SyBase IQ v15.1

Bhakta, Tejash
Jain, Harsh Manoj

SyBase IQ

- High-performance decision support server designed specifically for data warehousing
- Cross-platform product
- Integrate data from diverse sources
    - IQ databases
    - Adaptive Server family
    - non-SyBase databases and flat files

SyBase IQ: Capabilities

- Designed to facilitate **DSS** applications.
- DSS - Decision support System
  - A software application designed to analyze data in order to support business decision making.
- Data warehouses - Data stores on which to build DSS.

SyBase IQ has:

1. Unique Indexing System
2. Query Optimization
3. Concurrent Data Access - Multiplex SyBase IQ
4. Provide 24/7 access

Traditional indexes in SyBase

- B-tree index strategy
- Valuable only for small set of records
- Eg. Columns of order numbers, customer names
- Transaction-Processing System

SyBase IQ: Indexes

- How the data is stored?
  - Column-Store Architecture - Very useful in data warehousing environment
  - Focuses on select columns.
  - Logically, the data can still be accessed row-by-row.
  - Useful in data compression

- Column Index (default)
- Join Index

Column Index

- Default index that optimizes projections and storage is created for all columns
- Optimized Default Index
  - For columns with < 16M unique values - significantly reduces storage requirements
  - Improved performance on aggregate functions. Eg. SUM, MAX, MIN etc.

- Query performance can be improved by choosing additional index type on columns.

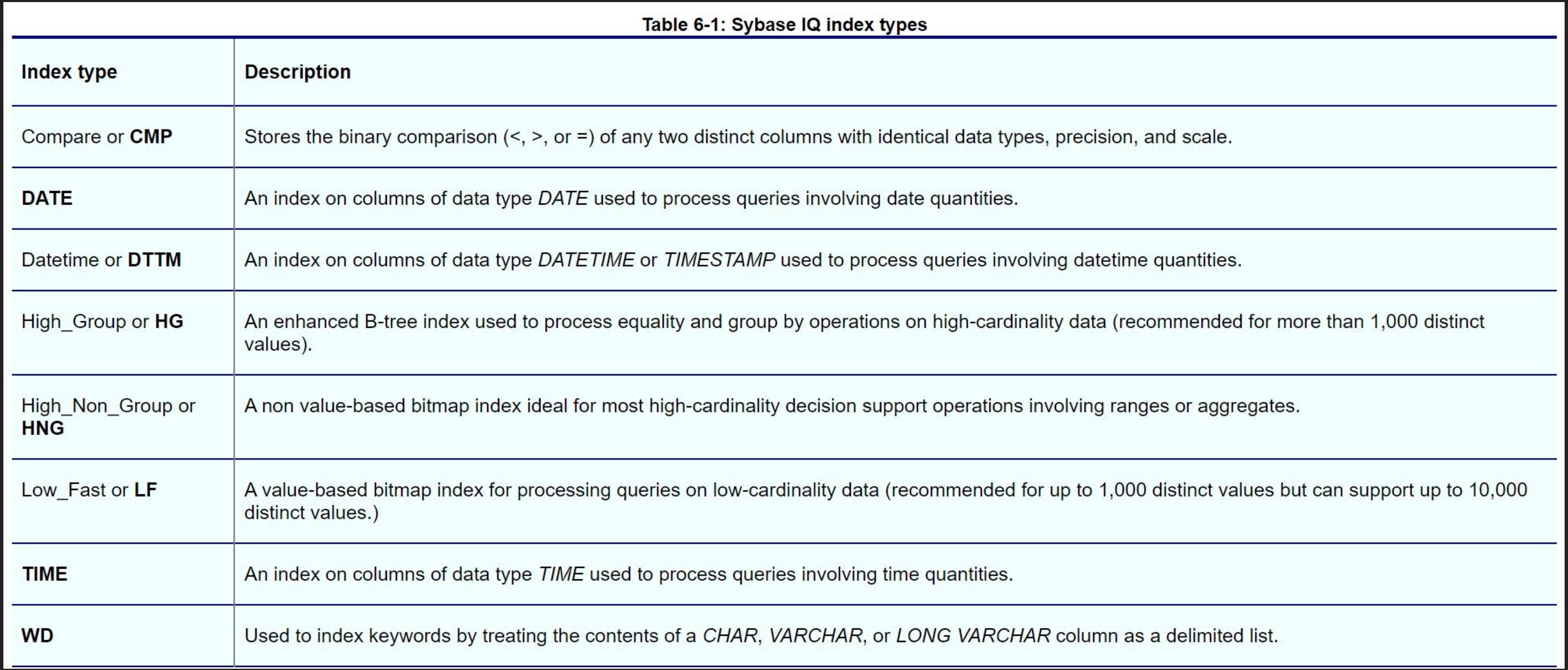


Table 6-1: Sybase IQ index types

Index type	**Description**
Compare or **CMP**	Stores the binary comparison (<, >, or =) of any two distinct columns with identical data types, precision, and scale.
DATE	An index on columns of data type *DATE* used to process queries involving date quantities.
Datetime or **DTTM**	An index on columns of data type *DATETIME* or *TIMESTAMP* used to process queries involving datetime quantities.
High_Group or **HG**	An enhanced B-tree index used to process equality and group by operations on high-cardinality data (recommended for more than 1,000 distinct values).
High_Non_Group or **HNG**	A non value-based bitmap index ideal for most high-cardinality decision support operations involving ranges or aggregates.
Low_Fast or **LF**	A value-based bitmap index for processing queries on low-cardinality data (recommended for up to 1,000 distinct values but can support up to 10,000 distinct values.)
TIME	An index on columns of data type *TIME* used to process queries involving time quantities.
WD	Used to index keywords by treating the contents of a *CHAR*, *VARCHAR*, or *LONG VARCHAR* column as a delimited list.

SyBase IQ 15.1- System Administration Guide: Volume I, http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00170.1510/pdf/iqapgv1.pdf, pg.243

Index Guidance

- INDEX_ADVISOR - from the optimizer
  - Issues messages to suggest additional indexes that might improve performance

- Choosing an index type
  - Number of unique values
    - Below 1K - LF (HG if table has more than 25K rows)
    - More than 1K - HG and/or HNG
  - Types of Queries
  - Disk space usage
  - Data types

Types of Queries

- Will the column be a part of a join predicate?
- If the column has high number of unique values, will the column be used in GROUP BY clause, be the argument of a COUNT DISTINCT, and/or be in the SELECT DISTINCT projection?
- Will the column frequently be compared with another column of the same data type, precision, and scale?

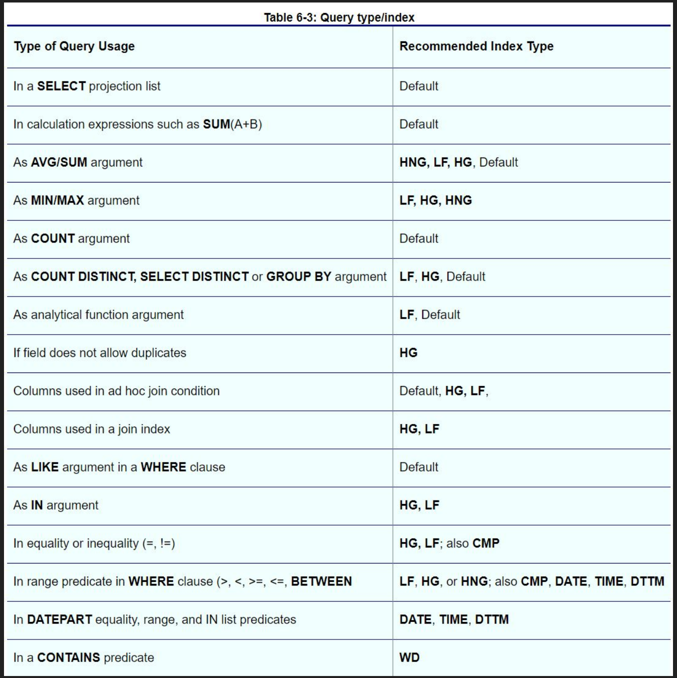


Table 6-3: Query type/index

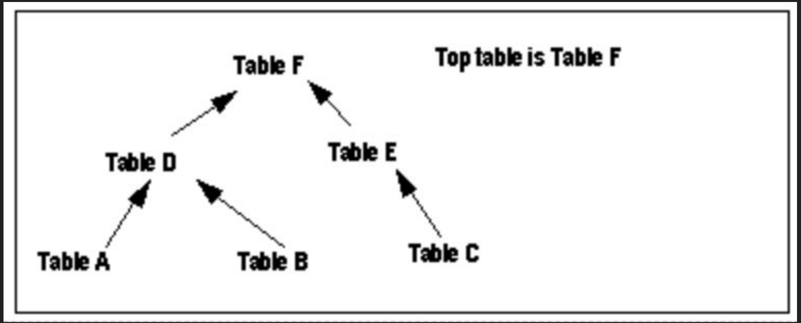
Type of Query Usage	Recommended Index Type
In a **SELECT** projection list	Default
In calculation expressions such as **SUM**(A+B)	Default
As **AVG/SUM** argument	**HNG, LF, HG**, Default
As **MIN/MAX** argument	**LF, HG, HNG**
As **COUNT** argument	Default
As **COUNT DISTINCT, SELECT DISTINCT** or **GROUP BY** argument	**LF**, **HG**, Default
As analytical function argument	**LF**, Default
If field does not allow duplicates	**HG**
Columns used in ad hoc join condition	Default, **HG, LF**,
Columns used in a join index	**HG, LF**
As **LIKE** argument in a **WHERE** clause	Default
As **IN** argument	**HG, LF**
In equality or inequality (=, !=)	**HG, LF**; also **CMP**
In range predicate in **WHERE** clause (>, <, >=, <=, **BETWEEN**	**LF**, **HG**, or **HNG**; also **CMP**, **DATE**, **TIME**, **DTTM**
In **DATEPART** equality, range, and IN list predicates	**DATE**, **TIME**, **DTTM**
In a **CONTAINS** predicate	**WD**

Join Indexes

- Create join index for tables that will be consistently joined.
- Improves query performance compared to when table joins are first defined at query time.
- Requires more space and time to load.
- Supports one-to-many join relationships.

Join Indexes - continued

- Hierarchy
- Two types of Join Index
  - Linear Joins
    - Tables A-D-F and C-E-F
    - Each pair holds a one-to-many relationship
  - Star Joins
    - Tables D-F-E
    - Center Table - “many” side of the relationship (F)
    - Each table around it - “one” side



Limited Modification

- Cannot DROP any table that participates in a Join Index
- Cannot ALTER table to add, drop or modify column that participates in a Join Index.
- First step to drop the join index and then the respective column or table

Benefits over Traditional Indexes

- Index sizes remain small
- Efficient query retrieval
- Minimized I/O
- More data in memory

OLAP

OLAP is an efficient method of data analysis on information stored in a relational database.

You can analyze data on different dimensions, acquire result sets with subtotaled rows, and organize data into multidimensional cubes, all in a single SQL query.

Analytical functions which facilitate OLAP:

- GROUP BY clause extensions – CUBE and ROLLUP
- Analytical functions: AVG, COUNT, MAX, MIN, and SUM, STDDEV and VARIANCE
- Numeric functions – CEIL, EXP, POWER, SQRT, and FLOOR
- Window functions:
    - Windowing aggregates – AVG, COUNT, MAX, MIN, and SUM
    - Ranking functions – RANK, DENSE_RANK, PERCENT_RANK, and NTILE
    - Statistical functions – STDDEV, VARIANCE, WEIGHTED_AVG.

GROUP BY Clause Extensions

If you have many groupings to specify and want subtotals included, you can use the ROLLUP and CUBE extensions.

Extensions to the GROUP BY clause allows you to:

- Create a “data cube,” providing a multi dimensional result set for data mining analysis
- Create a result set that includes the original groups, and optionally includes a subtotal and grand-total row

Group by ROLLUP and CUBE: http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00800.1530/doc/html/san1276751017610.html

Group by ROLLUP

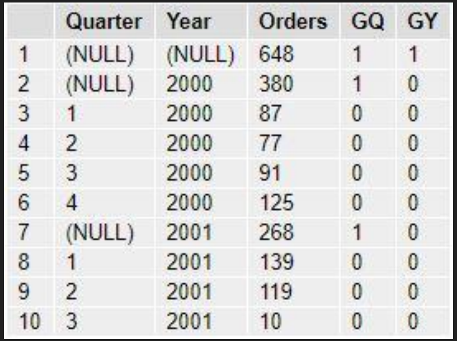
If n is the number of grouping columns, then ROLLUP creates n+1 levels of subtotals.

ROLLUP syntax:

```
SELECT … [ GROUPING (column-name) … ] …
GROUP BY [ expression [, …]
| ROLLUP ( expression [, …] ) ]
```

GROUPING takes a column name as parameter and returns a boolean value as listed:

Result	GROUPING returns
NULL created by ROLLUP	1
NULL indicating subtotal	1
Not created by ROLLUp	0
A stored NULL	0

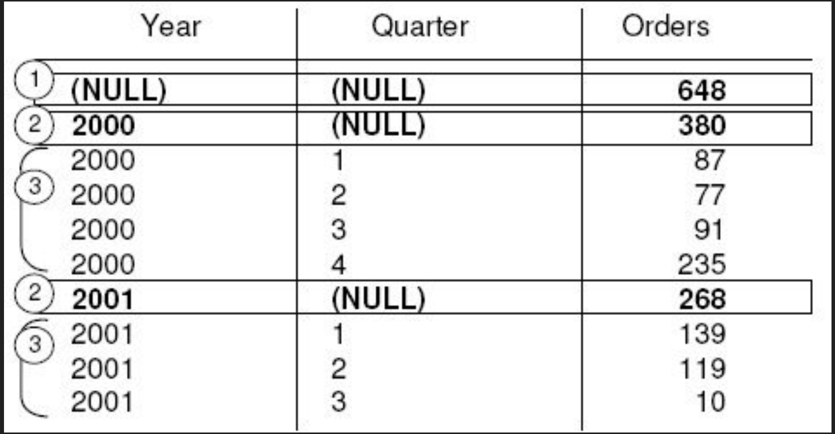


	Quarter	Year	Orders	GQ	GY
1	(NULL)	(NULL)	648	1	1
2	(NULL)	2000	380	1	0
3	1	2000	87	0	0
4	2	2000	77	0	0
5	3	2000	91	0	0
6	4	2000	125	0	0
7	(NULL)	2001	268	1	0
8	1	2001	139	0	0
9	2	2001	119	0	0
10	3	2001	10	0	0

ROLLUP Example

This query summarizes the number of sales orders by year and quarter:

```
SELECT year (OrderDate) AS Year, quarter (OrderDate) AS Quarter, COUNT (*) Orders
FROM SalesOrders
GROUP BY ROLLUP (Year, Quarter)
ORDER BY Year, Quarter
```



	Year	Quarter	Orders
1	**(NULL)**	**(NULL)**	**648**
2	**2000**	**(NULL)**	**380**
3	2000	1	87
	2000	2	77
	2000	3	91
	2000	4	235
2	**2001**	**(NULL)**	**268**
3	2001	1	139
	2001	2	119
	2001	3	10

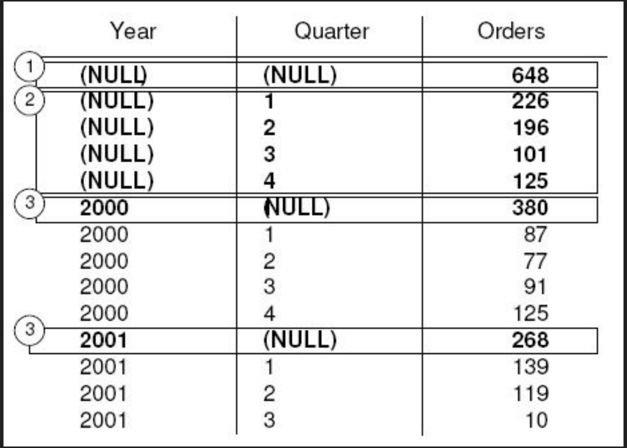
- Row [1] represents the total number of orders across both years (2000, 2001) and all quarters
- Rows [2] represent the total number of orders in the years 2000 and 2001, respectively
- Rows [3] provides the total number of orders for each quarter in both years

Group by CUBE

CUBE calculates subtotals for all possible combinations of the group of dimensions that you specify in the query. If n is the number of grouping columns, then CUBE creates 2^n levels of subtotals.

CUBE syntax:

```
SELECT … [ GROUPING (column-name) … ] …
GROUP BY [ expression [,…]
| CUBE ( expression [,…] ) ]
```



	Year	Quarter	Orders
1	**(NULL)**	**(NULL)**	**648**
2	**(NULL)**	**1**	**226**
	(NULL)	**2**	**196**
	(NULL)	**3**	**101**
	(NULL)	**4**	**125**
3	**2000**	**(NULL)**	**380**
	2000	1	87
	2000	2	77
	2000	3	91
	2000	4	125
3	**2001**	(NULL)	**268**
	2001	1	139
	2001	2	119
	2001	3	10

- Row [1] represents the total number of orders across both years and all quarters
- Rows [2] represents the total number of orders by quarter across both years
- Rows [3] represent the total number of orders across all quarters for the years 2000 and 2001, respectively

Sybase IQ features

Sybase IQ conforms to the ANSI SQL89 standard but has additional features not found in many other SQL implementations:

- Updates

  Sybase IQ allows more than one table to be referenced by the UPDATE command. Views defined on more than one table can also be updated. Many SQL implementations will not allow updates on joined tables.

- Joins

  Sybase IQ allows KEY joins between tables based on foreign-key relationships

- Dates

  Sybase IQ has date, time, and timestamp types that include year, month and day, hour, minutes, seconds and fraction of a second. The following operations are allowed on dates:

  - date + integer, Add specified number of days to a date
  - date - date, Compute the number of days between two dates

References

- SyBase IQ 15.1- System Administration Guide: Volume I, http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00170.1510/pdf/iqapgv1.pdf, pg.243
- SyBase IQ 15.1- System Administration Guide: Volume I, http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00170.1510/pdf/iqapgv1.pdf, pg.255
- Group by ROLLUP and CUBE: http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00800.1530/doc/html/san1276751017610.html