Session Z09

New Enhancements for DB2 Star Join Optimization

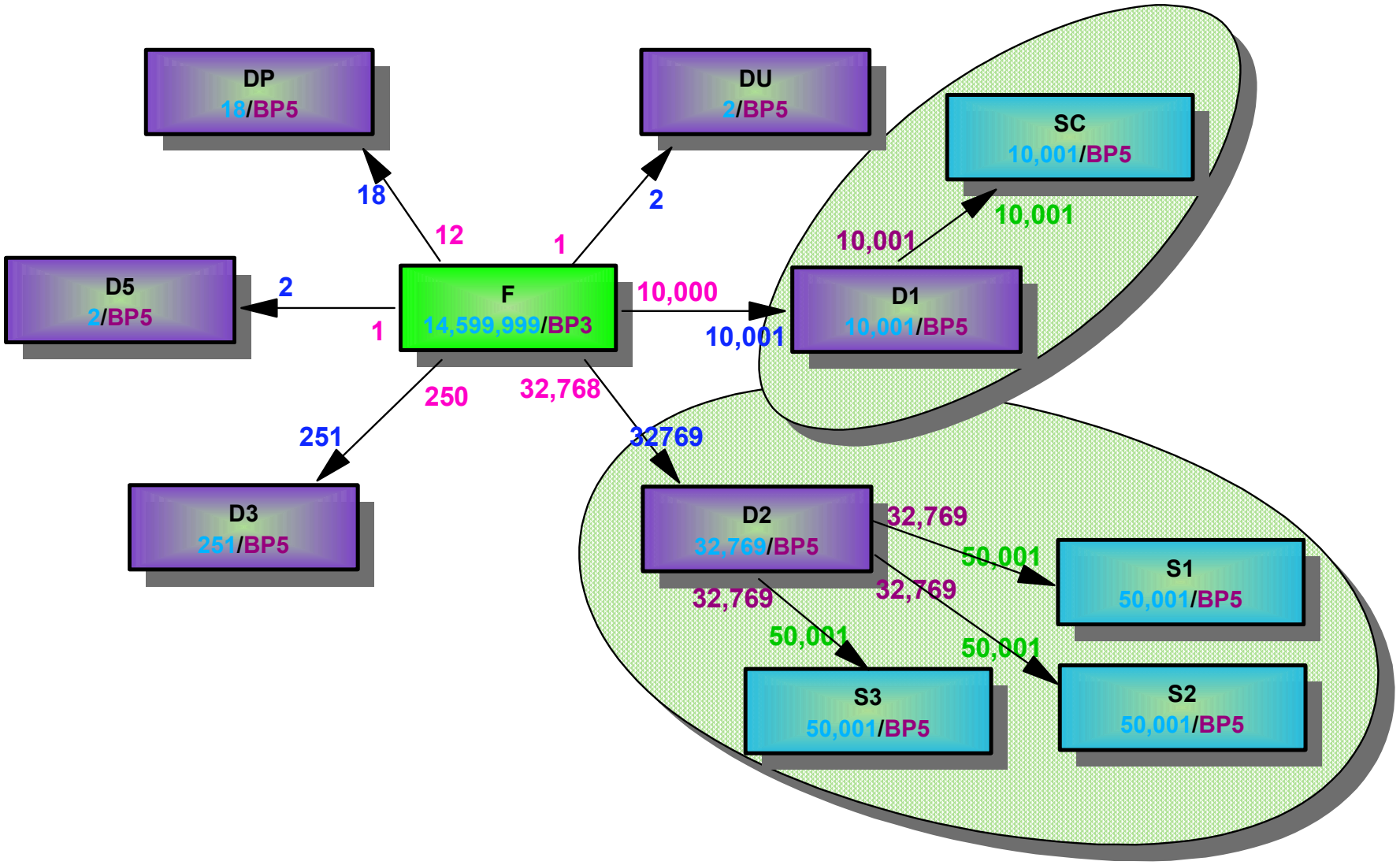
Gene Fuh
IBM Silicon Valley Lab

Anaheim, CA Sept 9 - 13, 2002

Presentation Outline

- Introduction
- New Enhancement
  - Star Schema Detection
  - Efficient Access to Workfile - Phase I
- Future Work
  - Efficient Access to Workfile - Phase II
  - In-Memory Workfile

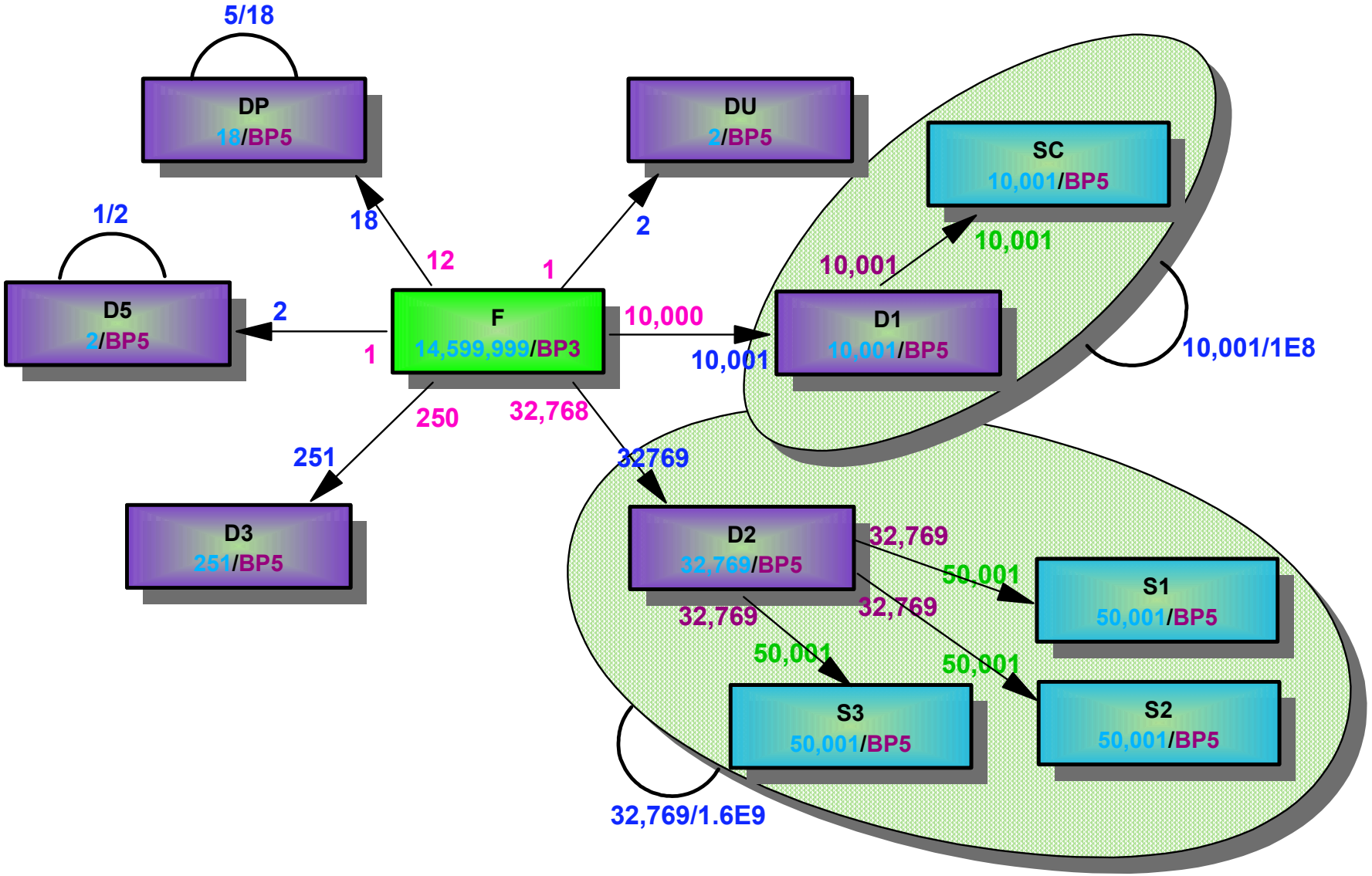
A Sample Star Schema



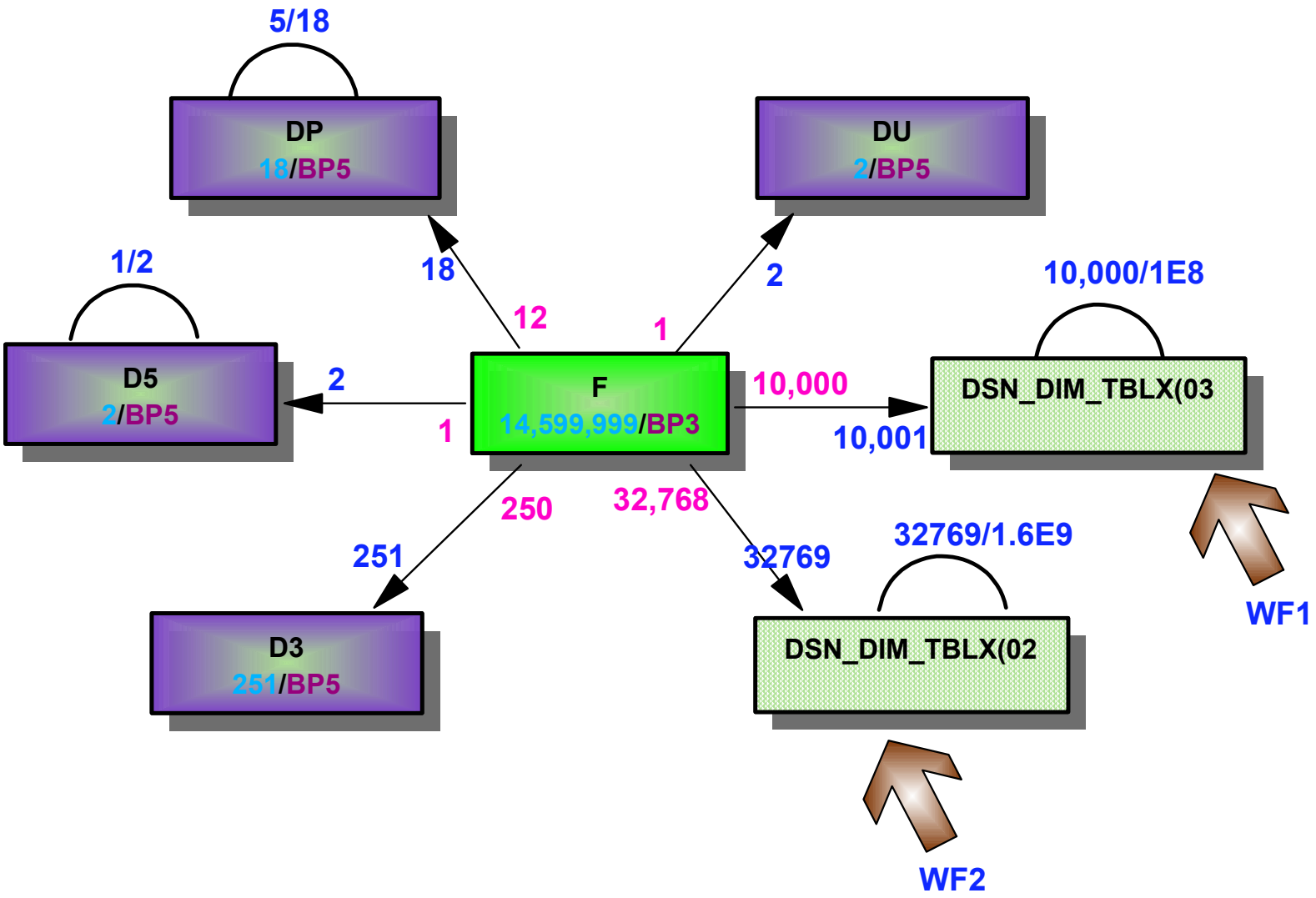
Fact Table Indexes

INDEX NAME	COLUMN SEQUENCE	CLUSTER RATIO	FIRSTKEY CARD	FULLKEY CARD
F~0	PID, TID, UID, 1ID, 2ID, 3ID, 4ID, 5ID	94	12	14,599,999
F~010	PID	99	12	12
F~020	TID	99	365	365
F~030	UID	99	1	1
F~040	1ID	30	10,000	10,000
F~050	2ID	25	32,768	32,768
F~060	3ID	99	1	1
F~070	4ID	99	1	1
F~080	5ID	40	250	250

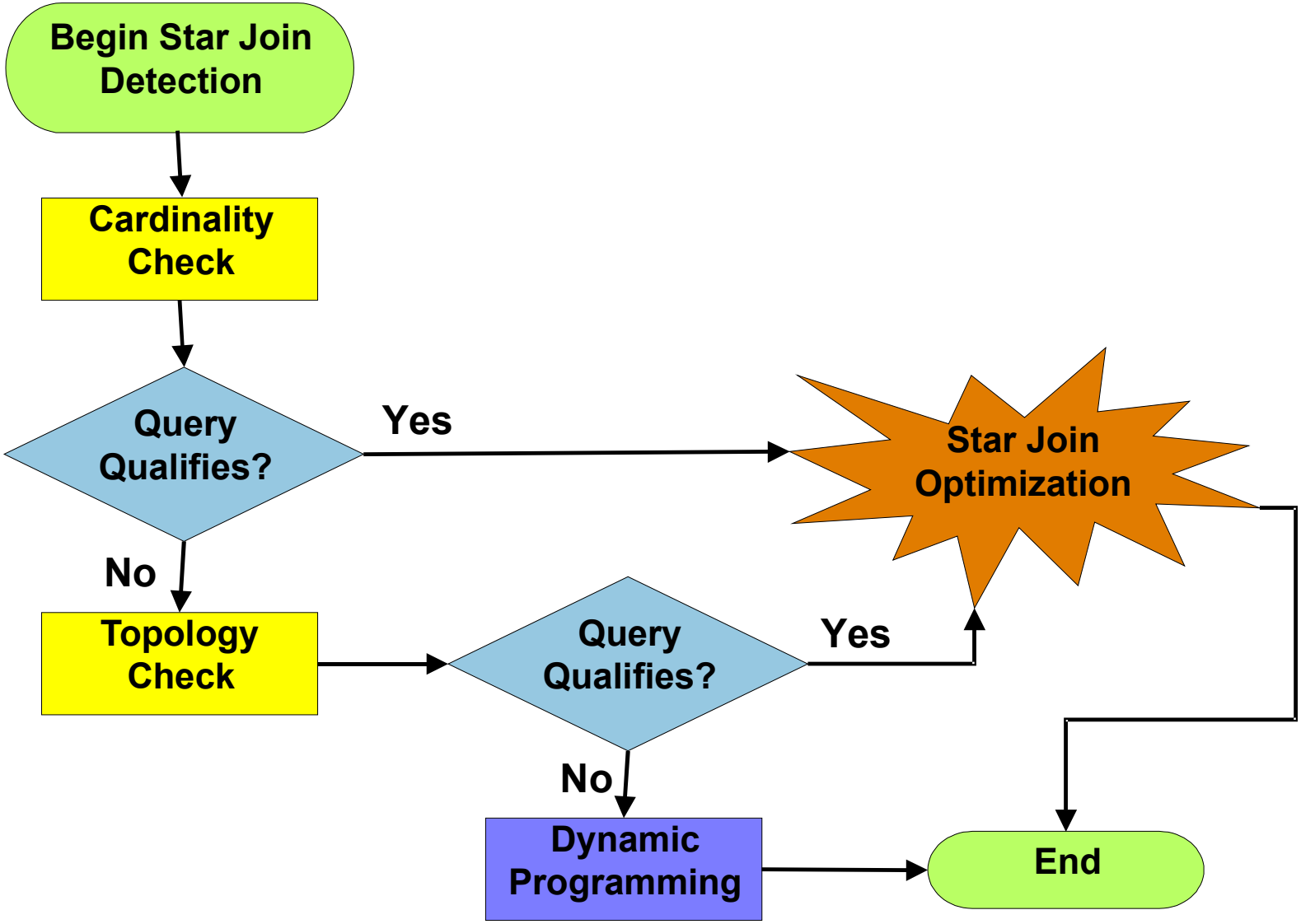
The Join Graph



Star Schema Detection and Transformation



Fact Table Detection



Fact Table Detection

❑ Cardinality checking - by table cardinality

- **STARJOIN zparm**

1. DSJ - Star join disabled (default)
2. ESJ - Star join enabled
3. 1 - Largest table is the fact table
4. 2-32768 - Star join enabled if the largest table is at least n times of the second largest table

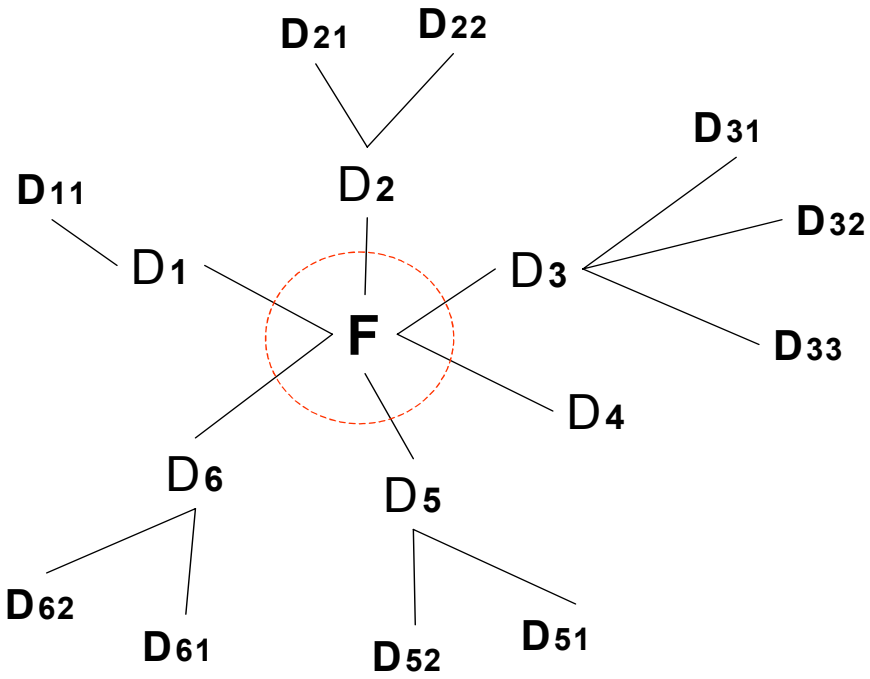
- **Issues**

1. Could have very large dimension tables
2. Difficult to pick the right value for STARJOIN zparm

Fact Table Detection

Topology Checking - by table connectivity

- The most "connected" table



- Issue - dimension tables may be more "connected"

Star Schema Detection

Global constraints

- At least 10 tables in the query

  Altered via zparm SJTABLES

- No outer join between the fact table and dimensions (after performing outer join reduction)

Star Schema Detection

Constraints on dimensions

- No correlated subqueries between any two dimensions
- No join predicates between any two dimensions
- A dimension local predicate cannot be "OR'd" with a local predicate from another dimension
- Can not be a table function

Star Schema Detection

Constraints on predicates

- Fact table and each dimension table are joined through a equal predicate
- Both sides of the join predicates is a column reference
- Both sides of the join predicates has the same type and the same length
- The same fact table column can not join to more than one dimension table
- All join predicates between the fact and dimension tables must be Boolean term

Index-Directed Join Permutation

Why special join permutation ?

- Too many join seqences in general

  7! = 5,040 10! = 3,628,800

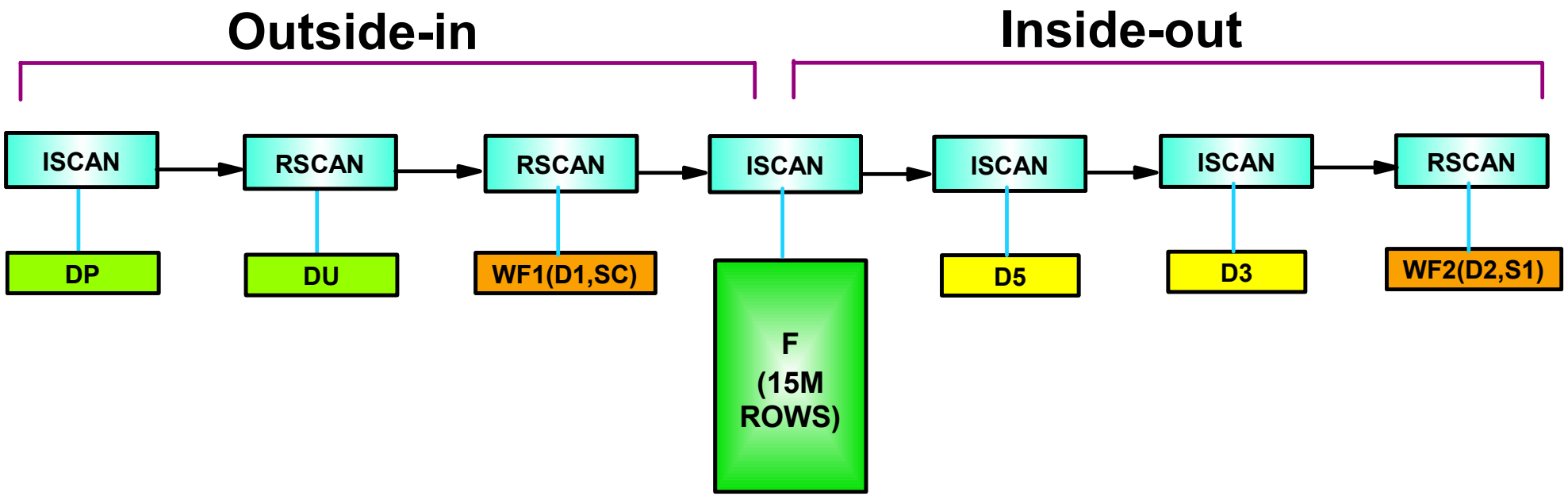
- ISCAN on fact table is preferrable

- Therefore, viable join sequences can be derived from the "prefix" of the columns of fact table indexes

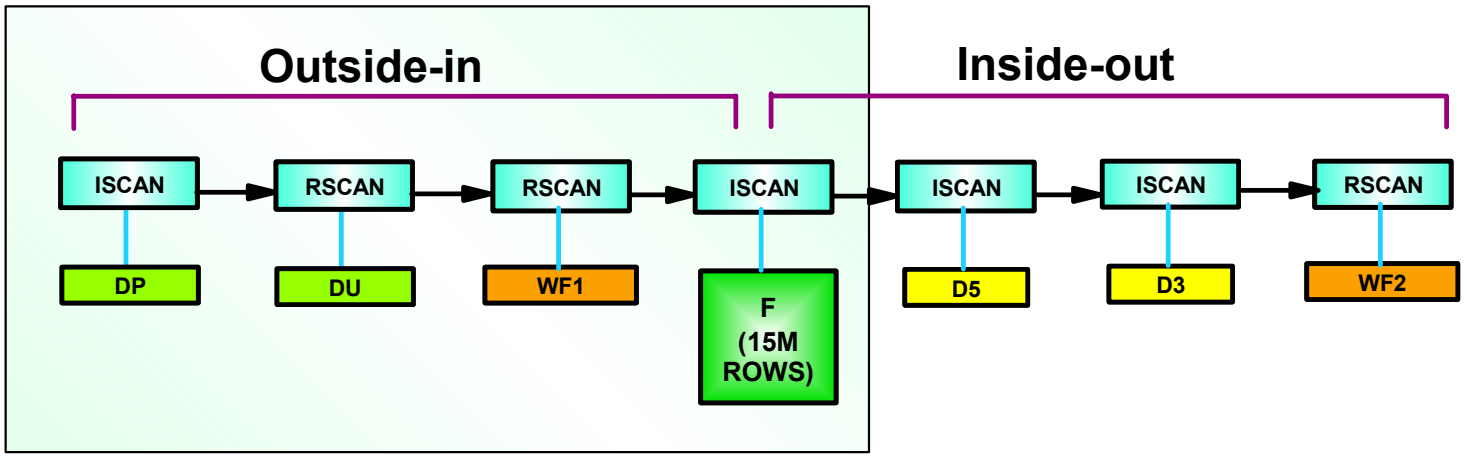
Access Plan

QUERYNOS	QBLOCK NO	PLAN NO	TNAME	MET HOD	ACCESS NAME	AT	MATCH COLS	N_U	N_J	JNTYPE	COR RNM
62713130	1	1	DP	0	DP~0	I	1	N	N	S	DP
62713130	1	2	DU	1		R	0	N	Y	S	DU
62713130	1	3	DSN_DIM_TBLX(03	1		R	0	N	Y	S	WF1
62713130	1	4	F	1	F~0	I	3	N	N	S	F
62713130	1	5	D5	1	D5~0	I	1	N	N		D5
62713130	1	6	D3	1	D3~0	I	1	N	N		D3
62713130	1	7	DSN_DIM_TBLX(02	1		R	0	N	Y		WF2
62713130	2	1	D2	0		R	0	N	N		D2
62713130	2	2	S1	1	S1~1	I	1	N	N		S1
62713130	3	1	SC	0	SC~0	I	0	N	N		SC
62713130	3	2	D1	2		R	0	N	Y		D1

Join Sequence



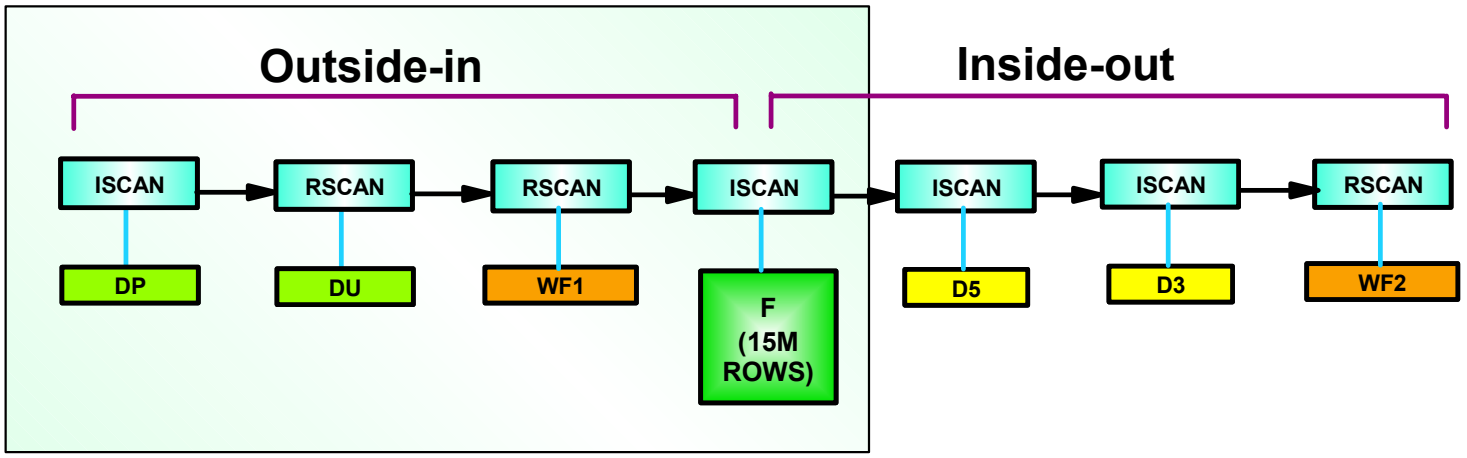
Index Key Feedback (Index Skipping)



- Cartesian Join before the fact table

1. No join predicates between dimensions --> Cartesian join
2. Dimesion tables are highly correlated --> many combinations do not exist
3. Excessive Cartesian-join leads to excessive ISCANs of the fact table

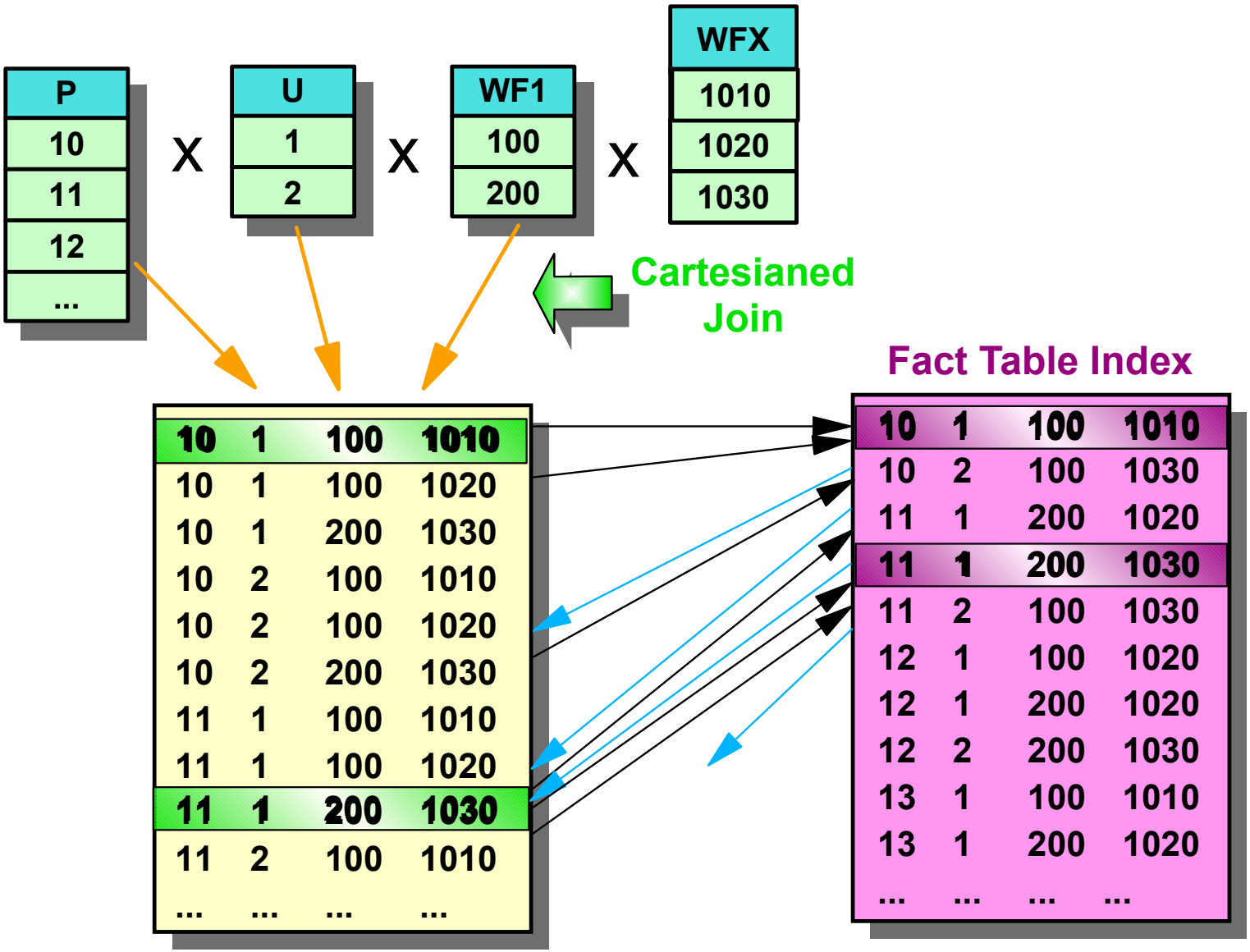
Index Key Feedback (Index Skipping)



The resolution - index key feedback

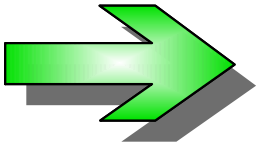
1. If a hit on the fact table index then business as usual
2. Otherwise the next valid index key feedback to the composite
3. The composite then skips to the next candidate key and repeat step 1-3
4. Significant skipping in the Cartesian join was expected *BUT ...*

Index Key Feedback (Index Skipping)



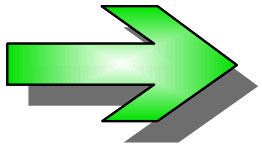
A Sample Star Join Query

```
SELECT  SUM(DU.BASE_UOM), SUM(D3.DISTR_CHAN), SUM(D3.DIVISION) ,
        SUM(DP.RECORDTP), SUM(SC.COUNTRY), SUM(DU.STAT_CURR),
        SUM(D5.VTYPE), SUM(S1.PRED)
FROM    F, DU, D3, D2, DP, D1, SC, D5, S1
WHERE   F.PID = DP.ID AND
        F.UID = DU.ID AND
        F.1ID  = D1.ID AND
        F.2ID  = D2.ID AND
        F.3ID  = D3.ID AND
        F.5ID  = D5.ID AND
```



JOIN PREDICATES

```
        F.PID" BETWEEN 0 AND 23  AND
        DP.ID BETWEEN 10 AND 14  AND
        DP.CHNGID = 0 AND
        DP.REQUID BETWEEN 0 AND 49  AND
        D1.SOLD_TO = SC.SID AND
        SC.OBJVERS = 'A' AND
        D5.VTYPE" = 10  AND
        D2.MATERIAL = S1.SUCC  AND
        S1.SUCC <> 2000008999  and
        S1.SEQ_NR = 0 ;
```



LOCAL PREDICATES

Index-Directed Join Permutation

The join enumeration algorithm

For each index (C_{i1}, C_{i2}, ..., C_{ik}) on the fact table

For each prefix (C_{i1}, ..., C_{ij}) of the index keys, where ij <= ik

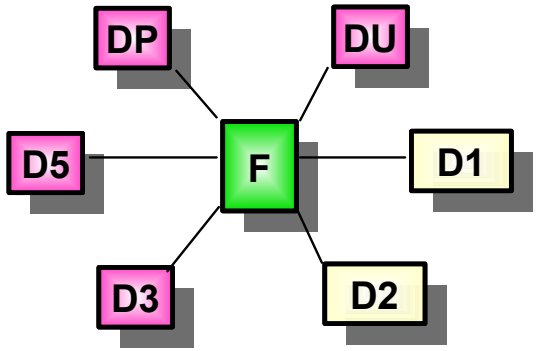
- **Form the (partial) join sequence D_{i1} --> D_{i2} --> ... --> D_{ij} --> Fact**
- **Complete the join sequence based on the following rules:**

1. **Remaining dimensions joined in the order of selectivity**
    1. **Based on selectivity only**
    2. **Indexed dimensions (on the join columns) first**
2. **Remaining dimensions joined in the order of the estimated cardinality**
    1. **Based on cardinality only**
    2. **Indexed dimensions (on the join columns) first**

Index-Directed Join Permutation

An example

INDEX NAME	COLUMN SEQUENCE	PRE-FACTJOIN SEQUENCES	# OF JOIN SEQUENCES
F~0	PID, TID, UID, 1ID, 2ID, 3ID, 4ID, 5ID	DP-F DP-DU-F DP-DU-D1-F DP-DU-D1-D2-F DP-DU-D1-D2-D3-F DP-DU-D1-D2-D3-D5-F	up to 4 up to 4 up to 4 up to 2 1 1
F~010	PID	DP-F	up to4
F~020	TID		
F~030	UID	DU-F	up to 4
F~040	1ID	WF1-F	up to 4
F~050	2ID	WF2-F	up to 4
F~060	3ID	D3-F	up to 4
F~070	4ID		
F~080	5ID	D5-F	up to 4



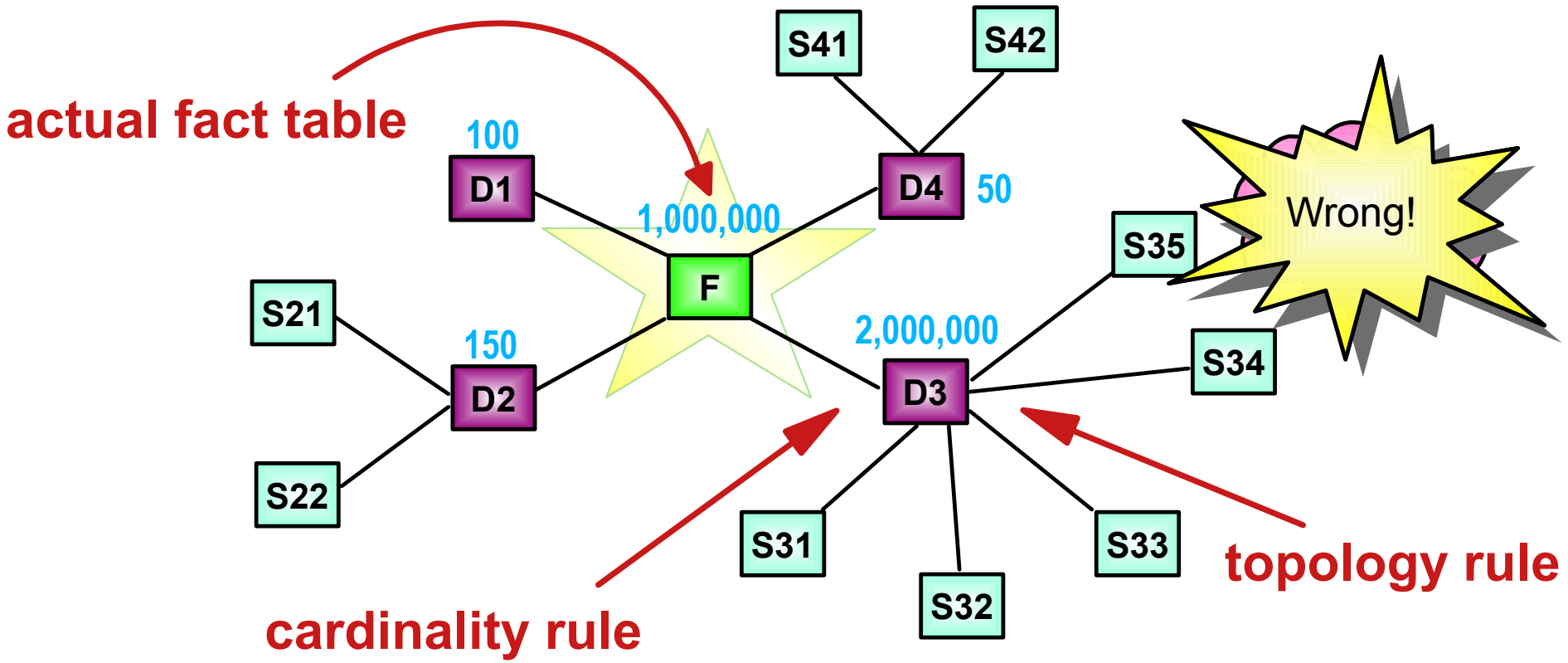
Total: Up to 40 join sequences !

APARs

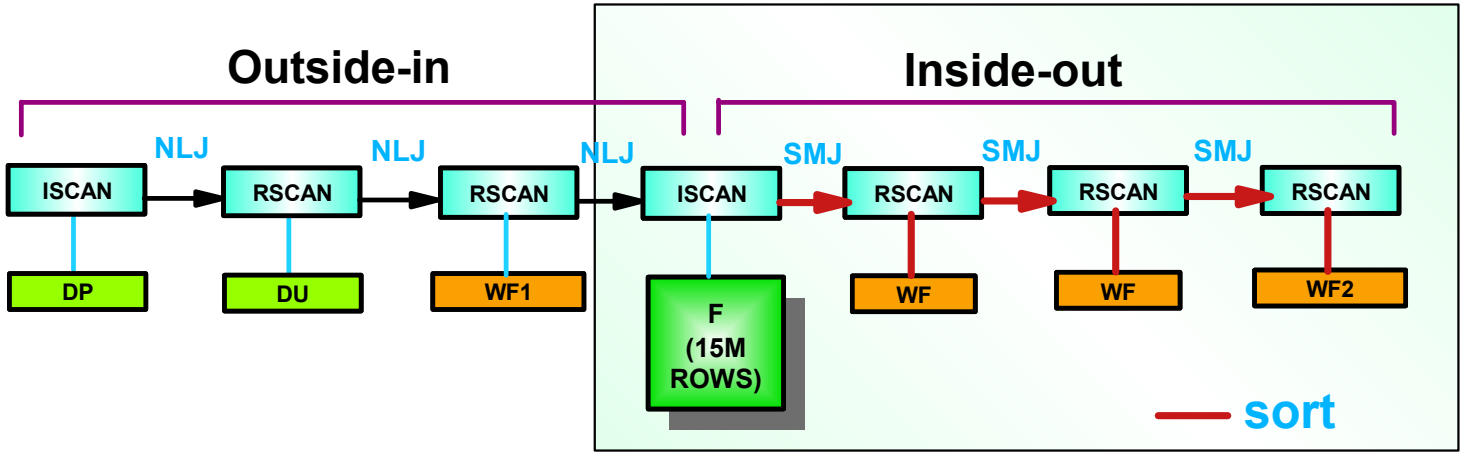
APAR NO.	PTF NO.	DESCRIPTION	DATE	DB2 RELEASE
PQ43846	UQ53875	INDEX DRIVE JOIN PERMUTATION	May 2001	V6
PQ47833	UQ57153	INDEX DRIVE JOIN PERMUTATION	May 2001	V7

Outstanding Issues - star schema detection

- **Fact table detection**
  1. Cardinality checking
  2. Topology checking



Outstanding Issues - workfile access



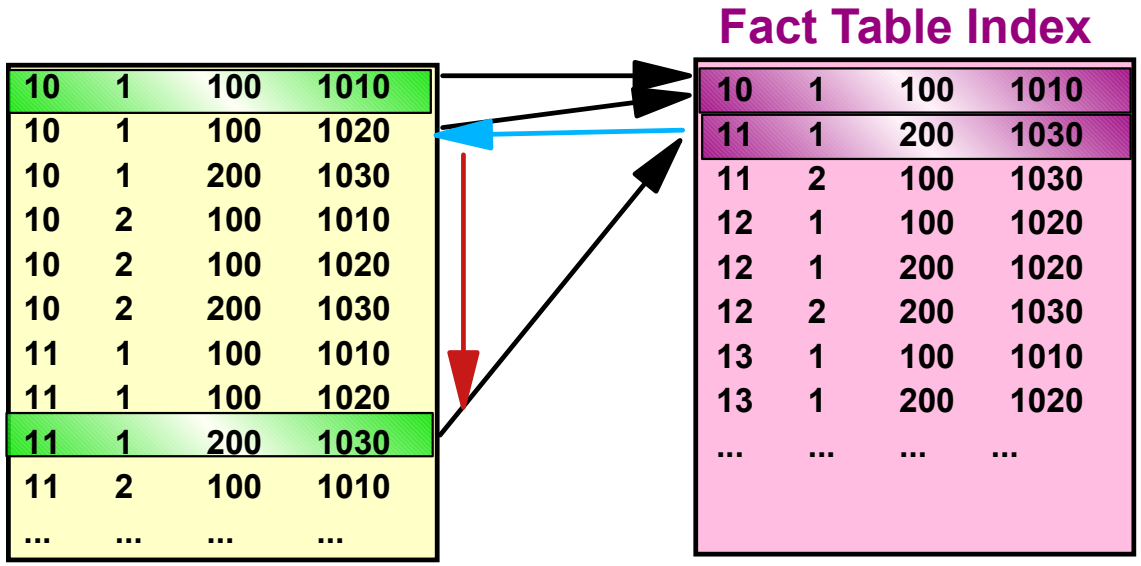
- The lack of index on workfiles leads to SMJ

1. Increased workfile space consumption
2. Excessive CPU and I/O consumption
3. Increased parallelism overhead due to merge/fork
4. Critical storage/performance issue for large intermediate result

Outstanding Issues - reposition overhead

- **Skipping the index keys is not free**

1. Dimensions workfiles are sorted in the join column order
2. Skipping index keys in a particular dimension requires sequential scan of the workfile
3. Therefore skipping of "inner" dimensions may be expensive



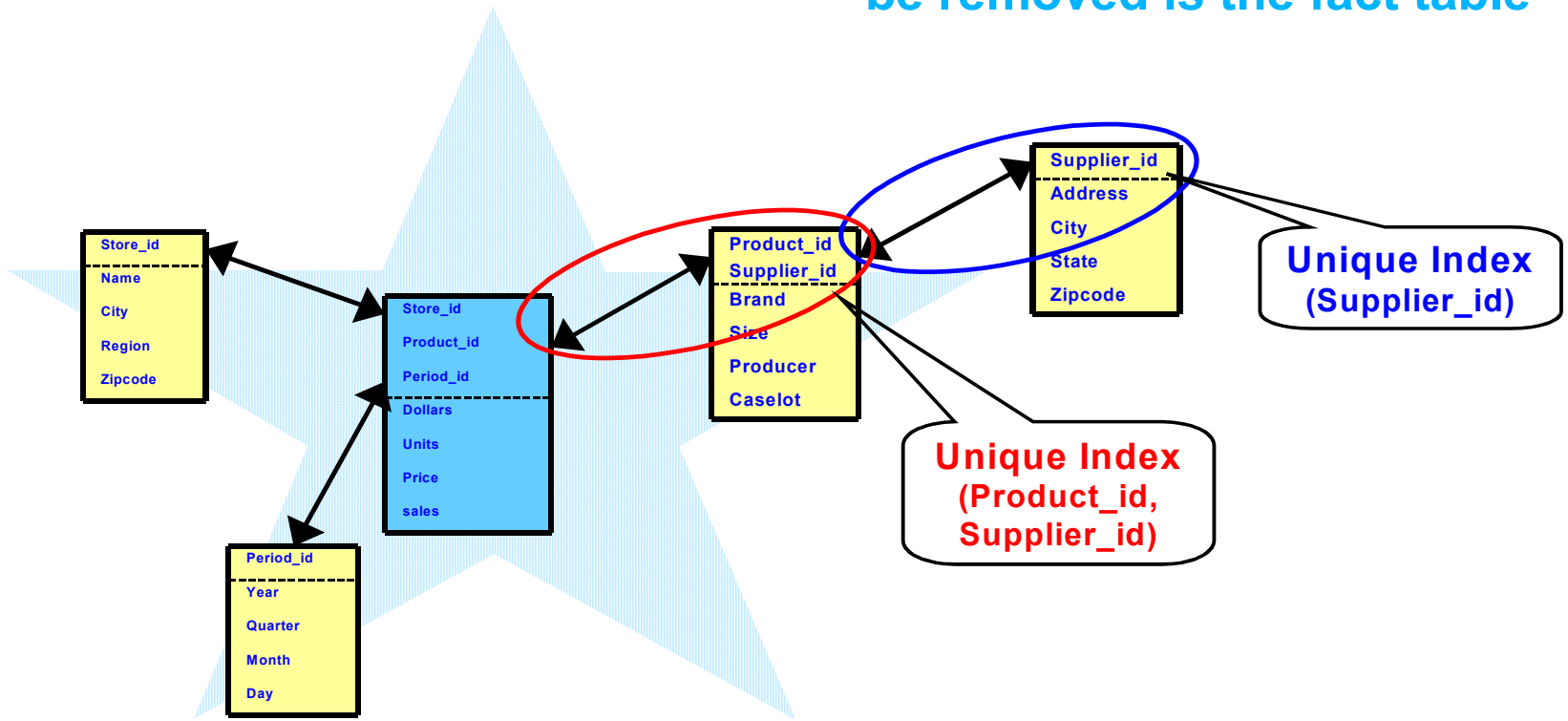
Presentation Outline

- Introduction
- New Enhancement
  - Star Schema Detection
  - Efficient Access to Workfile - Phase I
- Future Work
  - Efficient Access to Workfile - Phase II
  - In-Memory Workfile

Fact Table Detection

Unique Index Check

- Terminal tables
  - Join only to one table
  - Join column(s) is/are unique
- Repeatedly removing terminal tables
- The only table can not be removed is the fact table

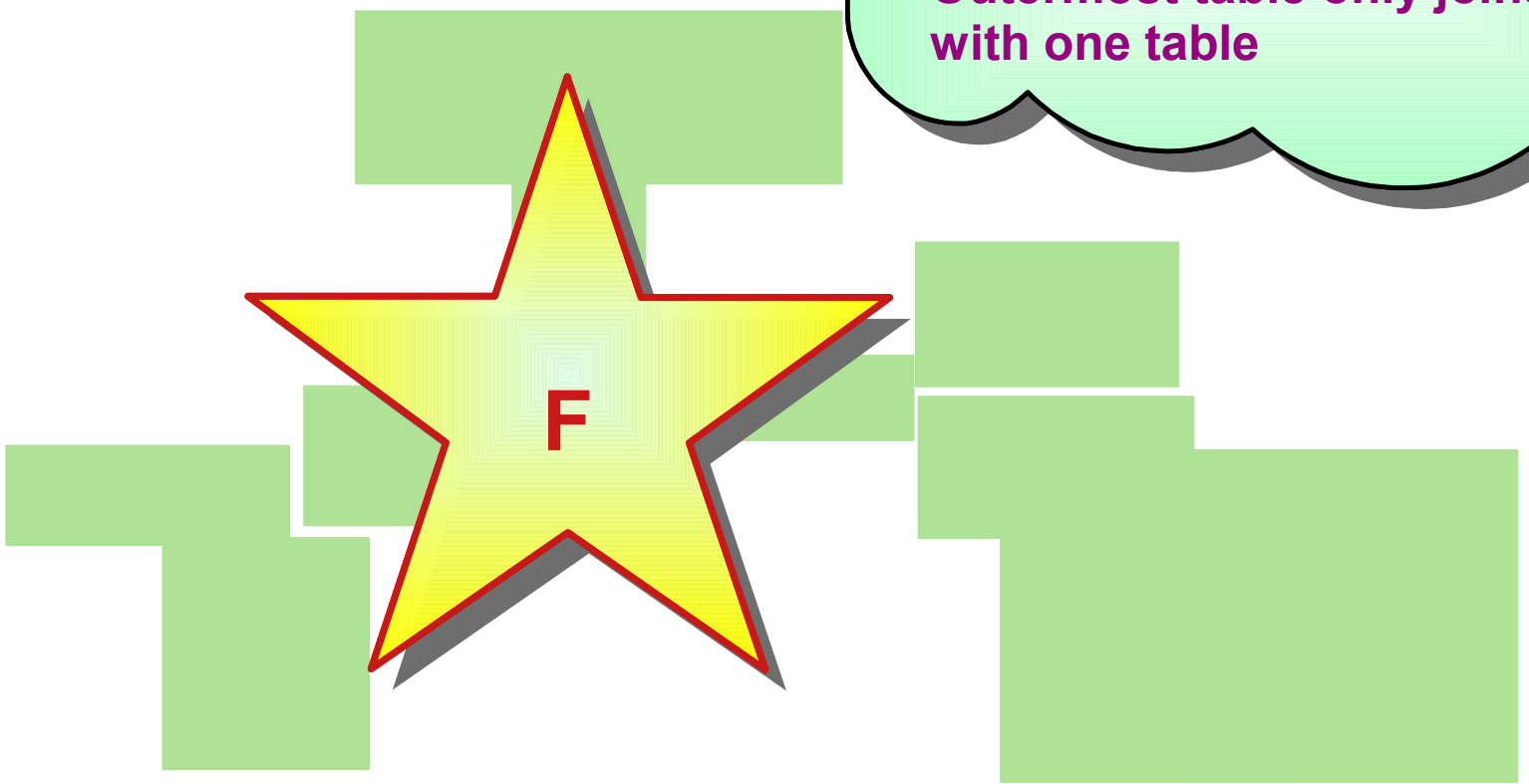


Fact Table Detection

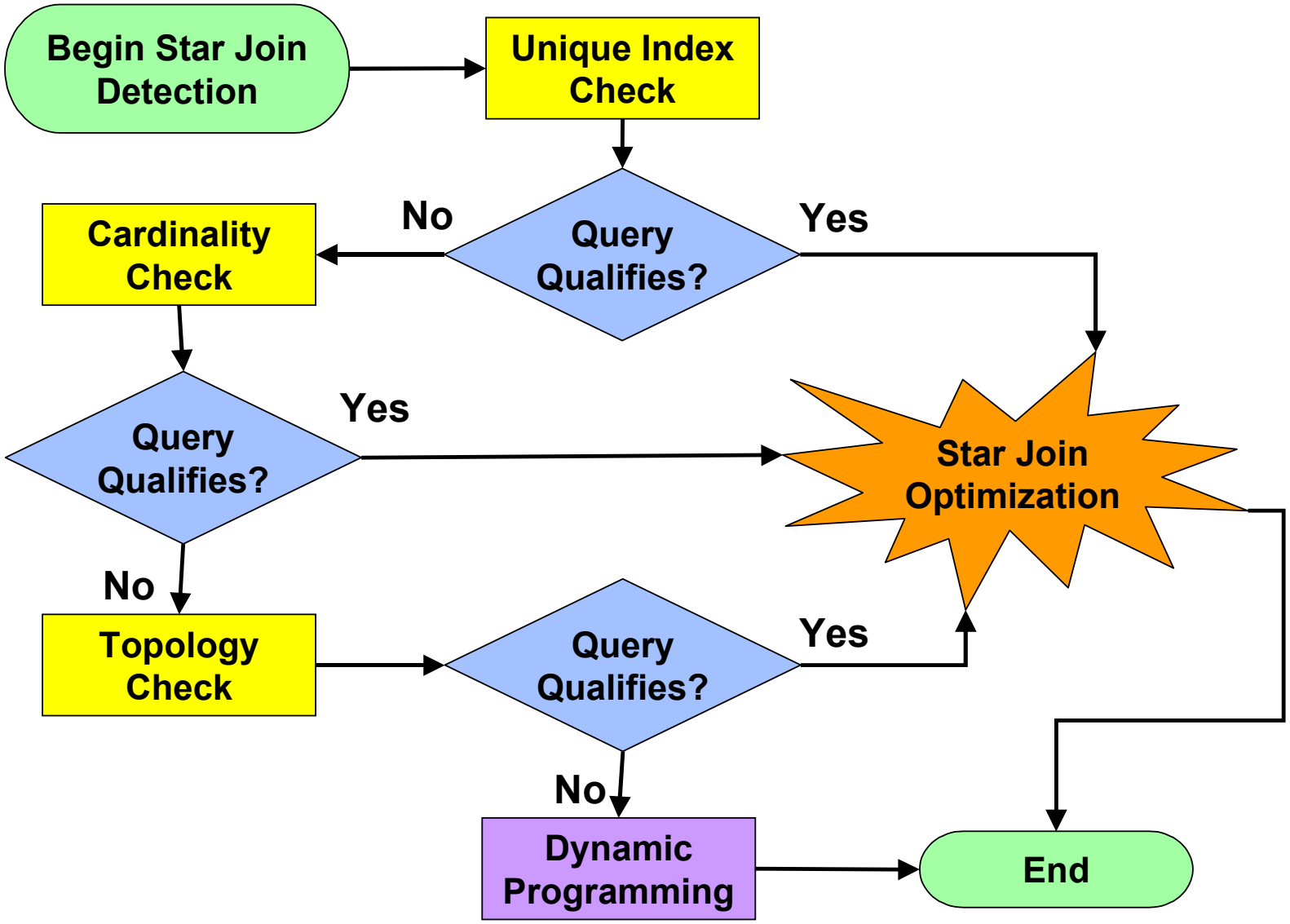
- Example

Outermost table has unique index on the join column

Outermost table only joins with one table



Procedure of Detecting Star Schema



APAR for the Unique Index Check

APAR NO.	PTF NO.	DESCRIPTION	DATE	DB2 RELEASE
PQ49925	UQ60214	Unique index check for FACT table detection	Dec. 2001	V6
PQ49925	UQ60215	Unique index check for FACT table detection	Dec. 2001	V7

Presentation Outline

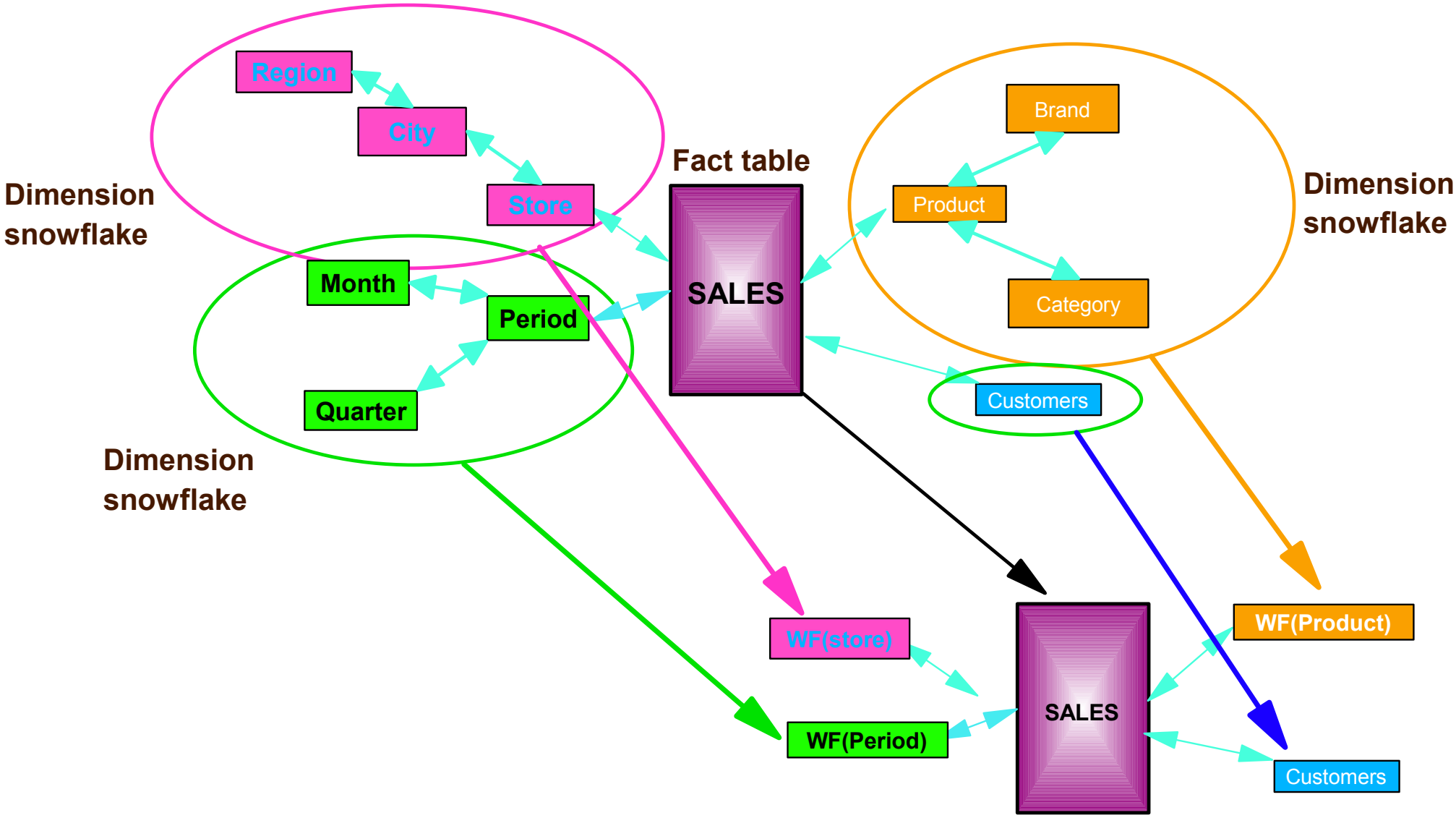
- Introduction
- New Enhancement
  - Star Schema Detection
  - Efficient Access to Workfile - Phase I
- Future Work
  - Efficient Access to Workfile - Phase II
  - In-Memory Workfile

Index on Workfile - Phase I

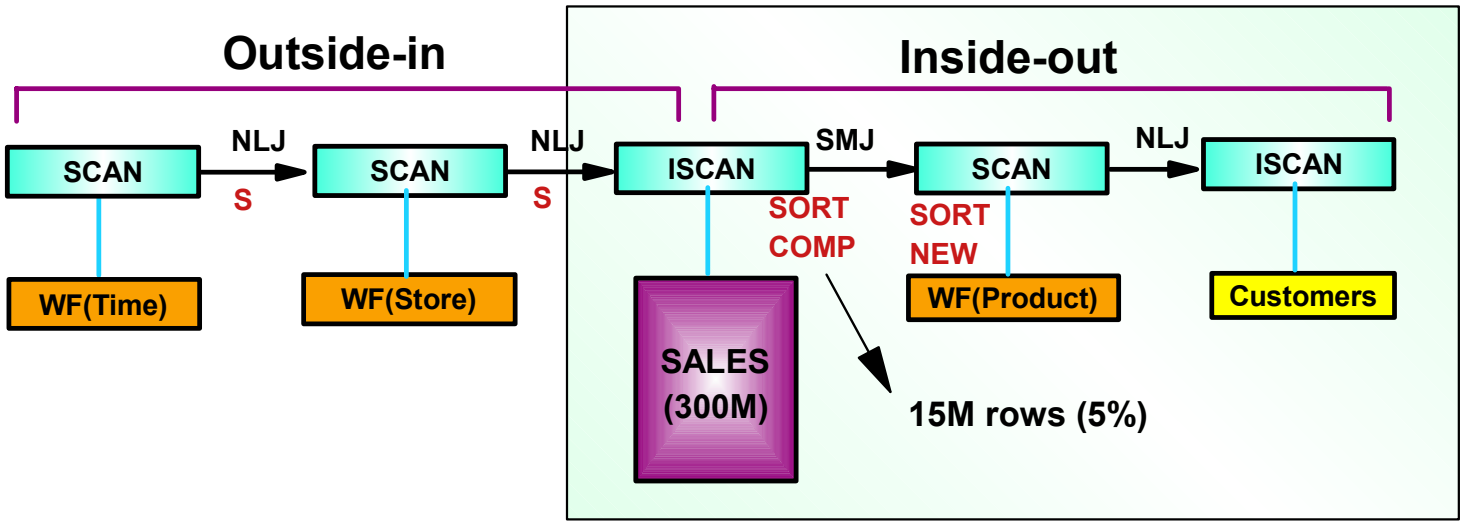
- **Characteristics of workfiles in star join**
  - Materialization of snowflakes is common for star schema queries.
  - In the context of star join, workfiles are typically small (hundreds or thousands of records).
  - Typical data warehouse queries touch significant portion of the fact table, which results in large composite result set in the "inside-out phase".

Index on Workfile - Phase I

Materialization of snowflakes



Index on Workfile - Phase I



- **Performance Issue**
  - **The lack of index on workfiles leads to SMJ**
    1. **Increased workfile space consumption**
    2. **Excessive CPU and I/O consumption**
    3. **Increased parallelism overhead due to merge/fork**
    4. **Critical storage/performance issue for large intermediate result**

Index on Workfile - Phase I

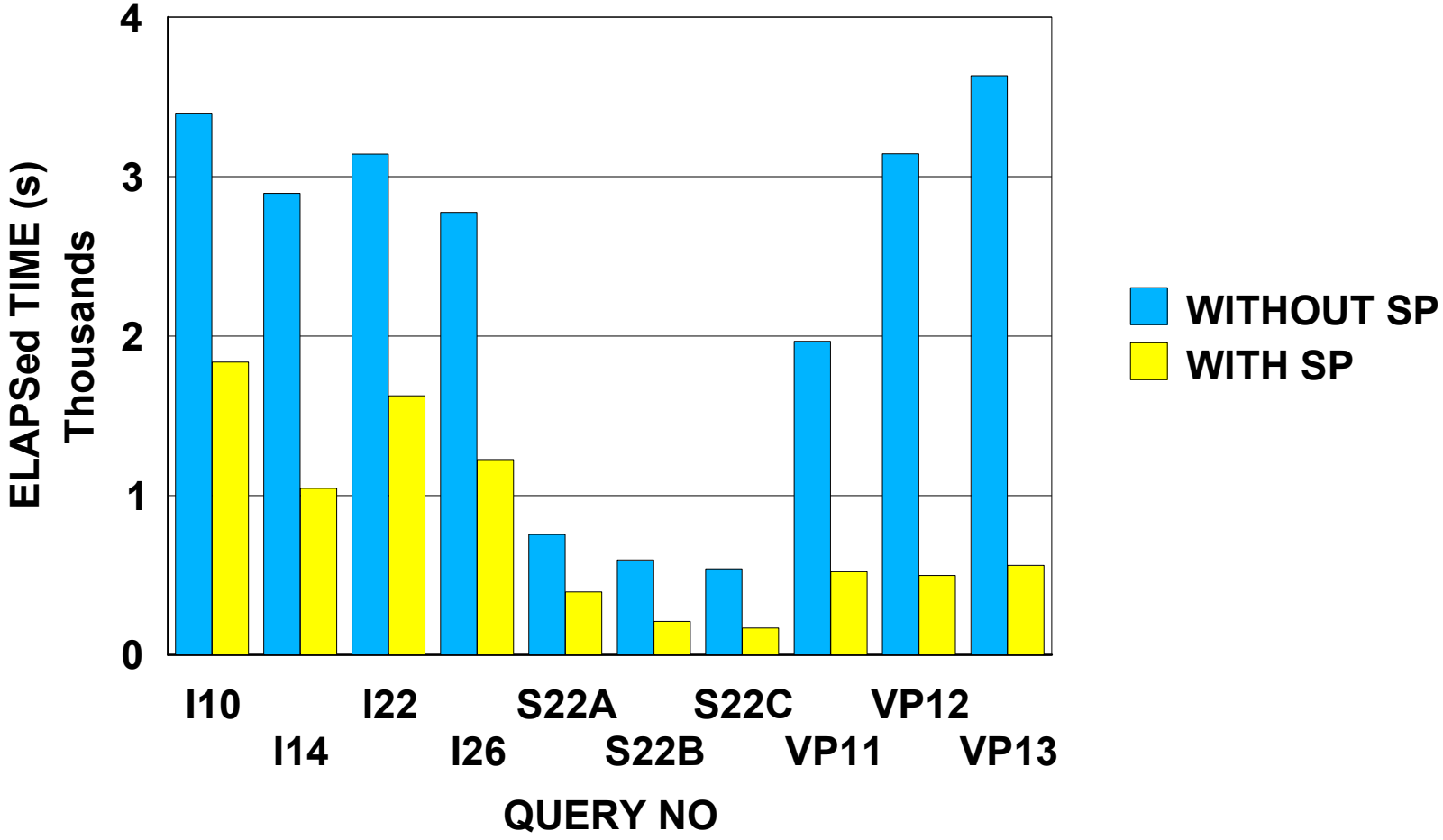
- **DB2 solution - Using sparse index**
  - In-memory array ordered in the join column.
  - Probed through an equal-join predicate.
  - Binary search for the target segment based on the value of the join column.
  - Sequential search within the target segment as needed.
  - The "denser" the faster - in favor of small workfiles.
  - More beneficial for large join composite.
  - Ideal solution for the access of dimension workfiles.

Index on Workfile - Phase I

- **Performance benefit - NLJ instead of SMJ**
  - No sort of large composite - CPU & I/O reduction
  - Reduction of workfile space
  - Reduction of parallelism overhead (merge & repartition)

Index on Workfile - Phase I

- Performance chart



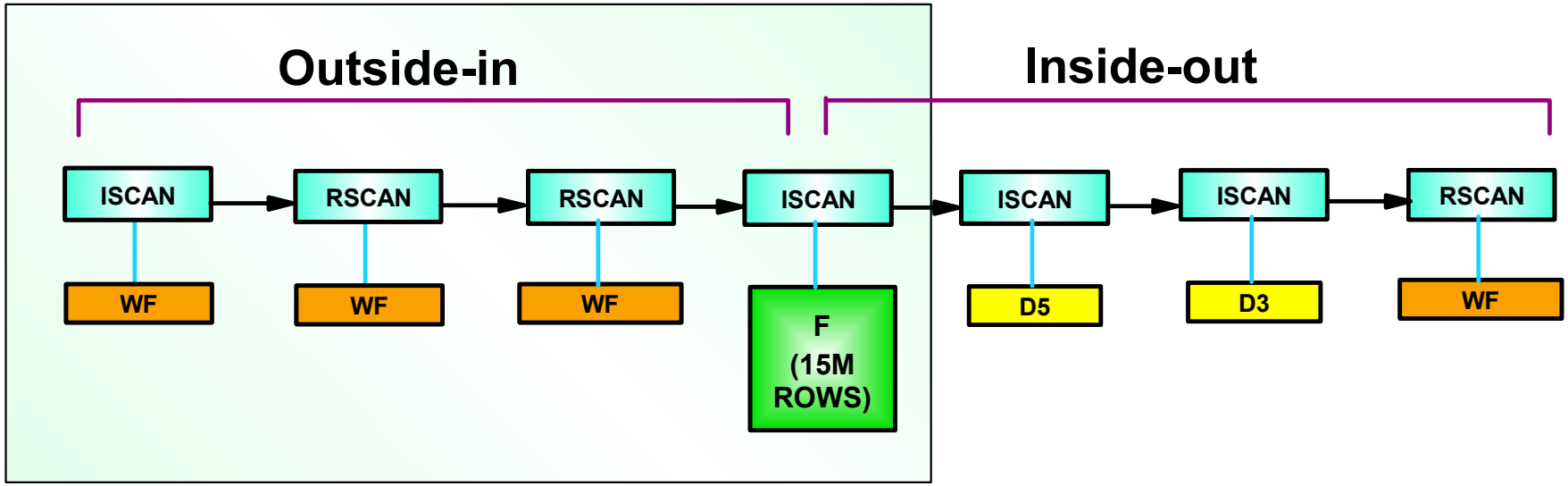
APAR for Index on Workfile - Phase I

APAR NO.	PTF NO.	DESCRIPTION	DATE	DB2 RELEASE
PQ61458	UQ67433	Sparse index for the access of workfiles	July 2002	V7

Presentation Outline

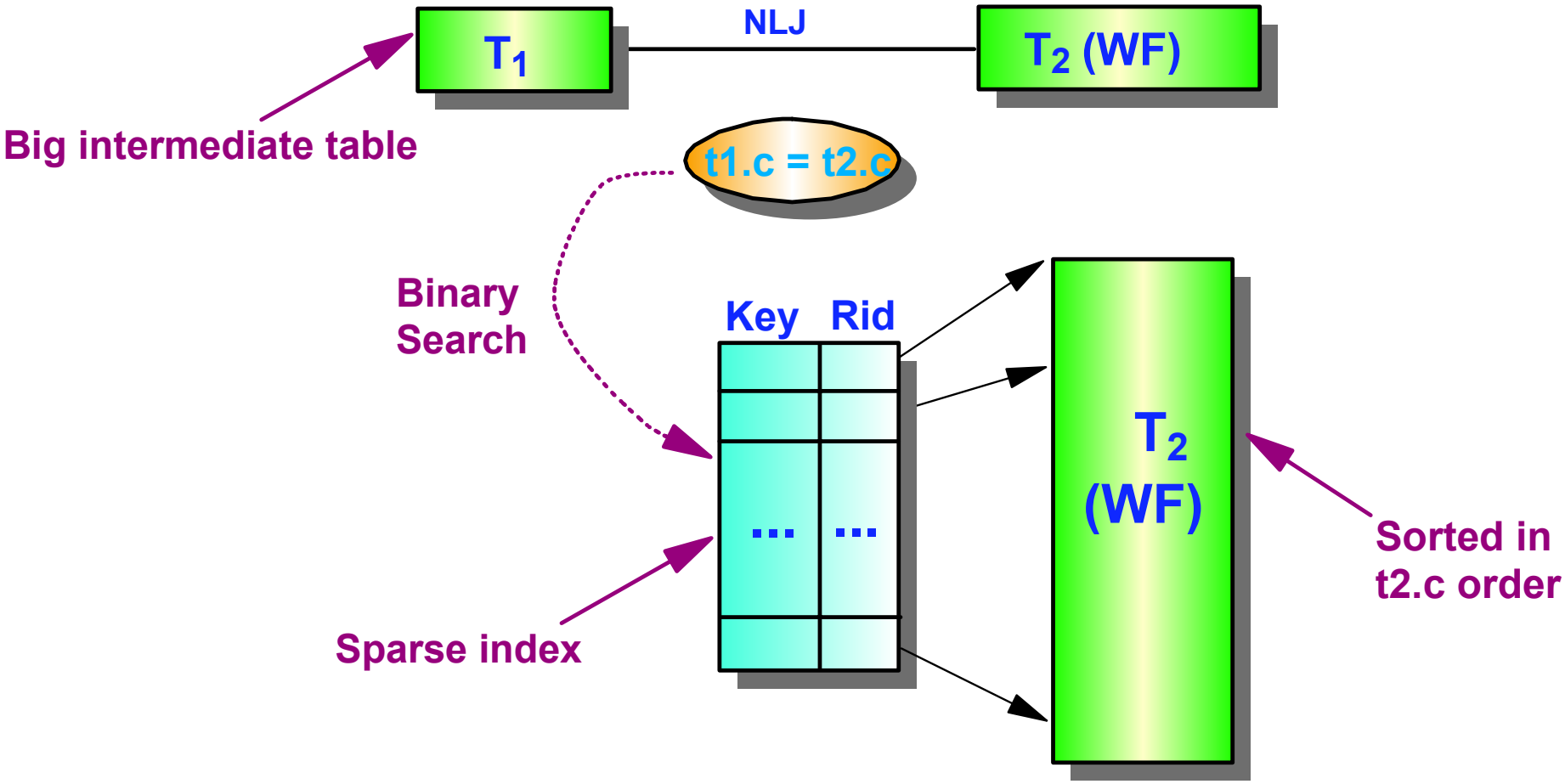
- Introduction
- New Enhancement
  - Star Schema Detection
  - Efficient Access to Workfile - Phase I
- Future Work
  - Efficient Access to Workfile - Phase II
  - In-Memory Workfile

Index on Workfile - Phase II

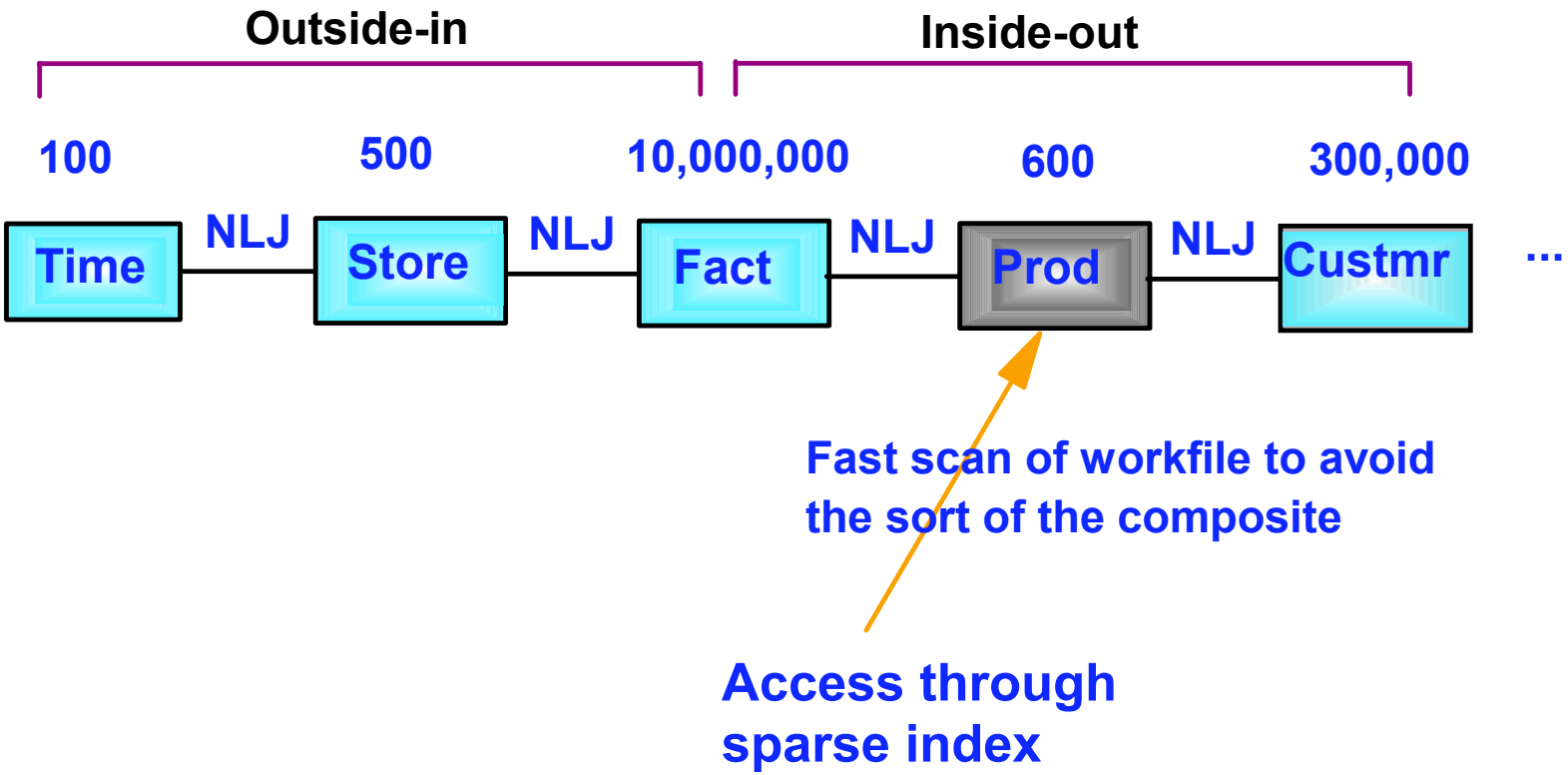


Index on Workfile - Phase I

- Sparse index



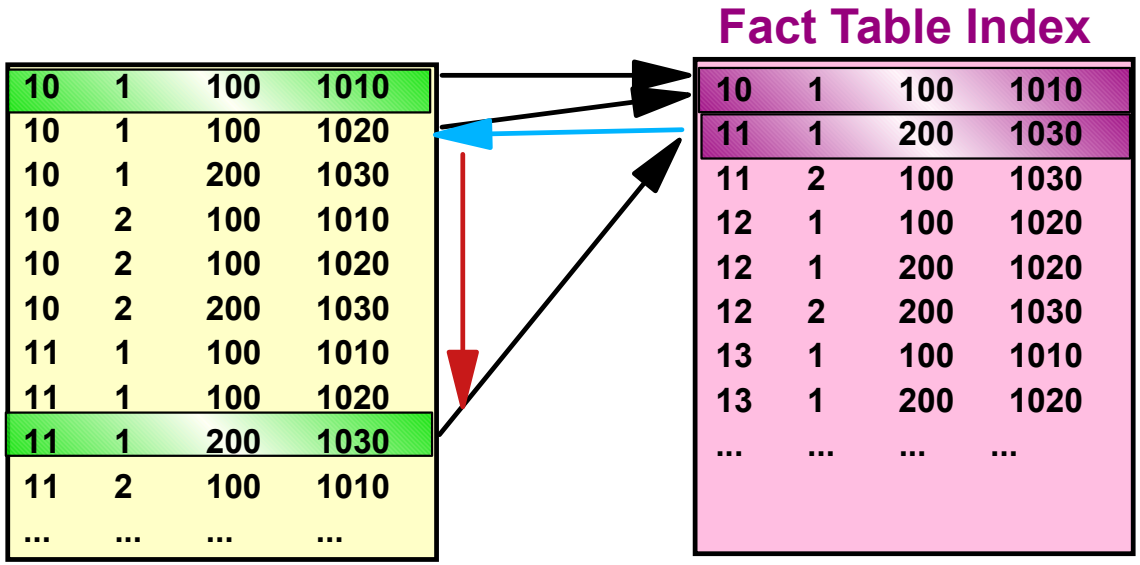
Index on Workfile - Phase I



Index on Workfile - Phase II

- Issue - skipping the index keys is not free

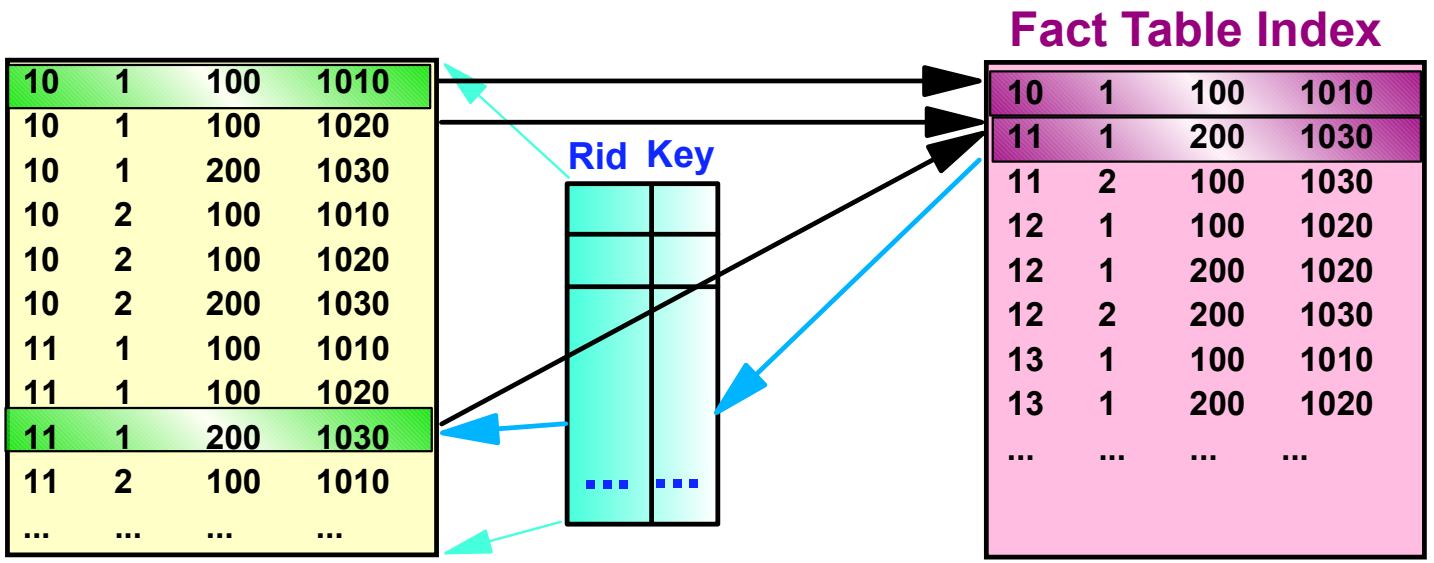
1. Dimensions workfiles are sorted in the join column order
2. Skipping index keys in a particular dimension requires sequential scan of the workfile
3. Therefore skipping of "inner" dimensions may be expensive



Index on Workfile - Phase II

Solution - Sparse index

1. In-memory index
2. Binary search
3. Efficent when there is a big jump

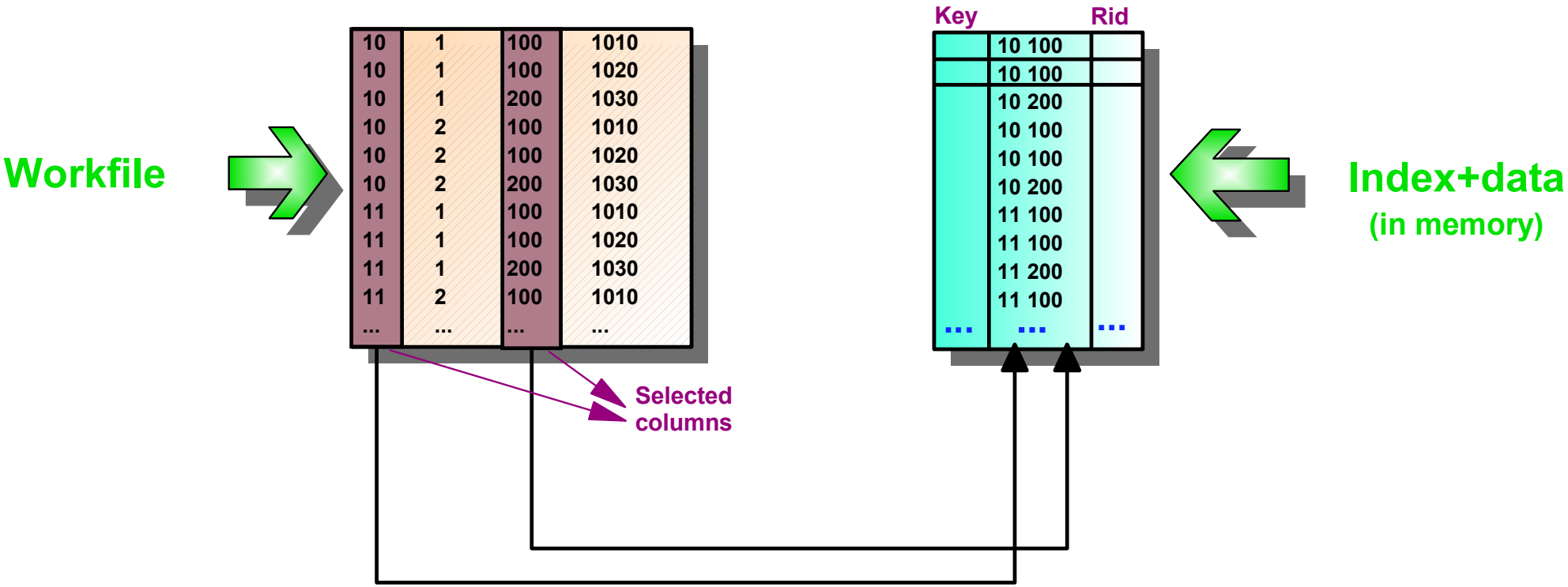


Presentation Outline

- Introduction
- New Enhancement
  - Star Schema Detection
  - Efficient Access to Workfile - Phase I
- Future Work
  - Efficient Access to Workfile - Phase II
  - In-Memory Workfile

In-Memory Workfile

- Caching the referenced columns in sparse index
  1. Build in-memory index on the join column
  2. Add the selected columns into the index
  3. Binary search



Star Join White Paper

- **Evolution of the Star join Optimization**

  DB2 UDB for OS/390 and z/OS

HTTP://WWW-3.IBM.COM/SOFTWARE/DATA/DB2/OS390/TECHDOCS/STARJOIN.PDF