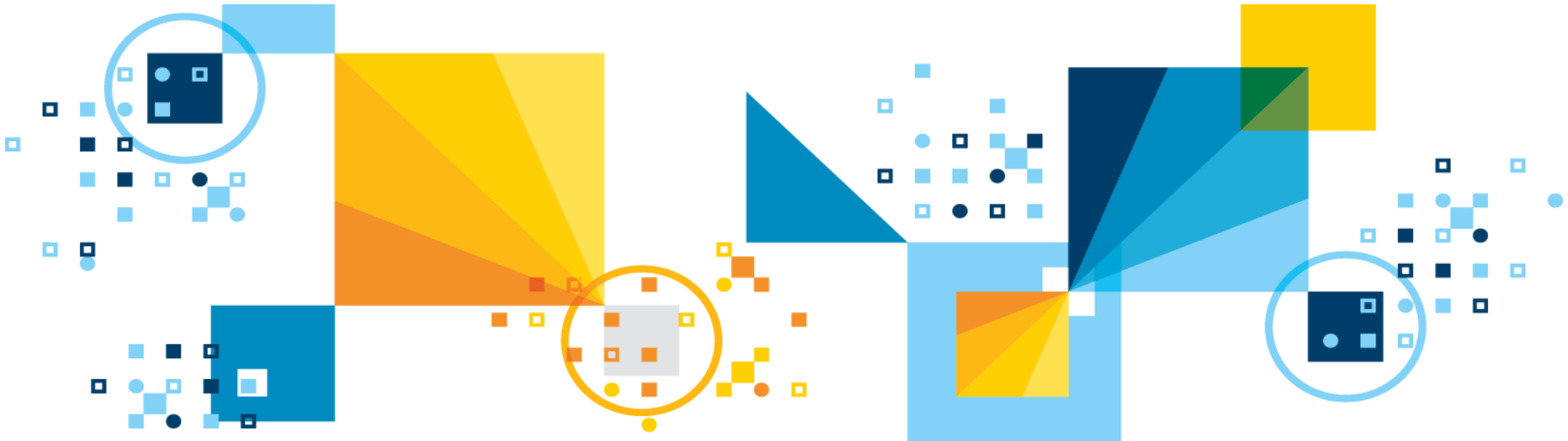


DB2 Enhanced Analytics

Module ID | 10114

Length | 1.5 hours



Disclaimer

© Copyright IBM Corporation 2015. All rights reserved.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

IBM, the IBM logo, ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

Module Information

- You should have completed or acquired the necessary knowledge for the following modules in order to complete this module:
 - DB2 Fundamentals

- After completing this module, you should be able to:
 - Explain the concepts of:
 - Multi-core parallelism
 - Faster and Reliable Query Performance
 - Enhancements for Faster Index Access
 - Optimization for Star Schema based queries

Enhancing Analytics with DB2 10

- Traditional Data Warehouses deal with large amounts of data needed for Business Analytics
- DB2 10 provides the mechanism to increase business insight and achieve high degree of data availability
 - **Enhanced out-of-the-box query performance**
 - Leverage multi-core parallelism benefits while lowering purchase and support costs
 - Easier to maintain performance for different workloads while changes occur in the environment
 - **Integration with Big Data (Hadoop) frameworks**
 - Java generic table functions facilitates access to Big Data frameworks through traditional applications and data warehouses
 - **Continuous data ingestion while maintaining availability**
 - New client-side tool for high speed, continuous data ingest

Enhanced Business Resiliency and Faster Query Performance

- **Multi-core parallelism improvements**
 - Enhanced and more flexible parallelization capabilities
 - Rebalancing imbalanced subagent workloads
 - Parallel scans on range partitioned indexes
- **Faster and Reliable Query Performance**
- **Enhancements for Faster Index Access**
- **Optimization for Star Schema based queries**

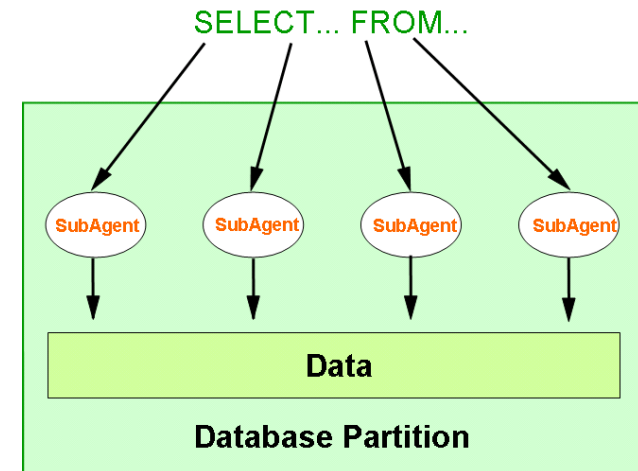


Greater Flexibility in Controlling Multi-Core Parallelism

- Data Warehouse workloads benefit greatly from parallelization
- Shrinking 'Batch Windows' on OLTP and Mixed workloads require machines to do more work in less time
- **Prior to DB2 10:**
 - **Degree** of parallelism used was controlled for the whole instance
 - INTRA_PARALLEL dbm configuration parameter (YES|NO)
 - Default is NO
 - Mixed workloads had no flexible approach to control parallelism
- **DB2 10 introduces greater flexibility in controlling degree of parallelism**
 - Enable/disable the degree of parallelism at the application level
 - For mixed workloads, provide parallelism settings that are optimal for the kind of workload

Multi-Core Parallelism Defined

- Multi-core parallelism is a significant enhancement to query parallelism
 - For a single partition database or per-partition of a partitioned database
 - Also called intra-partition parallelism or SMP parallelism
- Why leverage Multi-core parallelism?
 - New hardware is constantly providing more cores
 - With new enhancements, the complexity of DPF may be avoided for smaller warehouses
 - In a DPF environment, exploiting multi-core parallelism is easier than increasing the number of partitions
- SELECT queries benefit from increased parallelism



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 1
```

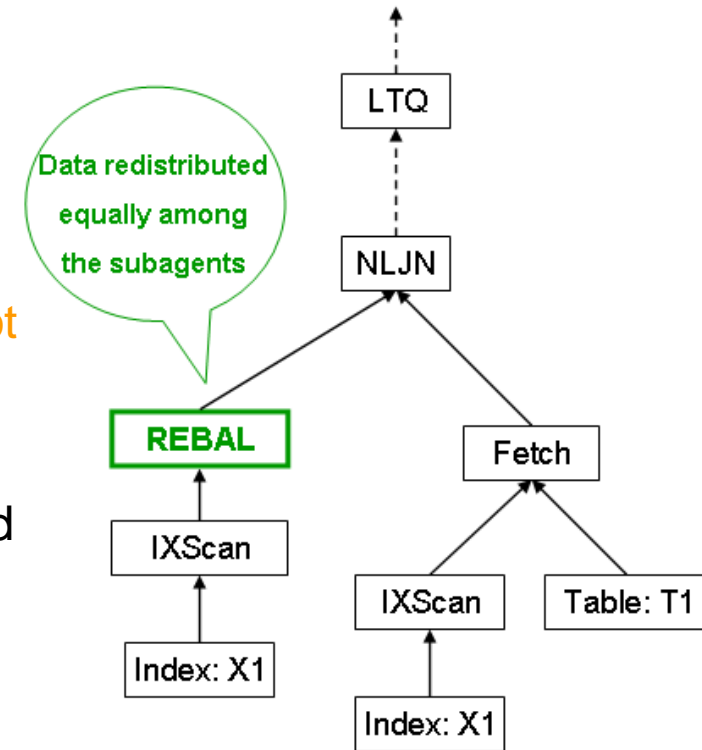
Only applies
to specified
workload

- Takes effect at transaction boundary; NO instance or database recycle
- Better built-in runtime decision of parallelism degree when **ANY** is specified
 - let DB2 decide the level of parallelism at runtime
- Determine degree of parallelism based on:
 - Number of processors, characteristics of query, amount of activity on the system
- **ANY** can be specified in several ways:
 - Use the statement: **SET CURRENT DEGREE = 'ANY'**
 - Use the **DEGREE ANY** option of the BIND command
 - Set the database configuration parameter **DFT_DEGREE** to the value **ANY**

Parallelism and Workload Imbalance

- Data filtering and data skew can cause subagent workloads to become imbalanced
 - Few do 'meaningful' work, others wait at the next blocking operator
 - Inefficiency is magnified by joins and other expensive operations
 - A query plan that exhibits these properties will not scale
- **Rebalance (REBAL)** operator:
 - New access plan operator specific to parallelized environments
 - A mechanism for transferring rows between subagents to rebalance their workload
 - Lighter weight mechanism for rebalance
 - This operator could put "idle" subagents to work

Rebalanced data stream

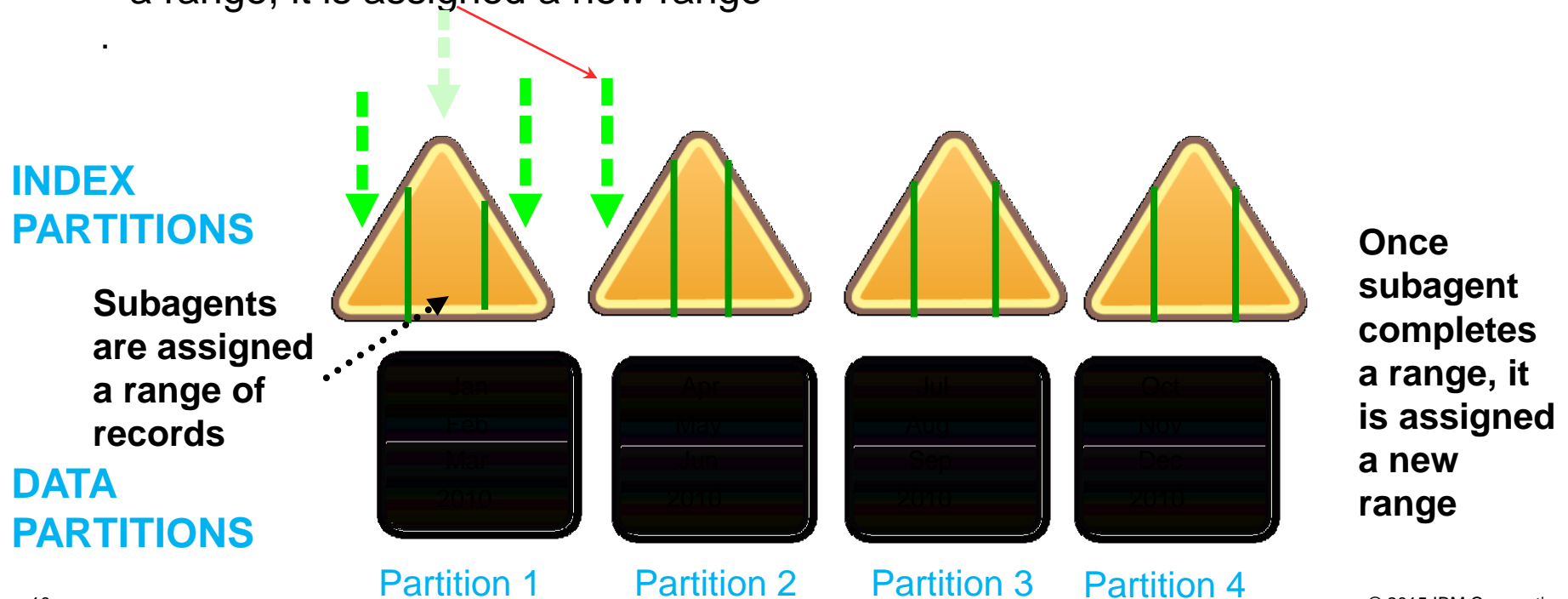


Parallel Scanning of Partitioned Indexes Improves Performance


- Parallel scans provide an even distribution of work among subagents

Balanced workload among subagents → **efficient use of CPU resources**

- Parallel scans can now be run against partitioned indexes
 - Partitioned indexes are divided into ranges of records
 - Subagents are assigned a range of records, and once the subagent completes a range, it is assigned a new range



Enhanced Business Resiliency and Faster Query Performance

- **Multi-core parallelism improvements**
- **Faster and Reliable Query Performance** 
 - Queries with SELECT DISTINCT, Aggregate clauses
 - Fine tune individual SQL statements
 - Improvements to RUNSTATS
 - Statistical views
- **Enhancements for Faster Index Access**
- **Star Schema Optimization**



Faster and Reliable SQL Queries

- Issues with SQL query performance through application life cycle:
 - Optimizing queries may require complex configuration and tuning
 - Queries may run slower after maintenance
 - DB configuration changes can adversely affect certain queries
 - RUNSTATS may not complete within the available maintenance window
- **DB2 10 has a solution!**
 - Reduces total cost of ownership by eliminating complex tuning actions
 - Optimal query performance 'out-of-the-box', with minor tuning.
 - Consistent query performance, with minimal administration effort
- Focused on improving the performance of typical warehouse queries
 - **Improved** logic for hash join, aggregation, sort
 - **Improved** cost estimations for queries
 - RUNSTATs **performance improvements**
 - **Better usage** of statistical views

Improved Performance of Queries

- **DB2 10** introduces several changes to improve performances of SQL queries
- Two **new** concepts to improve query performance are:
 - **Partial Early Distinct (PED)**
 - Remove most duplicate rows early in the processing
 - **Partial Early Aggregation (PEA)**
 - Aggregate records earlier in the processing than a standard GROUP BY could be applied
- **Selection of hash joins** has been **enhanced** for a wider range of SQL queries:
 - Queries with data type mismatches
 - Expressions used in join predicate
- **Benefits:**
 - Reduces data to be processed prior to the sort or join
 - Speeds up the query without affecting final result set
 - Reduces possibility of running out of sort heap memory
 - Does not require any settings or changes to SQL

Fine Tune Individual SQL Statements for Better Performance

- **Optimization Profiles** are used to provide explicit optimization guidelines to the DB2 Optimizer
 - Created as an XML document containing guidelines for SQL statements
- Setting different registry variables in optimization profiles can increase flexibility
- **DB2 10** introduces **statement level** registry variable **settings**
 - Registry variables can be set at both global and statement level
 - Profiles can tailor variables to specific statements

Targeted SQL statement → `<![CDATA[select t1.c1, count(*) from t1,t2 where t1.c1 = t2.c1 group by t1.c1]]>`
Registry variable setting → `<REGISTRY>`
ID='S1' signifies statement to which these guidelines are to be applied → `<STMTPROFILE ID='S1'>`

```

<STMTPROFILE ID='S1'>
  <STMTKEY>
    <![CDATA[select t1.c1, count(*) from t1,t2 where t1.c1 = t2.c1 group by t1.c1]]>
  </STMTKEY>
  <OPTGUIDELINES>
    <!-- New: registry variables that JUST applies to the above
         SQL statement when using this profile -->
    <REGISTRY>
      <OPTION NAME='DB2_REDUCED_OPTIMIZATION' VALUE='NO_SORT_NLJOIN'/>
    </REGISTRY>
    <NLJOIN>
      <TBSCAN TABLE='T1'/>
      <TBSCAN TABLE='T2'/>
    </NLJOIN>
  </OPTGUIDELINES>
</STMTPROFILE>
  
```

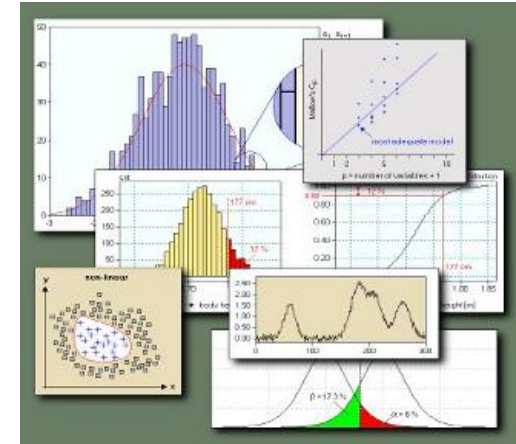
Enhanced Business Resiliency and Faster Query Performance

- **Multi-core parallelism improvements**
- **Faster and Reliable Query Performance**
- **Enhancements for Faster Index Access**
 - Improved performance for queries with composite indexes
 - More efficient index and data prefetching
- **Optimization for Star Schema based queries**



New Index /Data Access Techniques Improve Performance

New Technique	Objective
Jump Scan	<ul style="list-style-type: none"> Improve performance of queries with index gaps Reduced number of indexes to maintain
Smart Index prefetching	<ul style="list-style-type: none"> Improve reliability of index scan performance Reduced need for index reorgs
Smart Data prefetching	<ul style="list-style-type: none"> Improve reliability of index scan for + data fetch performance for poorly clustered tables Reduced need for table reorgs
Predicate evaluation avoidance for duplicate keys	<ul style="list-style-type: none"> Improve performance of index scans with duplicate keys



DB2 10 provides
higher performance
&
lower costs of
maintaining indexes

Improved Performance for Queries with Composite Indexes

- For workloads with many ad-hoc queries, it is often difficult to optimize a database for high performance.
 - Index gaps are common in queries against tables with composite indexes

**Composite Index on
columns A,B,C**

```
SELECT * FROM t WHERE a=5 AND c=10
```

query contains an index gap on
column B in the composite index

- **DB2 10** intends to
 - improve the performance of queries involving index gaps
 - Eliminate need for additional indexes using Jump Scan
- **Jump scans** provide a solution to the index gap problem
 - Ad-hoc queries may utilize Jump Scan without additional index definitions if general-purpose composite indexes are available

Jump Scan Enhancement Demonstration

```
CREATE INDEX empIndex ON employee
( division, department, jobTitle )
```

```
SELECT * FROM employee WHERE
division='Software' AND
jobTitle='Senior Software Engineer'
```

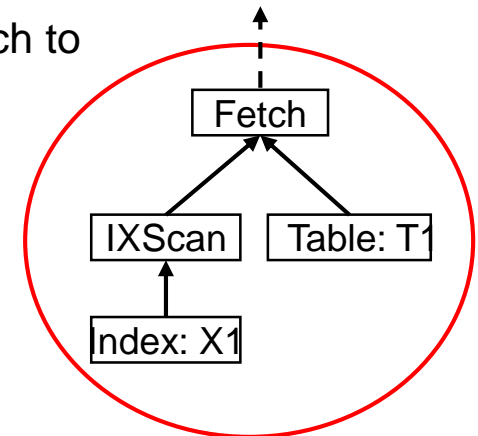
Processing involves two related scans:

- Position:** fills in the missing key parts
Within **Division**='Software',
Jump from
Department = "Development" -> "Research"
-> "Test"
- Consume** to locate matching keys
Within a **Department**
find **JobTitle**='Senior Software Engineer'

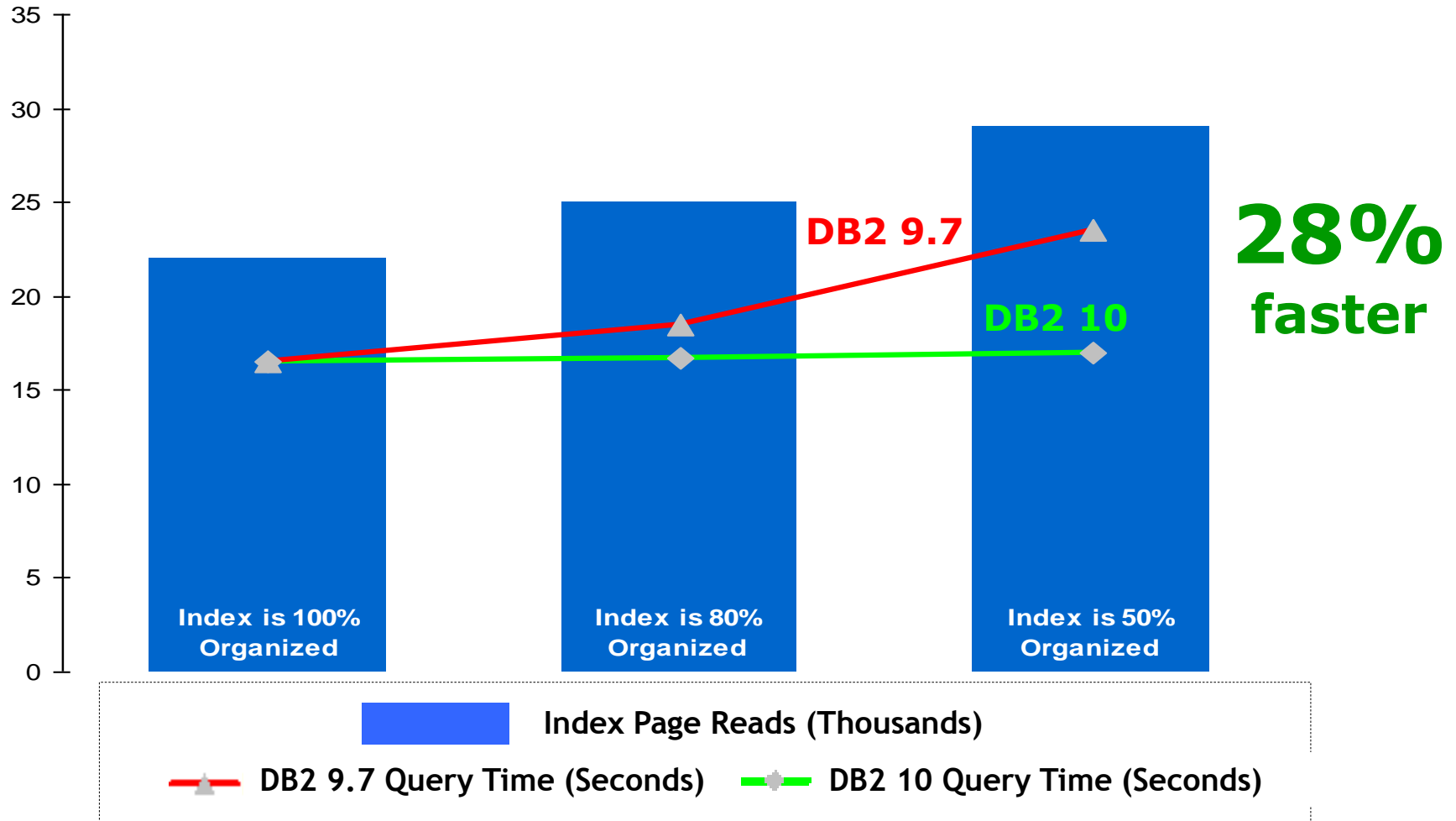
Division	Department	JobTitle
Sales	Customer Service	Sales Associate
Software	Development	Architect
Software	Development	Intern
Software	Development	Manager
Software	Development	Senior Software Engineer
Software	Development	Staff Software Engineer
Software	Research	Co-op
Software	Research	Senior Software Engineer
Software	Research	Senior Technician
Software	Test	Intern
Software	Test	Manager
Software	Test	Test Engineer
Software	Test	Test Engineer
Systems	Software	Requirements Engineer

Smart Index/Data Prefetching

- **Pre-DB2 10**, when data is loaded into or updated in an indexed table, index leaf pages become non-sequential
 - Poorly organized indexes reduce asynchronous I/O and increase synchronous I/O
 - Performance for such queries is sub-optimal
 - An index reorg is an expensive operation to fix this
- **In DB2 10**, **Smart Index Prefetch** minimizes negative I/O performance impact
 - Introduces **readahead prefetching** for Index Scans
 - Lets the Optimizer select the optimal prefetching method
 - When Sequential detection is not sufficient, dynamically switch to readahead pre-fetch during query execution
- **Smart Data Prefetch**
 - Similar approach to smart index prefetching
 - For data pages pointed to by the index
 - Used with Index Scan + Data Fetch [**ISCAN/FETCH**]



Smart Index Pre-Fetching Means Easier Management



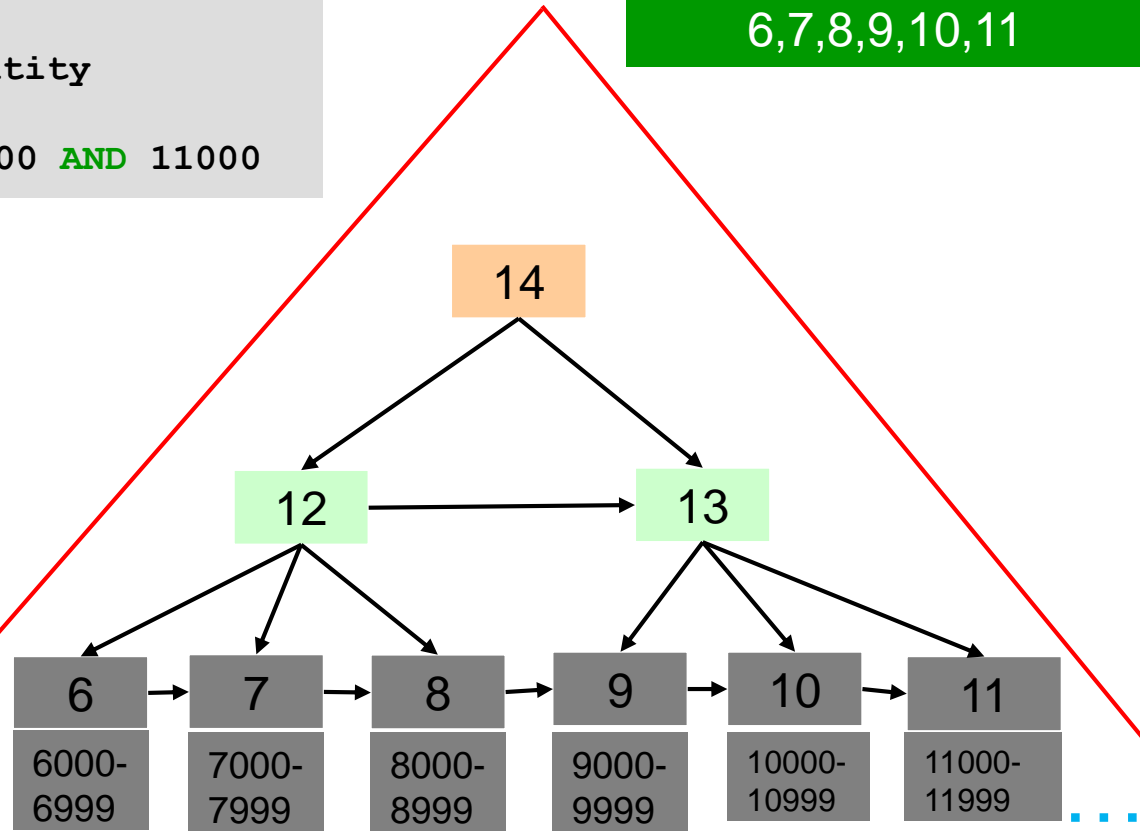
Results based on IBM internal testing.

Pre-DB2 10 Index Prefetching Against an Organized Index

```
CREATE INDEX order_ix ON TABLE order  
(order_id) INCLUDE (order_quantity)
```

```
SELECT order_id, order_quantity  
FROM order  
WHERE order_id BETWEEN 1000 AND 11000
```

Sequential detection
Pre-fetches leaf pages
6,7,8,9,10,11



Index Page #

order_id range

Smart index prefetching against disorganized index

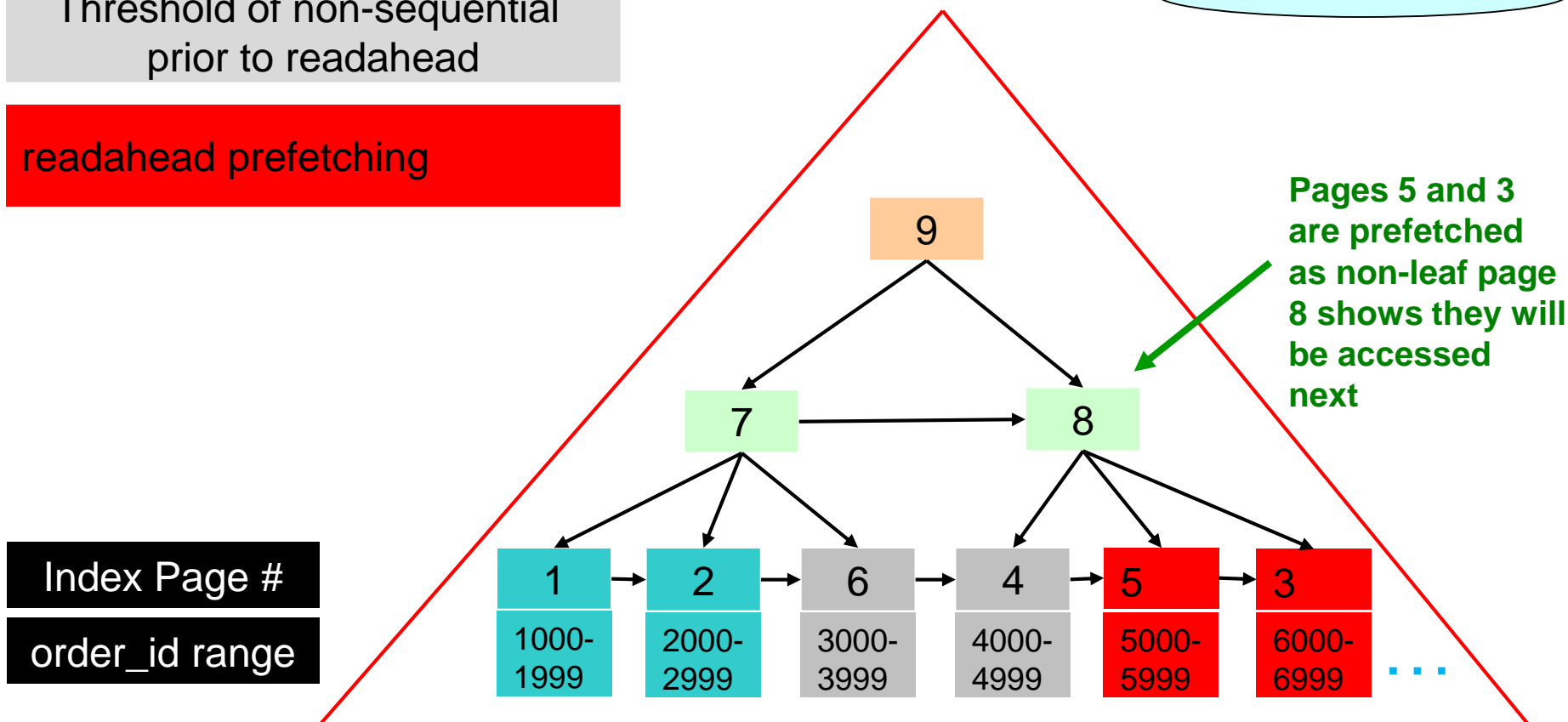
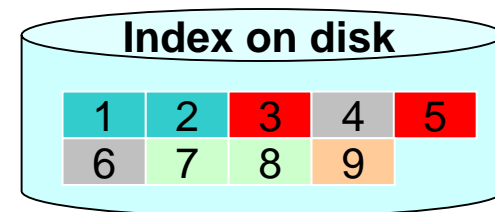
Combines sequential detection with readahead prefetching

Index scan needs leaf pages 1,2,6,4,5,3

Sequential detection prefetching

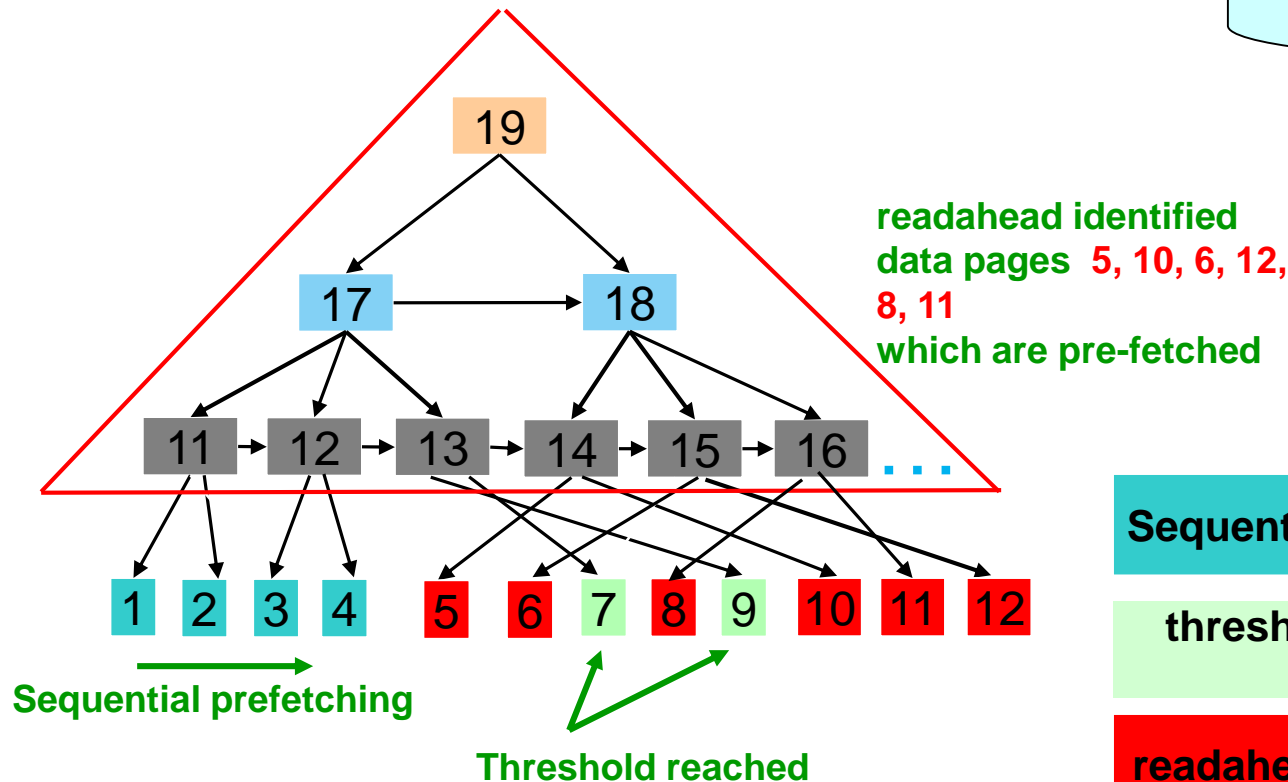
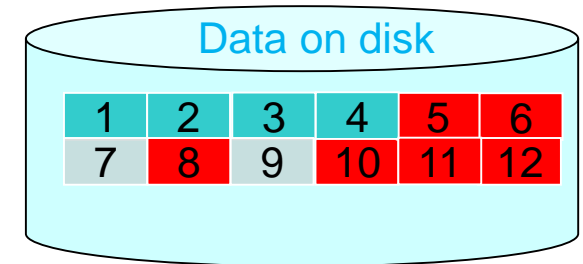
Threshold of non-sequential
prior to readahead

readahead prefetching



Smart Data prefetching

- In this example, index is organized but data is not clustered
- Index scan needs leaf pages in order 11, 12, 13, 14, 15, 16
- Data fetch needs data pages pointed to by these leaf pages in order 1,2,3,4,9,7,5,10,6,12,8,11



Sequential detection prefetching

threshold of non pre-fetched data pages

readahead prefetching

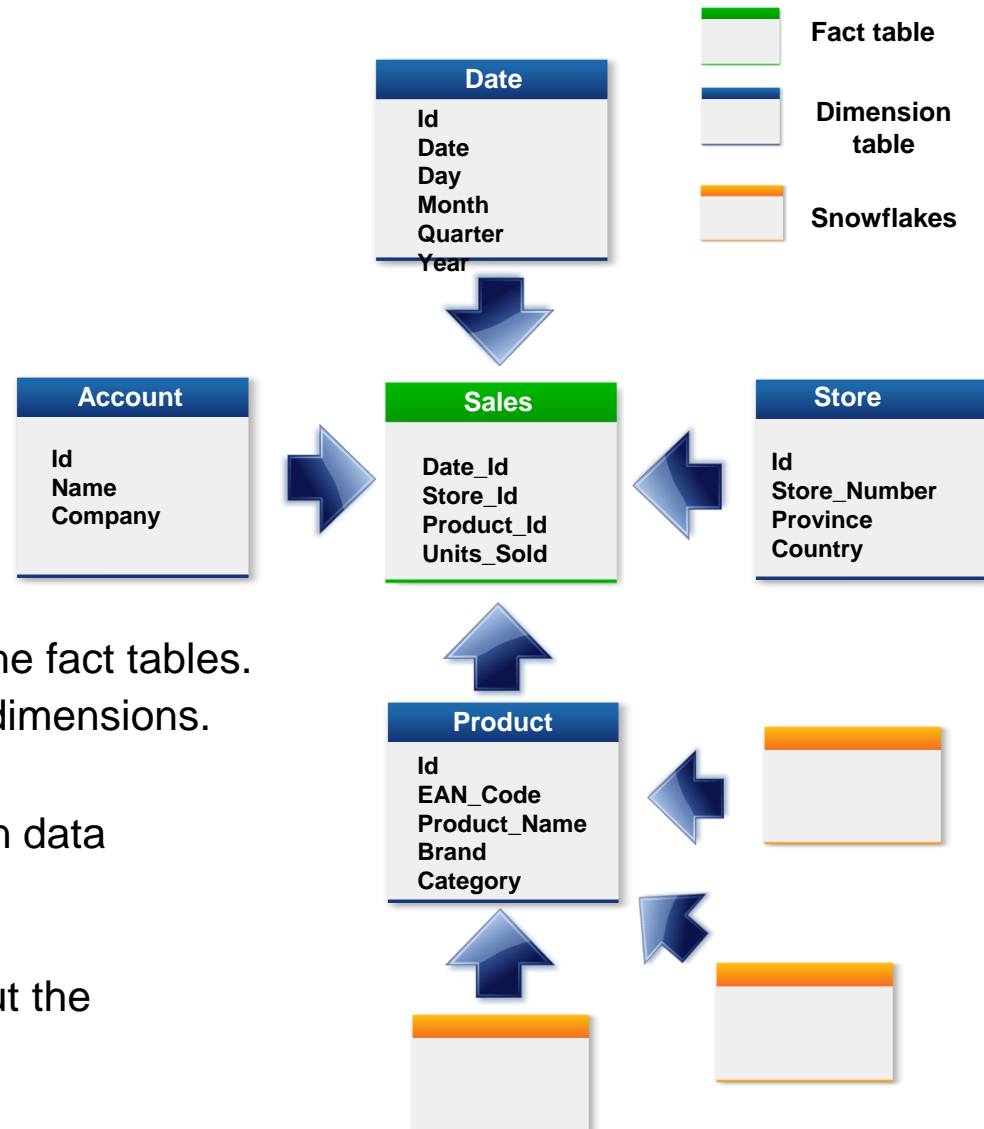
Enhanced Business Resiliency and Faster Query Performance

- **Multi-core parallelism improvements**
- **Faster and Reliable Query Performance**
- **Enhancements for Faster Index Access**
- **Optimization for Star Schema based queries**
 - new star schema detection algorithm
 - New zigzag method



Star Schema Defined

- **What is Star Schema?**
 - Simplest form of a dimensional model
- **How the data is organized?**
 - Facts
 - Dimensions
- **Optimized for**
 - Data warehouses
 - Data mart
- **A typical star schema based query:**
 - joins a subset of the dimensions with the fact tables.
 - Usually, there are no joins among the dimensions.
- **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 DB2 10

- **New star schema detection method**
 - Query optimizer can detect stars based on unique attributes
 - primary keys, unique indexes, or unique constraints
- **New zigzag join method**
 - Works seamlessly for range partitioned tables and serial, SMP and DPF, and pureScale
 - Provides consistent query performance in warehousing environments
- **Supports queries joining multiple fact table queries, snowflakes**
- **Exploits indexes even when there is a *gap in probing key***
- **Db2exfmt has been enhanced** to provide recommendation to enable zig-zag join
 - The following diagnostic message is returned in the db2exfmt diagnostic sections

Extended Diagnostic Information:

Diagnostic Identifier: 1

Diagnostic Details: EXP0256I Analysis of the query shows that the query might execute faster if an additional index was created. Schema name: "STAR". Table name: "FACT". Column list: "(F3, F2, F1, F0)".

- **Optim Query Tuner** will provide a workload based index advisor which uses the above feature to reach a consolidated set of index recommendations for a multi-column index

A Star Schema Based Query

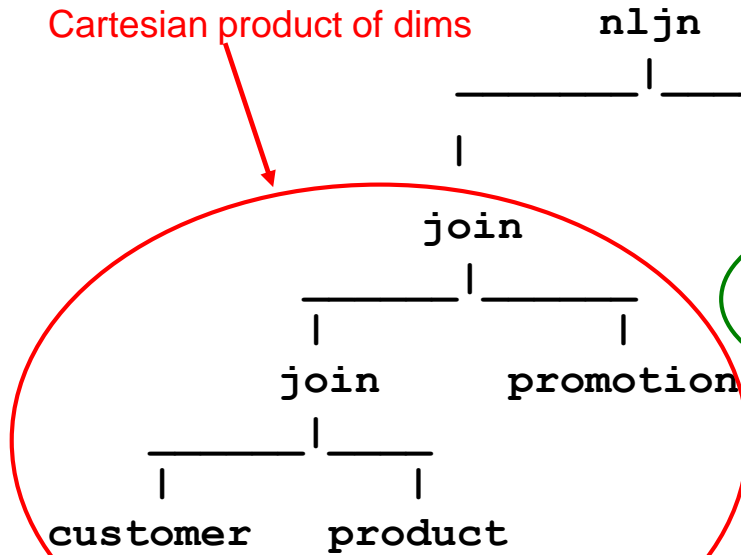
With Too Many Unproductive probes into the fact table

▪ Example

```
select count(quantity_sold)
from   daily_sales s, product prod, customer c, promotion promo
where  prod.category = 42 and promotype = 2 and
       c.age_level = 4 and c.income_level = 5 and
       prod.prodkey = s.prodkey and c.custkey = s.custkey and
       promo.promokey = s.promokey;
```

▪ Cartesian Hub Plan:

Cartesian product of dims



Multi-column index:
(custkey, prodkey, promokey)
on the fact table.

Only 1400 of them are
represented in the fact table



**320 million combinations
(and probes into the fact table)**

The New Zigzag Join Method for Star Schema Based Queries

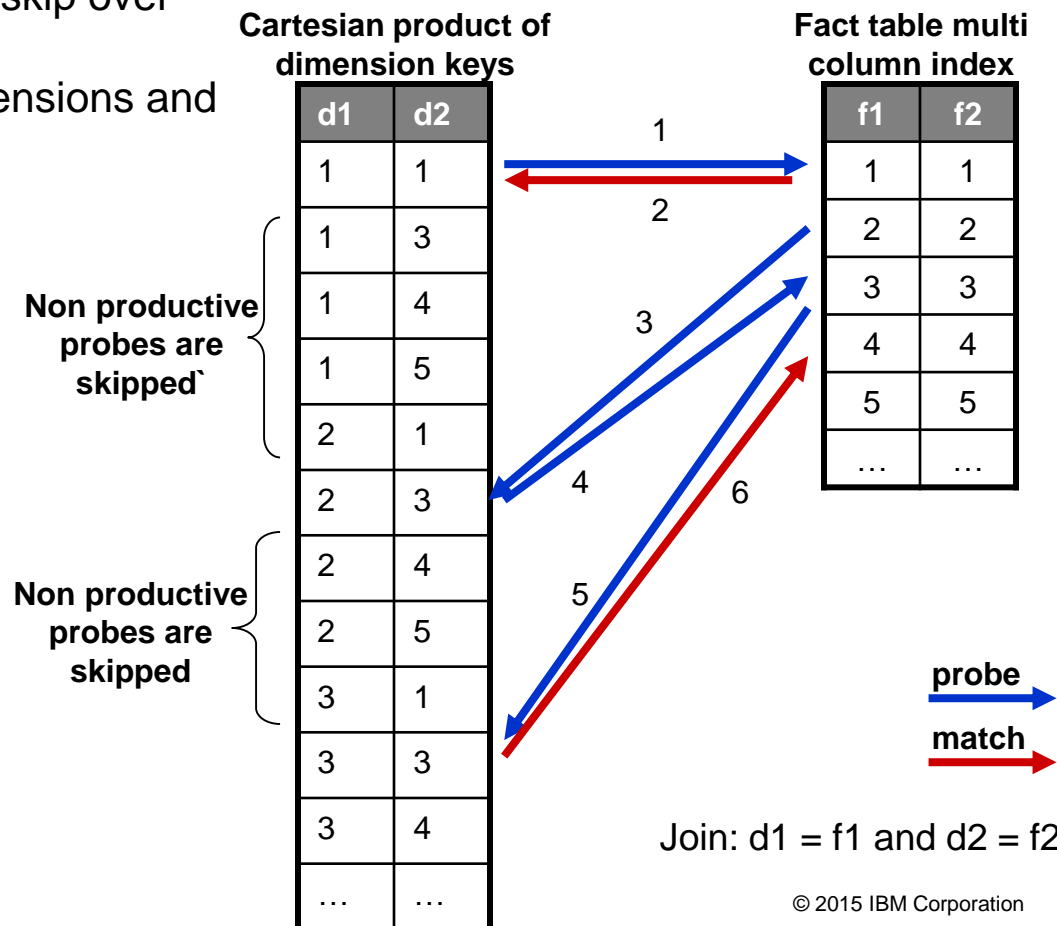
How does it work?

- Form the conceptual Cartesian product of dimensions keys.
- Use dimension keys to probe into the fact table to locate matching keys
- Use feedback from the fact table to skip over non-productive probes
- Continue to zigzag through the dimensions and fact tables.

- **Pre-requisite:** A multi-column index on the fact table on columns that join with the dimensions.

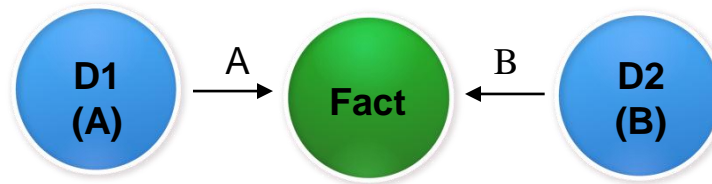
Dimension keys

d1	d2
1	1
2	3
3	4
4	5
...	...



Jump Scan in Probing Key of the Fact Table Index

- There is no need to define a separate index for each star schema based query with a different set of dimensions. This is made possible with the gap technology in DB2 10.
- For example, a fact table index on (A, C, B) can enable zigzag join for the following star schema based query where there is no join on column C.



- To confirm whether a fact table is chosen for the zigzag join with a gap, check the fact table IXSCAN operator detail in db2exfmt output:

Arguments:

JUMPSCAN: TRUE (JumpScan Plan)

Gap Info:

Status

Index Column 0:

No Gap

Index Column 1:

Positioning Gap

Index Column 2:

No Gap

Performance Results

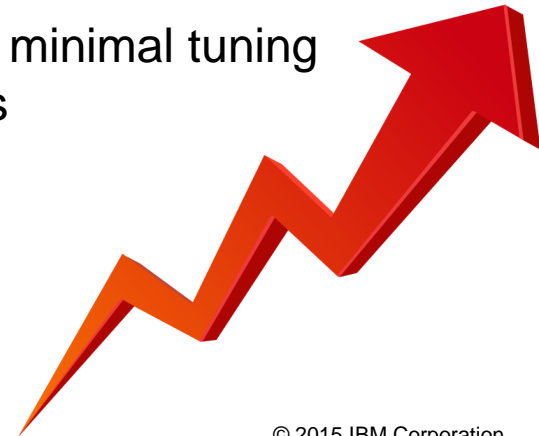
- **Effect of enabling zigzag join on the star schema workload with 29 queries in a 1TB database**

Total elapsed time: Zigzag disabled	Total elapsed time: Zigzag enabled	Total elapsed time speedup
7147 seconds	667 seconds	10x

Single user, "logical" DPF with 32 partitions, P7 AIX server with 32 processors

Overall Enhancements in Query Performance

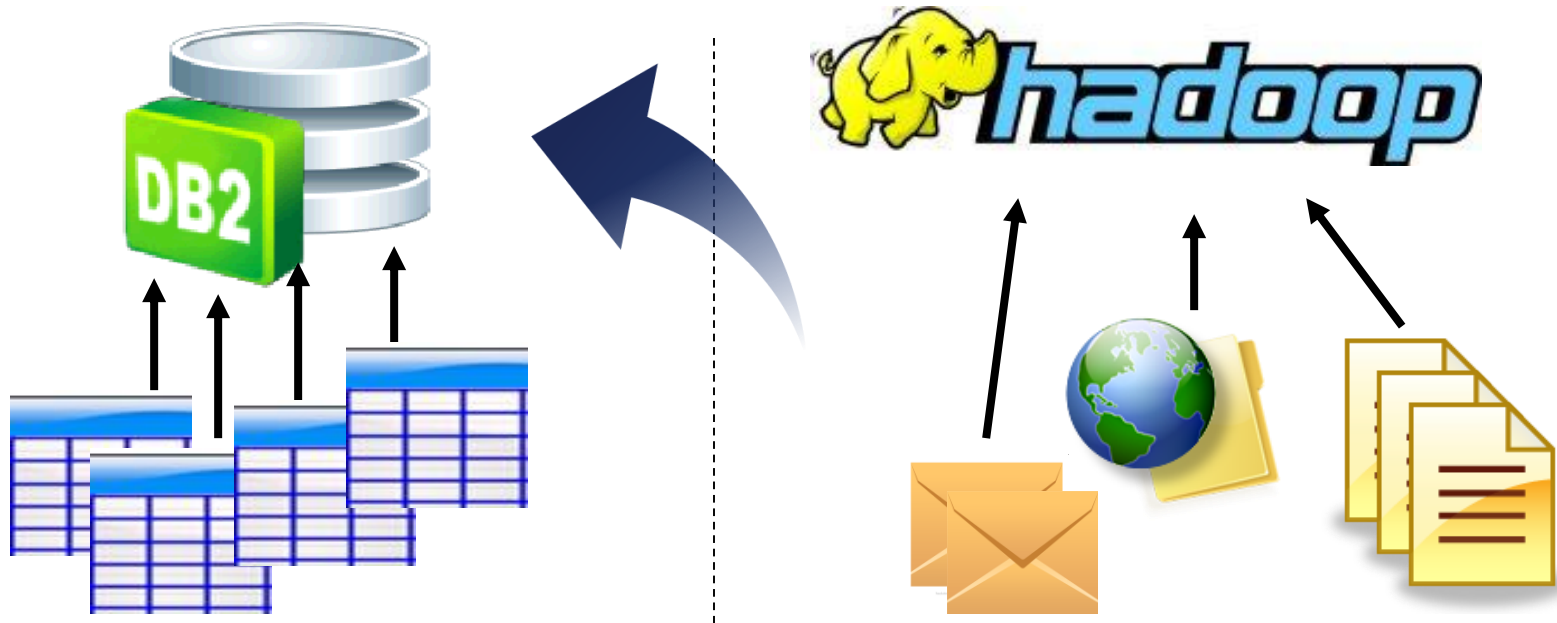
- Internal optimization improvements
 - Improve access path quality and runtime for key operations
 - Focus on complex queries typical of data warehouses
 - Drive greater performance “out of the box” by improving
 - Accuracy of cardinality estimates
 - Memory management
 - Hash join
 - Aggregations and sorts
 - Star schema optimization
 - More...
- Expected benefits
 - Improved runtime performance for target workloads with minimal tuning
 - Reduced cost of ownership due to reduced tuning efforts



Enhancing Analytics With Big Data

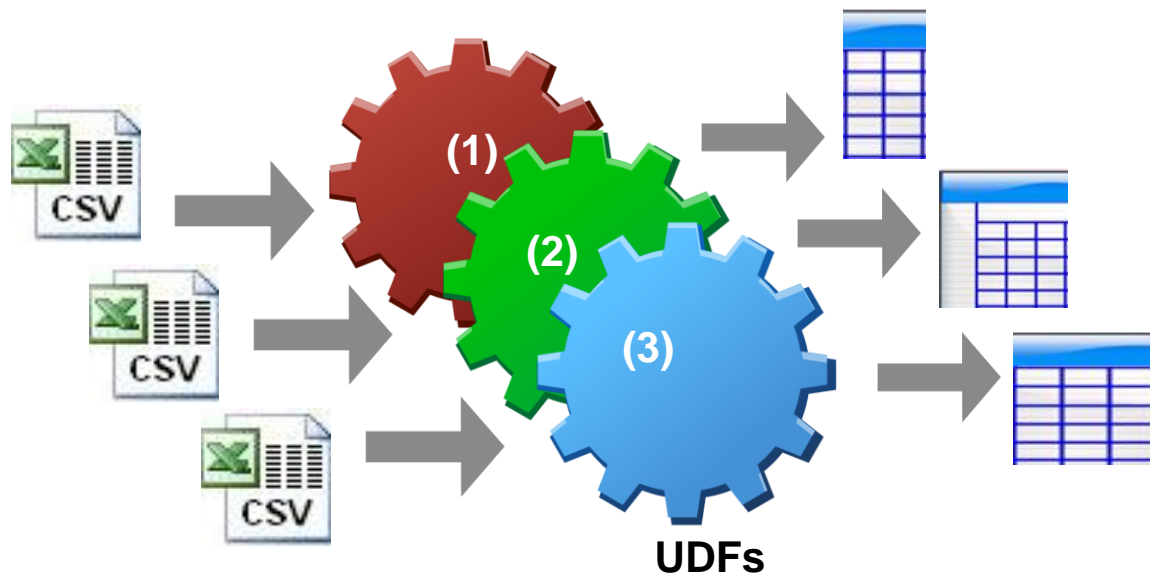
- **Quantity** and **types** of data being captured for business analysis is growing.
 - New framework are now available to process large amounts of complex data.
- How to efficiently integrate traditional BI systems with these new frameworks?

Use of
unstructured
raw data is
GROWING



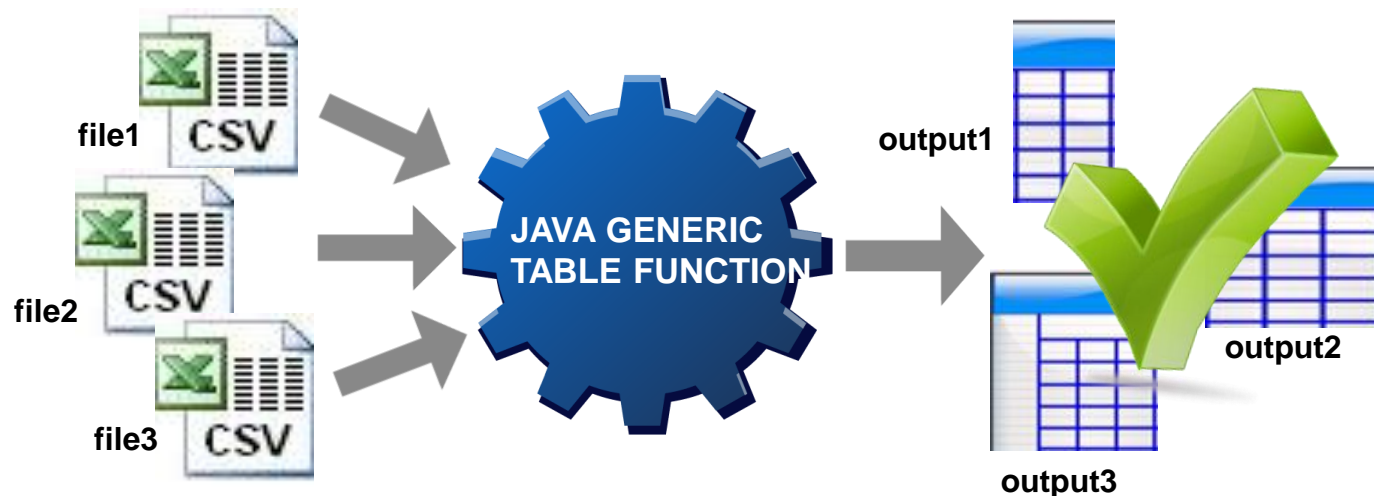
Enhancing Analytics On Big Data

- **Solution: Use Hadoop results in an SQL statement via embedded query in UDF**
 - UDFs implement analytic jobs submitted from DB2 or from SQL-based analytic applications
 - Changing business scenarios and various formats of data requires multiple UDFs to support different result sets
- Using Java Generic table functions simplifies the analytic process!



Java Generic Table Function

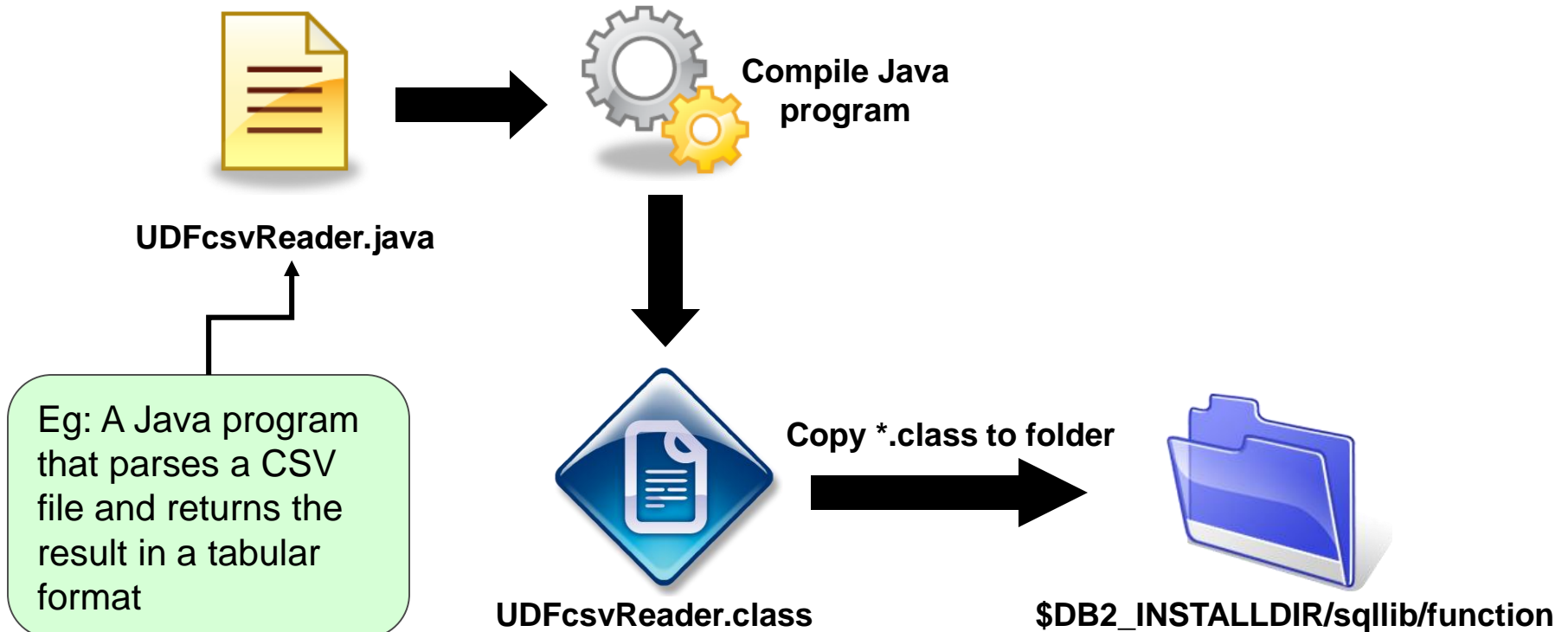
- An external user defined table function written in the Java programming language.
 - EXTERNAL: implementation is provided by a program outside of DB2
 - TABLE FUNCTION: a function that returns a result set (i.e. set of rows)
- Why 'Generic' ?
 - It supports varied output formats which are declared at runtime.
 - Output parameters not specified in CREATE FUNCTION DDL statement
- Output from function is accessed by SQL SELECT statement
 - Return different result set formats by changing SELECT statements



How to Create a Java Generic Table Function

Step 1: Create the external Java routine

- Code the Java routine logic
- Compile the program to produce a Java class file
- Copy the class file to the DB2 database server



How to Create a Java Generic Table Function

Step 2: Register the external Java routine with DB2

statement used to create table UDFs in DB2

clause specifies the language interface used to write the function body

```
CREATE FUNCTION csvRead(fnme VARCHAR(255))  
RETURNS GENERIC TABLE  
EXTERNAL NAME  
'UDFcsvReader.jar:UDFcsvReader!csvReadString'  
LANGUAGE JAVA  
PARAMETER STYLE DB2GENERAL;
```

specifies the external code invoked by DB manager to run the UDF

signifies the output of the table function is determined at runtime

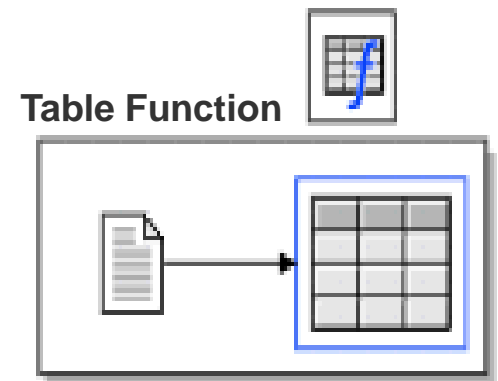
convention used to specify passing parameters /returning values

Using a Java Generic Table Function

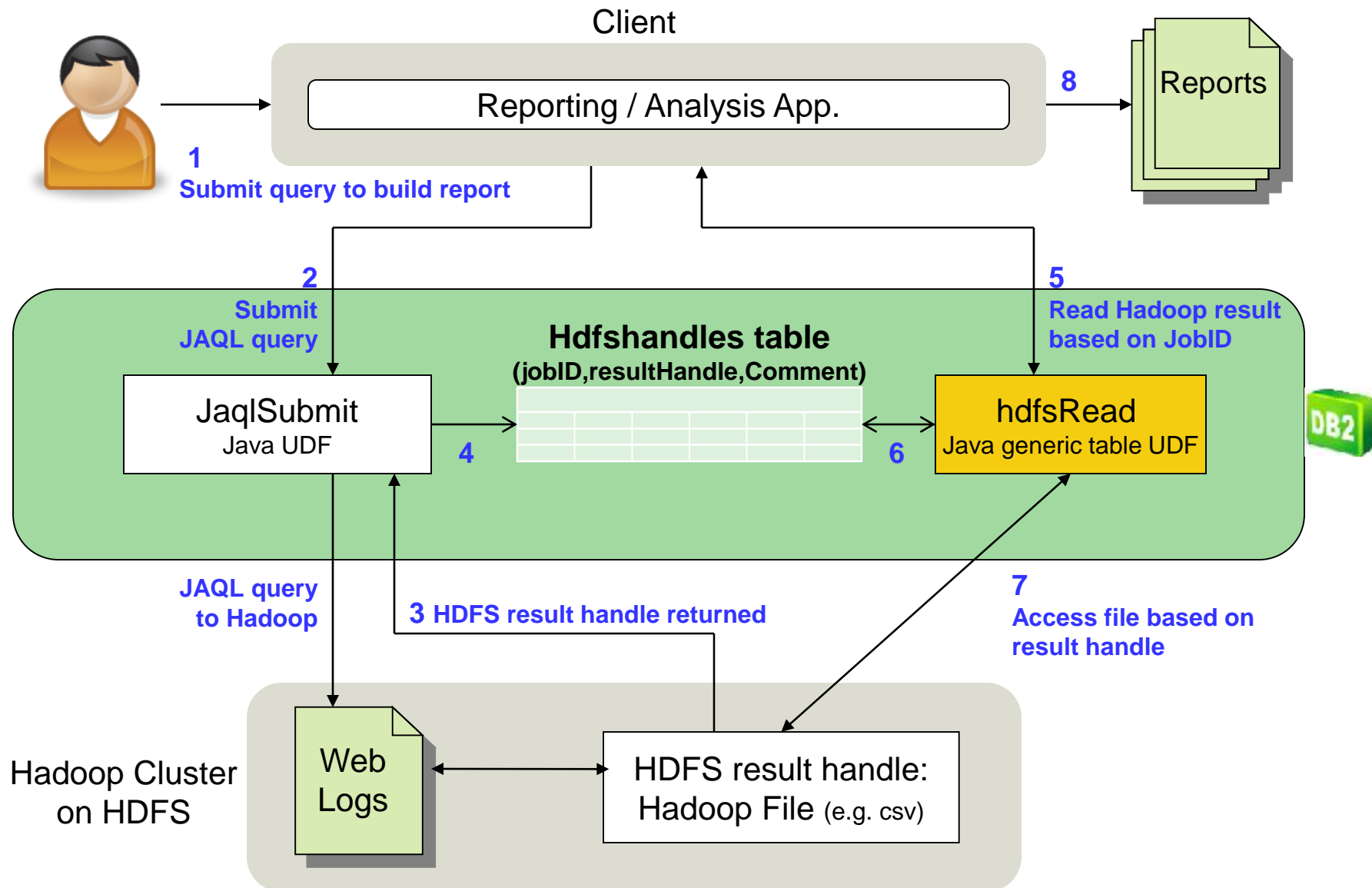
- Once registered, you can use the new Java function to work with your raw data

```
SELECT Clks.*  
FROM TABLE(CSVREAD('TMP/data/usrWebclicks.log'))  
AS Clks(LINK VARCHAR(128), USER INT)  
WHERE Clks.LINK LIKE 'www.ibm.com%';
```

- The generic table function is called within the FROM clause of the statement
 - returns a table to the calling statement, one row at a time
- The 'AS' clause is a typed correlation clause
 - defines the appearance and contents of the output table



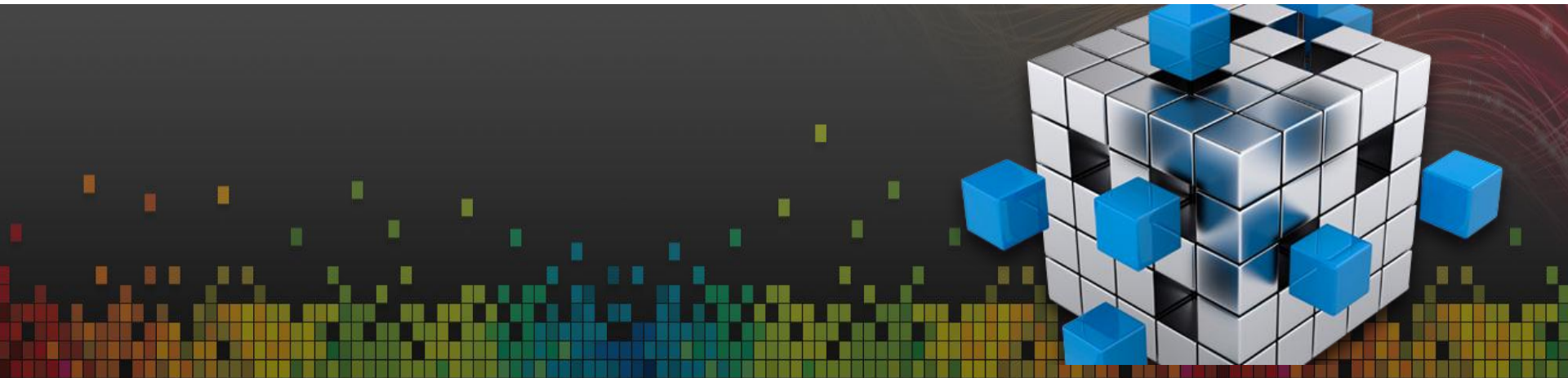
Example of Integration With InfoSphere BigInsights



Enhanced Analytics With Big Data Summarized

- Java Generic functions are a great way to leverage the strengths of DB2 and infrastructures such as Hadoop
- DB2 10 enhances the functionality of table functions by allowing the format of output to be determined at runtime, instead of, at creation
- This new and unique feature provides more flexibility and improves productivity
 - Development effort for multiple outputs is reduced
 - Increases ease for end-users and applications in need of complex reporting and analysis that brings in data from different sources

The next steps...



The Next Steps...

- Complete the online quiz for this module
 - Log onto SKI, go to “My Learning” page, and select the “In Progress” tab.
 - Find the module and select the quiz
- Provide feedback on the module
 - Log onto SKI, go to “My Learning” page
 - Find the module and select the “Leave Feedback” button to leave your comments



An abstract geometric composition featuring a variety of shapes including squares, circles, and triangles. The color palette is dominated by shades of blue (light, medium, and dark), yellow, and orange. The elements are arranged in a non-representational, overlapping manner. Notable features include a large yellow square with orange diagonal stripes, a dark blue square with a white circle and smaller squares inside, and several smaller squares and circles scattered throughout the composition. The overall effect is a dynamic and colorful abstract pattern.