Daytona And The Fourth-Generation Language Cymbal

Rick Greer
AT&T Labs - Research
Florham Park, NJ 07932
rxga@research.att.com

Abstract

The Daytona™ data management system is used by AT&T to solve a wide spectrum of data management problems. For example, Daytona is managing a 4 terabyte data warehouse whose largest table contains over 10 billion rows. Daytona's architecture is based on translating its high-level query language Cymbal (which includes SQL as a subset) completely into C and then compiling that C into object code. The system resulting from this architecture is fast, powerful, easy to use and administer, reliable and open to UNIX™ tools. In particular, two forms of data compression plus robust horizontal partitioning enable Daytona to handle terabytes with ease.

1 Daytona

The Daytona™ data management system is used by AT&T to solve a wide spectrum of data management problems. On the tiny end, Daytona provides the data manager for the DACS VI switch which only had 64MB of memory at the time. Since DACS VI used a real-time UNIX operating system, virtual memory could not be paged to swap disk. Consequently, the entire application, including the 15% that was allocated to the database, had to fit into the rather small amount of physical memory at all times. At the high end, SCAMP, the Security Call Analysis And Monitoring Platform, uses the same Daytona as DACS VI to manage sequential and direct access to 9 weeks of all of AT&T's call detail data, comprising more than 10 billion records in a single table (plus four other large collections of call detail and summary data). SCAMP is used to analyze and detect fraud perpetrated against the company and to fulfill (often emergency) information requests from law enforcement. SCAMP handles more than 70,000 queries a month.

Daytona offers all the essentials of data management including a high-level query language, B-tree indexing, locking, transactions, logging, and recovery. Users are pleased with Daytona's speed, its powerful query language, its ability to easily manage large amounts of data in minimal space, its simplicity, its ease of administration, and its openness to other tools.

In contrast to Daytona, other DBMS are much larger and tend to be closed systems (relatively speaking): they have chosen to implement their own (server-based) operating system, their own networking, user/login administration, performance monitoring, source code control and stored procedure management, and so on, and in some cases, even mail and cron job handling. Instead, Daytona reuses and leverages the software in its working environment. This makes it much smaller, simpler, and more open; in particular, Daytona users can work with Daytona using many of the tools they already have and know. Let's see how Daytona's low-overhead architecture and its query language lead to these several benefits.

2 Daytona Architecture

Daytona revolves around Cymbal, its multiparadigm query language. Cymbal is processed by translating it completely into C (and a makefile). It is this unusual code-generation-based architecture that enables one and the same Daytona to handle with ease problems ranging from doing embedded data management on small real-time systems to managing the 4 terabyte SCAMP call detail warehouse (with its 16 gigabytes of memory and 32 250MHz processors).

2.1 Three Modes Of Use

This architecture supports three modes of use:

- *Ad Hoc Queries.* The simplest mode occurs when the Daytona user asks the system to translate, C-compile and run a Cymbal query.
- *Pre-compiled.* Applications also have the option of pre-compiling parameterized queries. The applica-

SIGMOD '99 Philadelphia PA

tion's GUI collects the parameters needed to invoke the previously compiled executables, whose output is returned to the GUI. This is the analog of SQL-based stored procedures.

- *Code Synthesis.* The application writer can also use Daytona as a silicon programmer to generate C code according to high-level (meaning Cymbal) specifications. The corresponding object modules are then linked with the application's own object modules (and Daytona's libraries) into a single executable. In this synthesis of user and Daytona-generated code, user routines may call Daytona-generated routines which, in turn, may call user-coded C routines. This is an extremely efficient way to include data management in an application since Daytona is only a C function call away from the application code.

2.2 One Operating System Is Enough

Another implication of this architecture is that Daytona has no database server processes! In fact, it has no daemon processes of any kind. Every query executable is on its own to run and produce its answers. Most other DBMS have invested quite a bit of effort into creating database server processes, which provide many services including proprietary file systems, scheduling, caching, locking, parallelization, security, networking, and of course, query optimization and execution. Notice that with the exception of the last two, all of these services are provided by modern day operating systems to one degree/flavor or another. Instead of implementing another operating system, Daytona cuts out the middleman and in effect, uses the UNIX operating system itself as Daytona's server process.

Daytona's approach has several advantages. First, the same services are not being implemented twice and furthermore, one does not encounter the interference risks of "too many cooks spoiling the broth". Thus, Daytona is a far smaller DBMS than most. Consequently, it can fit on smaller machines and there is much less code to maintain (and correspondingly, much less opportunity for things to go wrong). As a result, Daytona has much easier OA&M (Operations, Administration, And Maintenance) requirements than most. As just one indicator, instead of the dozens of processes some other DBMS need to invoke on startup and keep healthy, Daytona has none at all; if your computer is up, Daytona is up.

2.3 Filesystem Issues

Daytona not only uses UNIX filesystems to store its data but the user even has the option of storing their data in the awk/Perl-compatible ASCII format of delimiter-separated fields, new-line terminated records. The use of this open format is reassuring to many users because they can actually see their data in their favorite text editor and because they can use standard UNIX tools on their data in the same form that is used by Daytona. Contrast this instead with storing data in a binary, proprietary format in 2K blocks, each containing a directory of pointers to slots within the block.

Daytona's data format can also optionally include field- and record-level compression. At the field level, various tricks are used such as eliding default values and using special ASCII code bytes to represent pairs of digits. At the record level, a static dictionary of strings is computed for the table in question and the table is compressed record-by-record by replacing dictionary strings with 8 bit codes. (The advantage of compressing each record individually is that B-trees can still point to the first bytes of (compressed) records and consequently, there is no need to decompress an entire file in order to read out a particular record of interest.) Each of these compression levels has proven capable of 50% reduction. When needing to store a terabyte, it is better to store 250 gigabytes.

3 The Cymbal Query Language

Cymbal is a multiparadigm, fourth generation language that seamlessly integrates a procedural dialect with a first-order logic subset, ANSI 89 SQL, a sublanguage having to do with (declarative) set/list-formers and another one for describing database records. The procedural dialect includes assignments, conditionals, loops, function definitions, and compilation units called tasks. The first-order logic component is a domain calculus that employs the full assortment of connectives and quantifiers in unconstrained first-order combinations and is treated in a model-theoretic manner. The process of finding all values for the free variables in an assertion is handled in a backtracking manner reminiscent of Prolog but without using Horn clauses or being unduly sensitive to the order of the conjuncts. Cymbal's SQL dialect is implemented by translating it into the first-order logic component. A fluent Cymbal programmer freely intermixes all of the dialects in their query programs according to which is the most convenient, powerful, or concise at the time.

4 Pointers

Much more on Daytona appears in the electronic proceedings of this conference. See also http://www.research.att.com/projects/daytona. Daytona can be obtained through Global Technologies, Ltd., an AT&T VAR. See http://www.gtlinc.com.