The Adabas Fastpath Optimization Solution

The Adabas Fastpath Optimization Solution

This section describes the solutions provided by Adabas Fastpath for optimizing database access:

- Database Query Optimization
- Database Read-Ahead Optimization

Database Query Optimization

This section describes the database query optimization provided by Adabas Fastpath:

- The Concept of Data Popularity
- Identifying Popular Data
- Building a Query Knowledge Base
- Sharing a Query Knowledge Base
- Integrity of Query Results

The Concept of Data Popularity

Software AG's considerable experience in DBMS technology indicates that as much as 80% of DBMS queries are directed at only 20% of the data. This is commonly referred to as *data popularity*.

At any given time, some amount of data is popular. However, today's popular data may not be the same as yesterday's or tomorrow's. For computer systems, the time span of popularity is probably not days, but rather seconds or less.

The popularity of any particular data in the DBMS tends to vary according to both commercial and system activity. In this sense, the popular data in the DBMS is a constantly moving target.

To effectively optimize queries to this data, it is necessary to identify and constantly re-evaluate the data that is popular.

How Adabas Fastpath has resolved this complex issue is described in the next section.

Identifying Popular Data

Adabas Fastpath provides the sampling tool AFPLOOK to identify the most commonly used data.

AFPLOOK can be used to quickly identify exactly which database queries can be optimized and their relative popularity.

Building a Query Knowledge Base

Adabas Fastpath accumulates a knowledge base by building a number of indexed query models. Each query model represents a set of query results that are available for optimization. The results within a set are indexed by search argument, ISN, etc. to provide an efficient means for locating results.

Once started, Adabas Fastpath analyses the queries being made and the data that is currently popular. For each type of query, various algorithms are used to

- recognize and retain the most popular data; and
- discard or overwrite the least popular data.

Adabas Fastpath does not attempt to load all data because not all data is used at the same time.

Rather, you instruct Adabas Fastpath to use a particular amount of memory. Adabas Fastpath will then make the best use of that memory by retaining the results of popular data queries so that they can be resolved in the client process when repeated. Large Objects (LOBs) can be retained by Adabas Fastpath but due to their size they are unlikely to offer the best use of the memory available to Fastpath.

Sharing a Query Knowledge Base

Adabas Fastpath's knowledge base is maintained in memory that is available to all clients within an operating system. One client may therefore benefit from the results collected from another client query.

Before a query is passed to the DBMS, the Adabas Fastpath optimizer in the client process attempts to resolve the query from the knowledge base. If the query is satisfied, response times are shortened and processing overhead is reduced significantly because:

- less time is needed for the query;
- interprocess communication is avoided; and
- DBMS access is avoided.

The least popular data held by Adabas Fastpath in the dynamic knowledge base is overwritten. This means that the longer Adabas Fastpath is active, the more certain you can be that the most popular data currently in use is retained.

Integrity of Query Results

Using the Adabas Fastpath solution, all clients receive data that is absolutely consistent with the DBMS at all times. An Adabas Fastpath component attached to the DBMS ensures that any changes to popular data are reflected in the results returned.

Database Read-Ahead Optimization

This section describes the database read-ahead optimization provided by Adabas Fastpath:

- The Concept of Read-Ahead Sequences

- Identifying and Processing Read-Ahead Sequences

The Concept of Read-Ahead Sequences

In a direct access query, the client identifies the data being sought using search data. There is another type of query where the client wishes to access a series of data items that are related by perhaps sequence or search criteria.

This series of related access is viewed as a *sequence* by Adabas Fastpath. A sequence may be initiated by a direct access query to position to the data required but once that is done, the client enters into an iteration where the next data in the sequence is requested.

Adabas Fastpath optimizes sequences by dynamically applying *read-ahead logic* to the query so that the DBMS workload is again reduced. Adabas Fastpath uses the Adabas multifetch option to perform the read-ahead operation.

Using read-ahead, many data items can be retrieved in one access to the DBMS. Adabas Fastpath then satisfies subsequent client queries from within the client process.

Identifying and Processing Read-Ahead Sequences

The Adabas Fastpath query sampler identifies those queries that are used for read-ahead access.

When you direct Adabas Fastpath to apply read-ahead logic to a file, an algorithm is used to decide the read-ahead rate so that the maximum possible benefit is achieved. If Adabas Fastpath simply applies a read-ahead rate of 1, 50% of the interprocess communications to the DBMS for a sequence are saved. This is because for every data item requested by the client, Adabas Fastpath causes an additional data item to be returned with it.

Read-ahead processing handles both online and batch sequences, although online sequences tend to be much shorter than batch ones.

For example, an online transaction may allow entry of a surname such as SMITH. The DBMS may contain many data items for SMITH, so a sequence is used to show the first page of SMITH data items on a selection screen. The user may select a particular data item for more detailed processing or signal for the next page, thereby continuing the sequence.