of concepts without loss of content through the following steps:

- Consolidate singular/plural and other morphological and spelling variants (Fig. 12.1a to 12.1b, column 1 and then column 2);
- Consolidate synonyms (column 2 to column 3);
- Consolidate quasi-synonyms (column 3 to column 4).

Once the vagaries of terminology are dealt with, the designer can concentrate on the conceptual structure and, for a controlled vocabulary system, descriptor selection.

12.1 CONCEPTS VERSUS TERMS: THE SYNONYM-HOMONYM STRUCTURE

One must carefully distinguish between the plane of concepts and the plane of terms (or other concept identifiers), lest confusion reign. The relationships between concepts and terms (or other concept identifiers, such as class numbers and mathematical or pictorial symbols) are muddy at best; there is no one-to-one relationship. Often several terms designate the same concept; such terms are called *synonyms*. Some examples are *Natrium* and *Sodium, Lawyer* and *Attorney, Fixed in concentration* and *In exchange capacity* (both designating the same state of an ion in chemistry), and *Placed under government ownership* and *Nationalization*. Morphological and spelling variants add to the multiple term forms all designating the same concept. On the other hand, often one term has several meanings; such a term is called a *homonym.* Some examples are *Port-1 (harbor), Port-2 (telecommunication)^* and *Port-3 (wine) ox Socialization-! (economics)* and *Socialization-2 (social psychology).*

To ensure retrieval of all (or most) entities relevant to a concept requires *terminological control.* For this purpose, the synonym-homonym structure establishes a one-to-one correspondence between concepts and terms. The designer develops this structure by selecting or creating a preferred term or other concept designation from a group of synonyms (and quasi-synonyms) and disambiguating homonyms by a number and/or a parenthetical qualifier. (If there is only one term for a concept, that is the preferred term.)

The preferred term serves as the focal point where all information about a concept is collected, providing a basis for intelligent decisions in thesaurus development. For example, the entry under *Harbor* (preferred term with *Port* as nonpreferred synonym) gives the information that the concept can be expressed as a combination *(Trafficstation* AND *Water transport),* or that it occurred in 43 queries (term *Harbor* 25, term *Port* 18). This information is not repeated under *Port.* The result is a file that is both smaller (perhaps 2000 concepts versus 5000 original terms) and draws all information about a concept together in one place. Ina system with an uncontrolled vocabulary the preferred term for a concept serves as a focal point from which the searcher is led to other terms designating the concept and to broader, narrower, and related concepts. In a system with a controlled vocabulary the thesaurus developer must decide which concepts should be used as descriptors. This decision should be made only once for each concept (not first for *Harbor* and then again for *Port),* and it should use all available information about the concept. If frequency of occurrence is used as a criterion in descriptor selection, one should establish the *concept* frequency (as opposed to *term* frequency; see the preceding example). This discussion illustrates that selecting preferred terms is important for any type of system, not just for systems using a controlled vocabulary. Also, with a controlled vocabulary a preferred term is not always a descriptor, but rather a descriptor candidate.

12 2 GROUPING CLOSELY RELATED CONCEPTS: THE EQUIVALENCE STRUCTURE

There are some true synonyms. These are terms completely equal in meaning, such as different terms for the same concrete object or process, or different names for the same chemical substance. But usually differences in language indicate subtle differences in meaning, connotation, or affective value—for example, *Developing countries* and *Underdeveloped countries.* These terms designate closely related or widely overlapping concepts; such concepts can be grouped in a class of *equivalent concepts.* The corresponding terms are called *equivalent terms* or *quasi-synonyms.* For example,

Disease, Illness, Sickness, and Ailment

For developing the conceptual structure and for indexing and retrieval, it is useful to form a new broader concept that contains all of the original concepts. The thesaurus builder must choose a term or notation to identify the new concept, perhaps creating a new term, such as *Disease, illness* (one term).

Equivalency occupies a middle ground between synonymy and hierarchy . For purposes of thesaurus development, particularly for constructing the overall hierarchy, it may be convenient to disregard small differences in meaning and consider only the newly formed concept, such as *Disease, illness,* thus further reducing the number of concepts to be considered. In an individual thesaurus it is often sufficient to treat equivalent terms as synonyms. For example, *Disease, Illness, Sickness,* and *Ailment* may be treated as synonyms of the new term *Disease, illness.* The four individual concepts lose their conceptual identity and are completely absorbed in the new concept. A thesaurus entry might be

Sickness USE ST Disease, illness (ST = Synonymous Term) or
Sickness USE ET Disease, illness (ET = Equivalent Term)

In a specialized thesaurus one may want to preserve the individual concepts, perhaps even use them as descriptors:

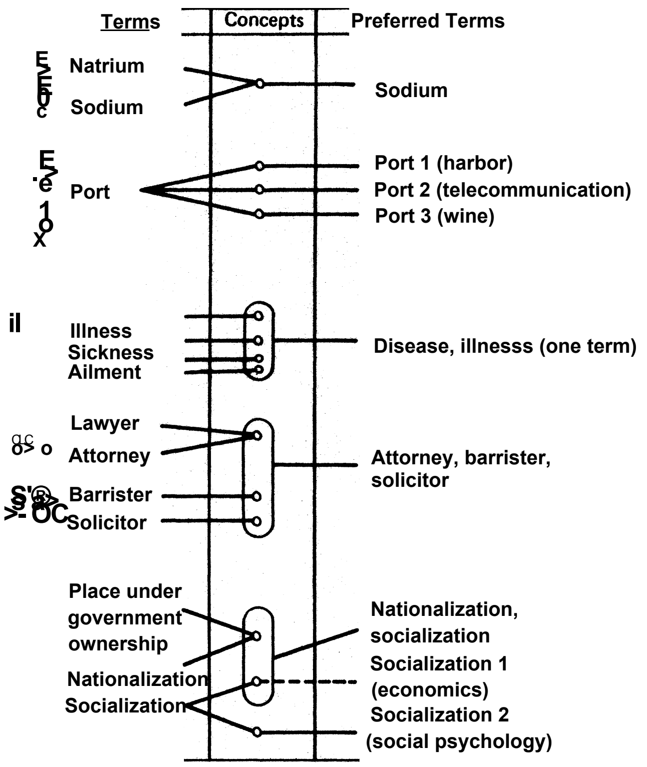
Sickness BT Disease, illness (BT = Broader Term)
(Sickness is a separate descriptor)

In a thesaurus data base, one should introduce a special relationship BT-EQ; for example

Sickness BT-EQ Disease, illness

The builder of an individual thesaurus using the data base can then decide in each case how to treat this relationship.

Figure 12.2 shows examples of synonyms, quasi-synonyms, and homonyms.


Terms
Concepts
Preferred Terms
Natrium
Sodium
Sodium
Port
Port 1 (harbor)
Port 2 (telecommunication)
Port 3 (wine)
Illness
Sickness
Ailment
Disease, illnesss (one term)
Lawyer
Attorney
Barrister
Solicitor
Attorney, barrister, solicitor
Place under government ownership
Nationalization
Socialization
Nationalization, socialization
Socialization 1 (economics)
Socialization 2 (social psychology)

Fig, 12.2 Synonyms, quasi-synonyms, and homographs.

12.3 CLASSIFICATORY STRUCTURE

The set of all concepts as represented by the preferred terms, together with the relationships among these concepts, is called the *classificatory structure* (our term). The main relationships in the classificatory structure are

BT Broader Term (really Broader Concept)
NT Narrower Term
RT Related Term

12.4 INDEX LANGUAGE

An index language is the set of subject descriptors used in an ISAR system. A *descriptor {broadly defined)* is any entity identifier, such as a person identifier, which is used for recording relationships to other entities and thus can be used for retrieval. A *subject descriptor* is an identifier for a subject or concept that can be used in retrieval. Following common usage, we will use the term *descriptor* with the meaning of subject descriptor unless stated otherwise.

In a system with an uncontrolled vocabulary, a descriptor can be any term. (Any term is admissible in the index language.) However , in the remainder of the book we shall assume a controlled vocabulary (terminological control in indexing) unless specified otherwise. A subject descriptor then designates unequivocally a concept used for indexing and retrieval; subject descriptors are selected from the pool of preferred terms (every subject descriptor is a preferred term but not vice versa). Most commonly the term *descriptor* is used with these connotations.

Likewise, the term *index language* in the broad definition given here includes natural language as used in an uncontrolled vocabulary. However, index language is commonly understood to imply a controlled vocabulary; we adhere to this common usage unless stated otherwise. The same is true for the terms *system vocabulary and classification (classification scheme),* which are quasi-synonyms of index language.

Examples of descriptors are *Pipe; Form of government; Theory; France; Graduate-level text;* 635.652 *Kidney beans* (as agricultural product), DDC; HD9235.B4 *Beans* (as agricultural product), LCC; NQCL.MACH.003 *Reactor,* a descriptor produced by combining two components and a serial number in the Semantic Code system; and A57B34C10D45 *Kidney bean Kernels Whole Canned,* a descriptor produced by combining four components, each from a different facet.

In the last four examples the concept is identified both by a notation and by a term. Which is used to record relationships with other entities depends on the system using the index language. The essential point is that the concept is available for retrieval, no matter how it is identified. Sometimes one distinguishes *descriptor text* (e.g., Kidney beans) and *descriptor notation* (e.g. 635.652).

Some index languages list all descriptors that can be used. Others allow for the building of compound descriptors from building blocks as in the preceding canned beans example. Sometimes a special type of descriptors, called role indicators, is used to produce compound descriptors:

Insulin— *Therapeutic use;* Diabetes—*therapy*
Noise—*Effect;* Children—*Cause*

In some index languages these may be further combined to form still more compound structures to be used as descriptors, for example,

(Noise—*Effect):* (Children—*Cause)* (one descriptor)
[{(Noise—*Effect):* (Aircraft—*Cause)]—Agenti:*
(Children—*Affected)* (one descriptor)

Relators serve the same function but have a different format:

Noise *Caused by* Children
(Noise *Caused by* Aircraft) *Acting on* Children

Role indicators and relators are syntactical elements. Not all index languages have a developed syntax.

12.5 THESAURUS

A thesaurus is an aid for searching and—in a controlled vocabulary system—for indexing. It gives relationships among concepts, between concepts and terms, and among terms.

In an ISAR system using a *controlled vocabulary,* the index language consists of descriptors selected from the pool of preferred terms. For the benefit of indexers and searchers a thesaurus also includes the remaining preferred terms and the nonpreferred terms as *lead-in terms,* which form the *lead-in vocabulary*. Thus the indexer or searcher need not guess the term used as descriptor; looking under any term that comes to mind leads to the descriptors) to be used (Fig. 12.1, using the index 12. la and then going from left to right in 12.1b). *Natrium* (nonpreferred lead-in term) leads to the synonymous descriptor *Sodium; Polyethylene* (preferred lead-in term) leads to the broader descriptor *Plastic*. The following table illustrates a more complex case:

Thesaurus builder's working file	User version
Port SEE ST Harbor	Port SEE ST Harbor USE BT :Traffic station : Water transport
Harbor ST Port USE BT :Traffic station : Water transport	Harbor ST Port USE BT :Traffic station : Water transport

In summary, a thesaurus for an ISAR system using a controlled vocabulary consists of an index language and a lead-in vocabulary. The index language consists of all the descriptors and the relationships among descriptors. The lead-in vocabulary consists of all lead-in terms and the relationships among them, and it leads from each lead-in term to the appropriate descriptor(s) to be used, possibly specifying the nature of the relationship between lead-in term and descriptor(s). This specific definition of thesaurus is illustrated in Fig. 12.3. Figure 12.4 further clarifies the distinction between nonpreferred terms, preferred lead-in terms, and descriptors.

A simple thesaurus for an ISAR system using an *uncontrolled vocabulary*

Synonym-homonym structure	Relates terms to concepts; preferred terms for 1-1 correspondence.
Equivalence structure (quasi-synonyms or equivalent terms)	Groups together closely related concepts resulting in new broader concepts.
Classificatory structure	Concepts as expressed by preferred terms, and their interrelationships.
Index language (system vocabulary, classification scheme)	Descriptors; selected preferred terms actuaUy used in entity representations and query formulations.

<u>Thesaurus</u> (Controlled vocabulary systent)

All terms

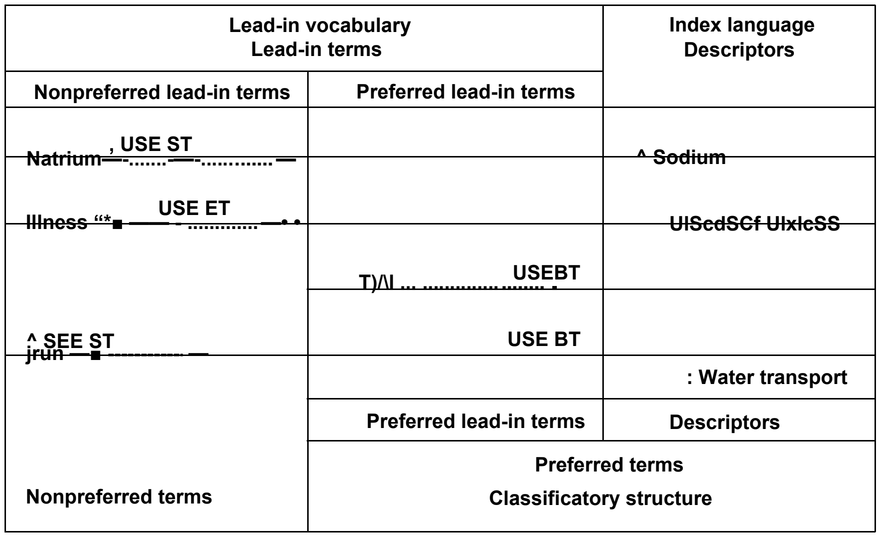


Fig. 12.3 Index language and thesaurus: definitions.

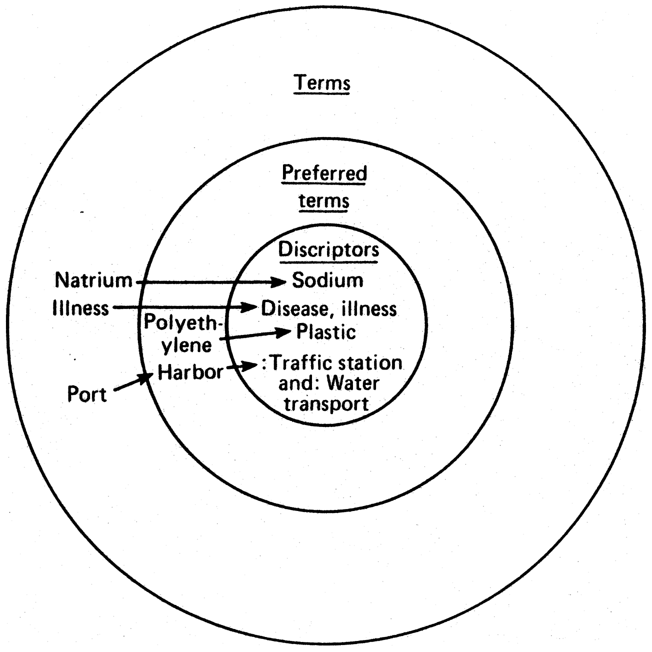


Fig. 12.4 All terms, preferred terms, descriptors.

consists of a classificatory structure and a set of further search terms. The concepts (preferred terms) in the classificatory structure lead to further search terms. Essentially , the structure of a thesaurus is the same whether it is used for a controlled vocabulary or an uncontrolled vocabulary ISAR system, except that with an uncontrolled vocabulary there is no need to select descriptors from the preferred terms. But the *use* of the thesaurus is different (see Fig. 12.1b). With a controlled vocabulary, the thesaurus structure is used from left to right for index term consolidation; with an uncontrolled vocabulary, the thesaurus structure is used from right to left for query term expansion. More complex thesaurus structures do not select preferred terms but link terms directly to each other, specifying the type and/or the strength of the relationship.

A computer can be programmed to look up terms from query statements, organization descriptions, documents, and the like, and convert them to the corresponding descriptors or expand a query term. A computer program needs a very complete thesaurus, since unlike humans it cannot find other terms under which to look if a term is not found. Even minor morphological and spelling variants must be included unless the program can recognize their relationship to the base form.

CHAPTER 13

Index Language Functions

OBJECTIVE

The objective of this chapter is to enable the reader to make intelligent decisions about the type of index language, indexing, and query formulation to be used in a given information system, considering costs and benefits of each alternative. This requires thorough understanding of the role of the index language in the 1SAR system, particularly its role as a communication device between user and indexer, and of the concept of request-oriented indexing.

13.1 REVIEW: THE INFORMATION RETRIEVAL PROBLEM

The retrieval problem is to find entities relevant to a query statement. We begin this chapter by summarizing the solutions discussed in Section 5.2. (We assume an ISAR system with one main entity type, for example *Document,* and we emphasize subject searching in the discussion.)

The most *direct solution* is to *search the whole* (*unorganized*) *collection*. During the search process, the query statement—the problem for which information or entities are needed—is foremost in the searcher's mind and provides a frame of reference for analyzing entities. The process is request- or problem-oriented. The searcher examines entities from the user's point of view, thus serving as the user's agent. This process is truly user-oriented, but it needs to be organized more efficiently.

A *first economy measure* is to *cut perusal time by preparing entity representations*. But this creates a problem: What information should be included in the entity representations so that they form a sufficient basis for judging relevance?

A *second economy measure* is to *batch queries* and search the entire collection from the point of view of several queries simultaneously . This causes another problem: The searcher must keep in mind many query statements at once and analyze each entity from the point of view of each of those query statements.

A *third economy measure* combines the previous two by *collecting anticipated queries and analyzing entities in advance*. For example, a reference librarian in a small library may keep an interest profile for each member of the clientele to be served, analyze incoming entities with respect to these interest profiles, and route them to users accordingly. This is a logical development of the method of batching queries. It is also a logical development of the method of preparing representations; the problem of which viewpoints to consider in preparing the representations is now solved: The anticipated queries provide the necessary guidance.

Some query statements are descriptors unto themselves, for example,

> I need to know about anything that endangers our business.

However, as a rule, a query statement combines several components, each of which is a separate descriptor. For example,

> New technological developments that endanger our business.
> Components: *Technology* AND *New developments* AND *Danger to our business*
> I need material on the corrosion of steel pipes.
> Components: *Corrosion* AND *Steel* AND *Pipes*

Entities are analyzed with respect to these components.

Lastly, *retrieval can be aided by providing a retrieval mechanism,* such as a computer search system, a printed index, or shelf arrangement by subject.

All methods discussed serve to provide more efficient organization for the basic retrieval process, which consists of looking at an entity and deciding whether or not it is relevant for a given query. In this process the query (the problem of the user) is the primary concern, and the entities are examined in light of this query. This might be obscured in the method of anticipated queries, where, for every incoming entity, the indexer looks at a list of queries. The indexer must be very careful to retain the basic purpose of the process in this case, too. He or she must internalize the conceptual framework derived from the anticipated queries, examine each entity (e.g., read and understand a document, find out about a course, analyze an organization) from the point of view of this framework (that is, examine the entity from the point of view of each of the queries listed), and for each make a judgment as to whether the entity is relevant.

Indexing must be request-oriented if is to be a full substitute for the elementary process of judging relevance from the point of view of a query. The indexer must judge relevance with respect to anticipated queries. The following section discusses in detail the method of request-oriented indexing and how it can be implemented through the checklist technique of indexing.

13 2 THE ROLE OF THE INDEX LANGUAGE IN INDEXING

Request-oriented indexing (problem-oriented indexing) is a logical development of the method of anticipated queries. It contrasts with entity-oriented indexing, where the focus is on the entity and its faithful description. Entity-oriented indexing is the approach most commonly used; many writers do not even consider alternatives. However, the preceding examination of the retrieval problem shows that request-oriented indexing promises improved performance, possibly at higher cost. The ISAR system designer should consider both approaches.

This section expounds request-oriented indexing so that the reader can (1) consider the full range of alternatives in ISAR system design and (2) appreciate the shortcomings that arise in existing systems using entity-oriented indexing and adjust search strategy accordingly.

13.2.1 Disadvantages of Entity-Oriented Indexing

The shortcomings of entity-oriented indexing can be seen most clearly in a system using *shelf arrangement as the retrieval mechanism.* The original idea behind shelf arrangement is to collocate entities relevant to the same anticipated query in one place to produce a request-oriented arrangement. Unfortunately, while many entities are relevant for several anticipated queries, an entity can be shelved in one and only one place. The indexer confronted with this problem easily falls into the trap of considering the arrangement of entities on the shelves as an end in itself, forgetting the ultimate purpose, retrieval. He or she concentrates on the entities and forgets about the queries. This entity-oriented approach has negative consequences for the design of the index language and for the process of indexing.

Design of the Index Language for Shelf Arrangement

The requirements of shelf arrangement determine the selection of descriptors, their arrangement, and the indication of relationships between them. Shelf arrangement requires suitable headings for groups of entities in the collection, pigeon holes into which entities will fit neatly. When several viewpoints can be used for the subdivision of a group of entities into narrower groups, the designer of the index language must choose one. For example, consider a group of documents on *Physics;* it could be subdivided by the following criteria:

- *by Quality:* Good, Medium, Bad
- *by Level of treatment:* Elementary school, High school, Undergraduate, Graduate
- *by Type of document:* Textbook, Journal article, Research report, and so on
- *by Subtopics of physics:* Mechanics, Acoustics, Electromagnetism , Optics, and so on .

If the subdivision by *Subtopic of physics* is chosen and if this leads to reasonably sized groups in the collection at hand, the other viewpoints are not considered in the index language; one cannot then search by *Type of document, Level of treatment,* or *Quality.* The designer could decide to further subdivide the documents within one area of Physics (e.g., *Optics)* by *Type of document.* The point is that this decision is not based on the importance of the aspect *Type of document* for searching but rather on its usefulness for defining the arrangement of the documents on the shelves. Very general concepts, such as *Threshold* or *Effects,* and very specific concepts that are not helpful for arrangement are omitted from the index language, although they are very useful for the formulation of queries. Furthermore, a large number of hierarchical relationships is needed for complete searching, but only some of these are selected for shelf arrangement, resulting in an artificial monohierarchical structure.

The Process of Indexing for Shelf Arrangement

In a system for shelf arrangement, indexing an entity by a descriptor is tantamount to assigning the entity to a group or class of entities. Only one descriptor must be used for an entity. This one descriptor is the entity representation, albeit a very incomplete one. It leaves out many aspects for which the entity is relevant.

In reality there are often three or four classes for which the entity is relevant and which, therefore, should be considered. The indexer must choose the optimal alternative, the one that would result in the highest benefit derived from retrieval and use of the entity. However, the indexer is usually under time pressure when looking for the proper class for an entity; as soon as he or she has found a class into which the entity fits reasonably well, the indexer is satisfied and does not look for other alternatives. Thus, the indexer often uses satisficing (stopping as soon as a satisfactory alternative is found) rather than optimizing (finding all alternatives, then selecting the best one). The indexer considers finding a place for an entity on the shelf as an end in itself, rather than as a means for later retrieval. This approach is heavily entity-oriented rather than request-oriented.

The entity-oriented attitude induced by the requirements of shelf arrangement often carries over to systems with much more flexible search devices, even those using computer searching, in which the technical constraints oi shelf arrangement no longer exist. For example, indexing in ERIC (Educational Resources Information Center) is based on the premise that the document is primary and that one must faithfully express the contents of the document by an appropriate combination of descriptors. The negative consequences are much the same as those discussed for shelf arrangement.

The Design of the Index Language with Entity Orientation

Concepts that might be very important for searching but do not suggest themselves from the entities are not included in the index language. For example, in a bibliographic index language developed from words in titles, one would hardly expect to find descriptors such as

Important for long range transportation planning
Danger to our business
Read immediately

The Process of Indexing with Entity Orientation

Entity-oriented indexing is carried out in two steps:

1. Identification of indexable matter: Examine the entity (read a document or abstract, look carefully at a painting, or analyze a food sample) and note subjects (For a document: What is it about?).
2. Translation into the index language: Using the terms noted in step 1, consult an alphabetically arranged thesaurus to find the descriptor(s) to be used.

As soon as the indexer has found a descriptor for every subject he has identified—that is, as soon as the indexer is satisfied that the entity is well represented through the descriptors assigned—the indexing task is considered finished, no matter how many other descriptors there might be under which the entity should be found.

An extreme form of entity-oriented indexing applied to documents is the *extraction-and-translation method of indexing,* in which the indexer simply underlines important terms in the text of the document or in an abstract and then translates these terms into the index language. The translation step can easily be automated. To a lesser extent this is true of the extraction step as well. In this method only concepts that are represented by a term in the text

1. A document or data item dealing with the
Percentage of children of blue-collar workers attending college.

Document-oriented indexing:
Blue-collar workers (social class); College attendance
Request-oriented indexing would add
Intergenerational social mobility.

For a sociologist this may be the most important descriptor under which this document should be found. The indexer acts as the sociologist's agent in reading the document or abstract and should, as would the sociologist, recognize that the document is relevant for *Intergenerational social mobility* ("scientific prethinking"). This requires that the indexer know the concepts of interest to the user and that the indexer know the field well so that he or she can recognize relevance even if it is not obvious.

2. Consider the topic
Treaty on Nonproliferation of Nuclear Weapons.
A thorny issue in negotiating this treaty was the inspection problem. Participants had to agree to inspection of their nuclear power plants by citizens from other countries, which is an infringement on sovereignty and furthermore provides an opportunity for industrial espionage. An indexer comes across a document that deals with the U.S. threat to Swiss pharmaceutical companies to ban their products from the U.S. market unless they allow inspections of their plants by Food and Drug Administration personnel. Document-oriented indexing would assign the descriptors *Switzerland*, *United States, International trade, Drugs (pharmaceutical),* and, perhaps, *Inspection.* But since this document deals with the problem of inspection by foreign nationals, it is highly relevant for somebody working on the *Nonproliferation Treaty* and should be indexed by that descriptor. Even better, introduce a descriptor *Inspection by foreign nationals* (inspired by the analysis of information needs) so that the concept can be expressed directly in indexing and searching.

Fig. 13.1 Entity- vs. request-oriented indexing. Examples.

are considered; other concepts suggested by the overall context of the document are neglected, although they might be clearly needed for a representation of the content. The method discussed first, while focused on the document and the representation of its content, at least allows the indexer to note such concepts.

13.2.2 Request-Oriented Indexing: General Approach

Request-oriented indexing has a different focus. The indexer asks: Under which descriptors should this entity be found? The examples in Fig. 13.1 show the difference. In request-oriented indexing the queries are primary and the entities are analyzed with a view to queries (see the examples in Fig. 13.2). Ideally, the indexer would think of all the possible queries and decide for which ones the entity at hand is relevant. A systematic procedure for both

1. In the *ISAR system of a company* the following descriptors are very useful:

Danger to our business
Technological developments that could improve our products
Market gaps for our products.

If competent indexers analyze entities from the point of view of these descriptors and make good relevance judgments, then management personnel responsible for planning and research and development can be provided with information important for their task. For example, an article on integrated circuit technology published 15 years ago should have been indexed under the first descriptor by an indexer working for a slide rule company because it foreshadowed cheap pocket calculators, which make slide rules obsolete. Management, made aware of these developments, could take proper planning steps (get out of the business of manufacturing slide rules and perhaps start developing pocket calculators). If management was not made aware of these developments, the company might well have gone out of business.

2. The purpose of an *ISAR system for curriculum development* is the retrieval of topics and instructional materials that contribute to specified educational objectives. (As an example, consider the argument that the study of Latin develops a sharpened sense of language and logical thinking.) Thus the entities sought are topics and instructional materials, and each of these must be indexed by the educational objectives it serves. But how could this be done without drawing up a list of all educational objectives in the ßrst place? Such a list will be hierarchically structured because there are objectives, subobjectives, sub-subobjectives, and so on.

3. A *curator in the Department of Anthropology* of the Smithsonian Institution is very much interested in any pictorial representation of Indians, however minor a part of a painting, drawing, or print it might be. Introducing the descriptor *Contains picture of Indians* communicates this interest to all staff members indexing pictures and thus instructs them to watch out for pictures of Indians. The curator can save a lot of work by simply using the new descriptor for retrieval.

Fig. 13.2 Request-oriented indexing. Examples.

the design of the index language and the process of indexing makes this feasible.

The Design of the Index Language with Request Orientation

Development of the index language starts from a list of anticipated queries, that is, statements of need, which in turn are derived from an analysis of the problem situation of the users. As a result, the index language contains a descriptor for any concept (viewpoint, aspect) occurring in anticipated queries, such that any anticipated query can be expressed by a combination of descriptors. Usefulness for searching is the primary criterion in descriptor selection. The descriptors are arranged in a meaningful hierarchy that communicates the user's conceptual framework to the indexer.

The Process of Indexing with Request Orientation

The index language thus derived from anticipated queries communicates to the indexer the user's needs; it establishes the framework for the analysis of entities. Request-oriented indexing proceeds primarily from the index language, not from the entities. The indexer assimilates the conceptual framework provided by the index language and examines the entity at hand with that framework in mind. He or she then considers each descriptor in the index language and makes a relevance judgment: Should the entity be retrieved under the descriptor? Through this procedure all aspects for which the entity is relevant are elicited. The index language serves as a checklist in indexing.

It could be argued that request-oriented indexing is not possible because one cannot anticipate *all* future needs or *all* descriptors that will be needed for expressing future queries. Nonetheless one *can* anticipate many future needs by considering all possible sources of data, including technological forecasts, and analyzing these data; this is better than nothing. Additional entity-oriented indexing can add descriptors that might prove useful later although a need for them was not anticipated (see Section 13.2.4). Ingenious searching can sometimes compensate at least partially for the omission of a descriptor from the index language (see Section 13.3). In other cases it may just not be possible to respond to an unforeseen request because it would involve examining a large number of entities.

Truly request-oriented indexing requires that the index language be adapted to the specific needs of the clientele to be served. While the index language designer need not know the interests or needs of any specific user, he must know the total "pool" of all interests or needs. Moreover, he must arrange them in a framework that is manageable by indexers. This is difficult in information systems that are geared to a very broad audience, such as *Chemical Abstracts* or the *Reader's Guide.* The number of interests may simply be too large. But request-oriented indexing can still be useful: Viewpoints such as reading level may be useful for the entire audience. Furthermore, the audience may consist of a number of distinct groups; in that case a separate indexer could be assigned to deal with documents of interest for one group using an index "sublanguage" adapted to the specific needs of that group. All of this requires a thorough study of user needs, using a representative sample of the clientele to be served.

The set of descriptors assigned to an entity through request-oriented indexing using a well-structured index language is biased by the anticipated queries represented in the index language. Many authorities interpret this as a bias imposed on the user by the system designer and conclude that hierarchically structured index languages and request-oriented indexing using the checklist technique should not be applied at all; instead they recommend relying solely on a faithful description of the entity at hand (e.g., the contents of a document), preferably in the language of the document itself. However, this conclusion is not warranted: Request-oriented indexing identifies important retrieval clues that would not be assigned in the usual entity-oriented approach to indexing. Indeed, the retrieval clues derived through request-oriented indexing are the more important dues. The indexing bias is imposed not by the system designer but by user interests uncovered in a study of needs; this is a desirable bias. Moreover, descriptors can be added through supplementary entity-oriented indexing as discussed in Section 13.2.4.

13.2.3 Request-Oriented Indexing: Implementation

This discussion leads to a procedure for request-oriented indexing: First learn and comprehend the conceptual framework provided in the index language. Then index each entity in two steps:

Step 1: Examine the entity, paying particular attention to the aspects covered in the index language.
Step 2: Decide for each descriptor in turn whether or not the entity is relevant—whether or not the entity should be retrieved under this descriptor (remember that each descriptor is a component of an anticipated query).

The descriptors should be printed on an indexing form. Ideally, the indexer should check "yes" or "no" for every descriptor in the index language; in reality, one makes do with checking only "yes" explicitly, assuming that no check means "no" (but it could be an oversight). The list of descriptors serves as a checklist for the indexer. We call this method *checklist technique* of indexing.

The checklist technique reduces the task of judging relevance with respect to all possible anticipated queries to the task of judging relevance with respect to a much smaller number of descriptors. Unfortunately, looking at 1000 or 5000 or even more descriptors for every entity to be indexed is still not a feasible task. However, request-oriented indexing can be made feasible by combining three techniques.

Selection of Checklist Descriptors

Select a limited number of descriptors that are particularly important to the organization setting up the ISAR system; these *checklist descriptors* receive special attention in indexing. The regular indexer must learn them and have them in mind so that he or she goes through them more or less subconsciously while indexing; an indexer can do this for up to 1000 descriptors. Proper display of the index language (or of the subset of checklist descriptors) aids in this task, as discussed in the following section. The not-so-regular indexer must rely on a well-structured display and more explicit checking of descriptors.

Display of the Index Language

The index language should be presented in a classified structure that is both conceptually and graphically well designed. Many index languages can be displayed on a few pages in such a way that the overall structure and the relationships between the descriptors become clear at a glance. It is known from learning theory that a meaningfully structured body of text can be learned and remembered more easily than a nonstructured body of text. A list of words can be remembered more easily than a list of nonsense syllables; a list of words arranged in a meaningful sequence (using a principle that is easily recognized by the learner) can be learned more easily than a list of words arranged in random order. Since the indexer should internalize the user's framework represented in the index language, classified structure is very important. A good classified display also refreshes the indexer's memory and leads him or her to the appropriate descriptors, thus making sure that no important aspects of the entity are overlooked. This is likely to enhance indexing consistency. The classified structure also assists a searcher in formulating the query (see Section 13.3).

Stepwise Refinement

The number of descriptor-entity comparisons can be reduced drastically by employing a top-down approach: The set of descriptors is subdivided into (usually overlapping) subject fields in such a way that the indexer looking at the heading of a subject field can immediately decide whether there is an expectation that the entity is relevant to any of the descriptors contained in the subject field. Each subject field is divided into subfields, and so on.

As an example, let us consider indexing a document on the problem of *Prayers in Public schools.* Scanning a list of top level subject fields, the indexer marks, among others, *Education* and *Politics* for further scanning:

```
X Education
   Communication and language
   Society
X Politics
   International politics
   Law
```

Economics
Technology
Problems of developing nations
Sociocultural change
•

In this way she discards many of the subject fields and reduces the number of descriptors to be considered to a fraction of the index language. Within each subject field the indexer, using the same procedure again, scans the subfields, and so on.

Politics
  System of government
X State and organs of the state

  X Constitution

  Political process
  Internal politics
  Public administration

This process ensures that the indexer does not overlook the descriptor *Constitution*, which is important for indexing the document. Another indexer may have checked *Law* rather than *Politics.* That, too, would have led to *Constitution* through the chain

Law
  Public law
    Constitution

Thus *Constitution* has two broader concepts. An index language that explicitly provides for this situation is called *poly hierarchical {an* index language for shelf classification restricts each narrow topic to one broadei topic and is therefore called *monohierarchical).* The hierarchy developed foi the purpose of the checklist technique of indexing has several levels of sub division. It is complemented by cross-references among descriptor; associated in other ways. The resulting structure, a network of descriptors leads the indexer to the descriptors for which the entity at hand is relevant

The interests of the users determine the selection of checklist descriptors and also the division of the descriptors into fields and subfields (note the field *Sociocultural change* in the preceding example).

In a very crude form, request-oriented indexing can be reduced to a few indexing instructions such as:

> Index each chemical compound, its effects and applications
>
> OR
>
> Index each topic by the educational objectives to which it might contribute

This is better than nothing, but hardly sufficient.

The request-oriented approach applies also to preparing abstracts and similar entity representations. The same article is abstracted differently for *Chemical Abstracts* and *fat Biological Abstracts.* The abstracting rules of RINGDOC (a pharmaceutical abstracting/indexing service) reflect the problems of the pharmaceutical industry; they instruct the abstractor to emphasize the chemical structure, the manufacture, and the effects of drugs, giving a detailed list of points to watch out for. The major part of the abstract may thus summarize perhaps one-fourth of the document; the other three-fourths are of minor interest for RINGDOC users and are mentioned only very briefly.

Figure 13.3 shows an excerpt from the indexing form of the *Chemical Kinetics Index,* which illustrates the principles discussed. The most important descriptors are preprinted; the indexer need only check the applicable ones. There is room for entering other applicable descriptors. The MEDLARS indexing form displays for check-off a small number of descriptors of general application, called check tags, such as *Animal experiments, Human, Male,* and *Female.* (These descriptors are also displayed on the MEDLARS search form shown in Fig. 13.4.)

13.2.4 Supplementary Entity-Oriented Indexing

The bias introduced through pure request-oriented indexing may result in the loss of parts of the information that might be important for unexpected queries coming up in the future. For example, if objects have been indexed by material and geometrical form, a user with the unexpected query *Very large objects* is out of luck. The size of an object is an aspect that might suggest itself to an indexer working in an entity-oriented mode; it could have been added easily to the representation of the object. For another example, consider an ISAR system on transportation technology. Assume a study of needs shows that only the technical aspects of transportation systems are of

WHAT TYPE OF REACTION MECHANISM?

Addition: $AB + CD — E$
  Insertion
Charge-Transfer: $A^+ + B — A + B^+$
  $A^“ + B - A + B^~$
Collisional-F ragmentation:
  $A + B —' C + D + E$
(re) Combination: $A + B — A - B$
  (Bond formation)
Cyclization
Decyclization
Dissociation: $A\text{-}B — A + B$
  (Bond fission)
Elimination: $E — AB + CD$
Energy-Transfer: $A^* + B — A + B$
  (Coitisional)
Fluorescence
Four-Center-Exchange:
  $AB + CD - AC + BD$
Substitution: $A + BC — AB + C$
  (Atom-trans, abstract, disproport, metathesis)

GAS PHASE ION OR ELECTRON REACTION? (Combine with mechanism types at will)

Collisional-Ionization (neutrals or ions)
Electron-Attachment (to neutral)
Electron-Detachment (from negative ions)
Electron-Impact-Ionization:
  $A + e — A^+ + ne$
Electron-Impact-Excitation
Ion-Electron Combination
Ion-Molecule (reaction, chemical change)
Ion-Pair-Formatioo
Ion-Pair-Neutralization

THEORETICAL DEVELOPMENTS OR FIELD?

Absolute-Rate-Theory
Anharmonic-Oscillator
Calculation (num. results)
  Exact (treatment of model)
Collision-Dynamics
Detailed-Rates (master eq.)
Force-Constant
Harmonic-Oscillator
Hartree-Fock
Irreversible-Stat.-Mech.
Irreversible-Thermodynam.
Molecular-Orbital
Monte-Carlo
Perturbation-T reatment
  Time-Dependent
  Stationary
Phenomenological (macroscopic)
Rate Theory (miscellaneous)
Scattering
  Classical
  Semiclassical (WKB)
  Quantum-Mechanical
    Born-Approx.
    Born-Oppenheimer
    Distorted-Wave
    Resonance
Statistical-Mechanics (eq.)
Statistical-Rate-Theory (phase space)
Stochastic (fluctuations)
Trajectories
Unimolecular (rate theory)
Debye
Fermi

WAS A TECHNIQUE DEVELOPED. INTERPRETED, OR VERY IMPORTANT?

Acoustics (ultrasonic)
Actinometry
Analysis (chemical)
Calorimetry
Computer (simulation or instrumentation)
Cryogenics
Electric-Discharge
Field-Emission
Flash-Filament
Flash-Photolysis
Flow-Reactor
Interferometry
Laser
Mass-Spectrometer
Mass-Spectrometry
Molecular-Beam
Optics
Polarography
Pressure
Shock-Wave
Spectrometer
Spectrometry (frequency)
Time-of-Flight (mass, spec.)
Vacuum-TechniqiK
G as-C hromatography
Field-Ionization
Microscopy
M össbauer
Diffraction
Crystal-Growth

Fig. 13.3 Indexing form for use with the checklist technique. (Chemical Kinetics Index)

interest; hence the place where these transportation systems operate is not indexed. If a query involving place comes up unexpectedly, it cannot be answered. Or assume that the needs of National Science Foundation staff are well met by broad indexing of research projects and proposals. A user who, at a later time, wanted to use the NSF data base to search for research projects on specific topics would have to make do with low precision.

One can take out insurance against the contingency that a concept will be needed for searching later, even though such use is not anticipated now. This is achieved by indexing with additional descriptors suggested by the entity at hand; The premium is increased cost for indexing and for updating and maintaining the files. As with any insurance, one must weigh the risk (the likelihood that a concept will be needed in searching) against the premium (the cost for indexing with the concept). Supplementary entity-oriented indexing does not ensure that every request put to a system can be treated satisfactorily. If the concepts in the request were neither anticipated (and therefore not used in request-oriented indexing) nor suggested by the entities (and therefore not used in supplementary entity-oriented indexing), then search performance for that request will be low. For example, when indexing objects, one would not normally consider the aspects *Weight, Price, Place where made,* or *Is decorated* unless specifically instructed to do so. Entity-oriented indexing does not absolve the system designer from the task of anticipating queries by thoroughly studying needs.

In document retrieval the author's vocabulary—terms found in title, abstract, or text—provide useful retrieval clues in addition to the standardized descriptors. This is true particularly if the index language is not capable of expressing very specific concepts or shades of meaning and if some users are interested in the usage history of a *term* rather than in documents on a *concept.* Using author's terms in retrieval also serves as a hedge against misinterpretations or omissions on the part of the indexer. Similarly, when indexing a person, we can use terms from a description of skills in his or her resume in addition to standardized descriptors.

While looking for descriptors expressing concepts suggested by the entity, the indexer will often use the alphabetical index to the index language (which is rarely used in request-oriented indexing as implemented by the checklist technique). However, the indexer should not rely on the alphabetical index in the entity-oriented indexing mode either; having found a descriptor through the alphabetical index, she should look it up in a well-structured display to make sure the descriptor is really appropriate. The alphabetical index should be used only as an entry to a classified display unless cost considerations dictate otherwise. One could even use the checklist technique in entity-oriented indexing; this would promote consistent use of descriptors.

13.3 THE ROLE OF THE INDEX LANGUAGE IN SEARCHING

13.3.1 The Checklist Technique Applied to Query Formulation

To arrive at an optimal query formulation, one must identify descriptors under which relevant entities might be found (to ensure appropriate recall) and descriptors that could be used to increase discrimination. The searcher should go through the list of descriptors and ask herself for every descriptor: Are entities indexed by that descriptor likely to be relevant to my (or the client's) problem? Could this descriptor be used to narrow the query formulation? Thus the searcher goes through the descriptor list, using the top-down approach discussed in Section 13.2.3. If this is too much effort, she should at least follow cross-references. For example, a searcher looking for documents on *Relation to own culture* should consider other descriptors as well and use the formulation

> Relation to own culture OR Socialization of
> the individual OR Culture and personality

to ensure reasonably high recall. On the other hand, the query formulation should include all limiting selection criteria explicitly. A fourth-grade teacher might submit a written search request for *Audiovisual materials on science,* which might be simply formulated as

> Audiovisual materials AND Science.

Much to his surprise, the teacher will get not only material suitable for fourth-grade level, but also material on junior high school and senior high school level (or even college level, depending on the scope of the collection of instructional materials). The query formulation should have been

> Audiovisual materials AND Science AND Fourth-grade level.

By browsing through an index language that is displayed in a well-designed classified structure, a user can clarify and focus on her own image or concept of her need. The structure of the index language serves as a catalyst to crystallize the need. This results in a better query formulation and better retrieval results. However, there is the danger that the user will slant his or her real needs toward what can be easily formulated in the index language at hand. This danger can be minimized by first exploring the need without the assistance of the index language structure. Preferably, the user formulates a written statement of the need before interacting with the structure of the index language.

Figure 13.4 shows a query form that illustrates some of the principles discussed.

13.3.2 Compensating for the Lack of Request-Oriented Indexing

If the index language does not provide the descriptors needed for an adequate expression of the need stated, the searcher must use ingenuity and think about various approaches to track down relevant entities as illustrated through the examples in Fig. 13.5.

The searcher must think of *alt* possible approaches; otherwise recall will suffer. It is one thing for an indexer who knows that *Long range transportation planning* is a topic of interest to recognize the relevance of the document being analyzed for this topic. It is quite another thing for the searcher to think of all the possible topics, such as *Shopping habits,* that might have implications for long-range transportation planning.

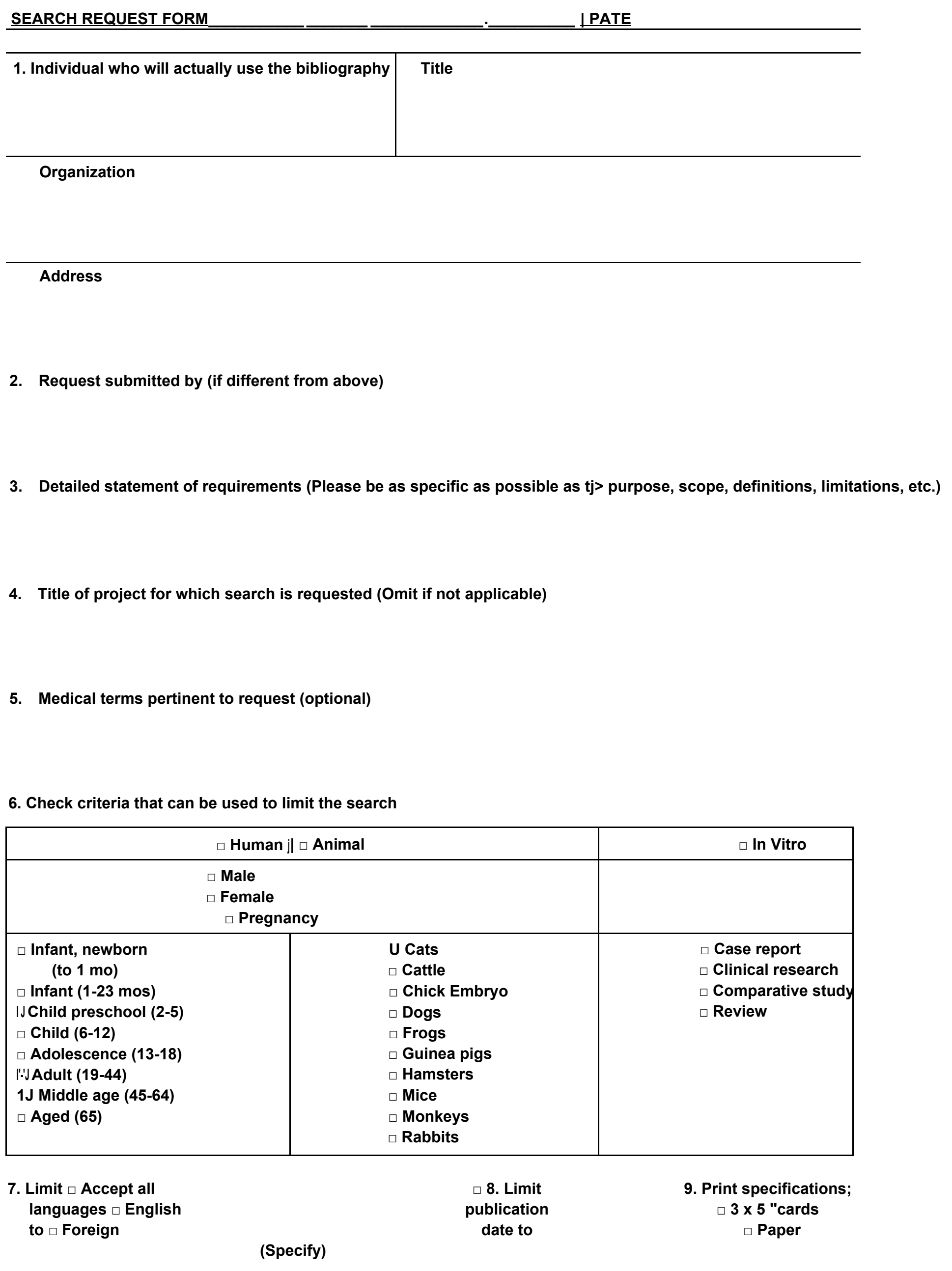
Another difficulty is that some documents on *Shopping habits* or *Flexible work time* are very important for *Long range transportation planning,* but others may have no or only very marginal relevance. The indexer who sees the document can make the judgment, but the searcher has no choice but to retrieve all documents on these topics; as a result, discrimination will suffer.

The examples suggest that *requests for specific empirical data* or facts or concretely described entities are usually *supported through entity-oriented indexing,* whereas *requests based on general, abstract, or theoretical concepts* usually *require request-oriented indexing.* For some general requests the searcher may be able to generate a number of concrete queries that among them will retrieve much of the material needed for the abstract request; to some extent low indexing effort can thus be compensated by increased search effort. This is an example of the familiar trade-off between effort expended in data base building and effort expended in data base use.

13.4 THE ROLE OF THE INDEX LANGUAGE IN DATA BASE ORGANIZATION

A data base or individual file provides the link between a query formulation and the entities indexed. The index language is of major importance for the file organization:

- The more specific the descriptors available for look-up in a file, the higher the degree of order in that file. For example, in a search for documents on *Job performance of firefighters,* looking under the specific descriptor *Job*



SEARCH REQUEST FORM___________ _______ _____________.__________ | PATE

1. Individual who will actually use the bibliography	Title

Organization

Address

2. Request submitted by (if different from above)

3. Detailed statement of requirements (Please be as specific as possible as tj> purpose, scope, definitions, limitations, etc.)

4. Title of project for which search is requested (Omit if not applicable)

5. Medical terms pertinent to request (optional)

6. Check criteria that can be used to limit the search

□ Human || □ Animal		□ In Vitro
□ Male □ Female □ Pregnancy		
□ Infant, newborn (to 1 mo) □ Infant (1-23 mos) IJ Child preschool (2-5) □ Child (6-12) □ Adolescence (13-18) □ Adult (19-44) 1J Middle age (45-64) □ Aged (65)	U Cats □ Cattle □ Chick Embryo □ Dogs □ Frogs □ Guinea pigs □ Hamsters □ Mice □ Monkeys □ Rabbits	□ Case report □ Clinical research □ Comparative study □ Review

7. Limit □ Accept all languages □ English to □ Foreign (Specify)

□ 8. Limit publication date to

9. Print specifications; □ 3 x 5 "cards □ Paper

Fig. 13.4 Search request form. (Modified from MEDLARS form PHS-4667-1, rev 5-66)

1. Intergenerational social mobility

One subsearch: all entities that report on a situation where parents are in one social stratum and the child or children attend a school that prepares for a job in another social stratum. We would need to think about many other subsearches to achieve reasonable recall.

2. Nonproliferation treaty

Assume there is no descriptor *Inspections by foreign nationals.* We could formulate one subsearch thus: all entities indexed by *Inspection* and a country and by at least one other country or international organization.

3. Long-range transportation planning

Some topics to be used in subsearches are:
Development of residential patterns
Where do firms locate
Forecasts of gasoline availability and pricing
Alternate fuels for individual cars
Shopping habits
Flexible work time
Carpooling
Attitudes to mass transit

Fig. 13.5 Compensating for the lack of request-oriented indexing. Sample searches for documents or survey/statistical data.

performance returns a much smaller set of documents than looking under the broad descriptor *Applied psychology.*

- The more specific the descriptors used in indexing, the more information they convey about the document (assuming that the indexing descriptors are included in the entry). To a searcher examining a document record, the specific descriptor *Job Performance* conveys more information about the document than the broad descriptor *Applied psychology.*

In many files, entities or entity representations should be physically collocated for browsing and examination in a logical sequence. The index language is at the heart of achieving such a useful arrangement. Examples are a mail-order catalog, a department store, a handbook of statistical data, a file of newspaper clippings, and a stockroom for parts.

The groups of entities or entity representations must be defined with the browsing needs in mind, and the specific groups must be arranged in broader groups again from the browsing point of view. In many applications the lowest level groups should be fairly broad; an example is the main section of an abstracting/indexing service in which collocation of entries serves primarily the purpose of current awareness. A group with the heading *Applied psychology* (possibly subdivided by type of document or another formal criterion) is appropriate for this purpose; a group with the specific heading *Job performance* would not be. From the point of view of building the file, it would be sufficient to index documents by the broad descriptor *Applied psychology.* However, using only such broad descriptors would be shortsighted; it reflects preoccupation with building the file. Assigning the descriptor *Applied psychology* would indeed be sufficient to file the abstract in its proper place. However, the user browsing the file would profit much from more specific indexing by descriptors such as *Job performance,* because it would provide more information about the document. The document should be indexed with the specific *indexing descriptor, Job performance,* and filed under the broader *filing descriptor, Applied psychology.* Likewise, for a file of organizations, one might use the broad filing descriptor

Information centers in the *Social sciences*

but index specifically by such combinations as

Special libraries	in	*Economics*
Data archives	in	*Sociology*
Clearinghouses	in	*Urban planning*

These specific indexing descriptors provide useful information to the searcher. Specific indexing also gives flexibility in file organization. If experience shows that narrower descriptors, such as *Job performance,* would be useful for filing after all, or that all *Data archives* should be collocated no matter what their subject field, it is an easy matter to rearrange the file cards accordingly. On the other hand, if we do not look ahead—if we index merely for the purpose of proper filing in the present arrangement, using the descriptor *Information centers in the social sciences*—then later reorganization of the file will require reindexing. Furthermore, a specific descriptor can do double duty; *Job performance* is appropriate for the subject index and also causes the abstract to be placed under *Applied psychology* in the main file. In a system that provides both a printed index and a computer search capability (for example, Index Medicus and MEDLINE), very specific descriptors are used for computer searching and mapped to descriptors of medium specificity appropriate for the printed index. Specific indexing, even if not needed to create files presently used, keeps future options open. To conclude, the specificity of indexing descriptors and the specificity of filing descriptors should be determined separately, each for its own purpose.

This principle is important also for cooperation in cataloging. For example, the Library of Congress (LC) assigns the most specific Dewey Decimal class that any library might need, but a library using LC's cataloging might well determine that a broader class would be more appropriate for filing arrangement on its shelves. For instance;

Class assigned by LC:	331.127	*Labor force mobility*
Class used by Library:	331.1	*Industrial relations*
or even	331	*Labor economics*

13.5 CHOOSING THE BEST INDEXING APPROACH

There is no one "correct" approach to indexing. In some situations entity-oriented indexing is appropriate; in others, request-oriented indexing; and in still others, a combination. In deciding on the approach to be used in a particular situation, one must consider the costs for indexing and the costs for searching weighed against the expected ultimate benefits (or at least against the expected ISAR system performance). Since hardly any experimental data comparing request- and entity-oriented indexing are available, the following discussion draws on the logical analysis of the two approaches presented in the previous sections.

In the case of descriptors of general application like the MEDLARS *check tags* (see Fig. 13.4), the checklist technique of indexing has only advantages; it can be used by indexers on all levels of sophistication, it is faster than writing these descriptors on the form, and it promotes consistently correct use of these descriptors. Thus the searcher can rely more on these descriptors. The following considerations apply to request-oriented indexing on a higher conceptual level.

13.5.1 COST OF INDEXING

Request-oriented indexing costs more than entity-oriented indexing. The construction of an index language for request-oriented indexing requires a study of needs and much thought and expertise. Indexing must be done thoroughly; intelligent relevance judgments take time. Reliable request-oriented indexing requires subject experts capable of judging the relevance of an entity for a given concept. (One might speculate that even nonsubject specialists would do better indexing in the request-oriented mode than in the entity-oriented mode, but the results of their judging relevance would not be as reliable.) Entity-oriented indexing, particularly of the extraction-and-translation variety, can be done by indexers who are not subject experts or even through a computer program. (Many bibliographic data bases do not even do their own indexing but use titles and/or abstracts or other existing representations for searching, in effect using the author or abstractor as indexer.) On the other hand, it would be extremely difficult to approximate the kind of relevance judgments needed for request-oriented indexing through a mechanized procedure. If request-oriented indexing is the main approach, then additional entity-oriented indexing increases the cost of indexing.

13.5.2 Quality of Indexing

An increase in indexing quality brings lower search cost and better search results. The checklist technique of indexing promotes correct and consistent use of descriptors, whether the overall approach to indexing is request- or entity-oriented. Request-oriented indexing provides a richer set of retrieval clues for an entity. Request-oriented indexing increases the likelihood that an entity (such as a document or a time series of statistical data) that is *Important for long-range transportation planning* is indeed indexed by that descriptor. However, some descriptors representing aspects for which an entity is relevant may not be assigned with the same reliability as descriptors that merely express obvious characteristics of the entity. Consider the following intuitive estimates for indexing a document entitled *Attitudes of the residents of the Washington, D. C. Metropolitan Area toward METRO.*

	Probability of assignment	
Descriptor	with entity-oriented indexing (%)	with request-oriented indexing *(Vo)*
Survey research	60	80
Attitudes	95	95
Washington, D.C. Metropolitan Area	95	95
Local rail transit	95	95
Mass transportation	70	90
Important for long-range transportation planning	10*	50

a\i that descriptor is in the index language for entity-oriented indexing

One might argue that indexers should be discouraged from indexing implications of a document, since indexing consistency would suffer. But that is a bit like rejecting half a loaf because one cannot have a whole one. It would certainly be better to find 50% or even 30% of the relevant entities in a search for

Important for long-range transportation planning

than only 10% or none at all. Furthermore, the entities that are retrieved because an ingenious indexer has seen their implications for long-range transportation planning might suggest topic areas under which to search further. In the *long-range transportation* search, for instance, other topic areas might be *A ttitudes* AND *Mass transportation, Shopping habits,* and *Flexible worktime.* A first-round search retrieves at least some relevant entities because an indexer working in the request-oriented mode recognized implications. The entities retrieved then help in the formulation of several second-round searches, which compensate for the indexer's failure to recognize implications in the case of other entities. While consistently good indexing is desirable, indexing consistency that results from reducing well-indexed entities to the level of badly indexed entities is detrimental to retrieval performance.

13.5.3 Cost and Quality of Searching

Whether request-oriented indexing is worth the price depends on the number and types of requests and on the importance of search quality. If most requests are specific and concrete, request-oriented indexing may not be needed. For requests that can be answered with entity-oriented indexing, albeit with greater search effort, the matter can be settled based on cost alone. If many requests need request-oriented indexing for a high-quality answer, and if these requests are important, then the ISAR system designer must determine whether the ultimate benefits derived from these requests justify the increased cost of request-oriented indexing.

This discussion is based on the assumption that the requests fully reflect needs. However, this is not always the case. For example, an ISAR system may be used only to search for specific empirical data, because it is not suitable for general searches due to its entity-oriented indexing, thus compelling the users to submit only searches for specific empirical data. In this case entity-oriented indexing is not sufficient, because it does not support potential use.

13.6 THE FUNCTIONS OF HIERARCHY: A SUMMARY

This section summarizes the functions of hierarchy and classified arrangement in indexing and query formulation, searching and processing, data base organization, and cooperation.

Functions in Indexing and Query Formulation

1. Facilitate the *checklist technique for indexing and for query formulation* (especially browsing through the descriptors available).
2. *Assist the indexer and the searcher in the choice of the appropriate level of generality.* A general rule for searching is: Use the most specific descriptor

that still includes the topic requested. A general rule for indexing is: Use the most specific descriptors that still cover the aspects or concepts for which the entity is relevant. Since the classified display shows at a glance descriptors both broader and narrower than the descriptor being considered, it assists the indexer and searcher in the application of this rule.

Functions in Searching and Processing

3. Facilitate *inclusive searches.* For example, a search for *Meat, inclusive* would retrieve entities indexed by the narrower descriptors *Beef, Pork, Lamb,* and *Venison* as well; this would be particularly useful for searches combining two broad descriptors, such as *Food additives* in *Meat.* In mechanized ISAR systems inclusive searching is implemented through a search program that utilizes hierarchical relationships stored in the computer; in manual systems inclusive searching is facilitated through classified arrangement. Inclusive searching is particularly important for SDI, where broad queries are common.
4. Facilitate the formation of *aggregates in statistical analysis.* For example, one may want to know the per capita consumption of *Meat* (including *Beef, Pork, Lamb,* and *Venison)* or of *Meat, Poultry,* and *Seafood* combined.

Functions in Data Base Organization

5. Facilitate *specific indexing and more general filing arrangement* where appropriate.
6. Facilitate the *collocation of related entities* in files. Collocation in turn facilitates inclusive searching; for example, a search for *Meat, inclusive* can be conducted much more easily in a classified subject catalog (in which *Beef, Lamb, Pork,* and *Venison* follow immediately after *Meat)* than in the more customary alphabetic subject catalog. Helpful collocation also brings related entities to the attention of the searcher; it facilitates browsing.

Functions in Cooperation between Systems

7. Facilitate *shared subject indexing.*

Hierarchy also plays a very important role in the construction of index languages and thesauri.

13.7 A PHILOSOPHY OF INDEXING AND CLASSIFICATION

This section summarizes the rationale for request-oriented indexing using a logically structured index language.

Two opposing principles for building an index language and thesaurus can be found in the literature:

1. Follow the vocabulary of the user as closely as possible; omit terms not contained explicitly in the user's vocabulary, even if they are necessary for logical coherence.
2. Follow, insofar as possible or even exclusively, the vocabulary of the entity creator (e.g., document author) so as not to distort the meaning of the entity creator. Accordingly, include in the index language only terms appearing in author-prepared entity representations such as the text or title of documents, food names given by manufacturers, or self-descriptions of persons or organizations; omit terms not appearing explicitly in such sources, even if they are necessary for logical coherence. (Principle of literary warrant.)

Each of these principles has merit; but the exclusive use of one or the other fails to solve the problems of communication that have been outlined in the previous sections. It is the task of a thesaurus to support optimal service to the user by providing the foundation for indexing and retrieval operations. This task requires more than following the user's or the author's vocabulary as closely as possible. There is no such thing as "the user"; there are many users, and their viewpoints often contradict each other* There is no such thing as "the author" either; there are many authors, and they often use different terminology. Authors and users often have different purposes: The use a user makes of an entity is often quite different from what the author thought the entity would be useful for. The indexer serves as the user's agent by indicating possible uses of each incoming entity. The indexer must analyze the entity at hand and then make a sound relevance judgment that is as useful as (or perhaps even more useful than) the user's own relevance judgment would be. At his best the indexer does "scientific prethinking." By analyzing entities as the user's agent, the indexer saves the user time. Ideally, the indexer evaluates each entity critically, something the user may not be able to do for lack of time or lack of expertise or both.

In order that the indexer can fulfill this demanding role, he or she must have a clear picture of the problems or tasks of the user and the information or entities needed to solve these problems. If there were only very few users, they could communicate their interests directly to "their" indexer. However, the normal situation is quite different: There are many users, most of whom the indexer does not know. Hence the mental frameworks of many users must be combined into one logical coherent structure that can be understood and internalized by the indexer. Careful analysis of needs and critical examination of the conceptual structure of the subject field at hand are needed to develop such a framework. The index language thus constructed serves as a communication device from the users to the indexers; it provides the framework that allows for a meeting of minds to take place.

The index language, once constructed based on the analysis of the needs of all users, also serves as a communication device from the information system to the individual user. It gives the user a mental framework, a knowledge map, a guide through the collection of information or entities available in the information system. (In a library where materials are arranged in a meaningful order or in a grocery store the user literally has a map of where to find what.) If the structure of such a knowledge map can be made congenial to the user's own mental framework, so much the better. But the user's framework may be less suitable, less powerful for organizing the subject matter at hand than an index language (or classification) constructed through careful consideration of the foundations of the subject. The index language then ceases to be a mere tool for retrieval and becomes a powerful agent for education, enriching the user's mind. The conceptual framework developed for the external information system can be used to improve organization of the user's own internal information system. This takes on particular significance with an information system for children or students, since young minds are apt to absorb the organizing principles used in such a system and use them to build their own view of the world. Hence, an index language should use structural principles derived from modern classification theory—such as the principle of facet analysis to be discussed in Chapter 14—and a semantic organization based on the newest insights and paradigms of the subject fields covered.

To conclude, the maker of an index language and thesaurus is confrontec with the challenge of clarifying the muddled terminological and conceptua systems of a field (or perhaps several fields combined) and detecting it! underlying logical structure, thus laying a foundation for successful com munication.

CHAPTER 14

Index Language Structure I: Conceptual

OBJECTIVES

The objectives of this chapter are to disaiss the principles of conceptua structure—hierarchy, concept combination, and their interaction—and th< application of these principles to searching; and to enable the reader to us» facet analysis to uncover the conceptual structure of a field for improved in dexing, searching, and index language construction.

*

INTRODUCTION

The entity type *Concept (Subject, Topic)* plays an important role in th retrieval of entities, either directly, as in a search for all *Documents o Nongraded grouping* (of students), or indirectly, as in a search for a *Students* attending *Schools* that are on the *Elementary level* and u« *Nongraded grouping.* Retrieval based on concepts is called *subject retrieva* There are many relationships among concepts; these make up the inde language structure. This structure serves essential functions in indexing, dai base organization, and searching. Many of the considerations in th chapter, especially the discussion of hierarchy, apply to the relationshi] among entities of other types as well.

Index language structure has two intertwined aspects: conceptual and da base organizational. We first give some examples of conceptual analysis, concept, such as *Frozen beans*, can be generated by combination:

Food product [< has source > *Bean* AND < is in state > *Frozen*);

Or we can analyze a compound concept, such as *Ship:*

Object [<is a> *Vehicle* AND <serves for> *Water transport*]

Or, shortened,

> *Ship* = *Vehicle* : *Water transport.*

Or we can establish hierarchical relationships between concepts:

> *Beans* < is a > *Vegetable* or, less precisely,
> *Beans* < has Broader Term> *Vegetable.*

The question of data base organization arises in the decision whether to use *Frozen beans* as a descriptor, thus grouping all *Frozen bean* products together, or whether to use just the elements *Beans* and *Frozen* as descriptors, which must then be combined in retrieval. Hierarchical relationships, too, serve for data base organization—such as putting all *Vegetables (Beans, Peas, Spinach,* etc.) together on grocery shelves. This chapter emphasizes the conceptual aspects of index language structure; in the next chapter emphasis shifts to data base organization.

14.1 HIERARCHY

Consider the following examples. In a search for *Frozen vegetable* products, all *Frozen bean* products should be found; *Frozen vegetable* is broader than *Frozen beans. The Food and Drug Administration* includes the *Bureau of Foods.* In some universities, the *Psychology* Department covers *Social psychology.* An indexer having determined that a document is relevant for the concept *Method of instruction* should check further whether the document is relevant for *Individualized instruction.*

These are examples of hierarchy. Hierarchy serves many functions: It facilitates the checklist technique of indexing and query formulation, assists in the choice of the appropriate level of generality, facilitates aggregation in statistical analysis, allows for specific indexing and more general filing arrangement, facilitates the collocation of related entities, and facilitates shared subject indexing. A hierarchical relationship should be introduced whenever it serves one of these functions. With respect to the retrieval function there is a pragmatic *hierarchy test,* stated here for retrieval of documents:

> Should a search for documents dealing with *A* find all (or most) documents dealing with *B?* If yes, *A* is broader than *B* (and conversely, *B* is narrower than *A).* (This formulation can easily be generalized to other types of entities and relationships used in retrieval.)

Even though hierarchical relationships exist between *concepts,* the usual expressions are *Broader Term* (BT) and *Narrower Term* (NT); we follow this widespread usage.

The *traditional approach to hierarchy building* (exemplified in such systems as the Library of Congress Classification and the Dewey Decimal Classification) derives from the attempt to create a neat and meaningful arrangement for a set of entities in which every entity has its unique place; this is a problem in data base organization, not primarily a problem of conceptual structure (compare Section 13.1.2). Such an arrangement can be brought about by first grouping the entities so that each group corresponds to a concept and then constructing a neat and meaningful arrangement of these concepts. This can be done from the top down, subdividing the set of concepts into mutually exclusive groups, subdividing each group in turn into mutually exclusive subgroups, and so on. Or the arrangement can be developed from the bottom up, assembling the concepts into larger and larger groups. If a concept does not fit naturally anywhere into the arrangement, it is forced somewhere. If a concept would fit into different places, it is more or less arbitrarily assigned to one of them; no concept is allowed tc have more than one broader concept. This is the principle of *monohierarchy*, it is artificial and imposes many constraints.

In contrast, the *modern approach to hierarchy building* establishes al hierarchical relationships that are useful for searching and the other func tions just listed. Each pair of concepts (A, B) is analyzed to see whether i meets the hierarchy test. If so, a hierarchical relationship is established.

Examples:

BT = Broader Term (really Broader Concept)

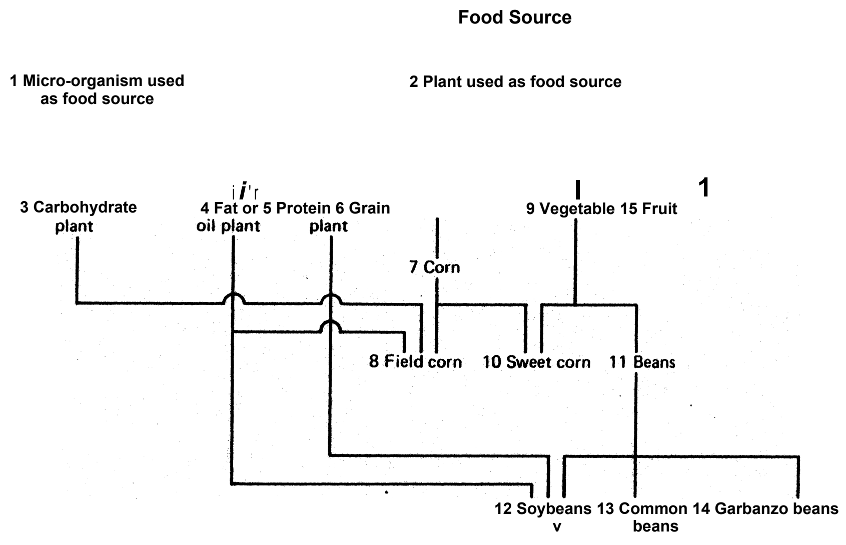
Beans	BT	Vegetable
Zoology	BT	Biology
Biology	BT	Science
Constitution	BT	The state (BT Govt, and politics)
	BT	Public law (BT Law)
Social psychology	BT	Sociology
	BT	Psychology

In the example many concepts have two broader concepts; this situation called *poly hierarchy.* Other concepts end up having just one broader coi cept, but with polyhierarchy this is not a restriction imposed by the systen Still other concepts may be left without any broader concept at all. The? concepts form the top of the hierarchy; they may be broad subject fields sue as *Science,* but they may also be specific concepts that happen to have no broader concepts such as *Packaging* (in the Dewey Decimal Classification there is no class number for this concept as a whole, only class numbers for the compound concepts *Economic aspects of packaging* and *Technical aspects of packaging)* or *Weights and measures* (in DDC this is wrongly placed under 380 *Commerce,* even though *Weights and measures* may occur in a purely scientific-technical context). Figure 14.1a shows a polyhierarchical structure.

The principle of poly hierarchy is very important. If the rigid principle of monohierarchy is used, many arguments result from the question of which of several broader concepts is the "true" broader concept, since only one is allowed. But there is no point in arguing whether *Social psychology* should be placed under *Sociology* or under *Psychology.* Either solution would be inadequate; it belongs under both. Polyhierarchy avoids such futile arguments. This illustrates one of the most important insights of modern classification research; A polyhierarchical scheme allows for a better representation of the conceptual structure of any field; many problems encountered in the construction of rigid monohierarchical schemes are revealed as fictitious. In this book *hierarchy* means *poly hierarchy.*

These ideas are also useful in designing organizational structures where rigid monohierarchy does not do justice to the complex interrelationships between parts of an organization that are due to the intrinsic interrelatedness of the real-world problems they are dealing with. For example, the responsibility of the *Bureau of Foods* has strong linkages with the *Food and Drug Administration* (in the *Public Health Service),* particularly with respect to food safety, but equally strong linkages with the *Department of Agriculture,* particularly with respect to the nutritional value of food. These linkages should be reflected in the organizational structure, which could show the *Bureau of Foods* as subordinate to and dealing with both the *FDA* and the *Department of Agriculture.* Likewise, it makes little sense if the Sociology Department and the Psychology Department argue over who should cover *Social psychology.* They should cooperate and develop a joint plan of courses.

The hierarchical relationships detected must be displayed clearly in order to communicate to a reader the conceptual structure of a field, in particular to facilitate the checklist technique of indexing and query formulation. A graphical display as shown in Figure 14.1a is useful for small sets of concepts (e.g., in an on-line display), but for a large printed display all concepts must be arranged in a linear sequence with headings and subheadings for easy scanning. The arrangement should express as many of the hierarchical relationships as possible and should collocate related concepts. *Any hierarchical relationships that are left over are expressed through cross-references* as shown in Figure 14.1b. The problem of choosing one place for a concept that



(a) Representation as a graph.

Food Source
- 1 Microorganisms used as food source
- 2 Plant used as food source
  - 3 Carbohydrate plant NT 8
  - 4 Fat or oil plant NT 8,12
  - 5 Protein plant NT 12
  - 6 Grain
    - 7 Corn NT 10
      - 8 Field corn BT 3,4
  - 9 Vegetable
    - 10 Sweet corn BT 7
    - 11 Beans
      - 12 Soybeans BT 4, 5
      - 13 Common beans
      - 14 Garbanzo beans
  - 15 Fruit
- 16 Animal used as food source

(b) Representation as a linear sequence with cross-references.

Fig. 14.1 Polyhierarchy. Excerpt from Food source facet.

has two broader concepts now *does* arise, since listing the concept in both places is often impractical. Furthermore, concepts that do not have broader concepts need a place. New broader concepts may be needed as headings to facilitate a rapid grasp of the overall arrangement and to support scanning the list of descriptors in the checklist technique of indexing and query formulation. The linear sequence of concepts is also important for data base organization, especially the arrangement of entities (e.g., documents or groceries on shelves).

Complementing the hierarchical relationships are *associative relationships,* usually called *Related Term* (RT) relationships. Concept A is related to concept B if an indexer or searcher weighing the use of A should be reminded of the existence of B.

Examples:

Political ideas	RT Social philosophy
Nongraded grouping	RT Montessori method
	RT Individualized instruction
Copyright	RT Reprography

To sum up: hierarchy must never be a straightjacket in which the universe of knowledge has to fit somehow or other. On the contrary, a properly designed hierarchy shows the manifold relationships between concepts and thus assists in indexing and searching. Whenever a hierarchy sets constraints, it is faulty; whenever it helps the indexer or searcher, it serves its purpose.

14.2 CONCEPT COMBINATION AND SEMANTIC FACTORING. FACET ANALYSIS

Concepts can be combined to form new concepts, such as *Beans*: *Frozen* or *Meat*: *Canned.* Conversely, a compound concept, such as *Frozen beans,* can be analyzed to determine its components, *Beans* and *Frozen.* Earlier we gave a less trivial example:

Ship - Vehicle : Water transport.

Likewise,

Automobile = Vehicle : Road transport
Aircraft = Vehicle : Air transport

The components are called *semantic factors,* and the process of analyzing a compound concept into its components is called *semantic factoring.*

Semantic factoring can be carried further; for example:

Vehicle = Means : Transportation : Mobile or
Water transport = Water : Transportation

Carrying this process to the end leads to *elemental concepts* that cannot be factored further. In terms of elemental concepts, *Ship* can be expressed as

Ship = Means : Transportation : Mobile : Water

Semantic factoring establishes conceptual relationships between a compound concept and less compound concepts; for example:

Ship < has semantic factor > *Vehicle*
Vehicle < is semantic factor of > *Ship*

This gives the index language builder two options with respect to treating *Ship.* Option 1 is to use *Ship* as a subject descriptor; the relationships to the concepts that are its semantic factors should then be shown in the index language for the benefit of the searchers. Option 2 is to omit *Shipirom* the index language and instruct the indexer to use instead the combination *Vehicle : Water transport*, if these are descriptors, or *Means : Transportation Mobile : Water.* The choice between these two options is a matter of datJ base organization to be discussed in Chapter 15. Depending on the optior chosen, a searcher can use *Ship* as the query formulation, or he must use th< appropriate combination (e.g., *Vehicle* AND *Water transport).* Either way a search for *Vehicle* will find documents dealing with *Ship* or objects that an *Ships* (in general, entities that are in a relationship with *Ship).*

A compound concept used as descriptor is called a *precombined descriptor;* an elemental concept used as descriptor is called an *elemental descriptor* An ISAR system that uses primarily precombined descriptors, so that concepts are combined prior to indexing, is called a *precombination system.* Ai ISAR system that uses primarily elemental descriptors, which then are combined in query formulations (after the indexing is completed), is called *postcombination* system. (The terms *precoordinate* and *postcoordinate* ar also used, but they do not express the idea as well.) (See Chapter 15 fc elaboration.)

Semantic factoring, the *conceptual* decomposition of compound concept according to their meaning, should not be confused with the linguisti decomposition of multiword (composite) terms. For example, *Rare eart metals* is a multiword term designating an elemental concept (a class c chemical elements); semantic factoring is not useful in this case. On the oth< hand, the single-word term *Ship* designates a compound concept. The sing words in a multiword term that designates a compound concept are not always the appropriate terms for the semantic factors. A striking example is *White House* (in the sense "the White House announces" or "a White House aide"):

White House = Administrative agency : Chief executive
officer : United States

This meaning of *White House* has nothing to do with *White* and *House; White House* is related much more closely to *10 Downing Street* than to the *White rabbit* from *Alice in Wonderland.* On the other hand, the linguistic structure often *does* reflect the conceptual structure, as in the *Frozen beans* example.

The following examples illustrate semantic factoring further.

Bus = Vehicle: Road transport : Passengers: Large capacity
Truck = Vehicle: Road transport : Freight : Large capacity
Trade negotiations = International negotiations : Foreign trade
Lego standard block = Rectangular block: Thick: 2 rows wide: 4 rows long
Bean curd = Cheese-product analog : Soybean : Protein concentrate : Semisolid

(Bean curd, or tofu, is made by preparing a liquid from soybean flour and water, curdling, draining, and pressing into molds.)

Semantic factoring is best understood by doing it. The task of expressing a compound concept through components occurs also in formulating a query and in indexing an entity, but then the compound concept to be expressed may not have a name. Usually one can readily discern the semantic factors of a concept.

Facet analysis is helpful for finding all semantic factors and for solving difficult cases. Facets are aspects or viewpoints from which entities—such as food products or subjects (topics, themes) in an area such as education—can be analyzed. Fig. 14.2 gives the outline of a faceted classification for *Food products.* If the system rules specify indexing only by main ingredient, this scheme comes close to the ideal type of a faceted scheme: Every food product has exactly one descriptor from each facet. (Indexing by all ingredients results in many combinations of a *Food source* term with a *Part* term.)

It is best to use a list of facets that is adapted to the subject field at hand. The list of questions in Fig. 14.3a are helpful in eliciting the semantic factors of a concept. This leads to elemental concepts. Figure 14.3b shows sample facet headings to be used for the arrangement of elemental concepts. It can serve as a starting point for developing a subject-specific list. Fig. 14.4 shows the outline of a faceted classification for *Education.*

- Product type
- Ingredients
  - Food source (Species or variety of plant or animal)
  - Part of plant or animal
- Physical state and form
  - Physical state
  - Physical form
- Processing
  - Degree of cooking
  - Treatment applied
  - Preservation method used
- Packaging
  - Packing medium
  - Container type
  - Food contact surface
- User group

Fig. 14.2 Facets for the analysis of food products.

Of which class is it (the concept) a member or a subclass?
What is it made of?
What are its distinctive properties? Is it in a specific state, condition, or circumstance?
Does it participate in a process? What is it capable of?
Does it determine, cause, influence, produce, or act upon something else? Is it determined, caused, influenced, produced, or acted upon by something else?
Has it a specific purpose, is it a means or instrument to achieve something else? Is it a goal or end achieved or to be achieved by something else?
Is it a theory of something or an aspect of looking at something? Is it looked at under a specific aspect or viewpoint?
Is it a part of something?
Is it or is it not accompanied by something else or accompanying something else? Is it in a specific environment?

(a) Questions to elicit the semantic factors of a concept.

Things, objects, ideas
Materials
Properties, states, conditions, characteristics
Processes
Goals, objectives, purposes

(b) Facet headings to arrange elemental concepts.

Fig. 14.3 General facets.

Persons
Subdivided by role
Students, educands
Teachers, educators
Subdivided by . . .
(A faceted classification of personal attributes)
Educational objectives and content
Subdivided by general objectives
Subdivided by subject taught
1 Science
1.1 Physics
1.1.1 Mechanics
1.1.2 Optics
1.2 Chemistry
1.3 Biology
1.3.1 Botany
1.3.2 Zoology
2 Social studies
3 Language arts
Curriculum (the structure of a subject)
(RT Learning-teaching processes)
Learning-teaching processes and activities
Methods of instruction
Learning-teaching materials
Learning-teaching environment
For example, School, Other formal group, Home
One subfacet: Sponsorship
Grade level
1 Elementary school (ES)
1.1 First grade

1.6 Sixth grade
2 Junior high school (JH)
3 Senior high school (SH)
3.1 10th grade

Fig. 14.4 Outline of a faceted classification for education (some detail filled in for illustration).

A list of facets, once established, serves as a checklist for analyzing compound concepts: Each facet can be construed as aquestion (e.g., What is the product type? What is the food source?) and the proper semantic factor for the food product (compound concept) at hand is the answer to that question. A list of facets thus serves as a framework for the analysis of compound concepts or topics; it is a*facet frame.* Some facets may not be applicable, or the concept may be too broad to have a value specified. Take, for example, the food product *Cut green beans;* the indexer cannot specify values for the facets *Preservation method used* (it could be frozen or canned), or *Container*. On the other hand, one facet may take several values, as in a bread made from wheat and rye.

14.3 INTERACTION BETWEEN CONCEPT COMBINATION AND HIERARCHY

We have discussed two principles of index language structure: hierarchy and semantic factoring or concept combination. These two principles do not exist in isolation from each other; Section 14.2 already gave examples of their interaction. *Beans* is broader than *Frozen beans:* A search for all bean products (search term: *Beans)* certainly should find all frozen bean products. *Frozen* is also broader than *Frozen beans. Frozen* arid *Beans* are also the semantic factors of *Frozen beans.* Likewise, the two semantic factors of *Ship—Vehicles* and *Water transport*—are also broader terms of *Ship.* See Section 14.2 for many more examples.

Hierarchical structures can be generated using this interaction principle. A simple example is shown in Fig. 14.5; the structure is simple because there are no hierarchical relationships within the facets. Starting from four *generate ing concepts* (in bold frames), arranged in two facets, one proceeds through the following steps:

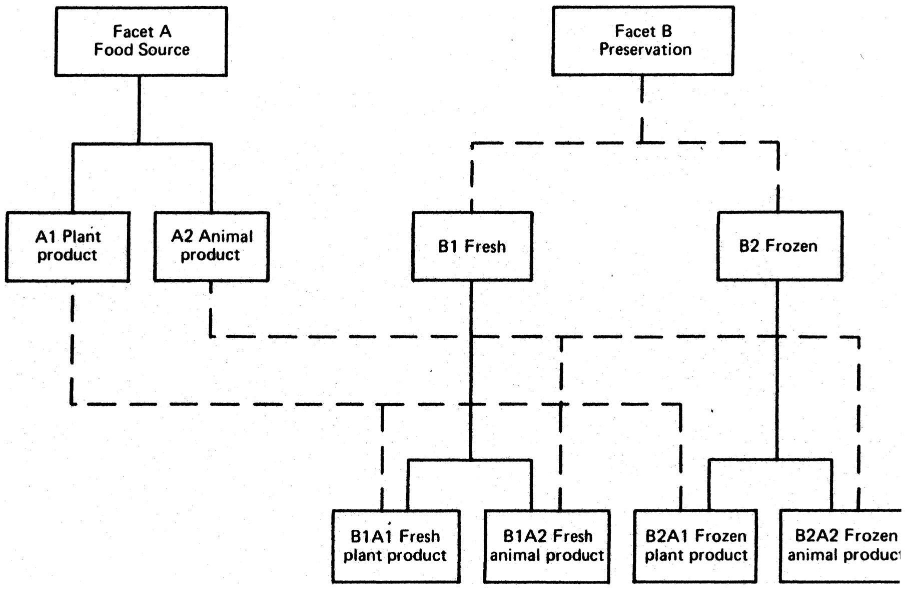
1. Form all possible between-facet combinations; there are 2 x 2 = 4 combinations and four original concepts, for a total of eight concepts (four generating, four produced by combination). (Within-facet combination are omitted to keep matters simple.)
2. Using the hierarchy test, determine all hierarchical relationships in the set of eight concepts; there are eight:

Fresh plant product	Frozen plant product
BT Fresh	BT Frozen
BT Plant product	BT Plant product
Fresh animal product	Frozen animal product
BT Fresh	BT Frozen
BT Animal product	BT Animal product

3. Represent the polyhierarchical structure in a graph as in Fig. 14.5a.
4. Represent the polyhierarchical structure as a linear sequence with cross-references as in Fig. 14.5b. This figure shows two out of many possible arrangements.

Note how the pattern of facet A is repeated under each element of facet B (solid lines) and how the pattern of facet B is repeated under each element of facet A (broken lines). Thus, one partial view of this structure is that the pattern given in facet A is used to subdivide each element of facet B (linear arrangement 1); another partial view is that the pattern of facet B is used to subdivide each element of facet A (linear arrangement 2). Projecting these two partial views into one results in the total view of the structure as represented in the graph. In each of the linear arrangements representing a partial view, we introduce cross-references to make the representation complete. (A1B2 and B2A1 designate the same combination of concepts; the sequence of the notationai elements is adapted to arrangement 1 and arrangement 2, respectively.)

When dealing with the conceptual aspects of index language structure, the designer is concerned with introducing *all* useful hierarchial relationships: Both viewpoints are equally important and should have equal weight, as in the graph. But the designer must also deal with the data base organizational aspects of index language structure. A linear sequence representing the hierarchical structure can be used as a guide in arranging entities, such as groceries or documents, or entity representations, such as abstracts. In such an arrangement, collocation is used for retrieval, and it does make a difference whether one collocates all *Fresh* products and all *Frozen* products while distributing *Plant products* and *Animal products* across the file, as in arrangement 1, or whether one collocates all *Plant products* and all *A nimal products* while distributing *Fresh* products and *Frozen* products across the file, as in arrangement 2; see Section 15.5.2.



(a) Graphical presentation.

Arrangement 1	Arrangement 2
Preservation more important (bold and solid lines)	Food source more important (bold and broken lines)
A Facet A. Food Source	A Facet A. Food source
A1 Plant product NT B1A1, B2A1	Al Plant product A1B1 Fresh plant product BT B1 A1B2 Frozen plant product BT B2
A2 Animal product NT B1A2, B2A2	A2 Animal product A2B1 Fresh animal product BT B1 A2B2 Frozen animal product BT B2
B Facet B. Preservation	B Facet B. Preservation
B1 Fresh B1A1 Fresh plant product BT A1 B1A2 Fresh animal product BT A2	B1 Fresh NT AIB1,A2B1
B2 Frozen B2A1 Frozen plant product BT A1 B2A2 Frozen animal product BT A2	B2 Frozen NT A1B2, A2B2

(b) Representation as a linear quence with cross-references.

Fig. 14.5 Hierarchical structure generated b two facets. No hierarchy within facets.

A more complex example is given in Fig. 14.6, where two generating concepts are added to facet A, introducing hierarchy in facet A. Now there are eight combinations, for a total of 14 concepts. The hierarchical relationships in facet A are given, not the result of combining concepts; they are due to *autonomous subdivision* (our term). There are now many additional hierarchical relationships, for example,

```
A1.1B2 Frozen vegetable BT   A1B2 Frozen plant product
                        BT   Al I Vegetable
                        BT   B2 Frozen
                        BT   A1 Plant product
```

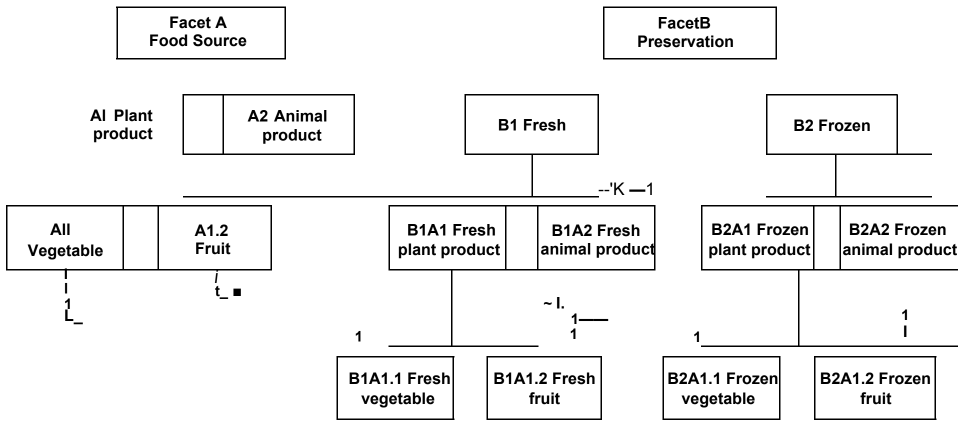
The first broader concept has itself two components; in this case the first step of broadening is achieved not through omitting a component but through substituting the broader concept *Plant product* for the original concept *Vegetable;* the hierarchical relationship is due to *substitution* (our term) rather than combination. The last two broader concepts are implied by the first one:

```
                                               BT Frozen
Frozen vegetable BT Frozen plant product
                                               BT Plant product
```

The graphical representation shows both chains; this obviates the need for a direct line from *Frozen vegetable* to *Frozen* or from *Frozen vegetable* to *Plant product,* which would only be confusing. In arrangement 1, the chain to *Frozen* is shown through the arrangement itself; in arrangement 2, it is shown through the cross-references:

A1.1B2 Frozen vegetable BT A1B2 Frozen plant product
Al B2 Frozen plant product BT B2 Frozen

Showing cross-references only to the immediately superordinate or subordinate hierarchical level cuts down on the number of cross-references; furthermore, the user following such a chain sees each broader concept in its total environment within the index language structure and thus is better able to grasp the parts of that structure relevant to the task at hand.



(a) Graphical representation.

Arrangement 1
Preservation more important

A Facet <u>A Food Source</u>

- **Al Plant product NT B1A1, B2A1**
  - **A 1.1 Vegetable NT B1A1.1, B2A1.1**
  - **A 1.2 Fruit NT B1A1.2, B2A1.2**
- **A2 Animal product NT B1A2, B2A2**

B <u>Facet B Preservation</u>

- **B1 Fresh**
  - **B1A1 Fresh plant product BT Al**
    - **B1A1.1 Fresh vegetable BT A1.1**
    - **B1A1.2 Fresh fruit BT A 1.2**
  - **B1A2 Fresh animal product BT A2**
- **B2 Frozen**
  - **B2A1 Frozen plant product BT Al**
    - **B2A1.1 Frozen vegetable BT A 1.1**
    - **B2A 1.2 Frozen fruit BT A 1.2**
  - **B2A2 Frozen animal product BT A2**

■(b) Representation as a linear sequence with cross-references. Arrangement 1.

Fig. 14.6 Hierarchical structure generated by two facets.

This example is more indicative of the complexity in a real-life hierarchy. It shows how a linear arrangement with cross-references can help a searcher to find all descriptors under which to search and how it can help an indexer to find all descriptors that he or she should check when indexing an entity. Assume a system in which all compound concepts of Fig. 14.6 are used as precombined descriptors. A user entering under *Frozen plant product* should be reminded of the narrower descriptors *Frozen vegetable md Frozen fruit.* In arrangement 1 this is achieved through the sequence; in arrangement 2, through crossreferences.

Proper guidance of the user depends on correct analysis of the hierarchical relationships. Consider the common mistake of determining broader concepts by simply choosing the two components of a combined concept, neglecting substitution as a mechanism for forming broader concepts:

A1.1B2 Frozen vegetable BT AI.I Vegetable (1 level up)
BT B2 Frozen *(2* levels up,
omitting the intervening concept A1B2 Frozen plant product)

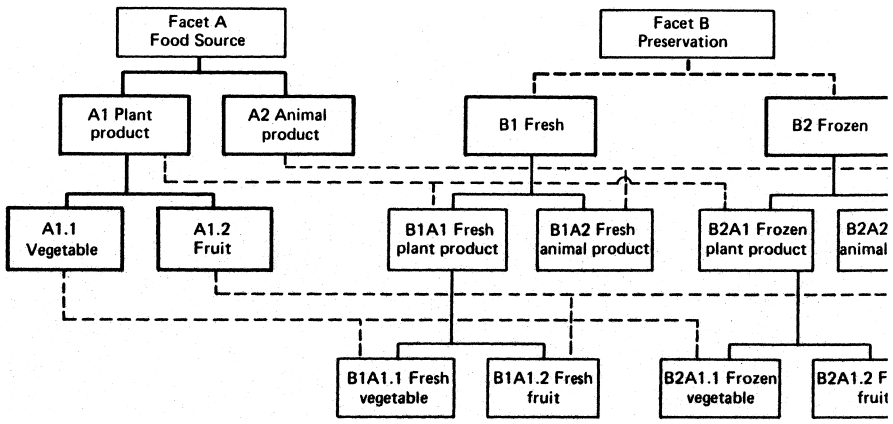
This mistake causes omission of the reciprocal cross-reference:

A1B2 Frozen plant product NT A1.1B2 Frozen vegetable

and thus the user searching for *Frozen plant product* is not reminded to search also under *Frozen vegetable.*

The same problem occurs in query formulation with elemental descriptors. Assume a search for B2 *Frozen* AND AI.I *Vegetable* did not turn up enough material; hence the query formulation must be broadened. Not considering substitution of AI *Plant product* for AI.I *Vegetable,* the searcher might use the broadened formulation B2 *Frozen.* This is a drastic move, and the query formulation is probably too broad. The searcher should consider B2 *Frozen* AND AI *Plant product.*

Note again how the elements of facet B, *Fresh* and *Frozen,* are subdivided by the pattern of facet A, and vice versa; using the same pattern of subdivision for both *Fresh* and *Frozen* makes for consistency in the structure. Schemes such as LCC and DDC show many inconsistencies that could have been avoided through proper facet analysis.



V

Arrangement 2
Food source more important

A <u>Facet A Food Source</u>

- **A1 Plant product**
  - **A1B1 Fresh plant product NT A1.1B1, A1.2BI; BT B.1**
  - **A1B2 Frozen plant product NT A 1.1B2, A1.2B2; BT B2**
  - **A 1.1 Vegetable**
    - **A1.1B1 Fresh vegetable BT AIBI**
    - **A1.1B2 Frozen vegetable BT A1B2**
  - **A1.2 Fruit**
    - **A1.2 B1 Fresh fruit BTA1B1**
    - **A1.2 B2 Frozen fruit BT A1B2**

- **A2 Animal product**
  - **A2B1 Fresh animal product BT B1**
  - **A2B2 Frozen animal product BT B2**

B <u>Facet B Preservation</u>

- **B1 Fresh NT A1B1, A2B1**
- **B2 Frozen NT A1B2, A2B2**

(b) Representation as a linear sequence with cross-references. Arrangement 2.

Fig. 14.6 Hierarchical structure generated by two facets (repeated).

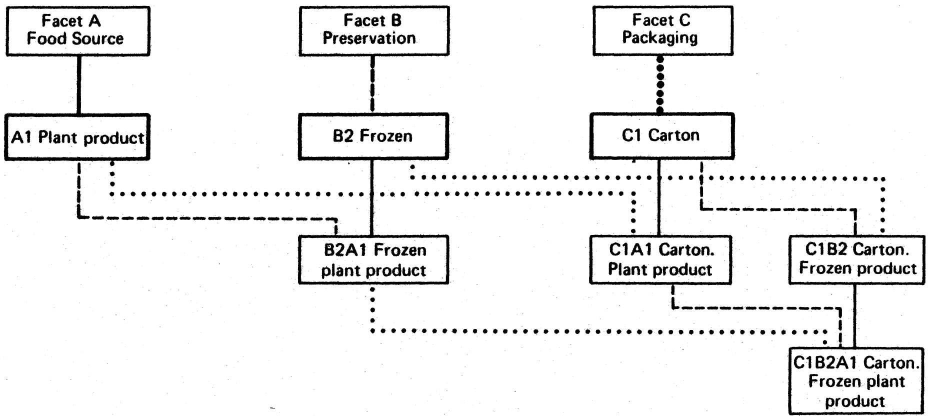
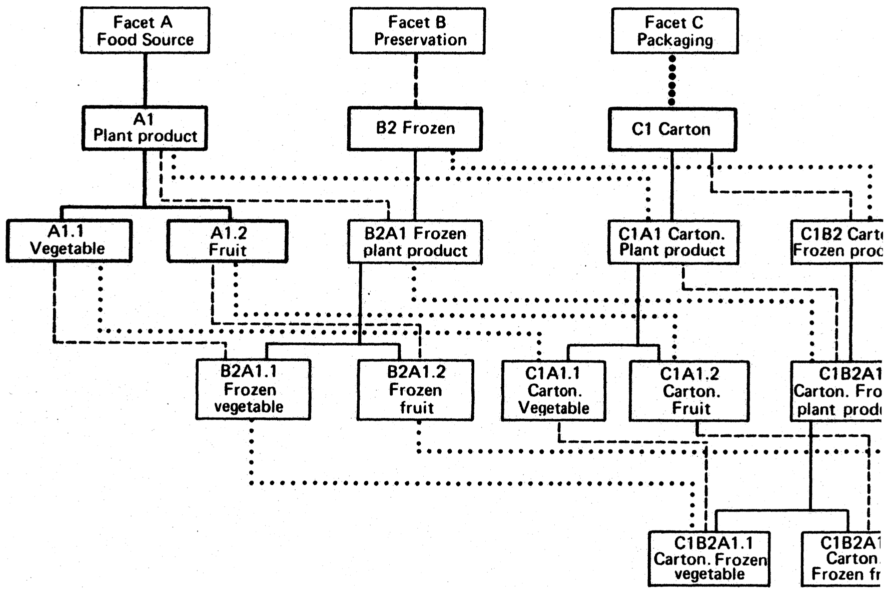

Facet A Food Source
Facet B Preservation
Facet C Packaging
A1 Plant product
B2 Frozen
C1 Carton
B2A1 Frozen plant product
C1A1 Carton. Plant product
C1B2 Carton. Frozen product
C1B2A1 Carton. Frozen plant product

Fig. 14.7 Hierarchical structure generated by three facets. No hierarchy within facets (only one concept in each facet).

Most subjects have more than two facets. Figure 14.7 shows a hierarchical structure generated by three facets, with just one concept in each facet for simplicity. One can now subdivide A1 *Plant product* by adding B2 *Frozen* (broken line) or by adding Cl (Packed in) *Carton* (dotted line). There is a third combination on the same level, C1B2 (Packed in) *Carton. Frozen.* Finally, one can combine all three concepts.

Figure 14.8 derives from Fig. 14.7 by adding Al.l *Vegetable* and A1.2 *Fruit* under A1 *Plant product.* Each of the three combinations in Fig. 14.7 that contain the component A1 *Plant product* is now subdivided into narrower concepts by substituting A1.1 *Vegetable and* A1.2 *Fruit,* respectively. A different way of looking at this structure is as follows: Starting with the structure generated by facets B and C, consisting of B2 *Frozen,* Cl *Carton,* and their one combination, Cl B2, one subdivides each of these three concepts by the pattern of facet A. One could also start with the structure generated by facets A and B and subdivide each of its seven elements by the pattern of facet C.



(a)

(a) Graphical representation.

Linear Arrangement (one of many possible)

A <u>Facet A. Food source</u>

A1 Plant product NT B2A1, C1A1

Al.l Vegetable NT B2A1.1, ClAl.l

A 1.2 Fruit NT B2A1.2, ClA1.2

B Facet B. Preservation

B2 Frozen NTC1B2

B2A1 Frozen plant product NT ClB2A1; BT A1

B2A1.1 Frozen Vegetable NT C1B2A1.1; BT Al.l

B2A1.2 Frozen fruit NT C1B2.A1.2; BT A1.2

C Facet C. Packaging

Cl Carton

C1A1 Carton. Plant product NT ClB2A1; BT Al

ClAl.l Carton. Vegetable NTC1B2A1.1; BT Al.l

Cl A 1.2 Carton. Fruit NT C1B2A1.2; BT A1.2

C1B2 Carton. Frozen BT B2

C1B2A1 Carton. Frozen. Plant product BT B2A1, ClA1

C1B2A1.1 Carton. Frozen vegetable BT B2A 1.1, ClAl.l

C1B2A1.2 Carton. Frozen fruit BT B2A1.2, C1A1.2

(b) Representation as a linear sequence with cross-references.

Fig. 14.8 Hierarchical structure generated by three facets. Hierarchy in Facet A. (Fig. 15. shows the same structure in a different format.)

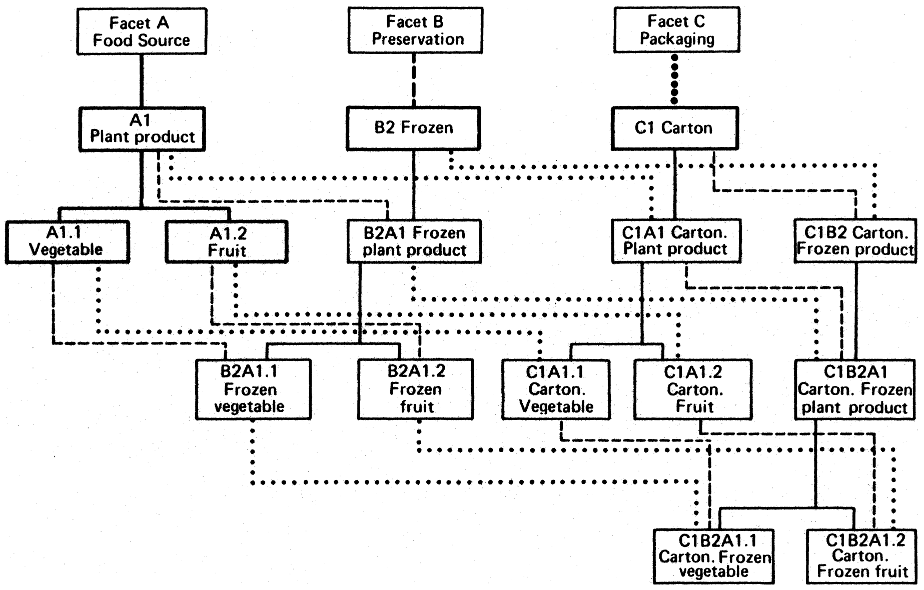


Figure 14.8a Hierarchical structure generated by three facets. Graphical representation (repeated).

In this structure one can follow the three kinds of hierarchical relationships (going from the top down). Starting from A1 *Plant product,* one finds A1.1 *Vegetable* and A1.2 *Fruit,* which are *narrower by autonomous subdivision;* and B2A1 *Frozen plant product* and C1A1 *Carton. Plant product,* which are both *narrower by combination* (adding the component B2 and CI, respectively). Continuing from B2A1 *Frozen plant product,* one finds B2A1.1 *Frozen vegetable* and B2A1.2 *Frozen fruit,* which are both *narrower by substitution,* and C1B2A1 *Carton. Frozen plant product,* which is narrower by combination (adding the component CI). This last concept can be further subdivided by substitution. To find broader concepts, one goes from the bottom up. Again, one can broaden a component to arrive at a concept broader by substitution, or drop a component to arrive at a concept broader by combination.

The following example shows a whole hierarchical chain generated by substitution:

	B2A	Frozen any food source	
	B2A1	Frozen plant product	
narrower	B2A1.1	Frozen vegetable	broader
	B2A1.1.2	Frozen beans	
	B2A1.1.2.3	Frozen garbanzo beans	

The bottom concept, *Frozen garbanzo beans* combines all *Frozen* products with a very narrow restriction with respect to facet A *Food source:* Only *Garbanzo beans* will do. A search with this query formulation finds only a tiny portion of all frozen products. The next level loosens the restriction somewhat: any type of *Beans* is satisfactory. The next level loosens the restriction still more, to *Vegetable* and—even broader—*Plant product.* A search now finds a sizeable portion, maybe more than half, of all frozen products. It is not such a big step, then, to be satisfied with *any* food source so that a search would find all frozen products. In short, starting from the very specific *Garbanzo beans,* the food source restriction is relaxed more and more until it fades away entirely. B2A *Frozen any food source* is the same as B2 *Frozen.* Broadening a concept by dropping a component turns out to be the last step in a series of substituting ever broader concepts in that component. Figure 14.9 illustrates this chain through nested Venn diagrams.

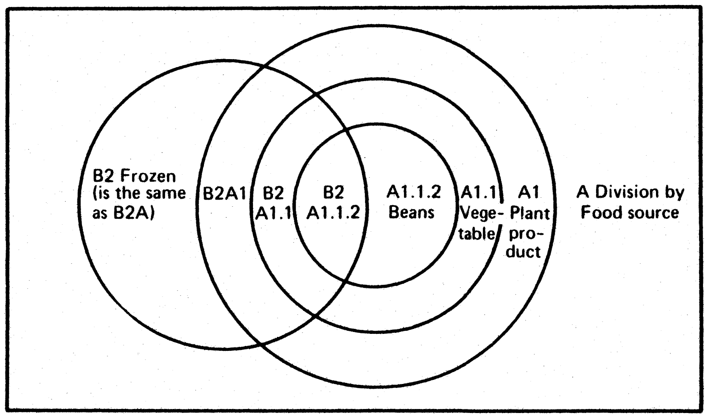


Fig. 14.9 Hierarchical chain generated through substitution.

The principles discussed in Sections 14.1-14.3 are basic to the understanding of index language structure and query formulation. The sample search discussed in the next section illustrates and clarifies these principles further.

14.4 APPLICATION AND ILLUSTRATION: SEARCHING

This section explores the effects of broadening and narrowing a query formulation and thus illustrates the application of hierarchy and concept combination to searching. It also introduces formally the idea of inclusive searching, which is closely wedded to hierarchy. This illustration uses a sample search in a small bibliographic ISAR system in the area of transportation. The index language used is shown in Fig. 14.10, a list of documents retrieved by various query formulations in Figs. 14.11 and 14.12. The query statement for the search is

Vehicles for rail transport.

The query formulation for this topic is straightforward:

E6 Vehicles AND B2 Rail transport.

Assume an ISAR system in which each document is shown in the index file only under the descriptors assigned to it. (Documents are not also shown under broader descriptors.) A search with this formulation finds the four documents indexed by B2 (see Fig. 14.12, the box labeled B2 Rail transport, general references). There are clearly other relevant documents in the collection; they are indexed by B2.3 *Intercity railroads* or B2.7 *Local rail transit.* To find these relevant documents, one should use the formulation:

E6 AND (B2 OR B2.3 OR B2.7).

This formulation asks for B2 OR any of its narrower terms. This type of query occurs often. The ISAR system should make the searcher's life easier by allowing for the query formulation:

E6 AND B2 Rail transport, *inclusive*

B2 *inclusive* means B2 OR any of its narrower terms. It is shorthand for (B2 OR B2.3 OR B2.7).

Sometimes one *wants* to restrict the search to documents indexed by B2 itself to find documents covering the whole area of rail transport, such as document 24. Then one should use

B2 Rail transport, *general references.*

B/D Division by mode of transportation
    BI Ground transport
        B2 Rail transport
            B2.3 Intercity railroad BT R4
            B2.7 Local rail transit BT R2
        B4 Road transport

E Division by facilities vs. vehicles
    E5 Methods to move persons or freight
        E6 Vehicles
            F Vehicles subdivided by power supply
            G Vehicles subdivided by type of propulsion
H Materials to build facilities or vehicles
J Passenger vs. freight transport
K Traffic operations
L Transportation providers
M Creation and maintenance of systems and components
N Organization and administration
Q General and other concepts
R Geographic range
S Geographic location

Fig. 14.10 A faceted classification for transportation.

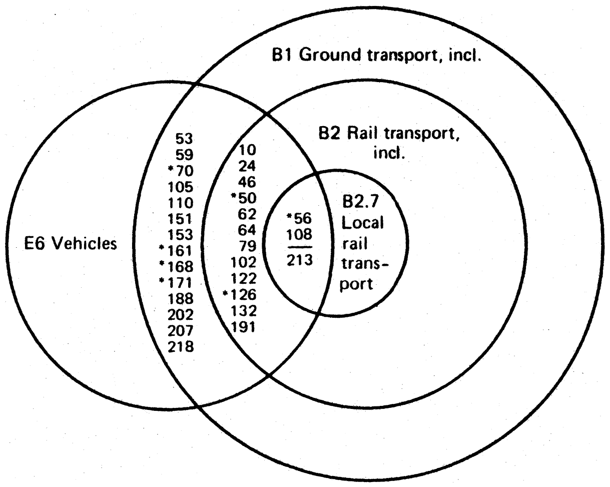


Fig. 14.11 Search results shown in Venn diagrams. (Documents marked by * are also relev for B2.7 *Local rail transit.)*

Query Formulation E6 Vehicles AND		Document	Indexed by E6, other descriptors, and by
[illegible]	B1 Ground transport, gen ref.	* 70 Electric vehicles, a bibliography	B1
		*161 High-speed ground transportation tube vehicle concept	B1
	B2 Rail transport, gen ref.	* 24 The concise encyclopedia of world railway locomotives (includes local rail transit locomotives)	B2
		* 50 Technical description of the Stockholm underground railway (should be indexed B2.7)	B2
		*126 Rolling stock for London Transports Victoria line (should be indexed B2.7)	B2
		191 Turbotrain (should be indexed B2.3)	B2
[illegible]	B2.3 Intercity railroad	10 Turbo train appraisal	B2.3
		46 Prediction of domestic air traffic passengers Abstract: Examines passenger preference between super express train and air travel	B2.3
		62 Progress in railway mechanical engineering	B2.3
		64 Le turbo train des Chemins de Fer Nationaux du Canada	B2.3
		79 Aerodynamics of high speed train	B2.3
		102 Diesel-electric locomotive handbook	B2.3
		108 The Alaska Railroad	B2.3 (also B2.7)
		122 The official railway equipment register (only intercity railroads)	B2.3
		132 Advanced passenger train	B2 3
[illegible]	B2.7 Local rail transit	* 56 Teito Rapid Transit Authority's automatic train operation	B2.7
		108 The Alaska Railroad (B2.7 erroneous descriptor)	B2.7 (also B2.3)
		*213 The rapid tramway	B2.7
[illegible]	B4 Road transport	53 National tank truck carriers directory	B4
		59 Russel's official national motor coach guide	B4
		105 An origin-destination study of truck traffic in Michigan	B4
		110 Where buses face air competition	B4
		151 Operation team valley Abstract: Bus service to new development	B4
		153 Truck equivalency	B4
		168 National electric automobile symposium	B4
		171 A system for rapid transit on urban freeways	B4
		188 Satellite airport systems and community	B4
		202 Specialized motor carriage	B4
		207 The downtown parking system	B4
		218 Official motor carrier directory	B4

Fig. 14.12 Inclusive mode versus general reference mode. (Documents marked by * are also relevant for B2.7 *Local rail transit.)*

To give another example, a query for *Psychology of marketing* should be formulated as *Psychology, inclusive* AND *Marketing* to find documents on people's attitudes, information-gathering habits, readability of advertisements, and decision making processes—all narrower terms of *Psychology.* On the other hand, when looking for an introductory psychology text, one should use *Psychology, general references.* The user is not interested in any document that deals with just one or a few narrower terms of *Psychology;* he wants only documents that deal with the whole of *Psychology.* Having just one mode to search for a broad descriptor like *Psychology* would never do: If the one mode is defined as *Psychology, general references,* the psychology of marketing search would require a very large OR-combination of *Psychology* and all its narrower terms. If the one mode is defined as *Psychology, inclusive,* the textbook search would have very low discrimination (there are many textbooks on subfields of psychology).

In summary, a descriptor that has narrower descriptors under it can be used in two modes in searching. The descriptor in the *general references* mode finds just the entities indexed by the descriptor itself. The descriptor in the *inclusive mode* finds all entities that are indexed by the descriptor itself *or any of its narrower descriptors.*

Inclusive searching can be implemented in two ways. One way is *expanding the query term* (e.g., B2 *Rail transport, inclusive),* resulting in the OR-combination of the appropriate narrower descriptors. The other way is *generic posting* in building the index file: Posting refers to the operation of showing a document (e.g., document 56) under one of its descriptors (e.g., *Till Local rail transit)* in the index (e.g., punching hole number 56 in the peek-a-boo card for B2.7, or listing document 56 with B2.7 in a printed index). Generic posting is posting a document to a descriptor broader than one of its assigned descriptors—for example, posting document 56 to B2 *Rail transport* (broader than B2.7) and to B1 *Ground transport* (still broader). Generic posting in a peek-a-boo file results in a peek-a-boo card for a broad descriptor in the inclusive mode, for example, B2 *Rail transport, inclusive.* But the original peek-a-boo card showing just the documents indexed by B2 itself—B2 *Rail transport, general references—should* also be preserved. The following table illustrates this.

Index term	Entries in the index file		
	First entry	Entries due to generic posting	
B2.7	B2.7	B2 inclusive	B1 inclusive
B2	B2 general references	B2 inclusive	B1 inclusive

We now return to the query on Vehicles for rail transport. Assume it requires very high recall. The query formulation should be broadened, for example, by substituting B1 *Ground transport, inclusive* for B2 *Rail transport, inclusive:*

E6 Vehicles AND B1 Ground transport, inclusive.

Figures 14.11 and 14.12 show the additional documents retrieved. Some of them are

S3 National tank truck carriers directory,
indexed by B4 Road transport
59 Russell's official national motor coach guide, indexed by B4
70 Electric vehicles, a bibliography
indexed by B1 Ground transport
105, 110, 151, 153: some more documents on trucks and busses.
161 High speed ground transportation tube vehicle concept .
indexed by B1 Ground transport
168, etc., some more documents on trucks, busses, and cars
indexed by B4 Road transport

Document 70 appears relevant because it covers vehicles for all kinds of ground transportation; document 161 appears relevant because it cover: vehicles for a specific type of ground transportation that is closely related tc rail transportation *(Tube transport* is not in the index language). The othei documents found are not relevant. With this particular query formulation, *i* slight increase in recall is paid for with a large decrease in discrimination; B1 *Ground transport, inclusive* brought in not only the two additional relevam documents indexed by B1 but also the many, many irrelevant documents in dexed by B4 *Road transport.* However, we can do much better; the following query formulation will add only the two additional relevant documents:

E6 AND B2 inclusive OR B1 Ground transport, general references

Now assume that you are to search for the topic

Vehicles for local rail transit.

In Figs. 14.11 and 14.12 the relevant documents are marked with an asterisk A good query formulation for this topic should follow the principles dis cussed.

14.5 CONCEPTUAL ANALYSIS, FACET ANALYSIS: ELABORATION

14.5.1 Developing a Scheme of Facets

A scheme of facets is very helpful for semantic factoring and for organizing elemental concepts into a coherent structure. But deriving such a scheme—or facet frame—is a difficult task. It is the task of discerning the structure of a field from an unwieldy set of 10,000 concepts compiled from patient histories, document titles and abstracts, query statements, or any other source.

This task can be accomplished with the *bottom-up approach:* The designer first factors each concept in the list as far as possible, resulting in a list of elemental concepts, and then arranges the elemental concepts into facets with hierarchical structure within each facet.

The task can also be accomplished with the *top-down approach:* The designer first examines the phenomena in a subject field, resulting in a facet frame. For example, medicine deals with the *Organisms affected*—which in turn can be characterized by *Species* (including humans), *Age,* and *Sex;* the *Organ, organ system, or body region* affected; the *Type of disease; Disease causes,* including disease causing agents; and *Therapeutic measures,* including drugs. Next the designer develops for each facet a hierarchically structured list of elemental concepts, starting from major subdivisions and working downward.

A *combination approach* is best. Problem analysis leads to a preliminary facet frame. A number of obvious concepts suggest themselves for each facet. The designer uses this preliminary classification as a guide in factoring the concepts of the original list, adding new elemental concepts, and even new facets, as needed.

A universal scheme of facets emerges from the comparison and integration of several schemes developed for individual subject fields. For example, the food scheme has a facet *Food source,* which covers the organisms used in the production of food. The same list of organisms can be used in medicine for the facet *Organism affected. Anatomical part* is another facet that medicine and food have in common. The food facets *Physical state* and *Physical form* are clearly very general, and so on.

14.5.2 Recognizing General Concepts

Semantic factoring leads to general concepts that are not explicitly recognized in a field but that may be quite useful, not only for retrieval but also for general discourse in the field. For example, *Railroad station, Harbor,* and *Airport* all contain a common aspect (what is left after extracting the semantic factors *Rail transport, Water transport,* and *Air transport,* respectively). Since there is no name for this general concept, the designer invents one: *Traffic station.* In a search for passenger handling in an airport, documents on passenger handling in railroad stations might be quite useful; thus the topic may be more appropriately stated as passenger handling in traffic stations.

Another example comes from the area of alcoholic beverages—more specifically, distilled spirits. Applicable government regulations define *Neutral distilled spirits,* which are distilled to such a degree that no distinctive flavor or aroma is left. Then they define a long list of other distilled spirits, such as *Whiskey, Rum, Brandy,* and *Tequila.* These are all compound concepts: The semantic factors corresponding to the food source *{Grain, Sugarcane, Plant producing fruit or berry,* and *Agave,* respectively) can be extracted easily. A general concept is needed to be used as the other semantic factor; since the subject field does not provide such a concept, the designer creates *Distinctive distilled spirits.*

Sometimes a general concept—perhaps not new but rather so obvious as to be easily overlooked—is detected "through contrast" in anew facet. For example, consider the original concept *Reading instruction for deaf children.* One semantic factor is *Deaf,* an elemental concept new to the scheme. It does not fit into any of the existing facets, so the designer introduces

```
Person by presence or absence of handicap
  Deaf
```

Immediately she sees that this needs to be fleshed out:

```
Person by presence or absence of handicap
  Handicapped
    Physically handicapped
      Hard of hearing, deaf
      Visually handicapped, blind
    Mental handicapped
  Not handicapped
```

Handicapped does not cover the universe of persons. The designer must introduce *Not handicapped* to make the facet complete. *Not handicapped* is easily overlooked both in index language construction and in indexing and searching. A teacher just looking for *Reading instruction* may have no use for material on reading instruction for the handicapped since it does not apply to his or her situation; the teacher should be able to search for

Reading instruction AND NOT handicapped

Option 2 is to omit the hierarchical relationship and please the restrictive group. In that case, the designer should at least introduce the associative relationship

Grain RT Corn

A user with an expansive definition of *Grain* could then use the query formulation

Grain, inclusive OR Corn

The associative relationship reminds this user to add *OR Corn.* Another example of the same problem is what relationship should be established between *Fruit* and *Tomato.* The index language designer should choose the relationship that minimizes total user effort.

14.6,2 Types of Hierarchical Relationships

The pragmatic definition of hierarchical relationships given in Section 14.1 is not concerned with the details of the "meaning" of a relationship. However, an examination of the meaning of relationships leads to a better understanding of their nature and is useful for the development of the hierarchical structure. The most important types of hierarchical relationships are the following.

Class Inclusion

A relationship of the form class/subclass or class/member of a class (hierarchy in the logical sense). For example,

Vegetable NT Leafy vegetable
Leafy vegetable NT Spinach

A logically narrower concept has all the characteristics of the broader concept and, in addition, at least one further characteristic. Thus, we can always say, "ß (the narrower concept) is an *A* (the broader concept) that has the characteristic C." For example: *B Leafy Vegetable* is an *A Vegetable* with the characteristic C *The leaves are eaten.*

Topic Inclusion

A relationship between two areas of knowledge, one including the other. For example,

Psychology NT Personality
Science NT Physics NT Optics

Whole—part

A relationship between two physical objects, one including the other. For example,

Automobile NT Automobile engine

(Note that we do not say *Automobile* NT *Engine,* because there are many engines that are not part of an automobile.)

Possible use

A relationship based on the possible use of a substance or idea. For example,

Solvent NT Alcohol
(Solvent is a possible use of *Alcohol)*

Other types of relationships

A relationship between two concepts that does not fall under any of the previous types but is nevetheless useful as hierarchical relationship. For example,

Electron tubes NT Characteristic curve of electron tubes

Some rules for the construction of index languages use a strict logical definition of hierarchy, admitting only class inclusion relationships, and excluding, for example, whole-part and use relationships. However, hierarchy serves a purpose and hierarchical relationships should be introduced whenever they serve this purpose, that is, whenever they serve any of the functions listed in Section 13.6.

14.6.3 Introducing New Broader Concepts

Building a hierarchy often leads to an awareness of gaps that should be filled by the introduction of a new, broader concept, particularly for searching. For example, a classification of *Government and politics* may have arrayed the following concepts:

The state
Constitution
Legislative body
Administration, executive branch

The designer feels that searches for the OR-combination of these four concepts may be frequent and therefore introduces a new, broader concept—*The state, broadly defined*—to include them all. This concept, once introduced, is also useful for indexing comprehensive documents. (The Medical Subject Headings Tree Structures include a number of such broader concepts; but these are available only for searching, not for indexing, as indicated by the label *Non-MeSH.* But a separate type of descriptors that are available only for searching is confusing and should be created only if the number of descriptors in a system is severely limited, which is not the case for MEDLINE.)

In the next example the individual concepts to be included under a new, broader concept are part of different subject areas, where they remain as narrower terms also.

Relation to culture, broadly defined
- Relation to own culture (Culture)
- Relation to other culture (Culture)
- Informal education (Education)
- Socialization of the individual (Sociology)
- Adaptation—readaptation (Sociology)
- Culture and personality (Social psychology)
- Attitudes, opinions (Social psychology)

Again the designer expects many searches in which all of these concepts would be ORed; introducing the broader concept makes the searcher's life much easier . Instead of entering seven descriptors connected by OR, he or she enters just one descriptor. Moreover, the searcher working without the benefit of this new, broader concept may well forget one or the other of the narrower concepts included. The new, broader concept is also useful to the indexer who considers, for example, using the descriptor *Informal education.* Following the BT cross-reference to *Relation to culture, broadly defined,* he or she finds a list of candidate descriptors to be used in addition to or instead of *Informal education.*

This example illustrates very clearly that hierarchy building must go beyond arranging a set of concepts in a neat structure; the designer must identify all concepts and all relationships that can help the indexer or searcher.

Sometimes a broad concept can replace several narrower concepts in the index language. If the number of descriptors in the index language is to be kept to a minimum, this may provide an impetus for finding new broad concepts. For example, consider the newly formed concept

Stonework, glass, ceramics
- Stonework
- Glass
- Ceramics •

It can replace the three narrower concepts:

Stonework USE BT Stonework, glass, ceramics

and likewise for *Glass*..and *Ceramics.*

Broad concepts also serve as headings to clarify the arrangement of th descriptors in the display of an index language. This is very important, pai ticularly for the checklist technique of indexing and query formulatior Sometimes the impetus for introducing a new, broader concept comes fror the need for such a heading. For example,

Employment of specific groups
  Employment of children
  Employment of women
  Employment of handicapped persons
  Employment of jail inmates

Meat part by presence of bone
  With bone
  Boneless

Broad concepts introduced originally for their value as headings are als useful for indexing comprehensive documents. Some systems, such as LC and Thesaurofacet, introduce broader concepts as headings but do not ii elude them in the index language as descriptors; this is confusing.

14.7 CONCEPT FORMATION IN THESAURUS BUILDING

The major concern in building an index language or thesaurus is tl development of a conceptual framework that mediates between the search and how he expresses his interest on the one hand and the information pr vided by the author and how she expresses her findings on the other. Tl must be accomplished within the limitations of the ISAR system at han especially within constraints on the size of the index language. This task r quires thorough conceptual analysis and the formation of new concepts; challenges the creativity of the thesaurus builder.

New concepts are formed in consolidating quasi-synonyms (Section 12.1 in semantic factoring or facet analysis (Section 14.5.2), and in building t hierarchy (Section 14.6.3).

Careful facet analysis reveals *cross-disciplinary concepts;* such concej should be "pitched at the level of abstraction permitting them to embra concepts that are substantially identical and whose differences are largely consequence of the idiosyncrasies of the fields in which they are used." Su concepts would contribute both to the efficiency of ISAR systems—by m2 ing it possible to reduce the number of descriptors in the index language—and to the "transferability of knowledge across disciplines."

Other problems of concept formation occur in ISAR systems that deal with socioeconomic information from different countries, for example, an ISAR system that deals with information on education in the *United States, France,* and *Germany.* Each country has its own structure of educational institutions. The challenge is to develop a common structure that is applicable to all countries. This would allow for a reduction in the number of descriptors. But even if we retain each country's terms in the index language, the newly developed common structure facilitates searching for general concepts, such as *Elementary schools,* in all countries. Such a common structure is also essential for the gathering of comparative educational statistics. A lot of careful work on definitions is needed to establish the entries in the common structure, as anybody having worked in or with comparative statistics can testify. The thesaurus builder should rely on work done by experts in comparative education in this instance.

A last example of concept formation in thesaurus building is the typology of international organizations given in Fig. 14.13. This typology was the result of the joint efforts of a classificationist (who contributed the approach of facet analysis) and a subject expert. Once this analysis is completed, it becomes clear that the facets derived, except for facet 4, are useful in a much wider context.

A further aspect of concept formation in thesaurus building and particularly in thesaurus updating is the broadening of the meaning of descriptors that occurs as they are used.

This finishes the discussion of the conceptual aspects of index language structure. The next chapter examines index language structure with emphasis on its functions in data base organization.

Facet 1: International organizations by level
- Private international organizations
- Quasi-governmental international organizations
- Governmental international organizations

Facet 2: <u>International organizations by membership</u>
- Universal membership
  - SN (Scope Note)
    - No restrictions as to geographical location; political system, main religion, or other characteristics of membership countries
- Limited membership
  - SN Members only from one region or, for example, from Islamic countries or industrial countries

Facet 3: <u>International organizations by scope and orientation</u>
- Covers entire range of politics
  - SN For example, United Nations; International Federation of Socialist Parties
- Covers only specific function
  - SN For example, World Health Organization; International Federation of Documentation

Facet 4: International organizations by internal cohesion
- SN Basic tendency, not momentary developments
- Loose groupings
- Cohesive organizations

Facet 5; <u>International organizations by organizational structure</u>
- Centralized structure ~
- Decentralized structure

Fig. 14.13 Facet analysis: Typology of international organizations.

CHAPTER I!

Index Language Structure 2 Data Base Organizatior

OBJECTIVES

The objectives of this chapter are to explain the relationship between th conceptual structure of the index language on the one hand and data bas organization on the other with a view to indexing and searching as well as t< system design; to elucidate the structure of traditional classificatioi schemes, such as DDC and LCC, thus enabling the reader to put them to bes use and to prepare user aids; and to set forth design characteristics for inde languages/classification schemes and their implications for index languäg use.

INTRODUCTION

While the distinction between conceptual structure and data base 01 ganization is important, in practice they both work together in the task o storage and retrieval; they are closely intertwined and one cannot be treate without reference to the other. While Chapter 14 emphasized conceptus aspects of index language structure, it referred to implications for data bas organization. This chapter applies the principles of conceptual structure t the analysis of data base organization and the closely related problem c presenting an index language, particularly an index language that include many precombined descriptors.

Sections 15.1-15.3 deal with general principles: a statement of the prob lem to be solved by data base organization, the principle of grouping entitic for ease of retrieval, and the relationship of grouping of entities to thei description. Sections 15.4-15.6 deal with practical applications: selection c precombined descriptors considering the search mechanism available organization of the index language so indexers and searchers can find th descriptors needed (descriptor-find index and designation and arrangemer of descriptors), and use of a unified index language for different searc mechanisms.

15.1 THE PROBLEM

A data base contains many entities linked through relationships. The data base organization must support retrieval of entities based on their relationships to other entities (see Chapters 3 and 11). This chapter concentrates on a specific case: the retrieval of focal entities, such as documents or food products, based on their relationships to just one other type of entity, namely, concepts (subjects). Each focal entity can be seen as linked to one very compound concept, made up of many components and called entity representation, to be matched against the search concept, usually a less compound concept called query formulation.

Example 1

Document concepts

1. Methods of reading instruction in first grade, good document
2. Needs for funding for the use of microcomputers in first grade reading instruction for visually handicapped children in New York City in 1984, good document

Document representations in terms of elemental concepts

1. Method of instruction; Reading; First grade; Good
2. Method of instruction; Reading; First grade; Good; Handicapped; Eyesight; Microcomputers; Funding; Needs assessment; New York City; 1984

Query formulation	Documents to be retrieved
Method of instruction AND Reading AND First grade AND Good AND Handicapped	2
Method of instruction AND Reading AND First grade AND Good (coextensive with document 1)	1,2
Method of instruction AND Reading AND First grade	1,2
Method of instruction AND Reading	1,2
Method of instruction AND Language	1, 2
Method of instruction AND Language AND Elementary school	IV 2
Method of instruction AND Handicapped	2
Equipment AND Elementary school	2
Funding AND Education AND NY City AND 1984	2

Example 2

- *Query formulation* (broad, asks for an aggregate number)
  Retail value AND Sales AND Large electrical appliances AND Florid
  AND 1980
- *Money number concept* (specific, to be included in aggregate)
  Retail value of refrigerators sold in Miami in July, 1980
- *Money number representation*
  Retail value; Sales; Refrigerators; Miami; July, 1980

To compute this number, the system must retrieve all money numbers wit) a representation narrower than the query formulation and then add them,

In each case the system must retrieve all entities that have a representatioi equal to or narrower than the query formulation. To visualize this process picture a wall-size hierarchy graph with concepts representing entitie and/or queries. Locate the query concept and from it follow all hierarchica lines all the way down, assembling a subset of concepts. Find all entitie linked to (represented by) any concept in the subset. The organization of th data base must support quick identification of all such entities at a reason able cost and within the constraints set by the technical device used (e.g., th printed page, a card catalog, or computers).

15.2 GROUPING ENTITIES. SEARCHING IN GROUPED FILES

15.2.1 The Idea of Grouping and Precombined Descriptors

This section explores the idea of grouping entities to aid in retrieval and it implication for the nature of the index language. The simplest ISAR systen is one without grouping, in which entities are indexed by (directly linked to elemental concepts. Compound concepts do not occur explicitly in the dat< base; they do not belong to the domain of the entity type *Concept.* However the linkage of entities to compound concepts is present in the data base im plicitly; all documents related to a compound concept, such as *Reading in struction,* can be found with the query formulation

Method of instruction AND Reading

Likewise, all food products related to the compound concept *Frozen cu beans,* can be found with the query formulation

Beans AND Cut into medium-sized pieces AND Frozen.

The retrieval mechanism (e.g., computer or peek-a-boo cards) then quickl identifies all entities indexed by the requisite elemental descriptors.

In such a system compound concepts are not used in indexing but are formed later in query formulation; hence the system is called a *postcombination system.* A sample system (main entity type: document) is shown in Fig. IS. 1. Figure 15.1a is the main file (a list of documents); Fig. 15. lb is a subject index (a peek-a-boo file here shown as a printed index); and Fig. 15.1c shows some query formulations and the documents retrieved by them.

A postcombination system permits searching for any and all combinations of elemental descriptors. The searcher can specify just the topic she is looking for and thus attain high discrimination. The principles of conceptual structure can be used directly in formulating the query in a straightforward manner. But postcombination is not possible in a card catalog, in a printed index, or in the retrieval of entities from shelves. Furthermore, a postcombination system requires many entries per entity in the index file, making for large index files. Each entity retrieved in the index by its accession number must be found separately in the main file, and with very large collections this may lead to excessive effort even in a computer system. Hence we are looking for ways to organize retrieval more efficiently.

One hint comes from the analysis of the sample peek-a-boo system shown in Fig. 15.1. Assume it accommodates 5000 documents (5000 hole positions, one document per position). What if the collection grows beyond that limit? Could the system somehow accommodate more than 5000 documents within the limits of 5000 hole positions per card without losing retrieval power? Examine the document representations in Fig. 15.1a. Pay close attention to the descriptors assigned. What about documents 1 and 5? 6 and 7? The descriptors assigned to documents 1 and 5 are identical. Consequently, the two documents exhibit identical retrieval behavior: Whenever document 1 is retrieved, document 5 is retrieved also; whenever document 1 is not retrieved, document 5 is not retrieved either. Using two different peek-a-boo card holes for these two documents is a waste of capacity; using number 1 for *both* documents frees hole number 5.

Accommodating more than 5000 documents is made possible by grouping—putting all documents having exactly the same descriptors into one group—and letting each peek-a-boo hole position correspond to a whole group rather than to an individual document. This is illustrated in Fig. 15.2; Figure 15.2a gives a list of the documents arranged according to the groups formed (the main file), and Fig. 15.2b shows the corresponding index (given here in lieu of a peek-a-boo file). The sample searches given in Fig. 15.2c show how the system works: A query formulated in terms of elemental descriptors serves to find group numbers in the index (Fig. 15.2b); the group numbers serve to find individual documents in the main file (Fig. 15.2a).

Since only documents having exactly the same elemental descriptors and thus exhibiting identical retrieval behavior are allowed in a group, a search via the groups retrieves exactly the same documents as a search in the original system shown in Fig. 15.1. Due to the very strict criterion, groups are small; in a real-life collection in which only the most important descriptors are assigned to a document, one might expect groups of 2-5 documents on the average. (If more and/or more specific descriptors are assigned, indexing reflects minor differences between documents and thus group size decreases and the number of groups increases.)

Both data base organizations can be viewed as essentially the same: The searcher consults an index file, finds numbers, looks under those numbers in the main file, and finds documents. The only difference is that in the data base organization in Fig. 15.1 the searcher finds just one document under each number, whereas in Fig. 15.2 he finds one or more documents since now *a number refers to a document group.* For example, in search c2 the searcher must locate four documents in the main file in Fig. 15. la but only two groups in the main file in Fig. 15 .2a. This could be viewed simply as a technical convenience: The index file is smaller by a factor of two or three, and as a bonus, each access to the main file nets two or three documents instead of just one. The system in Fig. 15.2 is still viewed as a pure postcombination system in which documents are indexed by (linked directly to) elemental descriptors, which are combined in searching to retrieve suitable documents.

However, the same physical data base can be viewed as a quite different system in which the nature of the main file has changed radically from the simple list of documents arranged by accession number given in Fig. 15.1a to the grouped list of documents given in Fig. 15.2a. The simple list of documents has no order with respect to subject. The grouped list brings together documents that deal with or are relevant for a given topic. This topic is expressed as a combination of elemental concepts, which is the same for all documents in the group; this combination of elemental concepts is the *group representation.* Put differently: For purposes of retrieval, each document is linked directly to a highly compound concept, such as concept #7:

#7 Curriculum; Biology; First grade; Good document.

It is linked to elemental concepts, such as *Biology*, indirectly via a chain such as

```
Document 1      0      #      7      ----————> Biology
                < deals with > <has component >
Document 10 <——-------—- —#7              ~ Biology
               < is treated in > < is component of >
```

Doc. no.	*Descriptors*
1	Method of instruction; Reading; First Grade; Good
2	Traffic station; Inland water transport; Medium
3	Method of instruction; Reading; First grade; Bad
4	Curriculum; Science; Fourth grade; Medium
5	Method of instruction; Reading; First grade; Good
6	Traffic station; Ocean transport; Freight; Bad
7	Traffic station; Ocean transport; Freight; Bad
8	Traffic station; Inland water transport; Freight; Medium
9	Curriculum; Biology; First grade; Good
10	Method of instruction; Reading; Elementary School; Bad
11	Curriculum; Biology; First grade; Bad
12	Curriculum; Biology; First grade; Medium
13	Curriculum; Biology; Elementary school; Medium
14	Method of instruction; Reading; Ninth grade; Good
15	Method of instruction; Physics; Fifth grade; Bad
16	Method of instruction; Biology; Sixth grade; Medium
17	Method of instruction; Reading; Elementary school; Bad
18	Method of instruction; Reading; First grade; Bad
19	Curriculum; Reading; Second grade; Bad
20	Curriculum; Biology; First grade; Good
21	Curriculum; Science; Fourth grade; Medium

(a) List of documents (main file)

Education
General concepts
  Curriculum 4,9,11,12,13,19, 20,21
  Method of instruction 1,3, 5,10,14,15,16,17,18
Subject
  Reading 1,3,5,10,14,17,18,19
  Science, incl. 4,9,11,12,13, 15.16.20.21
    Science, gen. ref. 4,21
    Physics 15
    Biology 9,11,12,13,16,20
Grade level
  El. school, incl. 1,3,4,5, 9,10,11,12,13,15,16,17,18, 19.20.21
    El. school, gen. ref. 10, 13,17
    First grade 1,3,5,9,11, 12,18,20
    Second grade 19
    Fourth grade 4,21
    Fifth grade 15
    Sixth grade 16
  Junior high, incl. 14
    Junior high, gen. ref.
    Ninth grade 14

Transportation
Traffic facil. vs. vehicles
  Traffic station 2,6,7,8
Mode of transportation
  Water transport, incl. 2,%t 7,8
    Water transport, gen. reJ
    Ocean transport 6,7
    Inland water transp. 2,
Passenger vs. freight transp.
  Freight 6,7,8
Document quality
Good 1,5,9,14,20
Medium 2,4,8,12,13,16,21
Bad 3,6,7,10,11,15,17,18,15

(b) Index to documents

cl Method of instruction AND Reading AND First grade AND Good
  Doc. no. 1,5

c2 Method of instruction AND Reading AND First grade
  Doc. no. 1,3,5,18

c3 Method of instruction AND Reading AND El. School, incl.
  Doc.no. 1,3,5,10,17,18 ~~~ “

c4 Method of instruction AND Reading
  Doc.no. 1,3,5,10,14,17,18

c5 Biology AND First grade AND (Good OR Medium)
  Doc. no. 9,12,20

c6 Elementary school, incl. AND Good
  Doc. no. 1,5,9,20

(c) Sample searches.

Fig. 15.1 No grouping. Extreme postcombination

Group #1	**Method of instruction; Reading; First grade; Good**
	#1.1 Doc. 1 Method ; Reading; First grade; Good
	#1.2 Doc. 5 Method ; Reading; First grade; Good
Group #2	**Traffic station; Inland water transport; Medium**
	#2.1 Doc. 2 Tr. station; Inland water transp.; Medium
Group #3	**Method of instruction; Reading; First grade; Bad**
	#3.1 Doc. 3 Method ; Reading; First grade; Bad
	#3.2 Doc. 18 Method ; Reading; First grade; Bad
Group #4	**Curriculum; Science; Fourth grade; Medium**
	#4.1 Doc. 4 Curriculum; Science; Fourth grade; Medium
	#4.2 Doc. 21 Curriculum; Science; Fourth grade; Medium
Group #5	**Traffic station; Ocean transport; Freight; Bad**
	#5.1 Doc. 6 Tr. station; Ocean transport; Freight; Bad
	#5.2 Doc. 7 Tr. station; Ocean transport; Freight; Bad
Group #6	**Traffic station; Inland water transp.; Freight; Medium**
	#6.1 Doc. 8 Tn station; Ini. w. transp.; Freight; Med.
Group #7	**Curriculum; Biology; First grade; Good**
	#7.1 Doc. 9 Curriculum; Biology; First grade; Good
	#7.2 Doc. 20 Curriculum; Biology; First grade; good
Group #8	**Method of instruction; Reading; El. school; Bad**
	#8.1 Doc. 10 Method; Reading; El. school; Bad
	#8.2 Doc. 17 Method; Reading; EL school; Bad
Group #9	**Curriculum; Biology; First grade; Bad**
	#9.1 Doc. 11 Curriculum; Biology; First grade; Bad
Group #10	**Curriculum; Biology; First grade; Medium**
	#10.1 Doc. 12 Curriculum; Biology; First grade; Medium
Group #11	**Curriculum; Biology; Elementary school; Medium**
	#11.1 Doc. 13 Curriculum; Biology; EL school; Medium
Group#12	**Method of instruction; Reading; Ninth grade; Good**
	#12.1 Doc. 14 Method; Reading; Ninth grade; Good
Group#13	**Method of instruction; Physics; Fifth grade; Bad**
	#13.1 Doc. 15 Method; Physics; Fifth grade; Bad
Group#14	**Method of instruction; Biology; Sixth grade; Medium**
	#14.1 Doc. 16 Method ; Biology; Sixth grade; Medium
Group#15	**Curriculum; Reading; Second grade; Bad**
	#15.1 Doc. 19 Curriculum; Reading; Second grade; Bad

(a) List of narrow groups with documents.

<u>Education</u>
General concepts
  Curriculum #4,#7,#9,
    #10,#11,#15
  Method of instruction #1,
    #3,#8,#12,#13,#14
Subject
  Reading #1,#3,#8,#12,#15
  Science, inch #4,#7,#9,#10,
    #11,#13, #14
    Science, gen.ref. #4
    Physics #13
    Biology #7,#9,#10,#11,#14
Grade level
  El. school, incl. #1,#3,#4,
    #7,#8,#9,#10,#11,#13,#14,
    #15
    El. school, gen. ref. #8,#11
    First grade #1,#3,#7,#9,#10
    Second grade #15
    Fourth grade #4
    Fifth grade #13
    Sixth grade #14
  Junior high, incl. #12
    Junior high, gen.ref.
    Ninth grade #12

<u>Transportation</u>
Traffic facil. vs. vehicles
  Traffic station #2,#5,#6
Mode of transportation
  Water transport, incl. #2,#5,#6
    Water transport, gen.ref.
    Ocean transport #5
    Inland water transp. #2,#6
Passenger vs. freight transp.
  Freight #5, #6
Document quality
Good #1,#7,#12
Medium #2,#4,#6,#10,#11,#14
Bad #3,#5,#8,#9,#13,#15

(b) Index to narrow groups (descriptor-find index).

cl <u>Method of instruction AND Reading AND First grade AND Good</u>
  Group no. #1
  Doc. no. 1,5

c2 <u>Method of instruction AND Reading AND First grade</u>
  Group no. #1, #3
  Doc. no. 1,5, 3,18

c3 <u>Method of instruction AND Reading AND El. School, incl.</u>
  Group no. #1, #3, #8
  Doc.no. 1,5, 3,18, 10,17

c4 Method of instruction AND Reading
  Group no. #1, #3, #8, #12
  Doc.no. 1,5, 3,18, 10,17, 14

c5 <u>Curriculum AND Biology AND First grade AND (Good OR Medium)</u>
  Group no. #7, #10
  Doc. no. 9,20, 12

c6 <u>Elementary school, incl. AND Good</u>
  Group no. #1, #7
  Doc. no. 1,5, 9,20

(c) Sample searches.

Fig. 15.2 Narrow groups. Extremely high precombination.
All documents within one group have exactly the same representation

The searcher profits from the order in the main file; he or she needs to access the main file in fewer places (in search c2, this means two group numbers instead of four document numbers). This is particularly important if a file access involves not just looking in a list but inserting a different microfiche in a reader or walking to a place in the stacks.

The order in the grouped main file does not come without a price. Whenever a new entity comes in and has been indexed, its representation in terms of elemental descriptors must be compared with all the group representations established so far. If an exact match is found, the entity is assigned to the existing group; otherwise, a new group is established.

Saying that an entity belongs to group #7 gives exactly the same information as indexing it by the four elemental concepts that make up #7. So the elemental concepts might as well be omitted once they have led to the group. Going one step further, the indexer could skip the step of indexing a new entity by elemental descriptors and instead compare it directly with the group representations. *If he finds a group topic that matches exactly,* he indexes the entity with that group number—that is, he links it to the specific topic that is expressed by the group representation. The group number serves as a precombined descriptor in indexing. If no exact match is found, the indexer creates a new group. However, in practice there would be a tendency to make do with a near match, which has detrimental effects on retrieval (see Section 15.2.2).

In searching, the main task is to identify the groups containing relevant entities; once that is done, the relevant entities can be found easily in the main file by means of the group numbers (in the sample system the searcher looks in the file in Fig. 15.2a to find the relevant documents). Group numbers are used as precombined descriptors in the file in Fig. 15.2a. Thus when searching the file in Fig. 15.2a, there is no need for combining descriptors in searching; combination searching has been shifted to the step of finding the appropriate group numbers (precombined descriptors) in the index file in Fig. 15.2b (See Section 15.5 for elaboration).

The change in viewpoint calls for a change in terminology; instead of *group* we now say *class.* The term *class* refers to three things:

- The *class representation,* a compound concept (e.g., Biology; Curriculum; First grade; Good)
- The *class number* (e.g., #7)
- The *set of entities* in the class (e.g., documents 9 and 20)

We use the term *class* to refer to all of these simultaneously, often with emphasis on the class representation.

The new terminology permits a concise description of the data base in Fig. 15.2 as seen from the new viewpoint: The classes serve as descriptors; the set of all classes forms the index language. Documents can be indexed by the ag propriate class. The main file in Fig. 15.2a is ordered by classes and thii makes it easy to find all entities belonging to a class. Put differently, in th data base in Fig. 15.2 compound concepts are included explicitly; they d< belong to the domain of the entity type *Concept.* Entities are linked explicit! to compound concepts, and their relationships to elemental concepts are in direct, via linkages among concepts. A system that uses compound concept in indexing (i.e., that uses precombined descriptors) is called a *precombina tion system.* The original set of elemental descriptors is the *core classified tion* (our term), and the set of all descriptors, including the precombinec descriptors, is the *extended classification* (our term). The index file in Fig 15.2b lets the searcher find classes (precombined descriptors) in terms ol their conceptual components; it is a *descriptor-find index.*

A class representation is a combination of many elemental concepts; it is *z* highly precombined descriptor. In the system in Fig. 15.2 precombination is carried to the extreme because a class includes just the entities whose representation in terms of elemental concepts is coextensive with the class representation. Put differently, the class representations have as many components as the entity representations constructed from elemental descriptors in the postcombination system in Fig. 15.1. With extreme precombination one class—one highly precombined descriptor—is sufficient to express all aspects for which an entity is relevant. This capability is maintained by adding newclasses (new precombined descriptors) whenever the need arises. The classes are mutually exclusive; there is never a need to assign an entity to two classes. Working with classes rather than with individual entities does not diminish retrieval power in any way; for every conceivable query formulation exactly the same subset of entities is retrieved. In other words, *extreme precombination* (using very highly precombined descriptors) and *extreme postcombination* (using elemental descriptors that can be combined in searching) *lead to the same retrieval results.* As we shall see in the following section, retrieval power suffers when the degree of precombination is lowered.

15.2.2 From Ideal to Reality: Limited Precombination

The advantages of grouping entities can be enhanced by forming broader classes. Class representations can be broadened by dropping a component or by substituting a broader component concept (Section 14.3). Both methods were used to arrive at the broad classes shown in Fig. 15.3a. But there is a price. Indexing an entity by one of these broad classes tells less about the entity than indexing it by the original elemental descriptors. Indexing is both less exhaustive (e.g., the quality aspect *Good, Medium, Bad* is lost in most

```
Class#! Method of instruction; Reading; El. school, incl.
        #1.1 Doc. 1 Method; Reading; First grade; Good
        #1.2 Doc. 3 Method; Reading; First grade; Bad
        #1.3 Doc. 5 Method; Reading; First grade; Good
        #1.4 Doc. 10 Method; Reading; El. School; Bad
        #1.5 Doc. 17 Method; Reading; El. School; Bad
        #1.6 Doc. 18 Method; Reading; First grade; Bad
Class #2 Traffic station; Inland water transport; Medium
        #2.1 Doc. 2 Tr. station; Inland water transp.; Medium
Class #3 Curriculum; Science, incl.; El. school, incl.
        #3.1 Doc. 4 Curriculum; Science; Fourth grade; Medium
        #3.2 Doc. 9 Curriculum; Biology; First grade; Good
        #3.3 Doc. 11 Curriculum; Biology; First grade; Bad
        #3.4 Doc. 12 Curriculum; Biology; First grade; Medium
        #3.5 Doc. 13 Curriculum; Biology; El. school; Medium
        #3.6 Doc. 20 Curriculum; Biology; First grade; Good
        #3.7 Doc. 21 Curriculum; Science; Fourth grade; Medium
Class #4 Traffic station; Water transport, incl.; Freight
        #4.1 Doc. 6 Tr. station; Ocean transport; Freight; Bad
        #4.2 Doc. 7 Tr. station; Ocean transport; Freight; Bad
        #4.3 Doc. 8 Tr. station; Inland water tr.; Freight; Med.
Class #5 Method of instruction; Reading; Junior high school
        #5.1 Doc. 14 Method; Reading; Ninth grade; Good
Class #6 Method of instruction; Science, incl.; El. school, incl.
        #6.1 Doc. 15 Method; Physics; Fifth grade; Bad
        #6.2 Doc. 16 Method; Biology; Sixth grade; Medium
Class #7 Curriculum; Reading; El. school, incl.
        #7.1 Doc. 19 Curriculum; Reading; Second grade; Bad
```

(a) List of classes with documents (main file).

classes) and less specific (e.g., the class may refer to *Elementary school* when the original elemental descriptor was *First grade).* Thus it may be useful to retain the original elemental descriptors even if retrieval access is based on classes. Note the alternate document numbers, such as #4.2.

Figure 15.3b shows the descriptor-find index to the broad classes, and Fig. 15.3c presents some sample searches. A specific query must first be broadened so that it combines only concepts used in the broad class representations; thus discrimination is lost when searching on the class level. If the entries contain the original specific elemental descriptors, discrimination can be restored in a second search step by examining document entries.

Thus, indexing by broad classes leads to a loss both in retrieval performance and in information given by an entity representation (which is the class by which the entity is indexed). With narrow classes there is no such loss.

Education
General concepts
  Curriculum #3,#7
  Method of instruction #1,#5,#6
Subject
  Reading, incl. #1,#5,#7
  Science, incl. #3,#6
Grade level
  El. school, incl. # 1 ,#3,#6,#7
  Junior high, incl. #5
Transportation
Traffic facil. vs. vehicles
  Traffic station #2,#4
Mode of transportation
  Water transp., incl. #2,#4
    Inland water transp. #2
Passenger vs. Freight transp.
  Freight #2,#4
Quality of document
Medium #2

(b) Index to broad groups (descriptor-find index).

cO Broad query formulation for cl,c2,c3	Method AND Reading AND El. school, incl. Class no. #1 Doc.no. 1,3,5,10,17,18
cl Original qf	Method AND Reading AND First grade AND Good Must broaden to cO, find Class # 1, examine document records, narrow results to doc. 1,5
c2 Original qf	Method AND Reading AND First Grade Must broaden to cO; find Class #1; examine document records, narrow to doc. 1, 3, 5,18
c3 Original qf	Method AND Reading AND El. school Broad to start with, co-extensive with cO, all documents in class # l are relevant
c4 Original qf	Method of instruction AND Reading Broad to start with, find Class no. #1, #5 Doc.no. 1,3,5,10,17,18, 14
c5 Broadened qf	Science, incl. AND Elementary school, incl. Class no. #3 Doc.no. 4,9,11,12,20,21
c5 Original qf	Biology AND First grade AND (Good OR Medium) Must broaden to c5 broad; find Class #3; examine, narrow results to doc. 9,12,20
c6 Original qf	Elementary school, incl. AND Good Must broaden by omitting *Good;* find classes # 1 ,#3,#6,#7. Examine, narrow to doc. 1,5,9,20

(c) Sample searches (qf =*' query formulation).

Fig. 15.3 Broad classes or groups, high precombination.
A class (group) may be broader than a document representation in the class (broadened by omitting quality descriptors and/or broadening other descriptors).

The listing of entities in broad classes is quite successful in collocating related entities. This is useful for browsing. It also reduces even more the number of points in which this file must be accessed to gather all entities about a topic. A broad search that corresponds precisely to a class (such as sample search c3 in Fig. 15.3c) has very good retrieval results, and all the relevant entities are very conveniently assembled in one place. On the other hand, a specific search (sample search c5) or a broad search that does not correspond to a ready-made class (sample search c6) requires examination of many entries. If the file does not allow for detailed examination because only the class numbers have been used for indexing, then discrimination is low. Searches for abstract concepts, such as *Structure,* are not possible at all, unless that concept was considered in the definition of classes.

An entity may belong to two or more broad classes. For example,

Document 8 Traffic station; Inland water transp.; Freight; Med.

belongs to both of the following:

Class #2 Traffic station; Inland water transport; Medium
Glass #4 Traffic station; Water transport, inclusive; Freight

A document on *Ports in the Great Lakes* belongs to both of the following:

DDC Class 386.5 Lake transport
DDC Class 386.8 Inland water transportation ports

Lunch meat made from *Turkey Liver* belongs to both of the following:

Lunch meat made from organ meat
Poultry-based lunch meat

If the system allows only one class for each entity, as is the case in shelf arrangement, then a more or less arbitrary decision must be made among several possible broader classes. For example, DDC instructs the cataloger to index a document on *Ports in the Great Lakes* by 386.8 *Inland water transportation ports.* As a consequence, this document will not be found under 386.5 *Lake transport.* In a thorough search for *Lake transport,* the searcher should look not only under 386.5, but also under 386.8.

15.2.3 Access Advantages of Grouped Files

Even in a postcombination system a grouped main file has advantages. A system that uses elemental descriptors in indexing and provides a postcombination index as in the data base in Fig. 15.1, but arranges the main file as in Fig. 15.3a, using the alternate document numbers in the index, permits specific and flexible retrieval a/irf increases the likelihood that documents retrieved through the index are collocated in the main file. A traditional library uses this setup. The subject catalog treats each book as an individual entity; yet due to the shelving by subjects, it is quite likely that the books found in a subject catalog search are all shelved on the same floor rather than scattered over several floors. This leads to a significant savings in working time. Grouping is also useful to decrease access time in large computer files.

One of two principles can be applied in forming groups:

1. *Request-oriented grouping brings together entities that are likely to be asked for together,* minimizing the average number of groups to be consulted in a search.
2. *Entity-oriented grouping brings together entities as they go together*, minimizing the average number of groups in which an entity must be included.

Under the first principle the groups overlap more than under the second principle; on the other hand, search costs are lower under the first principle.

The advantages of grouping are much enhanced if the groups themselves are arranged in a meaningful sequence; see Section 15.5.2 for elaboration.

15.3 GROUPING VERSUS DESCRIPTION OF ENTITIES

There are two approaches to organizing an unordered collection of entities (documents, parts, biological specimens, or whatever):

1. put like entities together and thus form classes, or
2. develop a list of descriptive characteristics and prepare for each entity a description (or representation) using these characteristics.

At first these two approaches seem quite different; they might even be perceived as diametrically opposed. The first tries to establish a global order within a set of entities, and thus concentrates on overall structure. The second concentrates on the individual entity and its description without regard to the overall structure. However, on closer examination, it becomes clear that the two approaches are intricately related and dependent on each other.

The formation of classes requires at least an implicit list of entity characteristics whereby the similarity of entities is judged. Explicit implementation involves developing descriptions of the entities and then forming classes based on these descriptions. This approach was used in Section 15.2.1. It is an approach now frequently used in biology, anthropology, and other sciences; it is known as numeric taxonomy. The methods used for class formation there (as well as in some document retrieval systems) are more complex than the very simple method used in Section 15.2.1 and usually require computer programs. In Section 15.2.1 the criterion to decide whether two entities should go in the same class was simple: Do they completely agree in their description in terms of elemental descriptors (i.e. , in their descriptive characteristics)? In a more sophisticated procedure one computes a quantitative measure of nearness between two entities based on the number of matches or near matches in their individual descriptive characteristics and then applies a more or less complex *clustering* program, which subdivides (partitions) the collection of all entities into clusters such that the entities within each cluster are nearer to each other than to entities in other clusters. The more exhaustively and the more specifically the individual features of each entity are described, the smaller the initial (narrow) classes. These initial classes, can be broadened as shown in Section 15.2.2.

This discussion has shown how description of entities can be used to arrive at a subdivision of the total collection into classes, that is, to derive an overall structure. Conversely, the assignment of an entity to a class constitutes a description of that entity. How good the description is depends on the nature of the classes. For example, in Section 15.2.1 assigning a document to class #7 (indexing the document by the precombined descriptor #7) provides as good a description as indexing by the four elemental descriptors

Curriculum; Biology; First grade; Good.

But in Section 15.2.2 assigning a document to class #3 provides only a more general description.

So far the discussion has assumed implicitly that the classes of entities would be mutually exclusive. In the context of traditional classification schemes the term *class* is often used with that connotation. However, in mathematics such a connotation does not exist at all; a class is simply a set of entities with some common characteristic. Thus, the set of all entities dealing with *Curriculum* is a class, and so is the set of all entities dealing with *Biology.* Thus, if an entity is described by five characteristics, it belongs to five classes, one for each characteristic. These classes are, of course, not mutually exclusive, but overlapping; there are entities that belong both to the class *Curriculum* and to the class *Biology.* In fact, the method of postcombination indexing is based on retrieving all entities that belong to two or more classes. In an ISAR system in which many elemental descriptors are used to index an entity, there is a high degree of overlap between the entity classes corresponding to the descriptors. Conversely, in a system with extreme precombination, such as the system described in Section 15.2.1, there is no overlap between classes at all; each entity belongs to one and only one class. The more the descriptors are precombined, the less the classes overlap, (see the examples in Section 15.2.2). Degree of precombination of descriptors and degree of overlap between classes measure essentially the same system characteristic (high precombination corresponding to low overlap). This underscores again the close interdependence between forming classes and describing individual entities.

15.4 POSTCOMBINATION AND PRECOMBINATION

15.4.1 Postcombination versus Precombination as a Matter of Degree >

Figure 15.4 illustrates that precombination is a matter of degree. The designer should choose the degree of precombination best suited to the types of queries expected and the retrieval mechanism used. The degree of precom-

Title

Needs for funding for the use of microcomputers in first grade reading instruction for the visually handicapped in New York City, 1984.

1. *Postcombination*. Each entity is indexed by many elemental descriptors (data base 15.1); many entries for each entity; small index language.

   Method of instruction; Reading; First grade; Good; Handicapped; Eyesight; Microcomputers; Funding; Needs assessment; New York City; 1984

2. *Moderate precombination.* Each entity is indexed by a few moderately precombined descriptors, as in a card catalog; a few entries for each entity; medium index language.

   Method of reading instruction; Instruction—Elementary school; Visually handicapped;
   Funding for educational equipment—New York City—1984;
   Funding for the handicapped—New York City—1984

   (Note the overlap among precombined descriptors. Note also the omission of the elemental concept *Needs assessment;* it is not contained in any precombined descriptor.)

3. *High precombination.* Each entity is indexed by only one highly precombined descriptor, as in a shelving system (data base 15.2); only one entry for each entity; large index language.

   Equipment for reading instruction for the handicapped. New York City .
   (Closest precombined descriptor available)

Fig. 15.4 Document representation in ISAR systems with different degrees of precombination.

bination is an important attribute in the analysis of an existing index language. Degree of precombination is an issue in data base organization, not an issue in the conceptual structure of the index language. The same basic conceptual structure can be used with any degree of precombination (see Section 15.6).

Postcombination should be used if the search mechanism permits easy retrieval by descriptor combinations. The lead-in vocabulary may retain compound concepts and show the combination of elemental descriptors to be used.

Moderate precombination is used, for example, in a regular library subject catalog that employs subject headings. Moderately precombined descriptors make it possible to search for compound concepts even if the search mechanism does not permit retrieval by combining descriptors, and they reduce the number of descriptors per entity (20 cards for each document in a card catalog would be impractical). The index language should include all concepts from the core classification so that the indexer can assign the elemental concepts that are not covered by any precombined descriptor assigned to the entity. In a computerized system precombined descriptors can be combined to form a still more compound query concept; for example,

Methods of reading instruction AND Visually handicapped

In a manual system the searcher looks under one descriptor and then examines the entities found to see whether the other required descriptors are present.

High precombination is needed in single-entry systems, such as arrangement of documents or groceries on shelves by subject, so that one descriptor assigned to an entity adequately reflects the topics for which an entity is relevant and makes them available as access points. The formation of classes of entities based on their relationships to elemental concepts introduces highly precombined descriptors, whether or not these are expressly named.

In a postcombination system the analysis of concepts into their semantic factors serves to detect compound concepts, which are then excluded from the index language; this keeps the number of descriptors down. In a precombination system the analysis of compound concepts into their semantic factors serves to establish relationships among the descriptors (precombined or elemental). These relationships can then be used for establishing a descriptor-find index, for creating a linear arrangement with cross-references, and, most importantly , for increasing the convenience and power of retrieval with a data base management system. For example, a document indexed by *Ship* and *Engine* can be retrieved in a search for *Vehicle* AND *Engine* due to the relationship *Vehicle* < is semantic factor of > *Ship.* A data base management system can use the relationships among concepts to find all entities related to a concept either directly (the concept is among the index terms for the entity) or indirectly (the concept is a component of or otherwise broader than an index term assigned to the entity).

Extreme postcombination may lead to *false drops*. The document

Cars used in subway link to airport

is indexed by

Vehicles; Local rail transit; Traffic stations; Air transport.

It is found in a search for *Aircraft,* since the query formulation is

Vehicles AND Air transport.

Elemental descriptors belonging to different compound concepts are mixed up. Role indicators and links or relators (see Section 12.4) prevent such false combinations; they make for a powerful and very flexible index language at the price of somewhat complex rules. To eschew the complexity of syntax while at the same time avoiding false drops, the designer can introduce precombined descriptors, such as *Subway car* or *Aircraft.*

15.4.2 Deciding on the Overall Degree of Precombination

The following points are important in making a decision.

Mechanical Devices Available in the ISAR System

If the search mechanism does not allow for combining descriptors in searching (e.g., shelf arrangement, card catalog, printed index), precombined descriptors are required, since most searches are for compound concepts. If combination searching is possible, such as in a computerized system, elemental descriptors suffice (unless ease of indexing, saving storage space, or avoidance of false drops dictate otherwise); in addition, fairly broad precombined descriptors may be useful for grouping of entities.

Number of Descriptors To Be Used for Indexing an Entity

If the descriptors are elemental, many are needed to index any one entity. This requires much effort unless at least the frequently used descriptors are printed on the indexing form so that the indexer can just check them. It also requires storage space. If the descriptors are precombined, a few will generally suffice to index an entity. The same considerations hold for queries.

Number of Descriptors Included in the Index Language and Difficulty of Indexing and Query Formulation

In a long list of precombined descriptors it is hard to find the appropriate descriptors for indexing or query formulation. A descriptor-find index is needed, and indexers and searchers must be trained. It may be easier to combine elemental or quasi-elemental descriptors (unless roles and links are used). However, this may not be true if the elemental descriptors are very abstract; it is easier for an indexer or searcher to use the descriptor *Ship* than to form the combination *Means: Transportation: Mobile: Water* (The combination *Vehicles: Water transport is* much easier). From the point of view of ease of understanding, a low degree of precombination is probably best, using as precombined descriptors those compound concepts that the user would expect to see as a unit. A facet frame can provide guidance in finding the proper combination of elemental concepts in indexing and query formulation.

Matching Descriptor Combinations Used by Searchers with Those Used by Indexers.

This problem arises when most descriptors are elemental. A good lead-in structure in the thesaurus promotes consistency.

15.4.3 Deciding on Individual Precombined Descriptors

Unless the system at hand uses extreme postcombination, the index language builder must decide which of the myriad possible compound concepts should be included in the index language, that is, selected as precombined descriptors. This section discusses the criteria used in these decisions; it also discusses the decision process.

With single entry, such as in shelf arrangement, the index language should provide for every entity a precombined descriptor that describes it reasonably well. In some systems new precombined descriptors are introduced as they are needed for new entities (see faceted classification later in this section). In other systems one makes do with the nearest precombined descriptor available, even if it does not cover all the elemental concepts for which the entity is relevant. In a system of moderate precombination with multiple entry, the following *rules* are helpful:

Use a compound concept as precombined descriptor *if it is used frequently in indexing or searching;* this *reduces the number of descriptors needed tain* dex an entity* or to formulate a query.

Use a compound concept as precombined descriptor *if its components occur frequently in different syntactic relationships;* this *prevents false combinations.* For example, retain *Library schools* (schools teaching about libraries) and *School libraries* (libraries serving schools) or *Administrative personnel* and *Personnel administration.* (But often the lack of syntactic relationships does not result in ambiguity, such as *Household : Tools.)*

Use a compound concept as precombined descriptor *if it is needed for logical completeness in the hierarchy or for the checklist technique of indexing.* For example, since *Agrarian reform* can be expressed easily as a combination of *Agriculture* and *Reform,* it may not seem required as a descriptor, However, an indexer using the checklist technique to index a document that is relevant to *Agrarian reform* will scan the descriptors listed under *Agriculture* to see whether the document is relevant. If the indexer finds *Agrarian reform* listed, he will surely use it. If the indexer does not find *Agrarian reform,* he might well overlook the descriptor *Reform.* The document would then be missed by searches for *Agrarian reform* as well as any other searches that include the component *Reform.* This example suggests a related rule: If a concept of general application is important for searching, the most important combinations containing that concept should be included as precombined descriptors to make sure that the general concept is not overlooked in indexing.

Use a compound concept as precombined descriptor *if any of its narrower concepts are descriptors;* for example, introduce the precombined descriptor *Aircraft* if its narrower concepts, such as *Airplane* or *Helicopter,* are descriptors.

Use a compound concept as precombined descriptor *if there is doubt.* A precombined descriptor can easily be replaced by a combination of elemental descriptors. The reverse is much more difficult; introducing a precombined descriptor after many documents are indexed requires that every entity indexed by the corresponding combination of elemental descriptors be examined to see whether it warrants indexing by the new precombined descriptor.

In each individual case, the benefits derived from the use of a precombined descriptor should be weighed against the cost, especially the increase in the number of descriptors in the index language and the concomitant complication of indexing. Statistics of descriptor use and retrieval performance are useful to monitor the effects of the decisions made.

As to the decision process, one option is to give the indexer complete freedom to introduce new precombined descriptors on the spot. An example is the use of a faceted classification with high precombination. While only elemental concepts (core descriptors, called “foci” in faceted classification) are listed explicitly, the indexer creates a combination of core descriptors that is just right for the entity at hand. The compound concept created is added to the data base—if it does not exist already—and the entity is linked to it. For example, a catalog card is filed under the compound concept. The new compound concept is a precombined descriptor. Since the list of precombined descriptors is continually growing without any control, a descriptor-find index is very important. Another example is the free formation of subject headings consisting of a main heading and a standard subheading. The other option is to require approval for precombined descriptors. This is usually done by permitting only precombined descriptors that are *enumerated* in a pre-made list, which may be updated from time to time, considering suggestions from indexers and—in good systems—searchers.

15.4.4 Precombined Descriptors in Indexing and Searching

>

A precombined descriptor available in the index language takes priority over a combination of two less compound descriptors. If the index language contains the precombined descriptor *Frozen beans,* the indexer should use it rather than the combination *Beans: Frozen.* Likewise, the indexer should use the descriptor *Visually handicapped* rather than the combination *Handicapped: Eyesight.* The searcher looking under *Visually handicapped* has a right to expect all relevant entities under this descriptor; this is why the precombined descriptor was introduced in the first place. The searcher should not have to look under a combination of elemental descriptors, too.

An entity should be indexed by all applicable precombined descriptors. If the index language contains a large number of precombined descriptors, thorough indexing can get a bit tricky. Assume the index language includes the precombined descriptors

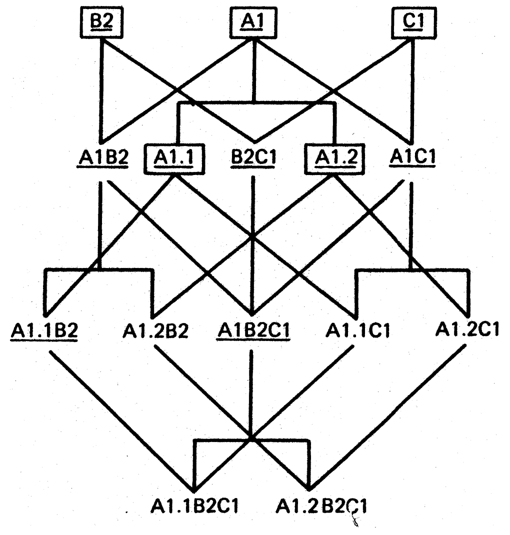
AI.I Vegetables : B2 Frozen AND
A1 Plant : B2 Frozen : CI Carton

but not

AI.I Vegetables : B2 Frozen : CI Carton

(Locate these concepts in the hierarchy displayed in Fig. 15.5; this is the same hierarchy as in Fig. 14.8; descriptors are underlined.) Assume an indexer must index the food product

Frozen vegetable packed in carton (AI.I B2C1)



Descriptors are underlined

Gore classification

A Food source
  AI Plant
    A 1.1 Vegetable
    AI.2 Fruit

B Preservation
  B2 Frozen

C Packaging
  CI Carton

Fig. 15.5 Finding precombined descriptors in a hierarchical structure.

How does he find the precombined descriptors he should use? The general rule of request-oriented indexing says to use all descriptors under which the entity is to be found. (A specific descriptor implies all its broader descriptors; for example, if the food is indexed by A1.1B2 *Vegetables. Frozen, it* need not also be indexed by AI.I *Vegetable*.) The food product at hand should be found under any descriptor that is broader than the product representation A1.1B2C1; locate this concept in the hierarchy and follow each hierarchical chain upward until you come to a descriptor. The descriptors thus found are A1.1B2, A1B2C1, and AI.I (via A1.1C1); AI.I is implied by A 1.1 B2, so it is not needed.

15.5 ORGANIZING AN INDEX LANGUAGE FOR ACCESS

Searching by subject or any other criterion requires two steps, the first of which is also needed for indexing:

Step 1: Find the descriptors needed for formulating a query or indexing an entity.
Step 2: Find the focal entities that are in the proper relationship to the descriptors.

Both steps involve retrieval operations in an integrated data base, and the ISAR system should support them.

With postcombination, step 1 is easy. There is a fairly small number of elemental descriptors. The relationships between descriptors are not overly complex; only a few have more than one broader term. The index language can be shown easily in a classified arrangement of descriptors (preferably divided into facets) with some cross-references added, supplemented by an alphabetical index. Even an alphabetical arrangement alone with all relationships expressed through cross-references might do.

The difficulty with a postcombination system comes in step 2. Since most searches are for compound concepts, the search mechanism must allow searches for *a combination of descriptors.*

With high precombination it is just the other way around: There is a large number of precombined descriptors. With extreme precombination—as in the data base in Fig. 15.2—the number of precombined descriptors is of the same order of magnitude as the number of main entities (in the example documents). The Library of Congress Classification fills 30 volumes. There is a complex web of hierarchical and associative relationships (see Chapter 14, especially Section 14.3). The searcher must find in this huge web all the precombined descriptors that are equal to or narrower than the search topic.

In other words, the searcher must find all precombined descriptors that contain among their components the elemental concepts that make up the search topic. That is a big job, and it requires the capability of *searching for precombined descriptors with a combination of elemental concepts.* Step 2 is easy with high precombination. The searcher simply looks under the precombined descriptors to find the focal entities.

In a postcombination system, combination searching is needed in step 2, retrieving focal entities; in a precombination system, combination searching is needed in step 1, retrieving precombined descriptors. With moderate precombination some combination searching is needed in step 1 and some combination searching in step 2.

The remainder of this section deals with the problem of finding precombined descriptors. With modern technology, the most natural solution is the one illustrated in the data bases in Figs. 15.2 and 15.3, namely, a *descriptor-find index*—preferably for on-line searching—in which the searcher can formulate a query as a combination of elemental concepts and find all precombined descriptors that contain these concepts. Such a descriptor-find index is part of an integrated data base that permits searching through a chain. The descriptor-find index leads from elemental concepts to compound concepts. Other parts of the data base structure—for example, a document-find index or a main file arranged by precombined descriptors—lead from compound concepts to documents or other entities. Descriptor-find indexes are discussed in Section 15.5.1.

Another, more limited way of providing access to precombined descriptors, which was developed before combination search mechanisms were available, is to arrange them in a meaningful sequence that collocates related descriptors. Arrangement of descriptors is discussed in Section 15.5.2.

15.5.1 DESCRIPTOR- FIND INDEXES

Two examples will illustrate this concept further.

Example 1: A Descriptor-Find Index for the LC Classification

In the data base in Fig. 15.2, a core classification of elemental concepts was given and the descriptor-find index developed naturally through the procedure by which precombined descriptors were formed. In the Library of Congress Classification the precombined descriptors were not formed in the same systematic procedure. There is no core classification (many core concepts may not even be included per se as descriptors), only a large unwieldy list of precombined descriptors. Facet analysis (see Sections 14.3 and 14.5) results in a core classification of elemental concepts and a representation for each class in terms of core concepts (Figs. 15.6a and b). These can be used to construct a descriptor-find index, preferably as an on-line file. For example, the query formulation

L20 Traffic facility AND L37 Water transport

retrieves a list of LC class numbers, such as

HE550-560 Ports, harbors, docks, wharves, etc.
NA6330 Dock buildings, ferry houses, etc.
VA67-79 Naval ports, bases, reservations, docks, etc.

Transportation and traffic
L10 Vehicles
L20 Traffic facilities
    L25 Traffic stations
L33 Airtransport
L37 Water transport

Buildings, construction
P23 Buildings
P27 Architecture
P43 Construction

R00 Engineering
    R30 Acoustics
        R37 Soundproofing

T70 Military vs. civilian
    T73 Military
    T77 Civilian

UOO America
    U15 US

(a) Core classification.

HE550-560 Ports, harbors, docks, wharves, etc.
= L25 Traffic stations : L37 Water transport: T77 Civilian

NA2800 Architectural acoustics
= P27 Architecture : R30 Acoustics

NA6300-6307 Airport buildings
- L25 Traffic stations : L33 Air transport: P23 Buildings : T77 Civilian

NA6330 Dock buildings, ferry houses, etc.
= L25 Traffic stations : L37 Water transport: P23 Buildings : T77 Civilian

TC350-374 Harbor works
= L25 Traffic stations : L37 Water transport: R00 Engineering

TH1725 Soundproof construction
= P23 Buildings : P43 Construction : R37 Soundproofing.

TL681.S6 Airplanes. Soundproofing
= L10 Vehicles : L33 Air transport: R37 Soundproofing.

TL725-726 Airways (Routes). Airports and landing fields. Aerodromes
= L20 Traffic facilities : L33 Air transport

VA67-79 Naval ports, bases, reservations, docks, etc.
= L25 Traffic stations : L37 Water transport: T73 Military : U15 US

VM367.S6 Submarines. Soundproofing
= L10 Vehicles : L37 Water trans. : R37 Soundproofing.

(b) LC classes with decomposition into semantic factors.

L10	L10:L33:R37	TL681.S6 Airplanes. Soundproofing
	L10:L37:R37	VM367.S6 Submarines. Soundproofing
L20	L20:L33	TL725-726 Airways, airports, etc.
L25	L25:L33:P23:T77	NA6300-6307 Airport buildings
	L25:L37:P23:T77	NA6330 Dock buildings, ferry houses, etc.
	L25:L37:R00	TC350-374 Harbor works
	L25:L37:T73;U15	VA67-79 Naval ports, bases, etc.
	L25:L37:T77	HE550-560 Ports, harbors, docks, etc.
L33	L10:L33:R37	TL681.S6 Airplanes. Soundproofing
	L20:L33	TL725-726 Airways, airports, etc.
	L25:L33:P23:T77	NA6300-6307 Airport buildings
L37	L10:L37:R37	VM367.S6 Submarines. Soundproofing
	L25:L37:R00	TC350-374 Harbor works
	L25:L37:R23:T77	NA6330 Dock buildings, ferry houses, etc.
	L25:L37:T73:U15	VA67-79 Naval ports, bases, etc.
	L25:L37:T77	HE550-560 Ports, harbors, docks, etc.
P23	L25:L33:P23:T77	NA6300-6307 Airport buildings
	L25:L37:P23:T77	NA6330 Dock buildings, ferry houses, etc.
	P23:P43:R37	TH1725 Soundproof construction
P27	P27:R30	NA2800 Architectural acoustics
P43	P23:P43:R37	TH1725 Soundproof construction
R00	L25:L37:R00	TC350-374 Harbor works
R30	P27:R30	NA2800 Architectural acoustics
R37	L10:L33:R37	TL681.S6 Airplanes. Soundproofing
	L10:L37:R37	VM367.S6 Submarines. Soundproofing
	P23:P43:R37	TH1725 Soundproof construction
T73	L25:L37:T73:U15	VA67-69 Naval ports, bases, etc.
T77	L25:L33:P23:T77	NA6300-6307 Airport buildings
	L25:L37:P23:T77	NA6330 Dock buildings, ferry houses, etc.
	L25:L37:T77	HE550-560 Ports, harbors, docks, etc.
U15	L25:L37:T73:U15	VA67-69 Naval ports, bases, etc.

(c) Descriptor-find index (KWOC format using notation).

Fig. 15.6 A descriptor-find index for LC classification

L10 Vehicles AND R30 Acoustics, inclusive

retrieves

TL681.S6 Airplanes. Soundproofing
VM367.S6 Submarines. Soundproofing

Fig. 15.6c shows a KWIC-type descriptor-find index. Two examples of descriptor-find indexes actually implemented as printed indexes are the *Relative Index* to DDC (with many limitations) and a *chain index* to a set of precombined descriptors generated from a faceted classification. All of these bring together distributed relatives—precombined descriptors that share an elemental concept but are scattered in the arrangement—and thus complement the access provided by a meaningful arrangement.

Example 2: A Newspaper Clipping File

A large newspaper clipping file contains from one million to ten million documents. Each document (each clipping) is very small compared to average journal articles or books. Usually many of these documents, perhaps 20 or more, are needed to answer a request. Dealing with each document as an individual unit would require an index of tremendous size. It would also be very cumbersome to access the file of the documents themselves in so many different places. Just picture accessing 20 articles appearing over three years in five different newspapers on microfilm. The situation cries out for forming classes. For each class (precombined descriptor) there is a folder containing the relevant clippings. The folder heading is the name of the class. Each folder has assigned to it a combination of elemental concepts (core descriptors), which is the folder representation. There may be 50,000-500,000 folders. Sample folders are:

Bilateral relations between *France* and the *US*
Soviet Reactions to *Bilateral relations* between *France* and the *US*
Bilateral relations between *Great Britain* and the *US*

In this file a user can find conveniently all clippings relevant to a topic, provided she knows in which folders to look. To find these folders, the user consults an index of folders (precombined descriptors), that is a descriptor-find index. A user interested in the topic

Bilateral relations between *France* and the *US*

would AND the three core descriptors and find the first two sample folders, among others. A user interested in

Bilateral relations of the *US* (with any other country)

would retrieve all three sample folders, among others.

Retrieving precombined descriptors based on their relationships to elemental concepts is the same type of operation as finding documents or other entities based on their relationships to concepts. Just using a different view to describe the system of newspaper clippings illustrates this point. We redefine documents and call each *folder* a document. (Some libraries do indeed periodically take all the clippings in an important folder, bind them into book form, and treat the result like any other book in their book collection.) The folder heading is seen as the title of this new document, and the folder representation is seen as the representation of this document. In this view, retrieving folders is seen as retrieving documents. Thus the same index can be viewed as a descriptor-find index (consider the folder headings as precombined descriptors assigned to all clippings in their folder) or as a document-find index (consider the folder headings as document titles).

These examples illustrate that *combination searching is needed in any system.* The only difference between postcombination and precombination systems is how the workload of combination searching is distributed between the two search steps. In most postcombination systems the user receives excellent assistance in combination searching as a matter of course through the data base organization and search mechanism (such as an on-line retrieval system for documents or other entities). In precombination systems the user is presented with an extensive list of precombined descriptors (with LCC this includes 30 volumes), but is left on her own in searching for the right descriptors, since there is usually no descriptor-find index. The only assistance the user has for combination searching is the more or less meaningful arrangement . It is high time for the construction of on-line descriptor-find indexes to assist the users of existing precombination systems that cover large collections and that will be with us for quite some time to come.

15.5.2 Arrangement and Designation of Descriptors

Arrangement and designation of descriptors play an important role in organizing an index language for access. Any system—regardless of degree of precombination—requires the helpful arrangement of core descriptors (the elemental concepts used as descriptors in a postcombination system or used to produce precombined descriptors in a precombination system). A helpful arrangement of precombined descriptors may serve as a limited substitute for a descriptor-find index or as a supplement to it.

Descriptors can be arranged in classified (subject) order or in alphabetical order. A meaningful classified arrangement serves the following interrelated functions (see Section 13.6 and 13.7 for a more complete discussion):

Orientation of the user in an index language.

Communication of a useful conceptual framework that helps users organize information or entities (e.g., in memory).

Organization of a collection of entities so that the user has a framework for orientation in the collection. Materials that are needed together or should be examined together should be collocated. Section 15.2 has shown that grouping entities into classes—each corresponding to a precombined descriptor—can be helpful for the user; grouping related classes together—arranging the precombined descriptors in a meaningful classified sequence—takes this principle a step further. This use of arrangement is also important when presenting to the user the mini-data base that answers his request. Computer output has the advantage that the arrangement can be tailored to the purpose at hand; many lessons for this task can be learned from writers oil library classification.

Organization of a set of precombined descriptors so that precombined descriptors can be found in terms of their components. With this function, the classified arrangement serves as a substitute for a descriptor-find index. However, a linear arrangement can express only a fraction of the many hierarchical and associative relationships that exist between the precombined descriptors. Collocating all the precombined descriptors in the area of *Law* disperses the descriptors for many other subjects, such as *Food, Transportation,* or *Education,* creating *distributed relatives.* Deciding on a particular arrangement means deciding which hierarchical relationships to show through the arrangement and which to show through crossreferences or not at all. The arrangement should be based on the hierarchical relationships most important to the user. To some extent a systematically developed network of cross-references can guide the user to relevant precombined descriptors not collocated in the arrangement; with a high degree of precombination the number of cross-references becomes unwieldy, and a descriptor-find index should take their place.

Arrangement of descriptors is closely intertwined with their designation—the choice of descriptor identifiers. The designation system must be chosen so that it leads to the desired arrangement. For example, a classified arrangement requires notations as descriptor identifiers.

In the arrangement of core descriptors the choices are limited. One can arrange them either in a meaningful classified order with notations as descriptor identifiers (of course accompanied by terms) or alphabetically with terms as descriptor identifiers. With precombined descriptors there are many more options that differ in the degree to which the components of a precombined descriptor determine its place in the sequence. These options are discussed in the remainder of this section. They apply whether the components (core descriptors) are identified by notations or by terms. While arrangement is more fundamental, and decisions on arrangement should logically precede decisions on designation, designation rules are easier to understand and therefore discussed first.

The following core classification is used in all examples.

Facet A Food source	Facet B Preservation	Facet C Packaging
A1 Plant product	B1 Fresh	CI Carton
Al.l Vegetable	B2 Frozen	C2 Bag
A2 Animal product	B3 Sterilized	

Designation of Precombined Descriptors

Option 1 is to create the identifier of a precombined descriptor by *combining the identifiers of its components infixed citation order* (in the example: Facet A, Facet B, Facet C):

A1.1B3C2 Vegetable : Sterilized : Bag
A2B1 Animal product: Fresh
A2B1C1 Animal product: Fresh : Carton

This rule for designation completely determines the arrangement of precombined descriptors. Examples are faceted classification used in a precombination system and the LC Subject Headings (with the citation order main heading—subheading). This rule should be used in a system that allows the indexer to freely introduce new precombined descriptors, because otherwise the same descriptor might end up with two or more designations, filed at two or more different places.

Option 2 is to create the identifier of a precombined descriptor by combining the identifiers of its components *in a free citation order,* determined ad hoc in each individual case. For example:

A2B1 Animal product: Fresh
B1A2C1 Fresh : Animal product: Carton
C2B3A1.1 Bag : Sterilized : Vegetable

This designation rule still determines the arrangement of precombined descriptors to a considerable degree, but not completely.

Option 3 is to *choose the identifier of a precombined descriptor without being bound by the descriptor components (t*he identifier is an *independent symbol).* Examples are LCC (the independent symbols being class numbers) and the main headings in a subject heading list, the independent symbols being terms. For example:

A1.5 Sterilized vegetables packed in a bag
A2.3 Fresh meat
A2.5 Fresh meat packed in cartons

This designation rule gives the designer complete freedom in the arrangement of precombined descriptors (including an arrangement following a fixed set of rules).

Arrangement of Precombined Descriptors

Option 1 is to arrange precombined descriptors in a *sequence that is mandated by their components in a fixed citation order.* This arrangement option allows designation option 1 or 3 (designation option 1 is shown in the example). The example includes only part of all possible combinations of core concepts.

- A1 Plant product
    - A1B1 Plant product: Fresh
        - A1B1C1 Plant product: Fresh : Carton
        - A1B1C2 Plant product: Fresh : Bag
    - A1B3 Plant product: Sterilized
        - A1B3C1 Plant product: Sterilized : Carton
        - A1B3C2 Plant product: Sterilized : Bag
        - A1.1B3 Vegetable: Sterilized
    - A1C1 Plant product: Carton
    - A1C2 Plant product: Bag
- A2 Animal product
    - A2B1 Animal product: Fresh
    - A2B2 Animal product: Frozen
    - A2B3 Animal product: Sterilized
- B1 Fresh
    - B1C1 Fresh : Carton

In the citation order chosen, subdivision by preservation is always second and follows the standard order B1 *Fresh,* B2 *Frozen,* B3 *Sterilized.* Subdivision by packaging is always third and follows the standard order Cl *Carton,* C2 *Bag.* The facet of highest importance to the user should be put first so that, for example, all *Plant product* descriptors are together. The precombined descriptors containing *Fresh,* an element of the second-placed facet, are scattered throughout the arrangement *(distributed relatives).* The citation order is independent of the order in which the facets are listed; for example, it could be BCA or CBA.

Option 2 is to arrange precombined descriptors in a *sequence that is mandated by their components but withfree citation order.* Arrangement option 2 allows designation option 2 or 3 (designation option 2 is shown in the example). Descriptors that are out of order as compared to option 1, fixed citation order, are marked by *.

A1 Plant product
- A1B1 Plant product: Fresh
- A1B3 Plant product: Sterilized
  - A1B3C1 Plant product: Sterilized : Carton
  - A1B3C2 Plant product: Sterilized : Bag
  - A1.1B3 Vegetable : Sterilized
- Aid Plant product: Carton
  - ♦A1C1B1 Plant product: Carton : Fresh[1]
- A1C2 Plant product: Bag
  - ♦A1C2B1 Plant product: Bag : Fresh[1]

A2 Animal product
- A2B2 Animal product: Frozen
- A2B3 Animal product: Sterilized

B1 Fresh
- * BI A2 Fresh : Animal product[1]
- B1 CI Fresh : Carton

In this method, subdivision by method of preservation, for example, may be first, second, or third. But wherever subdivision by preservation is used, the standard sequence B1 *Fresh,* B2 *Frozen*, B3 *Sterilized* applies.

Option 3 is to arrange precombined descriptors in a freely chosen sequence that is not mandated by the descriptor components. Arrangement option 3 requires designation option 3.

A1 Plant product
- AI.I Fresh plant product
- A1.2 Sterilized plant product
  - *A1.3 Sterilized plant product packed in bag[2]
  - *A1.4 Sterilized plant product in carton[2]
  - A1.5 Sterilized vegetable
- A1.6 Plant product packed in carton
  - * A1.7 Fresh plant product packed in carton[1]
- A 1.8 Plant product packed in bag
  - *A1.9 Fresh plant product packed in bag[1]

A2 Animal product
- ♦A2.1 Sterilized animal product[3]
- *A2.2 Fresh animal product[3]
- ♦A2.3 Frozen animal product[3]

B1 Fresh
- BI.I Fresh product packed in carton

[1] Nonstandard citation order
[2] Nonstandard sequence *Bag-Carton*
[3] Nonstandard sequence *Sterilized-Fresh-Frozen*

In this option the designer subdividing a topic—e.g., *Judaism* or *Reformation*—by geographic area can choose a sequence of countries most appropriate in the context of the topic. She would start with Israel and other Middle East countries for *Judaism* and with Germany and Switzerland for *Reformation.* Still another sequence of countries may be appropriate to subdivide *Economic development.* The designer should use this freedom judiciously. If a topic does not suggest a particular point of view for sequencing countries, a standard sequence is most helpful. Perhaps there should be two or three standard sequences, each appropriate for a range of topics.

The meaningful geographic subdivision of some concepts requires more than just varying the sequence of countries; countries may not be the most appropriate geographic areas. For example, the geographic subdivision of *Vegetation* should use such concepts as *Tropics, Subtropics,* etc. Arrangement option 3 leaves the designer free to create for each concept a tailor-made scheme of geographic areas. Again, in order to preserve the clarity of the index language, such schemes should not proliferate beyond what is necessary. In this discussion, geographic subdivision is just an example. The considerations apply also to subdivisions by historical periods or by any other concept.

The principles of designation and arrangement are further illustrated by their application to subject headings. The main headings and the subheadings are independent elements designated by terms and arranged alphabetically. Many subject headings are created by combining a main heading with a subheading, and for those headings the place in the arrangement is determined by the components. The citation order is fixed: main heading-subheading. A closer look reveals a more complex situation. Often the linguistic structure of the main heading reflects the conceptual structure, as in *Radio engineering* or *Aircraft.* Should the citation order be *Radio engineering* (direct form) or *Engineering, radio* (inverted form); *Aircraft* or *Craft, air?* The problem of direct versus inverted headings reveals itself as a problem of citation order.

15.6 A UNIFIED INDEX LANGUAGE FOR DIFFERENT SEARCH MECHANISMS

The degree of precombination in an index language must be adapted to the search mechanism; an on-line system and a printed index are best used with different degrees of precombination. Sometimes access to the same collection of documents or other entities is provided through several search mechanisms, for example, the on-line system *MEDLINE* and the printed *Index Medicus.* In such a case it is desirable to adapt the index language used to the search mechanism while maintaining a common conceptual base. More generally, if information systems overlap in their subject scopes or their col* lections, it is desirable, both from the point of view of their users and from the point of view of sharing indexing effort, to have index languages that share their conceptual base. A solution for this problem evolves from the concepts developed in this chapter.

At the beginning is a *core classification* consisting of elemental or quasi- . elemental *core descriptors* as defined in Section 15.2. This is the index language for a postcombination system. In a precombination system (e.g., a card catalog or a printed index), precombined descriptors are introduced as needed, extending the core classification to result in the *extended classification* with a descriptor-find index. The core classification stays the same, across systems of different types (e.g., on-line system and printed index) and across systems of the same type (e.g., the card catalogs of three libraries using the same core classification). The extended classification may vary from one precombination system to another, depending on local needs. Indexing can be done once and for all using the core classification. The precombined descriptors to be used in each system can then be derived automatically . One indexing step thus provides descriptors for on-line searching, subject headings, DDC classes, LC classes, and precombined descriptors in any other scheme. (Such an approach is used in the PRECIS system as applied in the British Library.) The core classification must be very specific, and indexing must consider the interests of all organizations involved. A properly designed core classification could thus take the role of the old dream, a universal classification. This is made possible by concentrating on the basic principles of conceptual structure and leaving aside details of arrangement and data base organization on which agreement cannot be reached and is not even always desirable.