Retrieving Cases from Relational Data-Bases: Another Stride Towards Corp orate-Wide Case-Base Systems

Hideo Shimazu* Hiroaki Kitano** Akihiro Shibata*
C&C Information Technology Research Laboratories*, NEC Corporation
4-1-1 Miyazaki, Miyamae
Kawasaki, 216 Japan
{shimazu,akihiro} joke.cl.nec.co.jp

Case Systems Laboratory**
NEC Corporation
2-11-5 Shibaura, Minato
Tokyo, 108 Japan
kitano spls26.ccs.mt.nec.co.jp

Abstract

Vital information for corporate activities is generally stored in large databases. While conventional data-base management systems offer limited query flexibility, systems capable of generating similarity-based queries, such as those seen in case-based reasoning research, would certainly enhance the utility of data resources. This paper describes a method for building case-based systems using a conventional relational data-base (RDB). The core of the algorithm is a novel approach to similarity computing in which database query form similarities, rather than similarities of individual cases, are computed. The method uses Standard Query Language (SQL) to achieve nearest neighbor matching, thus allowing similarity-based database retrieval. It has been implemented as a part of the CARET case retrieval tool and evaluated through the use of a newly developed corporate-wide case-based system for a software quality control domain. Experiments have shown the proposed method to provide retrieval results equivalent to those of non-RDB implementation at a sufficiently fast response time.

1 Introduction

This paper describes the architecture and performance evaluation of a case-based reasoning system built on a commercial relational data-base management system (RDBMS). Specifically, the authors have developed CARET (Case Retrieval Tool), a tool which generates appropriate SQL [ANSI, 1989] expressions to carry out similarity-based retrieval on a commercially available RDBMS, such as ORACLE [Oracle, 1989]. The core of the algorithm is a novel approach to similarity computing which pre-computes query form similarities, rather than the similarities of individual cases.

The motivation behind this research was two-fold. First, the use of a RDBMS, or a mechanism with equivalent functions, was found to be the minimum requirement for CBR systems to be installed as part of a corporate-wide information system. This requirement was pointed out by engineers and managers in corporate data processing divisions and other administrative divisions. Important reasons include the need for security control and integrity management.

Second, significantly large data-bases in various domains presenly available in many corporate information systems are built on commercial RDBMSs. Applications using CBR would allow exploring new horizons in corporate information systems, since current RDBMSs only provide relatively primitive (i.e. only exact match) query capability. Efforts to convert these data-bases into independent CBR systems, however, will inevitably undermine security control and waste computing resources (memory and CPU). In addition, since two separate data-bases would be created in such a conversion, the integrity of the case-base would not be maintained because the mechanism for automatically reflecting changes in the RDB to case-base would, for both technical and security reasons, not be installed

The underlying belief behind these observations is *the mainstream dogma,* which claims that a CBR system should be integrated into a mainstream corporate-wide strategic information system. This idea was first advocated in [Kitano *et a*l., 1992; Kitano *et a*l., 1993] and has been the central core of the CBR-related research activities at NEC Corporation. Development of a CBR architecture using RDBMS would be a major step forward toward achieving the goals of the mainstream dogma.

Unfortunately, however, no existing CBR system uses RDBMS as its case-base manager. Even commercial tools, such as CBR Express on ART-IM, ReMind, and ESTEEM, do not incorporate RDBMS. Of course, not all CBR systems require tight security control and integrity management. Nevertheless, the advantages inherent in building CBR systems using RDBMS are remarkable. The authors argue that this is one of the most significant unexplored avenues in CBR research and describe the architecture and evaluation results of their proposed CBR system for RDBMS.

2 Problems in Current Case-Base Management

The case-base management concept is critically lacking in the current research on CBR systems. This is mainly due to the fact that a vast majority, if not all, of CBR systems have been built as task-specific domain problem

solvers. These systems, similar to most expert systems, are detached from mainstream information systems for a corporation. However, as has been clearly demonstrated in the wide-spread use of data-base management systems (DBMS), data resource management is an essential issue the corporate information systems.

Ib be more specific, the following issues have not been addressed in previous studies on CBR systems.

Security Control: In real applications, collected cases include secret information for a corporation or a department. No CBR systems developed so far incorporated any security measures. In the absence of the security control, the system can not be used for highly confidential information where maximum value can be exploited.

Scalability The efficiency of a CBR application heavily depends on the number of cases collected. In some real applications, collected cases can increase drastically chronologically. For example, in the SQUAD system, a corporate-wide CBR system for software quality control domain [Kitano *et* a/., 1992], over 3,000 cases have been added into its case-base each year. Since real domains are often very complicated and ill-formed, use of complex indexing, as seen in [Hammond, 1986; Hunter, 1988; Ashley and Rissland, 1987; Kolodner, 1984], would require significant development costs and the system behavior would be unstable, as the experts themselves do not fully understand the nature of such domains. It would be beyond control of system engineers.

Speed Although various indexing and case-base organization methods have been investigated, only a few studies addressed the issue of computing cost. Fast case retrieval is an indispensable feature of real-world applications, particularly for very large case-bases.

3 CARET Design Decisions

This section describes major design decision made in developing CARET (CAse REtrieval Tool).

3.1 Case-Base Management using RDBMS

The most significant design decision was the use of a commercial RDBMS for its case-base manager. Each case is represented as a record of a relational database table. The use of a RDBMS offers several advantages, such as (1) data security, (2) data independence, (3) data standardization, and (4) data integrity. The use of RDBMS, however, automatically forces CARET to generate SQL specifications [Codd, 1970; Chamberlin *et* a/., 1976] to carry out any case-base retrievals. SQL is the standard relational database access language. However, SQL does not entail any similarity-based retrieval features. The essence of this work, therefore, is to find a method to carry out similarity-based retrievals using SQL.

The other constraint, imposed from the use of RDBMS, is that cases have to be represented as a flat record of n-ary relations. RDBMS do not entail a mechanism to support a complex indexing scheme, as seen in most CBR researches. However, this constraint is not necessarily a limiting factor for case representations. Through the SQUAD development, the authors learned that complex indexing schemes are too difficult to maintain, particularly for ordinary system engineers.

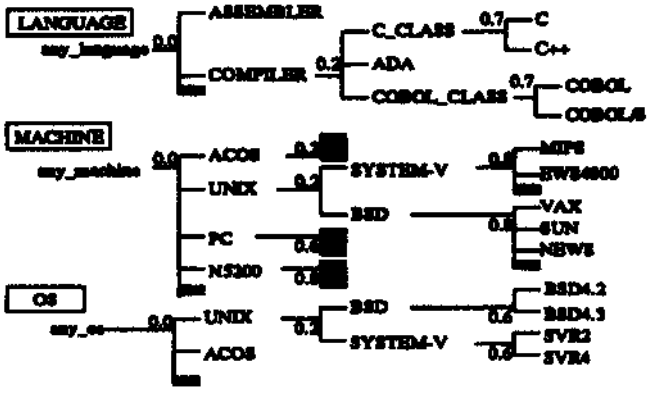

Figure 1: Abstraction Hierarchies Example

3.2 Similarity Definition using Abstraction Hierarchy

Similarity between values is defined using an abstraction hierarchy, shown in Figure 1. It is defined for each attribute in a flat table, such as language, machine, and OS. In this example, the similarity between C and C++ is 0.7. Similarity between the input query and each case in the case-base is calculated by referring to the similarity of values in each attribute.

3.3 Case Retrieval using Nearest Neighbor

CARET uses nearest neighbor retrieval, instead of indexing-based methods. Typically, a similarity between a query (Q) and a case (C) in the case-base *(S(Q,C))* is the weighted sum of similarities for individual attributes:

$$S(Q,C) = \frac{\sum_{i=1}^{n} W_i \times s(Q_i, C_i)}{\sum_{i=1}^{n} W_i} \qquad (1)$$

where W_i is the t-th attribute weight, $s(Q_i, C_i)$ is similarity between the t-th attribute value for a query (Q) and that for a case (C) in the RDB.

Traditional implementations would compute the similarity value for all records, and sort records based on their similarity. However, this is a time consuming task, as computing time increases linearly with regard to the number of records in the case-base (C : *Cardinality of* the data-base) and to the number of defined attributes (D : *Dtgree* of the data-base). This results in time-complexity of *0(C* x *D).* This implementation strategy for RDBMS would be a foolish decision, as individual records have to be retrieved to compute similarity. Thus, the total transaction number would be intolerable.

The challenge here is to discover an algorithm to implement the nearest neighbor algorithm using SQL sufficiently efficient to be practical. The following sections

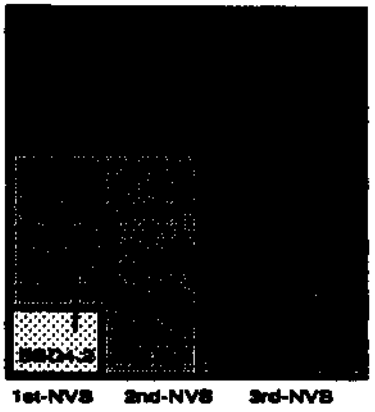

Figure 2: Neighbor Value Sets

Attribute	language	machine
User Query	ADA	VAX
1st-NVS	ADA	VAX
2nd-NVS	C, C++, COBOL, COBOL/S	SUN, NEWS, ...
3rd-NVS		MIPS, EWS4800,...

Table 1: NVSs for ADA and VAX

describe such an algorithm and report on its performance using the deployed system.

4 Generating SQL specifications with Similarity Measures

4.1 Creating Neighbor Value Sets

For each attribute, CARET refers to the abstraction hierarchies, as shown in Figure 1, to generate a set of values neighboring the value specified by the user. For example, assume that the user specified BSD4.2 in the hierarchy shown in Figure 2, BSD4.2 is an element in *the first-order neighbor value set (lst-NVS).* BSD4.3 is an element in *the second-order neighbor value set (2nd-NVS).* SVR2 and SVR4 are elements in *the third-order neighbor value set (3rd-NVS).* Such sets are created for each attribute.

Assume that the user specified VAX for an attribute machine and ADA for an attribute language in the hierarchies in Figure 1, Table 1 shows the generated NVS sets.

4.2 Enumerating the Combinations

Next, all possible neighbor value combinations are created from the n-th order neighbor value sets. Figure 3 illustrates how such combinations are created. This example assumes that the user specified values for attribute language and machine. All value combinations under attribute language and machine will be created. Weight of attributes (0.3 for language and 0.4 for machine) and similarity measure (such as 1.0, 0.2, 0.8 assigned to each value set) are used to calculate the similarity between a combination and the problem definition specified by the user. Using the neighbor value sets from the previous example, combinations shown in Table 2 are created.

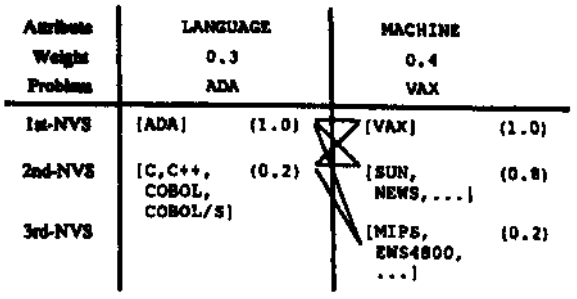

Figure 3: An Example showing Possible Neighbor Value Combinations

```
and(language(ADA), machine(VAX))
and(language(ADA), machine(or(SUN, NEWS, ...)))
and(language(ADA), machine(or(MIPS, EWS4800, ..)))
and(language(or(C, C++, COBOL, COBOL/S)), machine(VAX))
and(language(or(C, C++, COBOL, COBOL/S)),
    machine(or(SUN, NEWS, ...)))
and(language(or(C, C++, COBOL, COBOL/S)),
    machine(or(MIPS, EWS4800, ..)))
```

Table 2: Combinations created

Disjunction is created for values under the same node. Under this assumption, the maximum number of combinations (N) produced in a given query will be:

$$N = \prod_{i=1}^{r} d_i \qquad (2)$$

where r is a number of attributes specified by the user, and d_i is the depth of the tree of the attribute t. Augmenting this formula to a more general case which does not involve disjunction, would lead to the following equation:

$$N = \prod_{i=1}^{r} F_i^{d_i} \qquad (3)$$

where $F_i^{d_i}$ is an average fanout of the tree for attribute t.

4.3 Assigning Similarity Value

For each combination, a similarity value is calculated using similarity between values of attributes specified by the user and values of combinations created in the previous stage. The calculation is similar to the weighted nearest neighbor, except that not all attributes are involved. The CARET algorithm does not compute any attributes not specified by the user. The rationale for this approach is described in [Kitano *et al.*, 1992]. Whether the user specified the attribute or not is shown in a mask matrix (M), which is a one-dimension matrix whose size is equivalent to the case-base degree. The matrix element M_i, will be 1, if the user specified the value for the attribute *i.* Otherwise, M_i, will be 0. The formula for calculating the similarity is as follows:

$$S(Q,F) = \frac{\sum_{i=1}^{n} M_i \times W_i \times s(Q_i, F_i)}{\sum_{i=1}^{n} M_i \times W_i} \qquad (4)$$

where *F* is an NVS combination and *Fi* is the i-th attribute for the combination. It should be noted that similarity is calculated between the user specified attributes and combinations of NVSs which is the seed for SQL specifications. In essence, the similarity is computed between a user's specification and SQL specifications. This is counter to traditional methods, which compare a user's query specification with each case instance. For example, the similarity of a combination, ([C, C++, COBOL, COBOL/S], [SUN, NEWS, ...]) to the user's query specification is calculated as follows:

$$\frac{0.3 \times 0.2 + 0.4 \times 0.8}{0.3 + 0.4} = 0.54 \qquad (5)$$

The similarity value is 0.54 because only attributes language and machine are involved (the user specified only these attributes), whose weights are 0.3 and 0.4, respectively. The similarity between ADA and [C, C++, COBOL, COBOL/S] is 0.2 and that for VAX and [SUM, MEWS, ...] is 0.8.

4.4 Thresholding and N-Best Match

When there are too many combinations, translating all combinations and dispatching all SQL specifications are inefficient and wasteful. Methods to limit the number of SQL specifications to be created are necessary in real deployment. Two approaches are incorporated in the CARET system.

The first method is the N-Best match. CARET dispatches SQLs from highest similarity score, and counts the number of cases retrieved to stop the retrieval, when the number of retrieved cases exceeds a predetermined number.

Second, a threshold can be set in order to dispatch SQL specifications, which are sufficiently similar to the user's problem specifications. SQL specifications below a specified threshold will not be created.

4*5 Generating SQL Specifications

Each combination is translated into a corresponding SQL specification. Since SQL does not involve the similarity measure, the value assigned in the previous process is stored in CARET, and is referred to when the query results are returned.

The only SQL expression type used here is the SELECT-FROM-WHERE type. Its form is

SELECT *fields list*
FROM *case-base table*
WHERE *field conditions;*

This should read "Select records in a case-base table which satisfy specific field conditions in the WHERE clause, and return the value of the requested fields in the SELECT clause from the selected records".

Each element in a specific combination is translated into a certain condition expression of SQL. For example, machine(EVS4800) is translated into (machine = EVS4800) which means that the attribute machine must be EWS4800. Language(or(C, C++, COBOL, COBOL/S)) is translated into (language in (C, C++, COBOL, COBOL/S)) which means that the attribute language must be C, C++, COBOL, or COBOL/S. Then, each condition expression is connected with the logical-AND operators. For example, and(machine(VAX), language(or(C, C++, COBOL, COBOL/S)) is translated into (machine = VAX) and (language in (C, C++, COBOL, COBOL/S)).

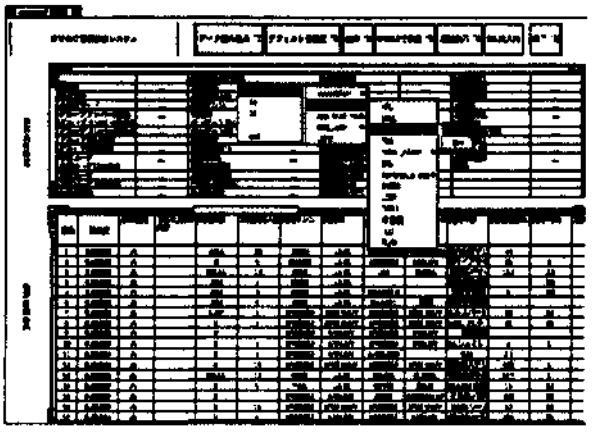

Figure 4: SQUAD-II Screen Image

As a result, the SQL specification would be produced. An example of the generated SQL is shown below.

```
SELECT *
FROM case-table
WHERE (machine = VAX) AND
      (language in (C,C++,COBOL,COBOL/S));
```

5 Performance Evaluation

CARET performance was evaluated using the SQUAD-II system which is a deployed case-base system built on CARET using ORACLE, a commercial RDBMS. SQUAD-II is a direct descendent of the SQUAD system which has been applied to the software quality control (SWQC) domain. The prime difference between SQUAD and SQUAD-II is the use of RDB in SQUAD-II, where SQUAD was built on a home-made case-base manager. SQUAD-II users are assumed to choose several attributes and their values which describes the problem which the user is facing. The user's choice, then, is interpreted by CARET, embedded in SQUAD-II, to produce SQL specifications. SQL specifications are sent to ORACLE to retrieve cases of software quality control. Results are shown in the SQUAD-II screen (Figure 4).

The experiments were carried out on SUN Sparc Station 2, using ORACLE version 6 installed on SunOS version 4.1.2. Figure 5 shows the response times measured for three user queries, and various sizes of case-bases. The three queries are:

Rank	Similarity	SQL Specification (only WHERE clause is shown)
1	1.00	(language = ADA) and (machine = VAX)
2	0.89	(language = ADA) and (machine in (SUN, NEWS, ...))
3	0.66	(language in (C, C++, COBOL, COBOL/S)) and (machine = VAX)
4	0.54	(language = ADA) and (machine in (MIPS, EWS4800, ...))
4	0.54	(language in (C, C++, COBOL, COBOL/S)) and (machine in (SUN, NEWS, ..))
6	0.2	(language in (C, C++, COBOL, COBOL/S)) and (machine in (MIPS, EWS4800, ...))

Table 3: SQL Specifications for Query-2

Factors	Query-1	Query-2	Query-3
Query length	1	2	3
Tree depth	3	2 × 3	2 × 2 × 2
Tree width	16	16+12	8+8+9
Generated SQL number	3	6	4
Cases matched	158+4+199	0+0+11+0+0+94	2+0+0+0

Table 4: Query Characteristics

Query-1: LANGUAGE = C

Query-2: (LANGUAGE = ADA) and (MACHINE = VAX)

Query-3: (PROBLEM-TIME-BEFORE-QC = SYSTEM-GENERATION-TIME) and (PURPOSE-TIME-OF-QC = SYSTEM-GENERATION-TIME) and (CHOSEN-METHOD = CHANGE-IN-PROCESS-SEQUENCE)

For query-2, the following SQL specifications and their similarity values are derived (See Table 3).

Characteristics for each query are shown in Table 4. Query length refers to the number of conjunctive clause in the WHERE clause. Tree depth shows abstraction hierarchy depth for each attribute. Tree width is the number of terminal nodes of the abstraction hierarchy of each attribute. Generated SQL number is the number of SQLs generated and actually sent to RDBMS. Cases matched shows the number of cases matched in each query. 158+1-4_1199 should read as 158 matches at the first SQL, 4 matches at the second SQL, and 199 matches at the third SQL. These numbers are measured with a case-base containing 800 cases.

The algorithm scalability has been tested by increasing the number of cases in the case-base. The number of cases was increased up to 1,600. Response times (in real-time, not in CPU time) are shown in Figure 5.

Query-1 returned the major part of cases in the case-base, resulting in slower response time due to the necessity to retrieve these cases. The response time for the query-1 increases linearly as the case-base size increases. Indexing methods for the commercial database systems are not effective for reducing response time in such situations. Also, most of the time is spent on retrieving matched cases, rather than on matching itself.

Fortunately, however, users generally specify a query in a much more detailed manner. An empirical analysis, using the SQUAD system, showed that the average number of specified attributes for each retrieval was 3.4 out of 75 features [Kitano *et* a/., 1992]. This would help RDBMS to constrain the search and the query time would be dramatically shorter. The second and third queries confirm this observation.

The second and the third queries show a nearly constant time responses, regardless of the case-base size. This is due to the fact that SQL specifications are specific enough to constrain the search on RDBMS. Since the system provides fast response time, it would suffice for most tasks.

A brief analysis of the performance results would shed light on the system behavior. There are two major factors which affect retrieval speed. They are: number of SQL specifications actually dispatched to the RDBMS, and number of cases retrieved. The maximum number of the SQL specifications which may be created is decided by a number of specified attributes by the user and the depths of the tree of the specified attribute (Equation 2). Longer response time is required with a larger number of cases to be retrieved. The number of cases to be retrieved depends upon the specificity employed in the SQL specifications. Query-1 resulted in longer response time than query-2 and query-3, because a large number of cases had to be retrieved. Query-2 and query-3 attained faster response time, because SQL specifications were specific.

6 Conclusion

This paper proposes a new method which enables the use of RDBMS as a case-base management system. Although case-base management, such as data-security and data-integrity, which are readily available in RDBMS, has not been the subject of significant case-base reasoning (CBR) research, case-base management functions are essential to the integration of CBR systems with mainstream corporate information *systems.*

This paper has proposed a novel method to implement similarity-based retrieval using SQL, hereby enabling CBR systems to use RDBMS. The algorithm was implemented as a CARET case retrieval tool. CARET generates SQL specifications of varying degrees of similarity, and the generated SQL specifications are dispatched to

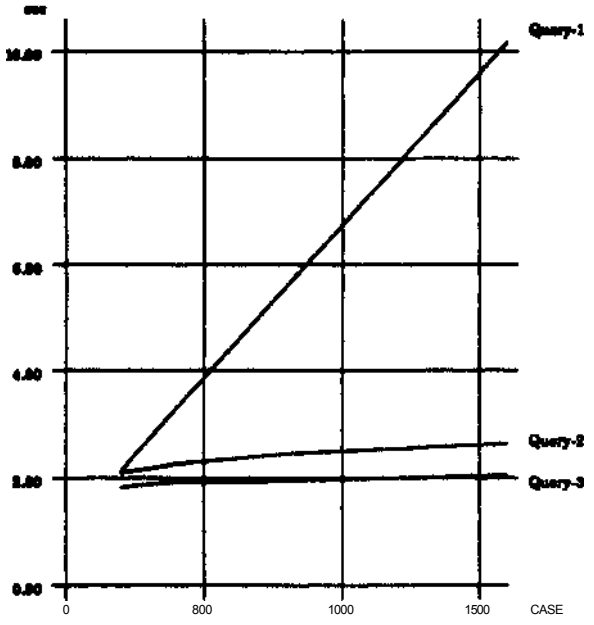

Figure 5: Response Times for Three User Queries

RDBMS to retrieve cases from RDB. The method we have proposed here determines a pre-compute similarities for individual query forms, rather than computing the similarities of individual cases.

A performance test using SQUAD-II, a corporate-wide CBR system built on CARET, has demonstrated that the proposed method attains acceptable performance and suggests that the proposed method is practical for large-scale case-bases used in workstations. The proposed method, then, is well-suited to serve as a basic algorithm for corporate-wide large-scale case-based systems.

References

[ANSI, 1989] ANSI Database Language SQL with Integrity Enhancement. ANSI X3.135.1-1989, 1989.

[Ashley and Rissland, 1987] K.D. Ashley and E.L Rissland. Compare and Contrast, A test of Expertise. In Proceedings of AAAI-87, 1987

[Chamberlin et a/., 1976] Chamberlin, D.D., et al., SEQUEL2: A Unified Approach to Data Definition, Manipulation, and Control. IBM J. Res. Develop., 1976.

[Codd, 1970] Codd, E.F.. A Relational Model of Data for Large Shared Data Banks. In Communications of ACM, 13, 6, 1970.

(Hammond, 1986] Hammond, K. Case-Based Planning: An Integrated Theory of Planning, Learning, and Memory. Ph.D. Thesis, Yale University, 1986.

[Hunter, 1988] Hunter, L„ The Use and Discovery of Paradigm Cases Ph.D. Thesis, Yale University, 1988.

[Kitano et a/., 1992] Kitano, H., Shibata, A., Shimazu, H., Kajihara J., Sato, A., Building Large-Scale Corporate-Wide Case-Based Systems: Integration of Organizational and Machine Executable Algorithms. In Proceedings of AAAI92, 1992.

[Kitano et a/., 1993] Kitano, H., Shimazu, H., Shibata, A., Case-Method: A Methodology for Building Large-Scale Case-Based Systems. In Proceedings of AAAI93, 1993.

[Kolodner, 1984] Kolodner, J., Retrieval and organizational strategies in conceptual memory: A computer model. Lawrence Erlbaum Associates, Hillsdale, NJ., 1984.

[Oracle, 1989] Oracle Database Administrator's Guide, and other ORACLE manuals. Oracle Corporation, 1989.