Managing Hierarchies and Taxonomies in Relational Databases

Ido Millet
Penn State Erie, USA

INTRODUCTION

The need to maintain classification and retrieval mechanisms that rely on concept hierarchies is as old as language itself. Familiar examples include the Dewey decimal classification system used in libraries and the system for classifying life forms developed in the 1700s by Carolus Linnaeus. A more recent example is Yahoo's subject taxonomy.

Information technology has led to an explosive growth in digital documents, records, multi-media files, and Web sites. To facilitate end-user access to these resources, topic hierarchies are frequently maintained to allow intuitive navigation and searching for resources related to specific categories. Maintaining such document, Web site, and knowledge taxonomies within relational databases is not trivial since the management of any hierarchical data in relational databases poses significant challenges (Millet, 2001). Taxonomies pose an additional challenge due to the typical need to classify a single document, concept, or Web site under multiple topics and due to the typical reliance on intelligent keys (Millet, 2003).

While, according to some views, non-relational database technologies and dynamic classification schemes may offer better ways for achieving our objectives, the use of relational database technology and concept taxonomies remains a contemporary practice and challenge.

BACKGROUND

Consider a document database where each document is classified into a hierarchy of topics shown in Figure 1.

Since each document may belong to more than one parent topic, we cannot record the data for this hierarchy by specifying in each topic record the topic above it. Figure 2 shows a data model for this situation. Note that the classify table allows us to assign a single document to multiple topics. If instead of a topic hierarchy, we need to assign each subtopic to more than one parent topic, we would insert a topic assignment table to represent such taxonomies.

To demonstrate the difficulty of hierarchical data retrieval against the normalized data model in Figure 2, consider the following requests:

- Show a list of all documents (at all levels) under Topic 1.
- Show how many documents (at all levels) are classified under each topic at level 1 of the hierarchy.

Using SQL, we can easily join each topic to all the documents associated with it via the classify records. However, we cannot easily identify the documents indirectly associated with a topic via its subtopics (at all levels). This difficulty in locating parent or child nodes at any given level is at the heart of the problem.

SQL-BASED SOLUTIONS

A request to show how many documents belong to each main topic, including all subtopics below it, can be handled using the SQL:1999 (ANSI/ISO/IEC 9075-2-1999) query shown in Listing 1. This query starts by creating a table expression (TOPIC_PATHS) populated with all main topic records as parents of themselves and appends (UNION) records for all paths of length one from these nodes to the topics directly below them. The RECURSIVE option continues the process to build all indirect paths from each topic to all its descendants.

The query then joins the end points (Topic_Below) of all paths in the TOPIC_PATHS result set to the documents assigned to these topics. By limiting the start points of these paths to main topics (topics at level 1) and

Figure 1. A topic hierarchy

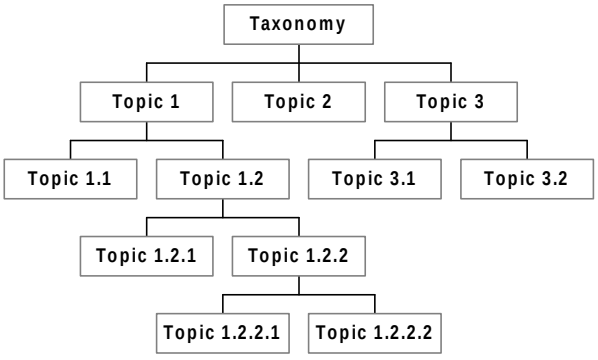
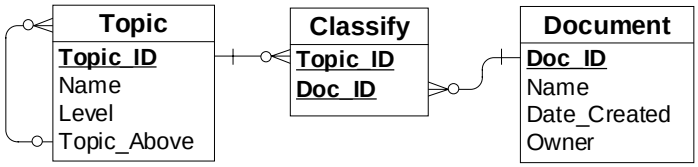


Figure 2. A normalized data model with a topic hierarchy



Listing 1. Recursive hierarchy retrieval using SQL:1999

```
WITH RECURSIVE TOPIC_PATHS (topic_above, topic_below, level) AS
(SELECT topic_id, topic_id, level FROM TOPIC
UNION ALL
SELECT TOPIC_PATHS.topic_above, TOPIC.topic_id, TOPIC_PATHS.level
FROM TOPIC_PATHS, TOPIC
WHERE TOPIC_PATHS.topic_below = TOPIC.topic_above)

SELECT TOPIC_PATHS.topic_above, DistinctCount (CLASSIFY.Doc_ID)
FROM TOPIC_PATHS, CLASSIFY
WHERE TOPIC_PATHS.topic_below = CLASSIFY.Topic_ID AND
TOPIC_PATHS.level = 1
GROUP BY TOPIC_PATHS.topic_above;
```

grouping the end result by those topics, we get the requested information. Avoiding double-counting of documents that were assigned to multiple topics is achieved by using *DistinctCount*(Classify.Doc_ID) rather than *Count*(Classify.Doc_ID).

Relying on such complex SQL is probably beyond the reach of many IT professionals. This can be addressed by implementing the complex portion of these SQL statements as database views. However, someone has to write the SQL for these views and the intensive nature of the required processing may lead to slow performance in reporting applications with large hierarchies and frequent queries.

Celko (2000) reports on a technique leading to significant improvements in query speeds by storing the hierarchy data not as parent-child relationships but as "nested sets" using a somewhat complex numbering scheme. However, alternative approaches can achieve very significant query performance gains while maintaining intuitive data storage and SQL syntax.

THE PATH TABLE APPROACH

The path table approach uses a "navigation bridge table" (Kimball et al., 1998) with records enumerating all paths starting from each node to all nodes in the branch above it, including itself. This approach provides flexibility in the sense that each subtopic node can belong to multiple direct parent topics and there is no limit on the number of levels in the taxonomy.

As demonstrated by Table 1, topic 1.1.1 would require four records in the path table reflecting the paths up to itself, topic 1.1, topic 1, and topic 0 (the top node of the hierarchy). These are just 4 of the 37 records required to capture all paths for the sample hierarchy in Figure 1.

To demonstrate how the path table can simplify data retrieval, consider the same challenge of showing how many documents belong to each main topic, including all subtopics below it. By joining the tables as shown in Figure 3, we can easily select all documents that belong to topics below each main topic. Since the path table includes a zero-length path between each topic and itself, documents that belong directly to a main topic would be included in the result set. Again, the classify table allows us to associate the same document with multiple topics.

Other requests for information can use the same approach or variations such as connecting to the topic table via the Topic_ID column in the path table or adding path length and terminal node information to the path table (Kimbal et al., 1998).

One limitation of the path table approach is that the number of records in the path table can grow quite large for deep hierarchies. The following section describes another approach that avoids that problem.

THE DENORMALIZED TOPIC TABLE APPROACH

The denormalized topic table approach maintains information about all higher-level topics within each topic