# Extracting Large Data Sets using DB2 Parallel Edition

Sriram Padmanabhan

IBM T.J. Watson Research Center
srp@watson.ibm.com

Commercial parallel database systems such as DB2 Parallel Edition (DB2 PE) [1, 2] are delivering the ability to execute complex queries on very large databases. However, the serial application interface to these database systems can become a bottleneck for a growing list of applications such as mailing list generation and data propagation from a warehouse to smaller data marts. In this abstract, we describe the `CURRENT NODE` and `NODENUMBER` functions provided by DB2 PE and show how these two functions can be used to retrieve data in parallel in a linearly scalable manner with respect to the number of nodes in the system.

Before proceeding further, we should point out that DB2 PE uses a hash partitioning strategy to distribute rows of a table to nodes in a nodegroup which is a user-specified subset of system nodes. We apply a system-specified hashing function on the user-specified partitioning key values to generate a partition number. This number is used as an index into a partition map (which can be modified by users) to find the node number where the row will be stored.

## CURRENT NODE and NODENUMBER functions

The `CURRENT NODE` value function has a value of the node where the application is connected. `CURRENT NODE` does not have any arguments and a query could use the `CURRENT NODE` function wherever other value functions can be used, e.g., in the select list or in search conditions. The `NODENUMBER` function returns the node number of each row to which it is applied. The argument of this function is a column name, i.e., `NODENUMBER(col)`, but in reality the column name is an alias to obtain the table name and apply the function on the partitioning columns of the table.

**Proceedings of the 22nd VLDB Conference Mumbai(Bombay), India, 1996**

## Extracting data in Parallel

If the task is to perform a large extract on a single table T with the following SQL statement:

```
SELECT T.a, T.b,... FROM T
WHERE (arbitrary predicates)
```

then we can write an application using the following modified statement.

```
SELECT T.a, T.b,... FROM T
WHERE (arbitrary predicates) AND
   NODENUMBER(T.a)=CURRENT NODE
```

The application will return all rows residing on the node where it is connected. In order to perform the complete extract, one must issue this statement from all nodes in the nodegroup containing table T and combine the set of partial results. The user may find the list of nodes containing partitions of table T by querying the system catalog tables. DB2 PE's optimizer recognizes the predicate `NODENUMBER(T.a)` = `CURRENT NODE` and creates an optimized plan which executes the query only on one node. For a large table, the parallel extract is only bounded by the time to retrive any one partition of the table and therefore, the solution scales linearly with the number of nodes across which a table is partitioned. This basic scheme can be extended to support complex queries but we cannot present the details due to the space limit.

**Acknowledgement:** Thanks to L. Kollar, A. Jhingran, and T. Malkemus for their significant contributions to the design and develeopment of these functions.

## References

[1] Baru, C. K., et al. DB2 Parallel Edition. *IBM Systems Journal 34*, 2 (1995), 292–322.

[2] IBM. *DB2 Parallel Edition for AIX, Administration Guide and Reference*, Sept. 1995. Product Manual SC09-1982-00.