Information Management

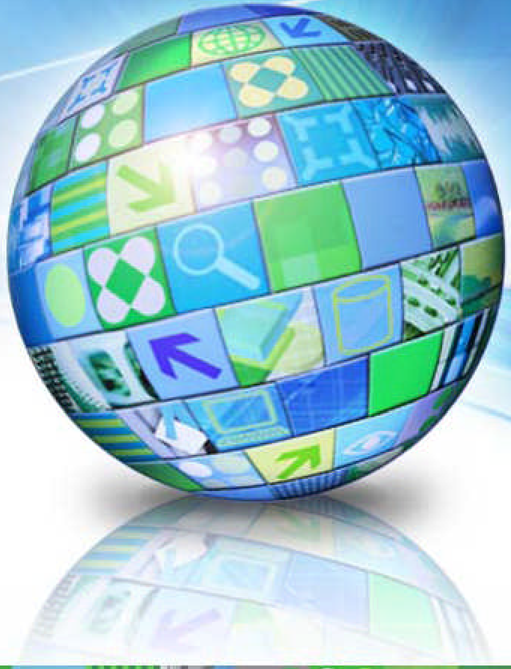
Educate. Inform. Entertain.

Growing the DB2 Community Since 2009

www.DB2NightShow.com

The DB2Night Show Episode #89

InfoSphere Warehouse V10 Performance Enhancements



Multi-Core Parallelism Improvements

- **What is new?**
  - Starting in DB2 10
    - Greater flexibility in controlling degree of parallelism through WLM for concurrent running applications and workloads
    - Better built-in runtime decision of parallelism degree when `ANY` is specified
    - DB2 10 reduces overhead for queries with no parallelism (`DEGREE=1`)
    - New REBAL plan operator rebalances work among subagents
    - Contention on various latches alleviated or eliminated
- **How to enable Multi-Core Parallelism?**
  - Multi-core parallelism is enabled by simply turning on the `INTRA_PARALLEL` dbm configuration parameter [`YES, NO`] and setting the degree of parallelism
    - `DFT_DEGREE` db configuration parameter
    - `SET CURRENT DEGREE [ANY,1,..,n]`
    - `BIND parameter DEGREE [ANY,1,..,n]`
    - `ALTER WORKLOAD myworkload MAXIMUM DEGREE [ANY,1..n]`

Greater Flexibility in Controlling Multi-Core Parallelism

- **Enable/disable parallelism within a database application**

```
CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL (‘YES’|‘NO’)
```

- **Controlling maximum parallelism degree through DB2 workload manager**
  - Set the `MAXIMUM DEGREE` workload attribute

```
ALTER WORKLOAD myworkload MAXIMUM DEGREE 2
```

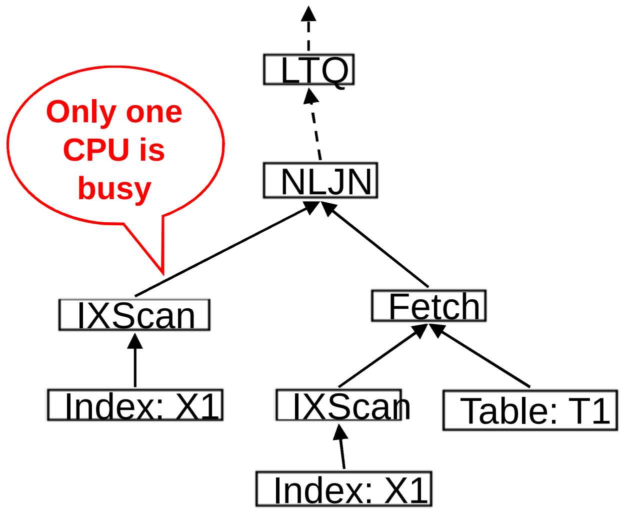
  - Only affects workload `myworkload`
  - Takes effect at transaction boundary
    - NO instance or database recycle

- **Precedence Order**
  - WLM: `MAXIMUM DEGREE`
  - Stored Procedure: `ADMIN_SET_INTRA_PARALLEL`
  - Instance level configuration parameter: `INTRA_PARALLEL`

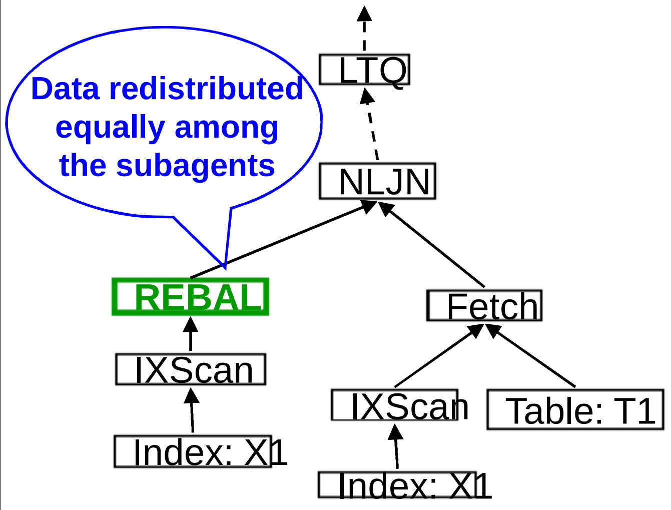
Parallelism and Workload Imbalance

- **Data filtering and data skew can cause workloads between subagents to become imbalanced**
- **Rebalance (REBAL) operator is a new access plan operator**
  - A mechanism for transferring rows between subagents to rebalance their workload
  - Light weight mechanism for rebalance
  - This operator puts underutilized subagents to work

Parallel plan with unbalanced workload



Rebalanced data stream



Jump Scan

- **What is Jump Scan?**
  - Improvement to avoid large and expensive index scans when the index has gaps

<u>Index:</u>
(c1, c2) [c1: **gap column**; c2 **non-gap column**]

<u>Query:</u>
```
SELECT    info
FROM      TPCD.LINEITEM T
WHERE     T.c2 = 10  (Equality Predicate on c2)
```
*(No predicate on c1 → **unconstrained gap**)*

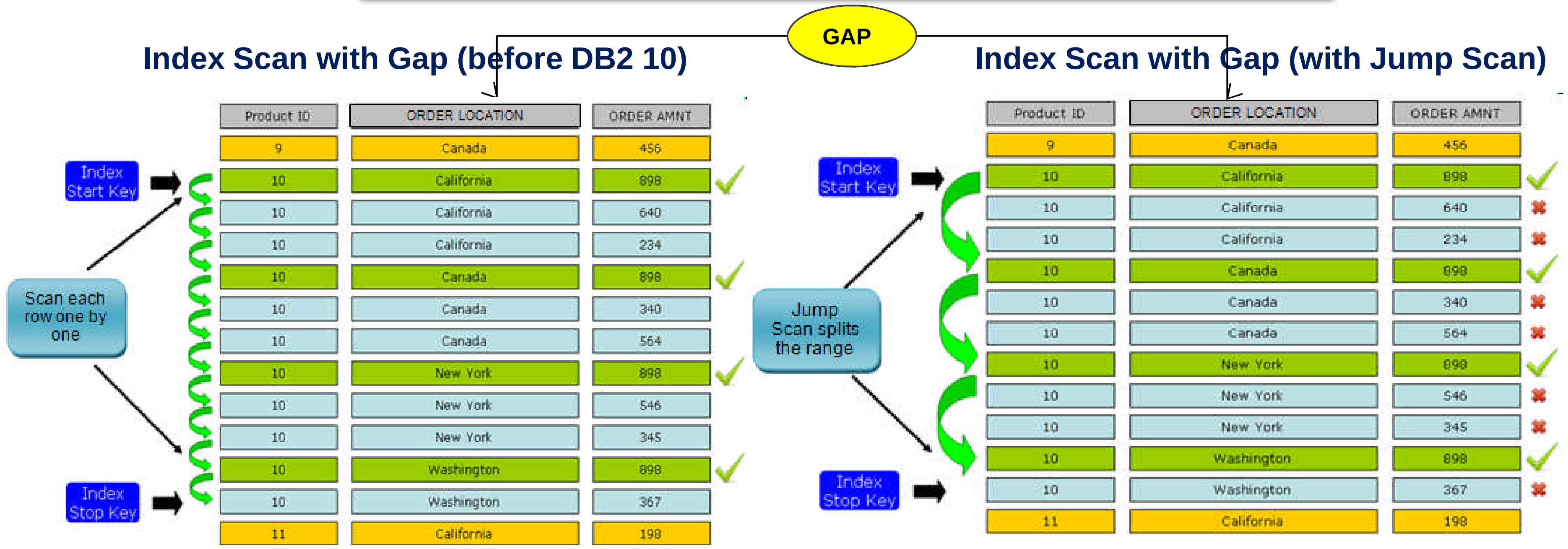
- **Why do we need Jump Scan?**
  - Difficult to find optimal set of indexes & optimize workloads with many ad-hoc queries
  - ad-hoc queries often have gaps in composite indexes
  - Jump Scan improves performance of Index Scans with gaps in the index
  - Avoids the need for additional indexes

- **When does Jump Scan work best?**
  - In data warehouse and SAP environments where query predicates leave gaps in the index, which slows down the index scanning process & leads to poor query performance
  - When the gap column has small cardinality (= number of distinct values that exist for the gap) and the non-gap column predicate is highly selective
  - Ideal case example
    - c1 has 10 distinct values
    - c2 contains a million distinct values, 10 of which satisfy the query predicate.

Jump Scan – How Does it Work?

Composite Index (product_id, order_location, order_amnt)

```
SELECT * FROM orders WHERE product_id = 10 AND order_amnt = 898
```



Index Scan with Gap (before DB2 10)

Product ID	ORDER LOCATION	ORDER AMNT	
9	Canada	456	
10	California	898	✓
10	California	640	
10	California	234	
10	Canada	898	✓
10	Canada	340	
10	Canada	564	
10	New York	898	✓
10	New York	546	
10	New York	345	
10	Washington	898	✓
10	Washington	367	
11	California	198	

Index Scan with Gap (with Jump Scan)

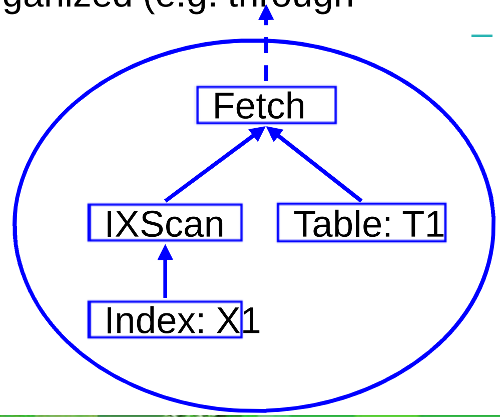
Product ID	ORDER LOCATION	ORDER AMNT	
9	Canada	456	
10	California	898	✓
10	California	640	✖
10	California	234	✖
10	Canada	898	✓
10	Canada	340	✖
10	Canada	564	✖
10	New York	898	✓
10	New York	546	✖
10	New York	345	✖
10	Washington	898	✓
10	Washington	367	✖
11	California	198	

- **DB2 scans a huge range of the index between start/stop keys while applying predicates to locate qualifying keys**
- **Queries against tables with composite (multi-column) indexes present a challenge when the query results in a gap (constrained or unconstrained)**
- **Index manager skips forward through the index while bypassing large sections that will not yield any results**
- **Processing involves two related scans**
  - Positioning: fills in the missing key parts
  - Consuming: locates matching keys

Smart Data and Smart Index Prefetching

(Data and Index page Sequential Detection + Read Ahead (SD+RA))

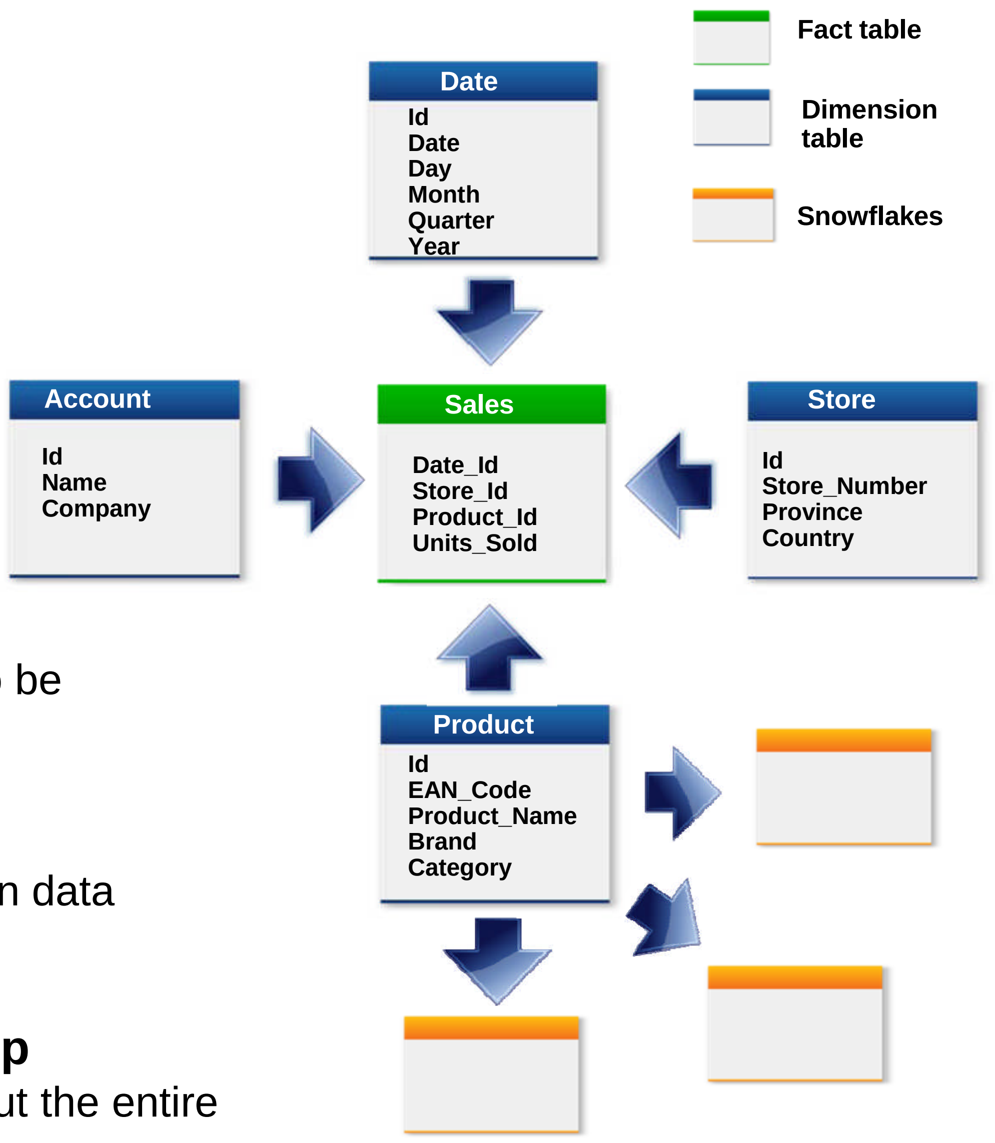
- **What is Smart Data/Index Prefetching?**
  - New Prefetching types in DB2 10 that switches between DB2's original Sequential Detection Prefetching (SD) to Read Ahead Prefetching when tables and indexes become unclustered
  - Uses the index to determine which index and data pages are accessed next (in contrast to SD, which guesses which pages are needed in the future)
- **When does it work best?**
  - When tables get very disorganized (e.g. through frequent IUD operations)



- **Why do we need Smart Data / Index Prefetching?**
  - Effective Pre-fetching is critical for MCP
  - Without pre-fetching, all subagents might wait while one performs I/O
  - Maximize data page prefetching and improve performance of IXSCAN and IXSCAN/FETCH for imperfectly clustered tables
  - Minimize need for table REORGs
  - Increase IXSCAN and IXSCAN/FETCH performance consistency independent from the degree of table and index organization

Star Schema Defined

- **What is Star Schema?**
  - Simplest form of a dimensional model
- **How the data is organized?**
  - Facts
  - Dimensions
- **A typical star schema based query**
  - Joins a subset of the dimensions with the fact tables
  - In a snowflake schema there may also be joins between dimension tables
- **Better performance**
  - Improves the performance of queries in data warehouse or data mart environments
- **Reduces the total cost of ownership**
  - Less complex tuning actions throughout the entire application life cycle

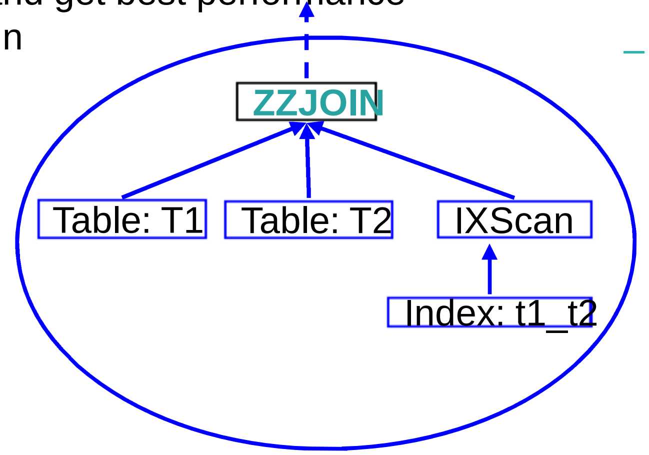


Star Schema Enhancements in InfoSphere Warehouse 10

- **New star schema detection method**
  - Allows the query optimizer to detect stars based on unique attributes
    - Primary keys, unique indexes, or unique constraints
- **New zigzag join method**
  - Provides consistent query performance in warehousing environments
- **Supports multiple fact table queries**
- **Exploits indexes even when there is a gap in probing key**
- **Optim Query Workload Tuner helps to determine optimal multi-columns indexes to enable the zigzag join**

Zigzag Join

- **What is zigzag join?**
  - A new join method for complex queries on dimensional schemas
  - Works for star schema queries in single or multiple subject areas with snowflakes
  - Works seamlessly with serial or DPF databases, range-partitioned and MDC tables
  - Simple index adviser integrated with db2exfmt and OQT to help with fact table index design and get best performance from zigzag join



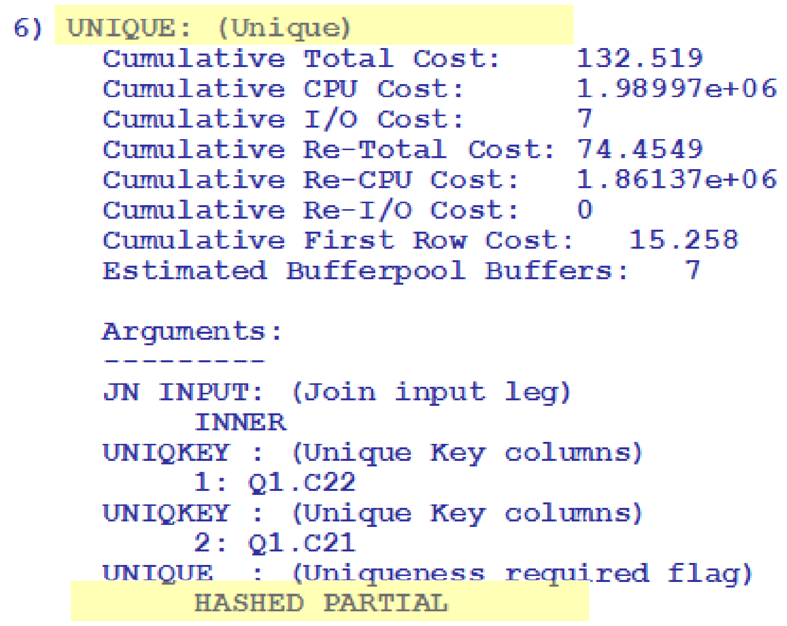
- **Why do we need zigzag join?**
  - Improved query performance
  - More stable performance that is less sensitive to small query or configuration changes
  - Easier logical database design
- **When can DB2 use zigzag join?**
  - Joins between fact table(s) and dimension tables on dimension unique keys
  - Fact table must have a multicolumn index that contains at least two of the join keys used in the query

Improved Performance of Queries

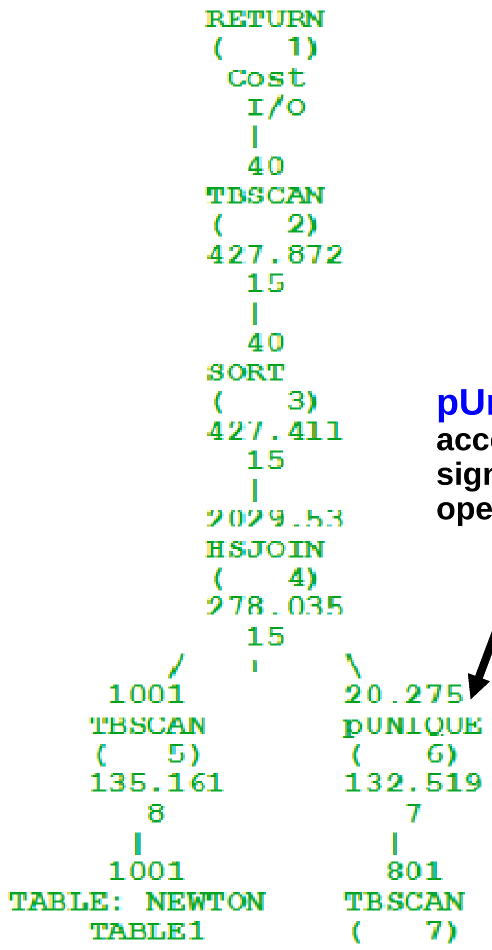
- View the new operators added as part of PED and PEA processing in EXPLAIN
- NEW values added to the EXPLAIN_ARGUMENT table
  - Indicate queries using the new hashing function
  - ARGUMENT_TYPE column = UNIQUE
  - ARGUMENT_VALUE column: HASHED_PARTIAL

```
SELECT DISTINCT c11, c12, c21, c22

FROM Table1, Table2

WHERE c11 = c21
```

db2exfmt output access plan details



```
6) UNIQUE: (Unique)
     Cumulative Total Cost:     132.519
     Cumulative CPU Cost:       1.98997e+06
     Cumulative I/O Cost:       7
     Cumulative Re-Total Cost:  74.4549
     Cumulative Re-CPU Cost:    1.86137e+06
     Cumulative Re-I/O Cost:    0
     Cumulative First Row Cost:   15.258
     Estimated Bufferpool Buffers:    7

     Arguments:
     ---------
     JN INPUT: (Join input leg)
          INNER
     UNIQKEY : (Unique Key columns)
          1: Q1.C22
     UNIQKEY : (Unique Key columns)
          2: Q1.C21
     UNIQUE  : (Uniqueness required flag)
          HASHED PARTIAL
```



```
                RETURN
                (   1)
                 Cost
                  I/O
                   |
                  40
                TBSCAN
                (   2)
                427.872
                  15
                   |
                  40
                SORT
                (   3)
                427.411
                  15
                   |
                2029.53
                HSJOIN
                (   4)
                278.035
                  15
              /    |     \
        1001             20.275
       TBSCAN            pUNIQUE
       (   5)            (   6)
       135.161           132.519
          8                 7
          |                 |
        1001               801
   TABLE: NEWTON         TBSCAN
       TABLE1            (   7)
```

pUnique in the access plan signifies the PED operation

RUNSTATS Enhancements

- **Index sampling**
  - In DB2 9.7 only data pages can be sampled
  - Global indexes on range-partitioned tables can be very large
  - All partitions of a partitioned index are read in DB2 9.7

- **Performance improvements**
  - Path length reduction
    - RUNSTATS is very CPU-intensive
  - Use new index readahead prefetching capability
    - More efficient I/O when sequential detect prefetching isn't possible

- **Usability improvements**
  - Specifying table or index schema will be optional
  - Support VIEW keyword
  - Make SAMPLED the default when DETAILED is specified
    - SAMPLED DETAILED is currently the recommended index option
    - Similar changes to CREATE INDEX … COLLECT STATISTICS clause

RUNSTATS Index Sampling

```
>>-RUNSTATS ON--+-TABLE-+--object-name-------------------------->
                '-VIEW--'
....

>--+-------------------------+--+-----------------------------+--->
   '-| Table Sampling Options |-' '-| Index Sampling Options |-'

Table Sampling Options
|--TABLESAMPLE--+-BERNOULLI-+--(--numeric-literal--)------------>
                '-SYSTEM----'
>--+------------------------------------+-----------------------|
   '-REPEATABLE--(--integer-literal--)-'

Index Sampling Options
|--INDEXSAMPLE--+-BERNOULLI-+--(--numeric-literal--)------------|
                '-SYSTEM----'
```