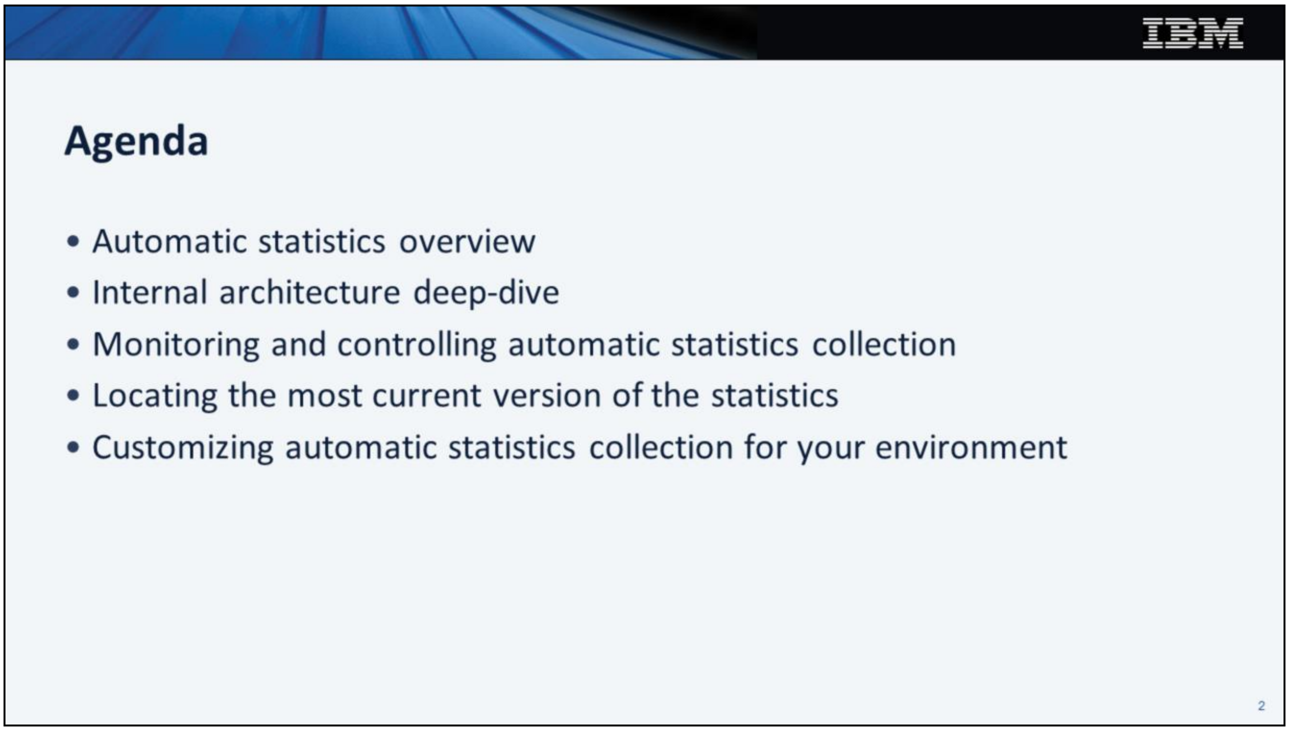
Db2 Automatic Statistics Deep Dive

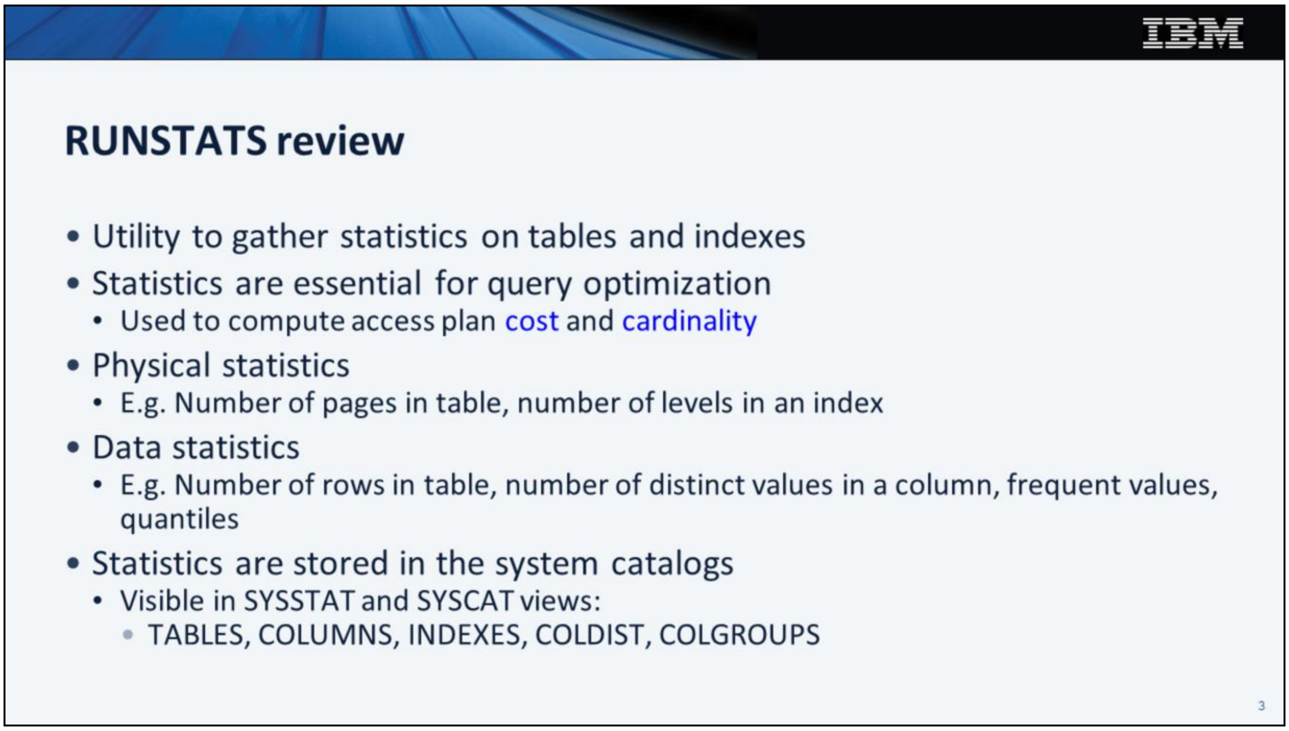
John Hornibrook
IBM Canada

Current and accurate catalog statistics are the lifeblood of the query optimizer Db2 has had automatics statistics support since version 8.2 and real-time statistics were introduced in version 9.5. Automatic statistics collection is enabled by default for new databases and is the recommend best practice. So if you are still hesitant to use them, or have tried and have had a bad experience, attend this session to gain a deeper understanding of how they work and why you can trust them to simplify your DB administration and improve overall query performance. In addition to providing details on the automatic statistics architecture, this presentation will provide details on ways to monitor and control automatic statistics collection so that it works well for any type of Db2 system.



Agenda

- Automatic statistics overview
- Internal architecture deep-dive
- Monitoring and controlling automatic statistics collection
- Locating the most current version of the statistics
- Customizing automatic statistics collection for your environment

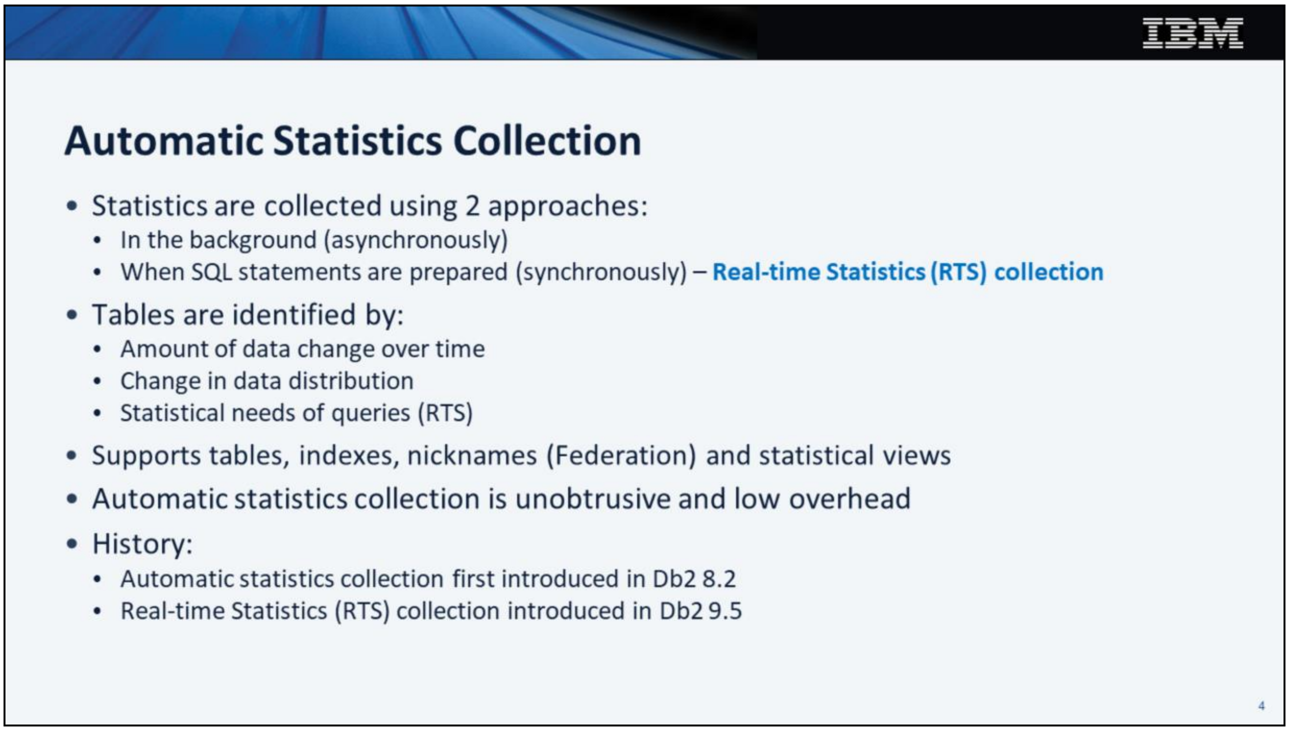


RUNSTATS review

- Utility to gather statistics on tables and indexes
- Statistics are essential for query optimization
  - Used to compute access plan cost and cardinality
- Physical statistics
  - E.g. Number of pages in table, number of levels in an index
- Data statistics
  - E.g. Number of rows in table, number of distinct values in a column, frequent values, quantiles
- Statistics are stored in the system catalogs
  - Visible in SYSSTAT and SYSCAT views:
    - TABLES, COLUMNS, INDEXES, COLDIST, COLGROUPS

3

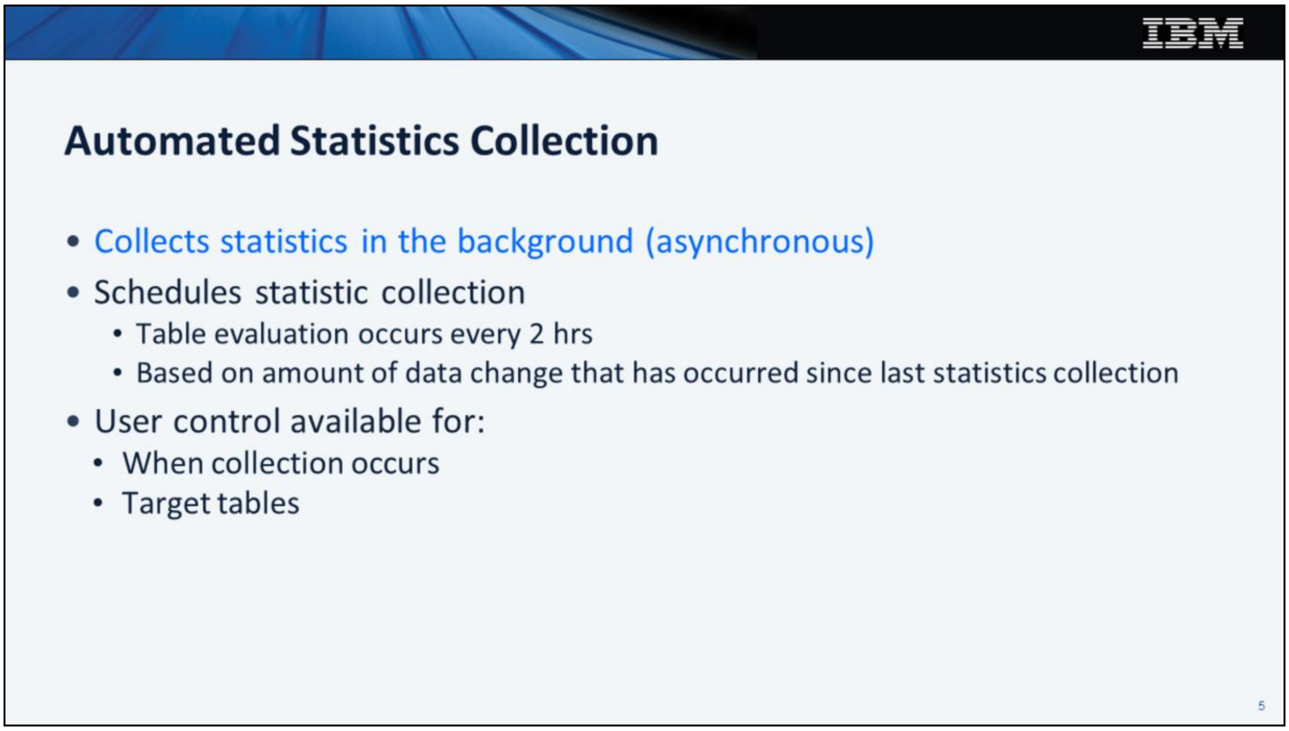
When the SQL compiler optimizes SQL query plans, its decisions are heavily influenced by statistical information about the size of the database tables and indexes. The optimizer also uses information about the distribution of data in specific columns of tables and indexes if these columns are used to select rows or join tables. The optimizer uses this information to estimate the costs of alternative access plans for each query. When significant numbers of table rows are added or removed, or if data in columns for which you collect statistics is updated, execute RUNSTATS again to update the statistics. Statistical information is collected for specific tables and indexes in the local database when you execute the RUNSTATS utility. Statistics are collected only for the table partition that resides on the partition where you execute the utility or the first partition in the database partition group that contains the table. The collected statistics are stored in the system catalog tables.



Automatic Statistics Collection

- Statistics are collected using 2 approaches:
  - In the background (asynchronously)
  - When SQL statements are prepared (synchronously) – **Real-time Statistics (RTS) collection**
- Tables are identified by:
  - Amount of data change over time
  - Change in data distribution
  - Statistical needs of queries (RTS)
- Supports tables, indexes, nicknames (Federation) and statistical views
- Automatic statistics collection is unobtrusive and low overhead
- History:
  - Automatic statistics collection first introduced in Db2 8.2
  - Real-time Statistics (RTS) collection introduced in Db2 9.5

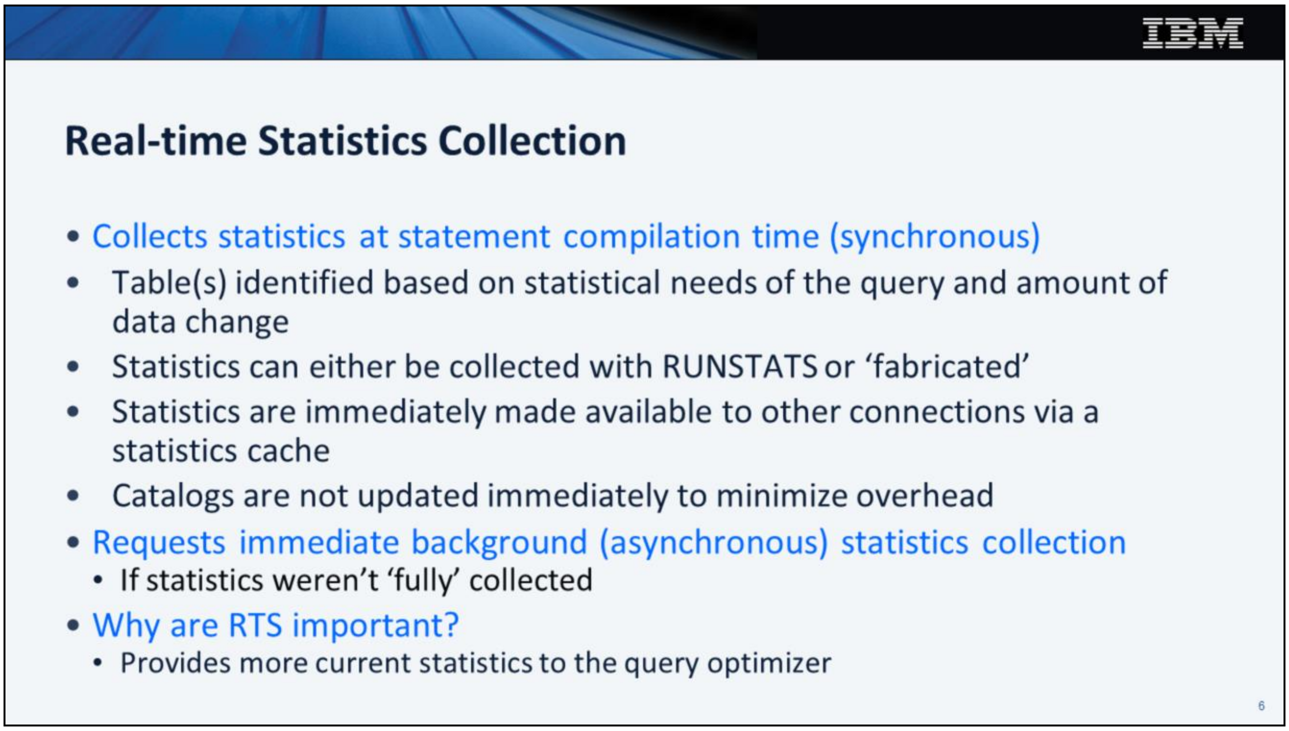
4



Automated Statistics Collection

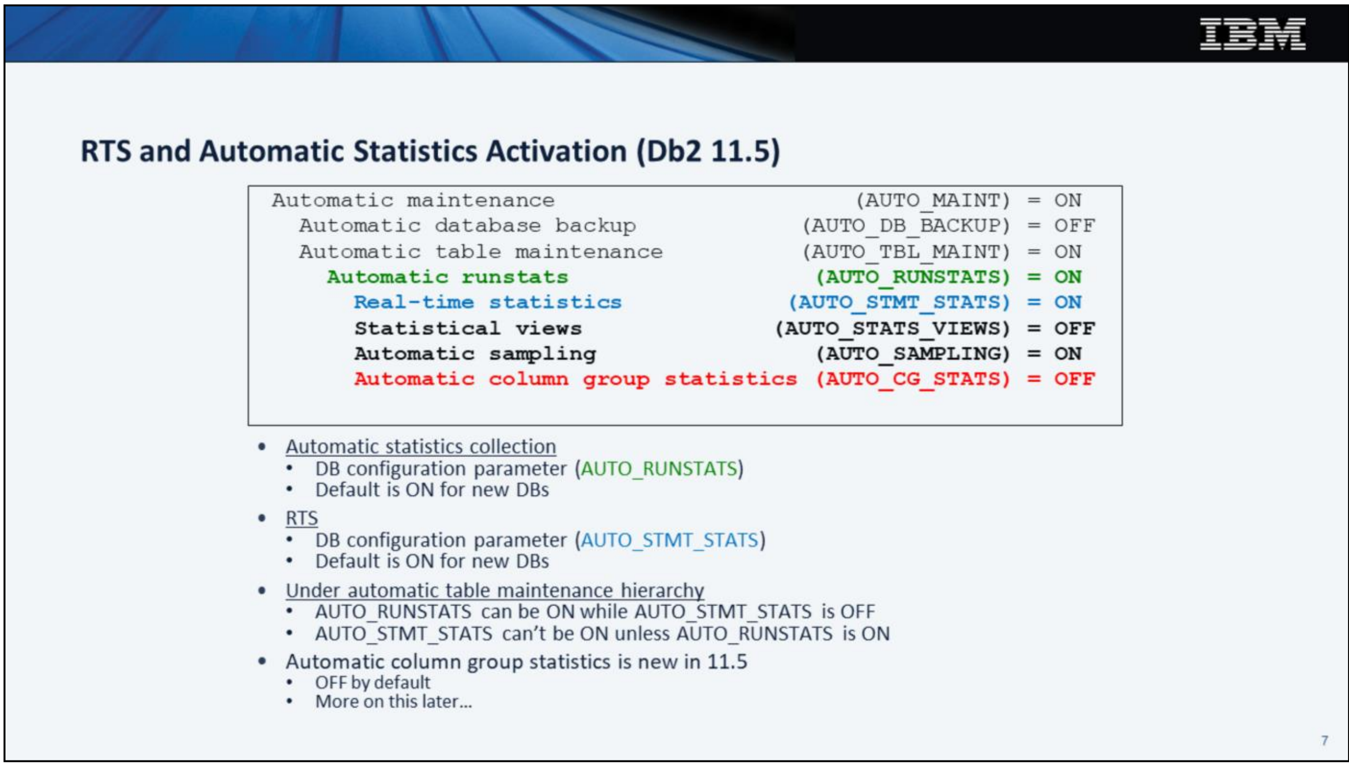
- Collects statistics in the background (asynchronous)
- Schedules statistic collection
  - Table evaluation occurs every 2 hrs
  - Based on amount of data change that has occurred since last statistics collection
- User control available for:
  - When collection occurs
  - Target tables

5



Real-time Statistics Collection

- Collects statistics at statement compilation time (synchronous)
- Table(s) identified based on statistical needs of the query and amount of data change
- Statistics can either be collected with RUNSTATS or 'fabricated'
- Statistics are immediately made available to other connections via a statistics cache
- Catalogs are not updated immediately to minimize overhead
- Requests immediate background (asynchronous) statistics collection
  - If statistics weren't 'fully' collected
- Why are RTS important?
  - Provides more current statistics to the query optimizer



RTS and Automatic Statistics Activation (Db2 11.5)

```
Automatic maintenance                          (AUTO_MAINT) = ON
  Automatic database backup                 (AUTO_DB_BACKUP) = OFF
  Automatic table maintenance               (AUTO_TBL_MAINT) = ON
    Automatic runstats                       (AUTO_RUNSTATS) = ON
      Real-time statistics                 (AUTO_STMT_STATS) = ON
      Statistical views                   (AUTO_STATS_VIEWS) = OFF
      Automatic sampling                     (AUTO_SAMPLING) = ON
      Automatic column group statistics (AUTO_CG_STATS) = OFF
```

- Automatic statistics collection
  - DB configuration parameter (AUTO_RUNSTATS)
  - Default is ON for new DBs
- RTS
  - DB configuration parameter (AUTO_STMT_STATS)
  - Default is ON for new DBs
- Under automatic table maintenance hierarchy
  - AUTO_RUNSTATS can be ON while AUTO_STMT_STATS is OFF
  - AUTO_STMT_STATS can't be ON unless AUTO_RUNSTATS is ON
- Automatic column group statistics is new in 11.5
  - OFF by default
  - More on this later…

7

RTS and Automatic Statistics Activation (Db2 11.1)

```
Automatic maintenance                          (AUTO_MAINT) = ON
  Automatic database backup                 (AUTO_DB_BACKUP) = OFF
  Automatic table maintenance               (AUTO_TBL_MAINT) = ON
    Automatic runstats                       (AUTO_RUNSTATS) = ON
      Real-time statistics                 (AUTO_STMT_STATS) = ON
      Statistical views                   (AUTO_STATS_VIEWS) = OFF
      Automatic sampling                     (AUTO_SAMPLING) = ON
```

- Automatic sampling (added Db2 10.1)
  - DB configuration parameter (AUTO_SAMPLING)
  - Default is ON for new DBs as of Db2 11.1

RTS and Automatic Statistics Activation (Db2 10.1)

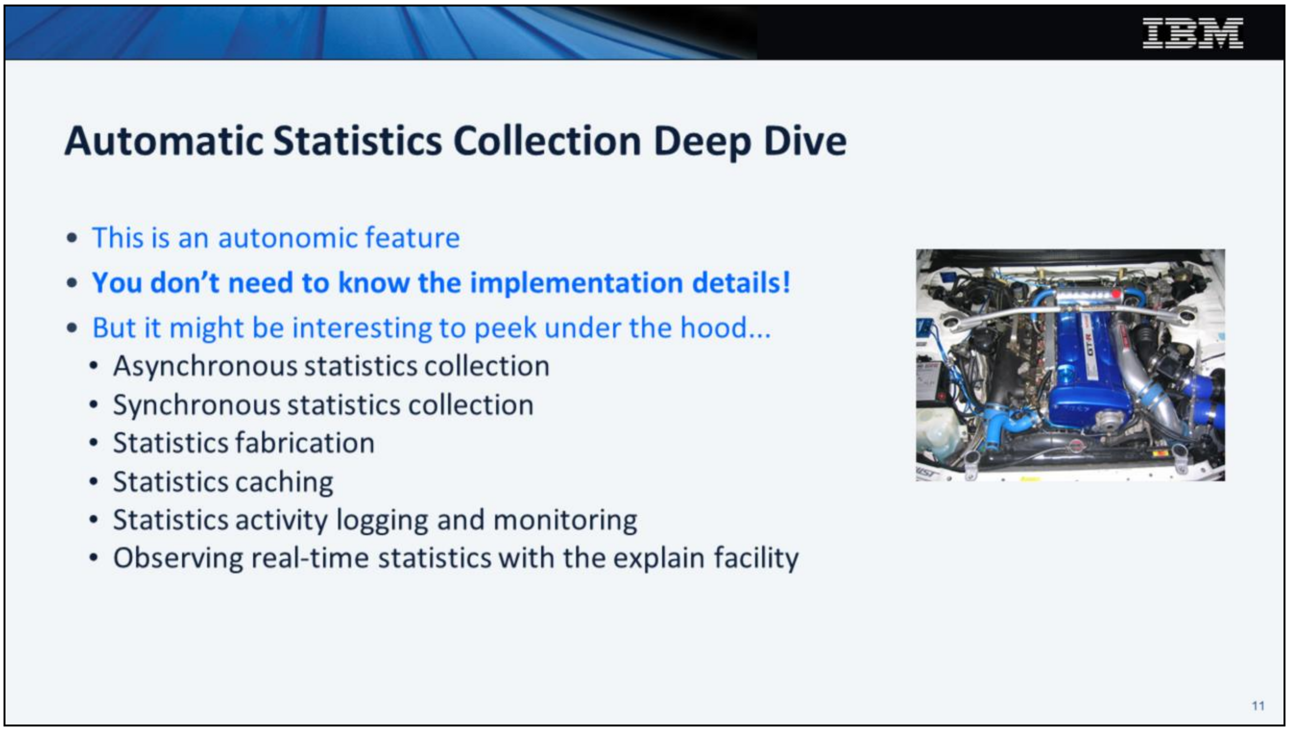
```
Automatic maintenance                          (AUTO_MAINT) = ON
  Automatic database backup                (AUTO_DB_BACKUP) = OFF
  Automatic table maintenance              (AUTO_TBL_MAINT) = ON
    Automatic runstats                    (AUTO_RUNSTATS) = ON
      Real-time statistics              (AUTO_STMT_STATS) = ON
      Statistical views                (AUTO_STATS_VIEWS) = OFF
      Automatic sampling                  (AUTO_SAMPLING) = OFF
```

- Automatics statistics collection for statistical views (Db2 10.1)
  - DB configuration parameter (AUTO_STATS_VIEWS)
  - Default is OFF for new DBs
- Automatic sampling (Db2 10.1)
  - DB configuration parameter (AUTO_SAMPLING)
  - Default is OFF for new DBs

RTS and Automatic Statistics Activation (Db2 9.7)

```
Automatic maintenance                             (AUTO_MAINT) = ON
   Automatic database backup                  (AUTO_DB_BACKUP) = OFF
   Automatic table maintenance                (AUTO_TBL_MAINT) = ON
     Automatic runstats                      (AUTO_RUNSTATS) = ON
       Automatic statement statistics  (AUTO_STMT_STATS) = ON
     Automatic statistics profiling    (AUTO_STATS_PROF) = OFF
       Automatic profile updates             (AUTO_PROF_UPD) = OFF
     Automatic reorganization                   (AUTO_REORG) = OFF
```

- Automatic statistics profiling is deprecated in Db2 10.5
- “Automatic statement statistics” label changed to “Real-time statistics” in Db2 10.1

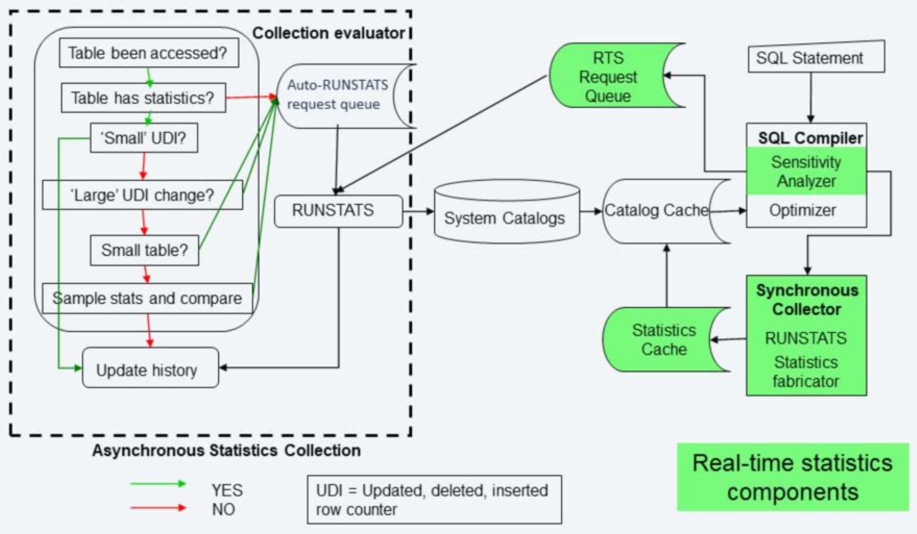


Automatic Statistics Collection Deep Dive

- This is an autonomic feature
- **You don't need to know the implementation details!**
- But it might be interesting to peek under the hood...
  - Asynchronous statistics collection
  - Synchronous statistics collection
  - Statistics fabrication
  - Statistics caching
  - Statistics activity logging and monitoring
  - Observing real-time statistics with the explain facility

11

Automatic Statistics Collection Architecture



12

The box on the left represents the background automatic statistics collection process. This process runs asynchronously to DB operations such as SQL statement execution. The statistics evaluator determines if statistics should be collected and then initiates a 'background' RUNSTATS. The collected statistics are stored in the system catalogs. The statistics for a particular table are loaded from the system catalogs into a catalog cache, whenever they are referenced by the SQL Compiler.

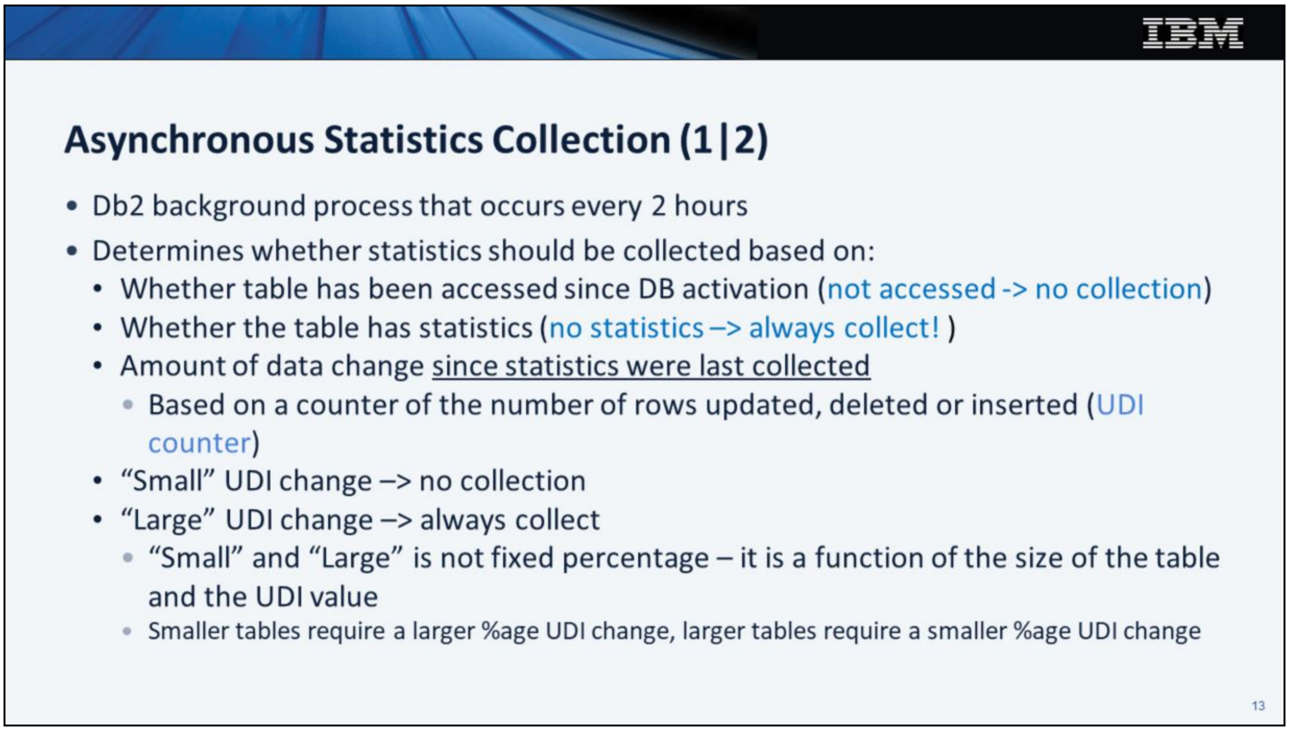
Real-time statistics components are represented by the green boxes. These boxes will be described in more detail on subsequent pages.

Statistics Cache – Synchronously collected statistics or fabricated statistics are stored here, until they are written to the system catalogs. The Catalog Cache loads new statistics from here, if they have not yet been written to the system catalogs

Sensitivity Analyzer – This component of the optimizer determines whether a statement requires statistics, what statistics it requires and how to collect them.

Synchronous Collector – This component collects statistics synchronously (as part of SQL statement compilation) by either invoking RUNSTATS or by fabricating statistics.

Asynchronous Request Queue – If the Sensitivity Analyzer determines that statistics should be collected asynchronously, a request is added to this queue. Every 5 minutes, a background process will be invoked to process requests in this queue.

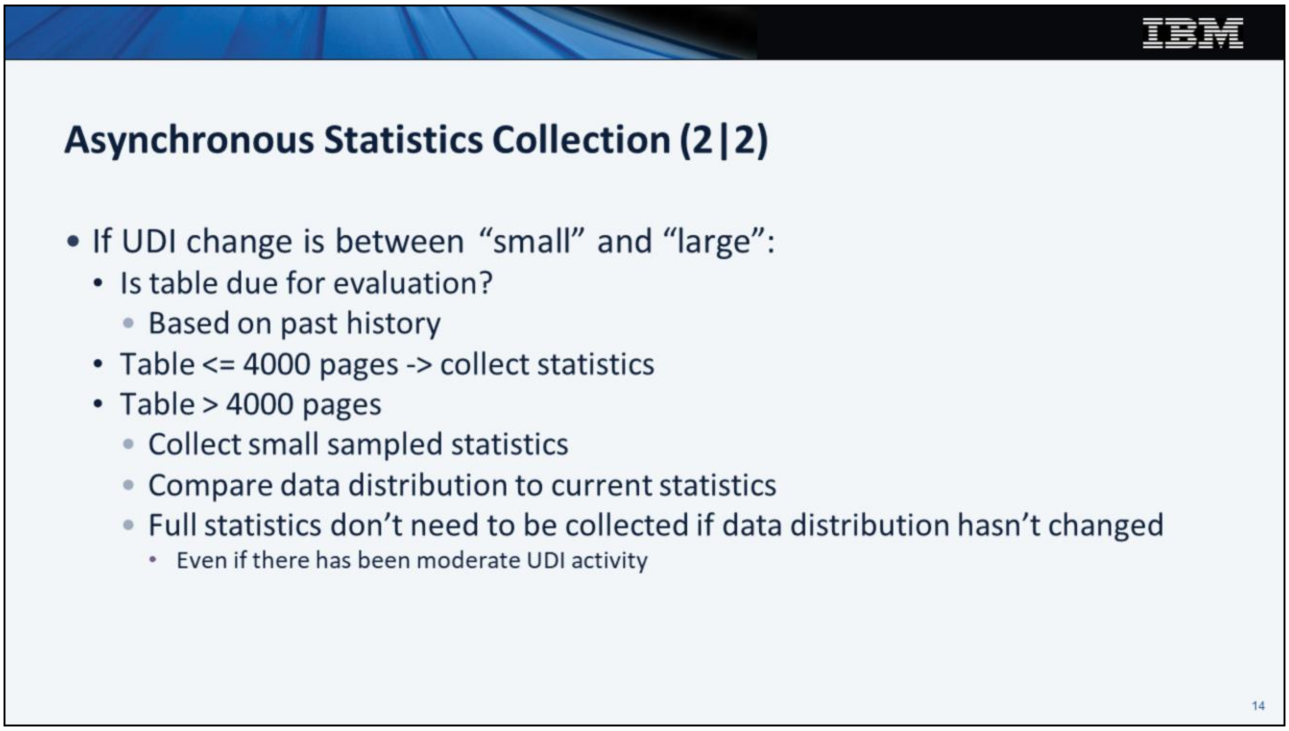


IBM

Asynchronous Statistics Collection (1|2)

- Db2 background process that occurs every 2 hours
- Determines whether statistics should be collected based on:
  - Whether table has been accessed since DB activation (not accessed -> no collection)
  - Whether the table has statistics (no statistics –> always collect!)
  - Amount of data change <u>since statistics were last collected</u>
    - Based on a counter of the number of rows updated, deleted or inserted (UDI counter)
  - “Small” UDI change –> no collection
  - “Large” UDI change –> always collect
    - “Small” and “Large” is not fixed percentage – it is a function of the size of the table and the UDI value
    - Smaller tables require a larger %age UDI change, larger tables require a smaller %age UDI change

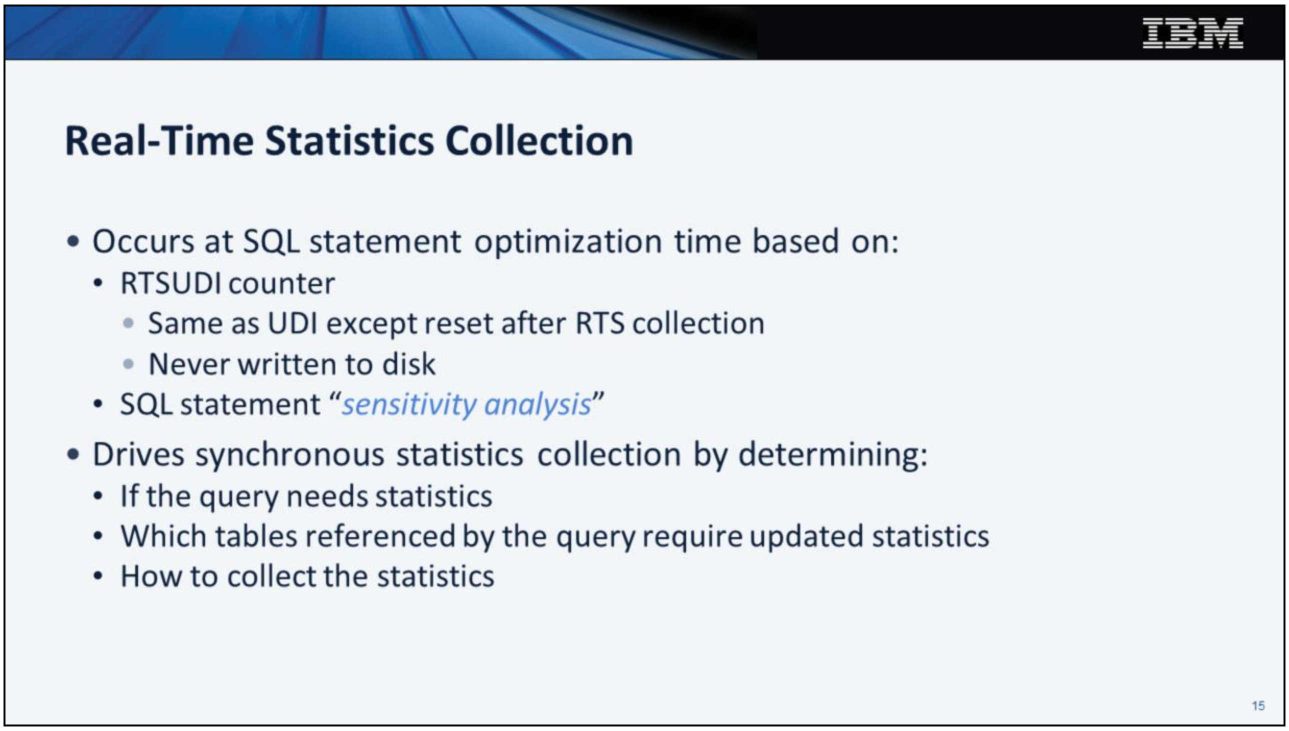
13



Asynchronous Statistics Collection (2|2)

- If UDI change is between "small" and "large":
  - Is table due for evaluation?
    - Based on past history
  - Table <= 4000 pages -> collect statistics
  - Table > 4000 pages
    - Collect small sampled statistics
    - Compare data distribution to current statistics
    - Full statistics don't need to be collected if data distribution hasn't changed
      - Even if there has been moderate UDI activity

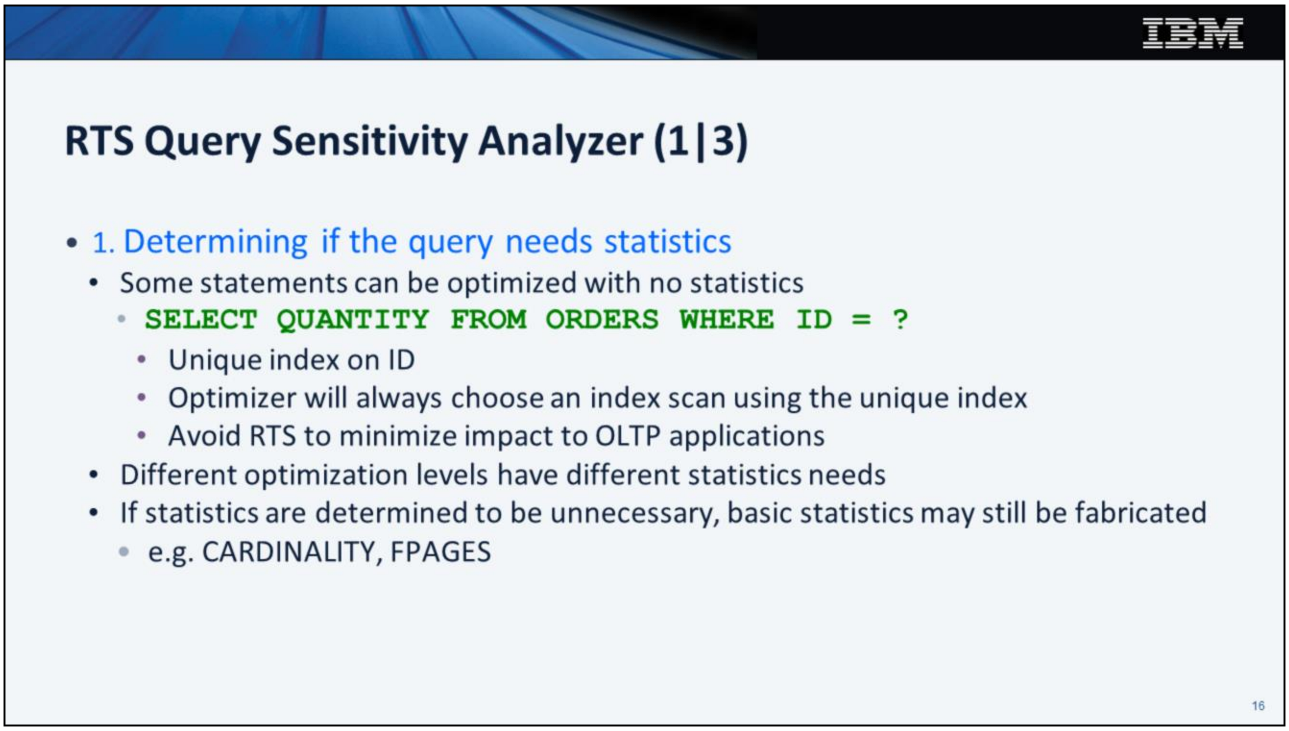
14



Real-Time Statistics Collection

- Occurs at SQL statement optimization time based on:
  - RTSUDI counter
    - Same as UDI except reset after RTS collection
    - Never written to disk
  - SQL statement "*sensitivity analysis*"
- Drives synchronous statistics collection by determining:
  - If the query needs statistics
  - Which tables referenced by the query require updated statistics
  - How to collect the statistics

15

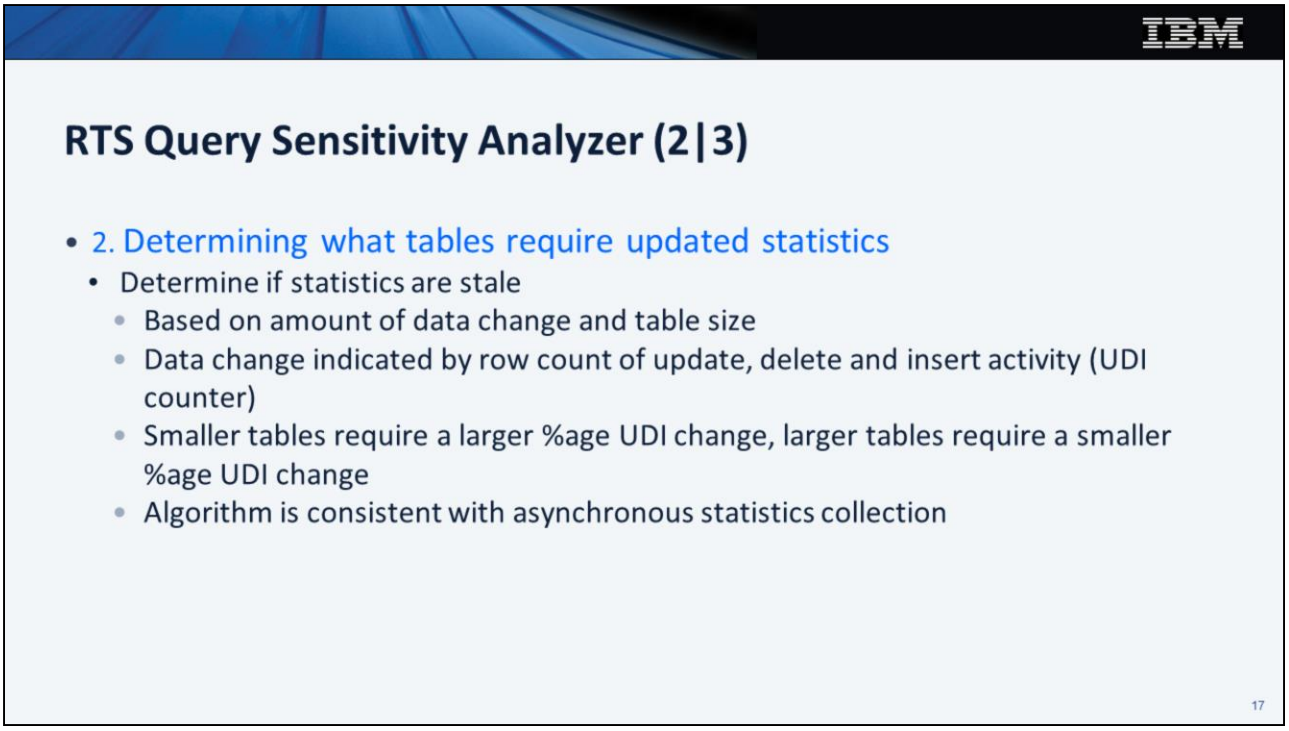


RTS Query Sensitivity Analyzer (1|3)

- 1. Determining if the query needs statistics
  - Some statements can be optimized with no statistics
    - `SELECT QUANTITY FROM ORDERS WHERE ID = ?`
      - Unique index on ID
      - Optimizer will always choose an index scan using the unique index
      - Avoid RTS to minimize impact to OLTP applications
  - Different optimization levels have different statistics needs
  - If statistics are determined to be unnecessary, basic statistics may still be fabricated
    - e.g. CARDINALITY, FPAGES

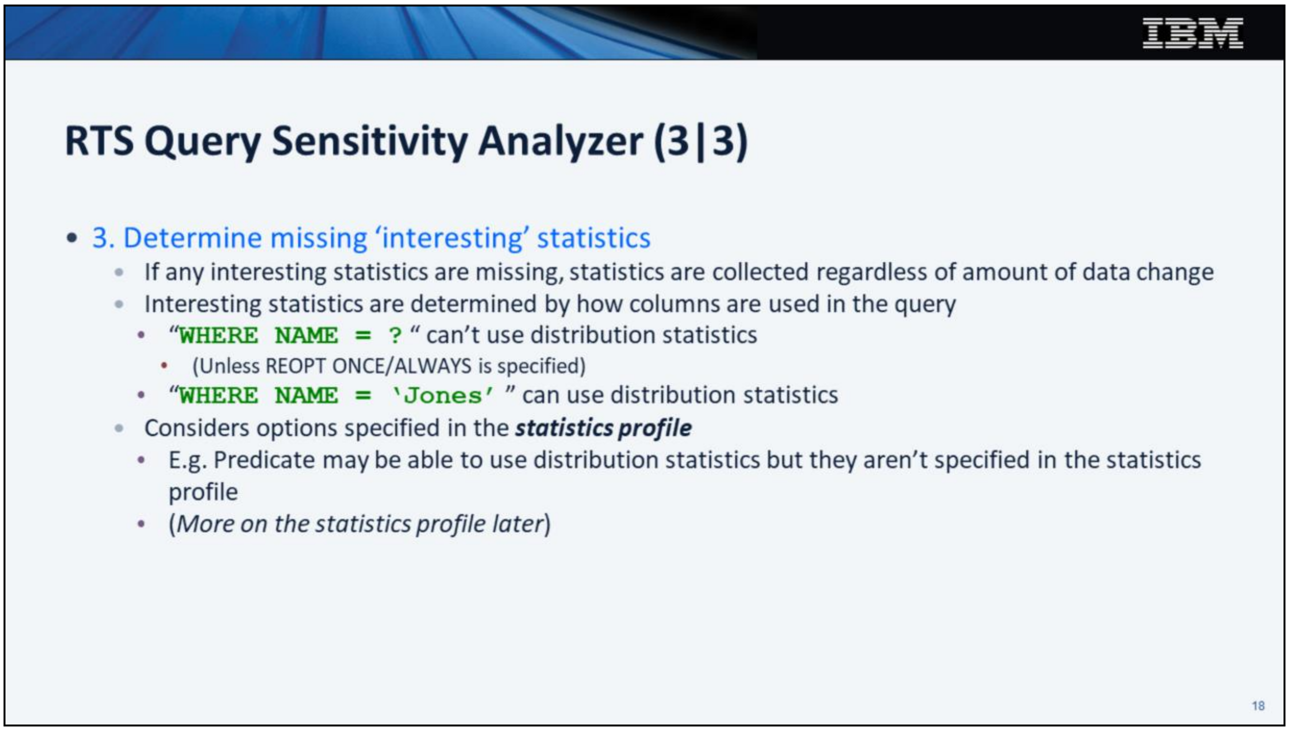
16

One of the goals of the sensitivity analyzer is to minimize the amount of synchronous statistics collection for OLTP applications. The overhead of synchronous collection may be more apparent to OLTP applications, however they may be able to tolerate less accurate statistics.



RTS Query Sensitivity Analyzer (2|3)

- 2. Determining what tables require updated statistics
  - Determine if statistics are stale
    - Based on amount of data change and table size
    - Data change indicated by row count of update, delete and insert activity (UDI counter)
    - Smaller tables require a larger %age UDI change, larger tables require a smaller %age UDI change
    - Algorithm is consistent with asynchronous statistics collection



RTS Query Sensitivity Analyzer (3|3)

- 3. Determine missing 'interesting' statistics
  - If any interesting statistics are missing, statistics are collected regardless of amount of data change
  - Interesting statistics are determined by how columns are used in the query
    - "`WHERE NAME = ?`" can't use distribution statistics
      - (Unless REOPT ONCE/ALWAYS is specified)
    - "`WHERE NAME = 'Jones'`" can use distribution statistics
  - Considers options specified in the ***statistics profile***
    - E.g. Predicate may be able to use distribution statistics but they aren't specified in the statistics profile
    - (*More on the statistics profile later*)

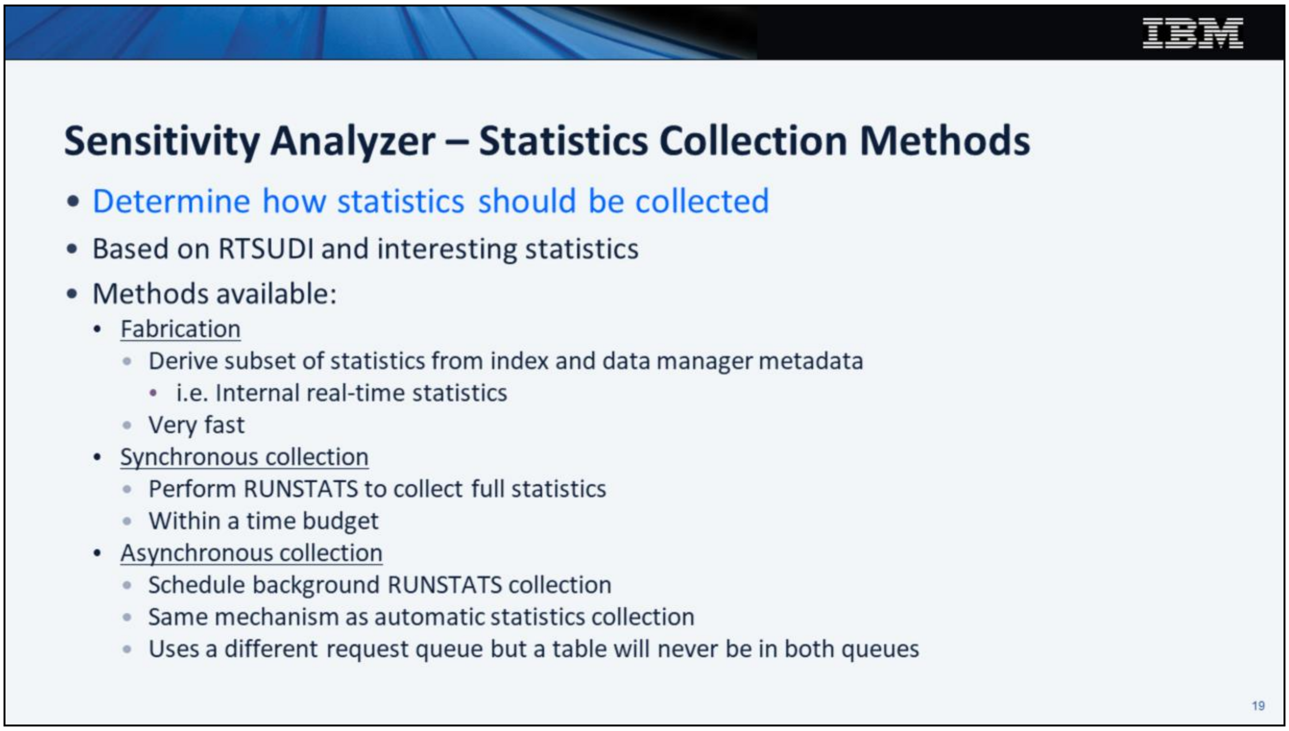
18

More details on statistical profiles and RTS are provided later in this presentation. Repeating an overview here:

The RUNSTATS utility provides a statistical profile facility to:
1) register a statistical profile, while optionally gathering statistics
2) modify an existing statistical profile stored in the catalogs, while optionally gathering statistics
3) repeatedly gather statistics on the table using an already registered statistics profile for that particular table.

This may be convenient for multiple scripts that need to perform RUNSTATS on the same set of tables so the RUNSTATS options don't need to be repeated in every script. Additionally, the statistical profile can be specified for LOAD so that consistent RUNSTATS options can be used and don't need to be repeated on the LOAD command.

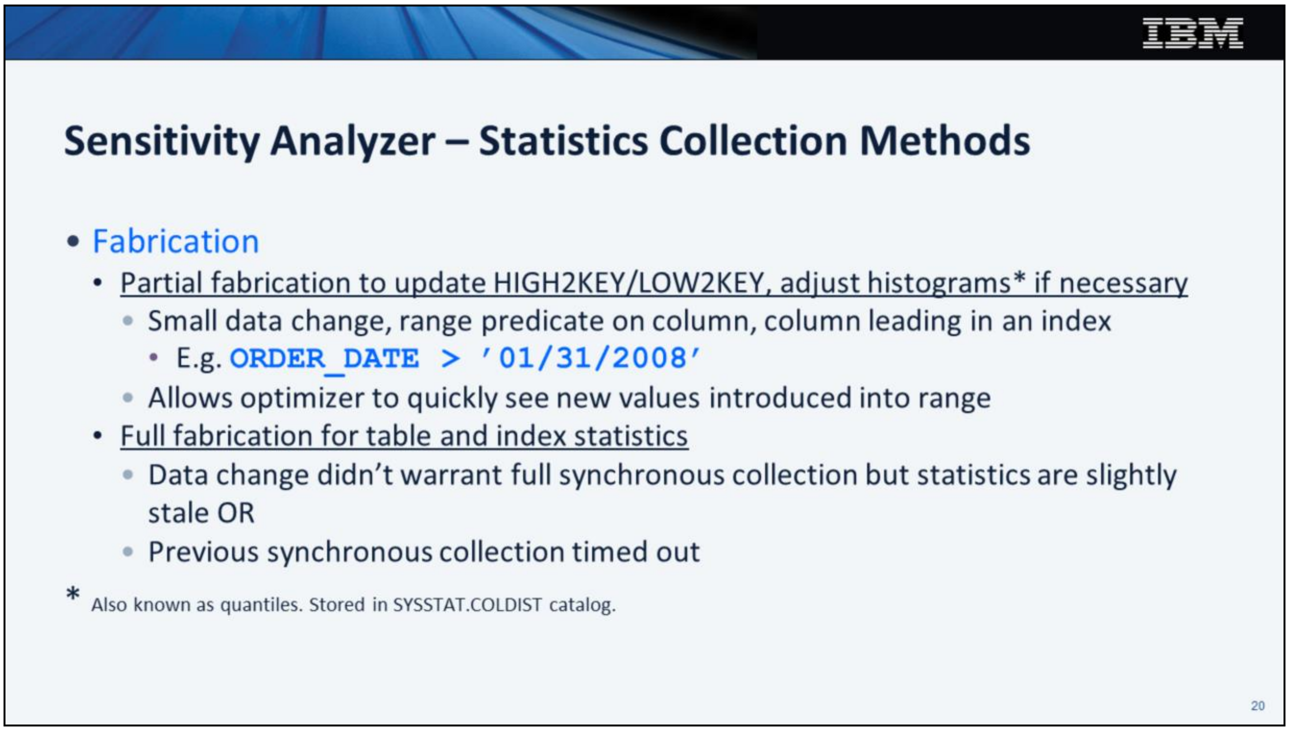
When a statistical profile is registered, a RUNSTATS command string corresponding to that profile is at the same time built and stored in the STATISTICS_PROFILE column of the catalog table SYSIBM.SYSTABLES. An internal version of the profile is also maintained in the system catalogs in SYSTABLES.PACKED_DESC.



Sensitivity Analyzer – Statistics Collection Methods

- Determine how statistics should be collected
- Based on RTSUDI and interesting statistics
- Methods available:
  - Fabrication
    - Derive subset of statistics from index and data manager metadata
      - i.e. Internal real-time statistics
    - Very fast
  - Synchronous collection
    - Perform RUNSTATS to collect full statistics
    - Within a time budget
  - Asynchronous collection
    - Schedule background RUNSTATS collection
    - Same mechanism as automatic statistics collection
    - Uses a different request queue but a table will never be in both queues

19



Sensitivity Analyzer – Statistics Collection Methods

- Fabrication
  - Partial fabrication to update HIGH2KEY/LOW2KEY, adjust histograms* if necessary
    - Small data change, range predicate on column, column leading in an index
      - E.g. `ORDER_DATE > '01/31/2008'`
    - Allows optimizer to quickly see new values introduced into range
  - Full fabrication for table and index statistics
    - Data change didn't warrant full synchronous collection but statistics are slightly stale OR
    - Previous synchronous collection timed out

* Also known as quantiles. Stored in SYSSTAT.COLDIST catalog.

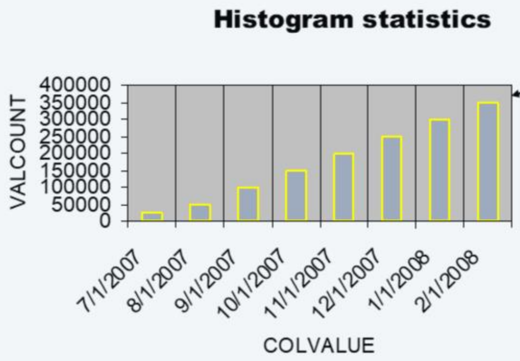
20

Some applications issue queries with search conditions looking for newly inserted data. However, if the statistics aren't current, they may not include the newly inserted data. For example, the HIGH2KEY statistic for the ORDER_DATE column may be '01/15/2008' but the query is looking for newer order dates. Consequently, the optimizer will assume few rows qualify. Partial statistics fabrication makes the optimizer aware of the newer order dates, providing there is an index that includes ORDER_DATE as a leading column. The index will be quickly probed to determine the second highest and lowest values, which are then used to update the HIGH2KEY and LOW2KEY statistics. Any existing histogram statistics will be adjusted too. Partial fabrication occurs more frequently than full fabrication or synchronous collection, however it has lower overhead. More details on full fabrication are on the next page.

Partial statistics fabrication

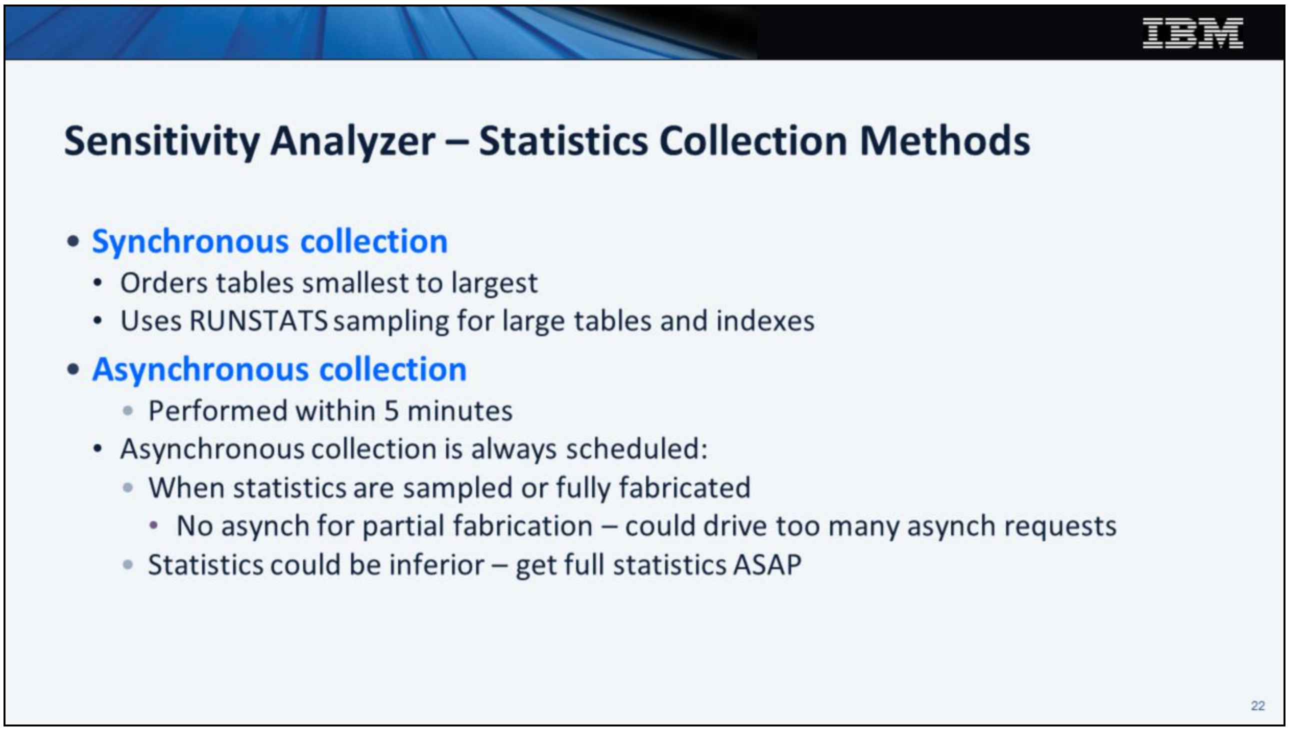
WHERE ORDER_DATE > '01/31/2008'

HIGH2KEY = ~~02/01/2008~~ -> 03/01/2008



VALCOUNT is the number of rows whose value is less than or equal to COLVALUE

The original value of the highest histogram entry is updated by probing an index on the column to determine the current highest value.



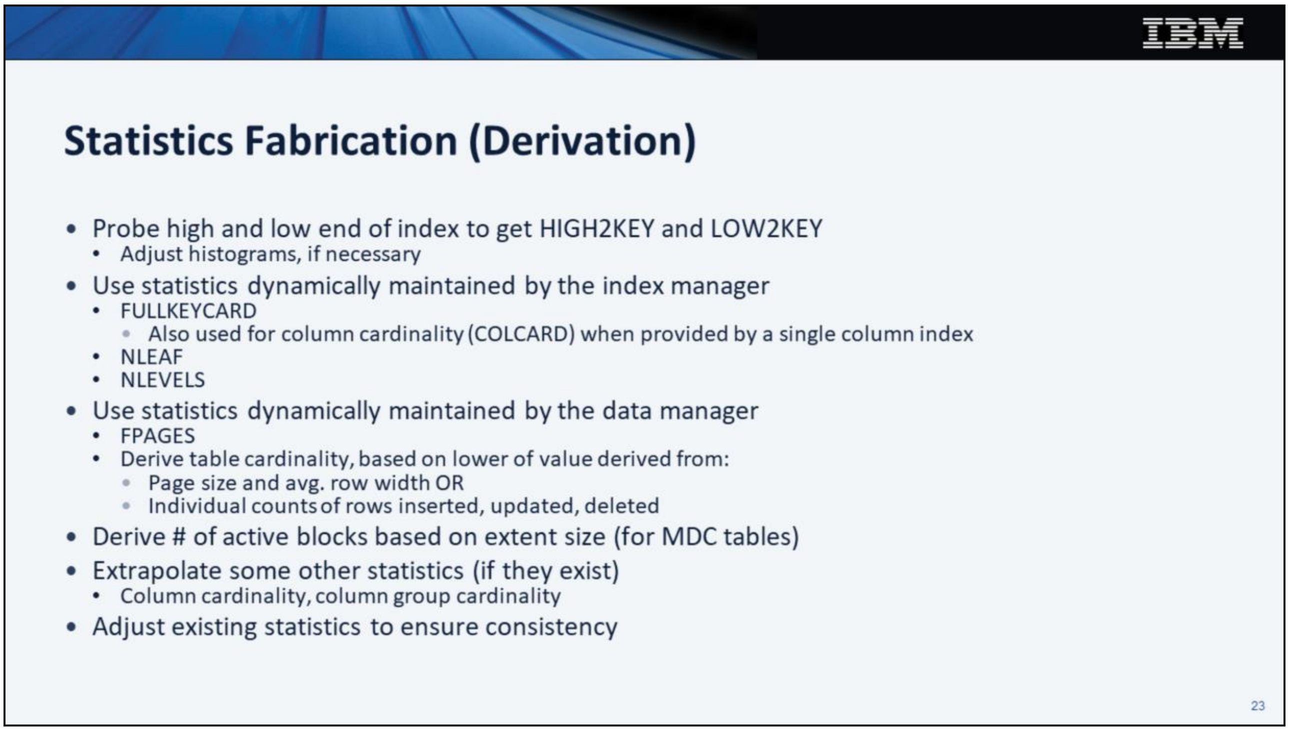
Sensitivity Analyzer – Statistics Collection Methods

- **Synchronous collection**
  - Orders tables smallest to largest
  - Uses RUNSTATS sampling for large tables and indexes
- **Asynchronous collection**
    - Performed within 5 minutes
  - Asynchronous collection is always scheduled:
    - When statistics are sampled or fully fabricated
      - No asynch for partial fabrication – could drive too many asynch requests
    - Statistics could be inferior – get full statistics ASAP

22

Synchronous statistics collection will sample tables that have more than 4000 pages. This value and algorithm may change in the future. Synchronous collection orders the tables smallest to largest in order to prevent small tables from being starved by larger tables.

Asynchronous collection is always scheduled if the statistics are sampled or fully fabricated, to ensure the most accurate statistics are stored in the system catalogs. The in-memory RTS are intended to provide the query optimizer with something more accurate than the stale statistics, however the statistics stored in the system catalogs should correspond to the options specified in the statistical profile, whether it is the default profile or a profile provided by the user.



Statistics Fabrication (Derivation)

- Probe high and low end of index to get HIGH2KEY and LOW2KEY
  - Adjust histograms, if necessary
- Use statistics dynamically maintained by the index manager
  - FULLKEYCARD
    - Also used for column cardinality (COLCARD) when provided by a single column index
  - NLEAF
  - NLEVELS
- Use statistics dynamically maintained by the data manager
  - FPAGES
  - Derive table cardinality, based on lower of value derived from:
    - Page size and avg. row width OR
    - Individual counts of rows inserted, updated, deleted
- Derive # of active blocks based on extent size (for MDC tables)
- Extrapolate some other statistics (if they exist)
  - Column cardinality, column group cardinality
- Adjust existing statistics to ensure consistency

23

There are a number of ways that statistics can be fabricated:

•The index can be probed to get the 2nd highest and lowest values in order to fabricate HIGH2KEY and LOW2KEY

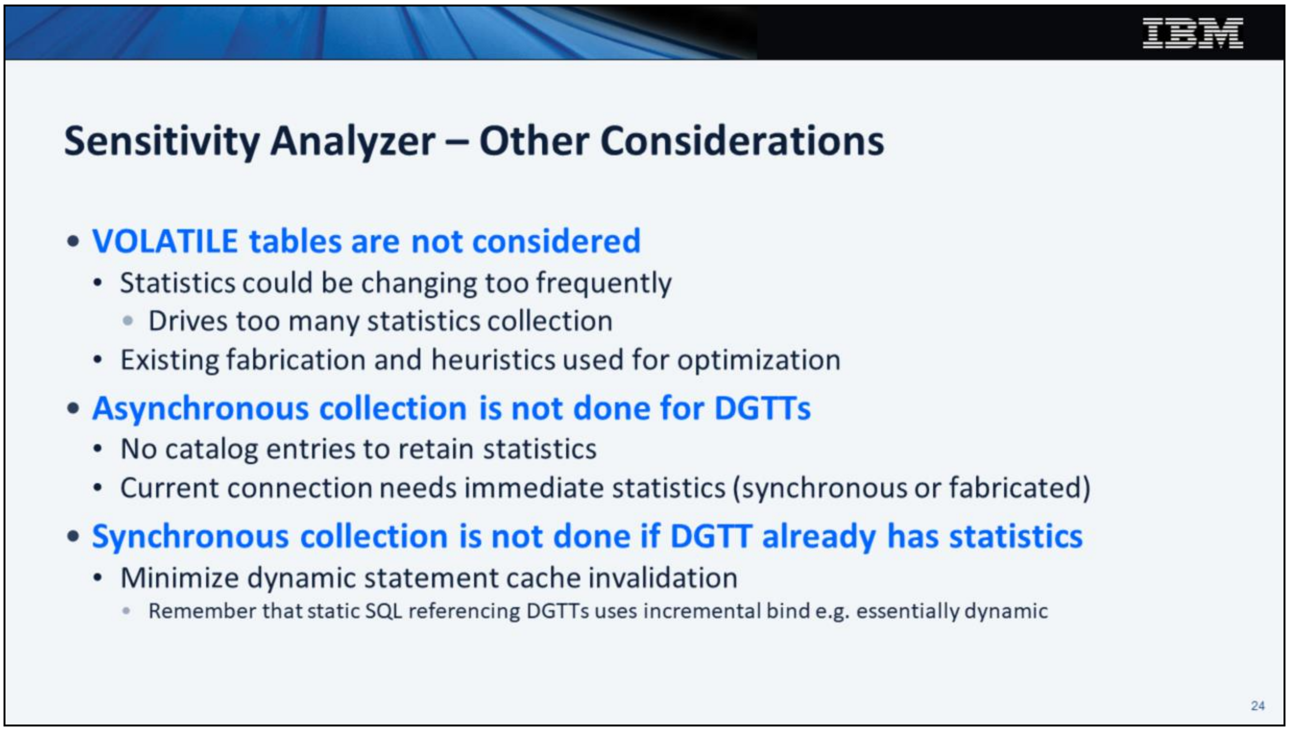
•The Index Manager component maintains the full key cardinality (FULLKEYCARD), the number of leaf pages (NLEAF) and the number levels (NLEVELS). These statistics can be used directly by the optimizer.

•The Data Manager component maintains the number of file pages (FPAGES). This value can be used directly by the optimizer. The table’s cardinality can be derived from FPAGES because the page size and the average row width are known. Also, the Data Manager maintains individual counts for the number or rows inserted, updated and deleted. These counts can also be used to compute the table cardinality. Statistics fabrication uses the lower value of these two methods.

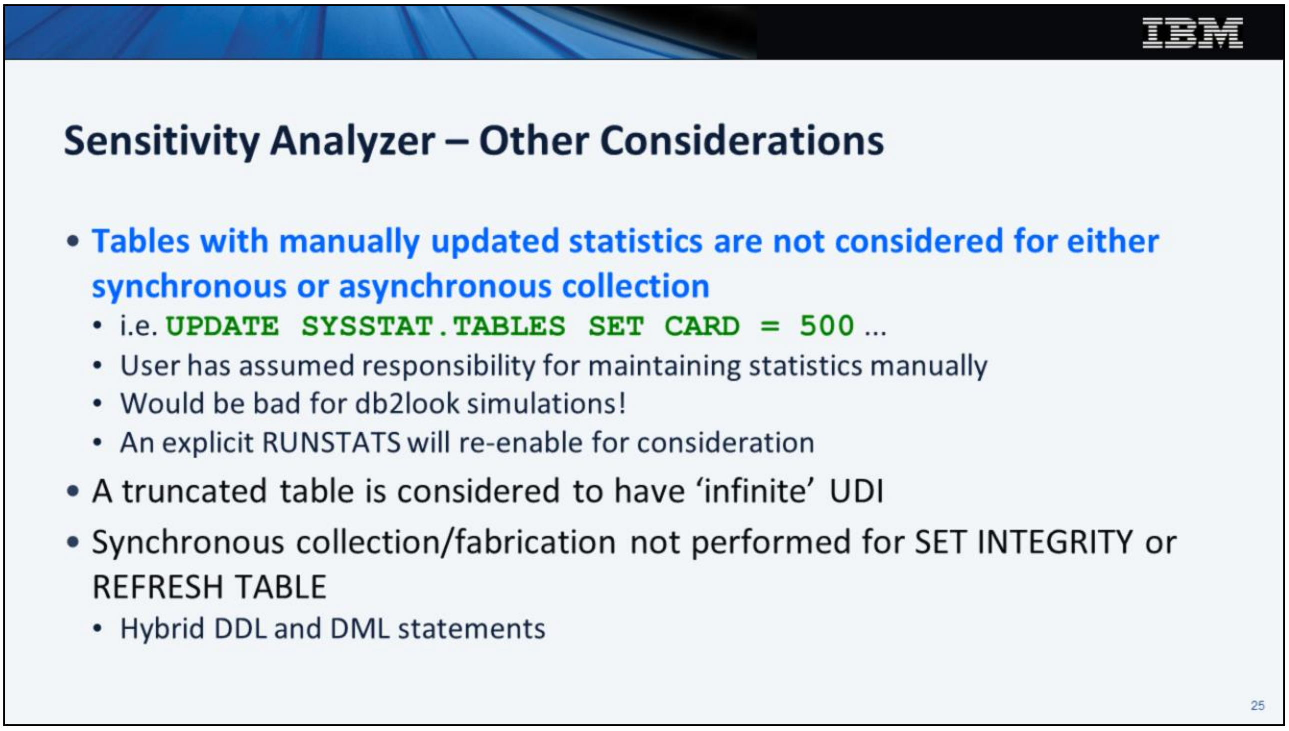
•The number of active blocks for multi-dimensional clustered (MDC) tables can be derived knowing the extent size and FPAGES.

•Other statistics that can’t be derived can be extrapolated from existing statistics.

The statistics fabrication process will ensure consistency between all derived and extrapolated statistics.


Sensitivity Analyzer – Other Considerations

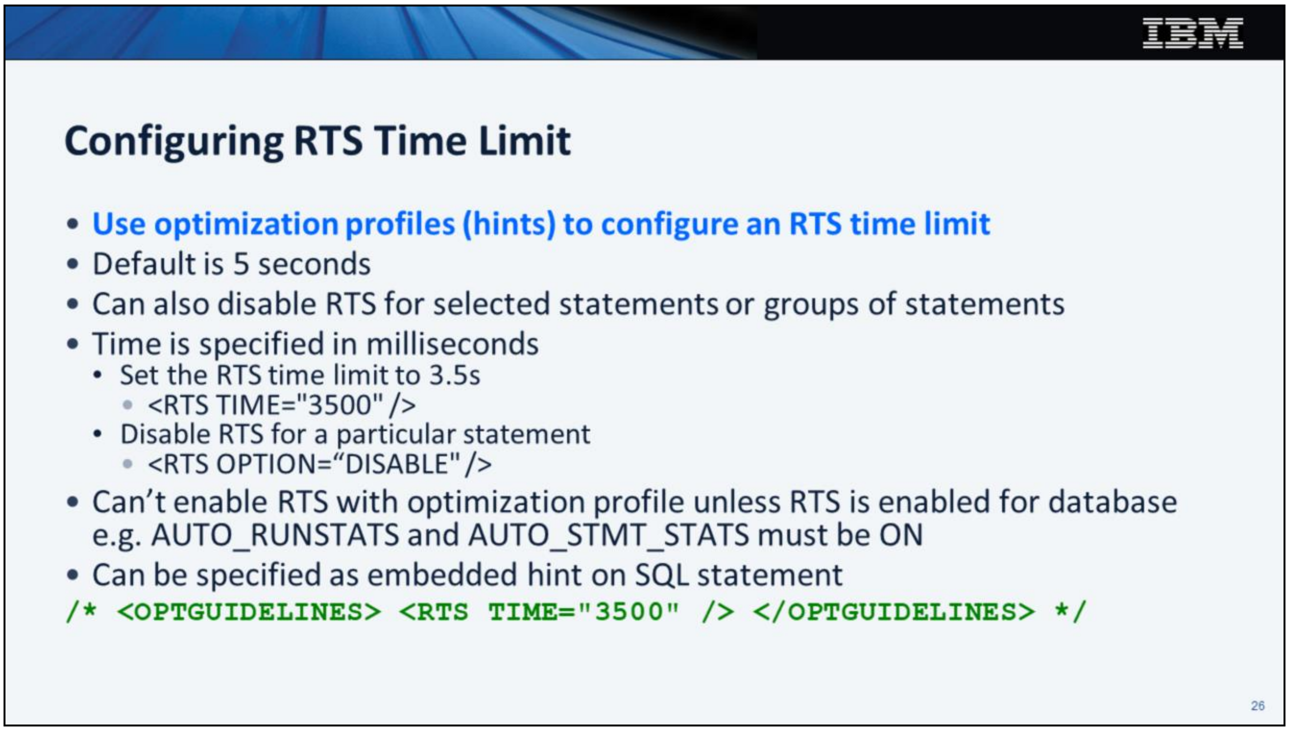
- **VOLATILE tables are not considered**
  - Statistics could be changing too frequently
    - Drives too many statistics collection
  - Existing fabrication and heuristics used for optimization
- **Asynchronous collection is not done for DGTTs**
  - No catalog entries to retain statistics
  - Current connection needs immediate statistics (synchronous or fabricated)
- **Synchronous collection is not done if DGTT already has statistics**
  - Minimize dynamic statement cache invalidation
    - Remember that static SQL referencing DGTTs uses incremental bind e.g. essentially dynamic



Sensitivity Analyzer – Other Considerations

- **Tables with manually updated statistics are not considered for either synchronous or asynchronous collection**
  - i.e. `UPDATE SYSSTAT.TABLES SET CARD = 500` ...
  - User has assumed responsibility for maintaining statistics manually
  - Would be bad for db2look simulations!
  - An explicit RUNSTATS will re-enable for consideration
- A truncated table is considered to have 'infinite' UDI
- Synchronous collection/fabrication not performed for SET INTEGRITY or REFRESH TABLE
  - Hybrid DDL and DML statements

25



Configuring RTS Time Limit

- **Use optimization profiles (hints) to configure an RTS time limit**
- Default is 5 seconds
- Can also disable RTS for selected statements or groups of statements
- Time is specified in milliseconds
  - Set the RTS time limit to 3.5s
    - <RTS TIME="3500" />
  - Disable RTS for a particular statement
    - <RTS OPTION="DISABLE" />
- Can't enable RTS with optimization profile unless RTS is enabled for database e.g. AUTO_RUNSTATS and AUTO_STMT_STATS must be ON
- Can be specified as embedded hint on SQL statement

```
/* <OPTGUIDELINES> <RTS TIME="3500" /> </OPTGUIDELINES> */
```

26

RTS requests

The RTS general request element can be used to enable or disable real-time statistics collection. It can also be used to limit the amount of time taken by real-time statistics collection. For certain queries or workloads, it may be desirable to disable or limit the time spent on real-time statistics collection to avoid extra overhead at statement compilation time.

Description

The RTS general request element has two optional attributes.

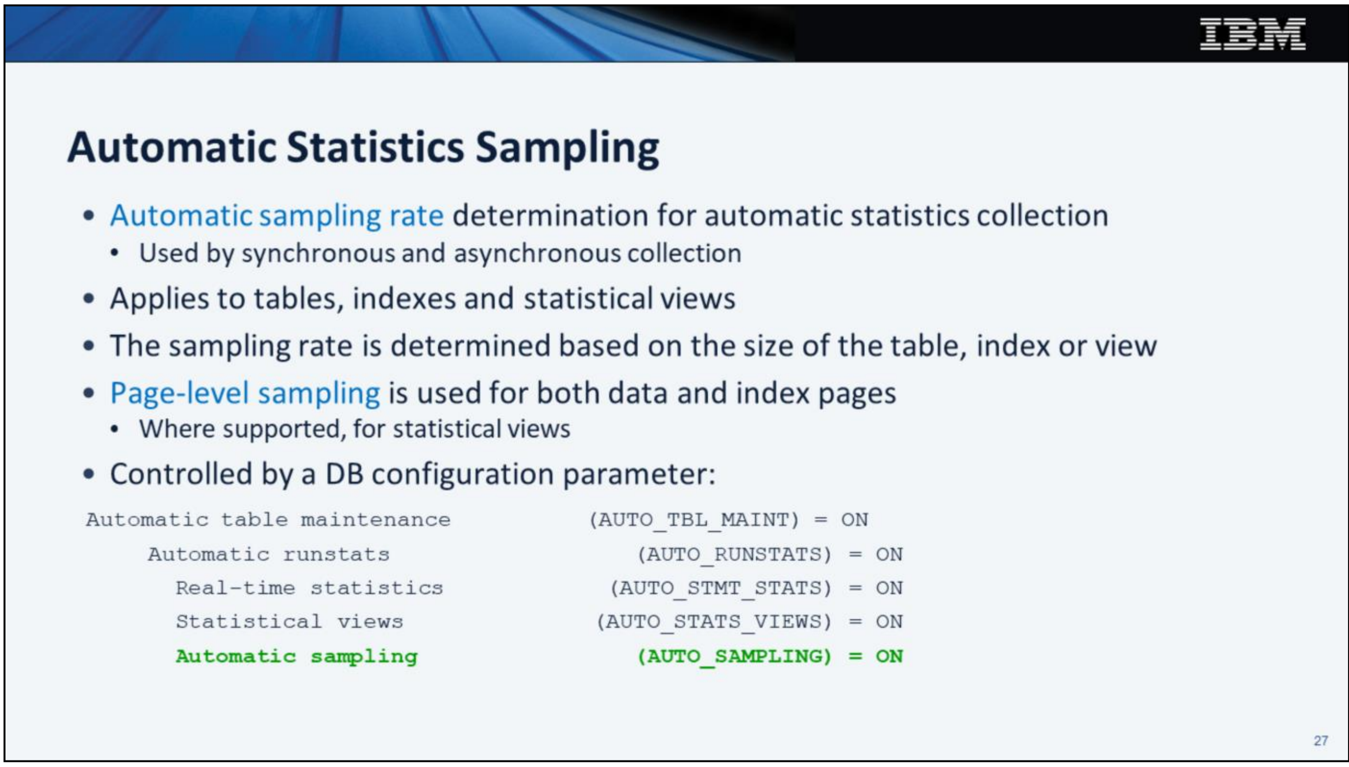
The OPTION attribute is used to enable or disable real-time statistics collection. It can take the values ENABLE or DISABLE. ENABLE is the default if no option is specified.

The TIME attribute specifies the maximum amount of time in milliseconds to be spent on real-time statistics collection at statement compilation time, for a single statement.

If ENABLE is specified for the OPTION attribute, automatic statistics collection and real-time statistics must be enabled by their corresponding configuration parameters. Otherwise, the optimization guideline will not be applied, and you will get warning message SQL0437W (reason code 13).

For example, the following RTS request enables real-time statistics collection and limits real-time statistics collection time to 3.5 seconds.

<RTS OPTION="ENABLE" TIME="3500" />

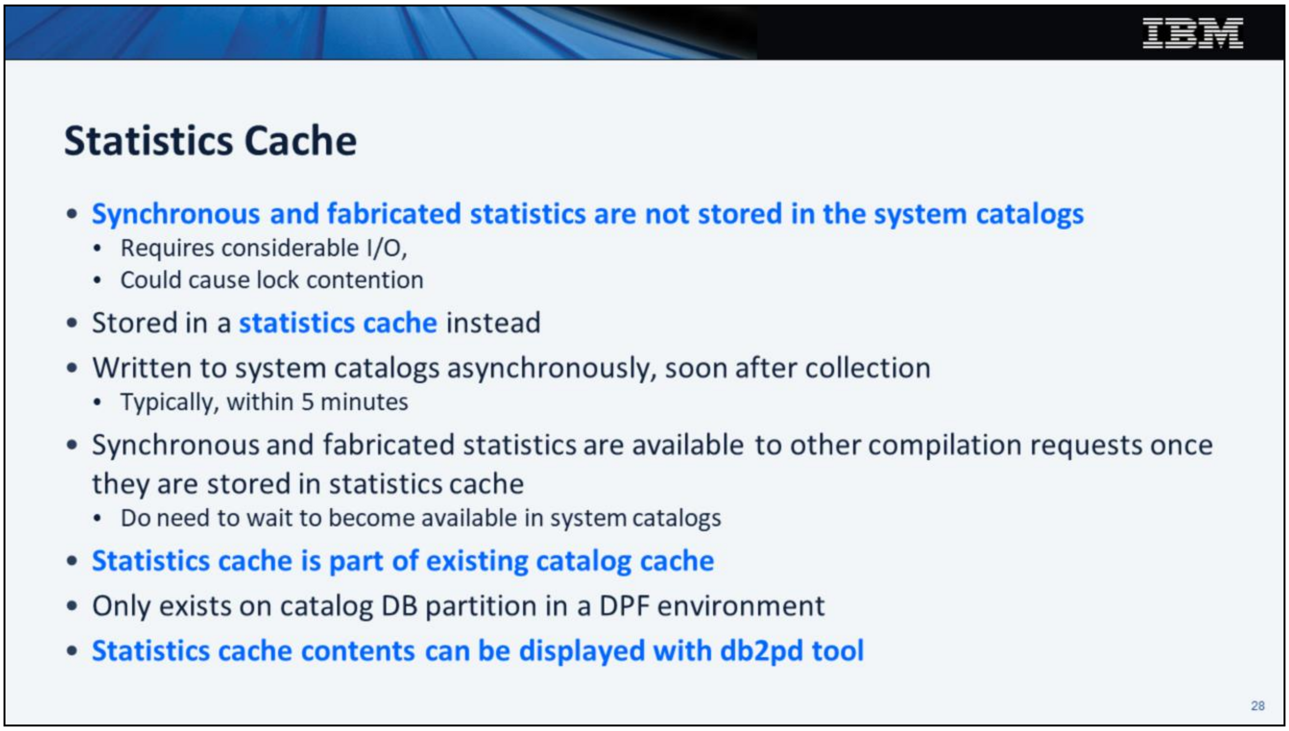


Automatic Statistics Sampling

- Automatic sampling rate determination for automatic statistics collection
  - Used by synchronous and asynchronous collection
- Applies to tables, indexes and statistical views
- The sampling rate is determined based on the size of the table, index or view
- Page-level sampling is used for both data and index pages
  - Where supported, for statistical views
- Controlled by a DB configuration parameter:

```
Automatic table maintenance          (AUTO_TBL_MAINT) = ON
    Automatic runstats                  (AUTO_RUNSTATS) = ON
      Real-time statistics            (AUTO_STMT_STATS) = ON
      Statistical views              (AUTO_STATS_VIEWS) = ON
      Automatic sampling                (AUTO_SAMPLING) = ON
```

27

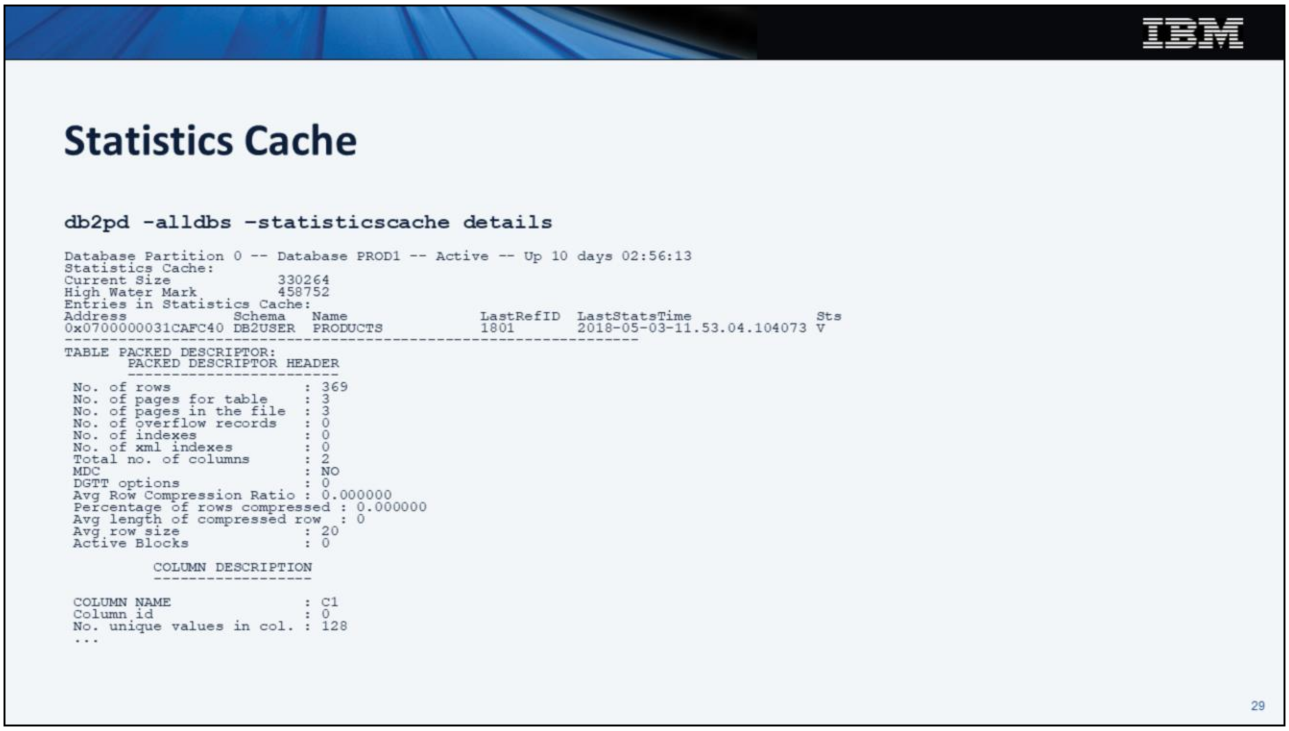


Statistics Cache

- **Synchronous and fabricated statistics are not stored in the system catalogs**
  - Requires considerable I/O,
  - Could cause lock contention
- Stored in a **statistics cache** instead
- Written to system catalogs asynchronously, soon after collection
  - Typically, within 5 minutes
- Synchronous and fabricated statistics are available to other compilation requests once they are stored in statistics cache
  - Do need to wait to become available in system catalogs
- **Statistics cache is part of existing catalog cache**
- Only exists on catalog DB partition in a DPF environment
- **Statistics cache contents can be displayed with db2pd tool**

28

A statistics cache is used to make synchronously-collected statistics available to all queries. This cache is part of the catalog cache. In a partitioned database environment, this cache resides only on the catalog database partition. The catalog cache can store multiple entries for the same SYSTABLES object, which increases the size of the catalog cache on all database partitions. Consider increasing the value of the **catalogcache_sz** database configuration parameter when real-time statistics collection is enabled



Statistics Cache

```
db2pd -alldbs -statisticscache details
```

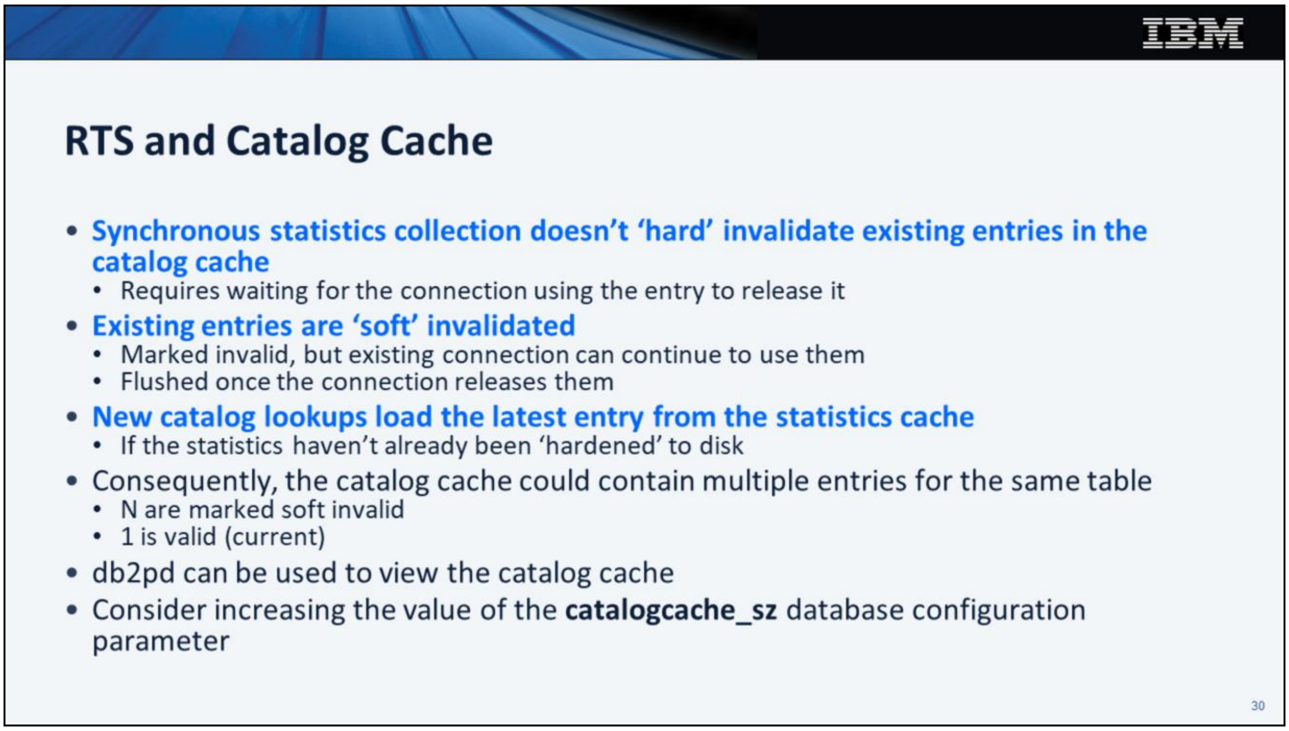
```
Database Partition 0 -- Database PROD1 -- Active -- Up 10 days 02:56:13
Statistics Cache:
Current Size            330264
High Water Mark         458752
Entries in Statistics Cache:
Address            Schema   Name                LastRefID  LastStatsTime              Sts
0x0700000031CAFC40 DB2USER  PRODUCTS            1801       2018-05-03-11.53.04.104073 V
-----------------------------------------------------------------
TABLE PACKED DESCRIPTOR:
       PACKED DESCRIPTOR HEADER
       ------------------------
 No. of rows                 : 369
 No. of pages for table      : 3
 No. of pages in the file    : 3
 No. of overflow records     : 0
 No. of indexes              : 0
 No. of xml indexes          : 0
 Total no. of columns        : 2
 MDC                         : NO
 DGTT options                : 0
 Avg Row Compression Ratio : 0.000000
 Percentage of rows compressed : 0.000000
 Avg length of compressed row  : 0
 Avg row size                : 20
 Active Blocks               : 0

          COLUMN DESCRIPTION
          ------------------

 COLUMN NAME                 : C1
 Column id                   : 0
 No. unique values in col. : 128
 ...
```

29

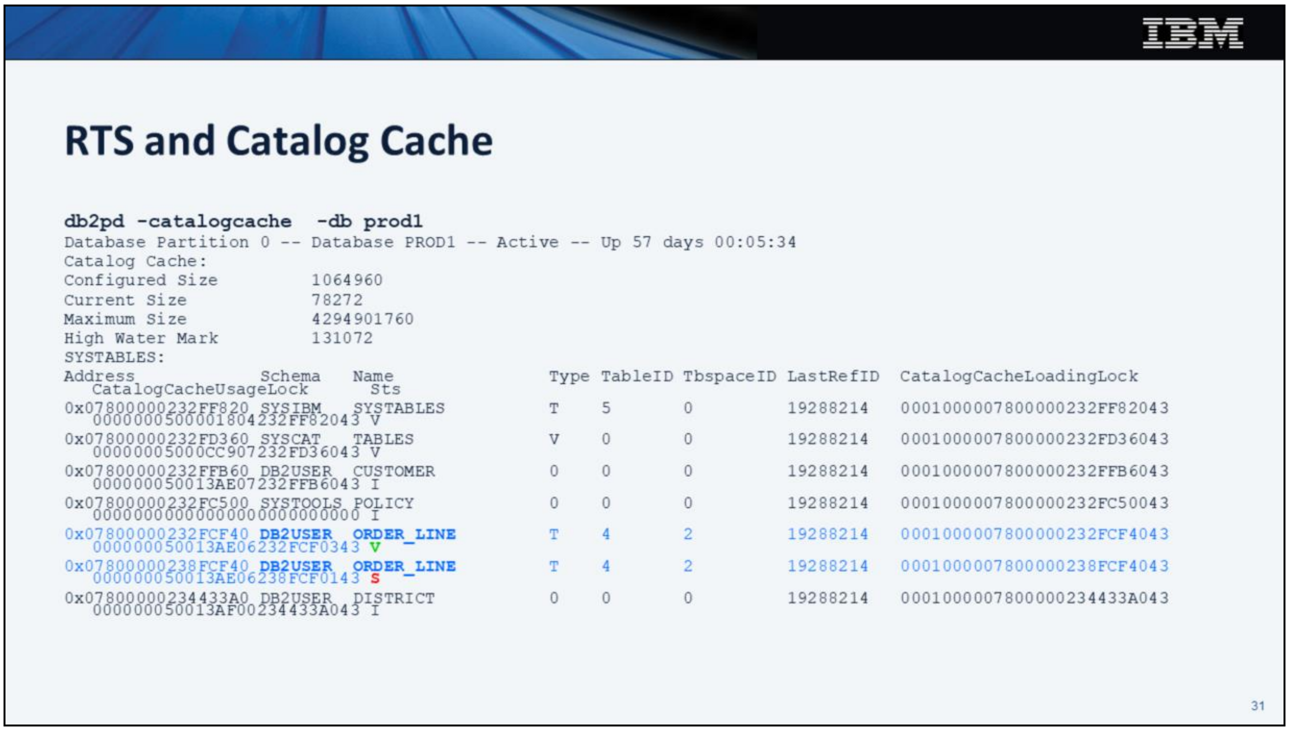
This page shows an example of the db2pd tool output displaying the contents of an entry in the statistics cache.



RTS and Catalog Cache

- **Synchronous statistics collection doesn't 'hard' invalidate existing entries in the catalog cache**
  - Requires waiting for the connection using the entry to release it
- **Existing entries are 'soft' invalidated**
  - Marked invalid, but existing connection can continue to use them
  - Flushed once the connection releases them
- **New catalog lookups load the latest entry from the statistics cache**
  - If the statistics haven't already been 'hardened' to disk
- Consequently, the catalog cache could contain multiple entries for the same table
  - N are marked soft invalid
  - 1 is valid (current)
- db2pd can be used to view the catalog cache
- Consider increasing the value of the **catalogcache_sz** database configuration parameter

30



RTS and Catalog Cache

```
db2pd -catalogcache  -db prod1
Database Partition 0 -- Database PROD1 -- Active -- Up 57 days 00:05:34
Catalog Cache:
Configured Size          1064960
Current Size             78272
Maximum Size             4294901760
High Water Mark          131072
SYSTABLES:
Address            Schema   Name                Type TableID TbspaceID LastRefID  CatalogCacheLoadingLock
   CatalogCacheUsageLock      Sts
0x07800000232FF820 SYSIBM   SYSTABLES           T    5       0         19288214   0001000007800000232FF82043
   0000000500001804232FF82043 V
0x07800000232FD360 SYSCAT   TABLES              V    0       0         19288214   0001000007800000232FD36043
   00000005000CC907232FD36043 V
0x07800000232FFB60 DB2USER  CUSTOMER            0    0       0         19288214   0001000007800000232FFB6043
   000000050013AE07232FFB6043 I
0x07800000232FC500 SYSTOOLS POLICY              0    0       0         19288214   0001000007800000232FC50043
   00000000000000000000000000 I
0x07800000232FCF40 DB2USER  ORDER_LINE          T    4       2         19288214   0001000007800000232FCF4043
   000000050013AE06232FCF0343 V
0x07800000238FCF40 DB2USER  ORDER_LINE          T    4       2         19288214   0001000007800000238FCF4043
   000000050013AE06238FCF0143 S
0x07800000234433A0 DB2USER  DISTRICT            0    0       0         19288214   000100000780000234433A043
   000000050013AF00234433A043 I
```

31

There are 2 entries for table DB2USER.ORDER_LINE in the catalog cache. The status (Sts) for the first one is V=valid. The status for the second is S=soft invalid. The second entry was soft invalidated because synchronous statistics were collected or fabricated while another DB connection was using the entry. Subsequent references to DB2USER.ORDER_LINE will use the new valid entry and the soft-invalid entry will be flushed from the cache after the current connection releases it.

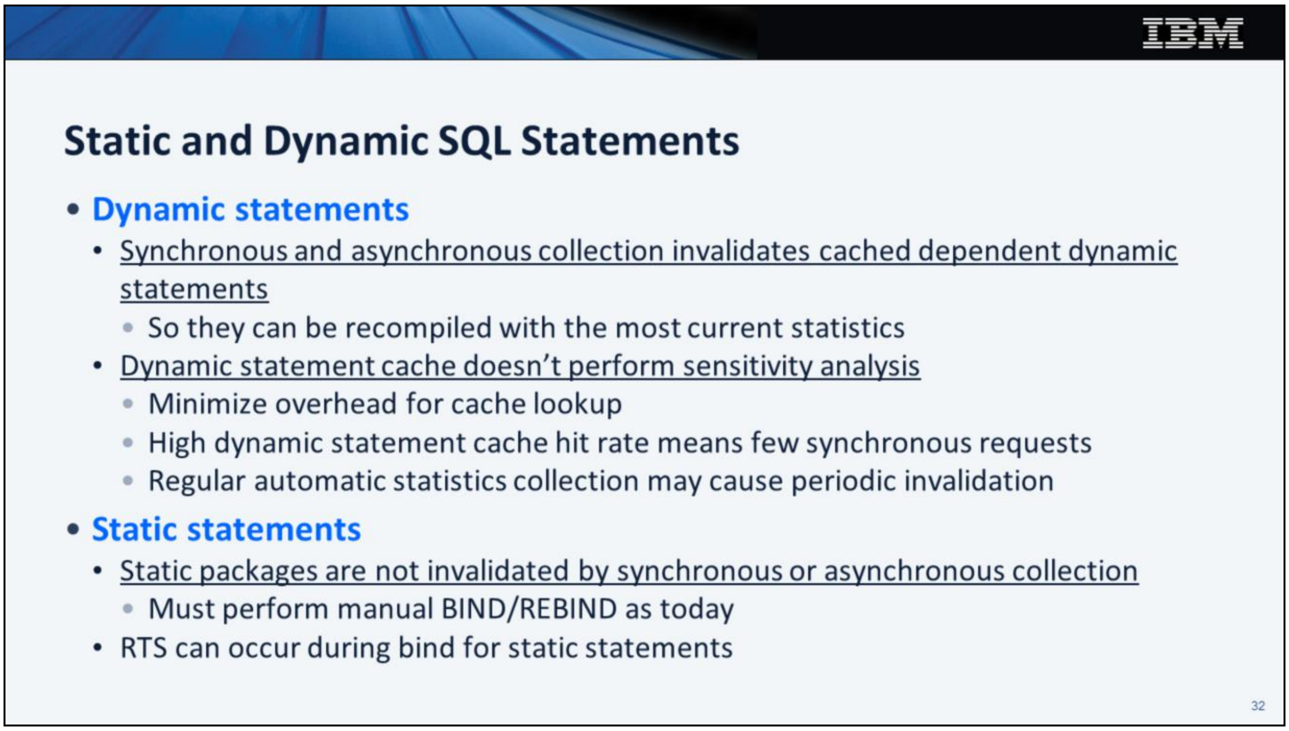
SYSTABLES:

Address	Address of catalog cache entry
Schema	The schema qualifier for the table.
Name	The name of the table.
Type	The type of the table.
TableID	The table identifier.
TbspaceID	The identifier of the table space where the table resides.
LastRefID	The last process identifier that referenced the table.
CatalogCache LoadingLock	The name of the catalog cache loading lock for the cache entry. A lock is acquired when the catalog cache entry is being loaded.
CatalogCache UsageLock	The name of the usage lock for the cache entry. A lock is acquired when the catalog cache entr is being referenced.
Sts	The status of the entry. The possible values are:

V (valid).

I (invalid).

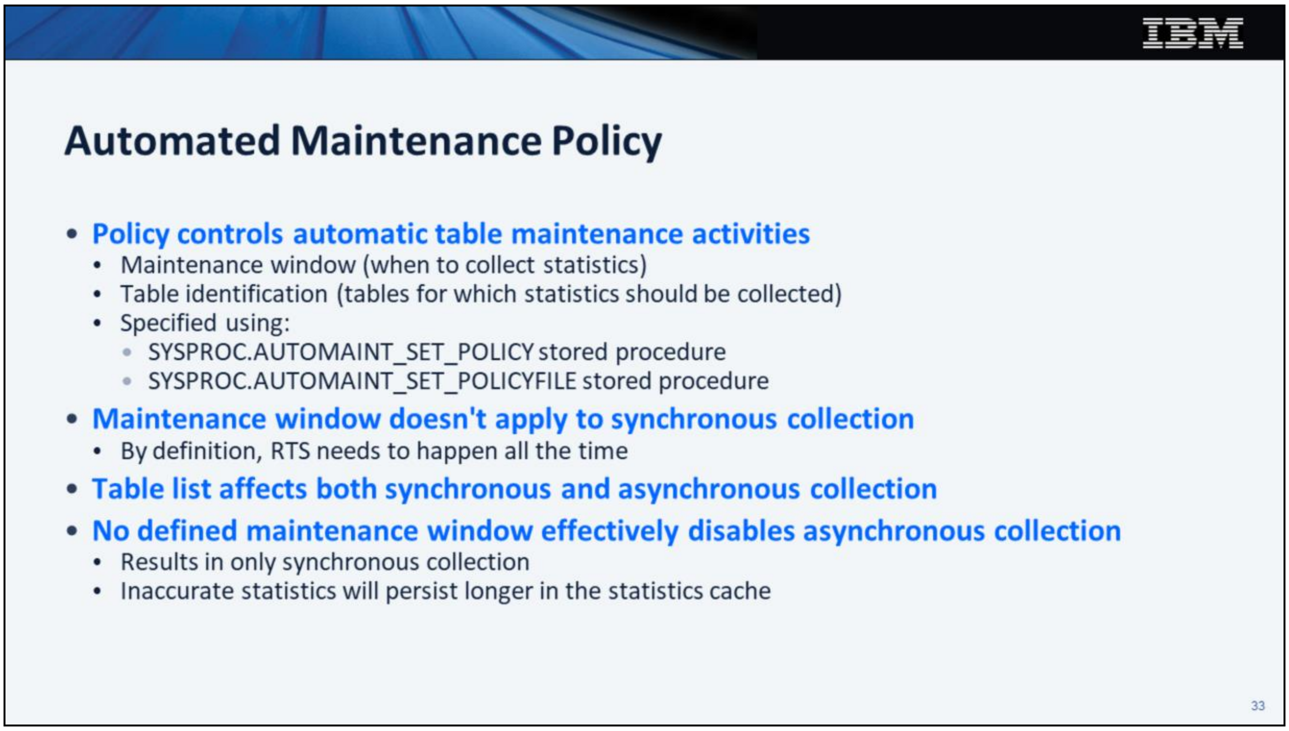
S (soft invalid. Catalog cache entries become *soft invalid* when statistics have been updated by real-time statistics collection. These catalog cache entries may still be used by a database agent, but they are not valid for use by a new catalog cache request. Once the soft invalid entry is no longer in use, it will be removed. New catalog cache requests will use the valid entry.)



Static and Dynamic SQL Statements

- **Dynamic statements**
  - Synchronous and asynchronous collection invalidates cached dependent dynamic statements
    - So they can be recompiled with the most current statistics
  - Dynamic statement cache doesn't perform sensitivity analysis
    - Minimize overhead for cache lookup
    - High dynamic statement cache hit rate means few synchronous requests
    - Regular automatic statistics collection may cause periodic invalidation
- **Static statements**
  - Static packages are not invalidated by synchronous or asynchronous collection
    - Must perform manual BIND/REBIND as today
  - RTS can occur during bind for static statements

32



Automated Maintenance Policy

- **Policy controls automatic table maintenance activities**
  - Maintenance window (when to collect statistics)
  - Table identification (tables for which statistics should be collected)
  - Specified using:
    - SYSPROC.AUTOMAINT_SET_POLICY stored procedure
    - SYSPROC.AUTOMAINT_SET_POLICYFILE stored procedure
- **Maintenance window doesn't apply to synchronous collection**
  - By definition, RTS needs to happen all the time
- **Table list affects both synchronous and asynchronous collection**
- **No defined maintenance window effectively disables asynchronous collection**
  - Results in only synchronous collection
  - Inaccurate statistics will persist longer in the statistics cache

33

Some examples:

The following causes the online maintenance to occur for 3 hours at the end of the first day of every month when ever falls on Monday:

```
<OnlineWindow Occurrence="During" startTime="21:00:00" duration="3">
  <DaysOfWeek>Mon</DayOfWeek>
  <DaysOfMonth>1</DayOfMonth>
  <MonthsOfYear>All</MonthOfYear>
</OnlineWindow>
```

You can specify which tables to exclude from the automatic statistics collection by using an expression similar to an SQL-style "where clause" in the FilterCondition. For example, the following specifies that all tables with names that match the pattern 'EMP%' should be excluded from the statistics collection:

```
<RunstatsTableScope>
  <FilterCondition>TABSCHEMA NOT LIKE 'EMP%' </FilterCondition>
</RunstatsTableScope>
```

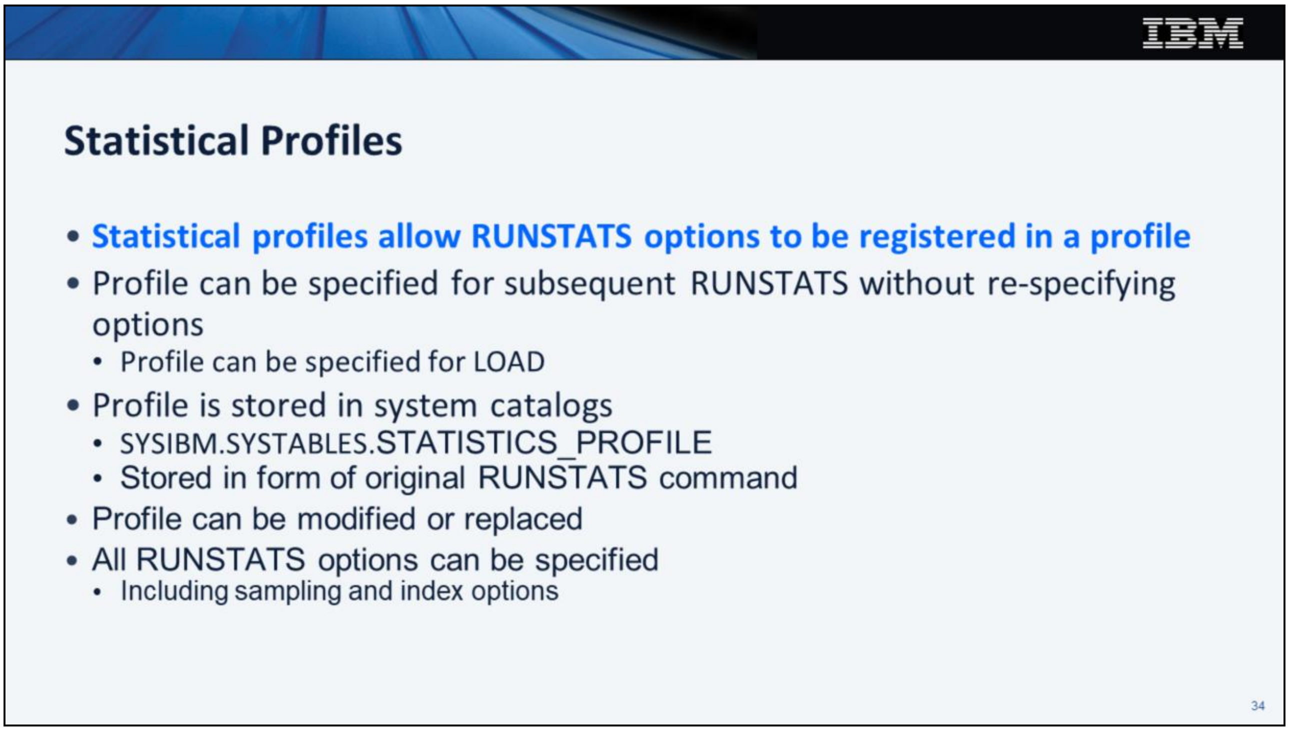
You can specify <FilterCondition/> to select all the tables.

For example, the following specifies that statistics should be collected for all tables, including system tables:

```
<RunstatsTableScope>
  <FilterCondition/>
</RunstatsTableScope>
```

For example, the following specifies that statistics should be collected for all tables except system tables:

```
<RunstatsTableScope>
  <FilterCondition>TABNAME NOT LIKE 'SYS%' </FilterCondition>
</RunstatsTableScope>
```


Statistical Profiles

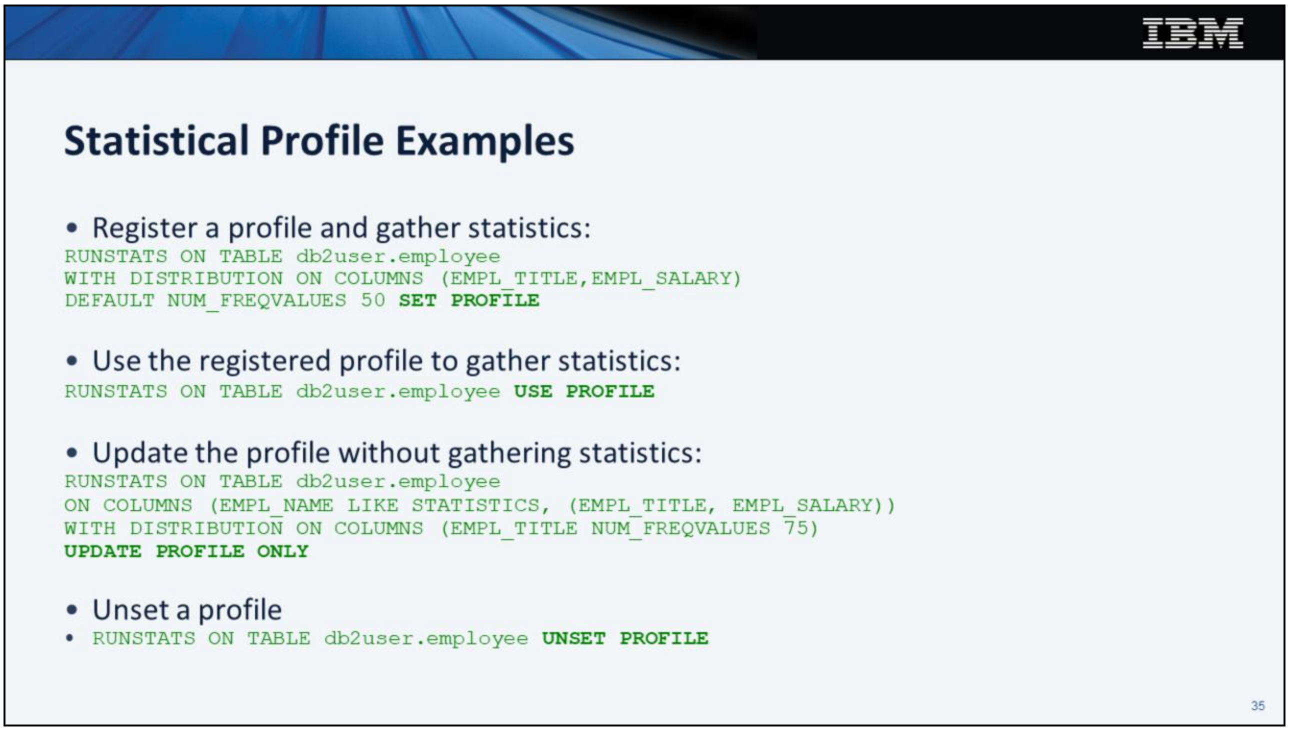
- **Statistical profiles allow RUNSTATS options to be registered in a profile**
- Profile can be specified for subsequent RUNSTATS without re-specifying options
  - Profile can be specified for LOAD
- Profile is stored in system catalogs
  - SYSIBM.SYSTABLES.STATISTICS_PROFILE
  - Stored in form of original RUNSTATS command
- Profile can be modified or replaced
- All RUNSTATS options can be specified
  - Including sampling and index options

34

The RUNSTATS utility provides a statistical profile facility to:
1) register a statistical profile, while optionally gathering statistics
2) modify an existing statistical profile stored in the catalogs, while optionally gathering statistics
3) repeatedly gather statistics on the table using an already registered statistics profile for that particular table.

This may be convenient for multiple scripts that need to perform RUNSTATS on the same set of tables so the RUNSTATS options don't need to be repeated in every script. Additionally, the statistical profile can be specified for LOAD so that consistent RUNSTATS options can be used and don't need to be repeated on the LOAD command.

When a statistical profile is registered, a RUNSTATS command string corresponding to that profile is at the same time built and stored in the STATISTICS_PROFILE column of the catalog table SYSIBM.SYSTABLES. An internal version of the profile is also maintained in the system catalogs in SYSTABLES.PACKED_DESC.



Statistical Profile Examples

- Register a profile and gather statistics:

```
RUNSTATS ON TABLE db2user.employee
WITH DISTRIBUTION ON COLUMNS (EMPL_TITLE,EMPL_SALARY)
DEFAULT NUM_FREQVALUES 50 SET PROFILE
```

- Use the registered profile to gather statistics:

```
RUNSTATS ON TABLE db2user.employee USE PROFILE
```

- Update the profile without gathering statistics:

```
RUNSTATS ON TABLE db2user.employee
ON COLUMNS (EMPL_NAME LIKE STATISTICS, (EMPL_TITLE, EMPL_SALARY))
WITH DISTRIBUTION ON COLUMNS (EMPL_TITLE NUM_FREQVALUES 75)
UPDATE PROFILE ONLY
```

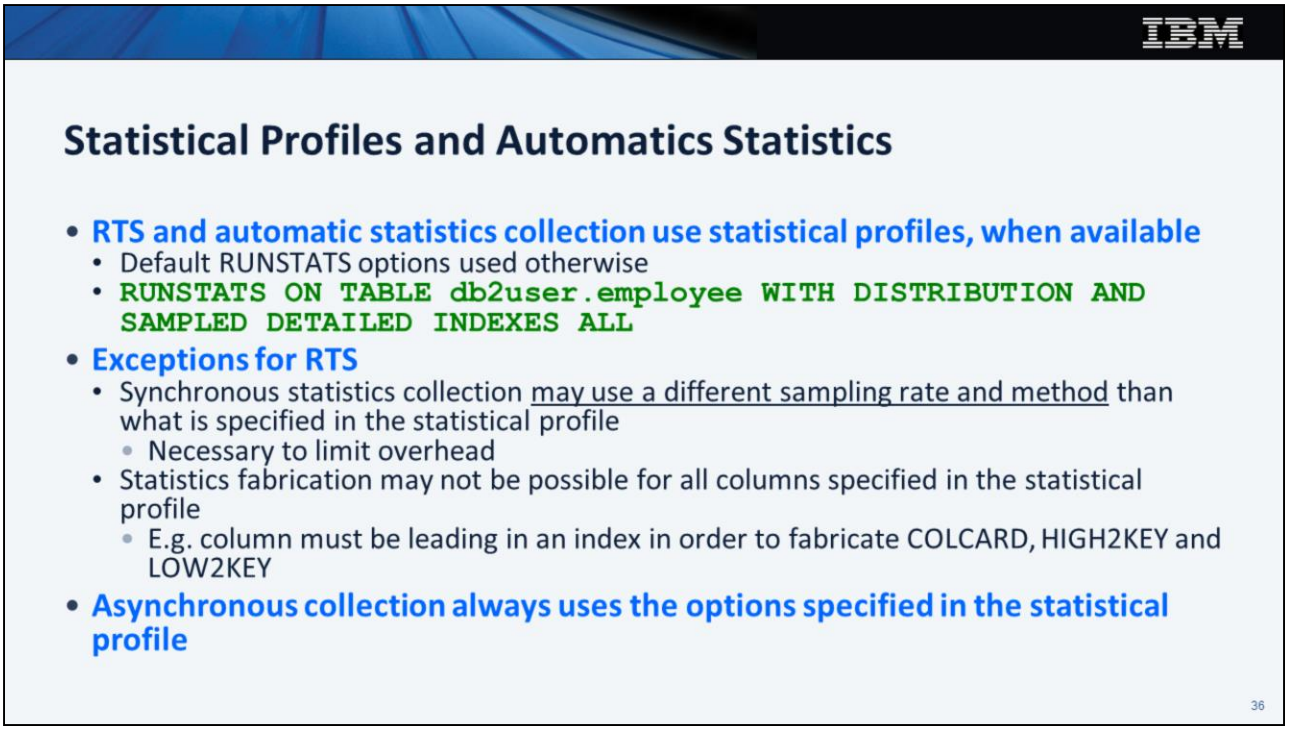
- Unset a profile
- `RUNSTATS ON TABLE db2user.employee UNSET PROFILE`

35

```
>>-RUNSTATS--ON TABLE--table name---+-- USE PROFILE ---------+-
><
                                    +-- UNSET PROFILE -------+
                                    '-- statistics-options -'
Statistics Options:
>--+---------- -----------+---+---------------------+-----><
   '-| (Other Options) |-'    '-| Profile Options |-`

Profile Options:

    .-- SET PROFILE NONE ------------------.
    |                                      |
|---+--------------------------------------+--|
    |                                      |
    +--| SET |----+-- PROFILE --+----------+
    |             |             |          |
    '--| UPDATE |-'             '-| ONLY |-`
```



Statistical Profiles and Automatics Statistics

- **RTS and automatic statistics collection use statistical profiles, when available**
  - Default RUNSTATS options used otherwise
  - `RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL`
- **Exceptions for RTS**
  - Synchronous statistics collection may use a different sampling rate and method than what is specified in the statistical profile
    - Necessary to limit overhead
  - Statistics fabrication may not be possible for all columns specified in the statistical profile
    - E.g. column must be leading in an index in order to fabricate COLCARD, HIGH2KEY and LOW2KEY
- **Asynchronous collection always uses the options specified in the statistical profile**

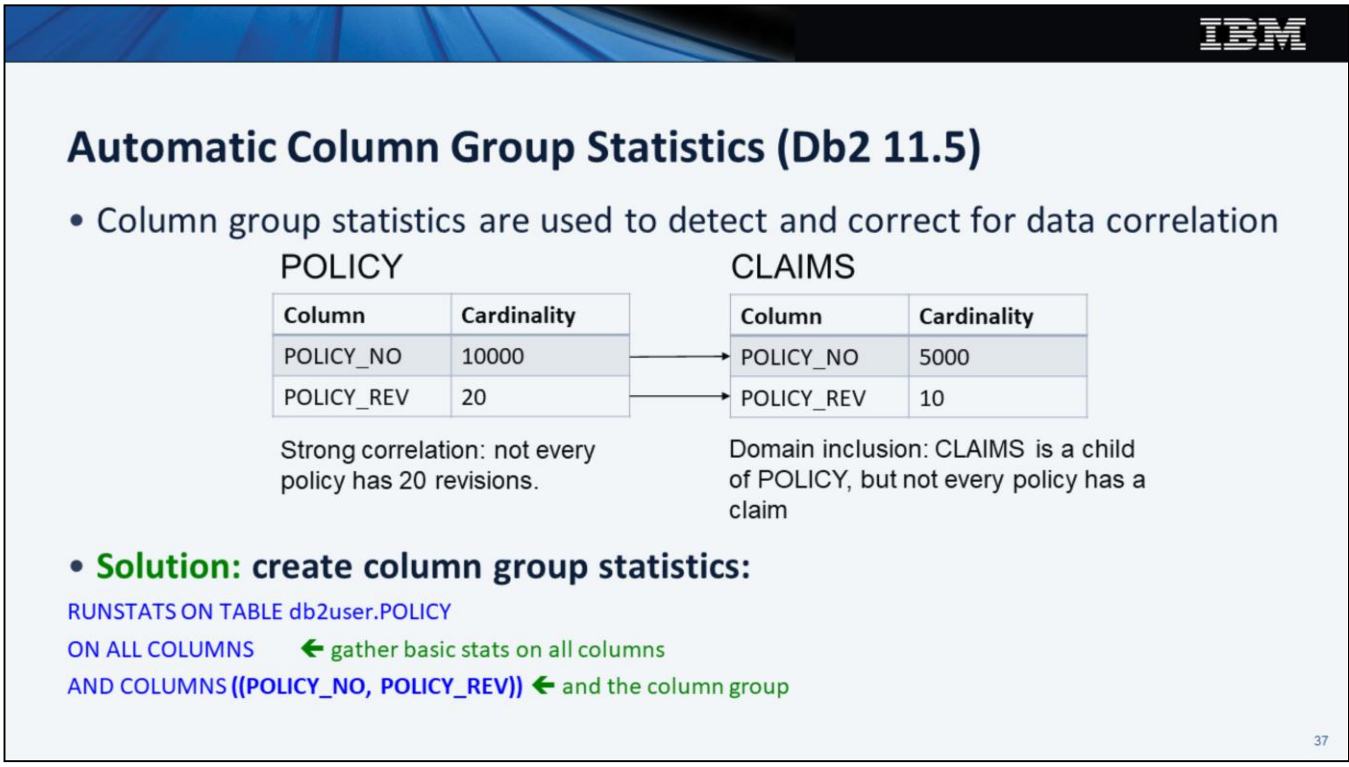
36

Synchronous and asynchronous statistics are collected according to a statistical profile that is in effect for a table, with the following exceptions:

To minimize the overhead of synchronous statistics collection, the database manager might collect statistics using sampling. In this case, the sampling rate and method might be different from those specified in the statistical profile.

Synchronous statistics collection might choose to fabricate statistics, but it might not be possible to fabricate all statistics specified in the statistical profile. For example, column statistics such as COLCARD, HIGH2KEY, and LOW2KEY cannot be fabricated unless the column is leading in some index.

If synchronous statistics collection cannot collect all statistics specified in the statistical profile, an asynchronous collection request is submitted.



IBM

Automatic Column Group Statistics (Db2 11.5)

- Column group statistics are used to detect and correct for data correlation

POLICY

Column	Cardinality
POLICY_NO	10000
POLICY_REV	20

Strong correlation: not every policy has 20 revisions.

CLAIMS

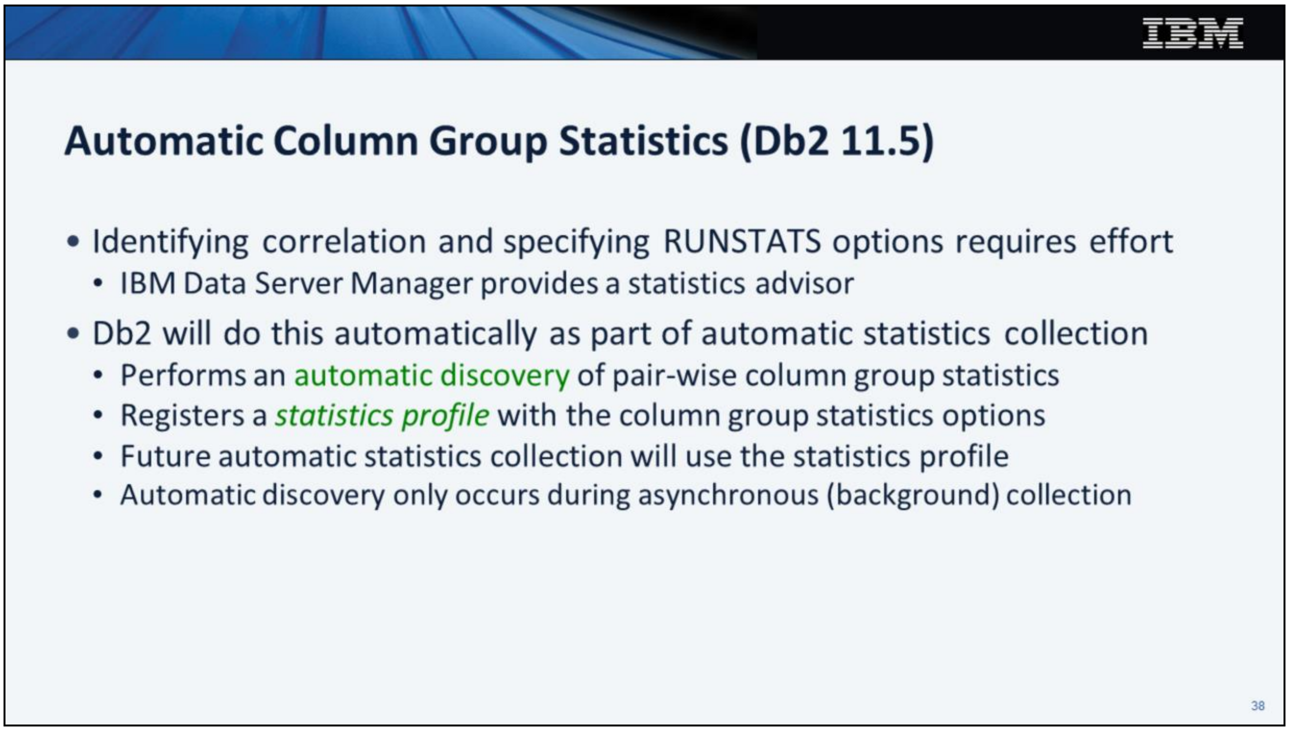
Column	Cardinality
POLICY_NO	5000
POLICY_REV	10

Domain inclusion: CLAIMS is a child of POLICY, but not every policy has a claim

- **Solution: create column group statistics:**

```
RUNSTATS ON TABLE db2user.POLICY
ON ALL COLUMNS        ← gather basic stats on all columns
AND COLUMNS ((POLICY_NO, POLICY_REV)) ← and the column group
```

37

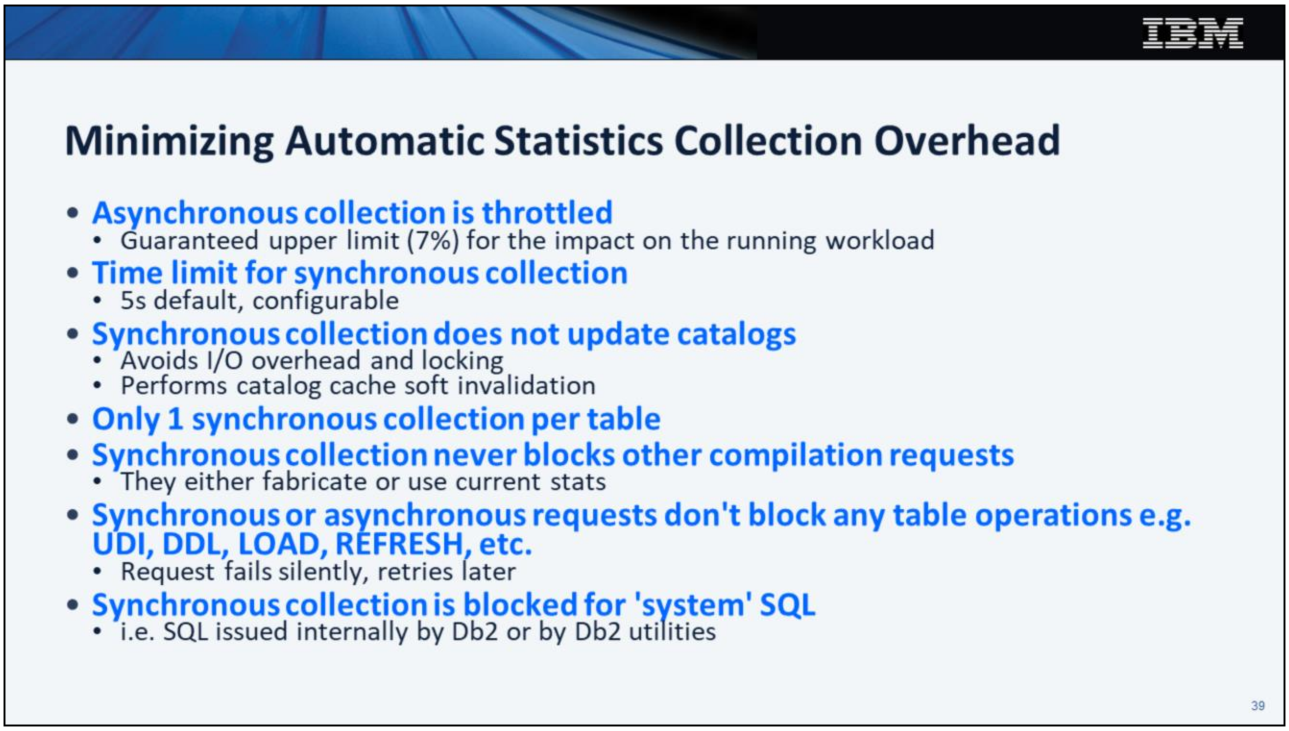


IBM

Automatic Column Group Statistics (Db2 11.5)

- Identifying correlation and specifying RUNSTATS options requires effort
  - IBM Data Server Manager provides a statistics advisor
- Db2 will do this automatically as part of automatic statistics collection
  - Performs an automatic discovery of pair-wise column group statistics
  - Registers a *statistics profile* with the column group statistics options
  - Future automatic statistics collection will use the statistics profile
  - Automatic discovery only occurs during asynchronous (background) collection

38

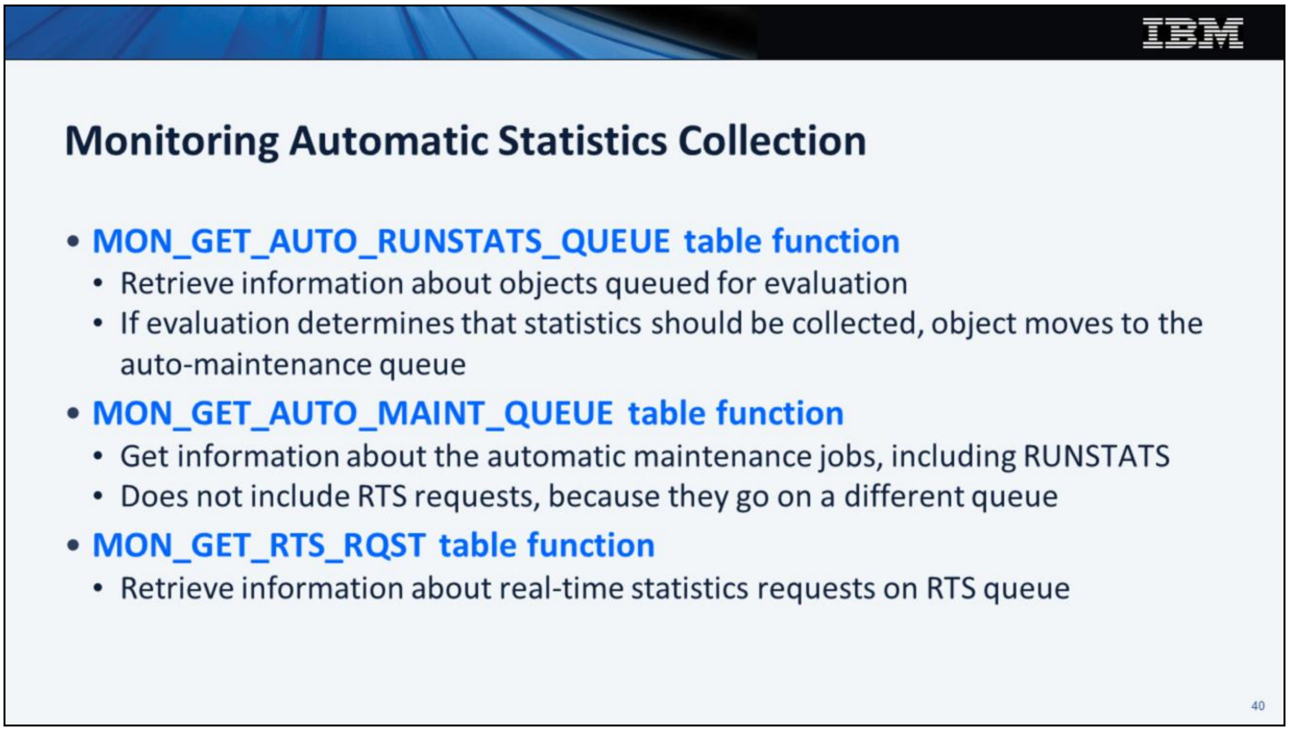


Minimizing Automatic Statistics Collection Overhead

- **Asynchronous collection is throttled**
  - Guaranteed upper limit (7%) for the impact on the running workload
- **Time limit for synchronous collection**
  - 5s default, configurable
- **Synchronous collection does not update catalogs**
  - Avoids I/O overhead and locking
  - Performs catalog cache soft invalidation
- **Only 1 synchronous collection per table**
- **Synchronous collection never blocks other compilation requests**
  - They either fabricate or use current stats
- **Synchronous or asynchronous requests don't block any table operations e.g. UDI, DDL, LOAD, REFRESH, etc.**
  - Request fails silently, retries later
- **Synchronous collection is blocked for 'system' SQL**
  - i.e. SQL issued internally by Db2 or by Db2 utilities

39

Careful consideration was given to minimize the overhead of Automatic Statistics Collection.



Monitoring Automatic Statistics Collection

- **MON_GET_AUTO_RUNSTATS_QUEUE table function**
  - Retrieve information about objects queued for evaluation
  - If evaluation determines that statistics should be collected, object moves to the auto-maintenance queue
- **MON_GET_AUTO_MAINT_QUEUE table function**
  - Get information about the automatic maintenance jobs, including RUNSTATS
  - Does not include RTS requests, because they go on a different queue
- **MON_GET_RTS_RQST table function**
  - Retrieve information about real-time statistics requests on RTS queue

40

MON_GET_AUTO_RUNSTATS_QUEUE :

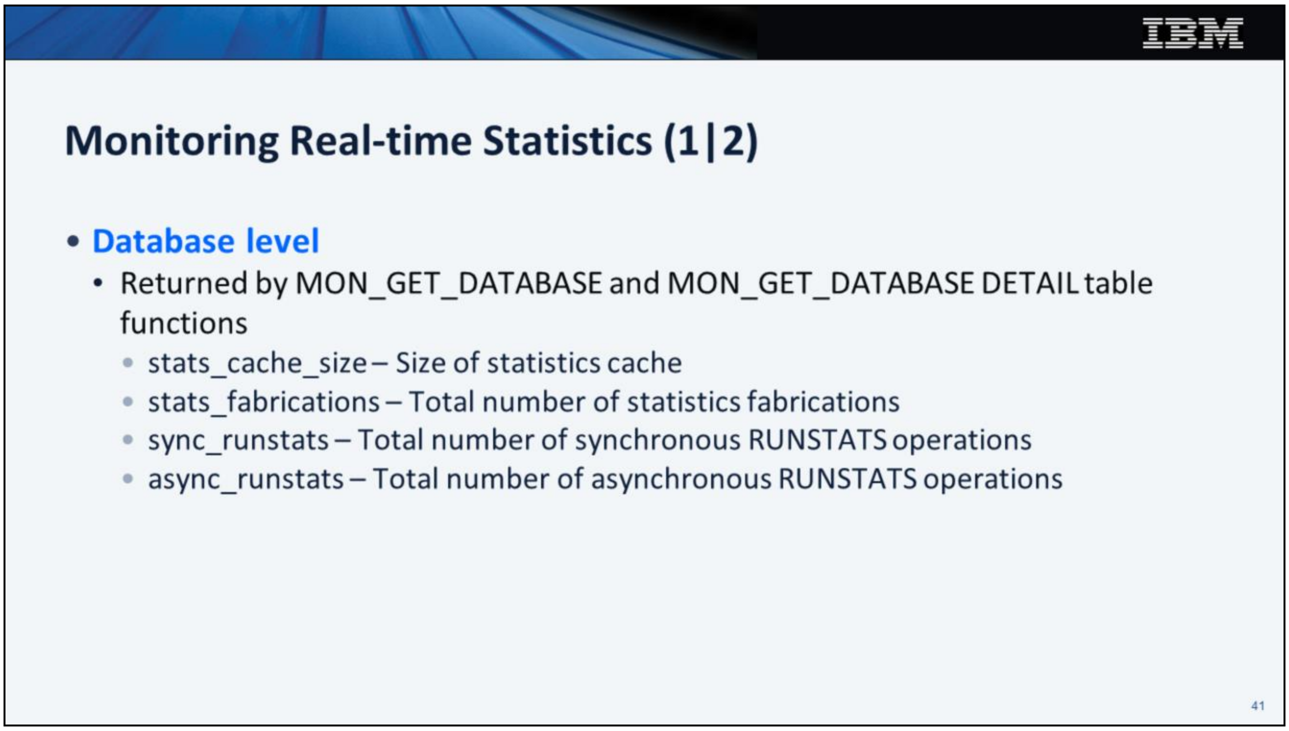
https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.sql.rtn.doc/doc/r0059254.html

MON_GET_AUTO_MAINT_QUEUE

https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.sql.rtn.doc/doc/r0059269.html

MON_GET_RTS_RQST

https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.sql.rtn.doc/doc/r0059255.html



Monitoring Real-time Statistics (1|2)

- **Database level**
  - Returned by MON_GET_DATABASE and MON_GET_DATABASE DETAIL table functions
    - stats_cache_size – Size of statistics cache
    - stats_fabrications – Total number of statistics fabrications
    - sync_runstats – Total number of synchronous RUNSTATS operations
    - async_runstats – Total number of asynchronous RUNSTATS operations

41

stats_cache_size – Size of statistics cache
The current size of the statistics cache, which is used in a catalog partition to cache statistics information generated by real-time statistics gathering.

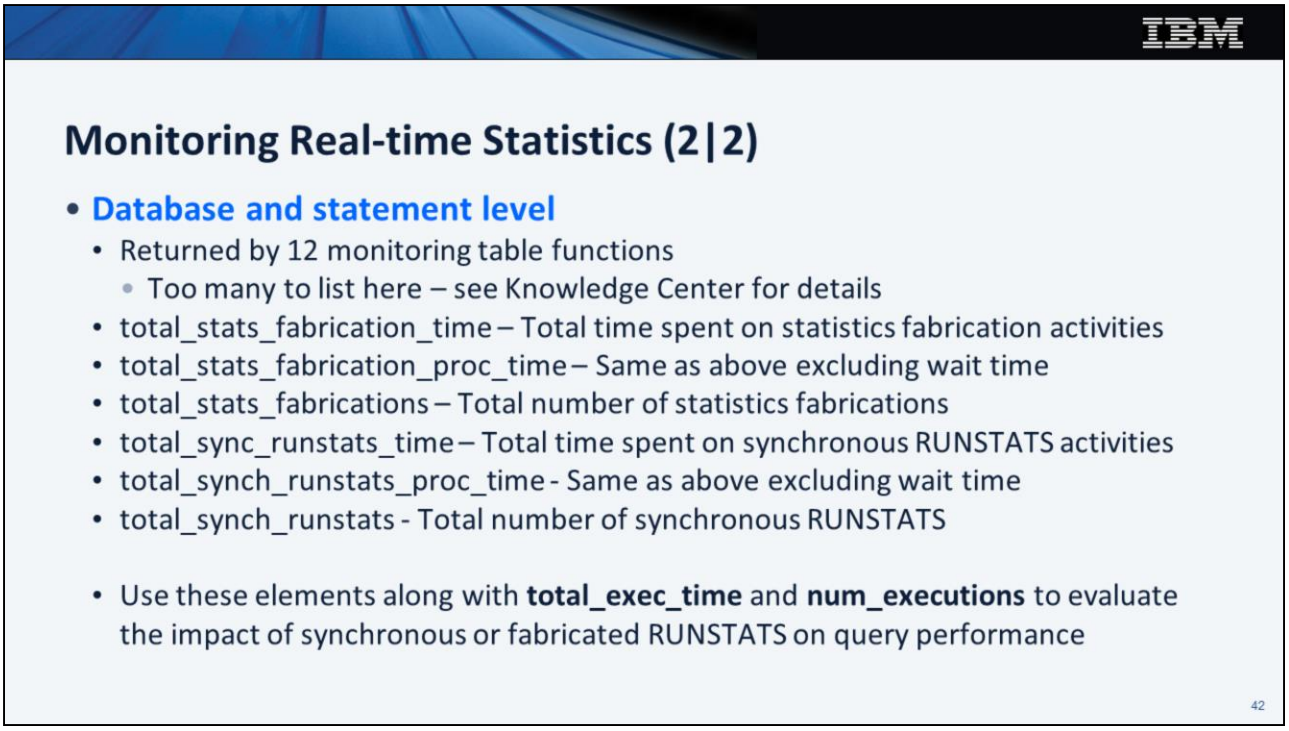
stats_fabrications – Total number of statistics fabrications
The total number of statistics fabrications performed by real-time statistics during query compilation for all the database applications. Rather than obtaining statistics by scanning data stored in a table or an index, statistics are fabricated based on metadata maintained by the index and data manager. Values reported by all the database partitions are aggregated together.

sync_runstats – Total number of synchronous RUNSTATS activities

The total number of synchronous RUNSTATS activities triggered by real-time statistics gathering for all the applications in the database. This value includes both successful and unsuccessful synchronous RUNSTATS commands. Values reported by all the database partitions are aggregated together.

async_runstats – Total number of asynchronous RUNSTATS requests

The total number of successful asynchronous RUNSTATS activities performed by real-time statistics gathering for all the applications in the database. Values reported by all the database partitions are aggregated together.

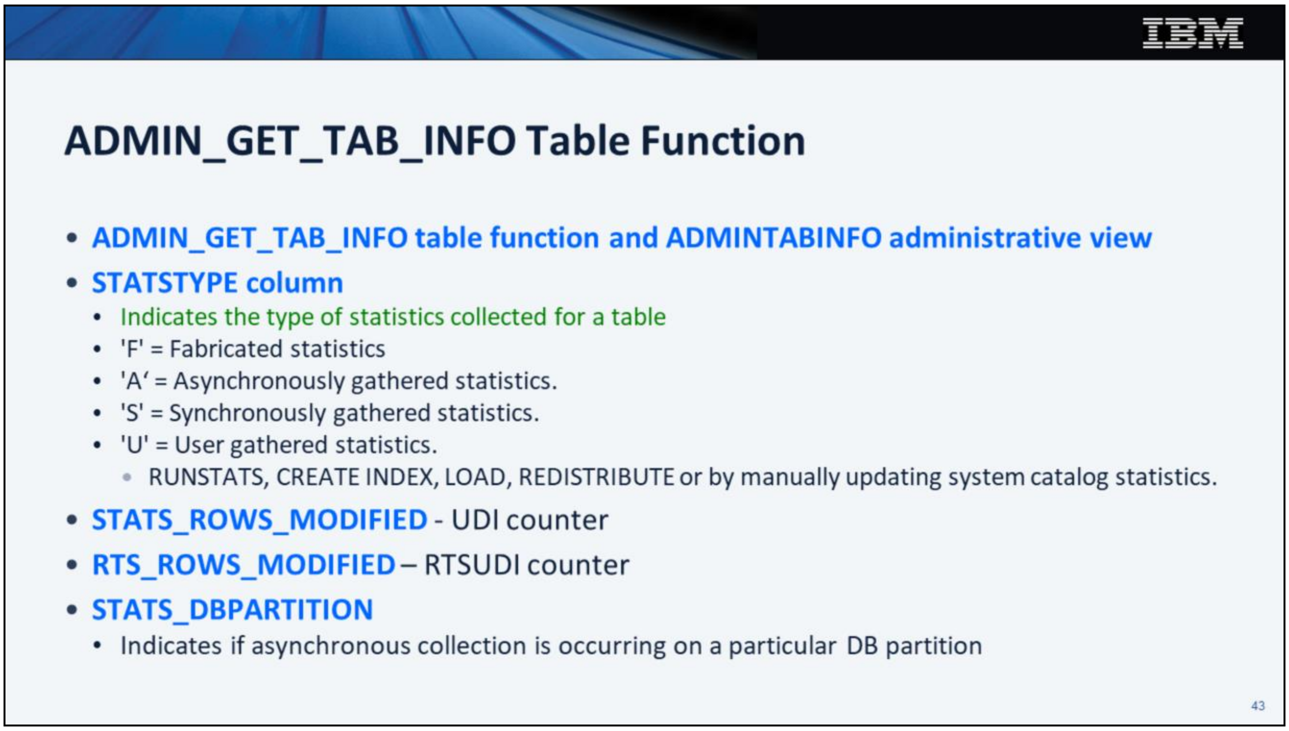


Monitoring Real-time Statistics (2|2)

- **Database and statement level**
  - Returned by 12 monitoring table functions
    - Too many to list here – see Knowledge Center for details
  - total_stats_fabrication_time – Total time spent on statistics fabrication activities
  - total_stats_fabrication_proc_time – Same as above excluding wait time
  - total_stats_fabrications – Total number of statistics fabrications
  - total_sync_runstats_time – Total time spent on synchronous RUNSTATS activities
  - total_synch_runstats_proc_time - Same as above excluding wait time
  - total_synch_runstats - Total number of synchronous RUNSTATS

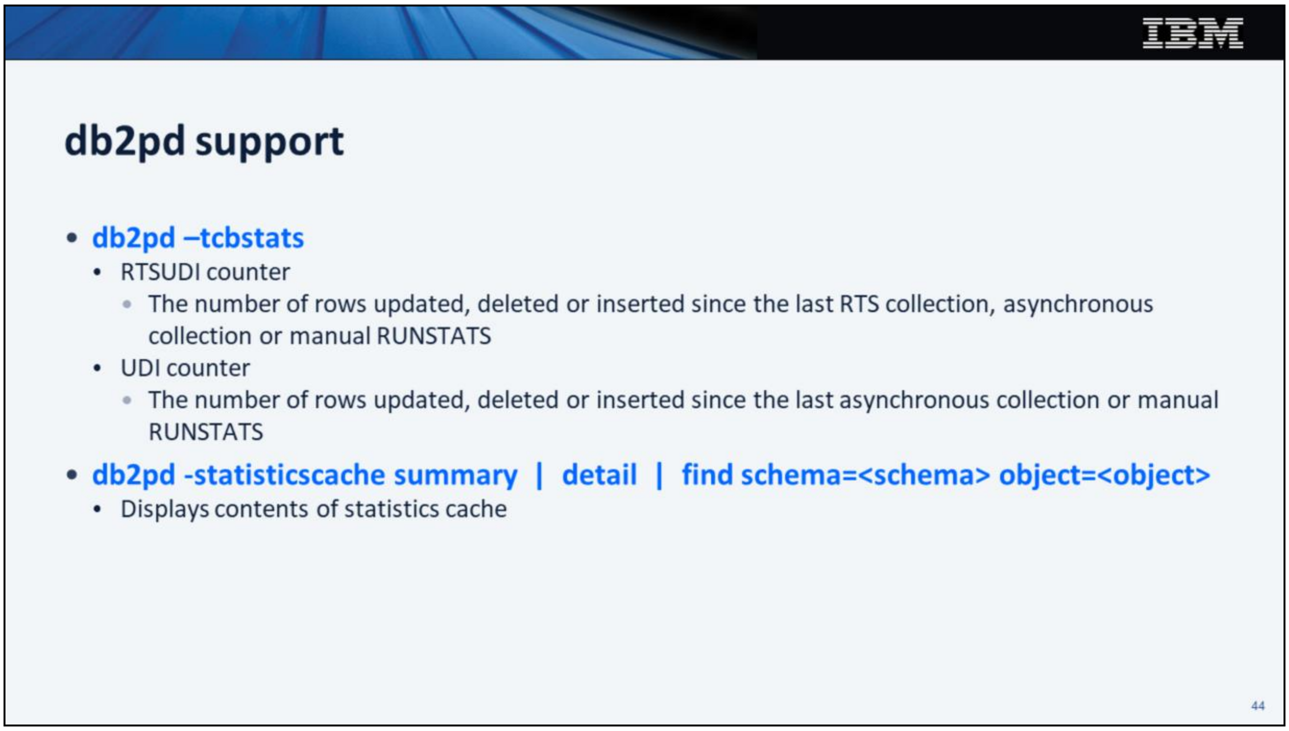
  - Use these elements along with **total_exec_time** and **num_executions** to evaluate the impact of synchronous or fabricated RUNSTATS on query performance

42



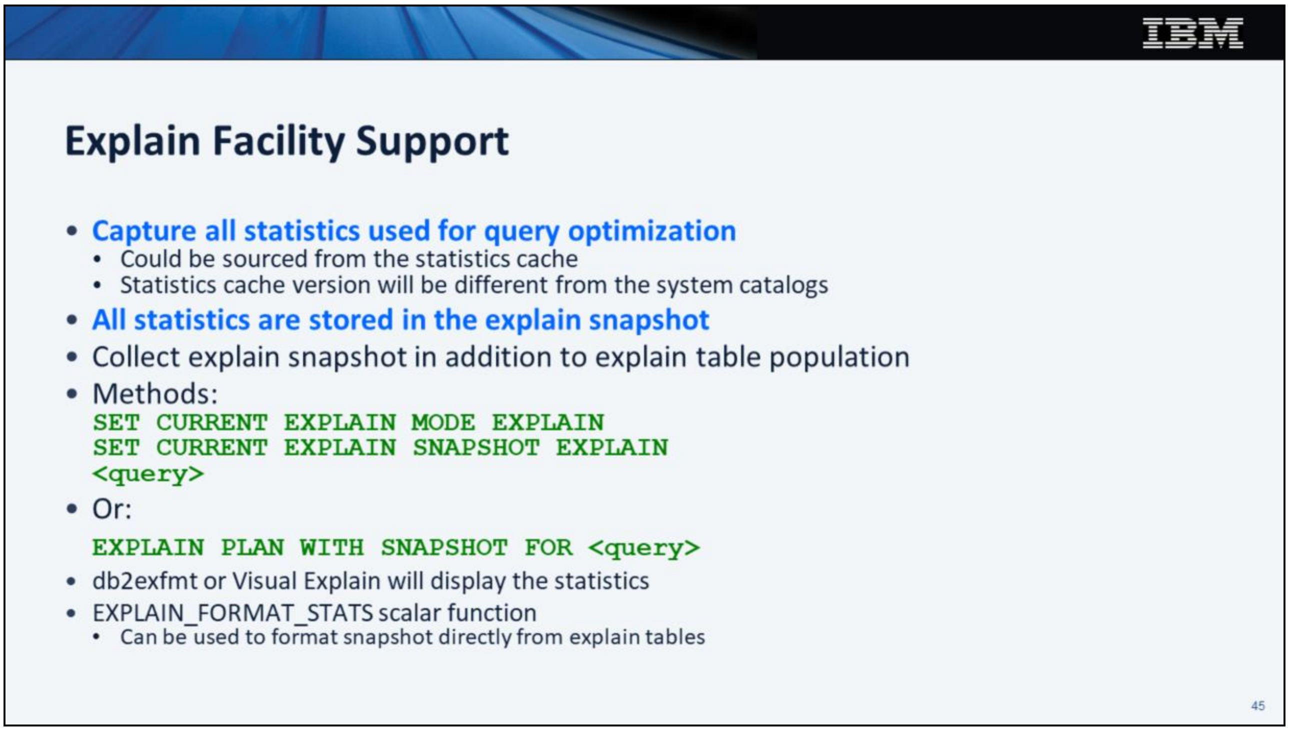
ADMIN_GET_TAB_INFO Table Function

- **ADMIN_GET_TAB_INFO table function and ADMINTABINFO administrative view**
- **STATSTYPE column**
  - Indicates the type of statistics collected for a table
  - 'F' = Fabricated statistics
  - 'A‘ = Asynchronously gathered statistics.
  - 'S' = Synchronously gathered statistics.
  - 'U' = User gathered statistics.
    - RUNSTATS, CREATE INDEX, LOAD, REDISTRIBUTE or by manually updating system catalog statistics.
- **STATS_ROWS_MODIFIED** - UDI counter
- **RTS_ROWS_MODIFIED** – RTSUDI counter
- **STATS_DBPARTITION**
  - Indicates if asynchronous collection is occurring on a particular DB partition



db2pd support

- **db2pd –tcbstats**
  - RTSUDI counter
    - The number of rows updated, deleted or inserted since the last RTS collection, asynchronous collection or manual RUNSTATS
  - UDI counter
    - The number of rows updated, deleted or inserted since the last asynchronous collection or manual RUNSTATS
- **db2pd -statisticscache summary | detail | find schema=<schema> object=<object>**
  - Displays contents of statistics cache


Explain Facility Support

- **Capture all statistics used for query optimization**
  - Could be sourced from the statistics cache
  - Statistics cache version will be different from the system catalogs
- **All statistics are stored in the explain snapshot**
- Collect explain snapshot in addition to explain table population
- Methods:

```
SET CURRENT EXPLAIN MODE EXPLAIN
SET CURRENT EXPLAIN SNAPSHOT EXPLAIN
<query>
```

- Or:

```
EXPLAIN PLAN WITH SNAPSHOT FOR <query>
```

- db2exfmt or Visual Explain will display the statistics
- EXPLAIN_FORMAT_STATS scalar function
  - Can be used to format snapshot directly from explain tables

45

EXPLAIN_FORMAT_STATS Scalar function

This new scalar function is used to display formatted statistics information which is parsed and extracted from explain snapshot captured for a given query. The data type of the result is CLOB(50M).

Syntax

_ >>-EXPLAIN_FORMAT_STATS--(--*snapshot*--)------------------------><

The schema is SYSPROC.

Scaler function parameters

snapshot

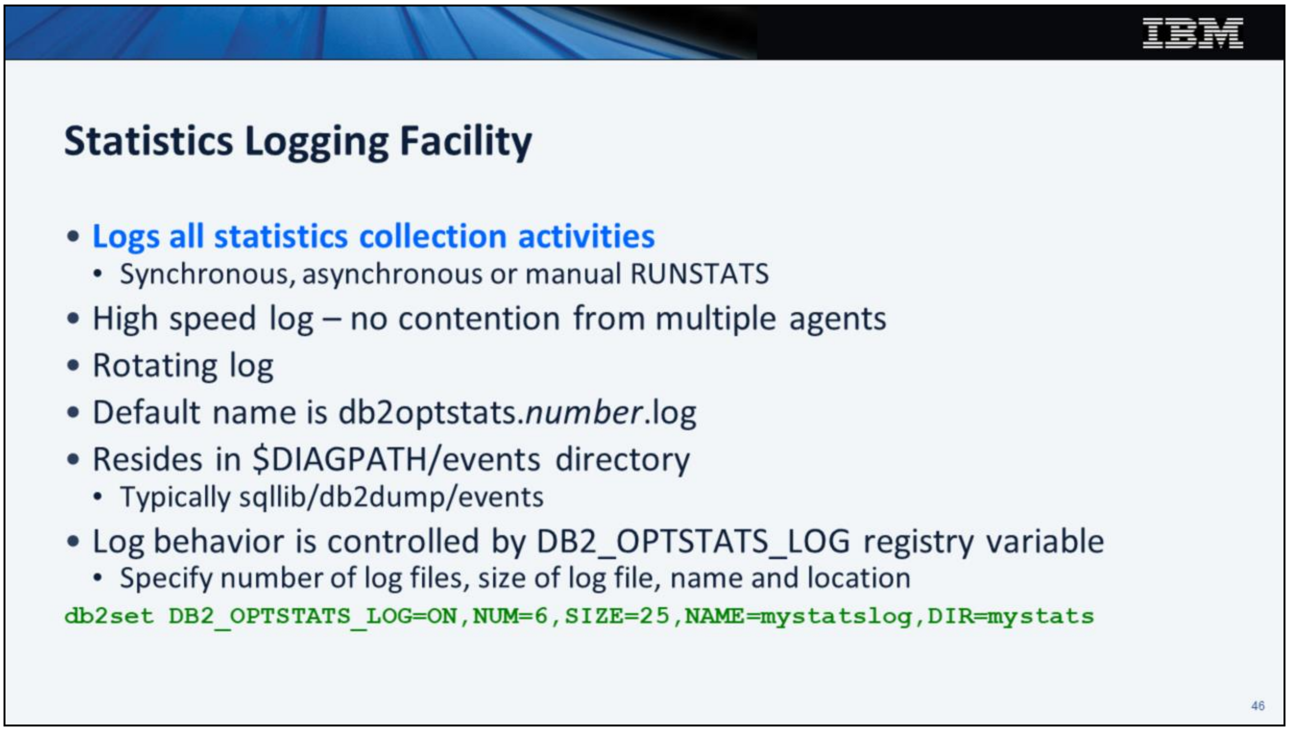
An input argument of type BLOB(10M) that is the explain snapshot captured for a given query. It is stored as snapshot column of explain table *EXPLAIN_STATEMENT*

Authorization

EXECUTE privilege on the EXPLAIN_FORMAT_STATS function.

Example

```
SELECT EXPLAIN_FORMAT_STATS(SNAPSHOT)
FROM EXPLAIN_STATEMENT
WHERE EXPLAIN_REQUESTER = 'DB2USER1' AND
EXPLAIN_TIME = timestamp('2006-05-12-14.38.11.109432') AND
SOURCE_NAME = 'SQLC2F0A' AND
SOURCE_SCHEMA = 'NULLID' AND
SOURCE_VERSION = '' AND
EXPLAIN_LEVEL = 'O' AND STMTNO = 1 AND SECTNO = 201
```


Statistics Logging Facility

- **Logs all statistics collection activities**
  - Synchronous, asynchronous or manual RUNSTATS
- High speed log – no contention from multiple agents
- Rotating log
- Default name is db2optstats.*number*.log
- Resides in $DIAGPATH/events directory
  - Typically sqllib/db2dump/events
- Log behavior is controlled by DB2_OPTSTATS_LOG registry variable
  - Specify number of log files, size of log file, name and location

```
db2set DB2_OPTSTATS_LOG=ON,NUM=6,SIZE=25,NAME=mystatslog,DIR=mystats
```

46

DB2_OPTSTATS_LOG

Operating system: All

Default=Not set (see details below), Values=OFF, ON {NUM | SIZE | NAME | DIR}

DB2_OPTSTATS_LOG specifies the attributes of the statistics event logging files which are used to monitor and analyze statistics collection related activities. When **DB2_OPTSTATS_LOG** is not set or set to ON, statistics event logging is enabled, allowing you to monitor system performance and keep a history for better problem determination. Log records are written to the first log file until it is full. Subsequent records are written to the next available log file. If the maximum number of files is reached, the oldest log file will be overwritten with the new records. If system resource consumption is of great concern to you, disable this registry variable by setting it to OFF.

When statistics event logging is explicitly enabled (set to ON), there are a number of options you can modify:

NUM: the maximum number of rotating log files. Default=5, Values 1 - 15

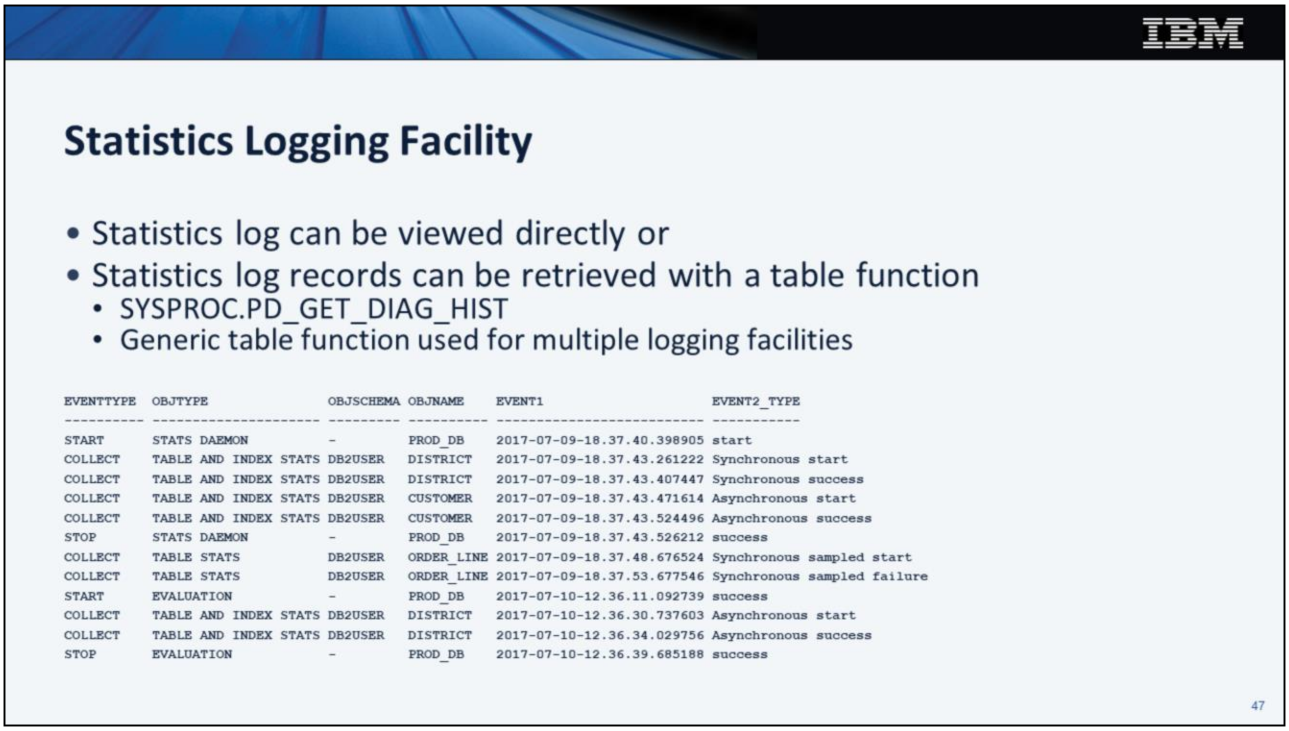
SIZE: the maximum size of rotating log files. (The size of each rotating file is SIZE/NUM.) Default=100 Mb, Values 1 Mb – 4096 Mb

NAME: the base name for rotating log files. Default=db2optstats.<number>.log, for instance db2optstats.0.log, db2optstats.1.log, etc.

DIR: the base directory for rotating log files. Default=$DIAGPATH/events

You can specify a value for as many of these options as you want, just ensure that ON is the first value when you want to enable statistics logging. For instance, to enable statistics logging with maximum of 6 log files, a maximum file size of 25 Mb, a base file name of mystatslog, and the directory mystats, issue the following command:

```
db2set DB2_OPTSTATS_LOG=ON,NUM=6,SIZE=25,NAME=mystatslog,DIR=mystats
```



Statistics Logging Facility

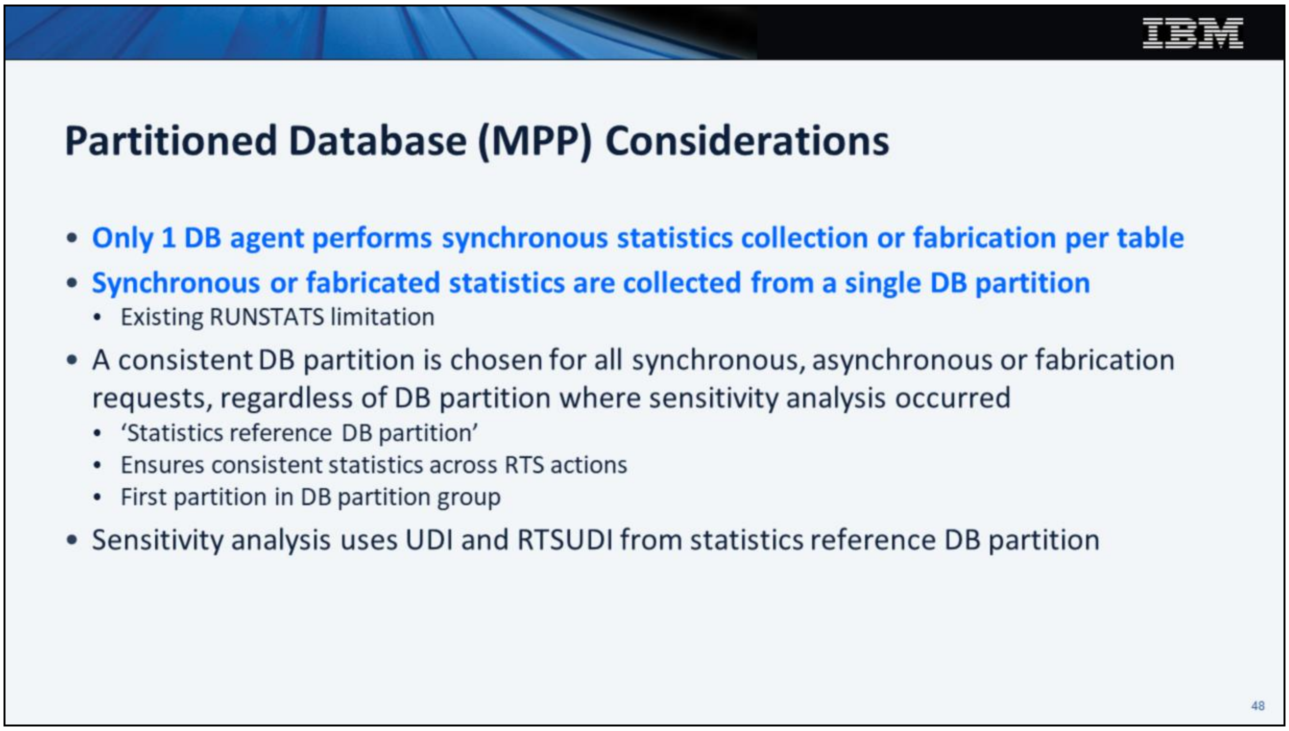
- Statistics log can be viewed directly or
- Statistics log records can be retrieved with a table function
  - SYSPROC.PD_GET_DIAG_HIST
  - Generic table function used for multiple logging facilities

```
EVENTTYPE  OBJTYPE              OBJSCHEMA OBJNAME    EVENT1                     EVENT2_TYPE
---------- -------------------- --------- ---------- -------------------------- -----------
START      STATS DAEMON         -         PROD_DB    2017-07-09-18.37.40.398905 start
COLLECT    TABLE AND INDEX STATS DB2USER   DISTRICT   2017-07-09-18.37.43.261222 Synchronous start
COLLECT    TABLE AND INDEX STATS DB2USER   DISTRICT   2017-07-09-18.37.43.407447 Synchronous success
COLLECT    TABLE AND INDEX STATS DB2USER   CUSTOMER   2017-07-09-18.37.43.471614 Asynchronous start
COLLECT    TABLE AND INDEX STATS DB2USER   CUSTOMER   2017-07-09-18.37.43.524496 Asynchronous success
STOP       STATS DAEMON         -         PROD_DB    2017-07-09-18.37.43.526212 success
COLLECT    TABLE STATS          DB2USER   ORDER_LINE 2017-07-09-18.37.48.676524 Synchronous sampled start
COLLECT    TABLE STATS          DB2USER   ORDER_LINE 2017-07-09-18.37.53.677546 Synchronous sampled failure
START      EVALUATION           -         PROD_DB    2017-07-10-12.36.11.092739 success
COLLECT    TABLE AND INDEX STATS DB2USER   DISTRICT   2017-07-10-12.36.30.737603 Asynchronous start
COLLECT    TABLE AND INDEX STATS DB2USER   DISTRICT   2017-07-10-12.36.34.029756 Asynchronous success
STOP       EVALUATION           -         PROD_DB    2017-07-10-12.36.39.685188 success
```

47

In this example, the query returns statistics log records for events up to one year prior to the current timestamp, by invoking PD_GET_DIAG_HIST.

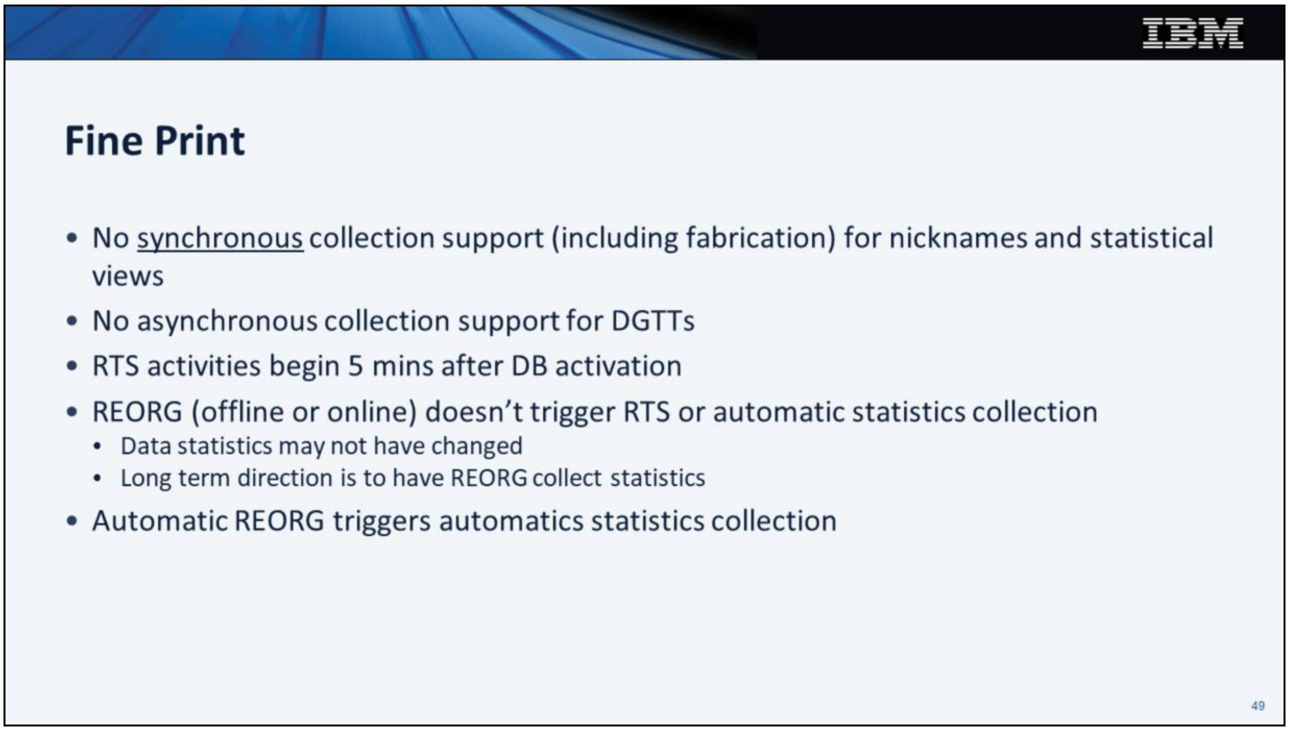
```
SELECT pid, tid, substr(eventtype, 1,10), substr(objtype,1,30) as
objtype, substr(objname_qualifier,1,20) as objschema,
substr(objname,1,10) as objname,
substr(first_eventqualifier,1,26) as event1,
substr(second_eventqualifiertype,1,2) as event2_type,
substr(second_eventqualifier,1,20) event2,
substr(third_eventqualifiertype,1,6) event3_type,
substr(third_eventqualifier,1,15) event3, substr(eventstate,1,20)
as eventstate FROM TABLE( SYSPROC.PD_GET_DIAG_HIST ( 'optstats',
'EX', 'NONE', CURRENT_TIMESTAMP - 1 year, CAST( NULL AS TIMESTAMP
))) as sl order by
timestamp(varchar(substr(first_eventqualifier,1,26),26)) ;
```



Partitioned Database (MPP) Considerations

- **Only 1 DB agent performs synchronous statistics collection or fabrication per table**
- **Synchronous or fabricated statistics are collected from a single DB partition**
  - Existing RUNSTATS limitation
- A consistent DB partition is chosen for all synchronous, asynchronous or fabrication requests, regardless of DB partition where sensitivity analysis occurred
  - 'Statistics reference DB partition'
  - Ensures consistent statistics across RTS actions
  - First partition in DB partition group
- Sensitivity analysis uses UDI and RTSUDI from statistics reference DB partition

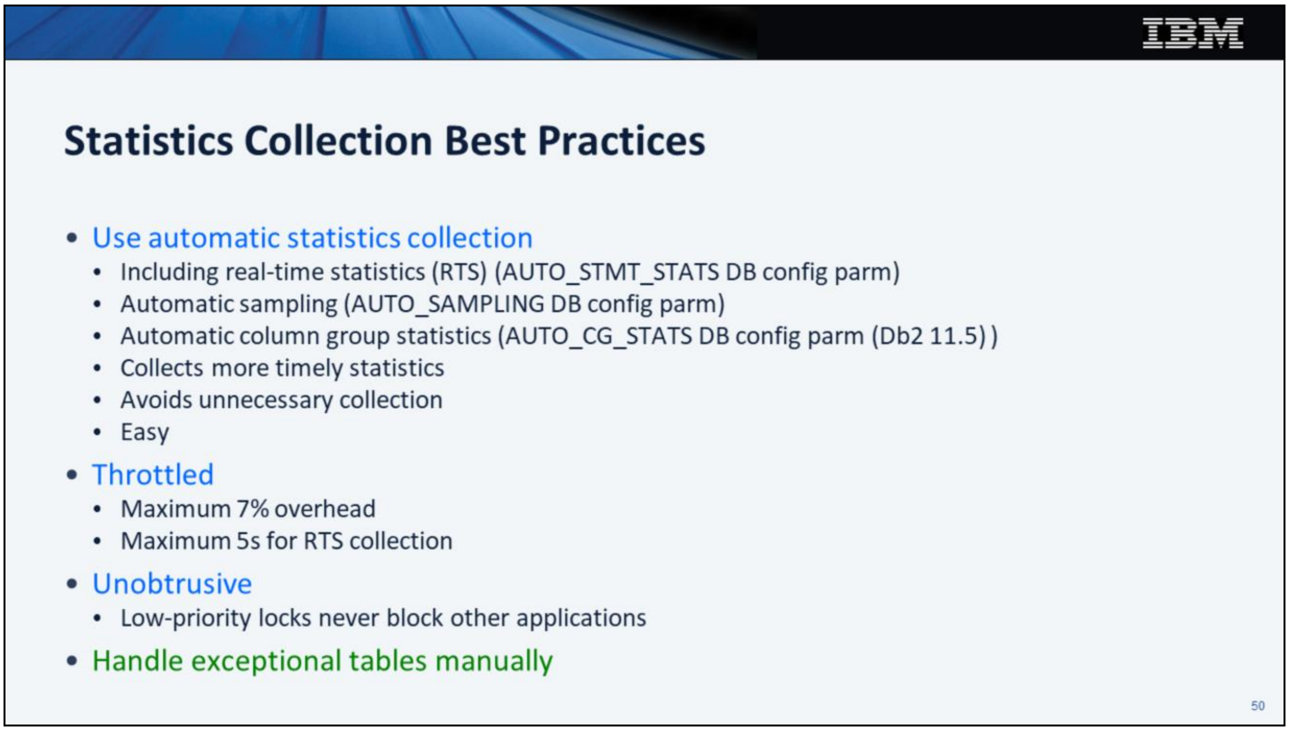
48



Fine Print

- No synchronous collection support (including fabrication) for nicknames and statistical views
- No asynchronous collection support for DGTTs
- RTS activities begin 5 mins after DB activation
- REORG (offline or online) doesn't trigger RTS or automatic statistics collection
  - Data statistics may not have changed
  - Long term direction is to have REORG collect statistics
- Automatic REORG triggers automatics statistics collection

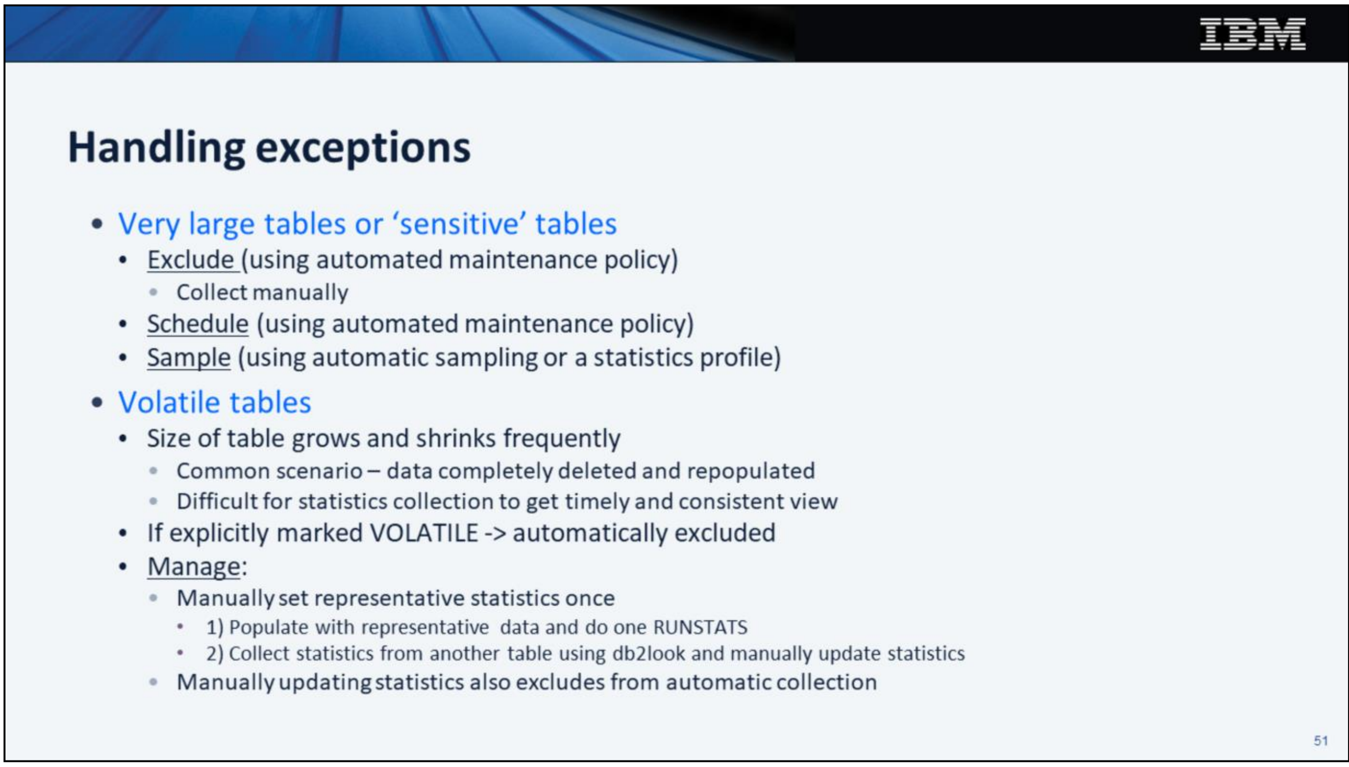
49



Statistics Collection Best Practices

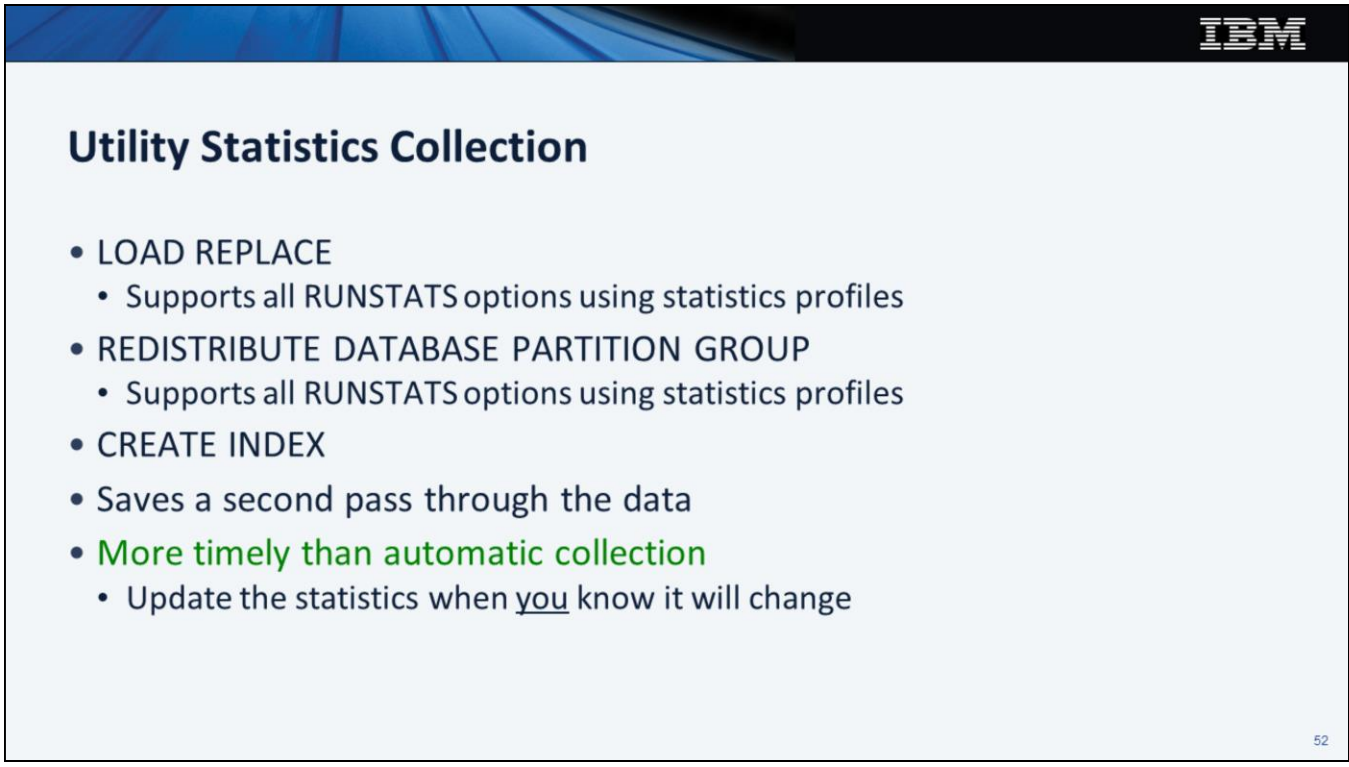
- Use automatic statistics collection
  - Including real-time statistics (RTS) (AUTO_STMT_STATS DB config parm)
  - Automatic sampling (AUTO_SAMPLING DB config parm)
  - Automatic column group statistics (AUTO_CG_STATS DB config parm (Db2 11.5))
  - Collects more timely statistics
  - Avoids unnecessary collection
  - Easy
- Throttled
  - Maximum 7% overhead
  - Maximum 5s for RTS collection
- Unobtrusive
  - Low-priority locks never block other applications
- Handle exceptional tables manually

50



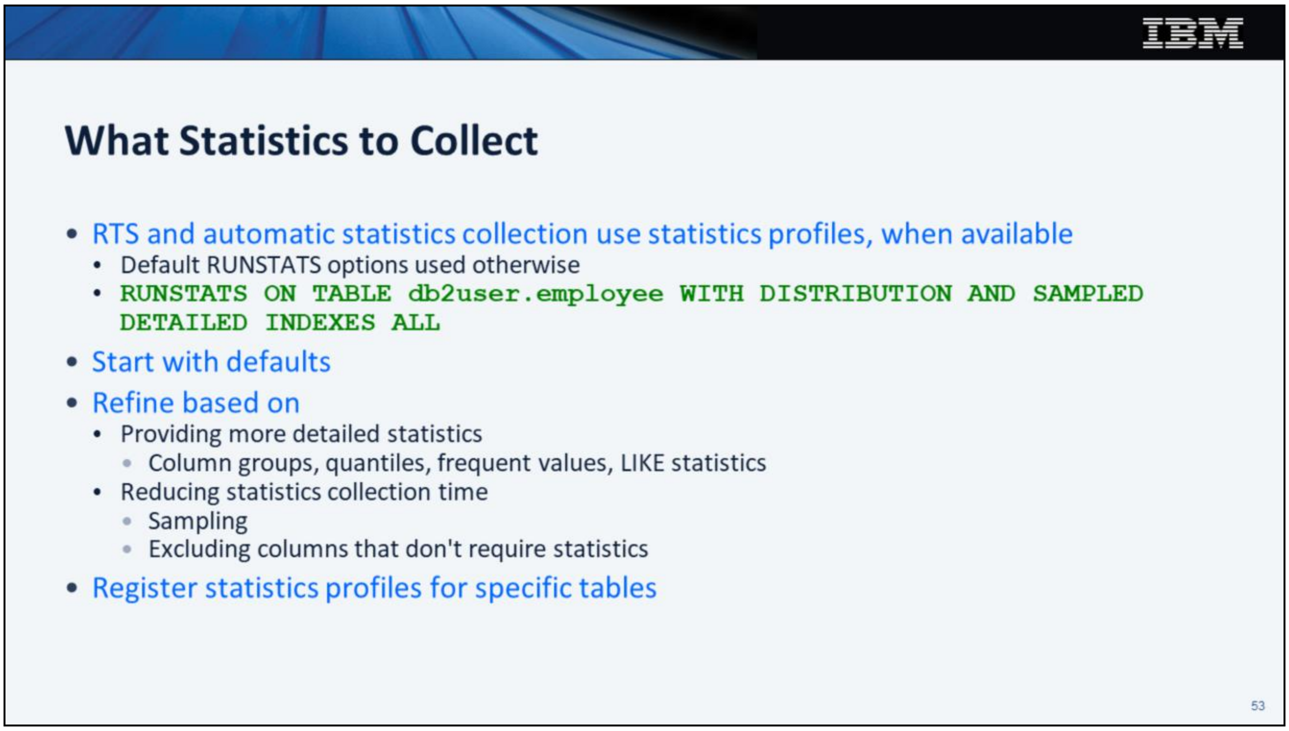
Handling exceptions

- Very large tables or 'sensitive' tables
  - Exclude (using automated maintenance policy)
    - Collect manually
  - Schedule (using automated maintenance policy)
  - Sample (using automatic sampling or a statistics profile)
- Volatile tables
  - Size of table grows and shrinks frequently
    - Common scenario – data completely deleted and repopulated
    - Difficult for statistics collection to get timely and consistent view
  - If explicitly marked VOLATILE -> automatically excluded
  - Manage:
    - Manually set representative statistics once
      - 1) Populate with representative data and do one RUNSTATS
      - 2) Collect statistics from another table using db2look and manually update statistics
    - Manually updating statistics also excludes from automatic collection



Utility Statistics Collection

- LOAD REPLACE
  - Supports all RUNSTATS options using statistics profiles
- REDISTRIBUTE DATABASE PARTITION GROUP
  - Supports all RUNSTATS options using statistics profiles
- CREATE INDEX
- Saves a second pass through the data
- More timely than automatic collection
  - Update the statistics when you know it will change



What Statistics to Collect

- RTS and automatic statistics collection use statistics profiles, when available
  - Default RUNSTATS options used otherwise
  - `RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL`
- Start with defaults
- Refine based on
  - Providing more detailed statistics
    - Column groups, quantiles, frequent values, LIKE statistics
  - Reducing statistics collection time
    - Sampling
    - Excluding columns that don't require statistics
- Register statistics profiles for specific tables

53