Z09

Query Parallelism Enhancements in DB2 UDB for z/OS - V8 and Beyond

Fen-Ling Lin

IBM Data Management

Technical Conference

Las Vegas, NV October 27 - October 31, 2003

RETURN TO INDEX

Presentation Outline

- Parallel Sort
- Enable Query Parallelism for Multi-columns Merge Join
- Query Parallelism for DPSI
- Query Parallelism for Star Join
- Future direction

How does DB2 Handle Sort in Query Parallelism

```
SELECT    LASTNAME, FIRSTNAME, HIREDATE
FROM      EMP
ORDER BY HIREDATE;
```

Sequential Sort

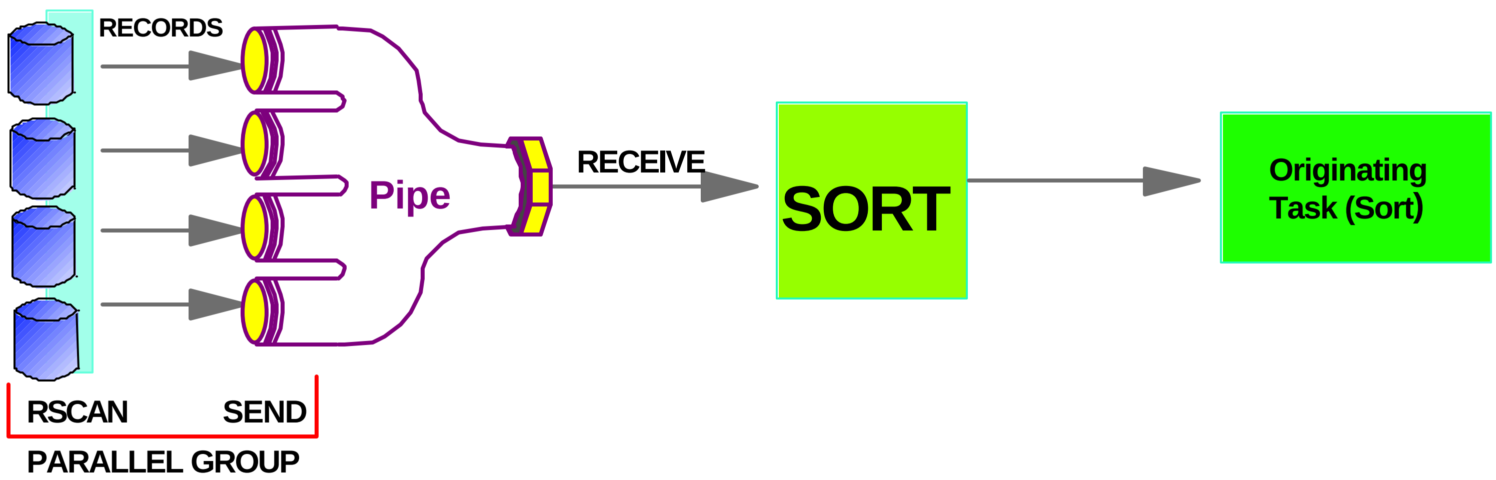
QUERYNO	PLANNO	METHOD	ACCESS TYPE	SORTC_ ORDERBY	PRE-FETCH	ACCESS_ DEGREE	ACCESS_ PGROUPID	SORTC_ PGROUPID
44	1	0	R	N	S	4	1	?
44	2	3		Y		?	?	?

Parallel Sort

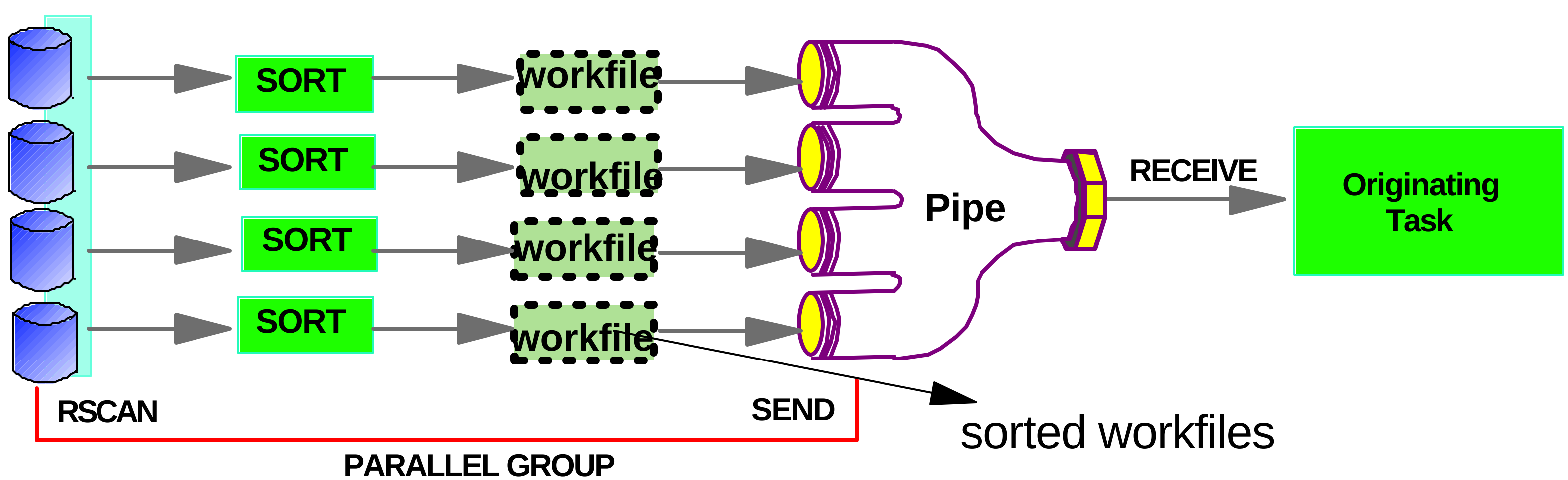
QUERYNO	PLANNO	METHOD	ACCESS TYPE	SORTC_ ORDERBY	PRE-FETCH	ACCESS_ DEGREE	ACCESS_ PGROUPID	SORTC_ PGROUPID
44	1	0	R	N	S	4	1	?
44	2	3		Y		?	?	1

RScan with ORDERBY Sort

Sequential Sort



Parallel Sort



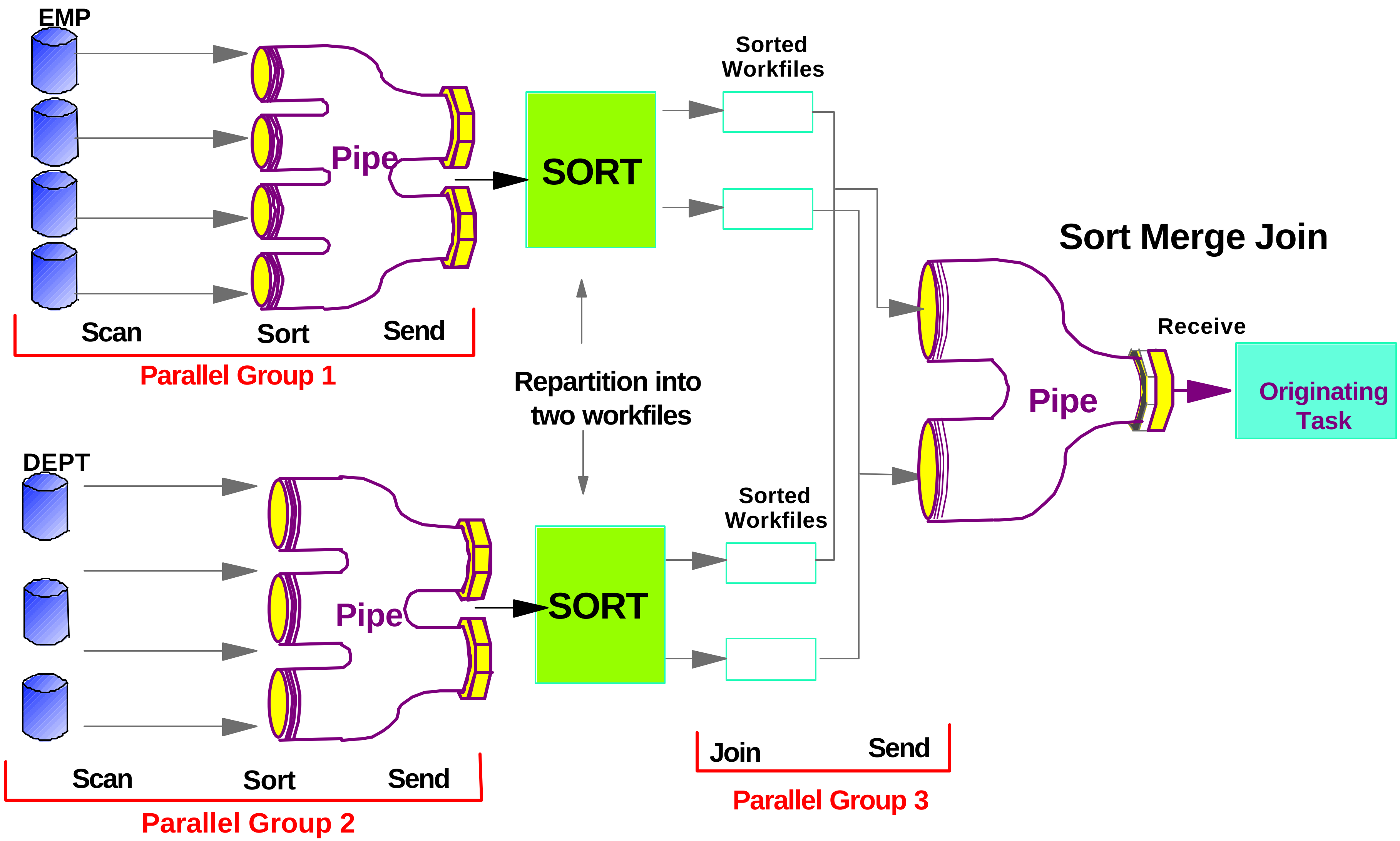
Sort Merge Join - Sequential Sort

```
SELECT  *
FROM    EMP, DEPT
WHERE   EMP.EMPNO = DEPT.MGRNO;
```

Sort Merge Join with Parallel Sort

QRY#	PLAN#	TNAME	METH	ACCESS TYP	PRE-FETCH	ACCESS_ DEGREE	ACCESS_ PGROUP	SORTN_ PGROUP	SORTC_ PGROUP	JOIN_ PGROUP	JOIN_ DEGREE
50	1	EMP	0	R	S	4	1	?	?	?	?
50	2	DEPT	2	R	S	3	2	?	?	3	2

Sort Merge Join - Sequential Sort



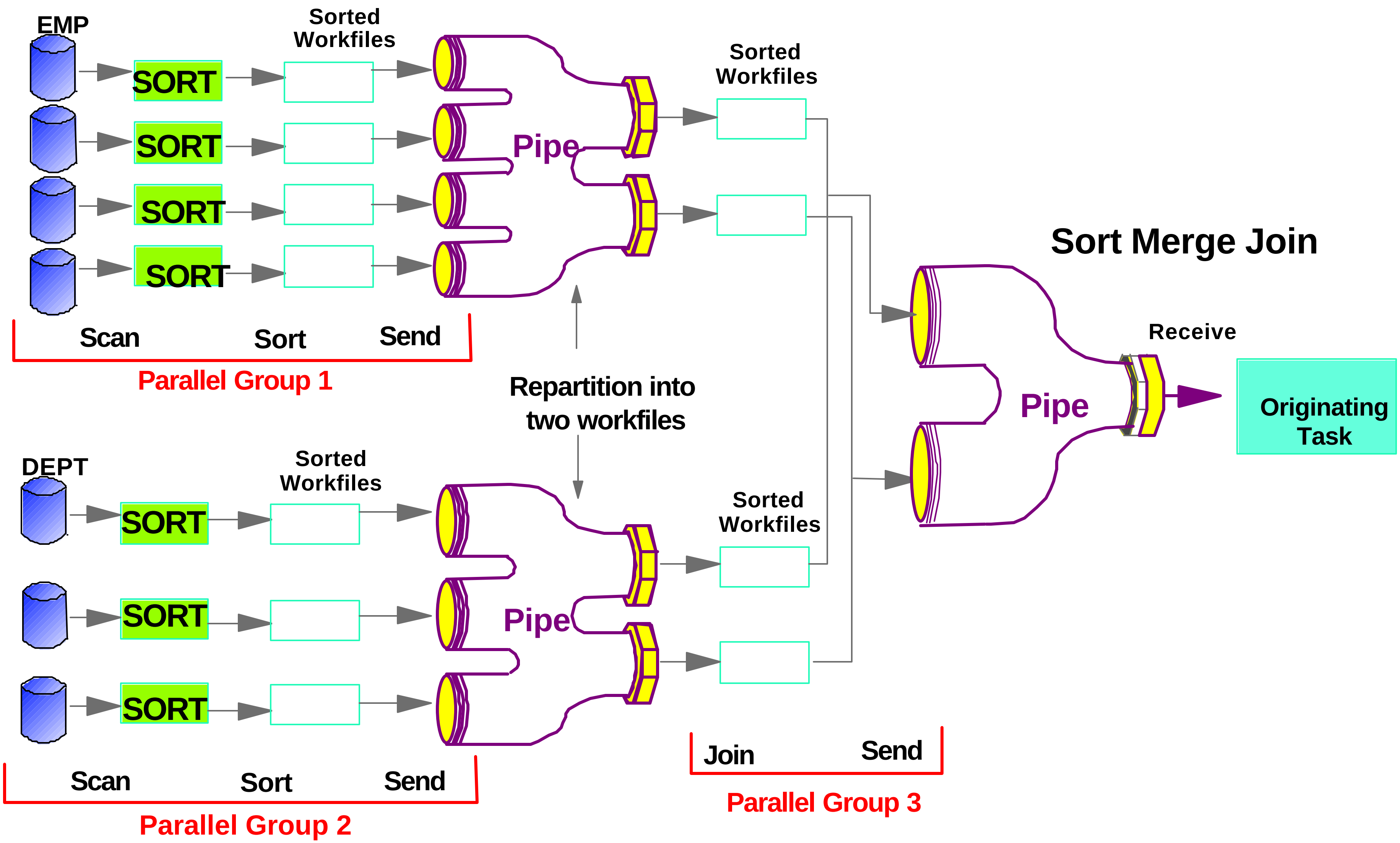
Sort Merge Join - Parallel Sort

```
SELECT   *
FROM     EMP, DEPT
WHERE    EMP.EMPNO = DEPT.MGRNO;
```

Sort Merge Join with Parallel Sort

QRY#	PLAN#	TNAME	METH	ACCESS TYP	PRE-FETCH	ACCESS_ DEGREE	ACCESS_ PGROUP	SORTN_ PGROUP	SORTC_ PGROUP	JOIN_ PGROUP	JOIN_ DEGREE
50	1	EMP	0	R	S	4	1	?	?	?	?
50	2	DEPT	2	R	S	3	2	2	1	3	2

Sort Merge Join - Parallel Sort



V8 Parallel Sort Enhancement

- Prior to V8, Parallel Sort can only be done when the parallel group is a parallel access group (single table)
- V8 support parallel sort on **multiple tables**
- **Cost-based** consideration for parallel sort - single or multiple tables
  - Pro - Elapsed time improvement
  - Con - more usage of workfiles and virtual storage
  - Threshold to disable parallel sort
    - Total sort data size (< 2MB, 500pages)
    - Sort data size per parallel degrees (< 100KB, 25 pages)

Sort Merge Join + Sort Merge Join (V7)

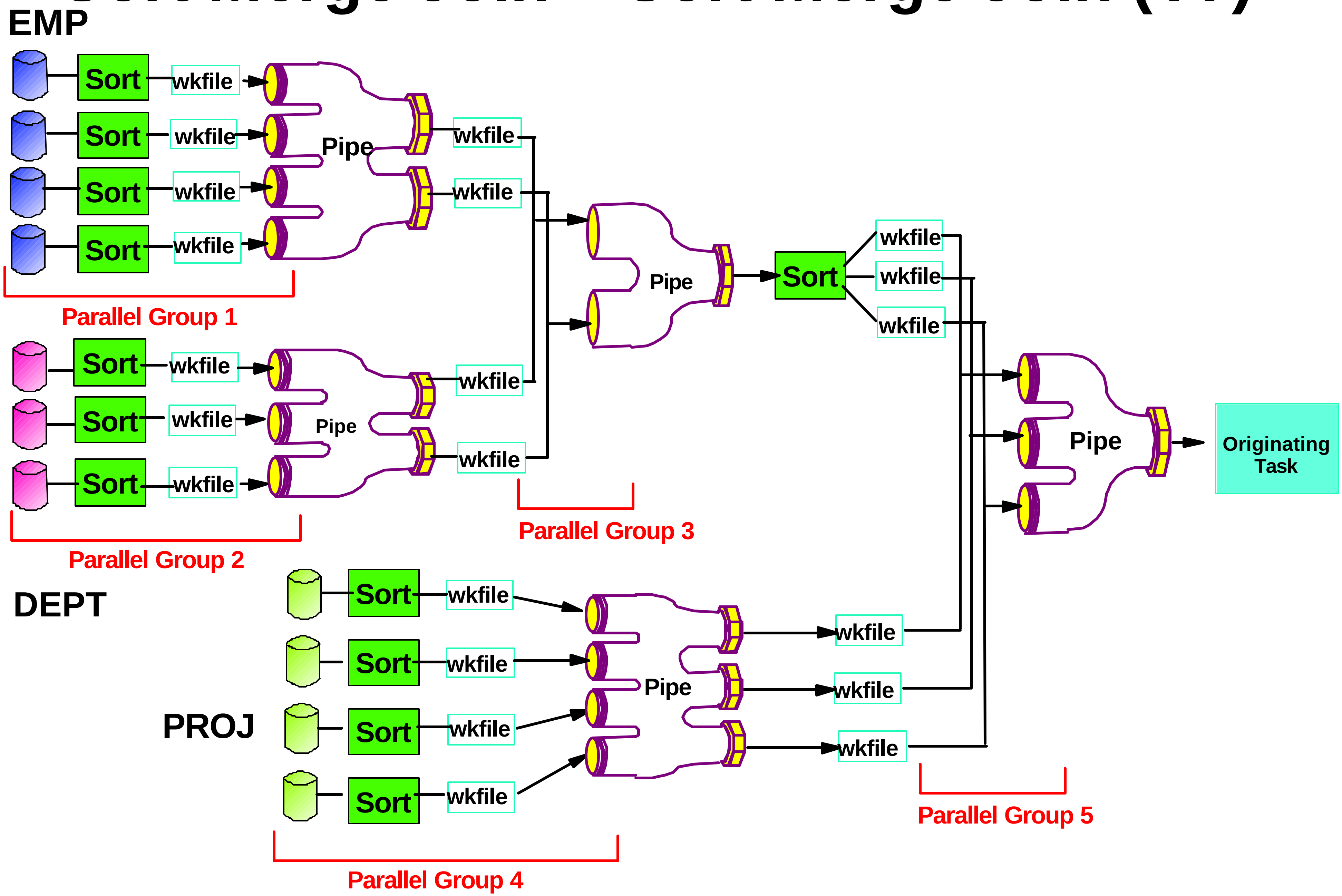
```
SELECT  *
FROM    EMP, DEPT, PROJ
WHERE   EMP.EMPNO = DEPT.MGRNO  AND
        DEPT.DEPTNO = PROJ.DEPTNO;
```

SMJ + SMJ

QRY #	PLAN #	TNAME	METH	ACCESS_TYP	PRE-FETCH	ACC_DEGREE	ACC_PGROUP	SORTN_PGROUP	SORTC_PGROUP	JOIN_PGROUP	JOIN_DEGREE
50	1	EMP	0	R	S	4	1	?	?	?	?
50	2	DEPT	2	R	S	3	2	2	1	3	2
50	3	PROJ	2	R	S	4	4	4	?	5	3

sequential sort

Sort Merge Join + Sort Merge Join (V7)



Sort Merge Join + Sort Merge Join (V8)

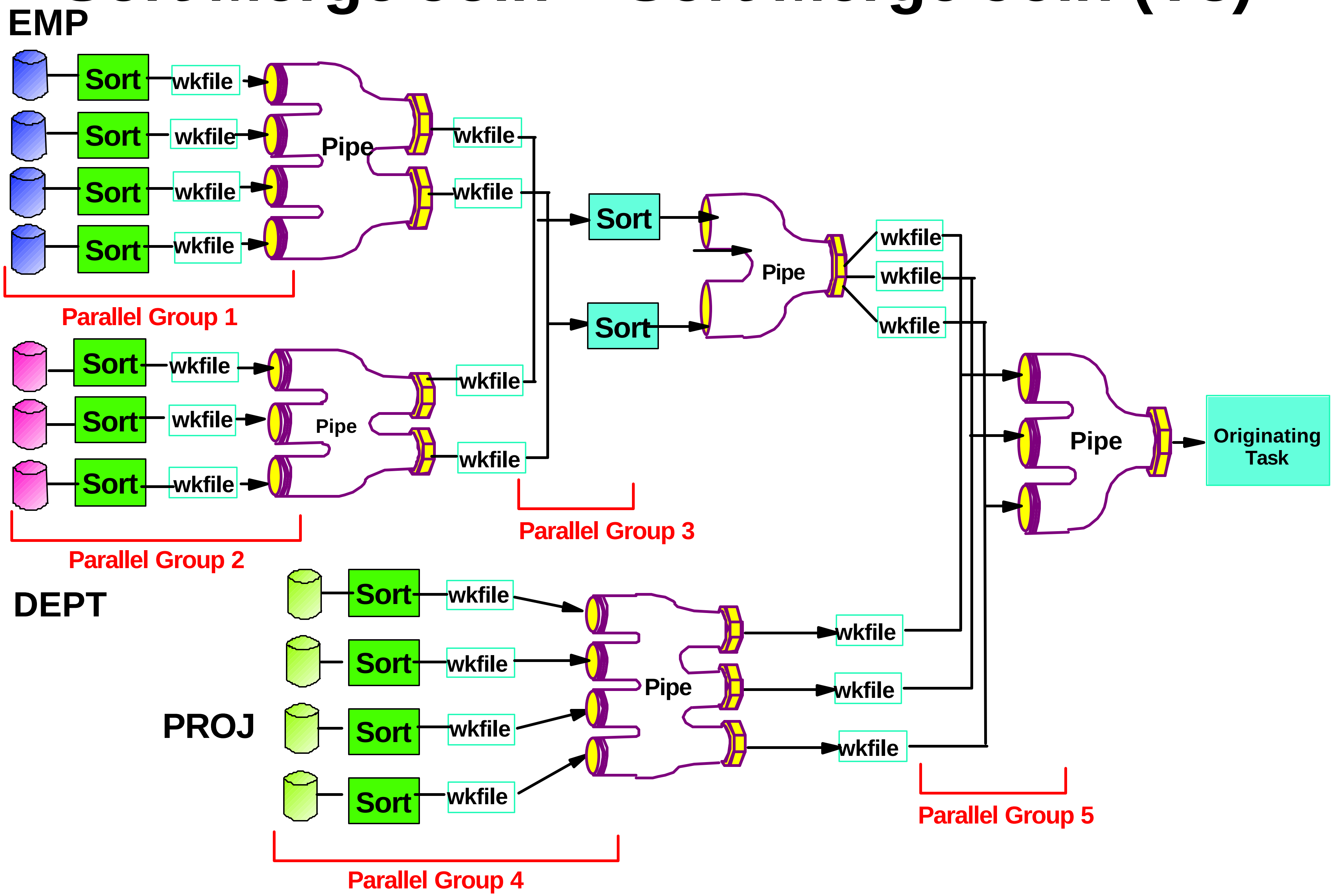
SELECT *
FROM EMP, DEPT, PROJ
WHERE EMP.EMPNO = DEPT.MGRNO AND
DEPT.DEPTNO = PROJ.DEPTNO;

SMJ + SMJ

QRY #	PLAN #	TNAME	METH	ACCES S_TYP	PRE-FETCH	ACC_ DEGREE	ACC_ PGROUP	SORTN_ PGROUP	SORTC_ PGROUP	JOIN_ PGROUP	JOIN_ DEGREE
50	1	EMP	0	R	S	4	1	?	?	?	?
50	2	DEPT	2	R	S	3	2	2	1	3	2
50	3	PROJ	2	R	S	4	4	4	3	5	3

parallel sort

Sort Merge Join + Sort Merge Join (V8)



Presentation Outline

- Parallel Sort
- Enable Query Parallelism for Multi-columns Merge Join
- Query Parallelism for DPSI
- Query Parallelism for Star Join
- Future direction

Parallelism for Multi-column SMJ

- **Enhanced Multi-column SMJ to allow the followings in the join keys**
  - Mismatched data type (string only),
  - Different length,
  - Different Nullability,
  - Mismatch CCSID

  NOTE: Mismatched numeric types and datetime types are NOT supported yet
- **Cost-based Decision**
  - ex, whether to do 2 columns Mulit-col SMJ via index or 3 columns Multi-col SMJ vis Sort
- **Enable Parallelism for multi-column SMJ**
  - **No parallelism for Mismatch CCSID**

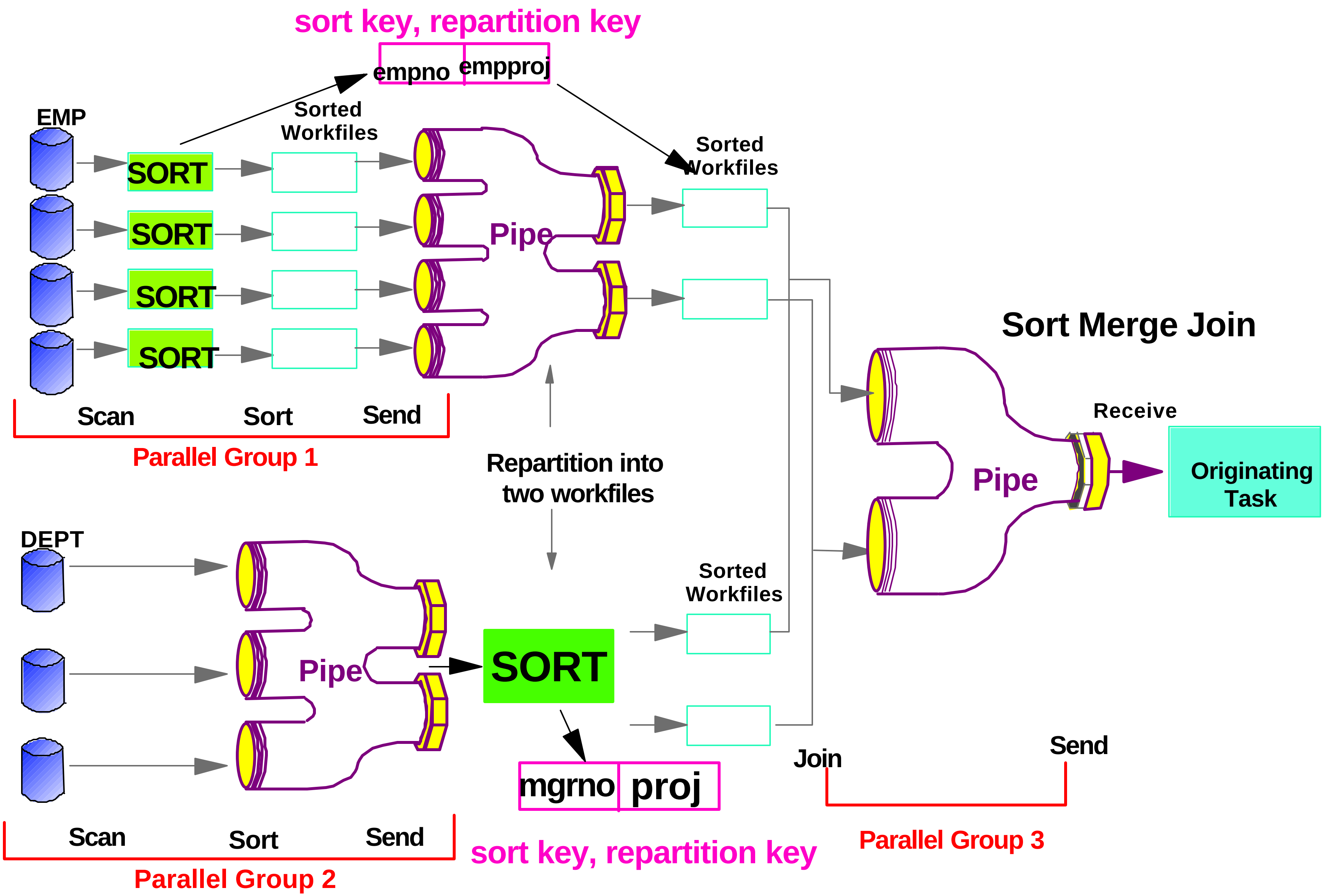
Sort Merge Join - Parallel Sort

```
SELECT  *
FROM    EMP, DEPT
WHERE   EMP.EMPNO = DEPT.MGRNO and
EMP.EMPROJ = DEPT.PROJ;
```

Sort Merge Join with Parallel Sort

QRY#	PLAN#	TNAME	METH	ACCESS TYP	PRE-FETCH	ACCESS_ DEGREE	ACCESS_ PGROUP	SORTN_ PGROUP	SORTC_ PGROUP	JOIN_ PGROUP	JOIN_ DEGREE
50	1	EMP	0	R	S	4	1	?	?	?	?
50	2	DEPT	2	R	S	3	2	2	1	3	2

Parallelism for Multi-Column SMJ



Presentation Outline

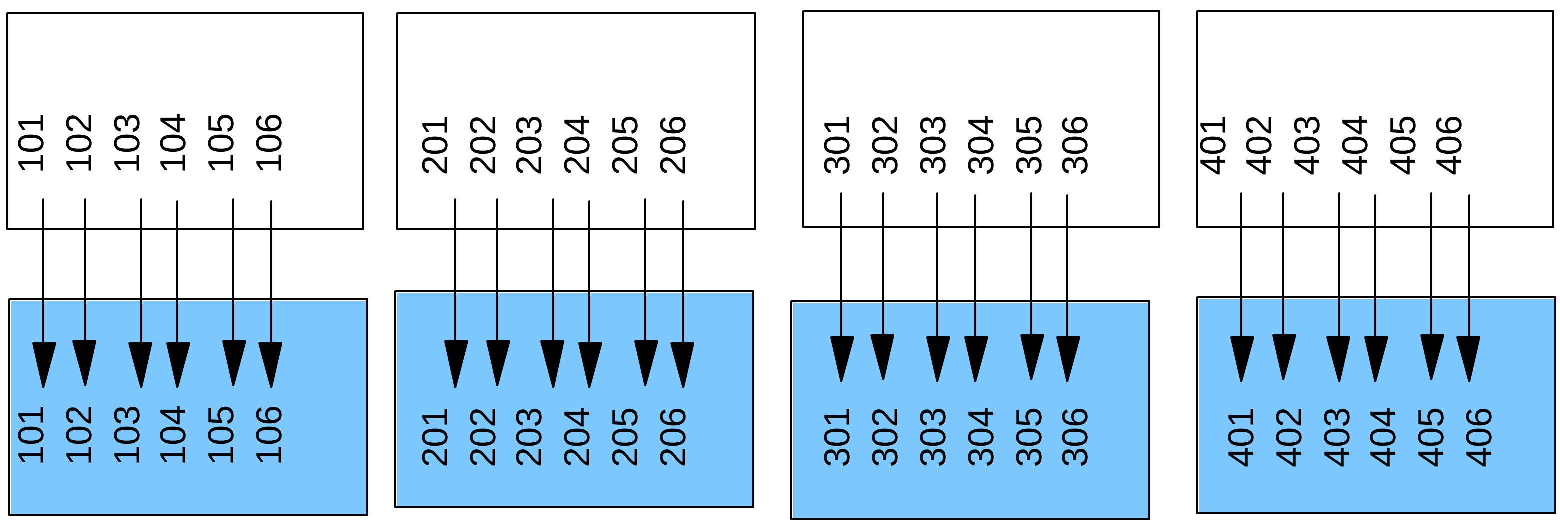
- Parallel Sort
- Enable Query Parallelism for Multi-columns Merge Join
- Query Parallelism for DPSI
- Query Parallelism for Star Join
- Future direction

V7 partitioned table and partitioning index -- index-contolled partitioning

```
CREATE TABLESPACE CUSTOMER_TS NUMPART4;
CREATE TABLE CUSTOMER (
ACCOUNT_NUM          INTEGER,
CUST_LAST_NM         CHAR(30),
. . .                                     )
IN TABLESPACE CUSTOMER_TS;

CREATE . . . INDEX  part_ix_1  ON CUSTOMER (
ACCOUNT_NUM          ASC) CLUSTER
(PART 1 VALUES (200), PART 2 VALUES (300),
 PART 3 VALUES (400), PART 4 VALUES (500));
```

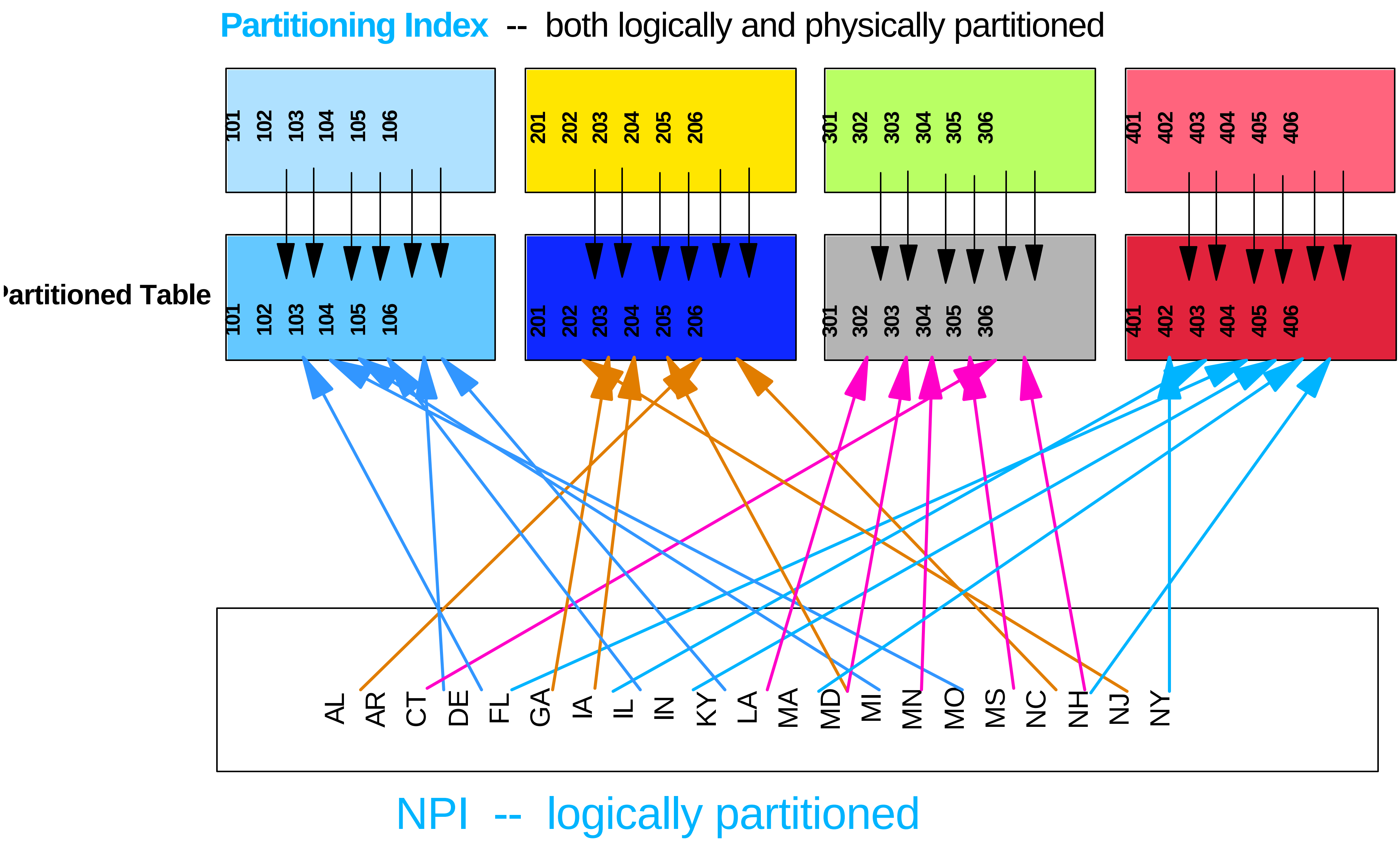
Partitioning Index



Partitioned Table

Limit keys in SYSIBM.SYSINDEXPART

Logical vs. physical partitions

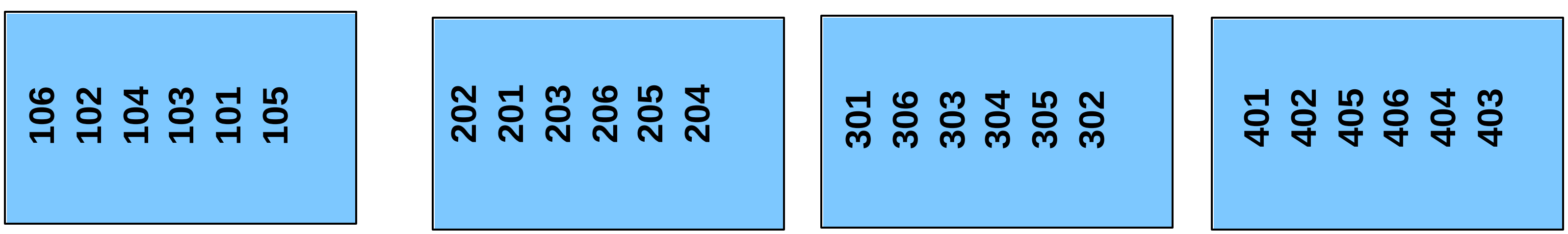


Non-partitioning index -- Secondary index -- Not partitioned -- Logically partitioned

V8 partitioned tables -- table-controlled partitioning

```
CREATE TABLE CUSTOMER (
ACCOUNT_NUM          INTEGER,
CUST_LAST_NM         CHAR(30),
. . .
LAST_ACTIVITY_DT           DATE,
STATE                      CHAR(20))
PARTITION BY (ACCOUNT_NUM         ASC)
(PART 1   VALUES (199  ),
PART 2    VALUES (299),
. . .
PART 4    VALUES (   499)       )
. . .
```

No indexes are required for partitioning!!

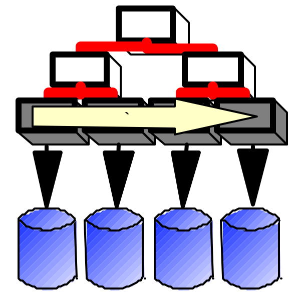


Partitioned table

Limit keys only in SYSIBM.SYSTABLEPART

V8 classification of indexes

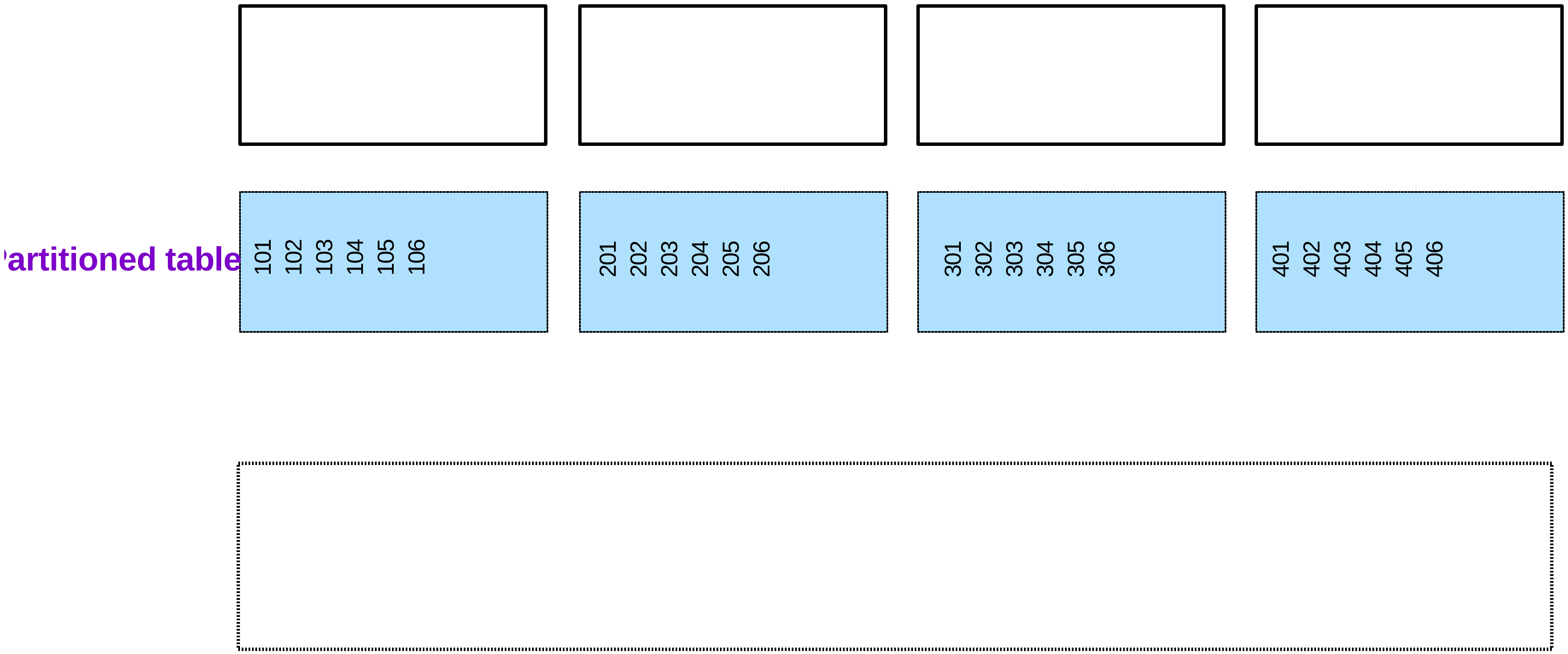
- An index may / may not by physically partitioned
  - ▶ Partitioned
    - – Index with multiple physical partitions
  - ▶ Non-partitioned
    - – Index with single physical partitions



- An index may / may not be correlated with the partitioning columns of the table
  - ▶ Partitioning index (PI)
    - – Index columns match the partitioning columns of the table
  - ▶ Secondary index (SI)
    - – Index columns do not match the partitioning columns of the table

Partitioned index vs. non-partitioned index

Partitioned index -- multiple partitions -- 1 per data partition

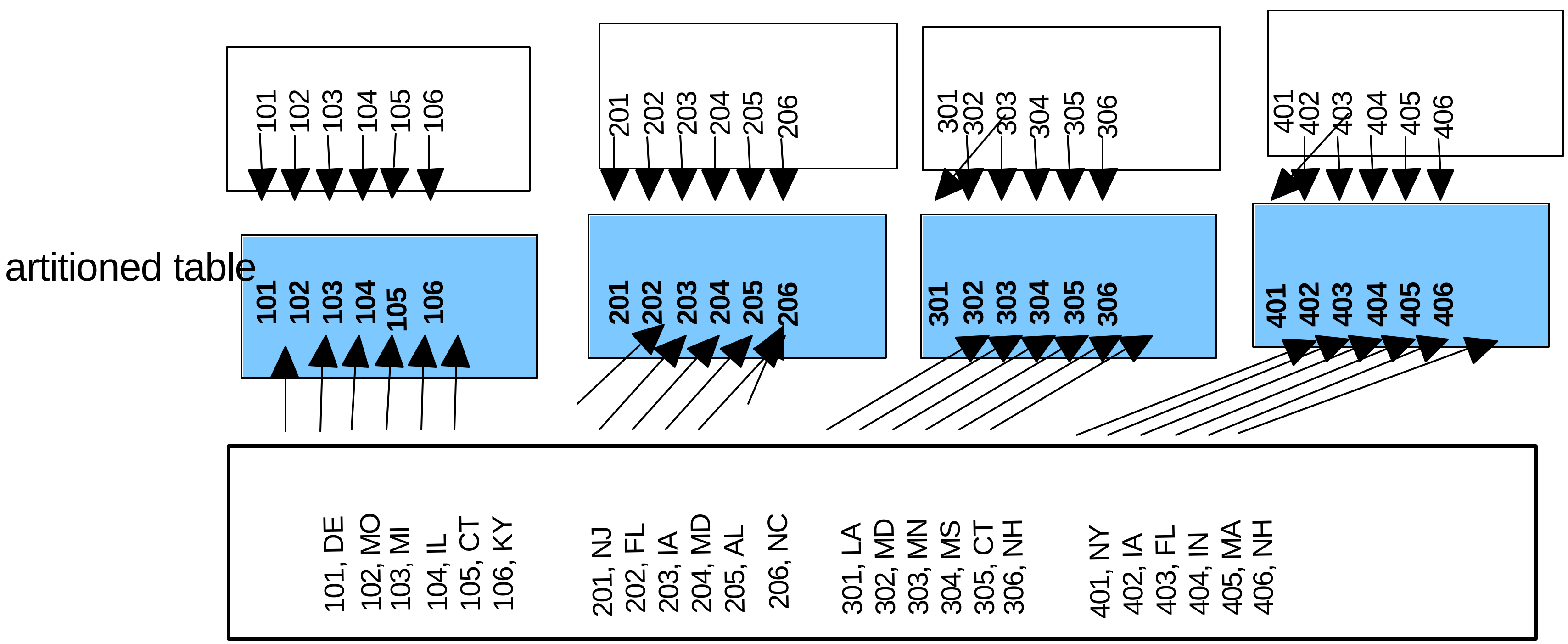


Non-partitioned index

Partitioning indexes

- A partitioning index has the same left-most columns, in the same collating sequence, as the columns which partition the table

Partitioned Partitioning index part_ix_1 (ACCOUNT_NUM)



Non-partitioned Partitioning index part_ix_2(ACCOUNT_NUM, CUST_LAST_NM)

Partitioning indexes

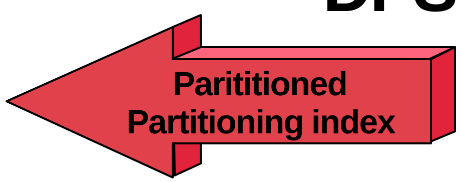
- A partitioning index
  - has the same left-most columns as the columns which partition the table
  - these columns have the same collating sequence (ASC / DESC)

```
CREATE TABLE CUSTOMER (
    ACCOUNT_NUM                  INTEGER,
    CUST_LAST_NM                 CHAR(30),
    . . .                                         )
    PARTITION BY (ACCOUNT_NUM          ASC)
    . . .

CREATE . . . INDEX  part_ix_1  ON CUSTOMER (
    ACCOUNT_NUM                  ASC) PARTITIONED;

CREATE . . . INDEX  part_ix_2  ON CUSTOMER (
    ACCOUNT_NUM                  ASC,
    CUST_LAST_NM                 ASC)
```

DPSI



NPSI

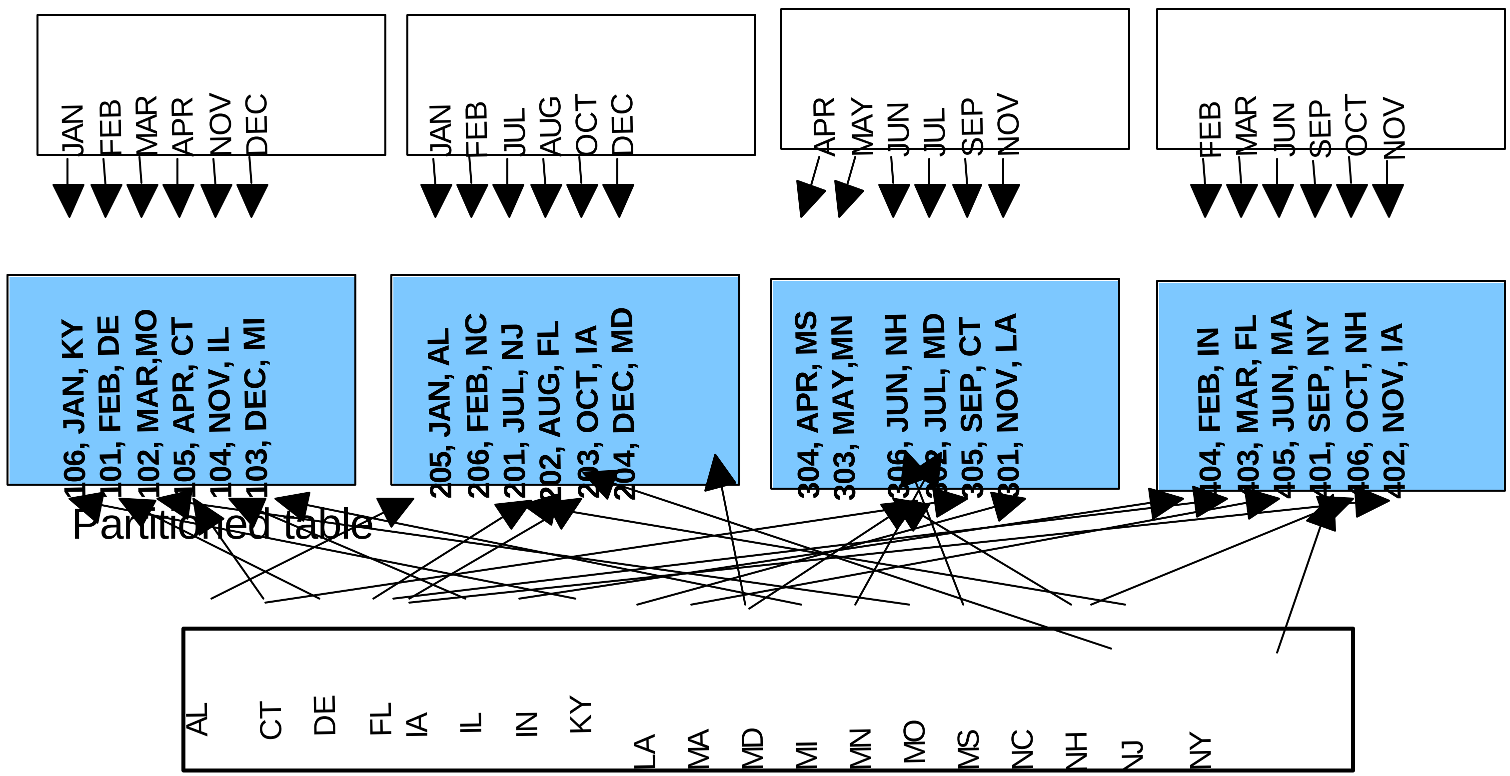
Secondary Indexes

- Data-partitioned secondary index (DPSI)
  - Index is partitioned based on data rows
  - Index contains different columns than partitioning columns
  - Must allow duplicates (be non-unique)
- Non-partitioned secondary index (NPSI)
  - Index is not partitioned
  - Index contains different columns than partitioning columns
  - May be unique

Secondary indexes

- A secondary index is any index which is **not** a partitioning index

Secondary Index -- partitioned like the underlying data (**DPSI**)



Secondary Index -- non-partitioned (**NPSI**)

V8 -- creating secondary indexes

- **CREATE . . . INDEX data_part_si_1 ON CUSTOMER**

  (LAST_ACTIVITY_DT ASC)

  . . .

  PARTITIONED

  PART 1 USING PQTY . . .

  PART 2 USING PQTY . . .

  . . .

DPSI

- **CREATE . . . INDEX non_part_si_2 ON CUSTOMER (**

  STATE ASC)

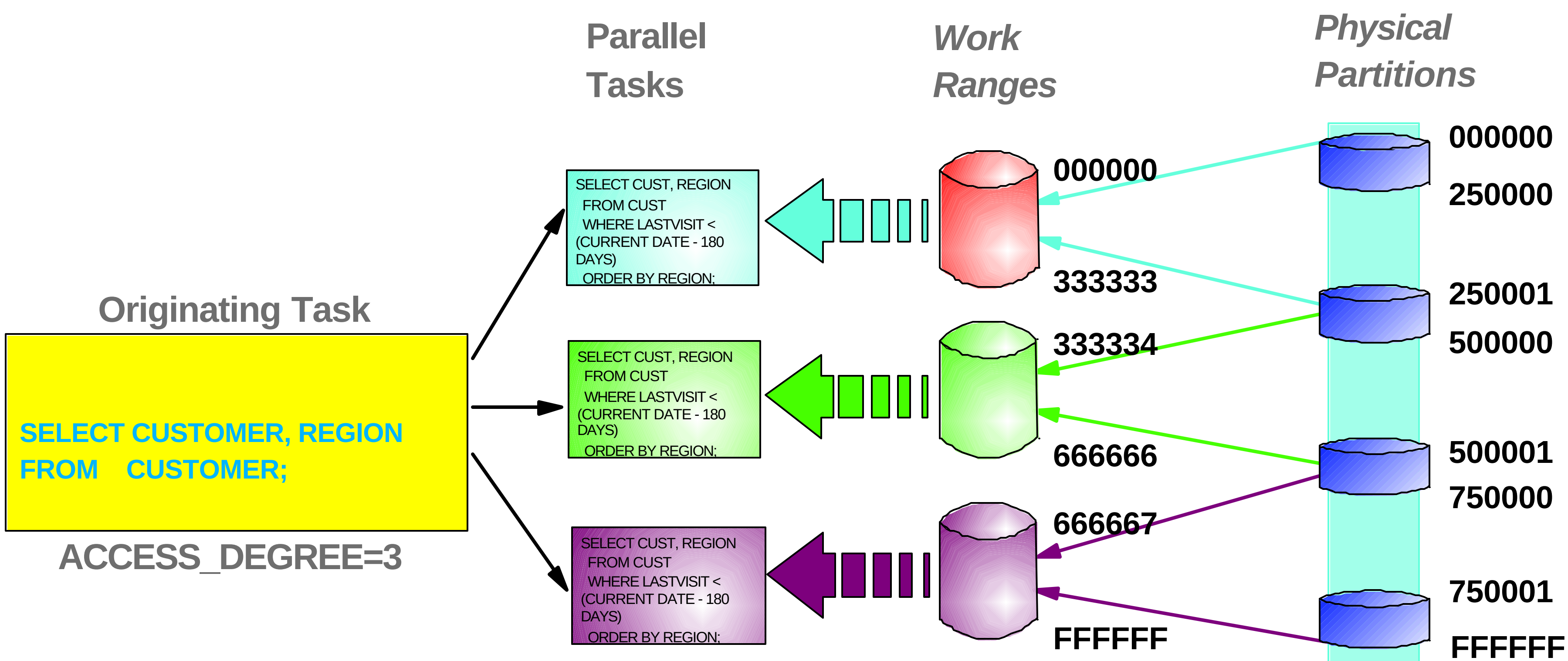
NPSI

Review of Parallelism Concepts

- How does DB2 split the work for Query Parallelism
  - DB2 splits the execution in a parallel group into pieces (DEGREE) --> work ranges
  - By page range for a tablespace scan
    - Non-partition table
    - Partition table
    - Access via DPSI index
  - By key range for an index scan
    - Non-partition index
    - Partitioning index

Review of Parallelism Concepts

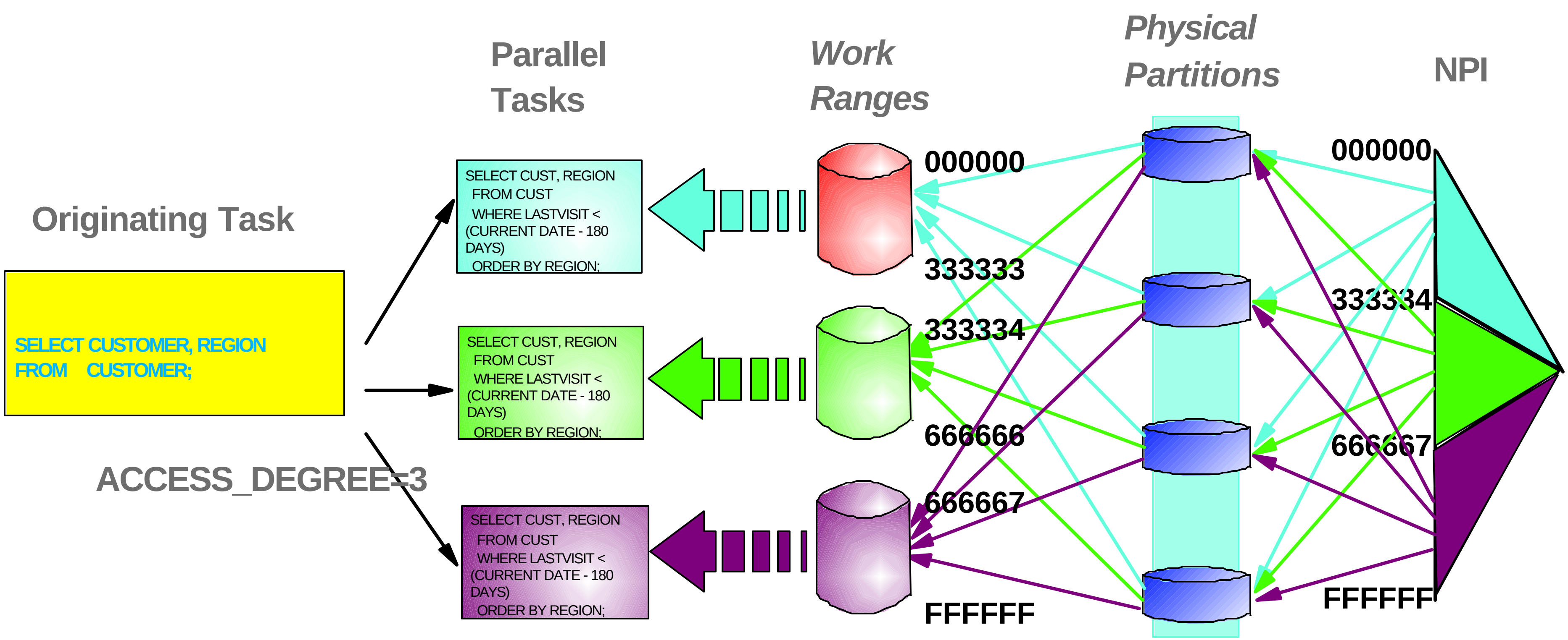
- **Work ranges in a parallel group** - page ranges



Note: Assume equal data distribution

Review of Parallelism Concepts

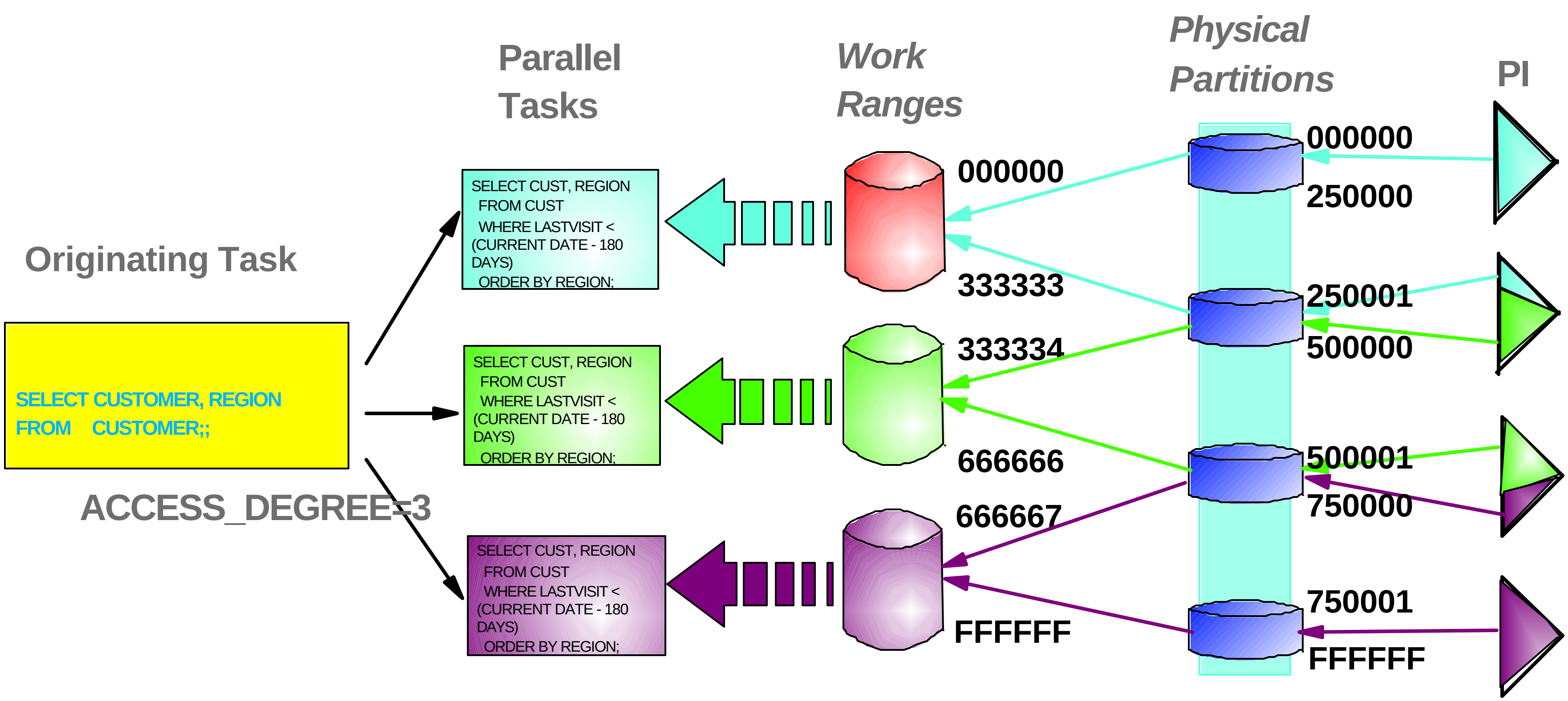
- **Work ranges in a parallel group** - NPI Key ranges



Note: Assume equal data distribution

Review of Parallelism Concepts

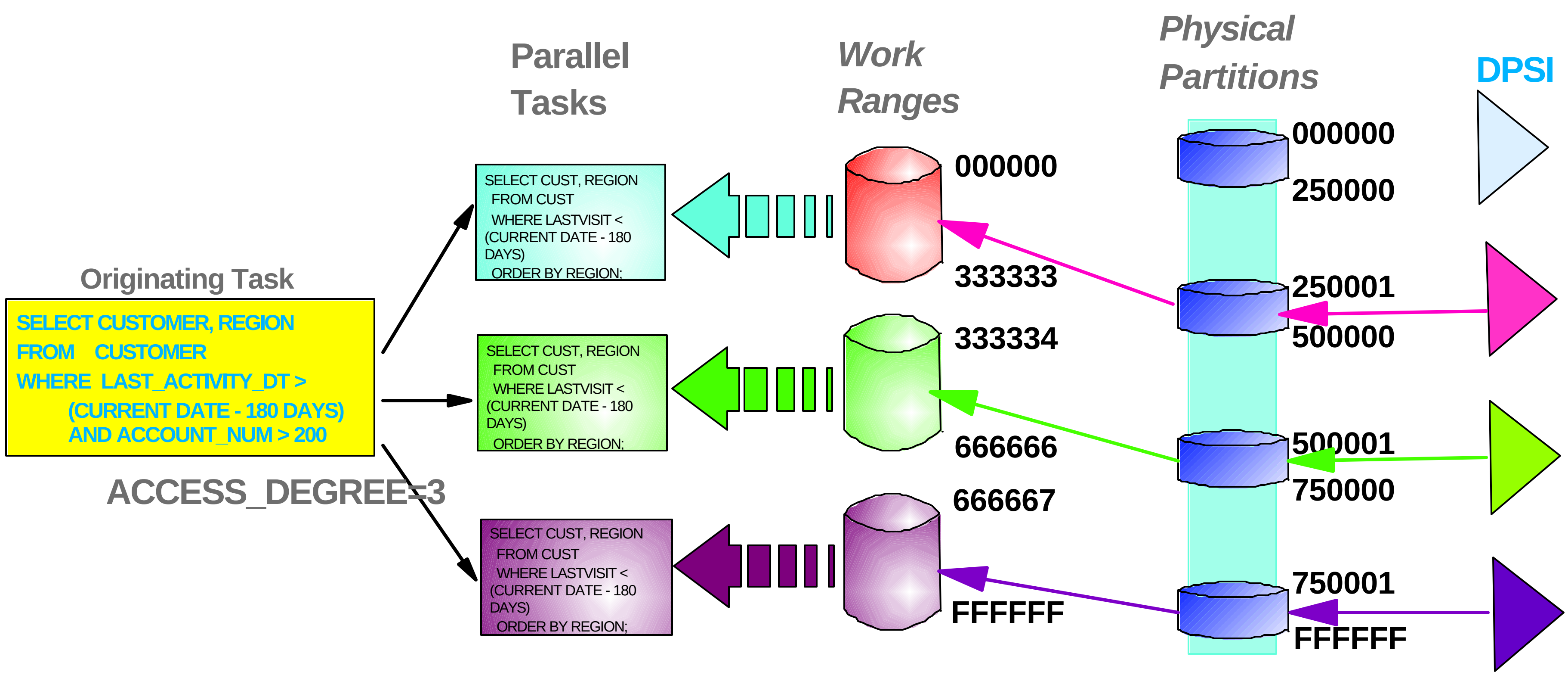
- **Work ranges in a parallel group** - PI Key ranges
- Cut on Logic boundary
- Partition pruning



Note: Assume equal data distribution

Review of Parallelism Concepts

- Work ranges in a parallel group - DPSI **PAGE** ranges
- Always cut in **Physical Partitions boundary**
- **Index access + Partition pruning**



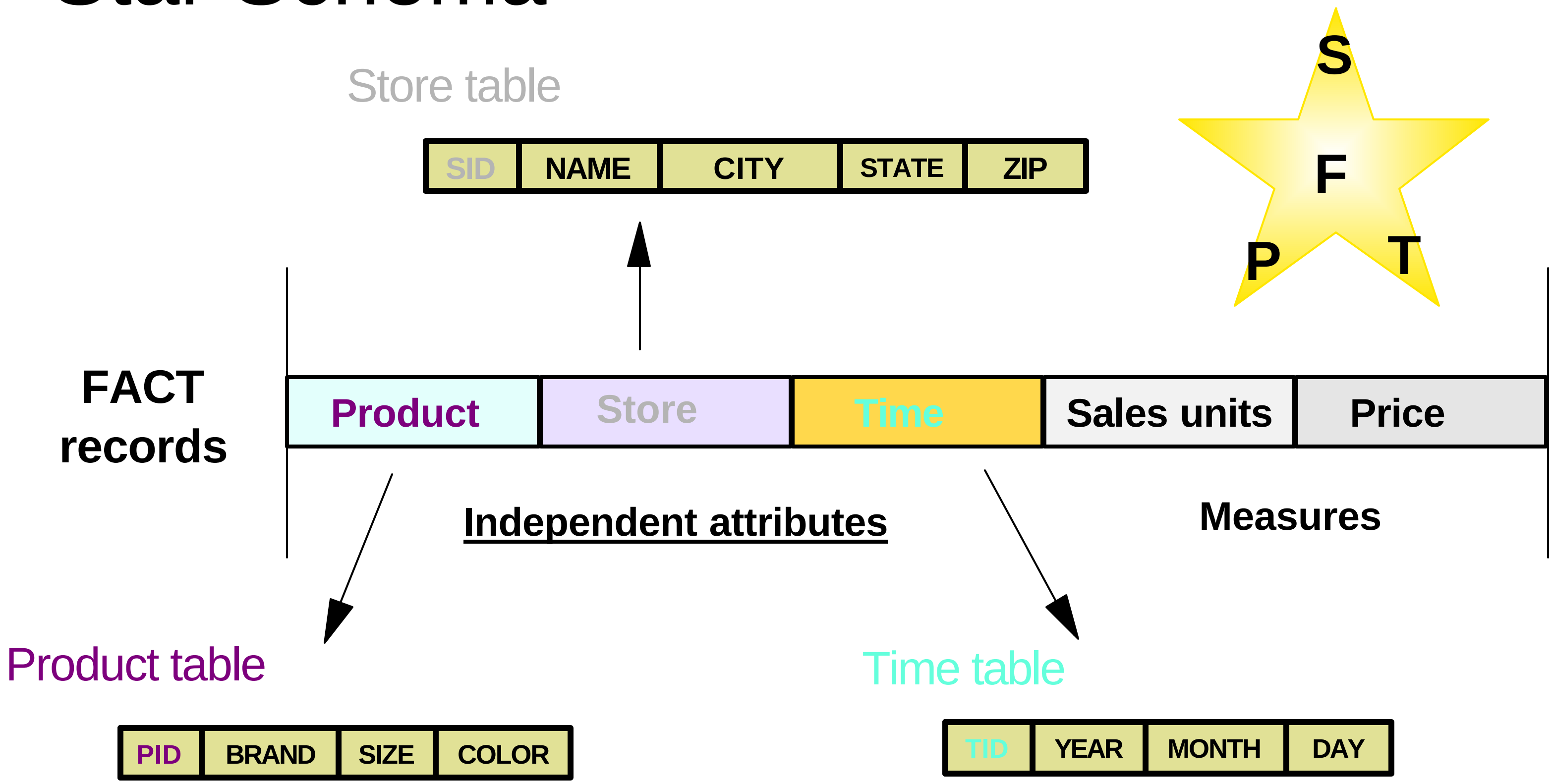
Note: Assume equal data distribution

Presentation Outline

- Parallel Sort
- Enable Query Parallelism for Multi-columns Merge Join
- Query Parallelism for DPSI
- Query Parallelism for Star Join
- Future direction

Introduction

- Star Schema



- **Integrity of data**
- **Multi-dimensional analyses**

Normalization

Degree of normalization can differ by the operational and environmental requirements

Introduction

- Star schema - what is it ?
  - A "**design principle**" of large volume databases - mainly for DW (but not limited to DW)
  - DW keep track of history data (transaction information)
  - Based on the semantics of data (Entity-Relationship)
  - Deals with multi-dimensional data
  - Normalization of a table (its independent attributes) ending up with a star like schema

Introduction

- Star schema and normalization

  - Decompose a big table which has many attributes into one **central table** referring to many **tables with fewer attributes**

    fact table (central table)

    dimension tables (tables with fewer attributes)

    - **Eliminate or reduce redundancy and dependency among the data in the central table**
    - **Dimension tables contain independent attributes**
    - **Joins will be involved to reference dimension attributes**
    - **Typically, a dimension table provides the primary key with the corresponding fact table column as the foreign key**

Introduction

Sales

BRAND	SIZE	COLOR	NAME	CITY	STATE	ZIP	YEAR	MONTH	DAY	UNITS	PRICE
Brand1	27	blue	Store1	City1	State1	95123	2002	07	28	1	60.00
Brand2	27	blue	Store1	City1	State1	95123	2002	07	29	1	55.00

redundancy

normalize

parent key

Store

SID	NAME	CITY	STATE	ZIP
01	Store1	City1	State1	95123

Dimension

foreign keys

transaction data

PID	SID	TID	UNITS	PRICE
01	01	01	1	60.00
01	01	02	1	55.00

Fact

millions of rows

Product

PID	BRAND	SIZE	COLOR
01	Brand1	27	blue

Dimension

attributes

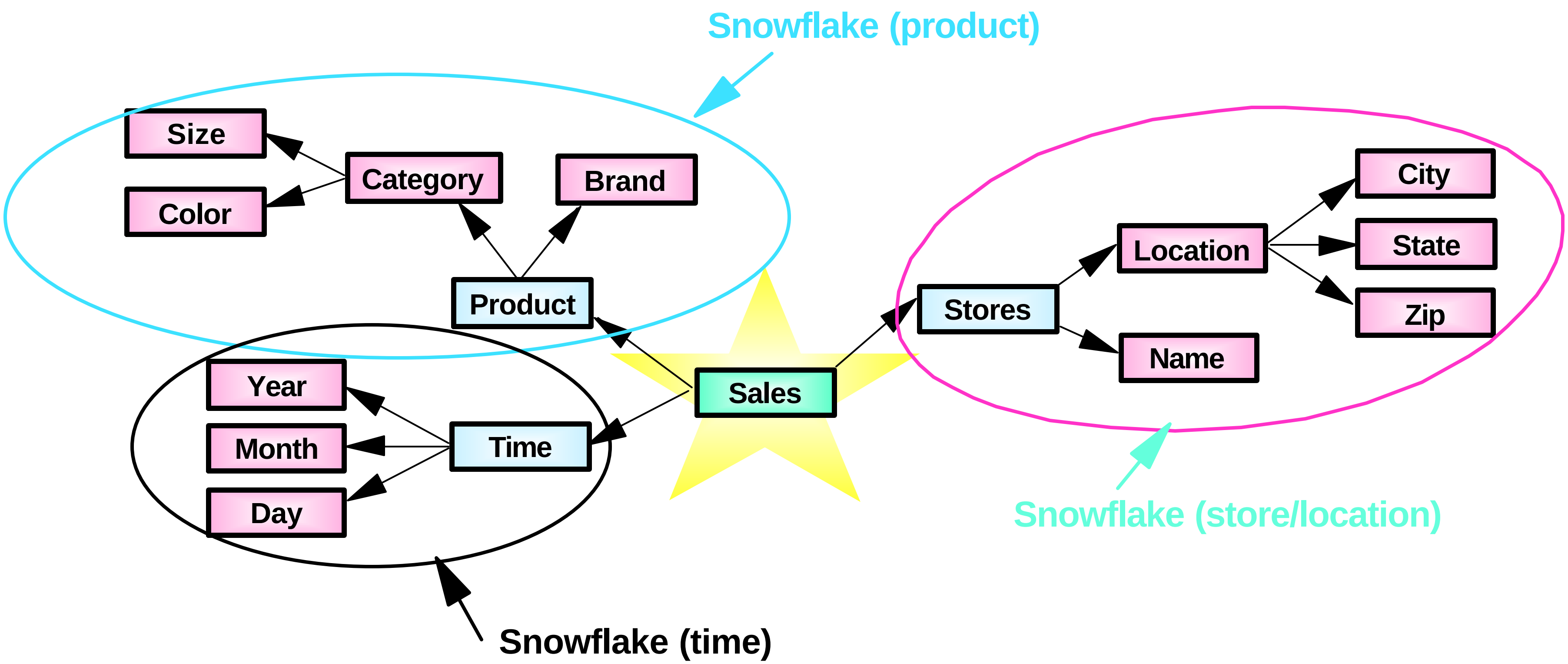
Time

TID	YEAR	MONTH	DAY
01	2002	07	28
02	2002	07	29

Dimension

Introduction

- Star schema and normalization
  - Further decomposition of dimensions -> **Snowflakes**

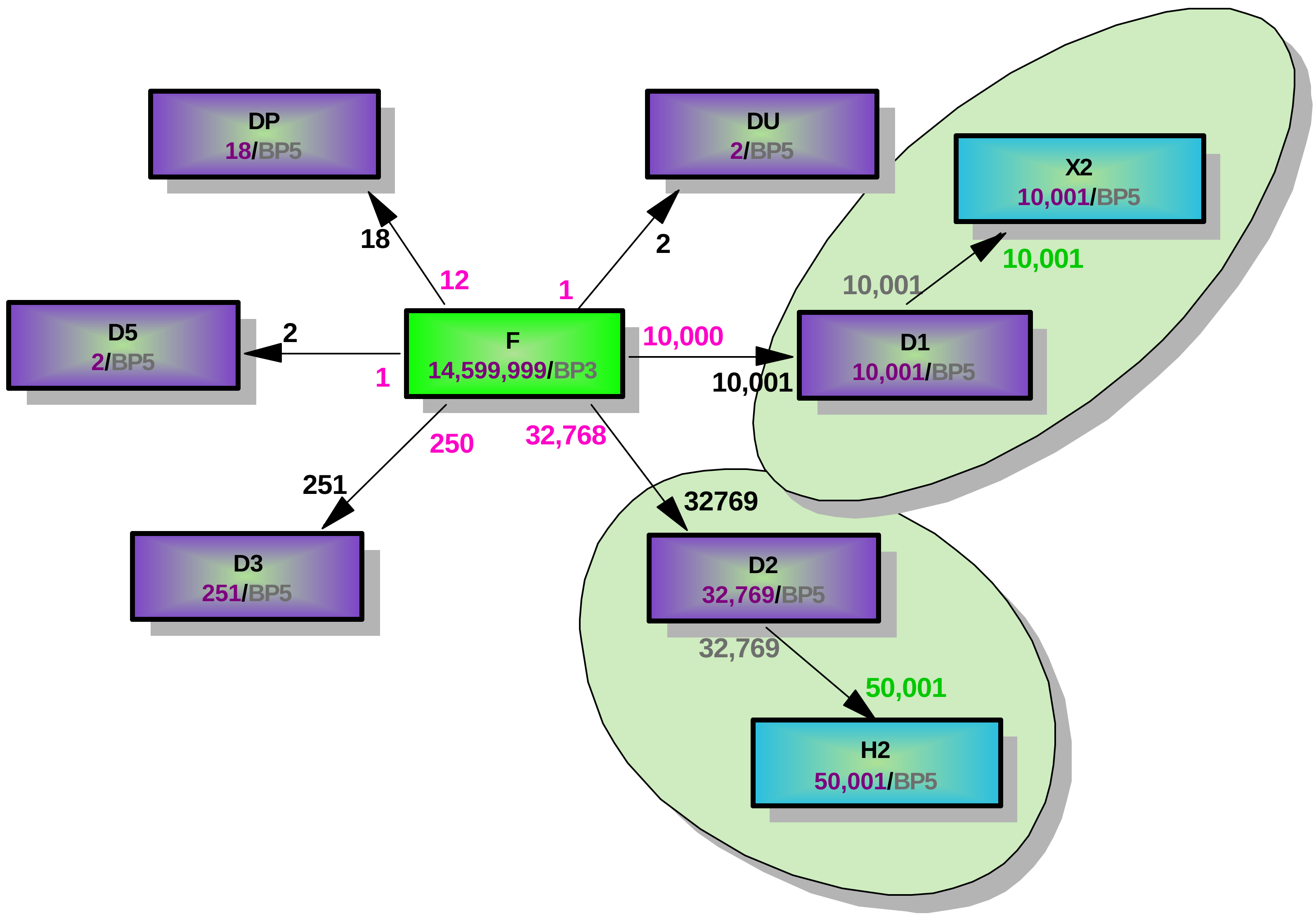


Introductin

- Star schema - general characteristics
  - Fact table
    - Large --> millions or billions of rows
    - Time variant data
    - Foreign keys + transaction data
  - Dimension table
    - Relatively stable "master" tables
    - Relatively small
    - Usually highly normalized
    - Highly correlated (sparse "hyper cube")
    - Meaningful attributes - describing the entities
  - Fact table dependent on the dimension tables

Introduction

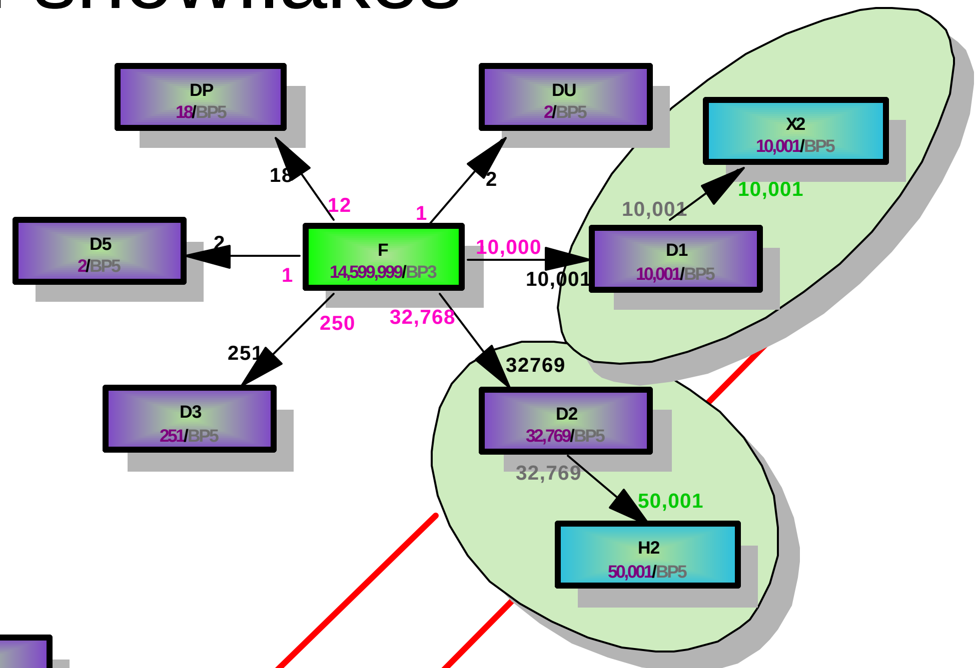
- Sample star schema (SAP/BW)

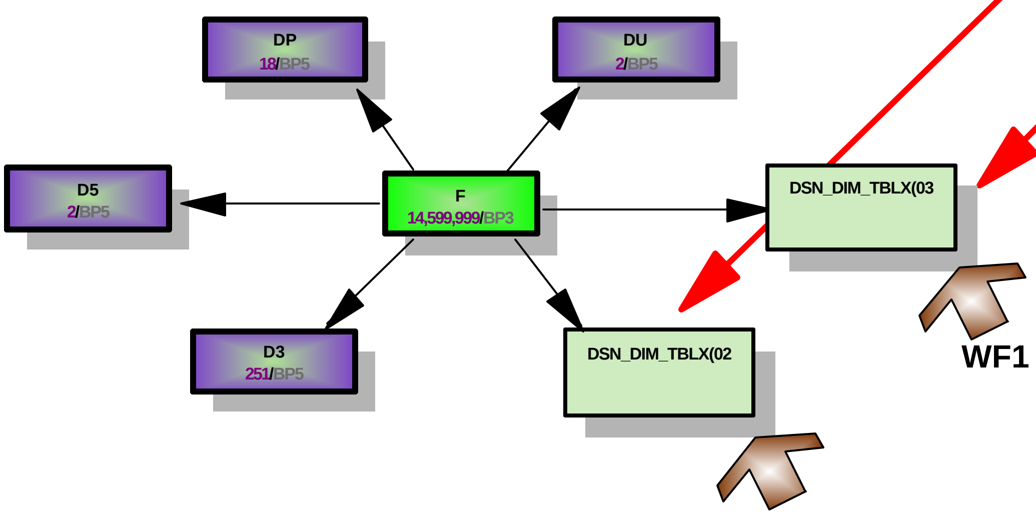


Star Join Detection & Transformation

- Materialization of snowflakes

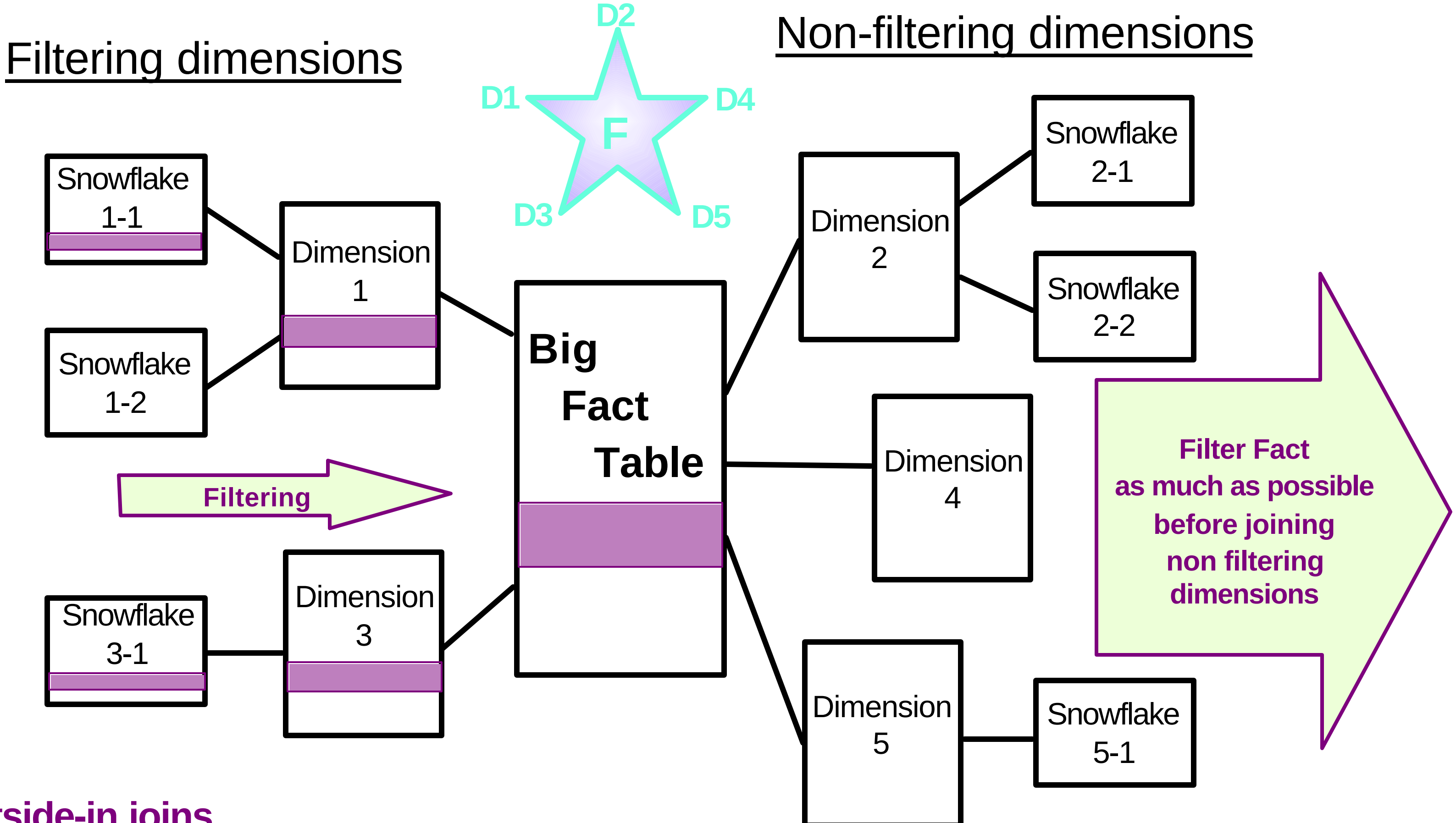
For a star join qualified query, snowflakes are materialized before the joins are optimized.





Introduction

- "Star Join" Optimization - Basic Ideas



Outside-in joins

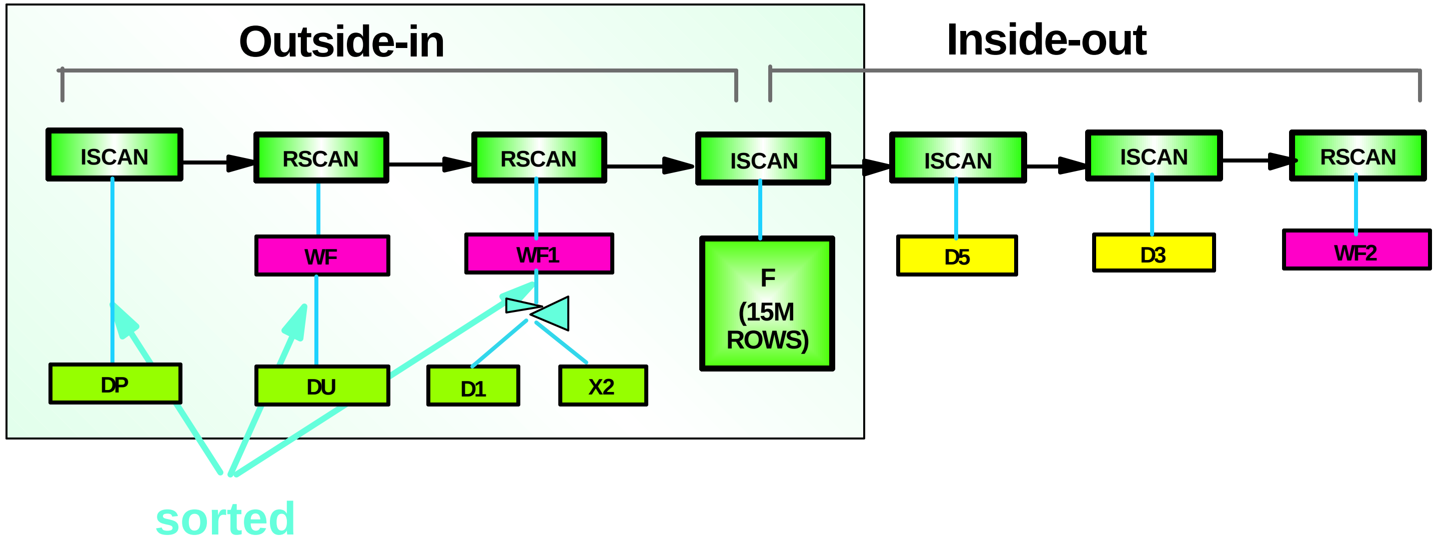
- No predicate among dimensions
- Dimensions materialized in WF
- Cartesian join performance

Inside-out joins

- Snowflakes materialized unconditionally.
- Sort may be involved (typically with SMJ and group-by)

Join/Access Methods

- Outside-In Join Phase
  - The pre-fact dimensions are sorted in the join columns orders --> workfiles
  - Index Key Feedback
  - Sparse Index
  - In-Memory workfile

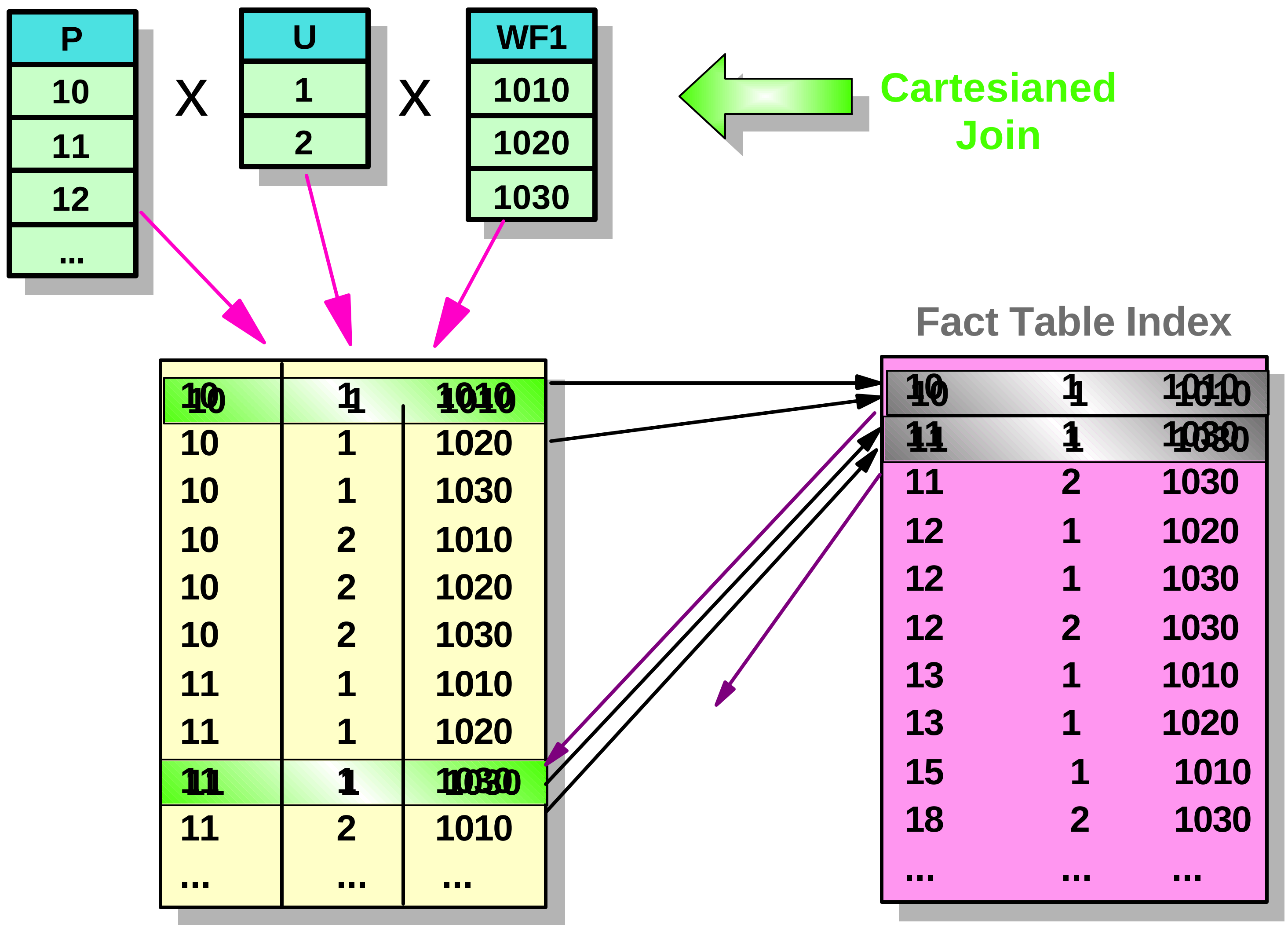


Join/Access Methods in the Outside-In Join Phase

- NLJ

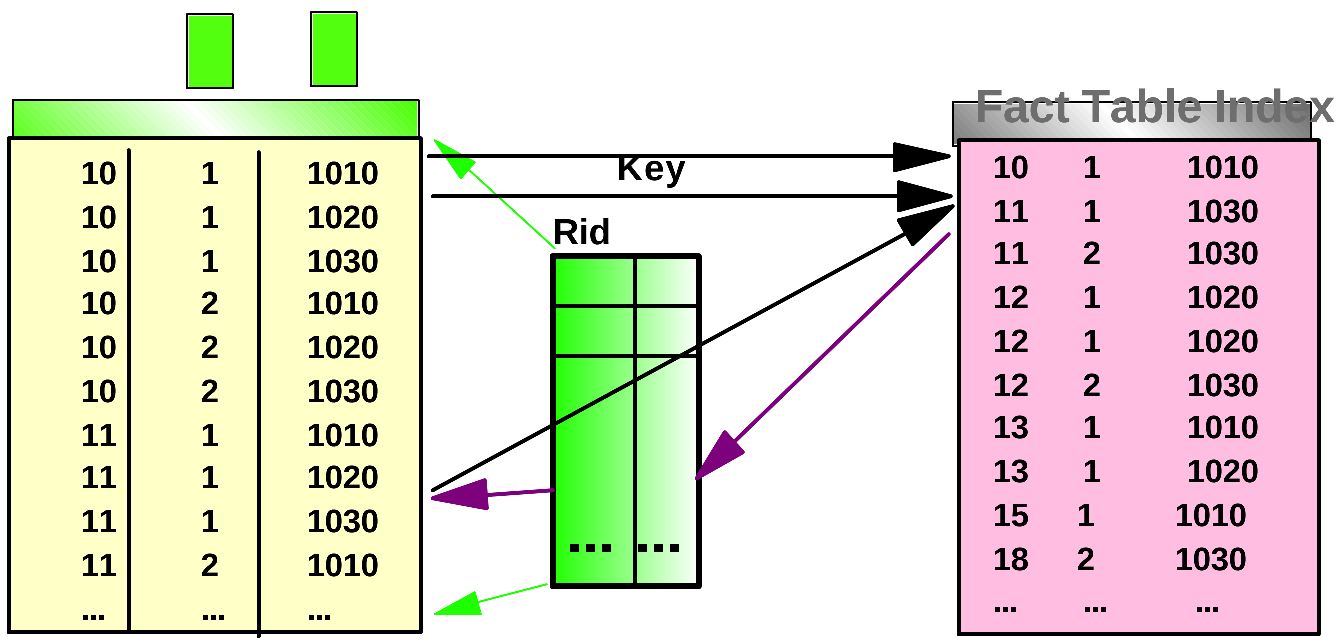
  - Leading dimension tables perform "Cartesian Joins"
    - Filtering dimensions are joined before the Fact Table
    - There is no join predicates between any two dimensions
    - Exploit a good index on Fact Table
    - Join predicates between Dimension Tables and Fact Table are supported via the Fact Table index
    - Index feedback and dimension repositioning
    - Push down join logic to lower level DB2 component
  - **Index Scan to fact table**
    - Exploit a good index to support the join in Fact Table
    - Index key feedback
    - Next valid index key is used to advance the dimension tables
    - Minimize the access to the fact table rows
    - In plan_table, "JTYPE" = "S"

Join/Access Methods in the Outside-In Join Phase



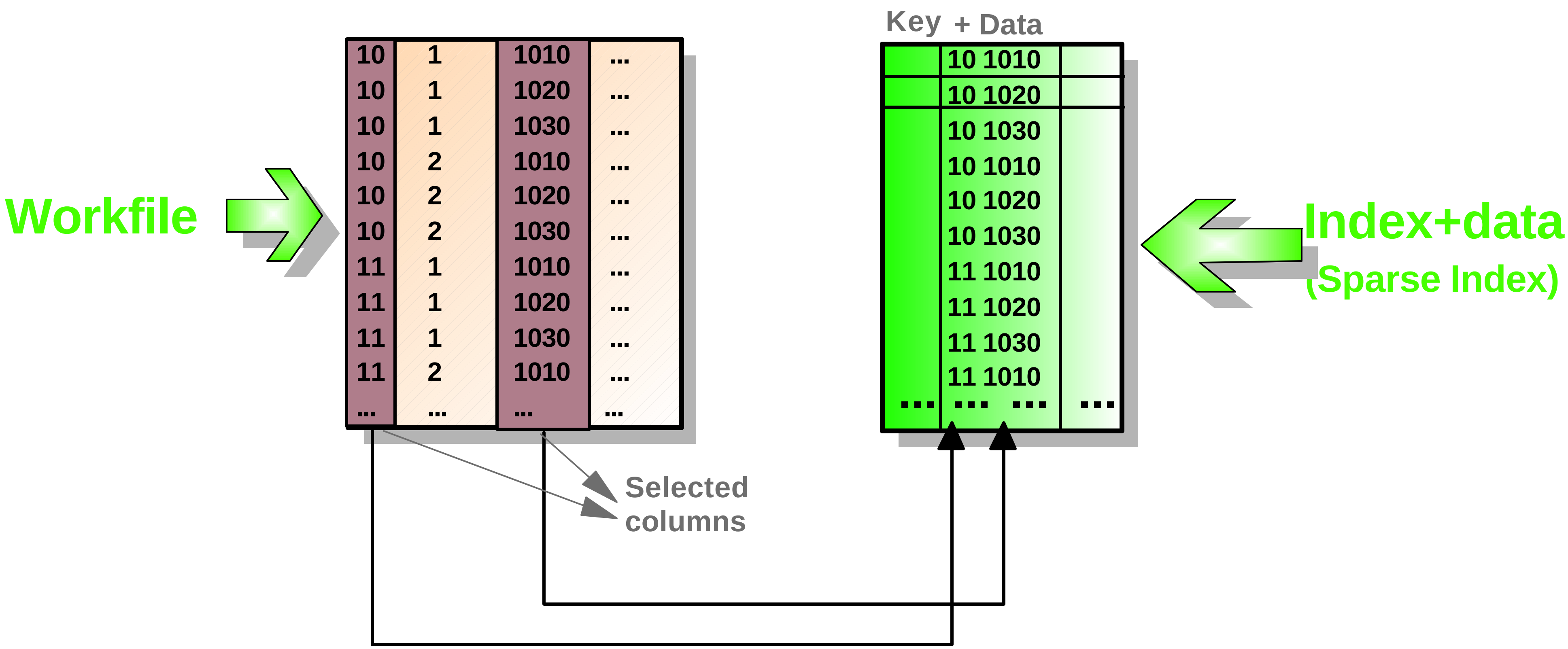
Join/Access Methods in the Outside-In Join Phase

- Sparse Index
  - In-memory index (up to 240 KB)
  - Created against dimension workfiles
    - Binary search for the target portion of the table
    - Sequential search within the target portion if it is sparse
    - Ideal solution for dimension workfile access under star schema scenario (dimensions and snowflakes are usually small)



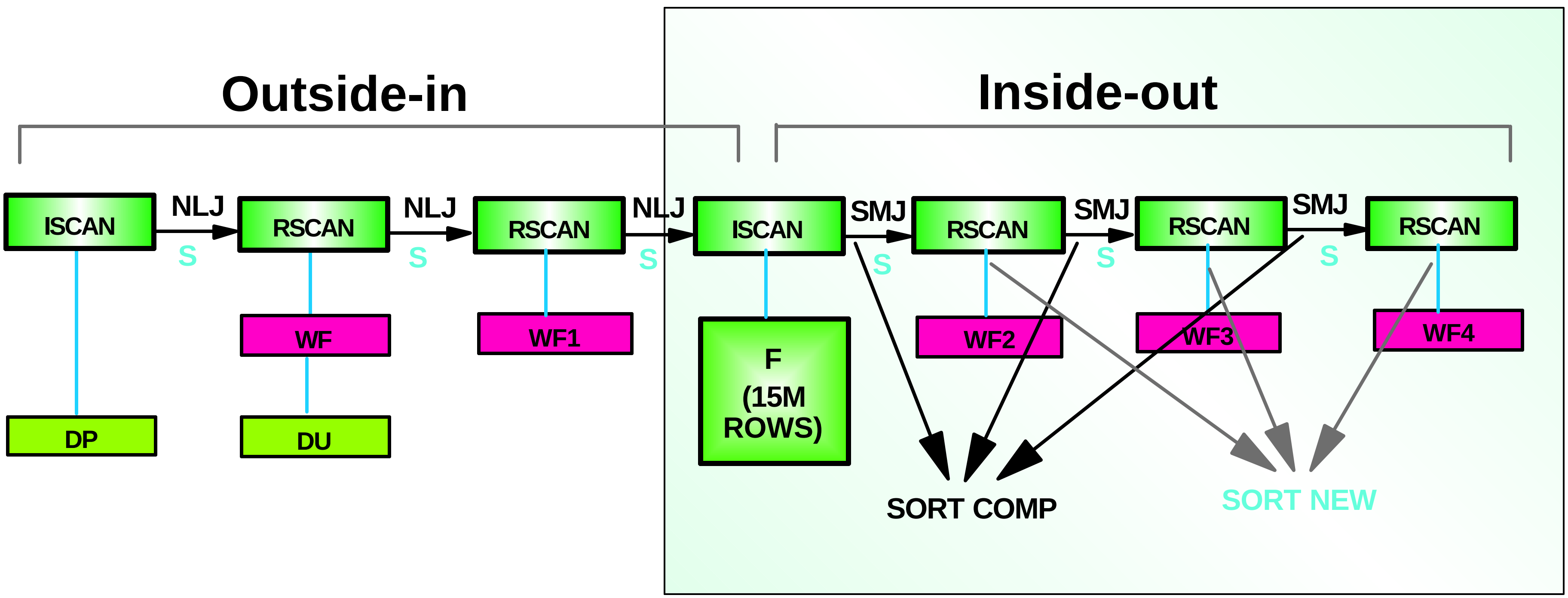
Join/Access Methods in the Outside-In Join Phase

- In-Memory Workfile



Join Methods in the Inside-Out Join Phase

- Inside-Out Join Phase
  - ► Sort merge join (SMJ) or NLJ in the inside-out join phase
  - ► NLJ + sparse index/in-memory workfile for the inside-out join phase is often used to overcome the sort composite problem of SMJ



Join/Access Methods - Summary

- Star join plan - Example #1

S for "pushdown" star join

Query#	Pln#	Corr Name	Table Name	Join Mtd	Join Type	Acc Type	Access Name	Sort New
11001	1	DP	/BI0/D0SD_C01P	0	**S**	I	/BI0/D0SD_C01P~0	**N**
11001	2	DT	/BI0/D0SD_C01T	**1**	**S**	T		**Y**
11001	3	DU	/BI0/D0SD_C01U	**1**	**S**	T		**Y**
11001	4	**F**	/BI0/F0SD_C01	1	**S**	I	/BI0/F0SD_C01~0	**N**
11001	5	D5	/BI0/D0SD_C015	1		I	/BI0/D0SD_C015~0	**N**
11001	6	D3	/BI0/D0SD_C013	1		I	/BI0/D0SD_C013~0	**N**
11001	7		DSN_DIM_TBLX(02)	**1**		T		**Y**
11001	8	D2	/BI0/D0SD_C012	1		I	/BI0/D0SD_C012~0	**N**

Access_type T indicates either "sparse index" or "data caching (V8)" is used.
(The final decision is done at runtime and can not be shown by EXPLAIN.)

Query blocks for snowflakes are not shown.

Parallelism for Star Join Queries

- General concepts
  - **Parallelism plan**
    - **Based on the optimal sequential plan, optimizer determines what operations can be done in parallel**

Sequential Plan

OP1 --> OP2 --> .... OPi --> ... OPj --> ... --> .... OPn

Parallel Plan

OP1 --> OP2 --> .... OPi --> ... OPj --> ... --> .... OPn

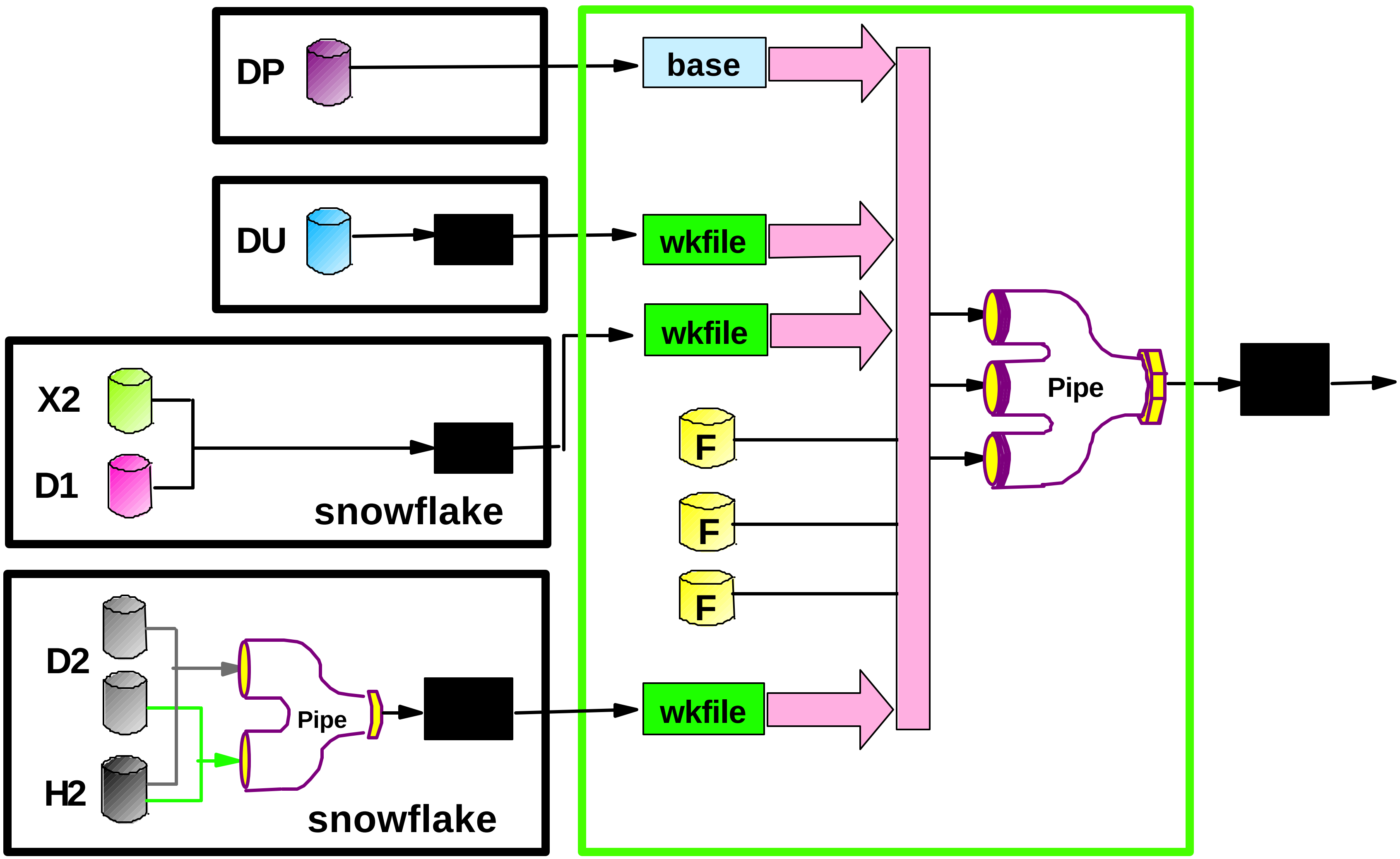
parallel **sequential** **parallel** **sequential** **parallel**

Parallelism for Star Join Queries

- Each Snowflake is handled independently and has its own parallel groups

- Outside- In Join Phase
  - The entire Ouside-In Join (push down join) is one single group
  - Partitioning on the Fact Table
  -
- Inside-Out Join Phases
  - Treated as regular NLJ and SMJ
  - One or more Parallel Groups according to join methods
- Sparse index, In Memory workfile still work

Star Join

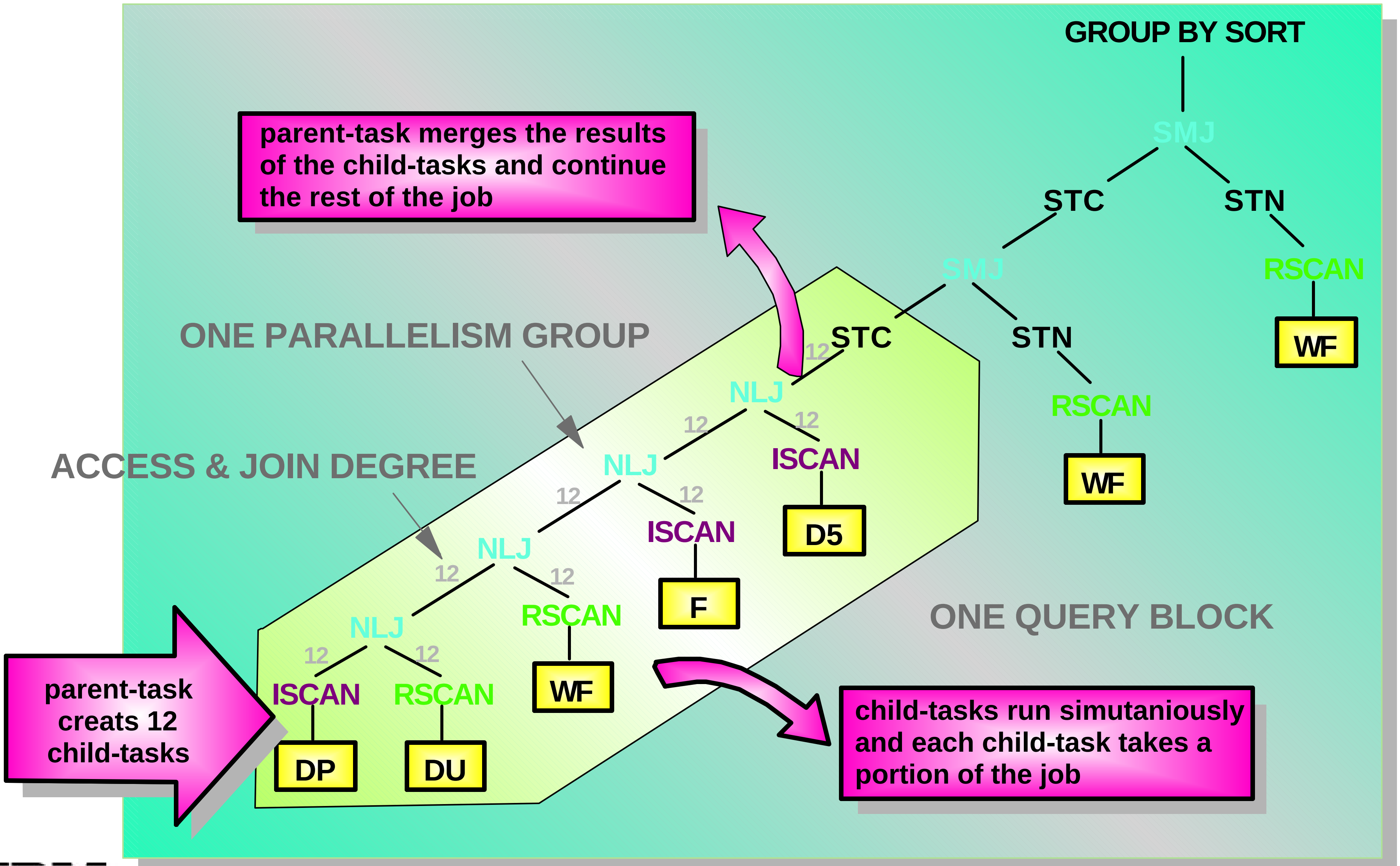

DP
base
DU
wkfile
wkfile
X2
D1
snowflake
F
F
F
Pipe
D2
H2
Pipe
wkfile
snowflake

Star Join

QRY#	BLK#	PLAN#	TNAME	METH	ACC_ TYP	PRE- FETCH	SORT_ NJ	SORT_ CG	ACC_ DEGREE	ACC_ PGROUP	JOIN_ PGROUP	JOIN_ DEGREE
100	1	1	DP	0	I		N	N	3	1	?	?
100	1	2	DU	1	R	S	Y	N	3	1	3	1
100	1	3	WF03	1	R	S	Y	N	3	1	3	1
100	1	4	F	1	I		N	N	3	1	3	1
100	1	5	WF02	1	T		Y	N	3	1	3	1
100	1	6		3			N	Y	?	?	?	?
100	2	1	D2	0	R	S	N	N	2	1	?	?
100	2	2	H2	1	I		N	N	2	1	1	2
100	3	1	X2	0	I	S	N	N	?	?	?	?
100	3	2	D1	2	R	S	Y	N	?	?	?	?

Parallelism for Star Join Queries

- An example of Star Join Parallelism

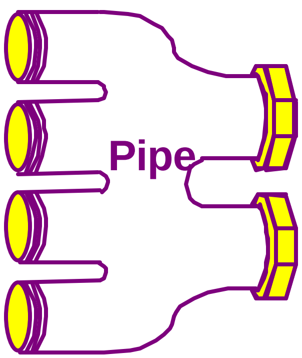


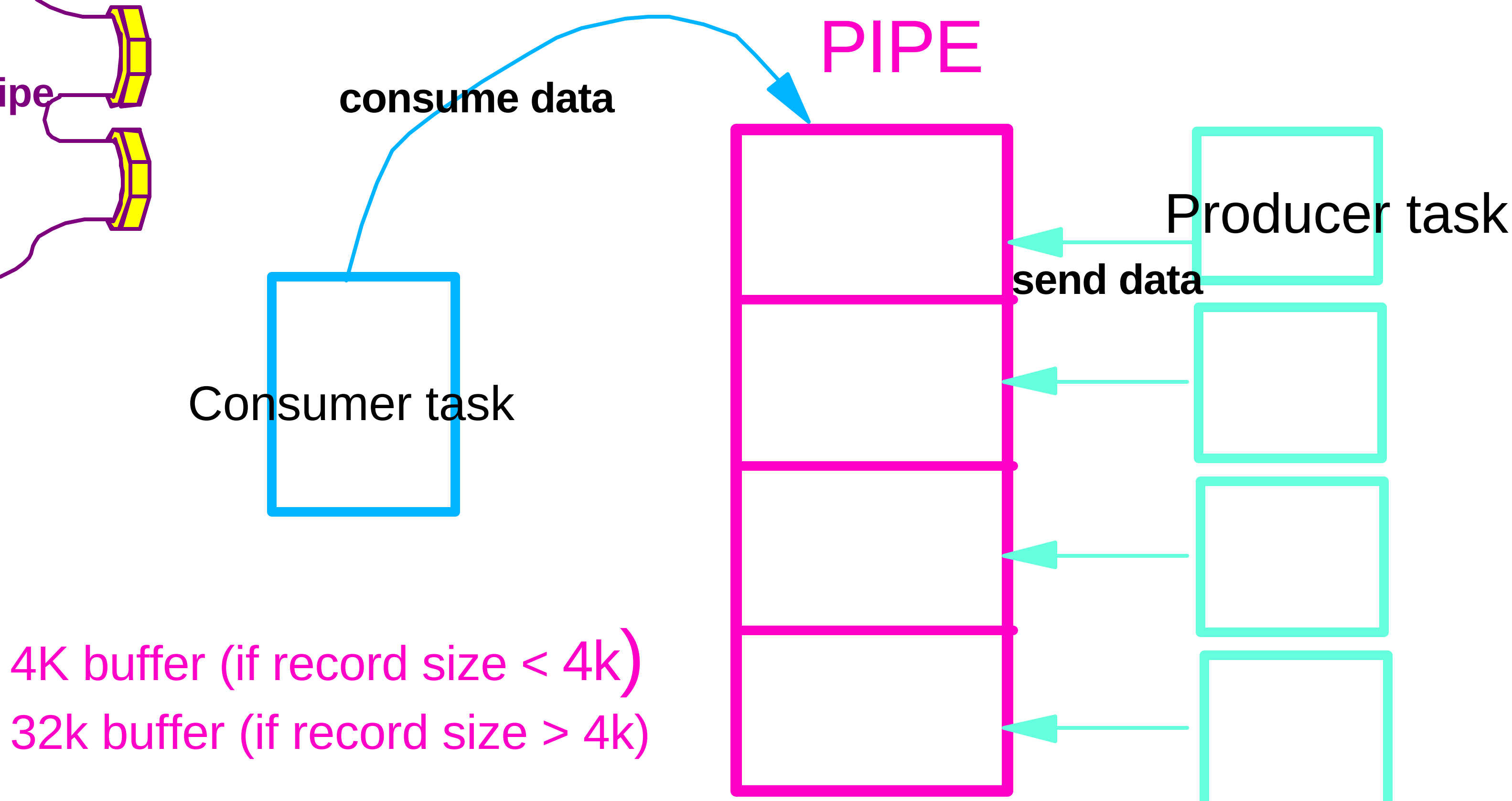
Presentation Outline

- Parallel Sort
- Enable Query Parallelism for Multi-columns Merge Join
- Query Parallelism for DPSI
- Query Parallelism for Star Join
- Future direction

Future Directions

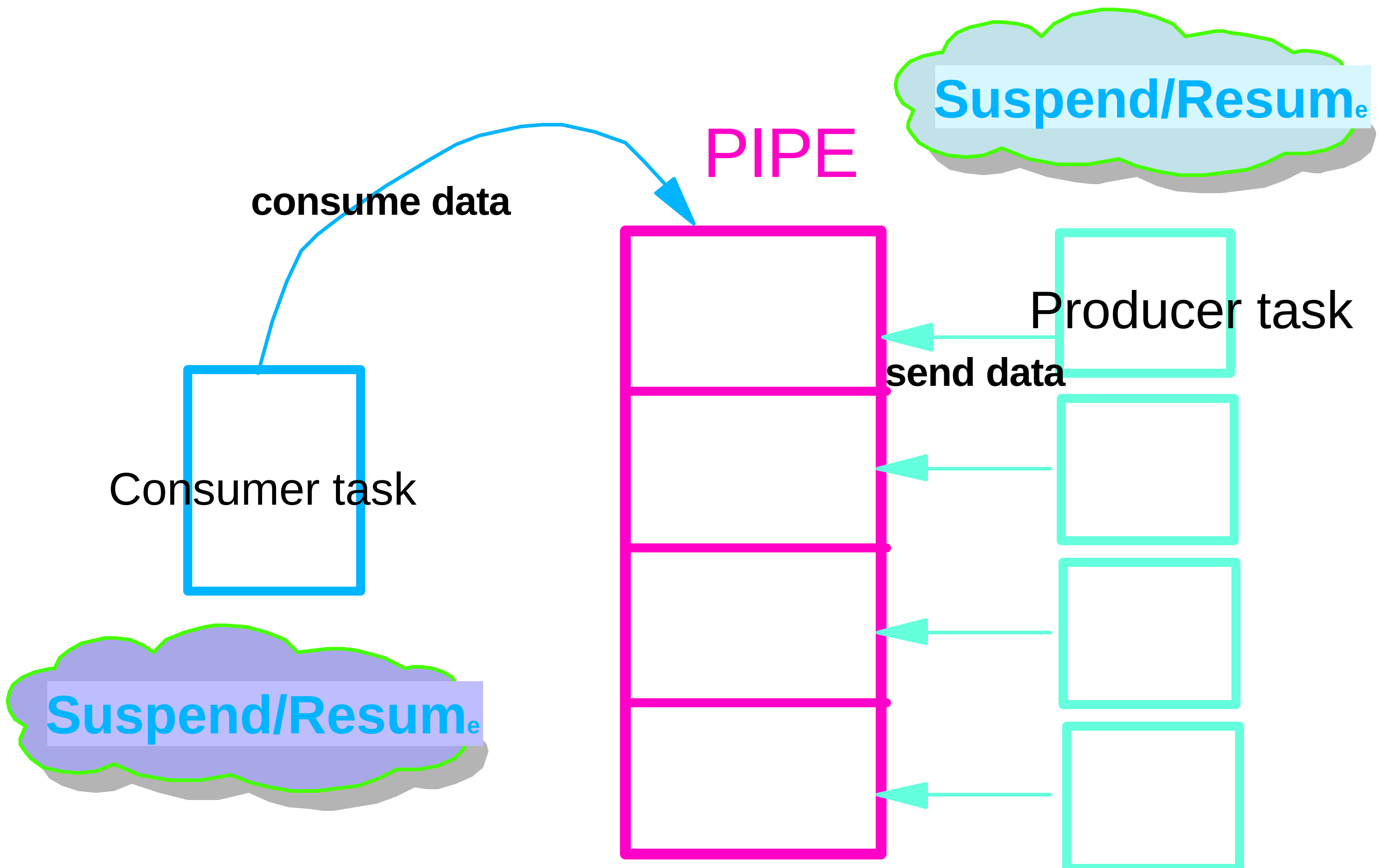
- **Improve the efficiency of data transfer between Producer Tasks (Child Tasks) to Consumer Tasks (Parent Tasks)**
  - **Make the Pipe Buffer relative to the data record size when data is transfered by records**





Future Directions

- Improve the Task synchronization -- Task Suspend/Resume
  - Allow parent task to focus on consuming the data



Future Directions

- **Storage Negotiation**
  - **Control amount of total DB2 storage allowable to support all of running parallel queries.**
  - **Each parallel group will check the system storage to fine tune their degree of parallelism.**
- **A command to monitor the execution of Parallel Queries**
  - **Display the statistics / accounting information and thread information**