DB2 Statistics

Craig Friske

DB2 Utilities Development

friske@us.ibm.com

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Agenda

- DB2 RUNSTATS Statistics Concepts
- RUNSTATS Recommendations
- DB2 Real Time Statistics
- REORG, COPY, and RUNSTATS Scheduling

DB2 RUNSTATS Statistics Concepts

Useful Statistics Concepts

- Access Paths and Filter Factors
- High/Low bound, cardinalities
- Distribution Statistics
- Histogram Statistics
- Clustering, Cluster Count, Off Position
- LEAFNEAR/LEAFFAR for indexes
- Pointers to overflow rows (indirect references)

Access Paths Lite (very light)

- Performance involves Access Path and organization of data and keys. The Optimizer is cost based. Cost is minimized by decreasing I/Os or the number of rows accessed.

- The Optimizer decides things like whether to use an index for accessing data, whether an index alone can be used, or in which order tables are accessed when doing JOINs.

- Other statistics don't affect access path selection, but they can be an indicator of performance degradation, and they may signal that action should be take for improved performance.

Simple Access Path Examples

- Indexes can be very useful and chosen, especially if the filter factor is good or clustering is good. Here are some examples for a table with NBA player info:
  - Example 1

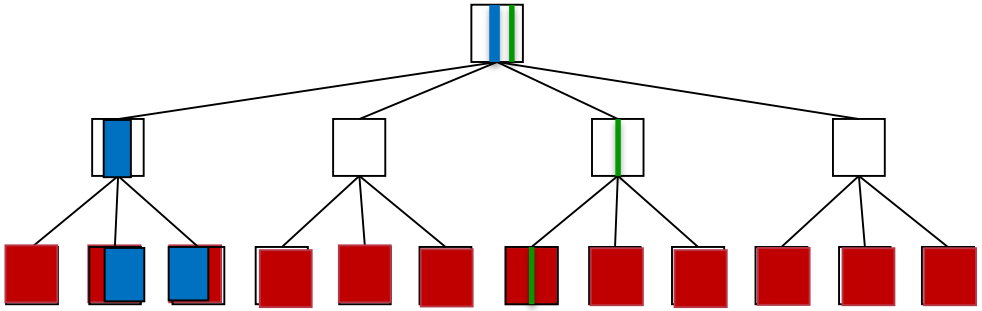
    SELECT JERSEY_NUMBER FROM NBA_PLAYERS WHERE NAME='STEPHEN CURRY';
  - Example 2

    SELECT NAME FROM NBA_PLAYERS WHERE TEAM = 'WARRIORS';
  - Example 3

    SELECT NAME FROM NBA_PLAYERS WHERE YEARLY_COMPENSATION > $1M;

Tale of 3 access paths

- Filtering restricts access to a subset of the index entries or data rows
  - Can reduce index I/O
  - Generally results in reduction in data I/O
  - Can't always filter



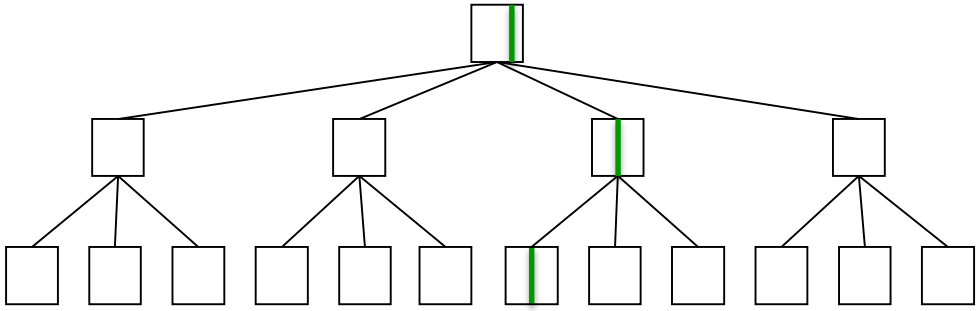
SELECT JERSEYNO WHERE NAME = name

SELECT NAME WHERE TEAM = team (cluster order by team)

SELECT NAME WHERE YEARLY > salary (no filtering)

Tale of 3 access paths

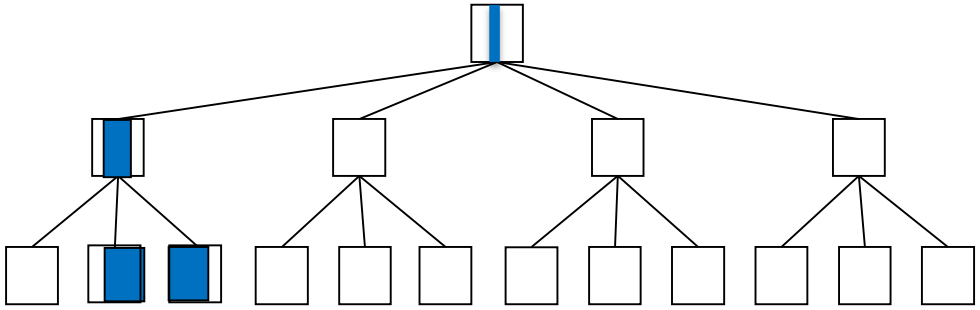
- Matching Index Probe
  - “Unique” index probe limited to 2 index pages and 1 data page
  - Access 1 data row



SELECT JERSEYNO WHERE NAME = name

Tale of 3 access paths

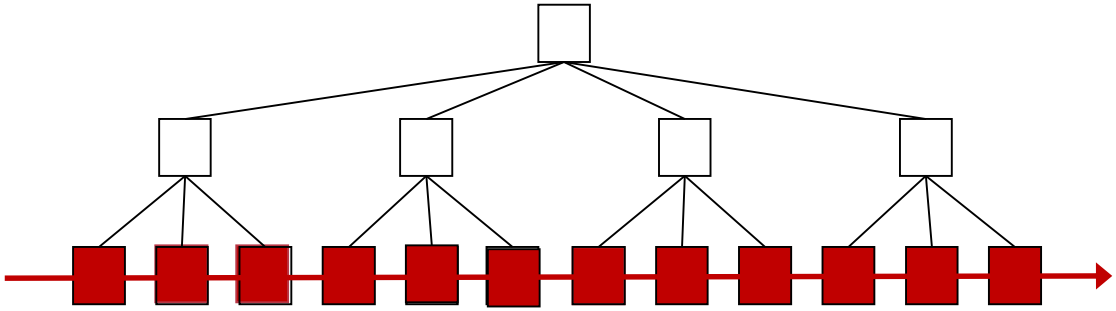
- Matching Index Scan
  - Index on Team with data ordered according to the index
  - Minimal leaf page and data pages access



SELECT NAME WHERE TEAM = team
(where cluster order is team)

Tale of 3 access paths

- If filtering cannot avoid any data pages
  - May as well to a table/table space scan



SELECT NAME WHERE YEARLY > salary (no filtering)

Oversimplified Optimizer Costing

- Optimizer assigns Filter Factors for each WHERE/ON predicate

- FFs are combined to determine the total filtering per object
  - Multiply “AND” predicate FFs
    - Available statistics determine “degree” of multiplication
  - Add “OR” predicate FFs

  - FF accuracy and how to combine these is important for costing
    - Index matching
    - Total index filtering
    - Total table level filtering

Access Path Attributes

- Key attributes collected by RUNSTATS for use by the optimizer:
  - Size of the objects
    - NPAGESF, NLEAF, NLEVELS etc.
  - Range on records/keys
    - LOW2KEY, HIGH2KEY, LOWKEY, HIGHKEY
  - Selectivity or number of records/keys
    - CARDF, COLCARDF, FIRSTKEYCARDF, FULLKEYCARDF, FREQVAL etc.
  - Other important statistics
    - CLUSTERRATIOF, PCTROWCOMP etc.

Filter Factors and cardinality

SYSCOLDIST and SYSYSCOLDISSTATS contain frequency (or distribution) values

If frequency stats do not exist, DB2 assumes that the data is uniformly distributed

For example:

NBA players table (450 rows)

Player	J#	Team	State
Stephen Curry	30	Warriors	CA
Lebron James	23	Cavaliers	OH
James Harden	13	Rockets	TX
Anthony Davis	23	Pelicans	LA
Chris Paul	3	Clippers	CA
Russell Westbrook	0	Thunder	OK
More…			

Cardinality	447	50	30	25
#Rows/Card	1.01	9	15	17 (est rows per value)

Filter Factors and skewed data

Distribution statistics can help produce accurate filtering for skewed data

if there are 30 teams and 66 champions, shouldn't I expect each team to have 2 entries (3.3%)?

NBA Historical Stats

Year	Team	Wins
2015	Warriors	67
2015	Cavaliers	53
1996	Bulls	72
1998	Nuggets	11
1973	Sixers	9
More…		

NBA Champs

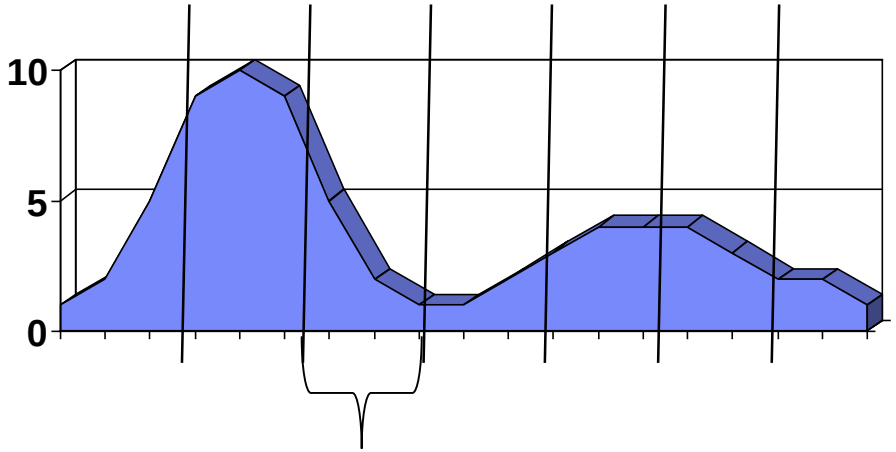
Year	Winner	Loser
2015	Warriors	Cavaliers
2014	Spurs	Heat
1997	Bulls	Jazz
1987	Lakers	Celtics
1950	Lakers	Sixers
More…		

#rows=1650?, Card=33 65?
Highkey 72
Lowkey 9
High2key 69
Low2key 11

#rows=66, Card=18 23
Distributions Stats on Winner
Value = 'Celtics' Count = 17 Freq=25%
Value = 'Lakers' Count = 16 Freq=24%
Value = 'Bucks' Count = 1 Freq=1.5%
Value = 'Hornets' ??

Histogram Statistics

Think of filter factor stats on a range (quantile) of data (helpful for range predicates)



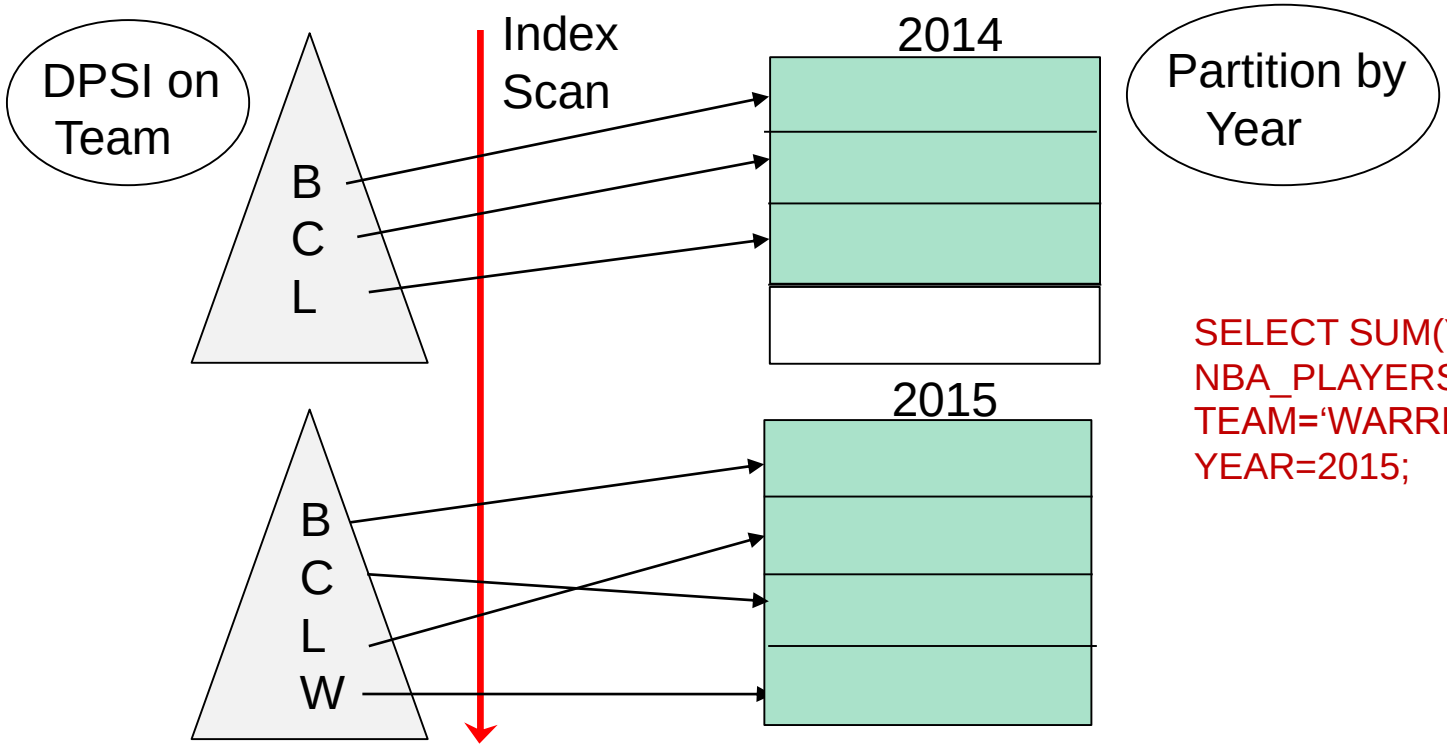
LOWVALUE, HIGHVALUE, CARDF, and FREQUENCYF

```
RUNSTATS TABLESPACE ts TABLE(tb) COLGROUP(C4)
      FREQVAL COUNT 20 MOST HISTOGRAM NUMQUANTILES 100
RUNSTATS INDEX(IX FREQVAL NUMCOLS 15 COUNT 10 MOST
                  HISTOGRAM NUMQUANTILES 100)
```

Catalog Table:
Kept in SYSCOLDIST and SYCOLDISTSTATS

CLUSTERING INDEX

An index that determines how rows are physically ordered (clustered) in a table space. If a clustering index on a partitioned table is a DPSI, the rows are ordered in cluster sequence within each data partition instead of spanning the partitions.




```
SELECT SUM(YEARLY)  FROM
NBA_PLAYERS WHERE
TEAM='WARRIORS' AND
YEAR=2015;
```

When data row obtained via index scan using “TEAM”. All rows are in optimal order (or clustering order, clustered) except when accessing row ‘L’akers in 2015.

CLUSTERRATIO

SYSIBM.SYSINDEXES.CLUSTERRATIO

- An access path statistic that can also help in determining when to REORG
- % of the rows that are in cluster order (adjusted formula) for clustering indexes (100% is ideal). There can only be 1 clustering index, but other indexes can be correlated with the clustering index.
- Rows are counted as being “clustered” if they are within the prefetch range for either a forward or backward reference.
- This is a statistic that describes the data in the table(space), even though it is reported in SYSINDEXES – REORG INDEX will never affect this statistic

Examples: CREATE INDEX ICLUST on TABLE TB1(TEAM) CLUSTER;

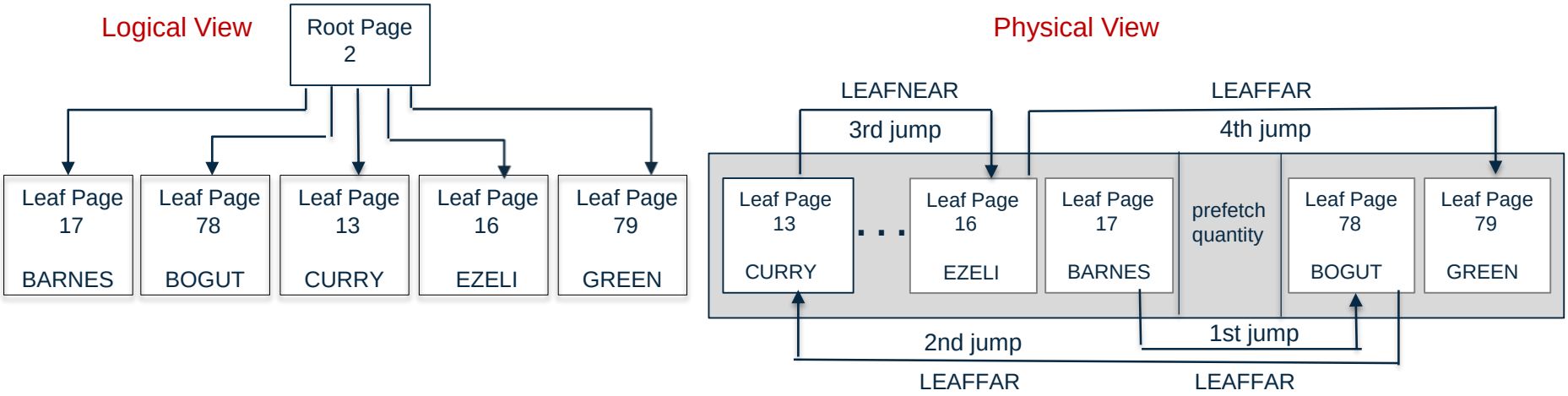
CREATE INDEX ICLUST2 on TABLE TB1(TEAM, JERSEY#);

CREATE INDEX NOCLUST on TABLE TB1(TEAM) DESC;

LEAFNEAR/LEAFFAR

Measures the disorganization of physical leaf page. Pages are not in an optimal position due to index pages being deleted or index leaf page splits caused by an insert that cannot fit onto a full page. Affects performance during an index scan.

Logical and physical views of an index in which LEAFNEAR=1 and LEAFFAR=3

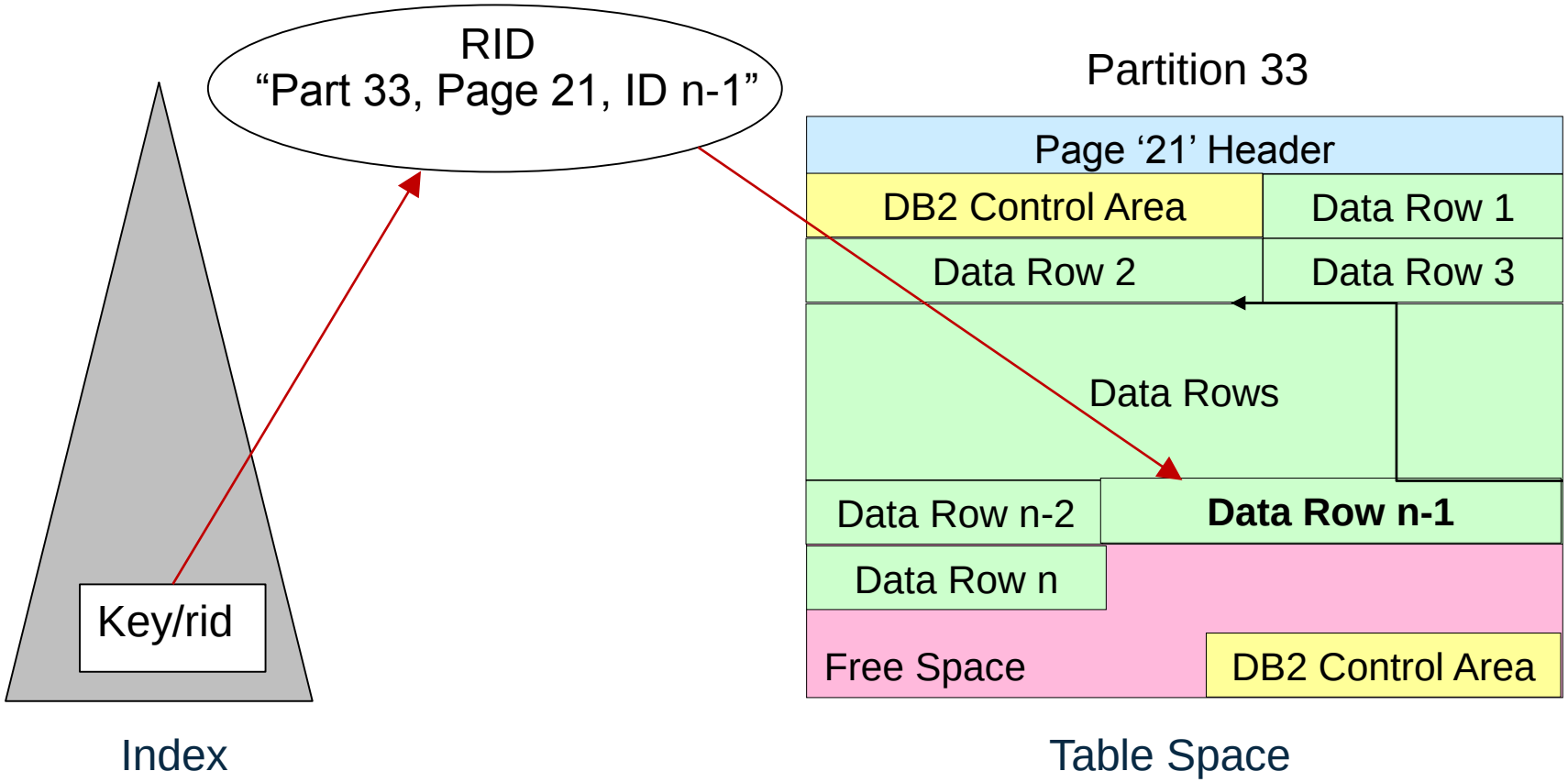


RUNSTATS: LEAFNEAR, LEAFFAR; RTS: LEAFNEAR, LEAFFAR

Action: REORG INDEX

(V10 list prefetch mitigates LEAFNEAR/FAR performance degradation)

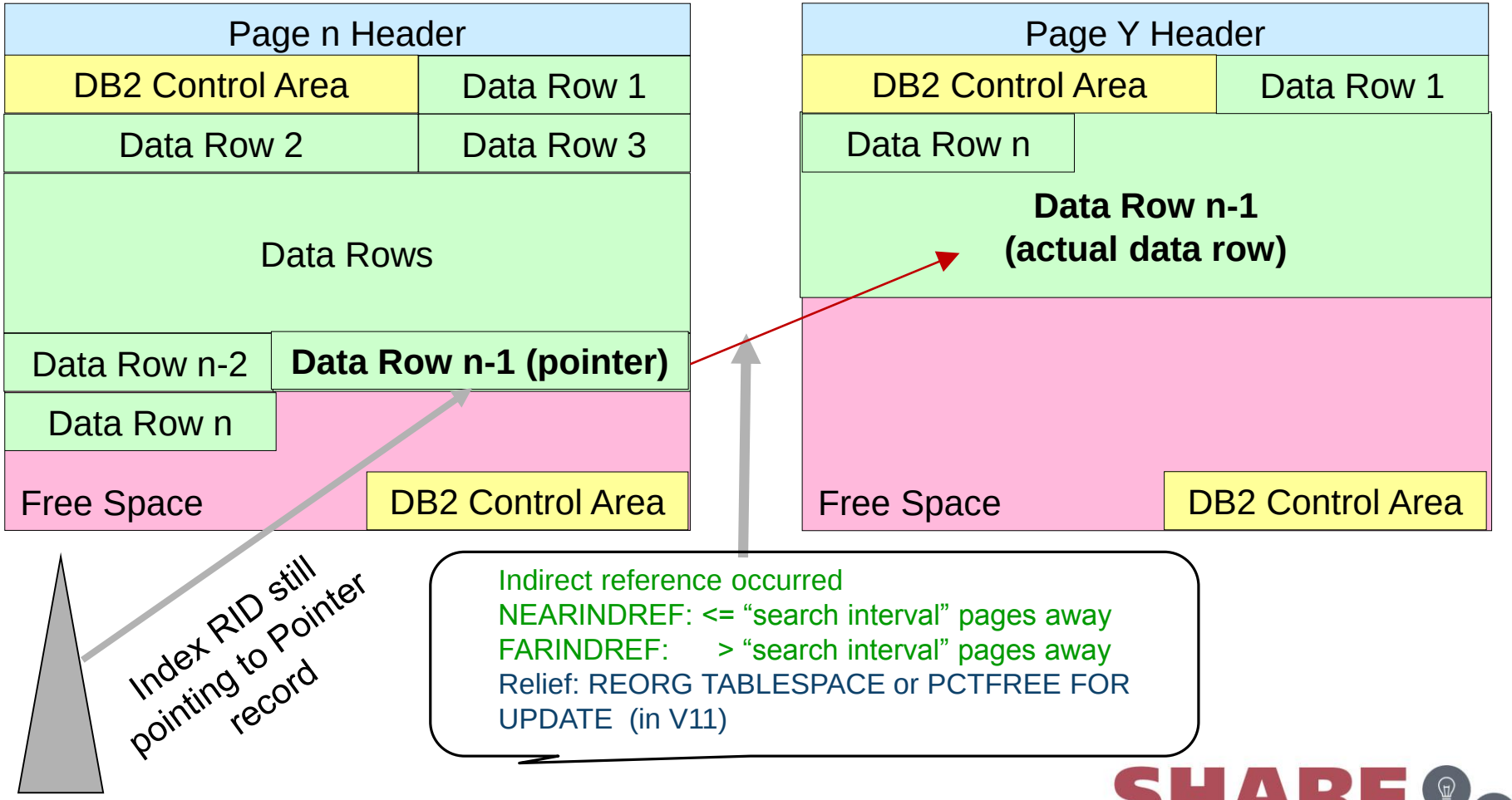
INDEX key and data row referencing



DB2 *Indirect* reference

UPDATE to the record increases the row length, no room to fit

(e.g. UPDATE ADDRESS = "…Oakland" WHERE NAME='KEVON LOONEY')



RUNSTATS stats gathering

Statistics gathered by RUNSTATS TABLESPACE

Access path statistic
Access path (not used)
Space statistic

SYSIBM.SYSTABLESPACE
- NACTIVEF
- AVGROWLENG
- SPACEF

SYSIBM.SYSTABLES/HIST
- CARD/F
- NPAGESF
- PCTPAGES
- PCTROWCOMP
- AVGROWLENG
- SPACEF

SYSIBM.SYSTABSTATS
- CARD/F
- NPAGES
- PCTPAGES
- NACTIVE
- PCTROWCOMP

SYSIBM.SYSTABLEPART/HIST
- AVGROWLEN
- CARD/F
- DSNUM
- EXTENTS
- NEARINDREF
- FARINDREF
- PAGESAVE
- PERCACTIVE
- PERCDROP
- SPACE/F
- PQTY
- SQTY
- SECQTYI

SYSIBM.SYSCOLUMNS/HIST
- COLCARD/F
- HIGH2KEY
- LOW2KEY

SYSIBM.SYSCOLSTATS
- COLCARD
- HIGHKEY
- HIGH2KEY
- LOWKEY
- LOW2KEY
- COLCARDDATA

SYSIBM.SYSCOLDIST/HIST/STATS
- NUMCOLUMNS
- COLGROUPCOLNO
- COLVALUE
- CARDF
- TYPE (‘C’,’F’,’H’)
- FREQUENCY/F
- KEYCARDDATA
- QUANTILENO

- Covered in RUNSTATS concepts section previously
- Overlap with RUNSTATS INDEX

Statistics gathered by RUNSTATS INDEX

Access path statistic
Access path (not used)
Space statistic

SYSIBM.SYSINDEXES/HIST
- CLUSTERRATIO/F
- CLUSTERED
- FIRSTKEYCARD/F
- FULLKEYCARD/F
- NLEAF
- NLEVELS
- AVGKEYLEN
- SPACEF

SYSIBM.SYSINDEXSTATS/HIST
- FIRSTKEYCARD/F
- FULLKEYCARD/F
- NLEAF
- NLEVELS
- IOFACTOR
- PREFETCHFACTOR
- KEYCOUNT/F
- CLUSTERRATIO/F
- FULLKEYCARDDATA

SYSIBM.SYSINDEXPART/HIST
- AVGKEYLEN
- CARDF
- DSNUM
- EXTENTS
- FAROFFPOSF
- LEAFNEAR
- LEAFFAR
- NEAROFFPOS
- LEAFDIST
- PSUEDO_DEL_ENTRIES
- SPACEF
- PQTY
- SECQTYI

SYSIBM.SYSCOLUMNS/HIST
- COLCARD/F
- HIGH2KEY
- LOW2KEY

SYSIBM.SYSCOLSTATS
- COLCARD
- HIGHKEY
- HIGH2KEY
- LOWKEY
- LOW2KEY
- COLCARDDATA

SYSIBM.SYSCOLDIST/HIST/STATS
- NUMCOLUMNS
- COLGROUPCOLNO
- COLVALUE
- CARDF
- TYPE (‘C’,’F’,’H’)
- FREQUENCY/F
- KEYCARDDATA
- QUANTILENO

- Covered in RUNSTATS concepts section previously
- Overlap with RUNSTATS TABLESPACE

RUNSTATS Statistics Collection

- Basic statistics foundation

CPU savings with V10 page sampling

```
RUNSTATS TABLE (ALL) TABLESAMPLE SYSTEM AUTO
         INDEX (ALL) KEYCARD
         SHRLEVEL CHANGE
```

KEYCARD is the default from DB2 10

- Supplement with more detailed statistics as needed
  - Distribution statistics
    - Frequencies
    - Histograms
  - Multi-column cardinality statistics

- Tradeoff with CPU savings and HISTOGRAM accuracy

How do I integrate supplemental statistics?

```
RUNSTATS LIST mylist
         TABLE (ALL) TABLESAMPLE SYSTEM AUTO
         INDEX (ALL) KEYCARD

RUNSTATS mydb.myts
         TABLE(NBA_PLAYERS)
         COLGROUP(TEAM,JERSEY#)
```

- - Tradeoff with CPU savings and HISTOGRAM accuracy

Mixing Regular and “special” RUNSTATS

- If I run the following

```
1. RUNSTATS TABLE(ALL)TABLESAMPLE SYSTEM AUTO INDEX(ALL)KEYCARD

2. RUNSTATS TABLE(NBAChamps) COLGROUP(WINNER) FREQVAL COUNT 20

3. RUNSTATS TABLE(ALL)TABLESAMPLE SYSTEM AUTO INDEX(ALL)KEYCARD
```

- Won’t “Regular” RUNSTATS overwrite the “special”?
  - NO: RUNSTATS will only overwrite similar statistics
    - COLGROUP(STATUS) FREQVAL COUNT 20 is only overwritten if default statistics are collecting FREQVAL on this column
    - Is there an index leading with STATUS? Default is to collect top 10 (not top 20).

- Default since V10

DB2 V10 Simplifies Integration of Supplemental Stats

- Integrate specialized statistics into generic RUNSTATS job
  - RUNSTATS TABLE (mytb) COLGROUP(STATUS)… **SET PROFILE**
    - Or **SET PROFILE FROM EXISTING STATS**
  - RUNSTATS … TABLE (mytb) …. **UPDATE PROFILE**

- Next usage
  - RUNSTATS LIST mylist TABLE(ALL) **USE PROFILE**
    - RUNSTATS will execute as if all saved options were specified

- Caveats
  - Cannot specify USE PROFILE for a table without a defined profile (no defaults)
    - Restricts LISTDEF support
  - USE PROFILE not supported with inline stats

V11 Improvements

- **Improved PROFILE usability with LISTDEF support**
  - Gather default statistics if no profile exists for table
- **More zIIP offload for RUNSTATS distribution statistics**
  - Up to 80% zIIP-eligible
- **Inline statistics RUNSTATS equivalence (avoid RUNSTATS)**
  - Inline statistics collection on NPSIs during REORG with SORTNPSI
  - Inline histogram statistics
  - Inline DSTATS
  - zIIP offload up to an additional 30%
  - Still missing PROFILE support
- **RUNSTATS RESET option deletes/clears all catalog stats for an object**

DB2 V11 Optimizer externalization of missing stats

- During access path calculation, optimizer will identify missing or conflicting statistics
  - On every BIND, REBIND or PREPARE
    - Asynchronously writes recommendations to SYSIBM.SYSSTATFEEDBACK
      - From DB2 11 NFM
  - DB2 also provides statistics recommendations on EXPLAIN
    - Populates DSN_STAT_FEEDBACK synchronously
      - Beginning in DB2 11 CM – provided explain table exists

- Contents of SYSSTATFEEDBACK or DSN_STAT_FEEDBACK can be used to generate input to RUNSTATS
  - Contents not directly consumable by RUNSTATS

Object, Type, and Reason for statistics recommendations

- Object is identified as table, index, or column
- TYPE specifies the statistics to collect

TYPE	CHAR(1)	The type of statistic to collect:
		'C' Cardinality.
		'F' Frequency.
		'H' Histogram.
		'I' Index.
		'T' Table.

- REASON identifies why statistics were recommended

Real Time Statistics - RTS

Real-Time Statistics (RTS) Objective

- Older DBA procedures and some home-grown monitor tools had no accurate data to identify objects that need maintenance
- Spending time performing maintenance on static and unchanged objects inefficient use of DBA's time, waste batch window time and CPU
  - The “best” utility is the one not needed
- DB2 systems becoming larger and more complex
  - A single DB2 for z/OS may have large amounts of tables/indexes - for ERP-packaged applications, it can be 80K +
  - Requires skilled DBAs (and lots of time) to identify unused / static objects
- Goal is to self-managed or automate the maintenance process
- DB2 Stored Procedures, *DSNACCOX*, IBM *DB2 Automation Tool* and the new *DB2 Management Console* exploit RTS

RTS Overview

- Runs in the background - automatically updates statistics, as the data rows and indexes for DB2 table spaces are modified
- RTS manager runs under a system task in DBM1 address space
  - CPU time is included in DBM1's SRB time
  - The system task is created during START DB2
- Statistics collected in memory, and periodically externalized
  - -ACCESS DATABASE … MODE(STATS)
- Contains space and as well as some access path statistics
- Externalized into DB2 Catalog – SYSIBM.DSNRTSTS:
  SYSIBM.SYSTABLESPACESTATS
  SYSIBM.SYSINDEXSPACESTATS
- Helps eliminate scheduling RUNSTATS (but can't replace RUNSTATS)

V11 RTS Tables – SYSTABLESPACESTATS

Global Statistics	Incremental Statistics		
	REORG-	**COPY-**	***RUN*STATS-**
DBID	LASTTIME	LASTTIME	LASTTIME
PSID	INSERTS	UPDATEDPAGES	INSERTS
PARTITION	UPDATES	CHANGES	UPDATES
INSTANCE	DELETES	UPDATELRSN	DELETES
DBNAME	DISORGLOB	UPDATETIME	MASSDELETE
NAME	UNCLUSTINS		
NACTIVE	MASSDELETE		
NPAGES	NEARINDREF		
EXTENTS	FARINDREF		
SPACE	CLUSTERSENS		
TOTALROWS	SCANACCESS		
DATASIZE	HASHACCESS		
UNCOMPRESSEDDATASIZE			
UPDATESTATSTIME			
HASHLASTUSED	**LOAD-**		
DRIVETYPE	RLASTTIME		
LPFFACILITY			
UPDATESIZE			
LASTDATACHANGE			
GETPAGES			

- New in V10
- New in V11

V11 RTS Tables – SYSINDEXSPACESTATS

Global Statistics	Incremental Statistics		
	REORG-	**COPY-**	***RUN*STATS-**
DBID	REBUILDLASTTIME	LASTTIME	LASTTIME
ISOBID	LASTTIME	UPDATEDPAGES	INSERTS
PSID	INSERTS	CHANGES	DELETES
PARTITION	UPDATES	UPDATELRSN	MASSDELETE
INSTANCE	DELETES	UPDATETIME	
NACTIVE	APPENDINSERT		
NLEVELS	PSEUDODELETES		
NPAGES	MASSDELETE		
NLEAF	LEAFNEAR		
EXTENTS	LEAFFAR		
SPACE	NUMLEVEL		
TOTALENTRIES	INDEXACCESS		
LASTUSED			
UPDATESTATSTIME	**LOAD-**	**REBUILD*X*-**	
DBNAME	RLASTTIME	LASTTIME	
NAME			
INDEXSPACE			
DRIVETYPE			
GETPAGES			

- New in V10
- New in V11

RTS Usage – History and trending

- There is currently no historical capability in RTS in DB2 itself
- Create a history table manually

  CREATE
  SYSIBM.(TABLE | INDEX)SPACESTATS_HIST
  LIKE
  SYSIBM.SYS(TABLE | INDEX)SPACESTATS

  then add

  CAPTURE_TIME AS TIMESTAMP NOT NULL WITH DEFAULT column
- Periodically insert into RTS history tables with a sub select from the RTS tables those rows that aren't already in the history tables; and delete old information.

  Some customers do this weekly, others monthly – depending on needs

RTS Usage – Monitor Object Activity

Object activity

- How active are my DB2 objects?
- What activity has taken place for a specific time for TS' and IX's
- Use UPDATESTATSTIME

```
SELECT DBNAME, NAME, PARTITION, UPDATESTATSTIME
        FROM SYSIBM.TABLESPACESTATS
        WHERE (JULIAN_DAY(CURRENT DATE) –
                JULIAN_DAY(UPDATESTATSTIME)) <= 14
          AND NAME = xxx;
```

Show me the activity during the last 14 days

- Use DB2 Administration Tool – DB2 Performance Queries

RTS Usage – Determine Index Value

Unused or (in)activity of INDEXES

- –LASTUSED column in SYSINDEXSPACESTATS
  - Is a **date** field
  - Consider using for identifying which IXs to drop
  - The date indicates the index is last used for SELECT, FETCH, searched UPDATE, searched DELETE, or used to enforce referential integrity constraints.
  - The default value is 01/01/0001.
- –REORGINDEXACCESS column in SYSINDEXSPACESTATS
  - # of times the IX was **accessed** (read and updates) since last reorg or since creation
  - NULL denotes never used

RTS Usage – Track Utility execution

- When was the last time a utility was run against my objects?
- When was COPY, REORG, LOAD REPLACE, and RUNSTATS last executed against objects ..

```
SELECT DBNAME, NAME, PARTITION, TOTALROWS, NACTIVE,
SPACE, EXTENTS, UPDATESTATSTIME, STATSLASTTIME,
LOADRLASTTIME, REORGLASTTIME, COPYLASTTIME
      FROM SYSIBM.TABLESPACESTATS
ORDER BY DBNAME, NAME, PARTITION
```

- Or use DB2 Administration Tool for reporting
- For object maintenance queries/info consider *DB2 Automation Tool*, free stored procedure *DSNACCOX* and *DB2 Management Console*

Object Maintenance (aka utilities scheduling)

What is DSNACCOX?

A DB2 stored procedure that accesses the RTS tables and looks at DBET states to give recommendations for when schedule table spaces or indexes maintenance for reorganization, taking image copies, or updating statistics:

- REORG TABLESPACE, REORG INDEX
- RUNSTATS TABLESPACE, RUNSTATS INDEX
- COPY TABLESPACE, COPY INDEX

Reorg table space recommendations

- #Inserts since last REORG > 25% of total rows and #Inserts > 0
- #Deletes since last REORG > 25% of total rows and #Deletes > 0
- #Cluster Accesses since last REORG > 0 and #Unclustered Inserts since last REORG > 10%
- #Overflow Rows since last REORG > 10%
- #Mass deletes since last REORG > 0
- #Extents > 254
- #Disorganized LOBS > 50%
- #Hash Index Entries since last REORG > 15%
- Object is in REORG Pending (Alter Limit Key, Add Identity Column blocks access)
- Object in Advisory REORG Pending AREOR (Pending Alter materialization)
- Object in Advisory REORG AREO* (Immediate Alter materialization)

Reorg index recommendations

- #Inserts since last REORG > 30% and #Inserts > 0
- #Deletes since last REORG > 30% and #Deletes > 0
- #Inserts appended since last REORG > 20%
- #Pseudodeletes since last REORG > 10%
- #Mass Deletes since last REORG > 0
- #LEAFFAR since last REORG > 10%
- #Levels > 0
- #Extents > 254
- #Extra formatted pages > 10%
- Object in Advisory REORG Pending AREOR (Pending Alter materialization)

Copy scheduling recommendations

Full Image Copy on a Table Space

- Table space has never had a full image copy
- Last image copy is older than 7 days
- #Updated pages since the last copy > 10% and # pages changed > 0
- The object is in Copy Pending

Incremental Copy on a Table Space

- Table space has never had an incremental image copy
- Last image copy is older than 7 days
- #Updated pages since the last copy > 1% of the total pages, and #updated pages > 0
- #Updated rows since the last copy > 1% of the total rows.

RUNSTATS scheduling recommendations

RUNSTATS on a Table Space

- If RUNSTATS has never been run
- #Inserts, #Deletes, and #Updates > 20%, and #changes > 0
- #Mass Deletes > 0

RUNSTATS on an INDEX

- If RUNSTATS has never been run
- #Inserts and #Deletes > 20%, and #changes > 0
- #Mass Deletes > 0

Remember, RUNSTATS followed by REBIND may alter the access path. Consider using plan stability (e.g. PLANMGMT and REBIND SWITCH) to avoid surprises.

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Acknowledgements and Disclaimers:

Availability. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. 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. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. 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 the applicable license agreement governing the use of IBM software.

Special thanks to Bryan Smith, Terry Purcell, Henni Mynhardt for their help.