Training Guide

QAD BI v3 Technical Level 2

70-3306-3.11
BI Technical Level 2 v3.11
March 2015

This document contains proprietary information that is protected by copyright and other intellectual property laws. No part of this document may be reproduced, translated, or modified without the prior written consent of QAD Inc. The information contained in this document is subject to change without notice.

QAD Inc. provides this material as is and makes no warranty of any kind, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. QAD Inc. shall not be liable for errors contained herein or for incidental or consequential damages (including lost profits) in connection with the furnishing, performance, or use of this material whether based on warranty, contract, or other legal theory.

QAD and MFG/PRO are registered trademarks of QAD Inc. The QAD logo is a trademark of QAD Inc.

Designations used by other companies to distinguish their products are often claimed as trademarks. In this document, the product names appear in initial capital or all capital letters. Contact the appropriate companies for more information regarding trademarks and registration.

Copyright ©2015 by QAD Inc.

BI_TechLevel2_TG_v0311.pdf/r9m/b3s

QAD Inc.
100 Innovation Place
Santa Barbara, California 93108
Phone (805) 684-6614
Fax (805) 684-1890
http://www.qad.com

Contents

ABOUT THIS COURSE 8

QAD BI V3 – TECH LEVEL 2 CERTIFICATION - PART 1 11

Course Objectives 13
- Course Benefits 14
- Day 1 Schedule 15
- Day 2 Schedule 16
- Day 3 Schedule 17
- Day 4 Schedule 18
- Day 5 Schedule 19

SQL – Preview 20
- What is SQL? 21
- Why is SQL important for QAD BI v3? 22
- Basic Components of a SQL Statement 23
- Log in to Training Environment 24
- Launch SQL Server 25
- SELECT 26
- FROM 30
- WHERE 35
- Comparisons in a WHERE clause 36
- WHERE – continued 40
- Dealing with NULL values 41
- Inline Views 43
- WHERE – continued 44
- GROUP BY 49
- ORDER BY 50
- Native Functions 52
- Non-Native Functions 55
- Data Types 56
- Divide by Zero Error 59
- Divide by Zero Error – Solution: use CASE 60
- Divide by Zero Error – Due to Rounding 61
- Comparing Two Results Sets 62
- RANK OVER 64
- SQL Formatting Standards 66
- INSERT 67
- UPDATE 68
- DELETE 69
- TRUNCATE 70
- SQL to Avoid – Select within Select 71
- SQL to Avoid – Exception 72

SQL Review 73
HISTORY TABLES 74
HISTORY TABLES – Example 75
SNAPSHOT FROM HISTORY TABLE 77
SQL Review – Exercise 78

QAD BI V3 – TECH LEVEL 2 CERTIFICATION - PART 2 79

Day 2 Agenda 81
Day 1 Review 82

DEBUGGING 83
DEBUGGING – Tools 84

Troubleshooting Tools 90
SQL Server (SSMS) for Troubleshooting 91
Microsoft Excel for Troubleshooting 92
DWD for Troubleshooting 94
DWD for Troubleshooting – Rerunning Jobs 95

DWD – Features – Column Track Back 123
DWD – Features – Column Track Back – Exercise 128
DWD – Features – Search For String 129
DWD – Features – Column Transformations 131
DWD – How Data Flows Into the Warehouse 135
DWD for Troubleshooting 136
Notepad for Troubleshooting 138

Troubleshooting Case Study 139
Troubleshooting – Case 1 140

Troubleshooting Tips 146
Troubleshooting – Failed Jobs 147

Scheduler – /F Switch Failure 150

Review of History and Snapshot Tables 155

Troubleshooting 156
Troubleshooting Data Issues 157
Troubleshooting – Missing Data 159
Troubleshooting – Incorrect Data 167
Troubleshooting – Too Much Data 174
Troubleshooting – Too Much Data – Duplicate Records 176
Troubleshooting – Case 2 184
Troubleshooting – Case 3 185
Troubleshooting – Live Case 186

QAD BI V3 - TECH LEVEL 2 CERTIFICATION - PART 3 187

Day 3 Schedule 189

QAD Data Warehouse Designer 190
Builder Window 191

Customizations of BI v3 192
The Problem with Customizations 193
Why We Chose Our Methodology 194
QAD BI 3 Customization Methodology 204
Custom Path Methodology – Pros/Cons 208
Integrated Methodology (Non-Preferred) 210
Customization Naming Conventions 212
Mapping Out Customization Path 214
Customization Exercise 1 235
Live Support Issue 279

QAD BI V3 - TECH LEVEL 2 CERTIFICATION - PART 4 280

Day 4 Schedule 282
Day 3 Review 283

Performance Tuning 286
Table Performance 287
Table Performance – Indexes 289
Performance Feedback – Key Commands 297
Performance Feedback 298
Performance Feedback – PERFMON 305
Execution Plans 321
Execution Plans – Example 322
Execution Plans – Sample Scenario 323

Disk Space Issues 331
Running Out of Disk Space 332
Running Out of Disk Space – tempdb 334
Running Out of Disk Space – Shrink Option 338
Running Out of Disk Space – log file 344
Running Out of Disk Space – Better Fix 352
Running Out of Disk Space – Other Option 353
Running Out of Space and Shrink Option. 357

Troubleshooting Performance 358
Common Performance Issues 359
Performance Issue Troubleshooting 363

Working With Large Tables 379
DELETE vs. COPY/TRUNCATE/INSERT 380

Index Issues 391
Index Fragmentation 392

Performance Issues – Class Excercise 396
stage_inv_trx_chg_months 397
stage_inv_bal_mth_date1 398
Live Support Issue 400

Exam Review 402

QAD BI Tech Level 2 Certification Change Summary

The following table summarizes significant differences between this document and the previous version.

Date/Version	Description	Reference
March 2015/QAD BI 3.11	Rebranded for QAD BI 3.11	--
March 2014	1st Release	--
		--
		--

About This Course

Course Description

This course builds on BI V3 Level 1 Certification combined with on-the-job experience to prepare the student to provide Tier 2 technical support for the BI V3 product.

The following topics are covered:

- **BI3 Customization.** Adding new data sources, fields, cubes, fact and dimension tables.
- **Troubleshooting.** Installation issues, including Data Warehouse Designer, Modules, BI Portal, Tomcat and ODBC connections. Historical data reloads.
- **Customer support.** Diagnose and resolve customer issues. Create patches as needed.

Course Objectives

The following are the objectives of the BI Technical Level 2 Certification Class:

- Introduction to/overview of SQL and the most common elements encountered in our BI solution.
- Overview and detailed hands on experience with BI troubleshooting tools and techniques.
- Customization methodologies to ensure minimal impact to upgrades and patches.
- Performance tuning methodologies.

This class provides the following benefits:

- Additional personal skill sets
- Customer satisfaction
- BI is a cornerstone of QAD's future plans
- Support – better able to troubleshoot customer issues and pass concrete bug reports to R&D
- Services – better able to meet customer's individual customization needs and ensure trouble free upgrades and patches

Audience

The BI Technical Level 2 Certification Class is intended for the following audience:

- QAD and QAD Partner services consultants who will provide Tier 2 support to other QAD and Partner consultants during BI implementation.
- QAD and Partner services consultants who will design, develop and implement customizations to BI v3.
- QAD support consultants who will trouble shoot customer issues and create patches to resolve customer problems.

Prerequisites

The following prerequistes are required for the BI Technical Level 2 Certification Class:

- Business Intelligence (BI) V3 - Technical - Level 1 Class and Certification
- SQL Server 2008 - Intermediate
- Transact-SQL - Intermediate
- Read: The Data Warehouse Toolkit: Second Edition. Ralph Kimball.

Course Credit and Scheduling

This course provides 30 credit units. It is designed to be taught in 5 days.

Virtual Environment Information

The hands-on exercises in this book should be used with the latest BI learning environment.

Additional Resources

If you encounter questions on QAD software that are not addressed in this book, several resources are available. The QAD corporate Web site (http://www.qad.com/) provides product and company overviews. From the main site, you can access the QAD Learning or Support site and the QAD Document Library. Access to some portions of these sites depends on having a registered account.

QAD Learning Center

To view available training courses, locations, and materials, use the QAD Learning Center. Choose Education under the Services tab to access this resource. In the Learning Center, you can reserve a learning environment if you want to perform self-study and follow a training guide on your own.

QAD Document Library

To access release notes, user guides, training guides, and installation and conversion guides by product and release, visit the QAD Document Library. Choose Document Library under the Support tab. In the QAD Document Library, you can view HTML pages online, print specific pages, or download a PDF of an entire book. To find a resource, you can use the navigation tree on the left or use a powerful cross-document search, which finds all documents with your search terms and lets you refine the search by book type, product suite or module, and date published.

QAD Support

Support also offers an array of tools depending on your company's maintenance agreement with QAD. These include the Knowledgebase and QAD Forums, where you can post questions and search for topics of interest. To access these, choose Visit Online Support Center under the Support tab.

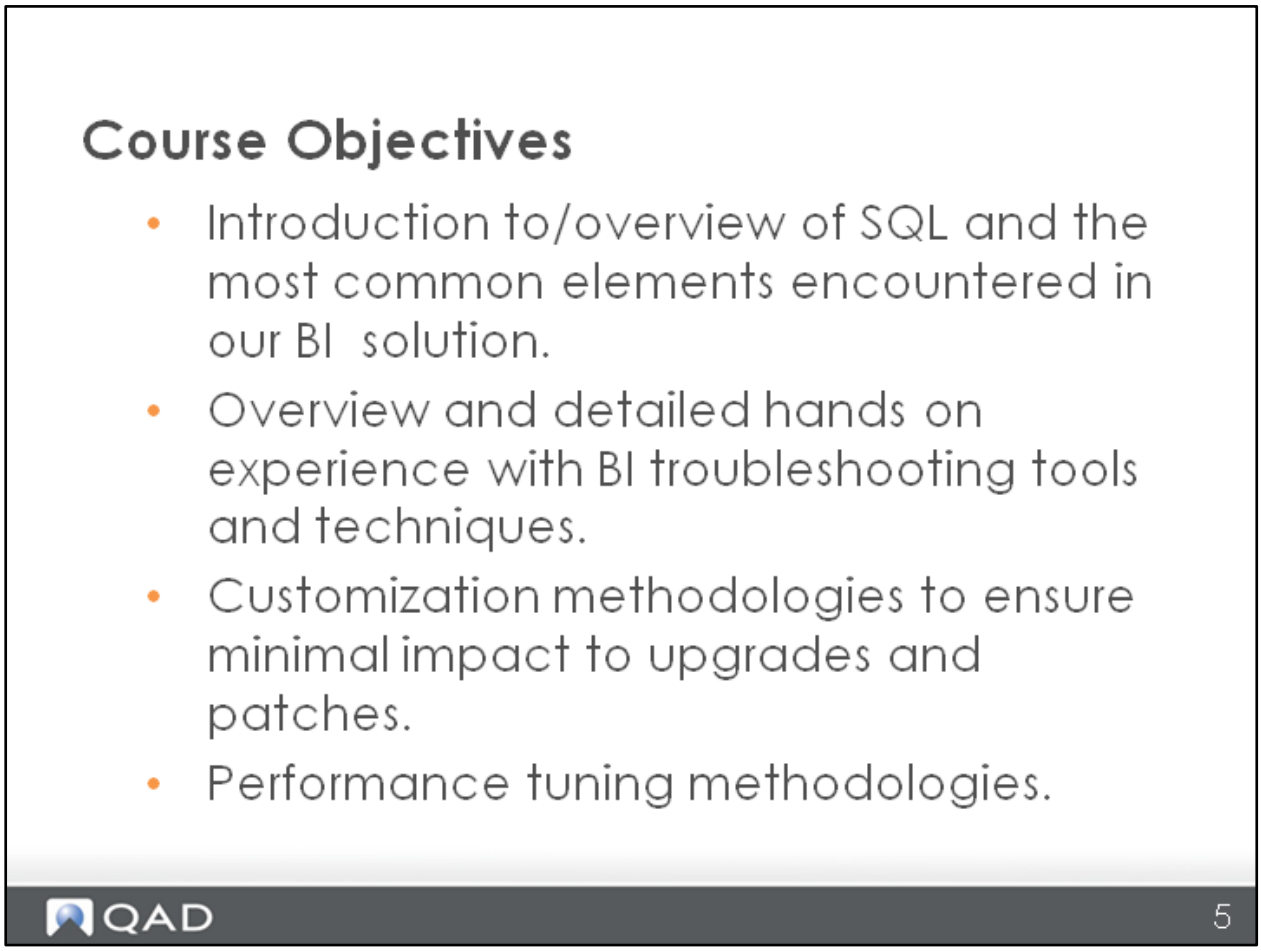
CHAPTER 1

QAD BI v3 – Tech Level 2 Certification - Part 1

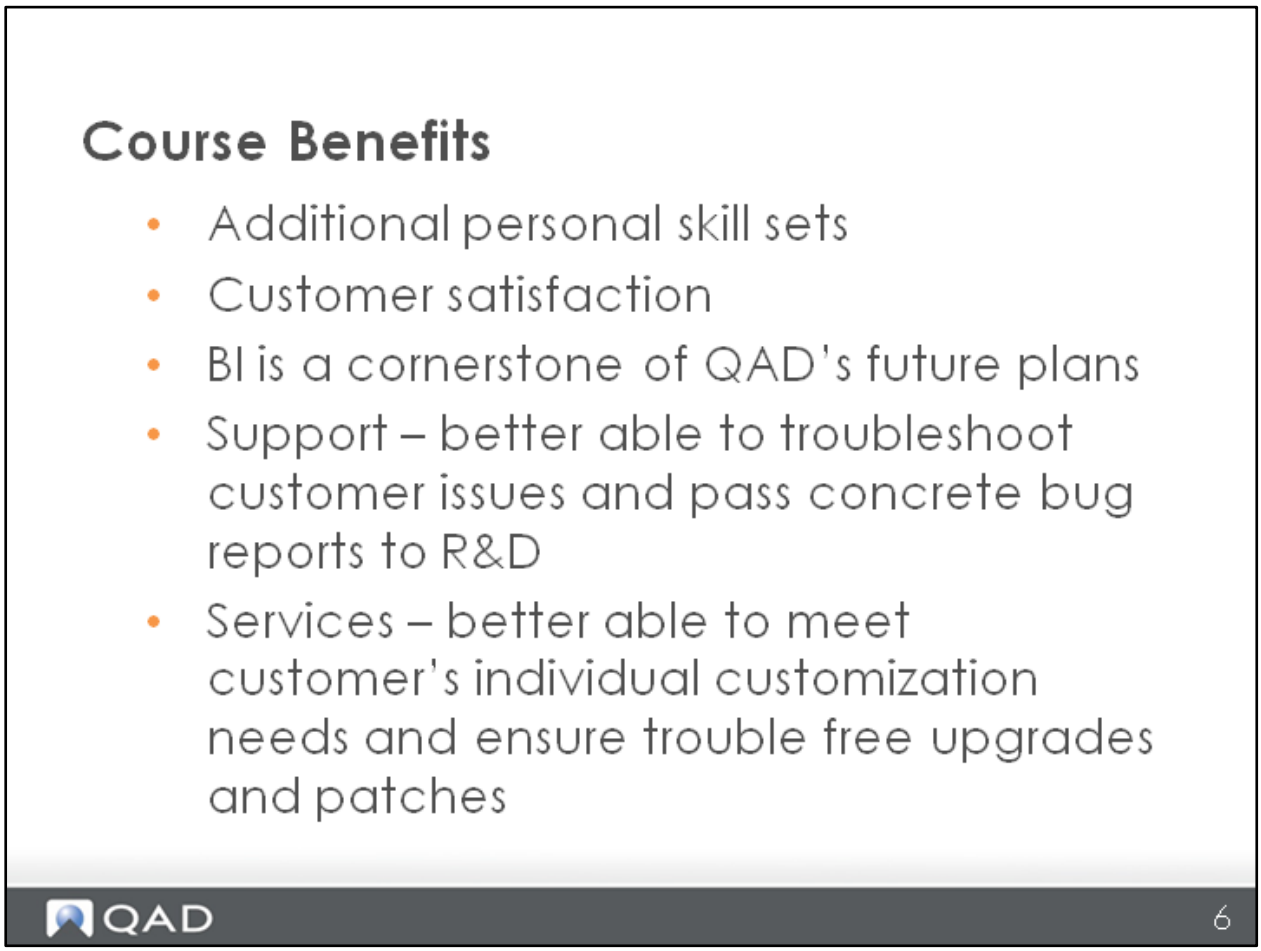
QAD BI v3 – Technical Level II Certification Class – Part 1



Course Objectives



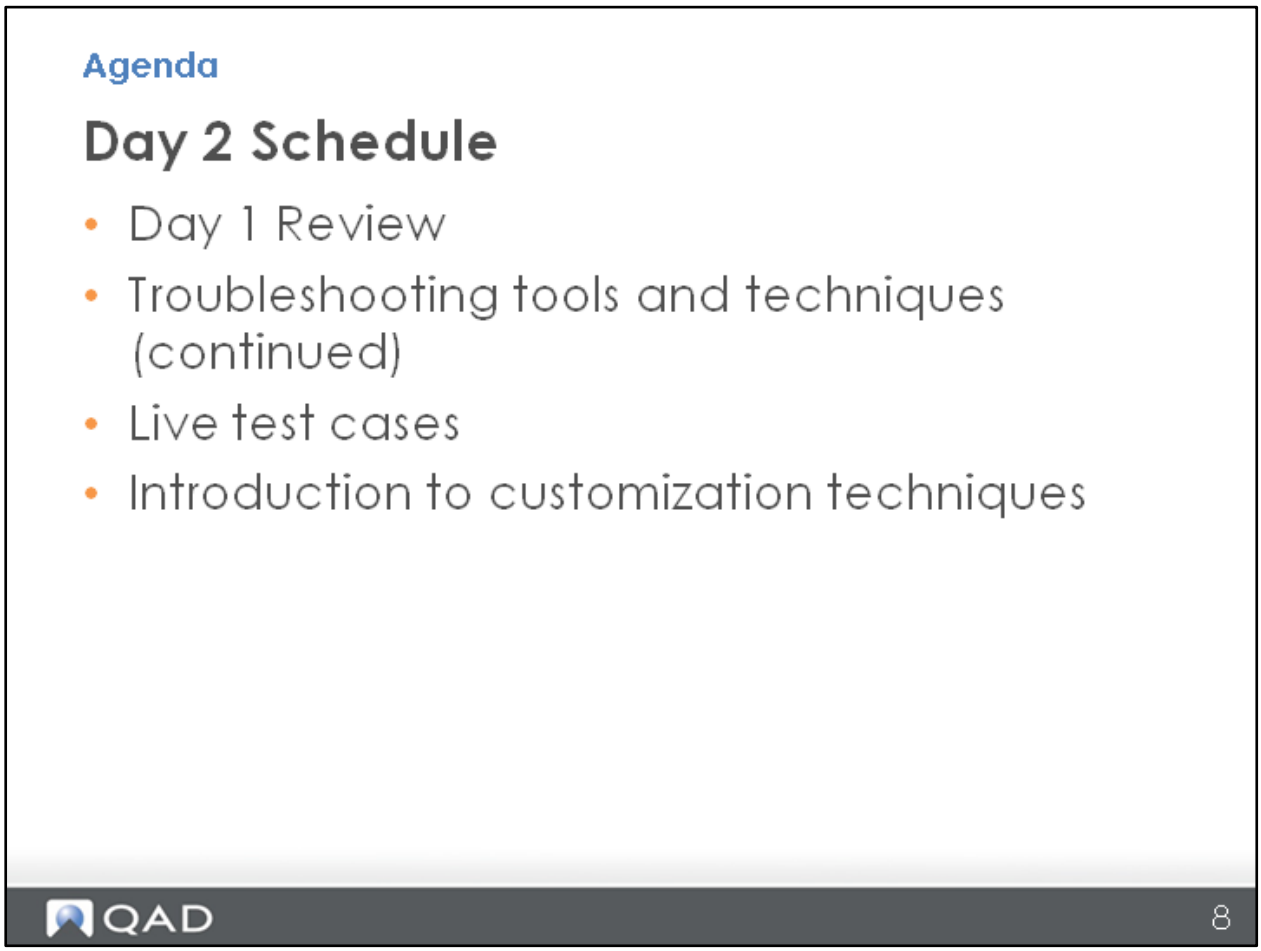
Course Benefits



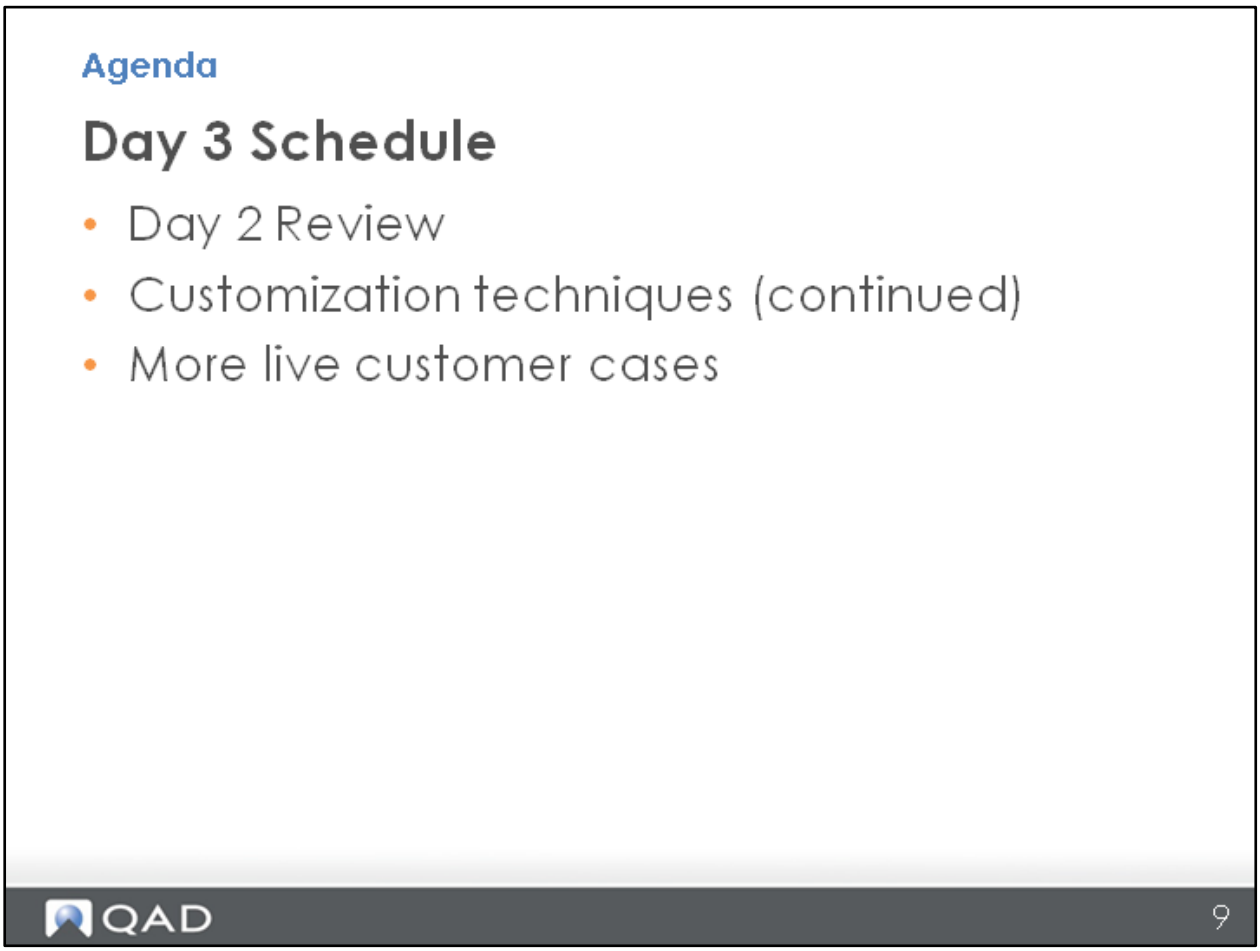
Day 1 Schedule



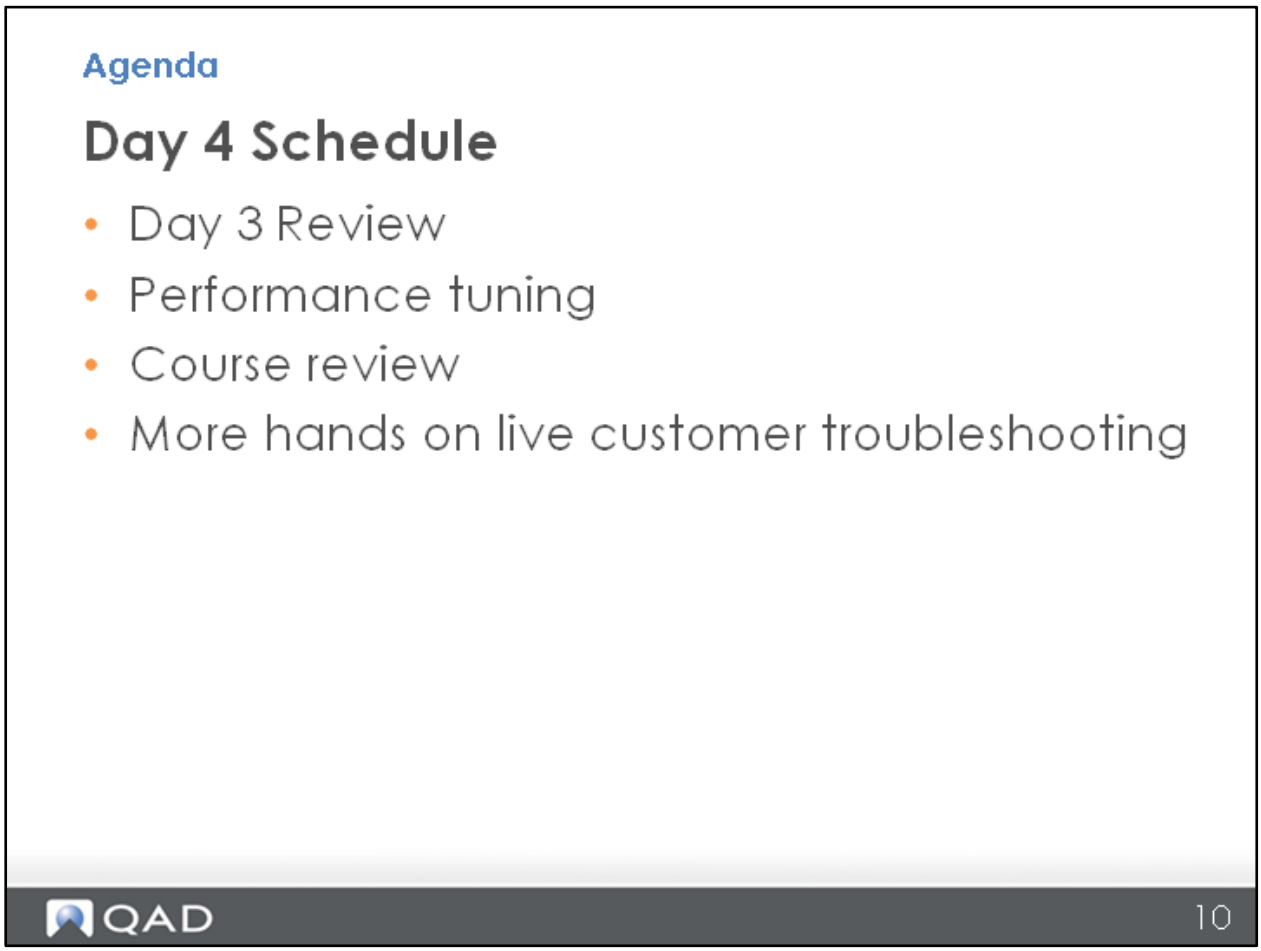
Day 2 Schedule



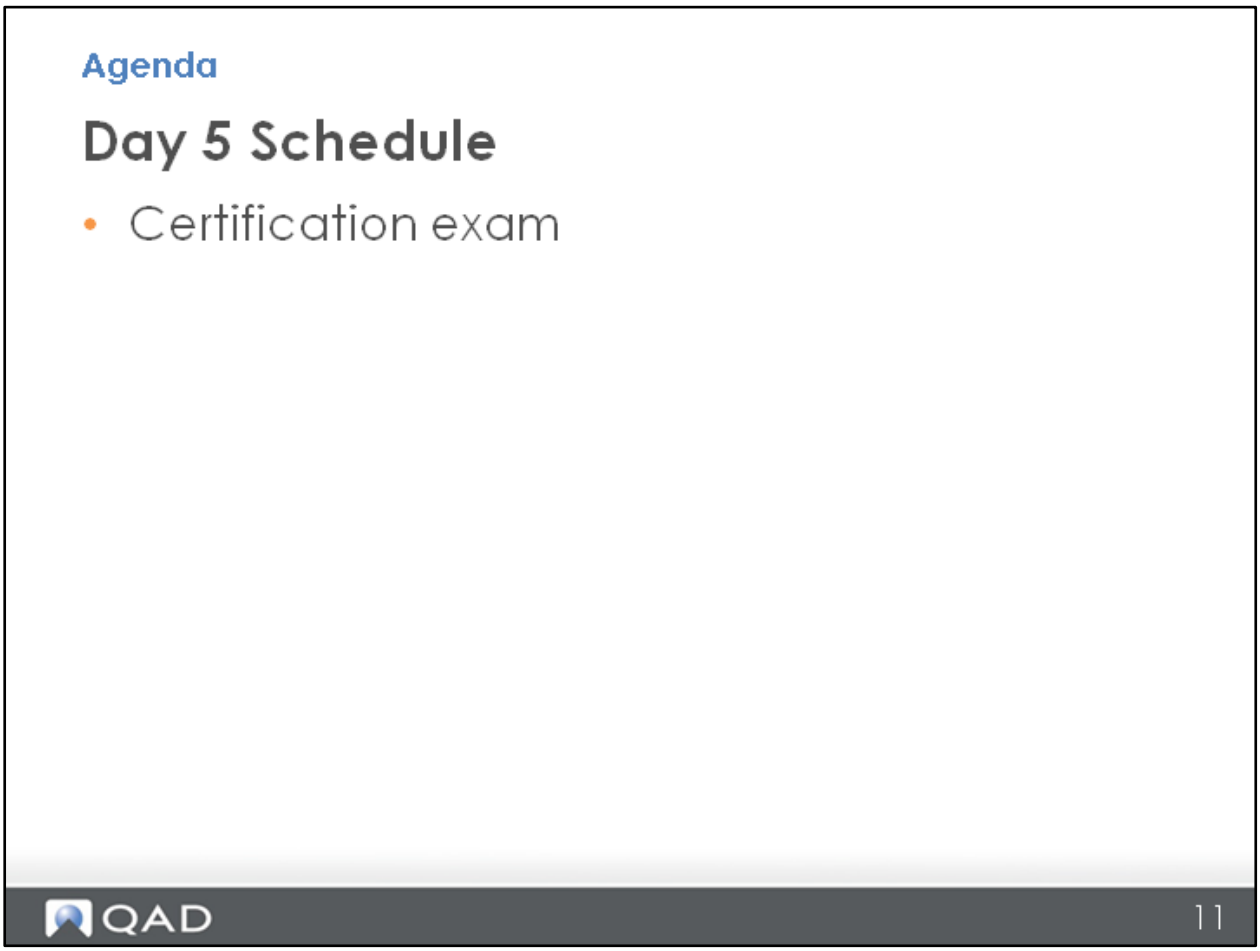
Day 3 Schedule



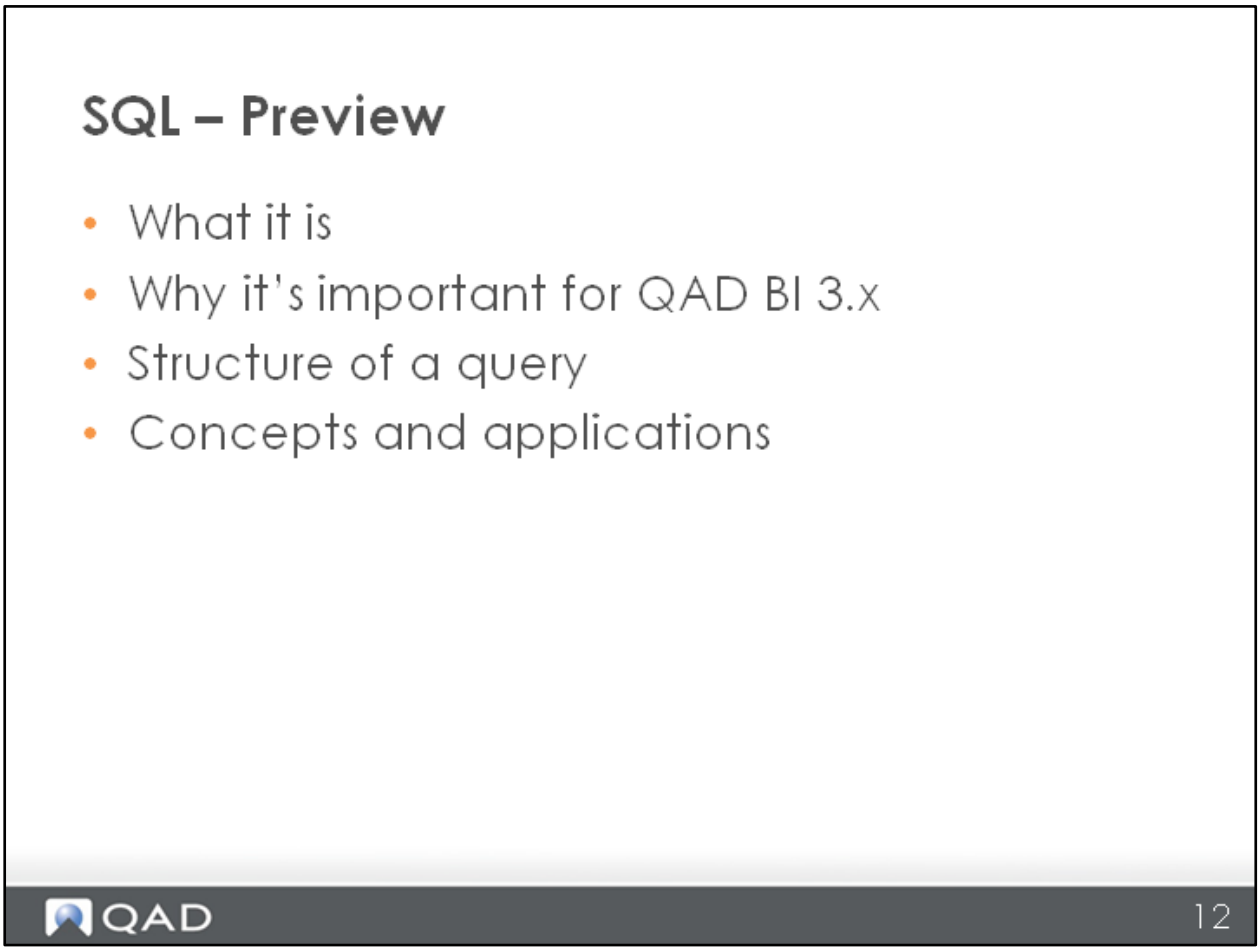
Day 4 Schedule



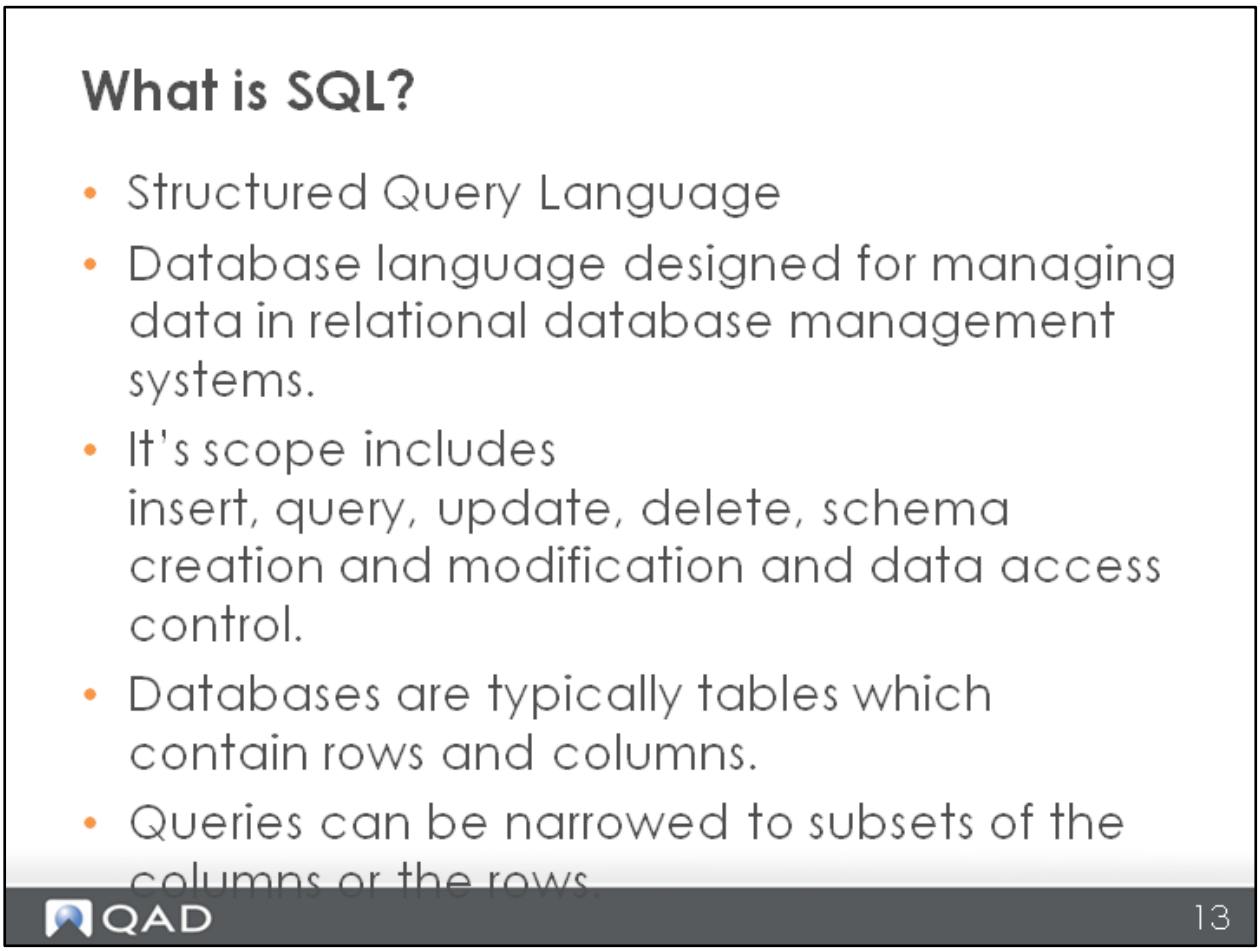
Day 5 Schedule



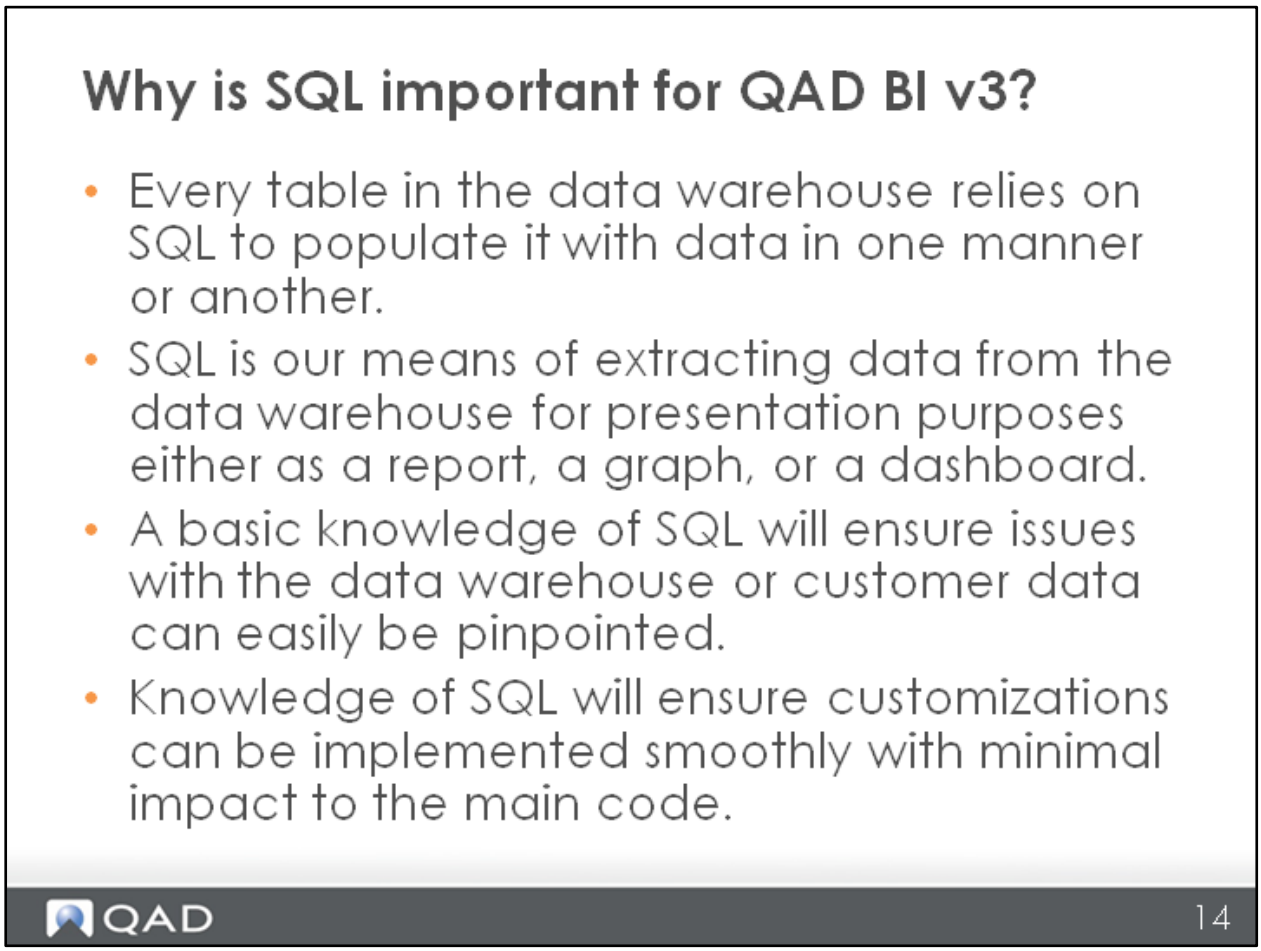
SQL – Preview


SQL – Preview
What it is
Why it's important for QAD BI 3.x
Structure of a query
Concepts and applications
QAD
12

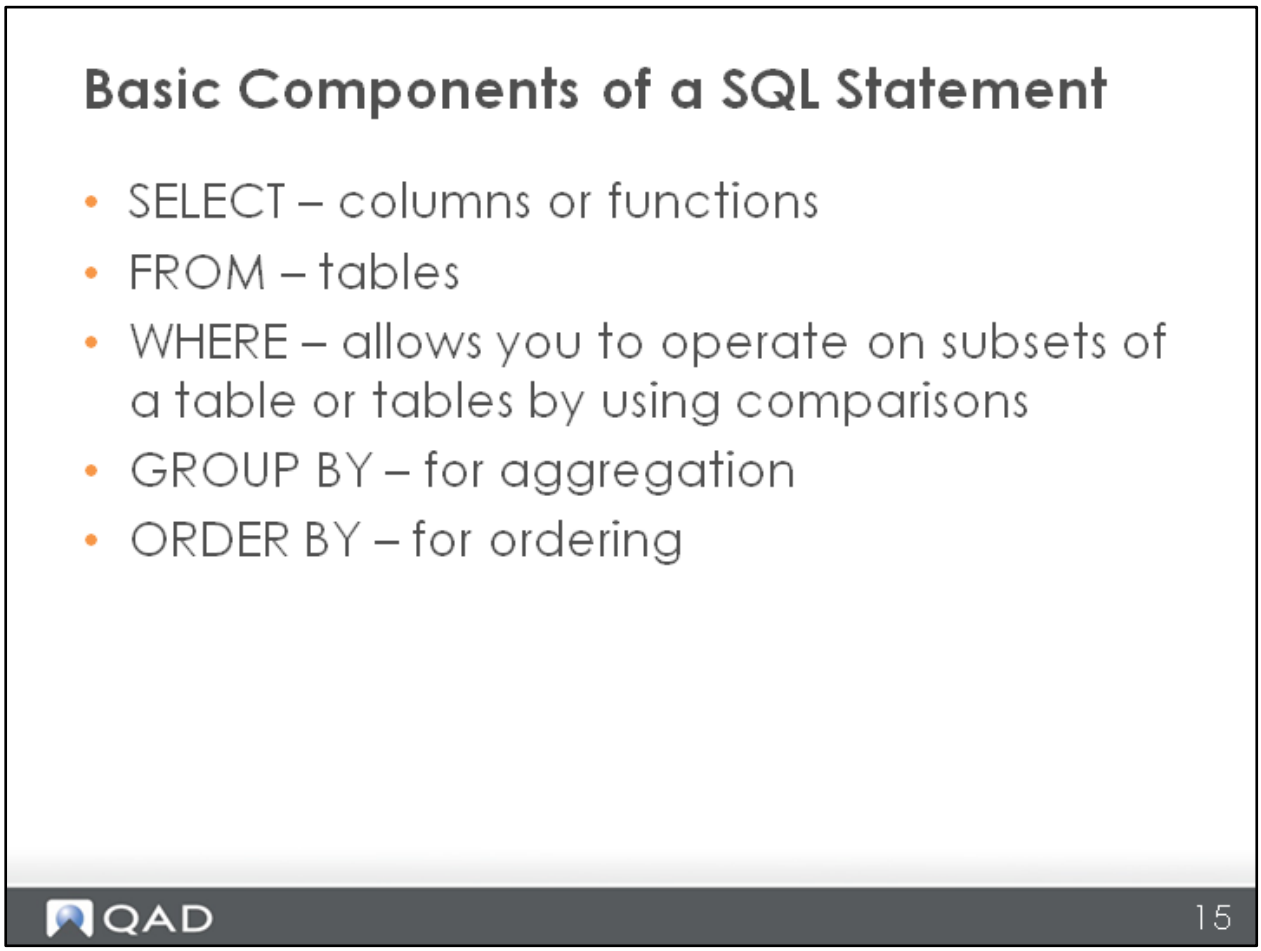
What is SQL?


What is SQL?
Structured Query Language
Database language designed for managing data in relational database management systems.
It's scope includes insert, query, update, delete, schema creation and modification and data access control.
Databases are typically tables which contain rows and columns.
Queries can be narrowed to subsets of the columns or the rows.
QAD
13

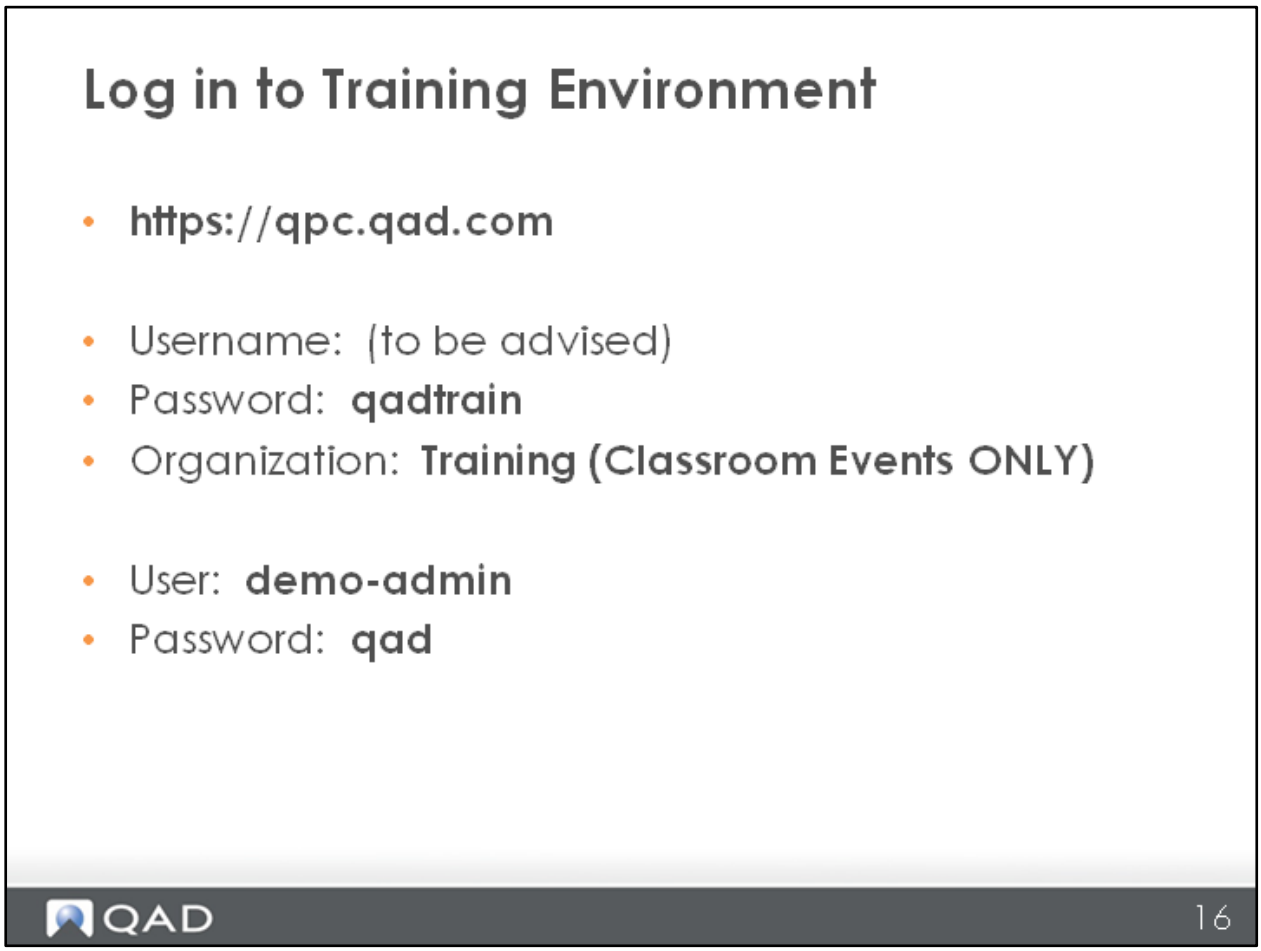
Why is SQL important for QAD BI v3?



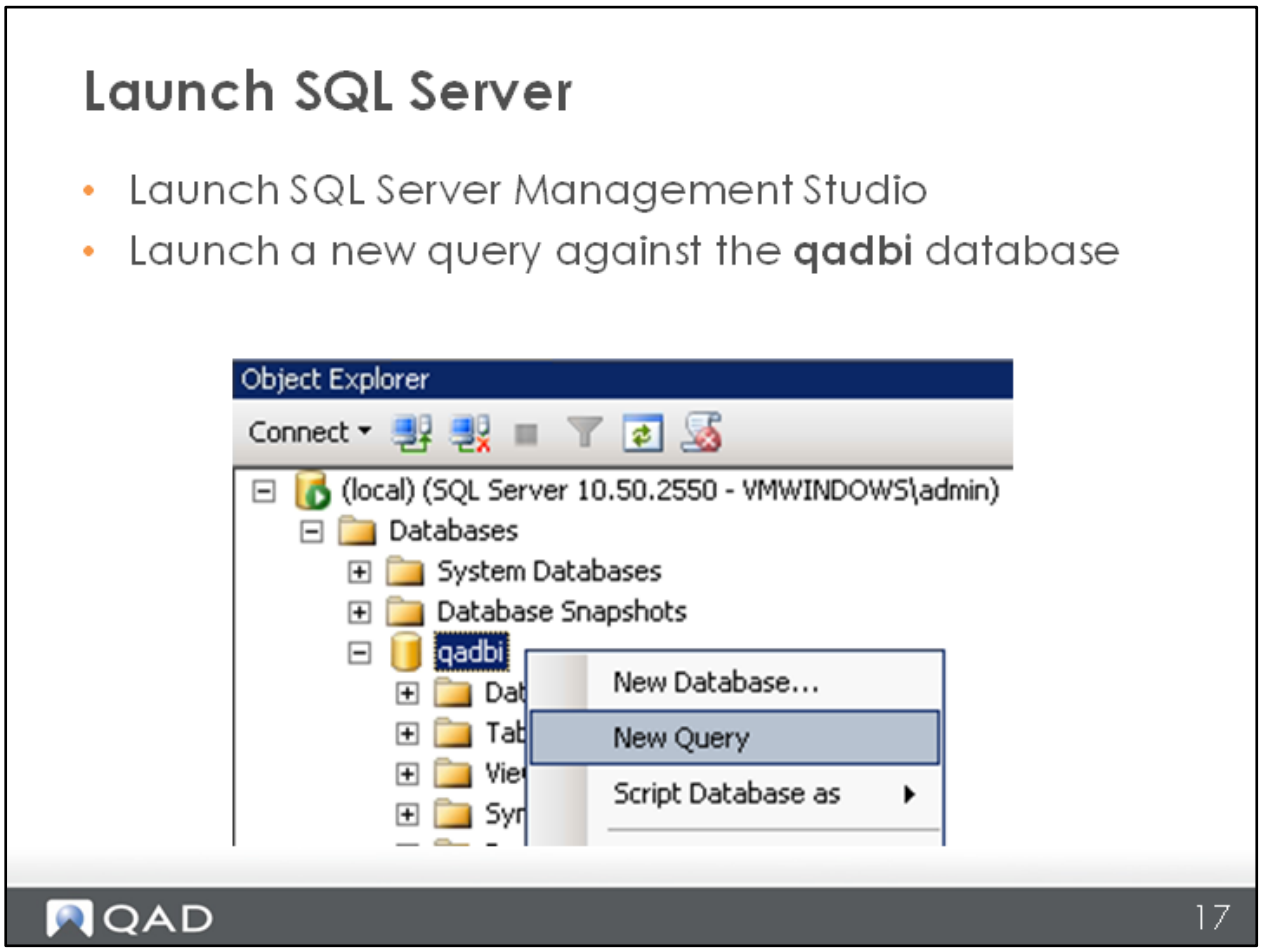
Basic Components of a SQL Statement



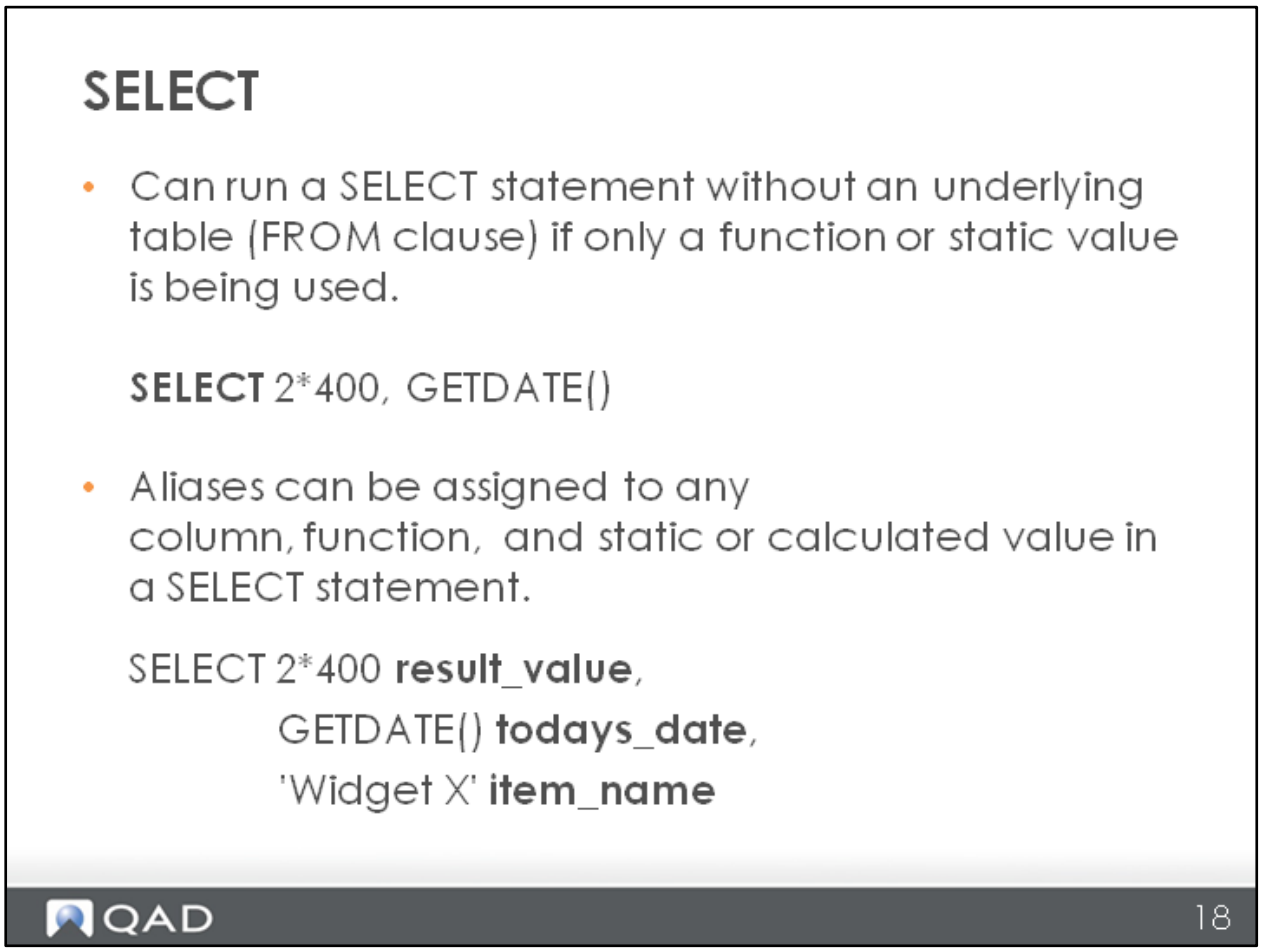
Log in to Training Environment



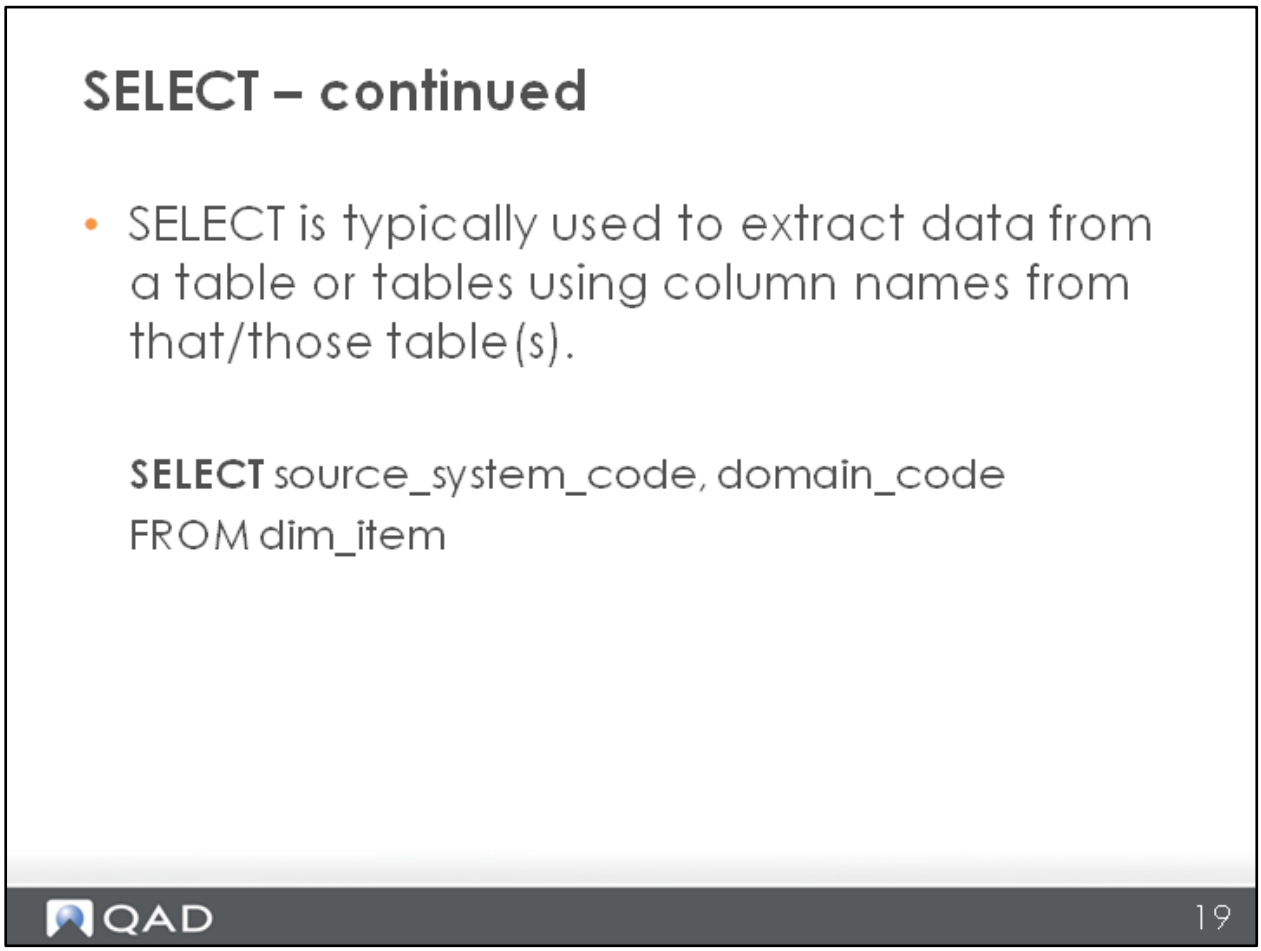
Launch SQL Server



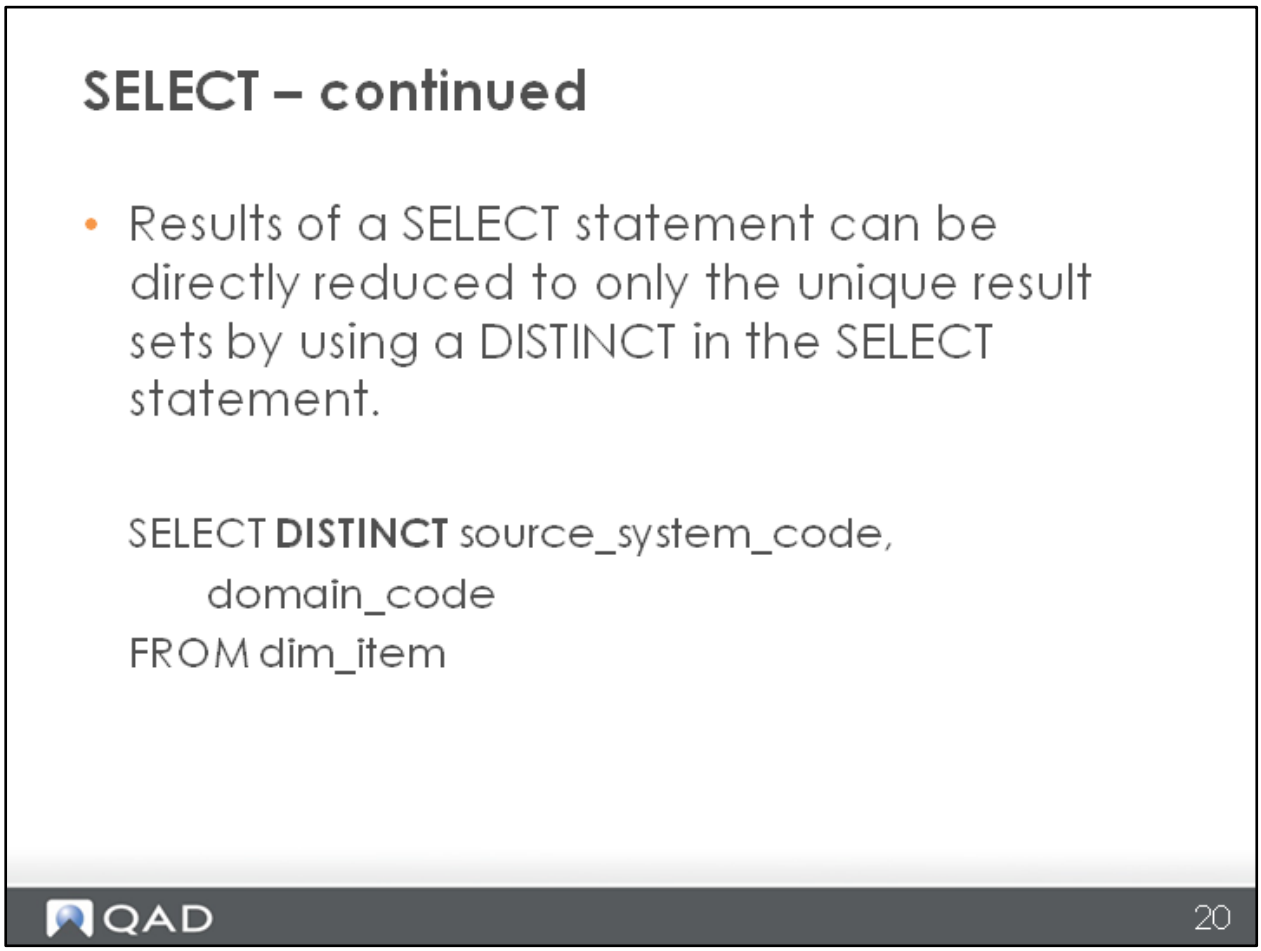
SELECT


SELECT
Can run a SELECT statement without an underlying table (FROM clause) if only a function or static value is being used.
SELECT 2*400, GETDATE()
Aliases can be assigned to any column, function, and static or calculated value in a SELECT statement.
SELECT 2*400 result_value,
GETDATE() todays_date,
'Widget X' item_name
QAD
18

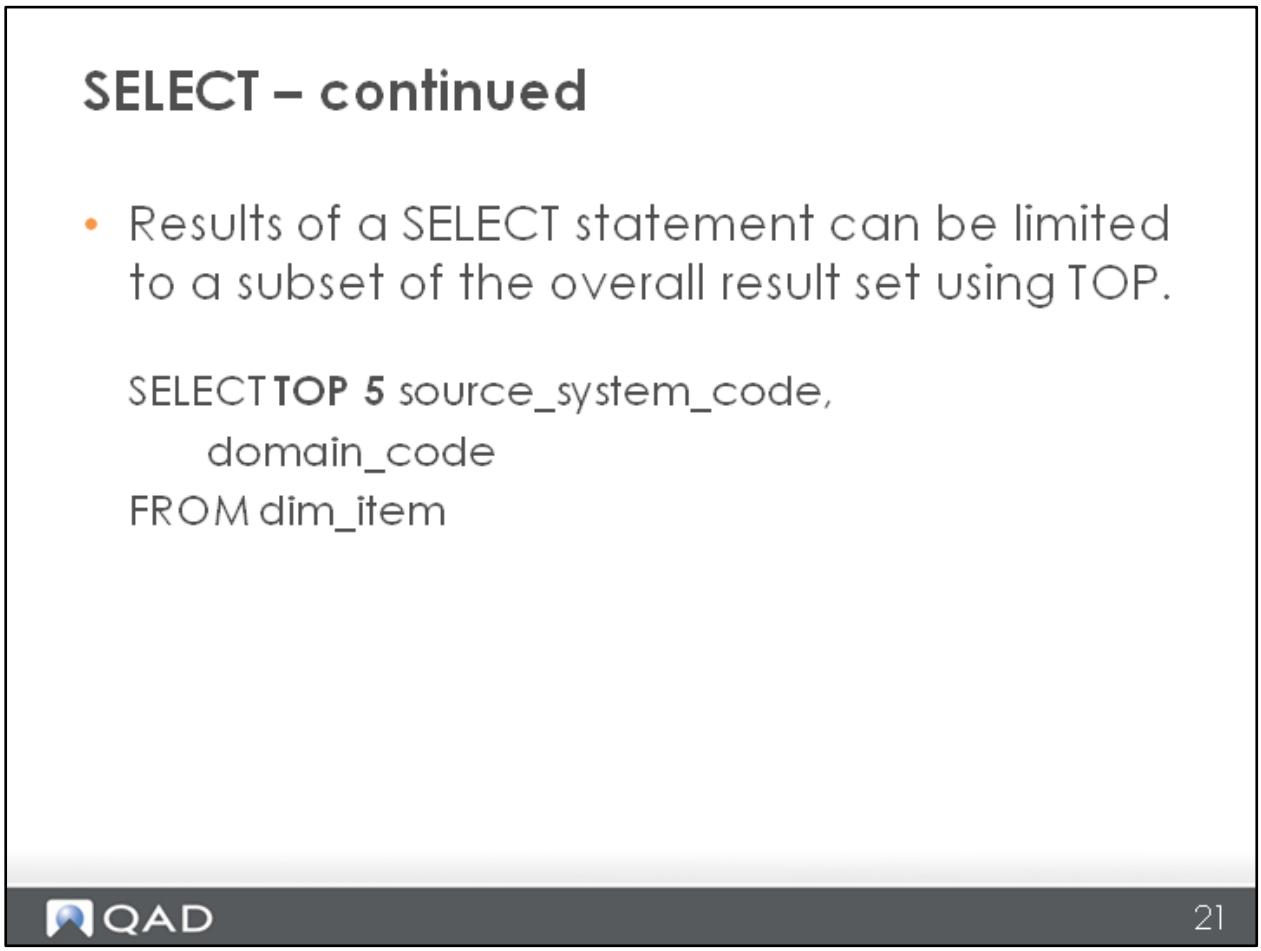
SELECT – continued



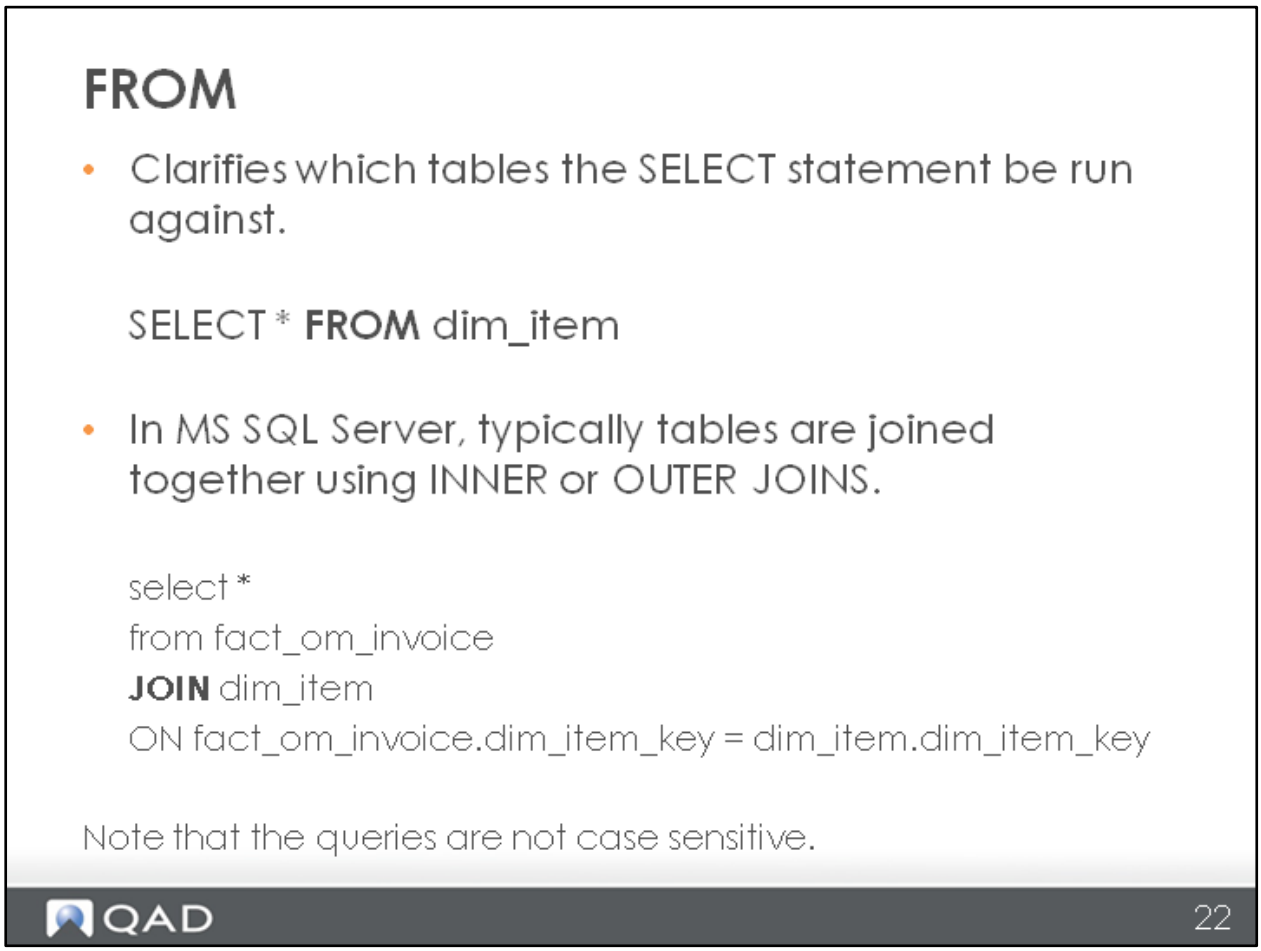
SELECT – continued



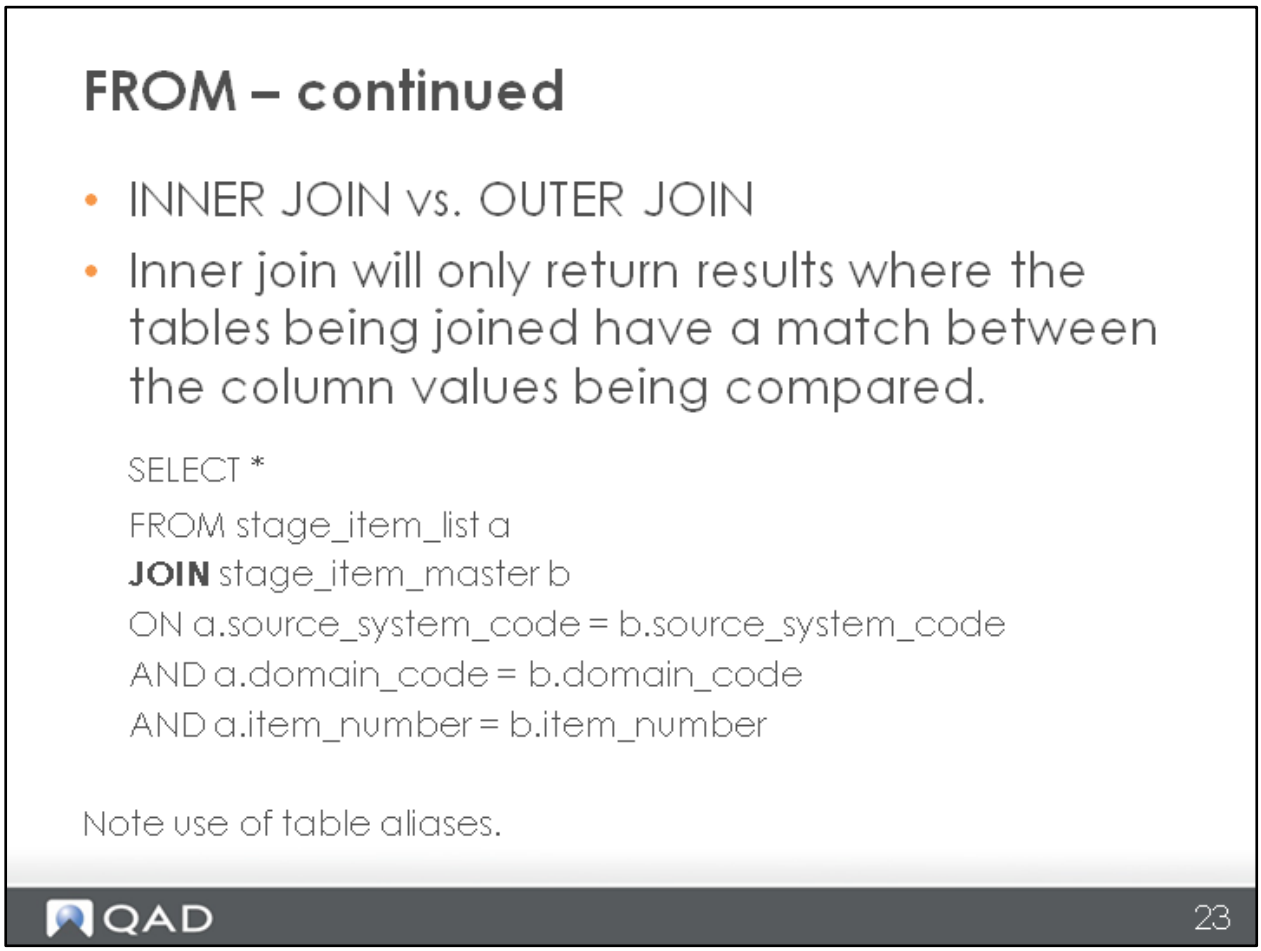
SELECT – continued



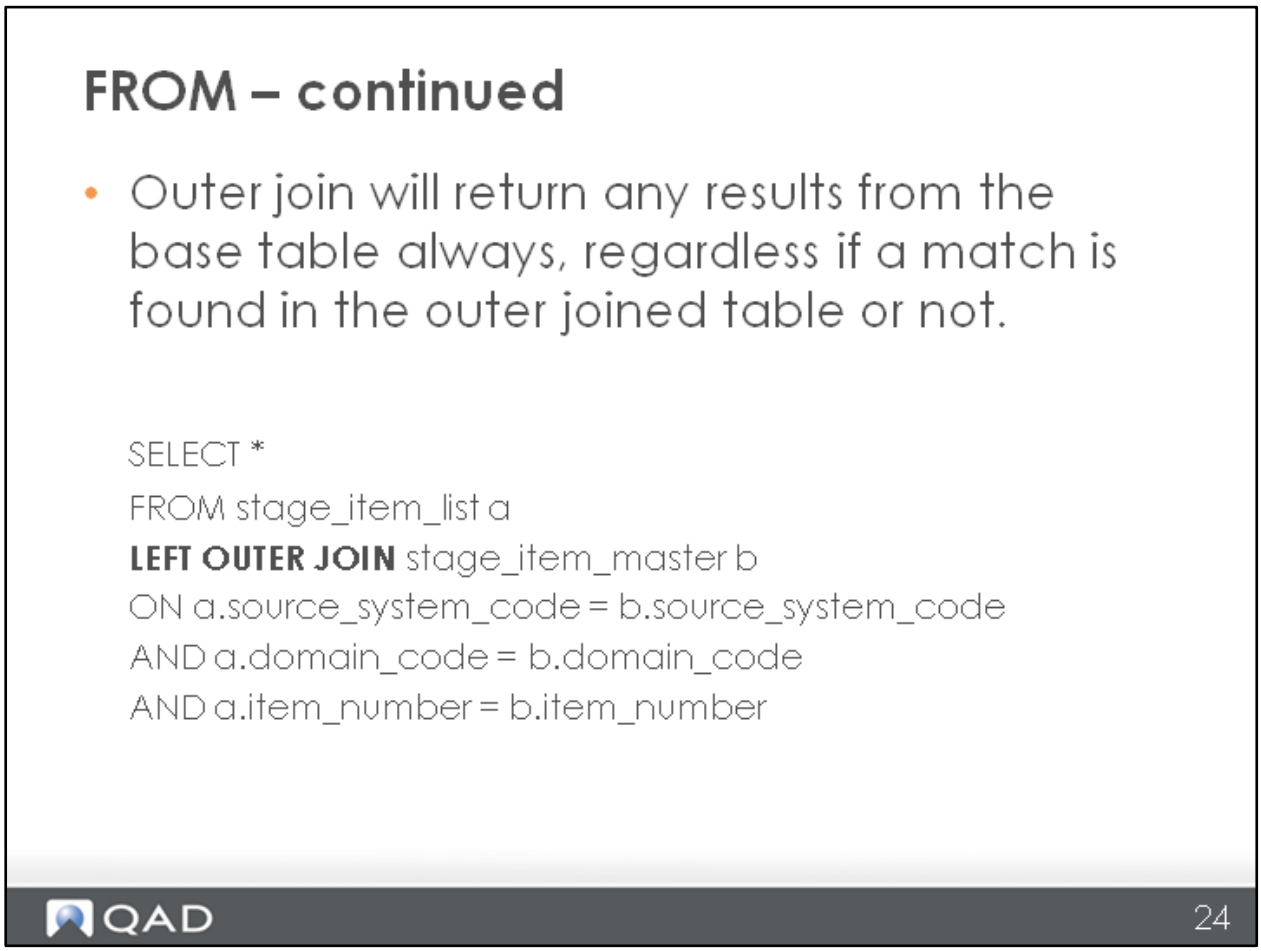
FROM


FROM
• Clarifies which tables the SELECT statement be run against.
SELECT * FROM dim_item
• In MS SQL Server, typically tables are joined together using INNER or OUTER JOINS.
select *
from fact_om_invoice
JOIN dim_item
ON fact_om_invoice.dim_item_key = dim_item.dim_item_key
Note that the queries are not case sensitive.
QAD
22

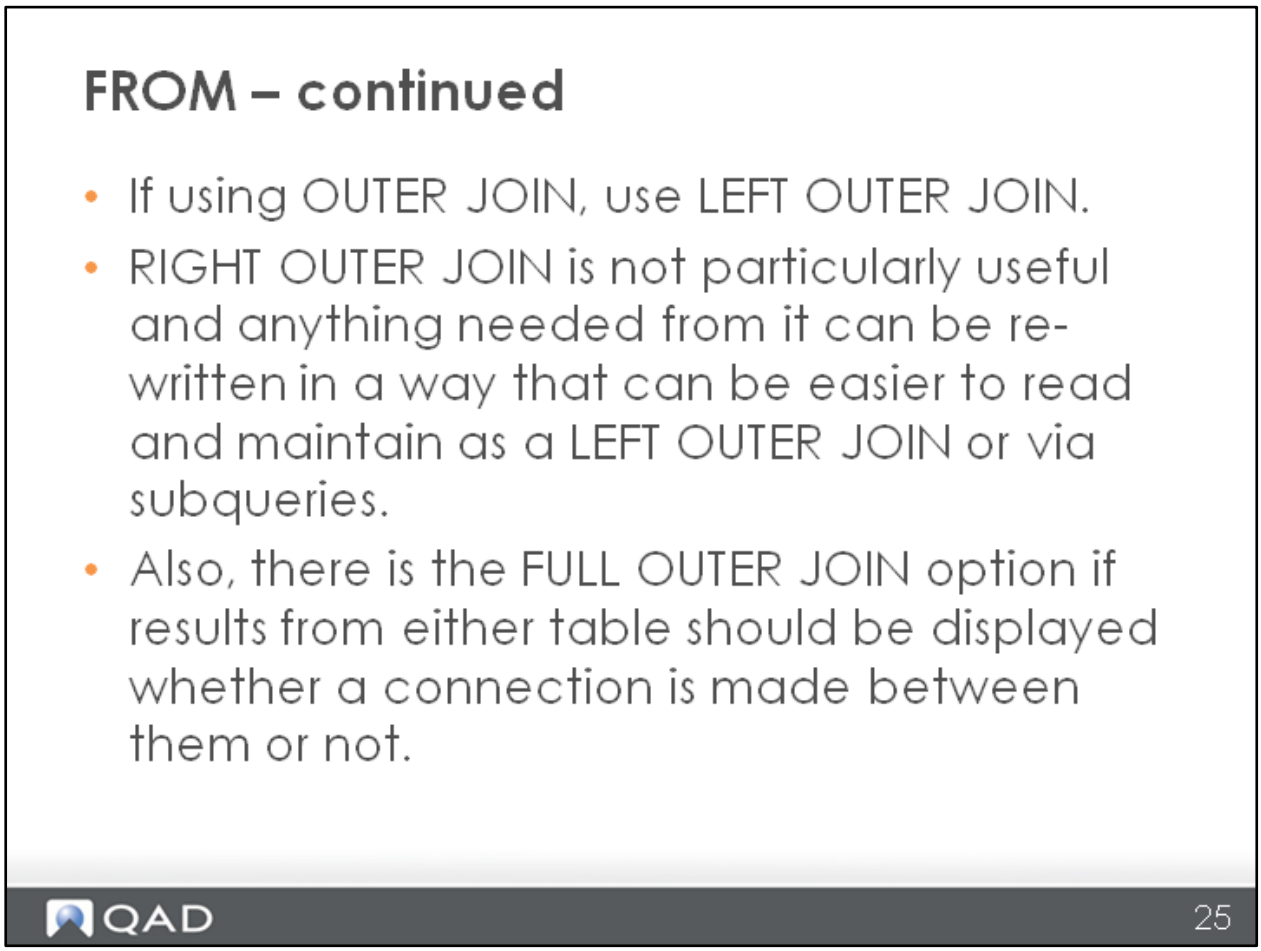
FROM – continued



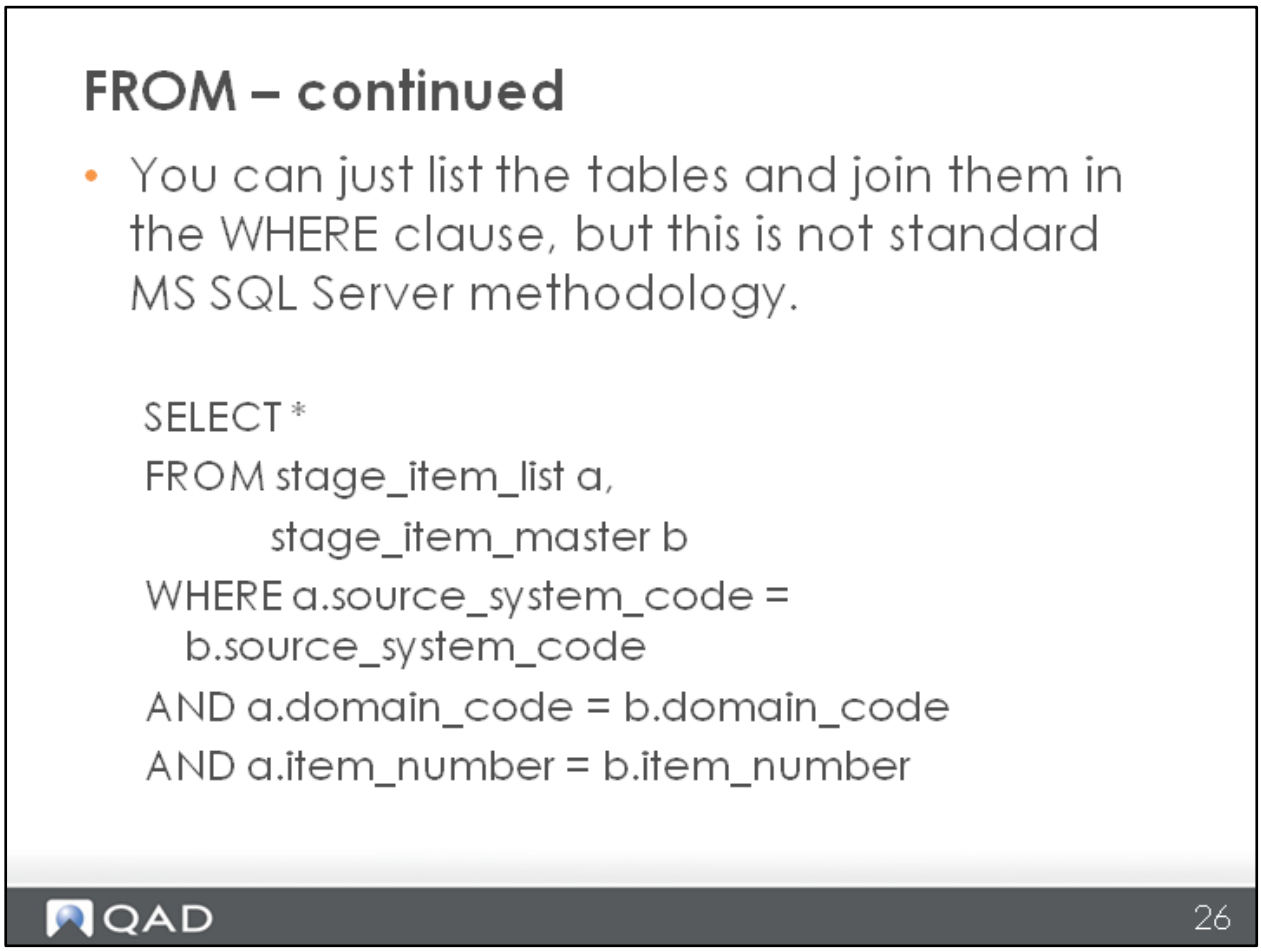
FROM – continued



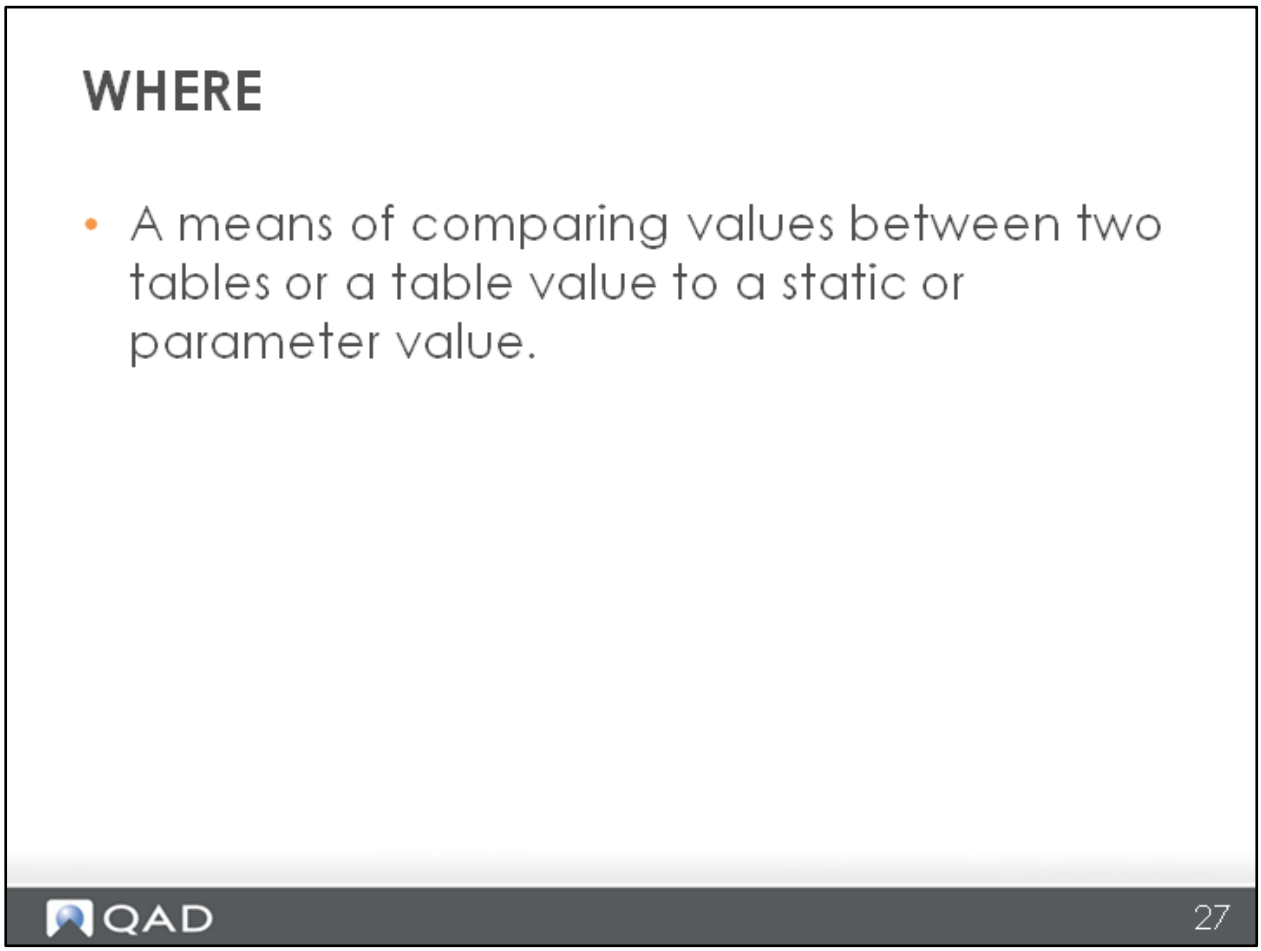
FROM – continued



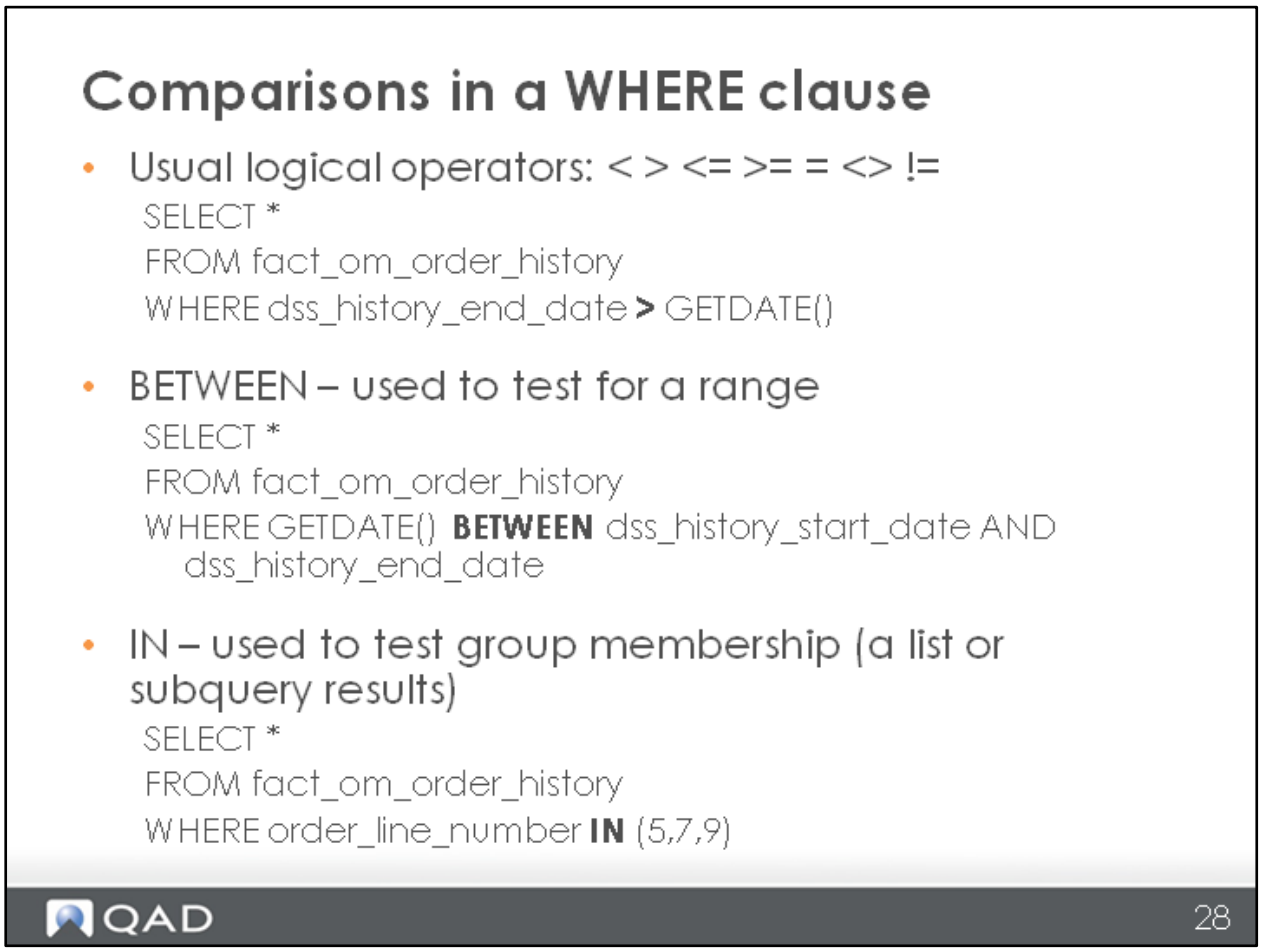
FROM – continued



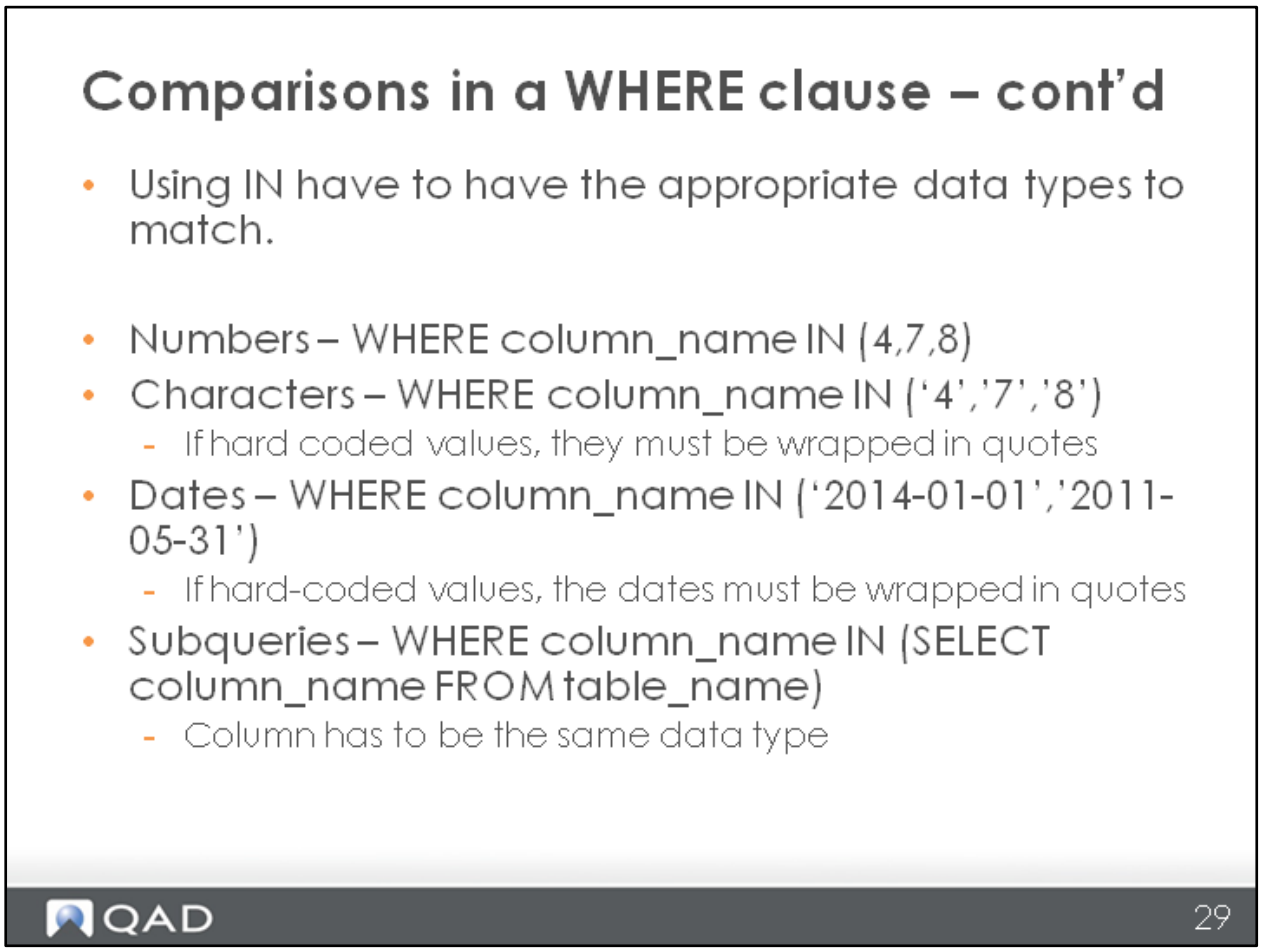
WHERE


WHERE
A means of comparing values between two tables or a table value to a static or parameter value.
QAD
27

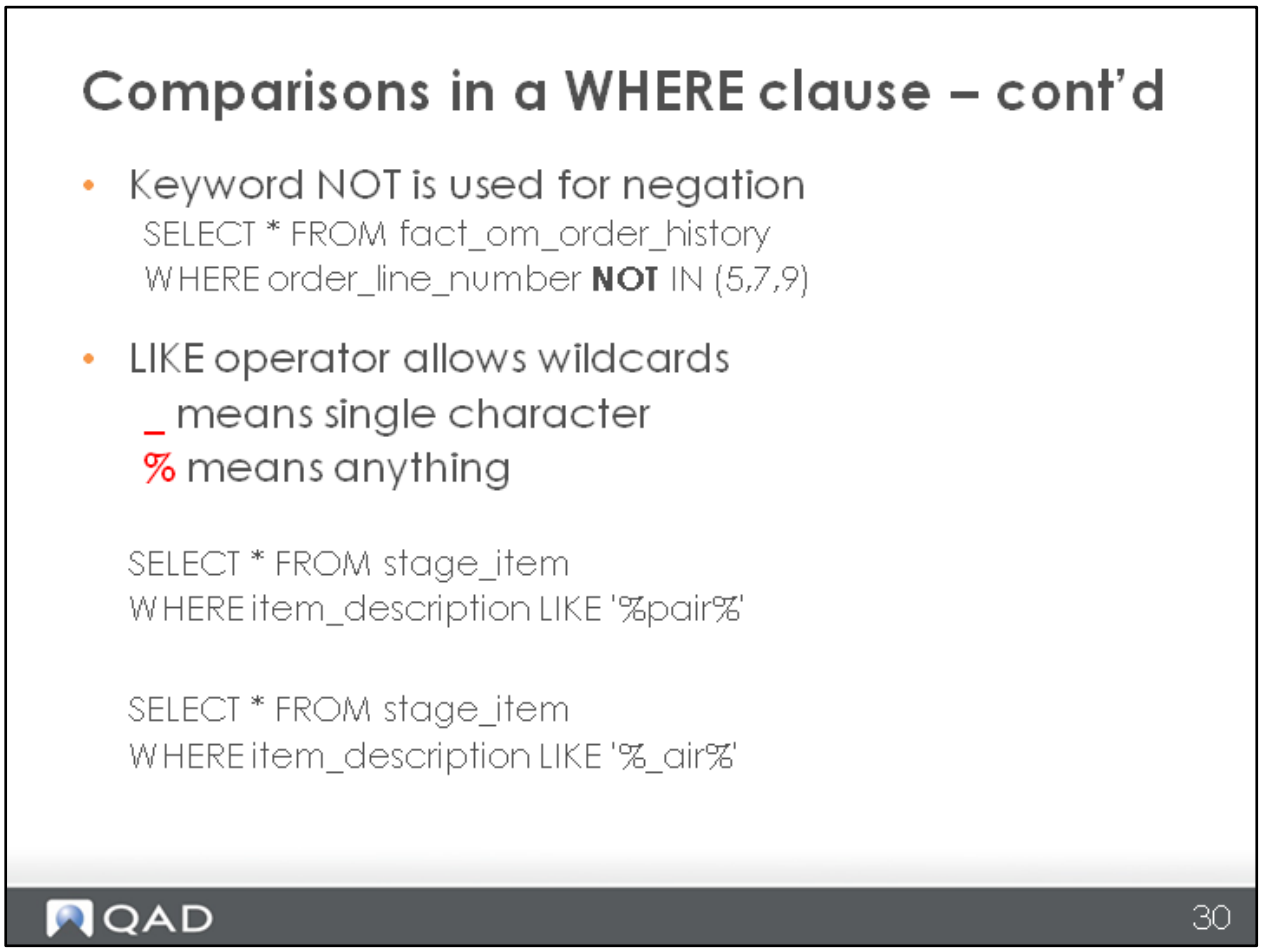
Comparisons in a WHERE clause



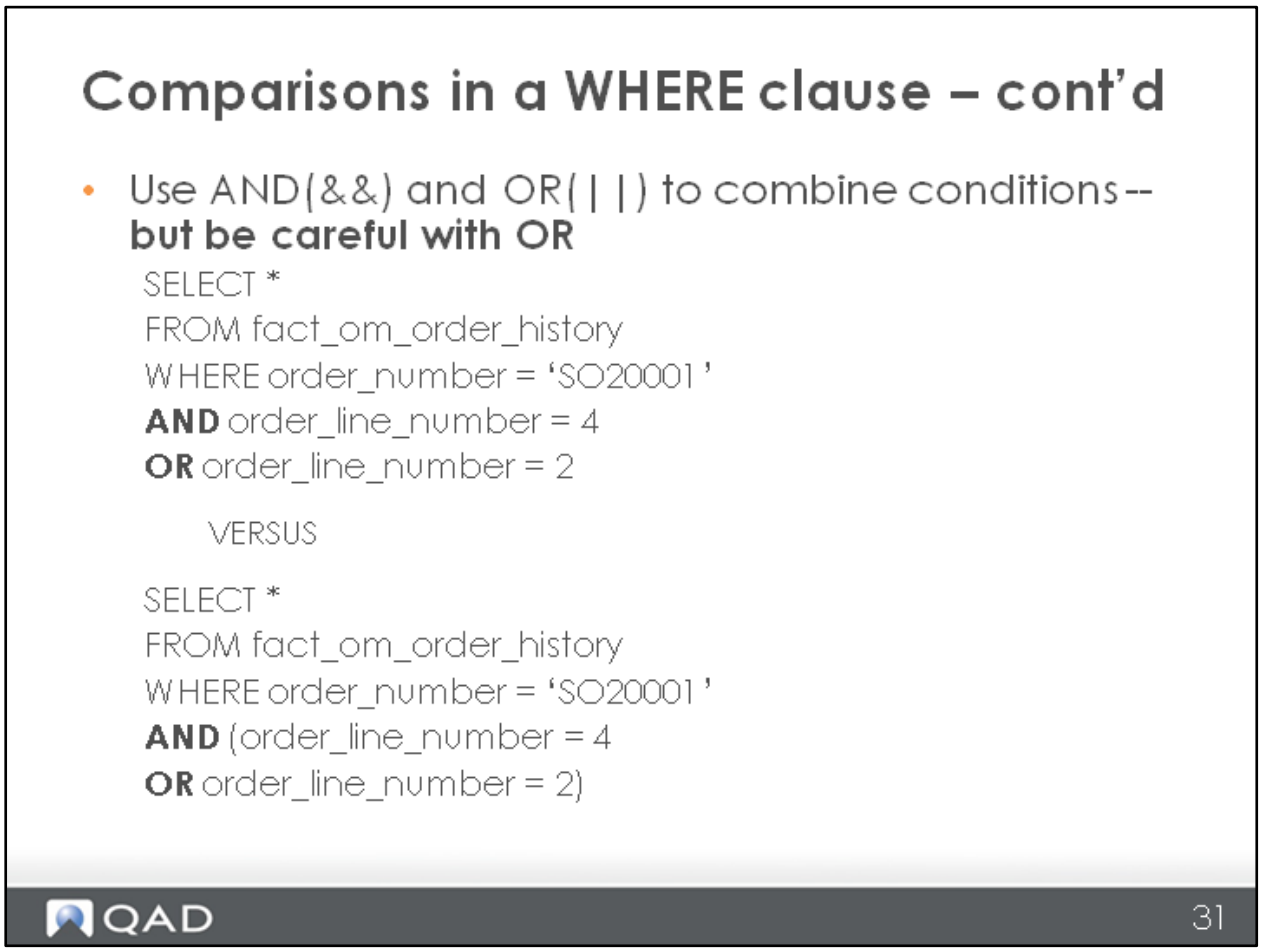
Comparisons in a WHERE clause – cont'd



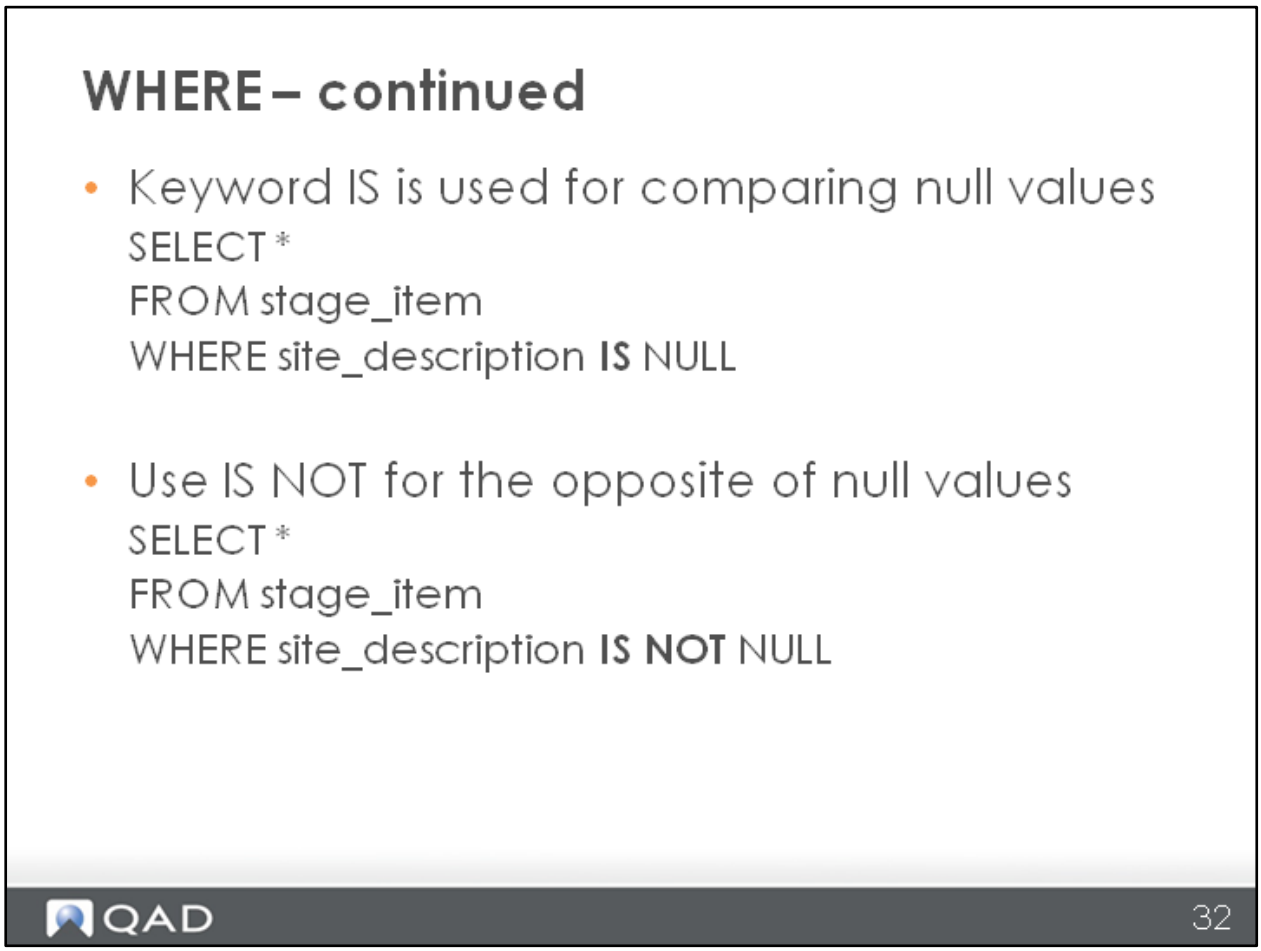
Comparisons in a WHERE clause – cont'd



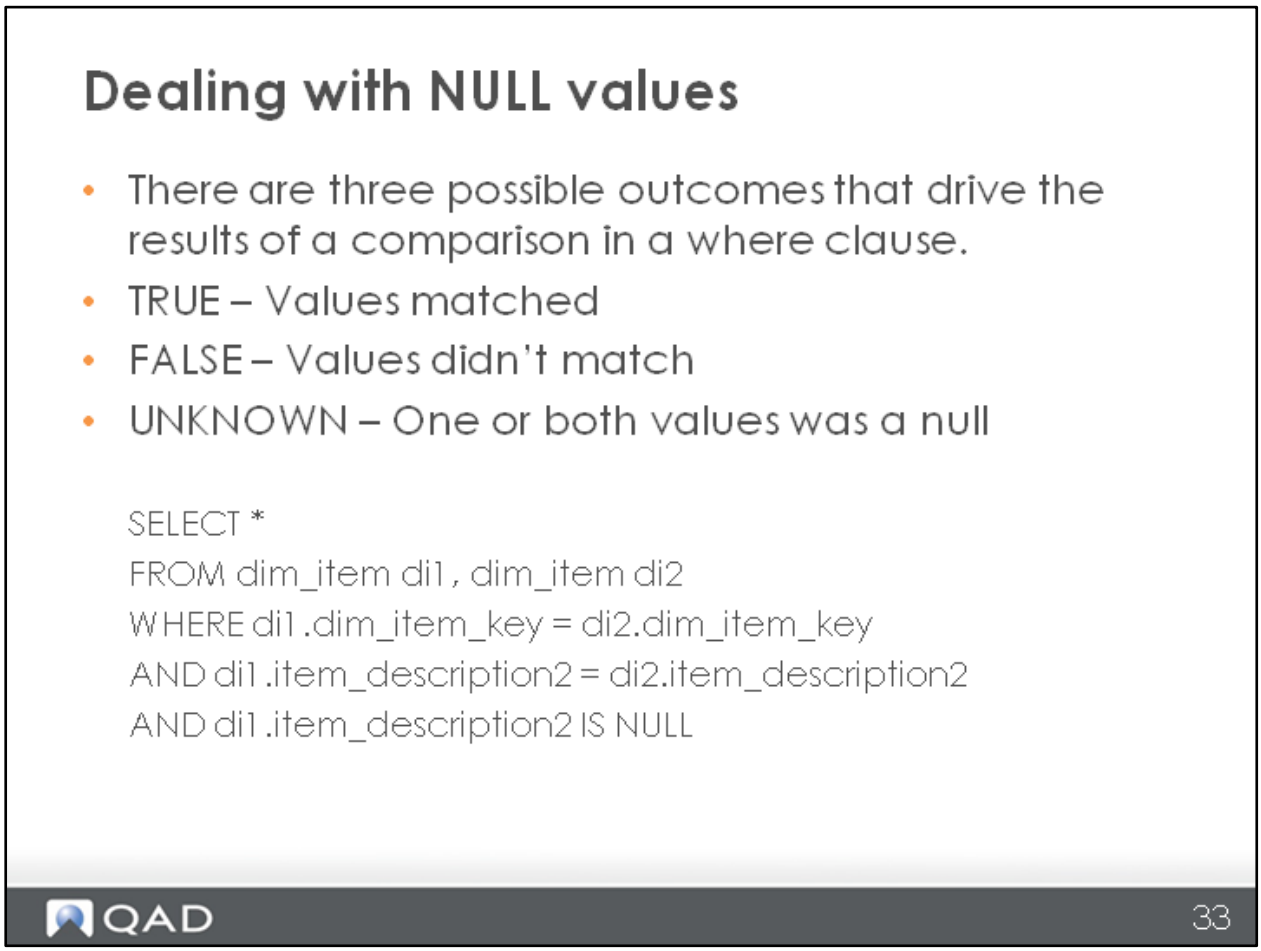
Comparisons in a WHERE clause – cont'd



WHERE – continued

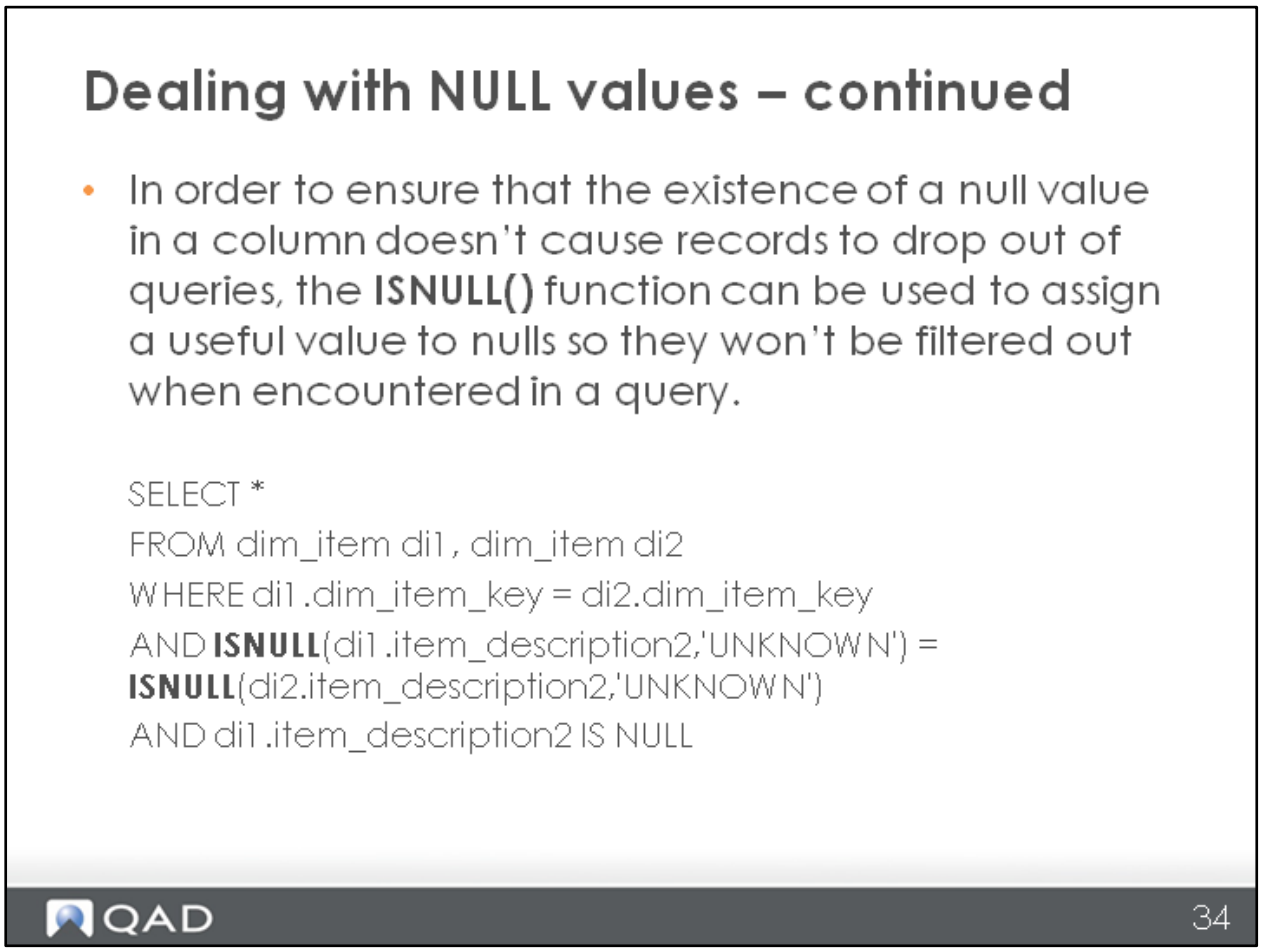


Dealing with NULL values



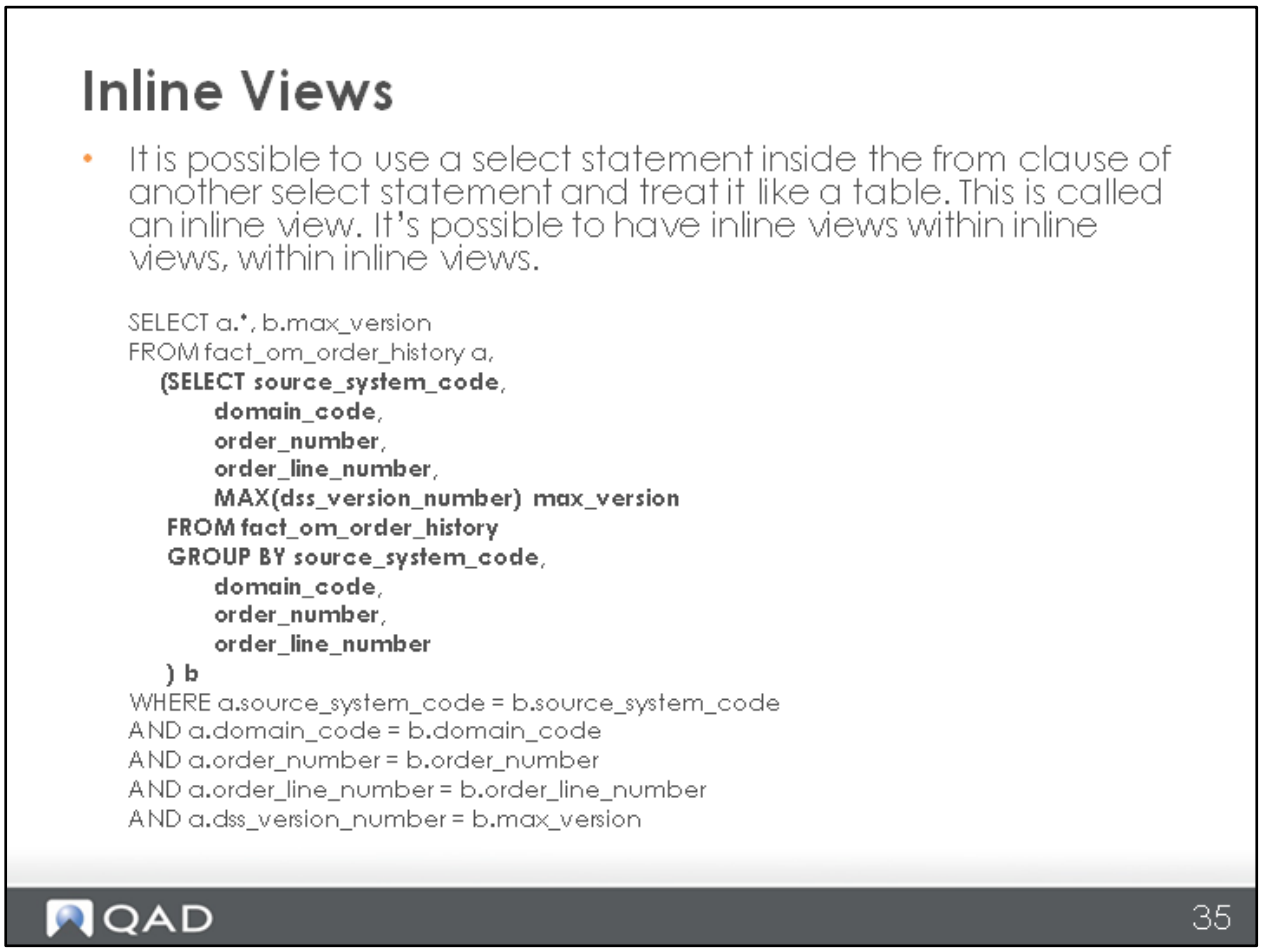
The result can be TRUE, indicating there is a match. The result can be FALSE, indicating there is not a match. The result can also be UNKNOWN, if either of the values being matched on have a null (blank) value. The result is UNKNOWN even if both values are null.

Dealing with NULL values – continued

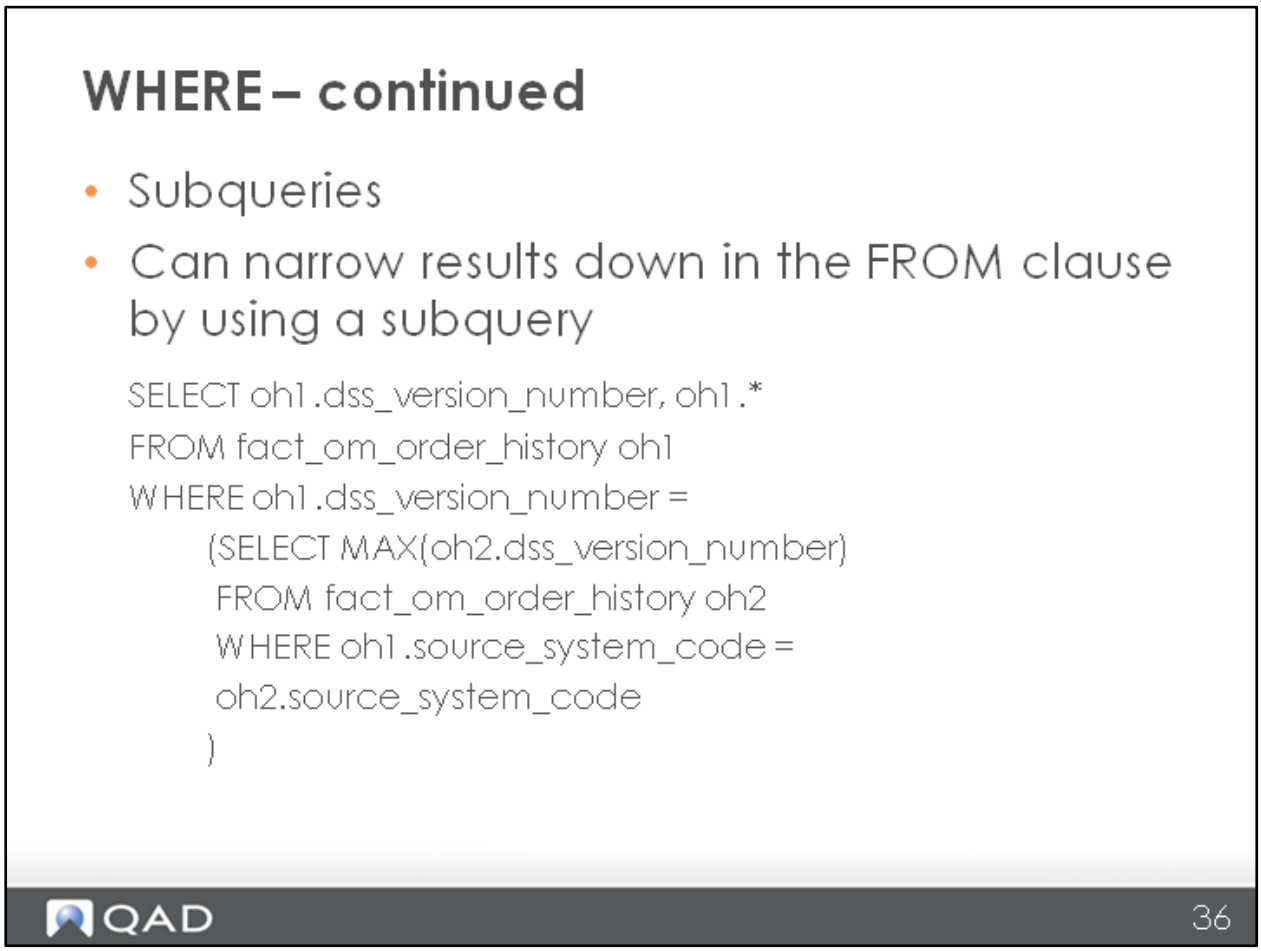


NOTE: NULL <> NULL → cannot use NULL value to compare

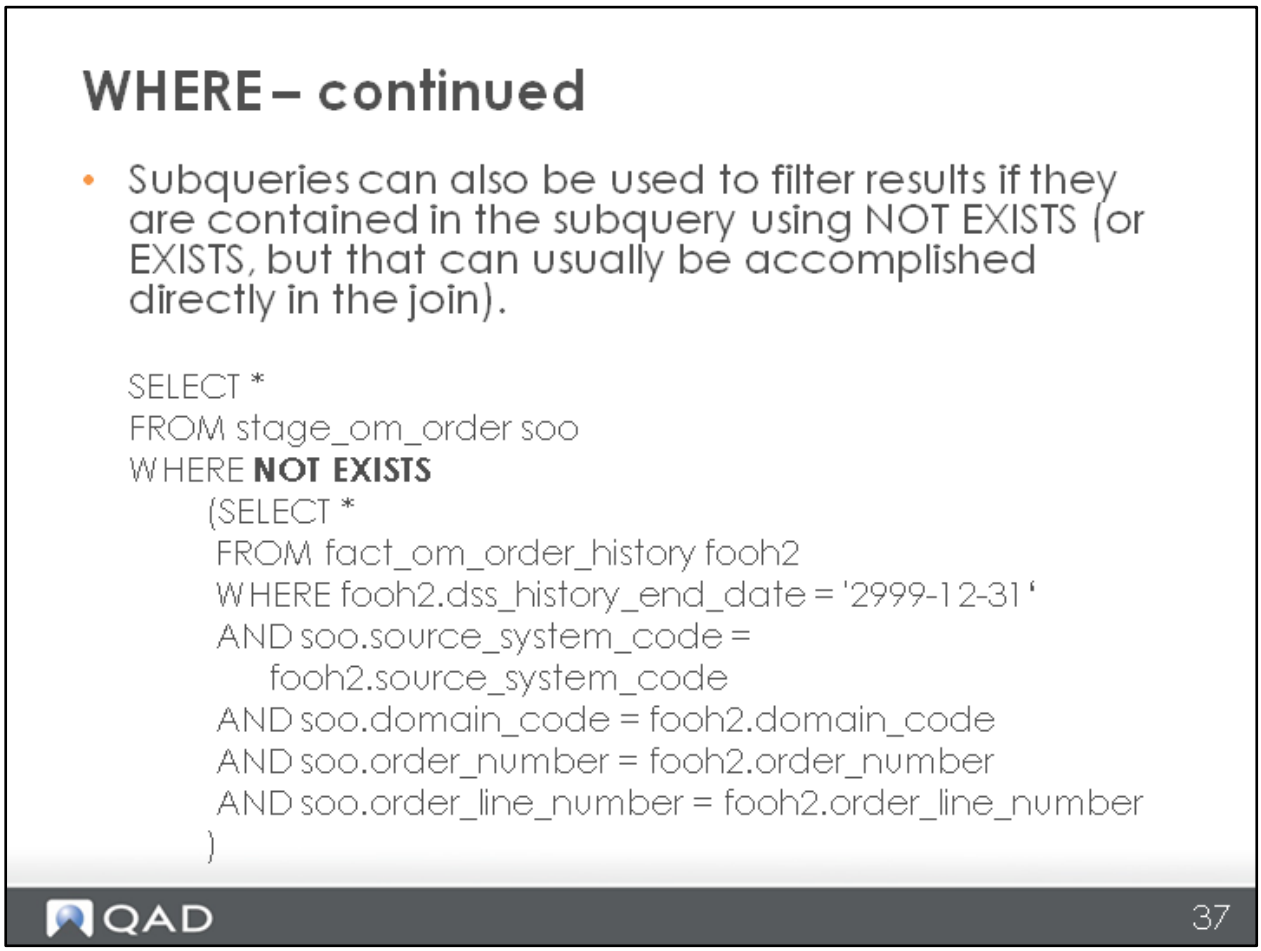
Inline Views



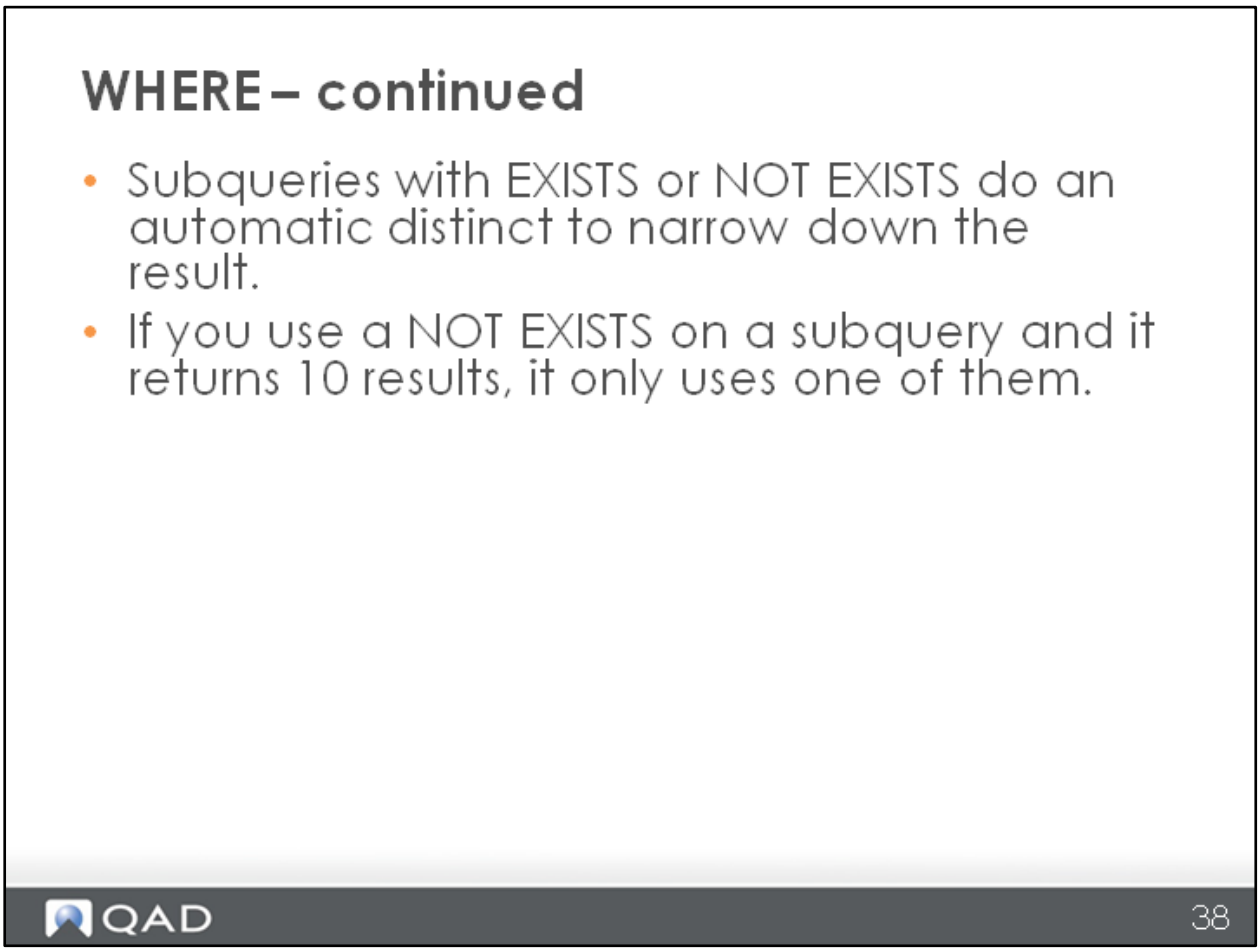
WHERE – continued



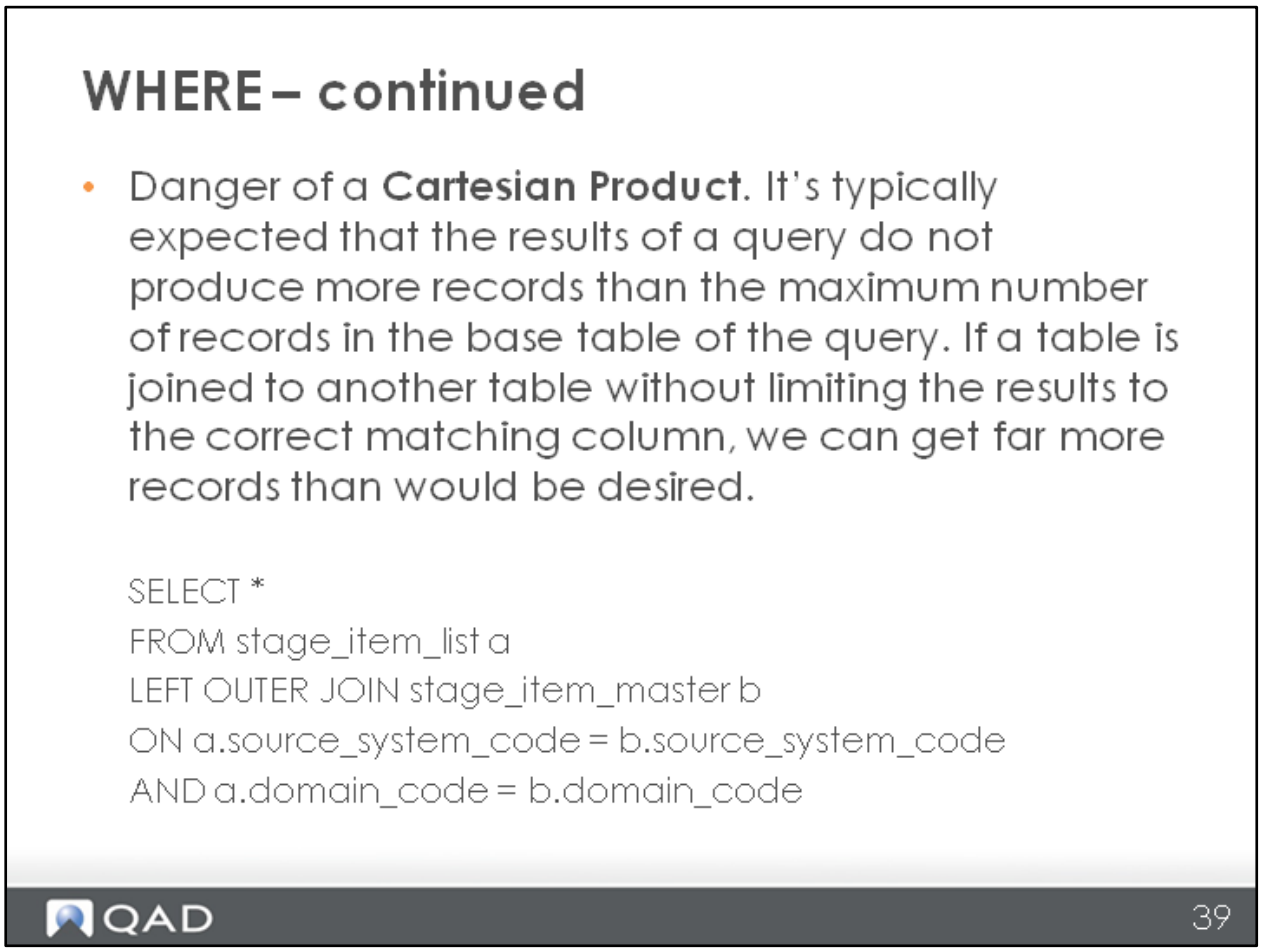
WHERE – continued



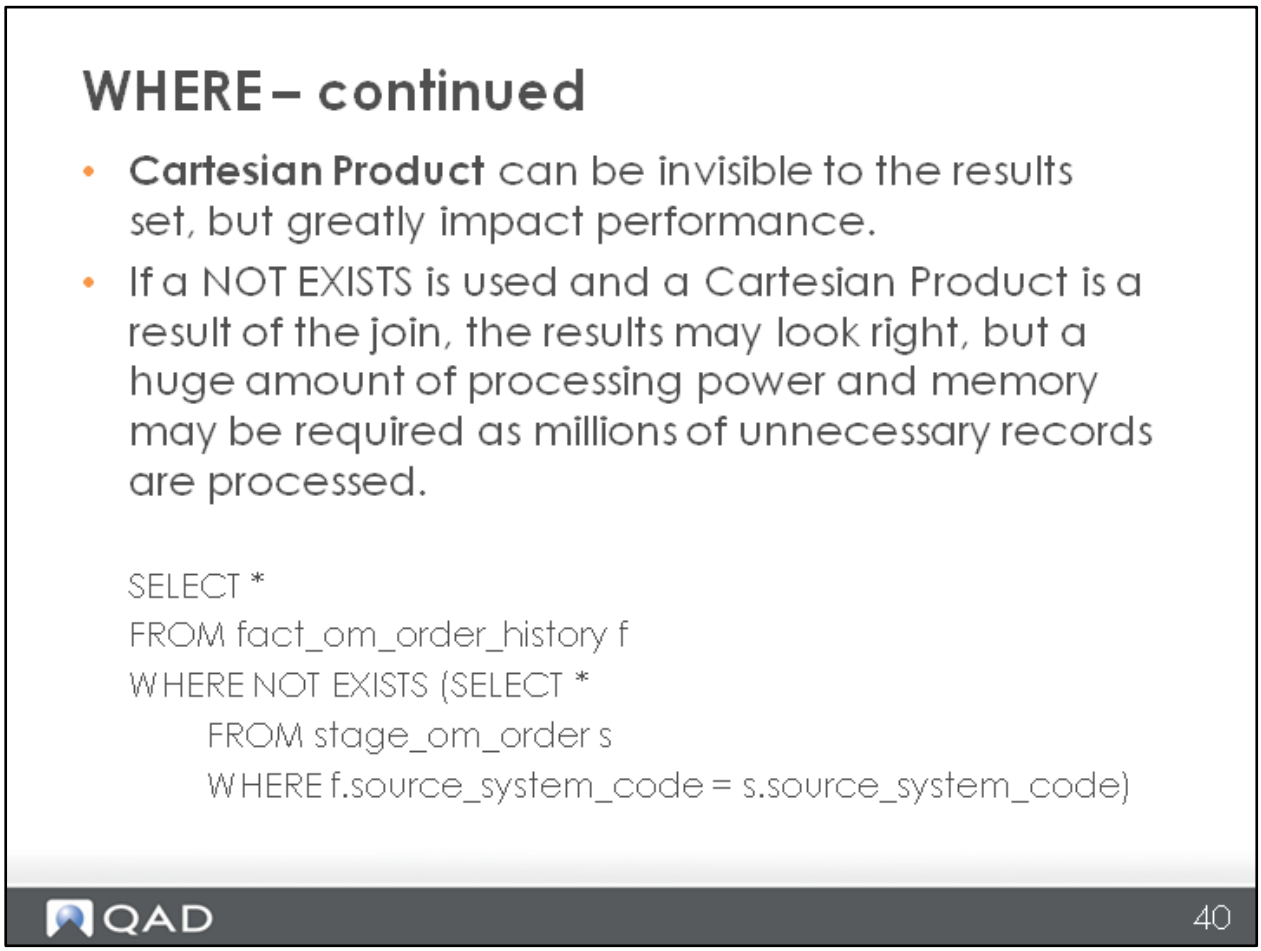
WHERE – continued



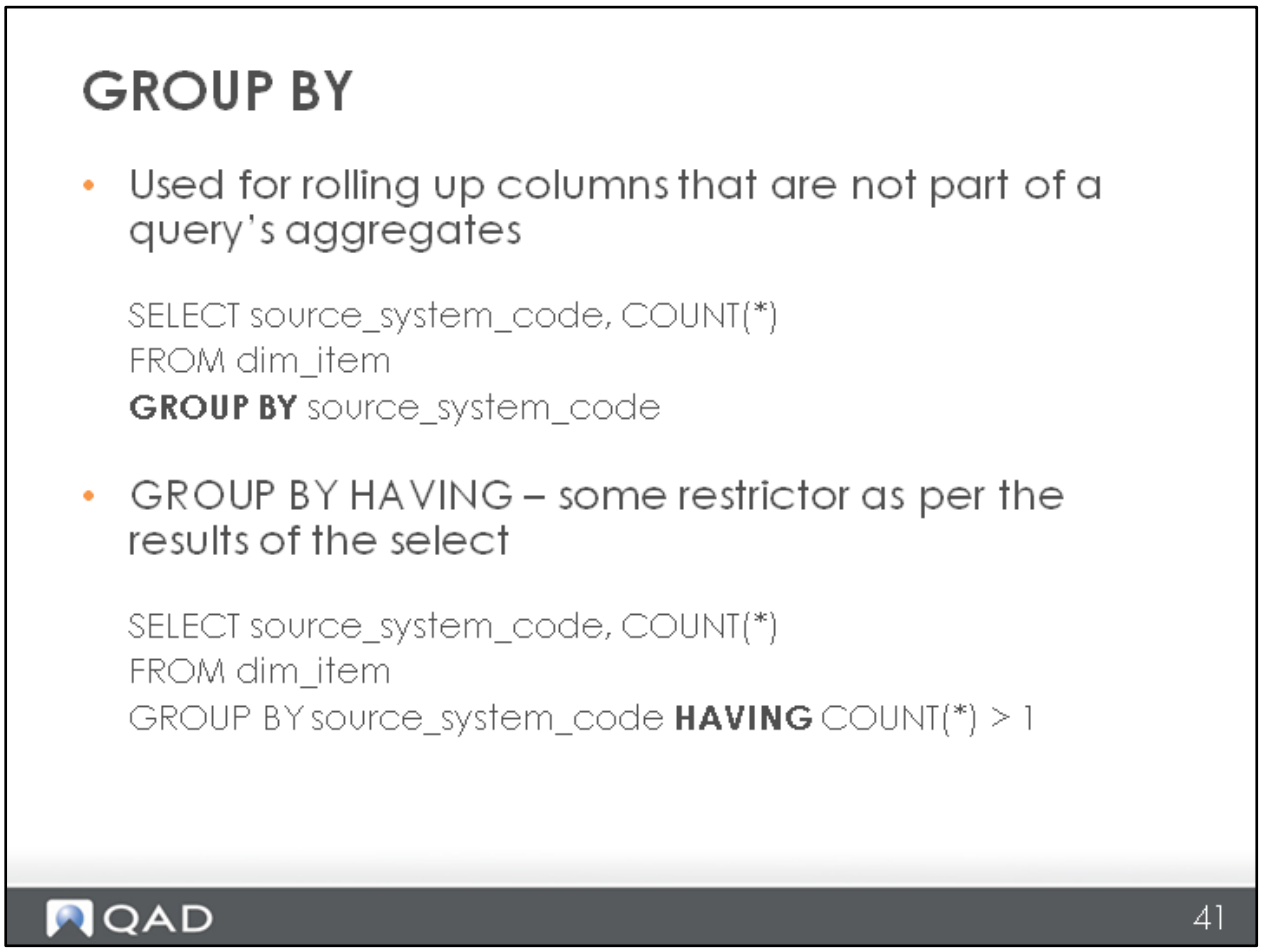
WHERE – continued



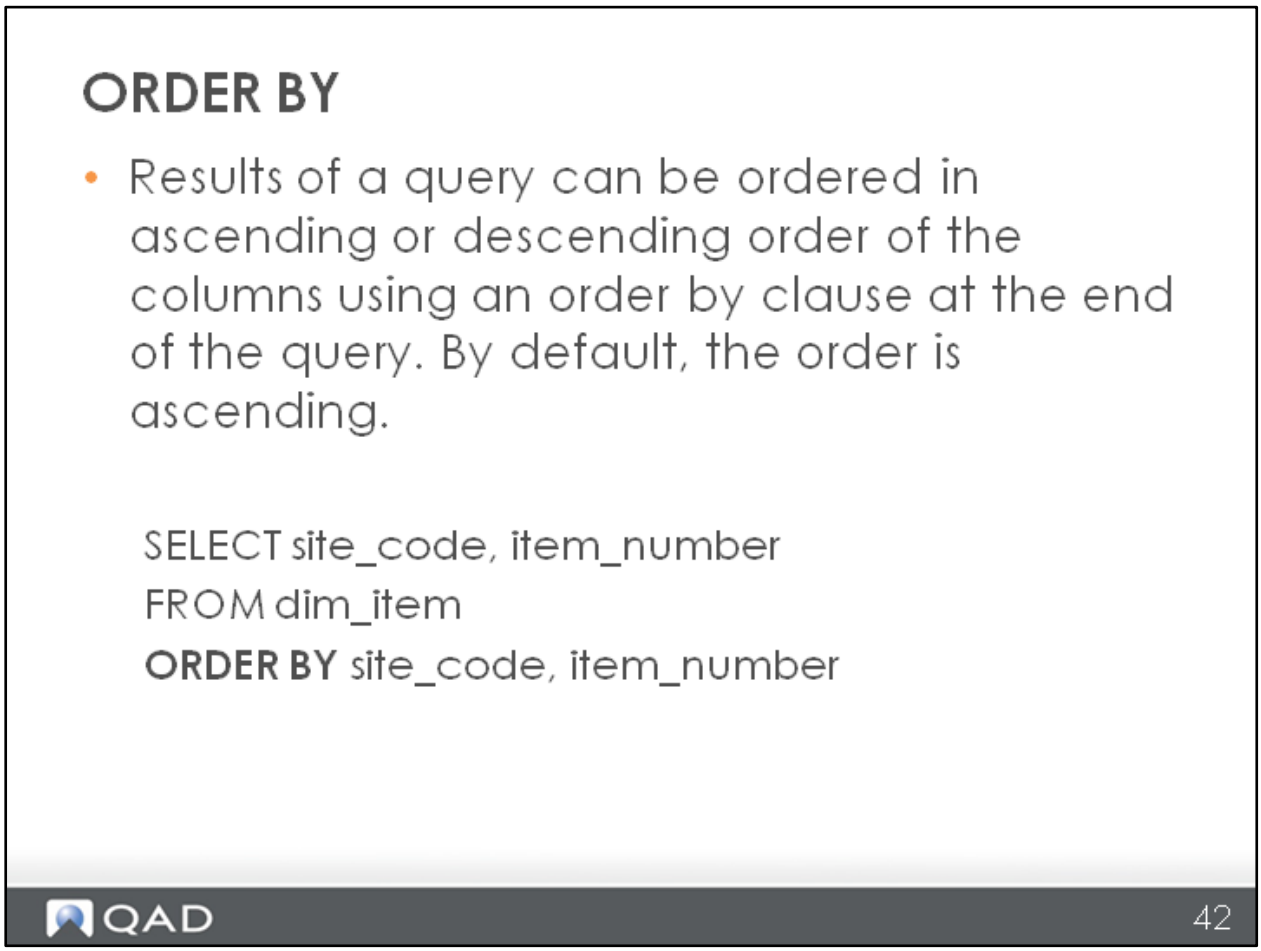
WHERE – continued



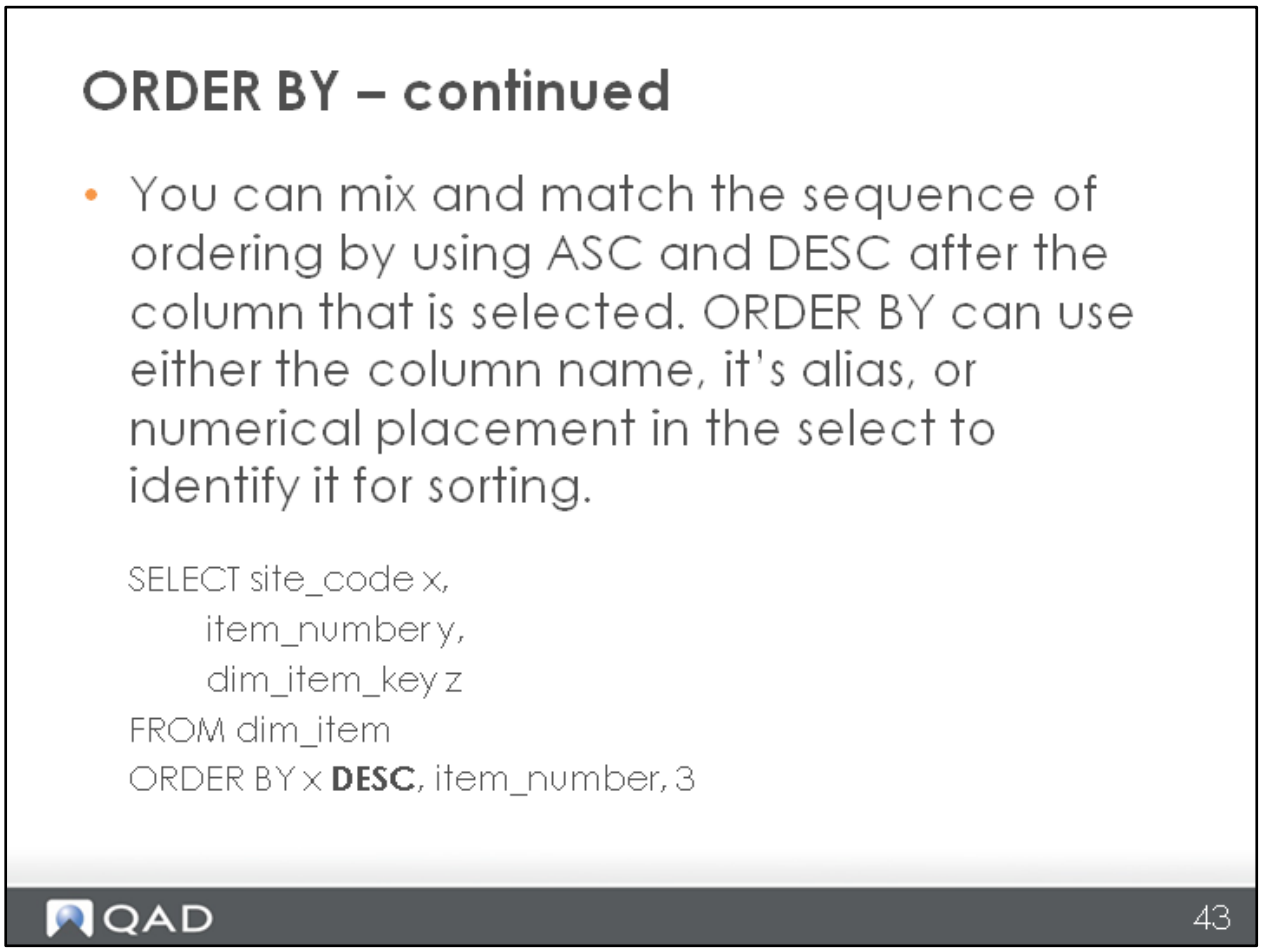
GROUP BY


GROUP BY
Used for rolling up columns that are not part of a query's aggregates
SELECT source_system_code, COUNT(*)
FROM dim_item
GROUP BY source_system_code
GROUP BY HAVING – some restrictor as per the results of the select
SELECT source_system_code, COUNT(*)
FROM dim_item
GROUP BY source_system_code HAVING COUNT(*) > 1
QAD
41

ORDER BY

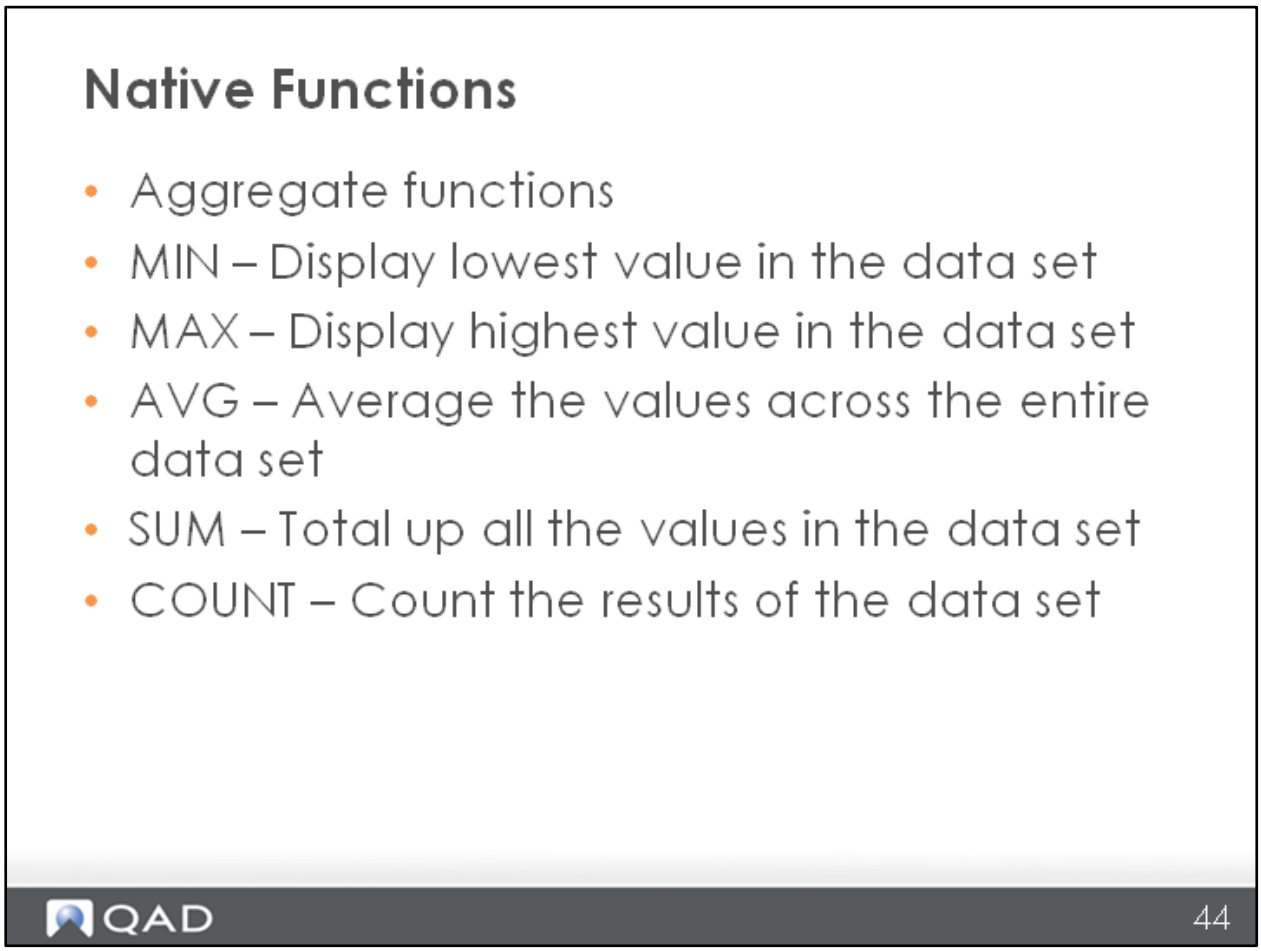

ORDER BY
• Results of a query can be ordered in ascending or descending order of the columns using an order by clause at the end of the query. By default, the order is ascending.
SELECT site_code, item_number
FROM dim_item
ORDER BY site_code, item_number
QAD
42

ORDER BY – continued

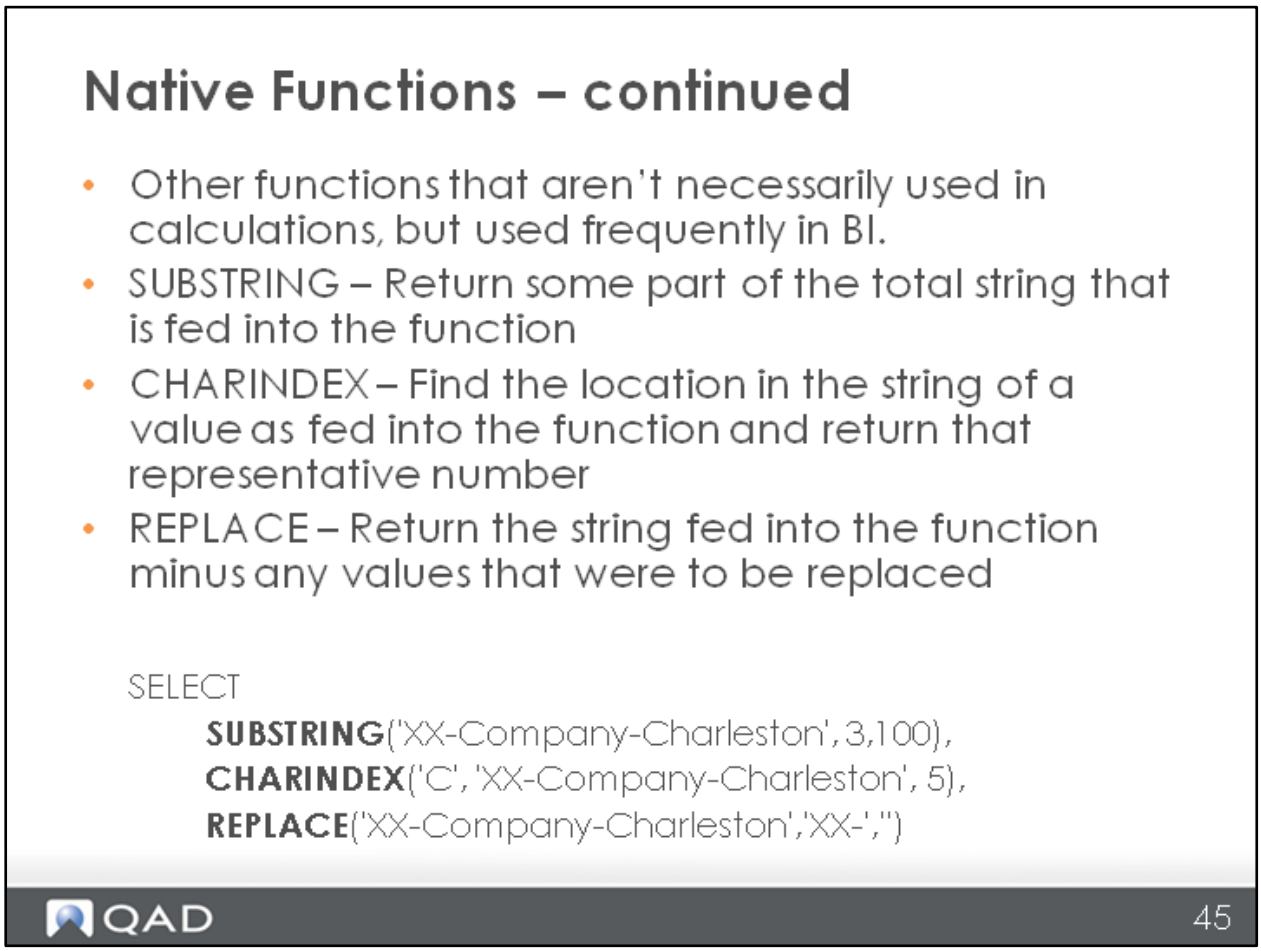


See how in the results, we were able to order by the alias, the column name and the column number.

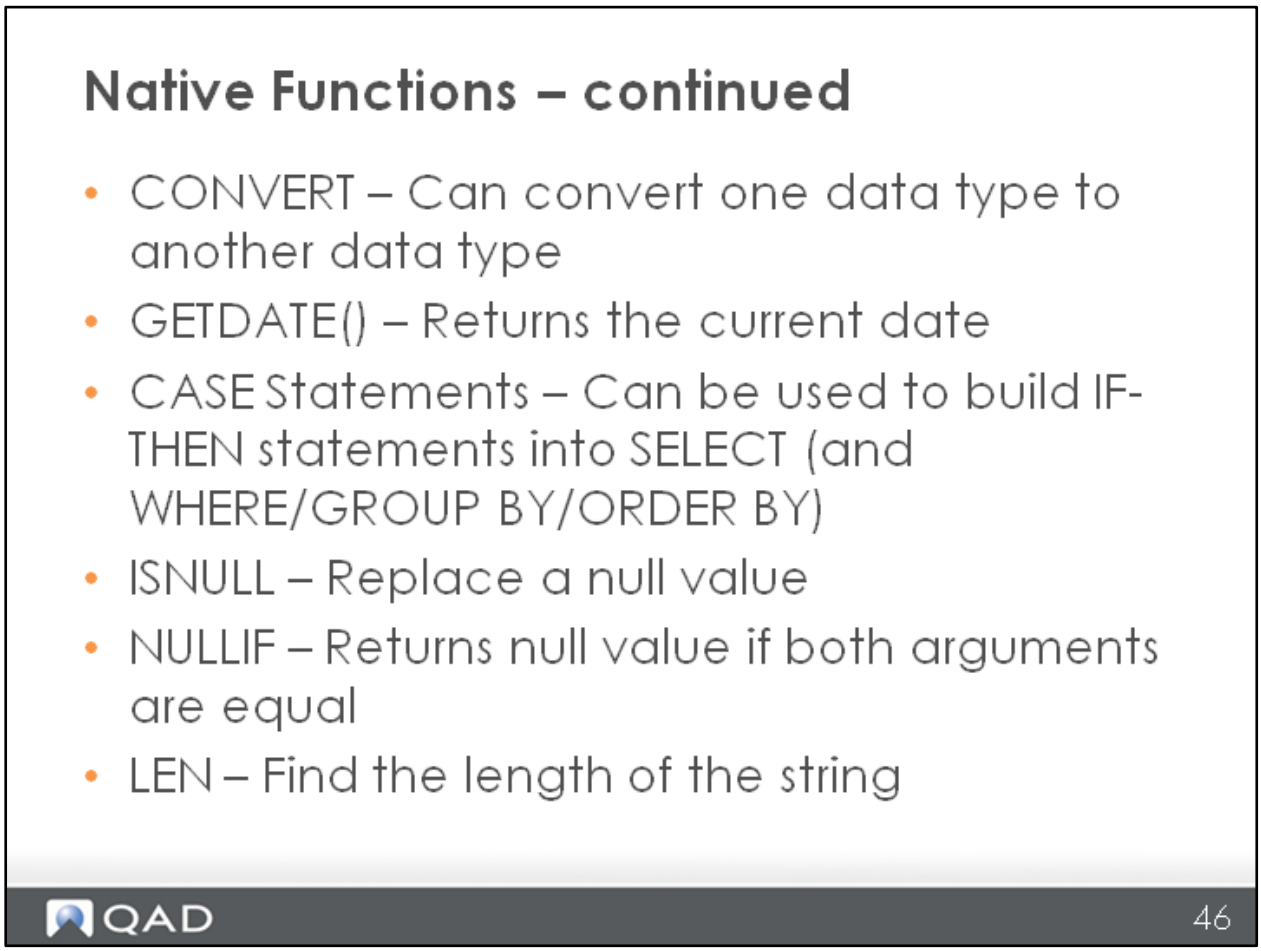
Native Functions



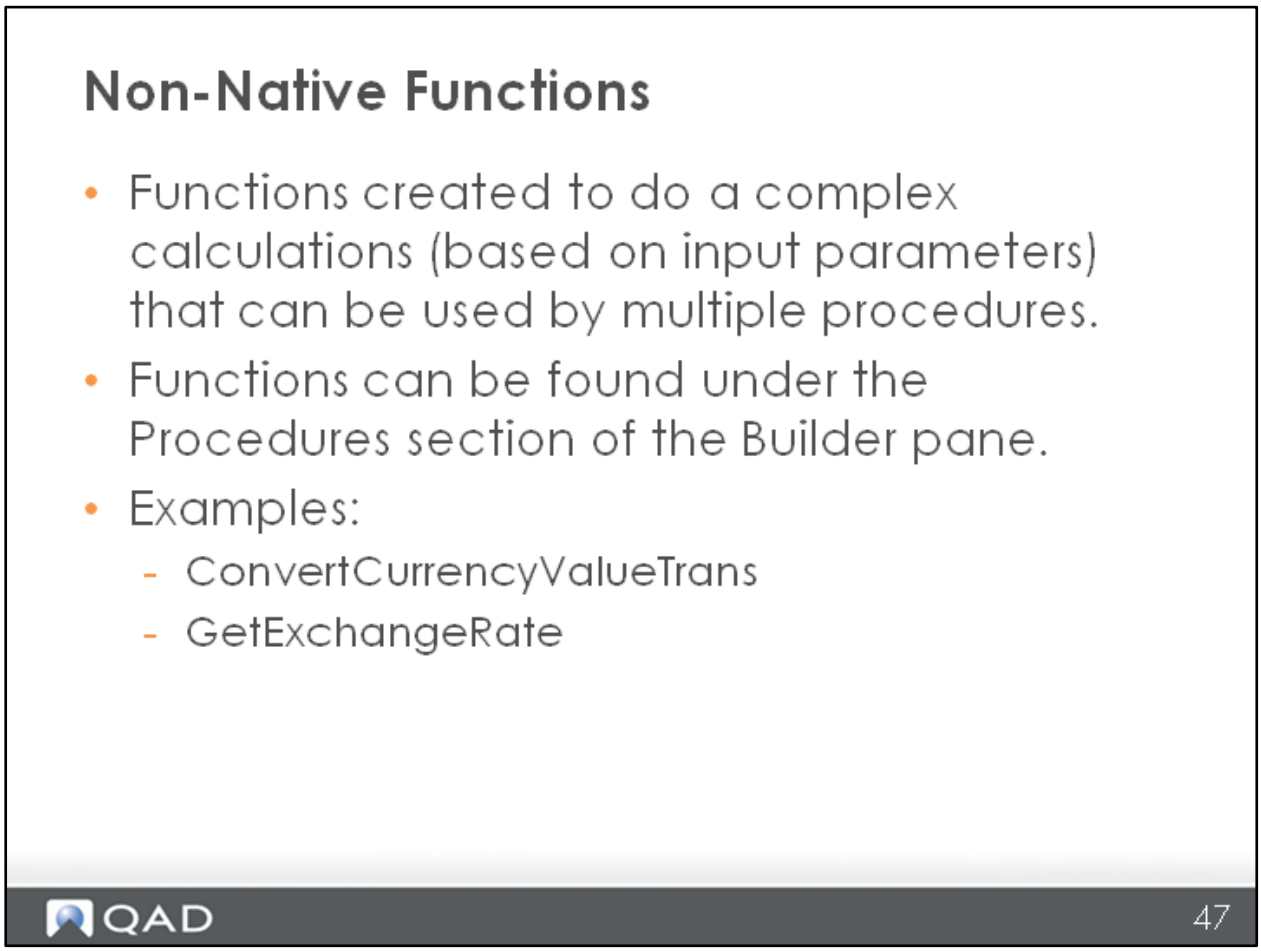
Native Functions – continued



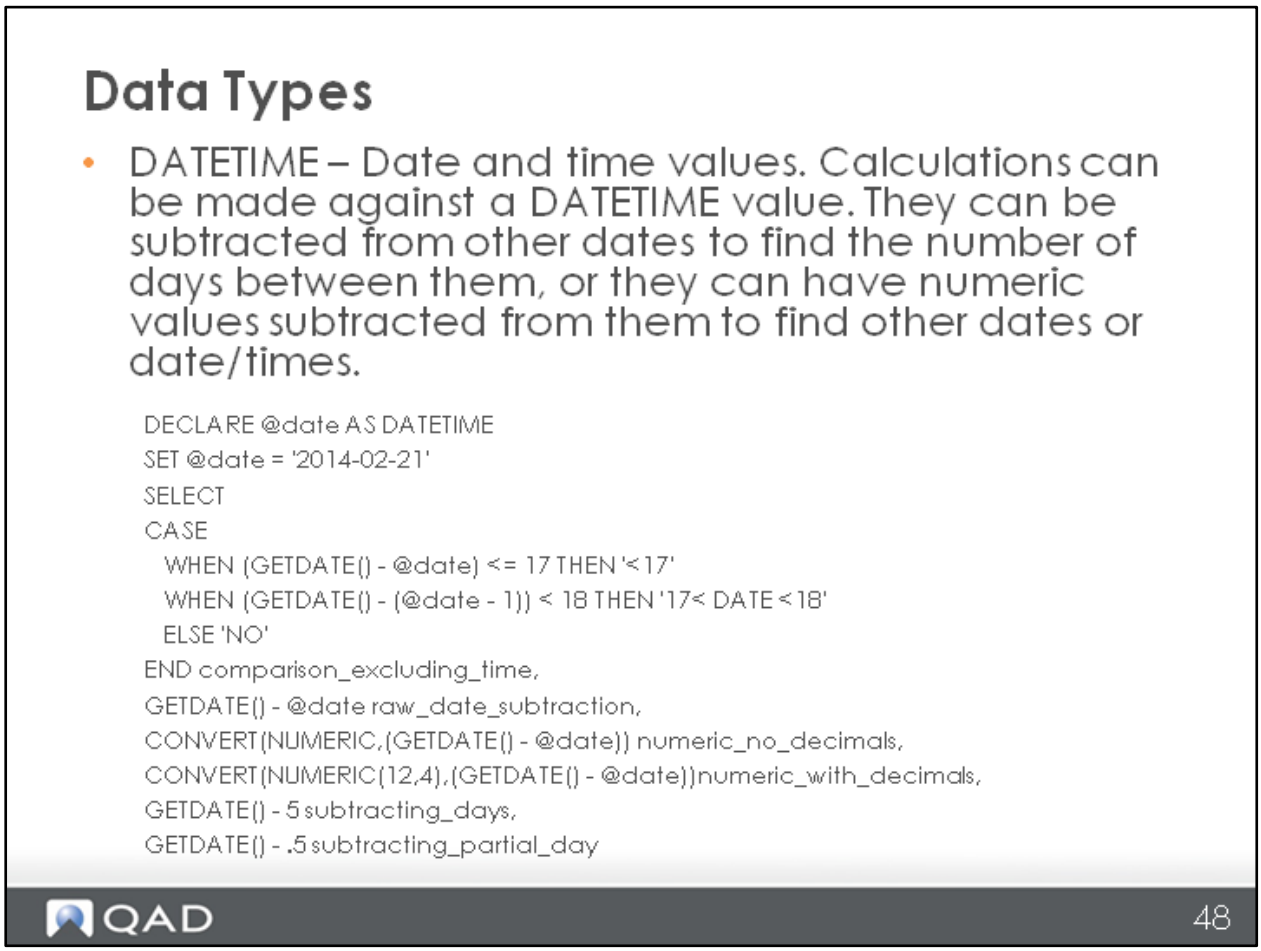
Native Functions – continued



Non-Native Functions

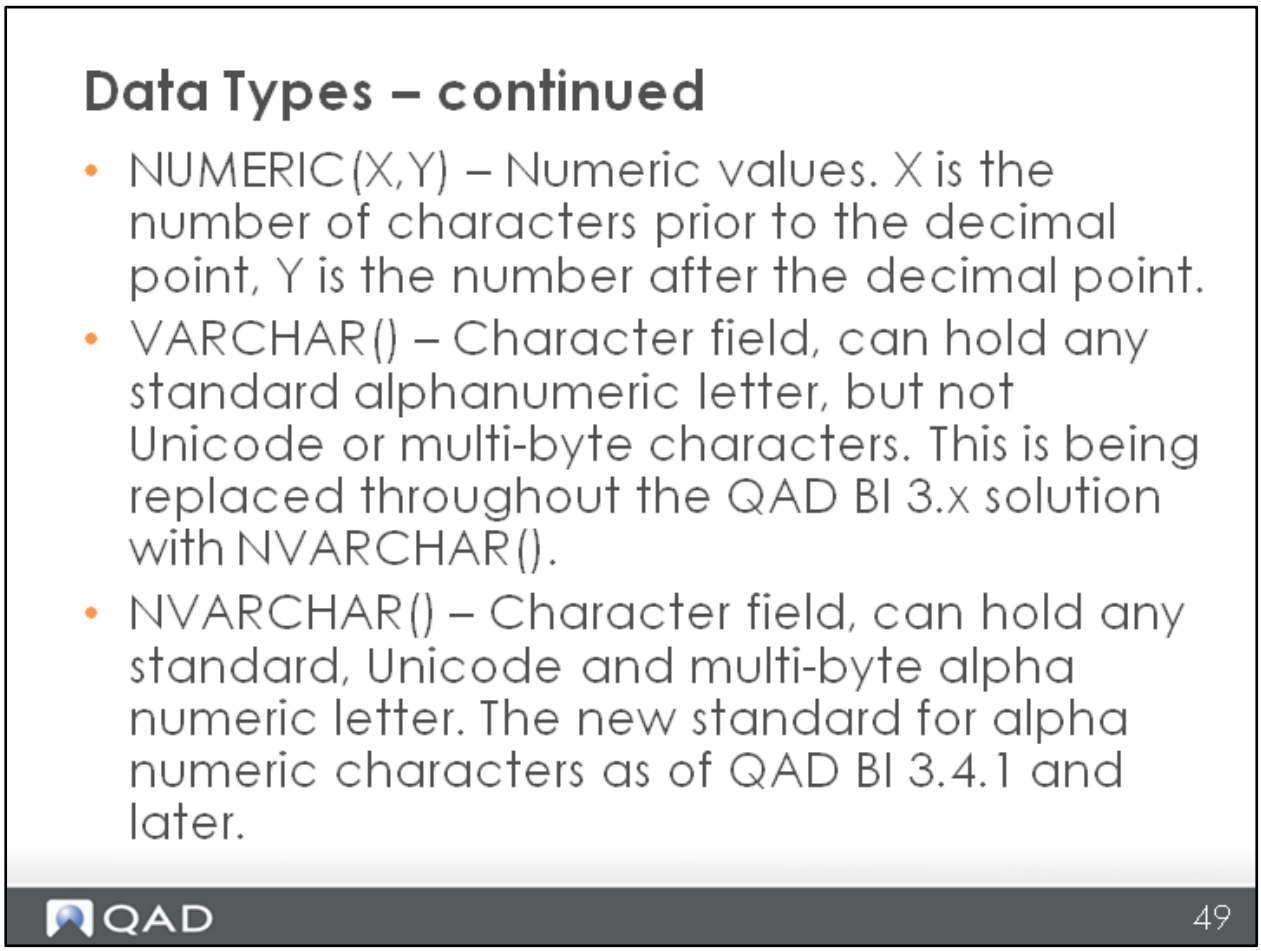


Data Types

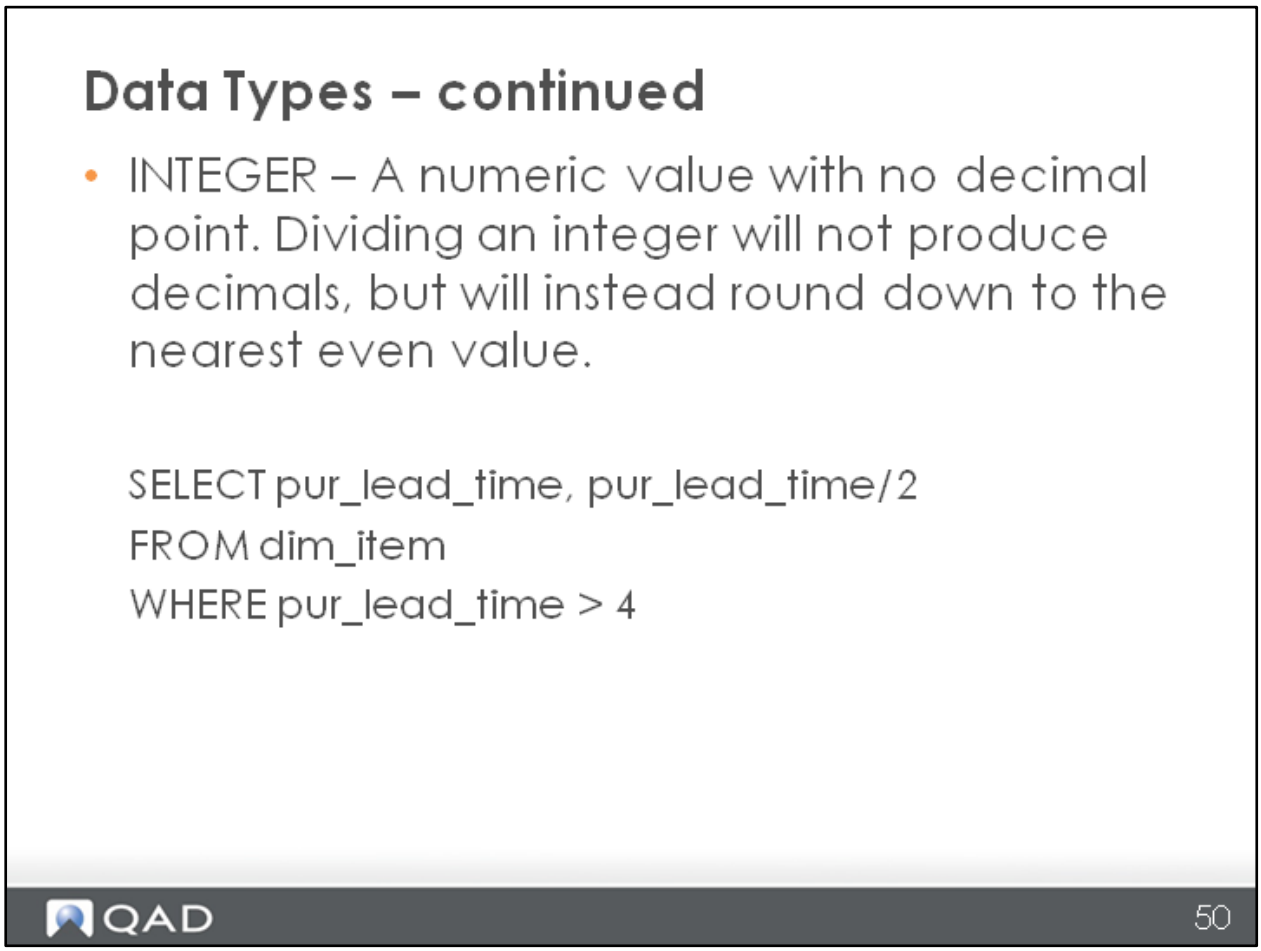


Local variables are prefixed by @ symbol.

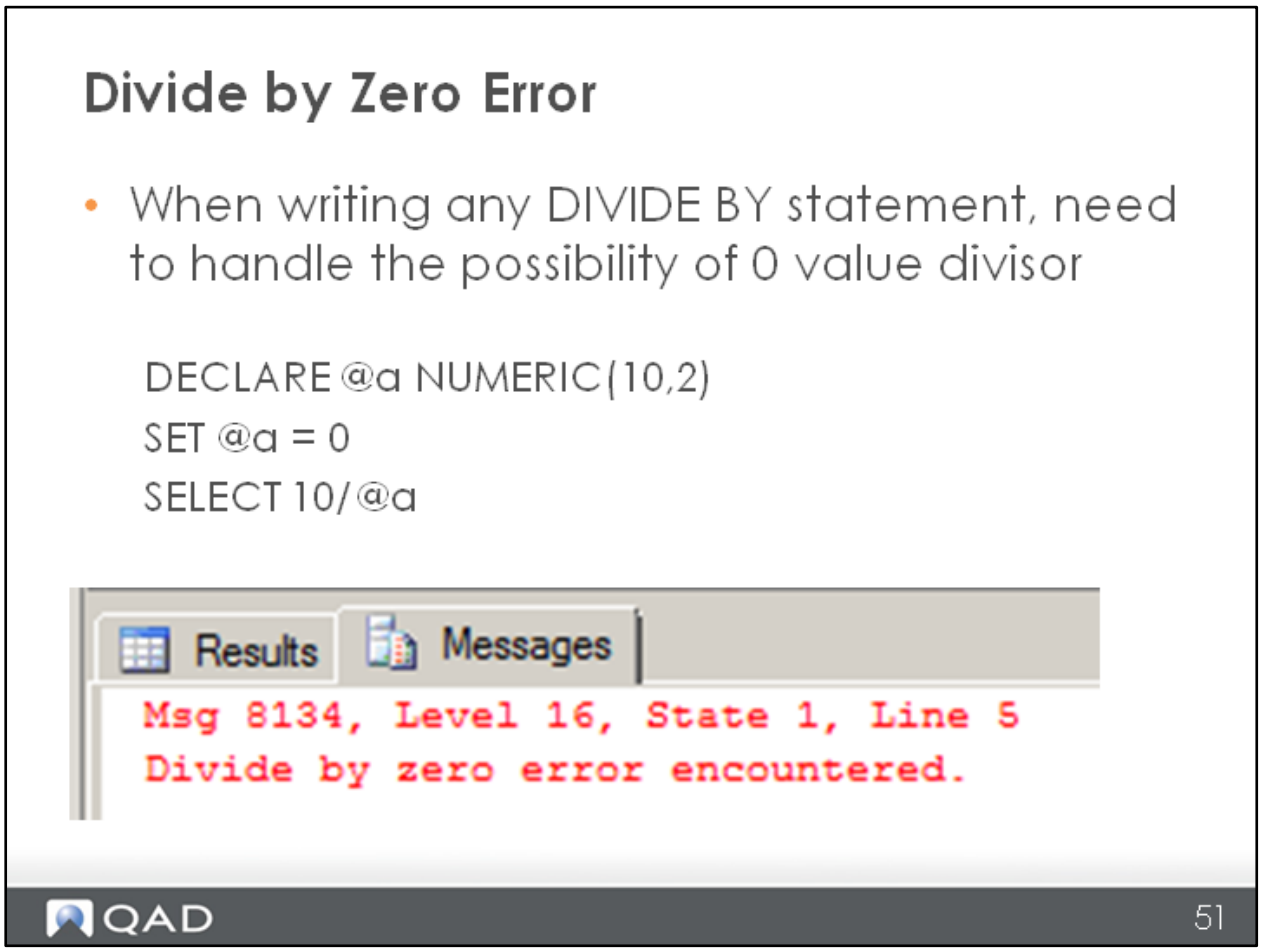
Data Types – continued



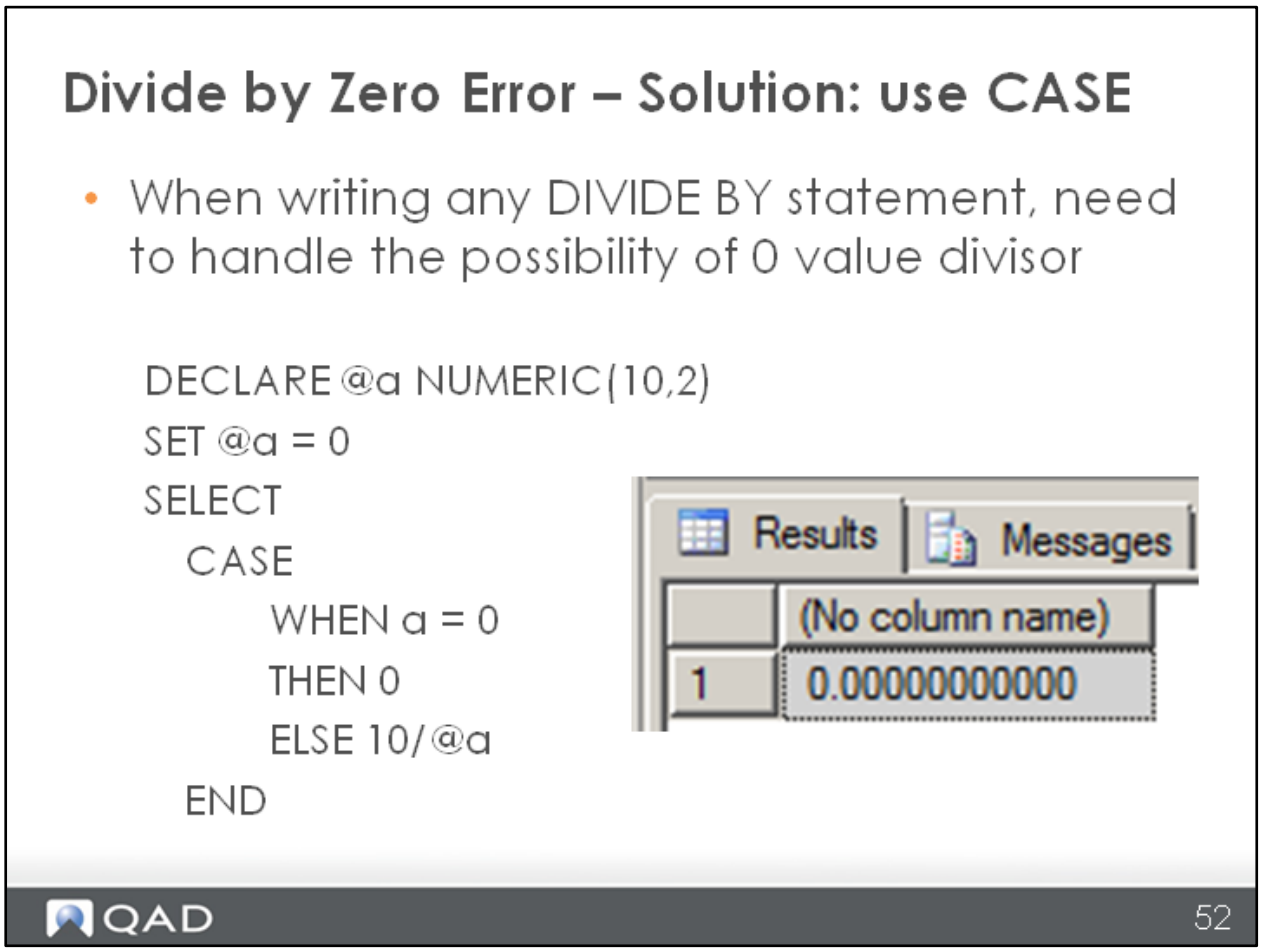
Data Types – continued



Divide by Zero Error

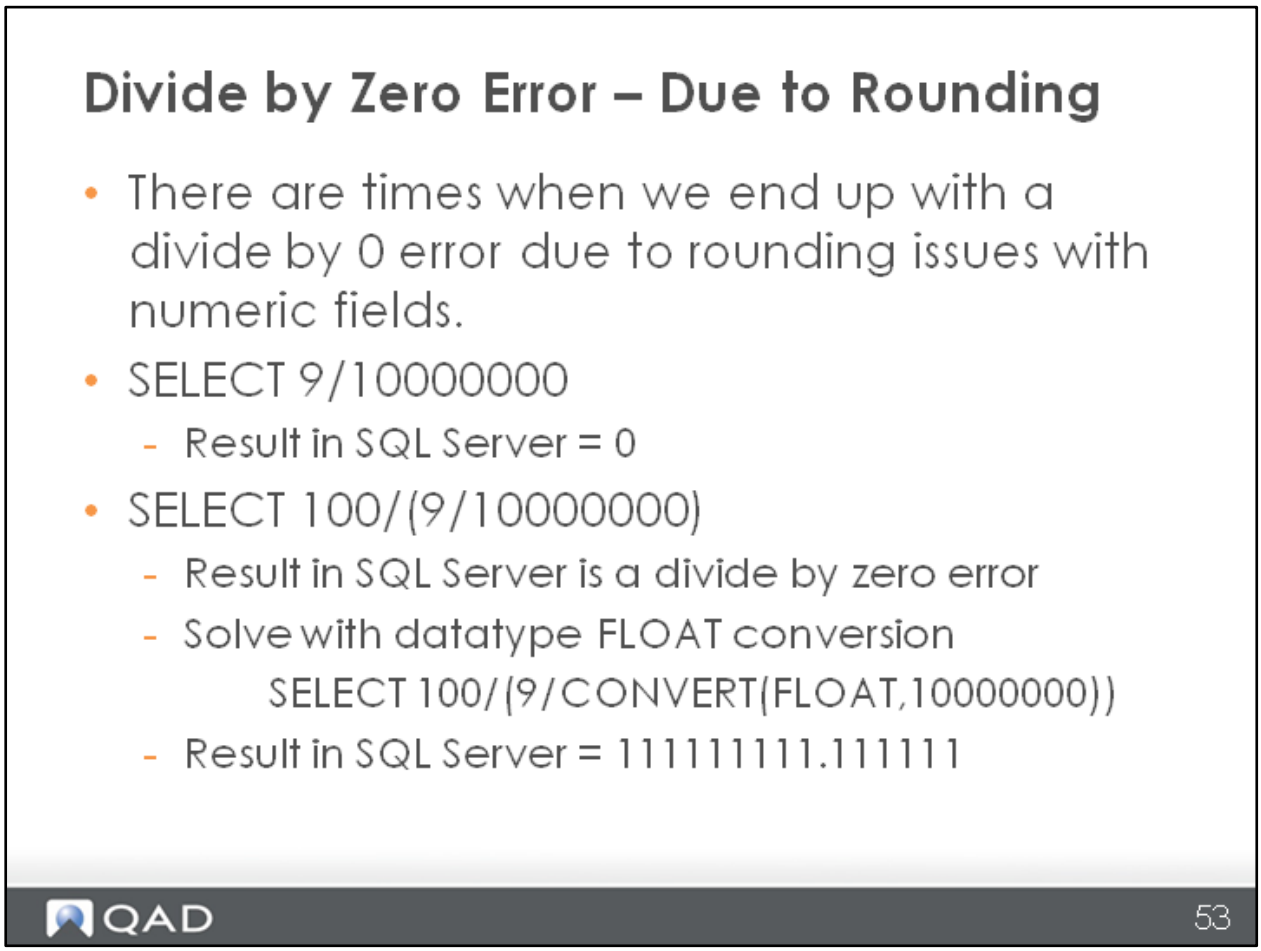


Divide by Zero Error – Solution: use CASE



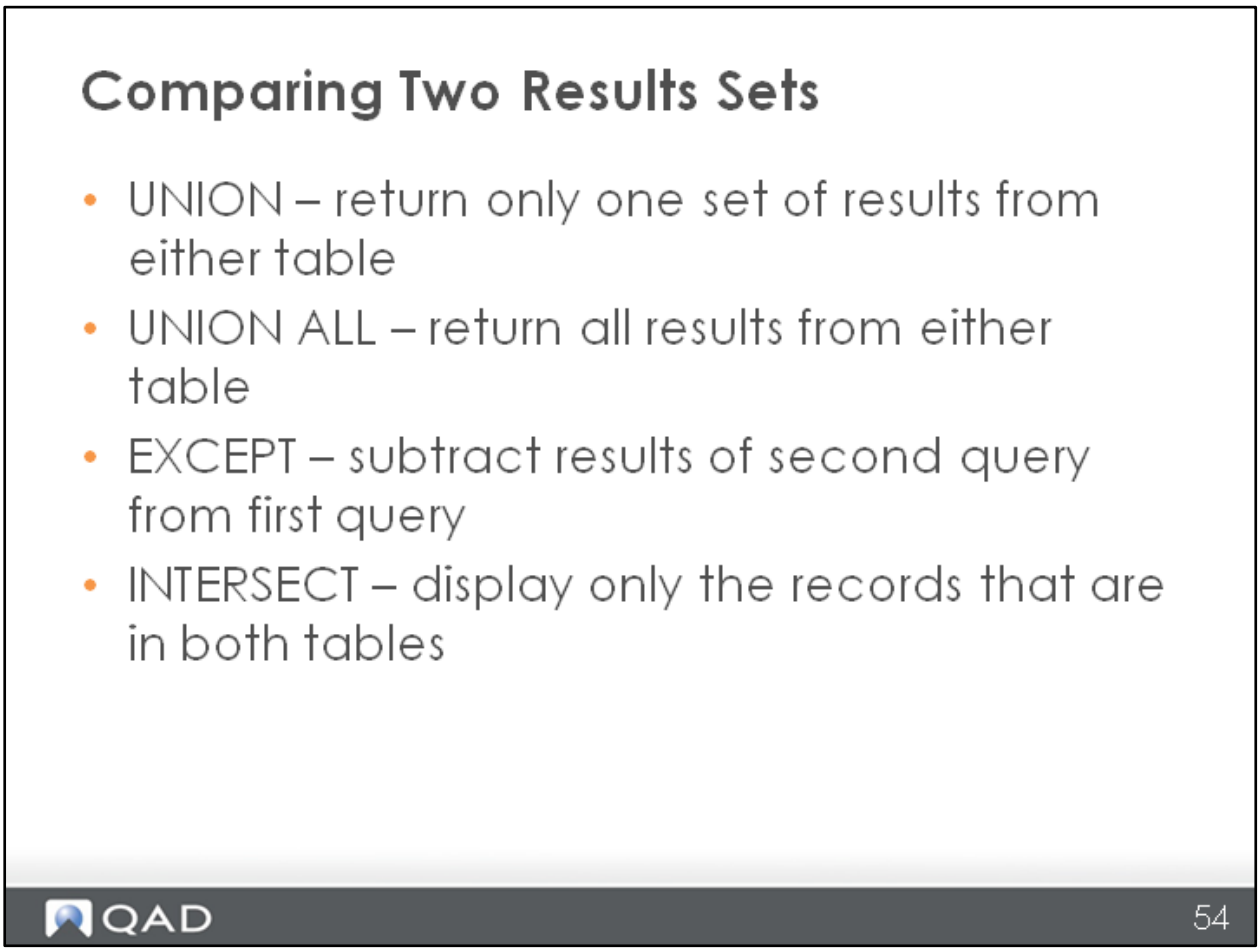
Any coding being done that involves division should note this and the possibility that the divisor may end up being 0 anyway. Usually we use case statements so that if the divisor does = 0, then the results are set to 0.

Divide by Zero Error – Due to Rounding

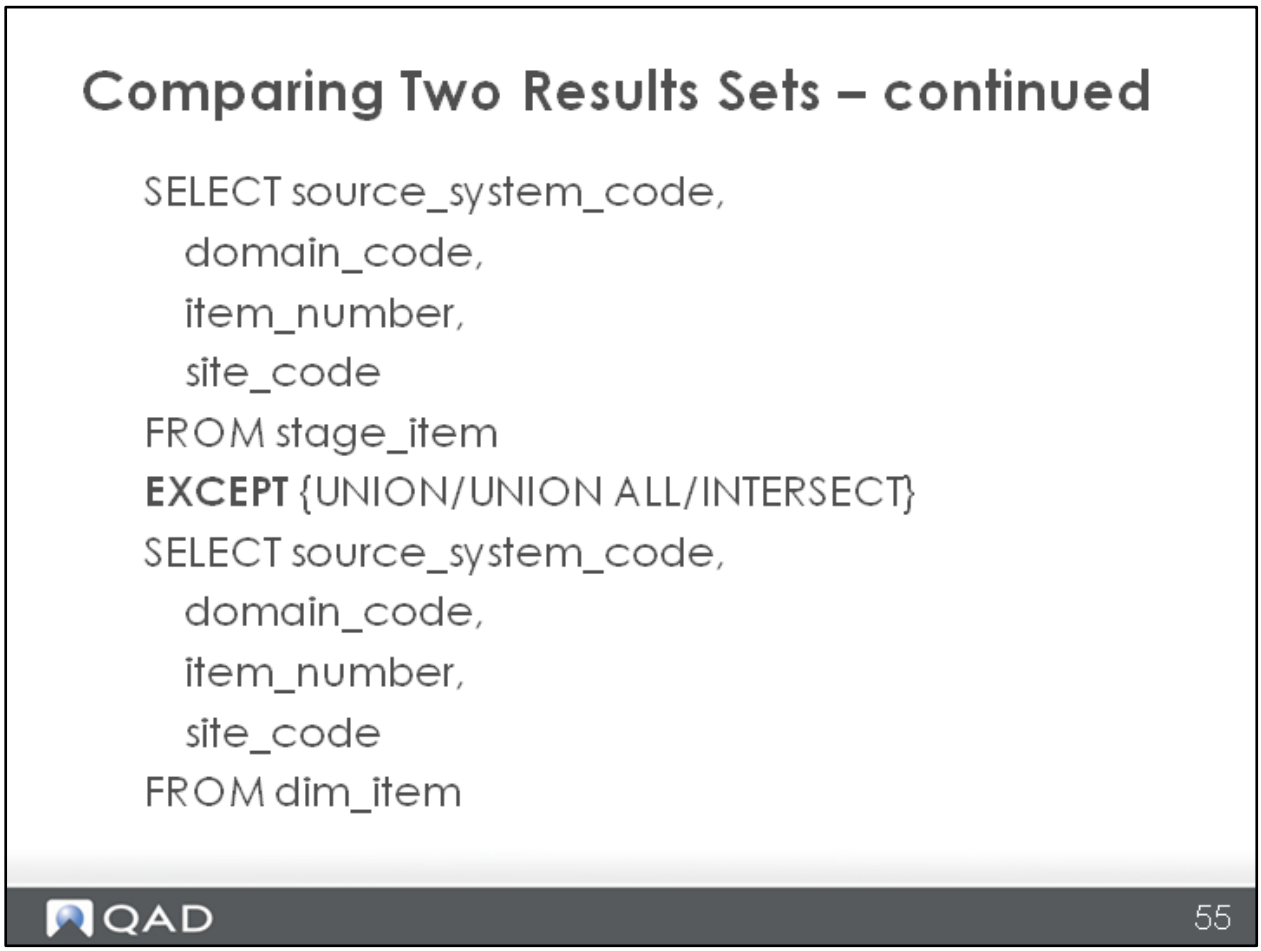


This query could still use a CASE statement.

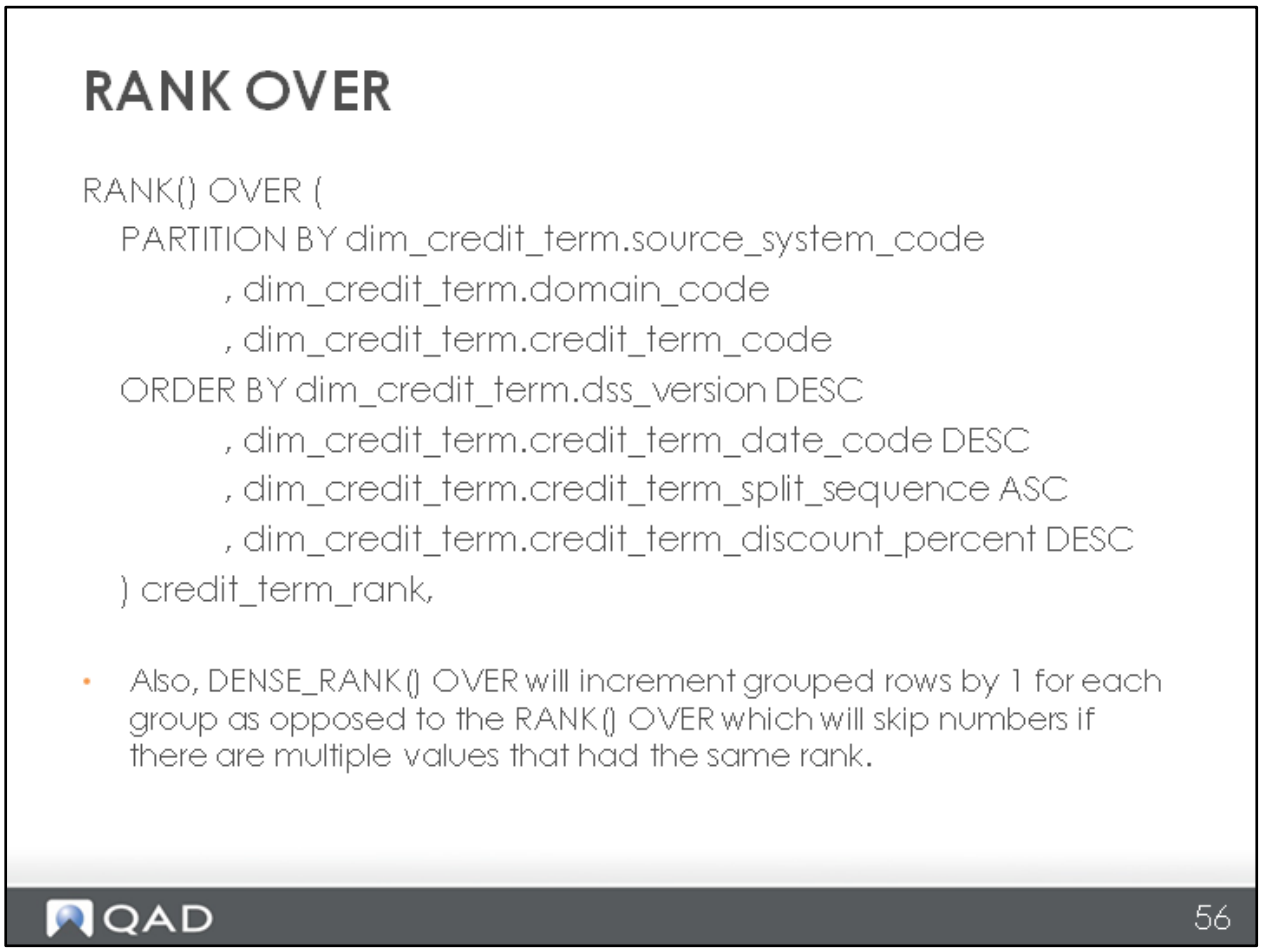
Comparing Two Results Sets



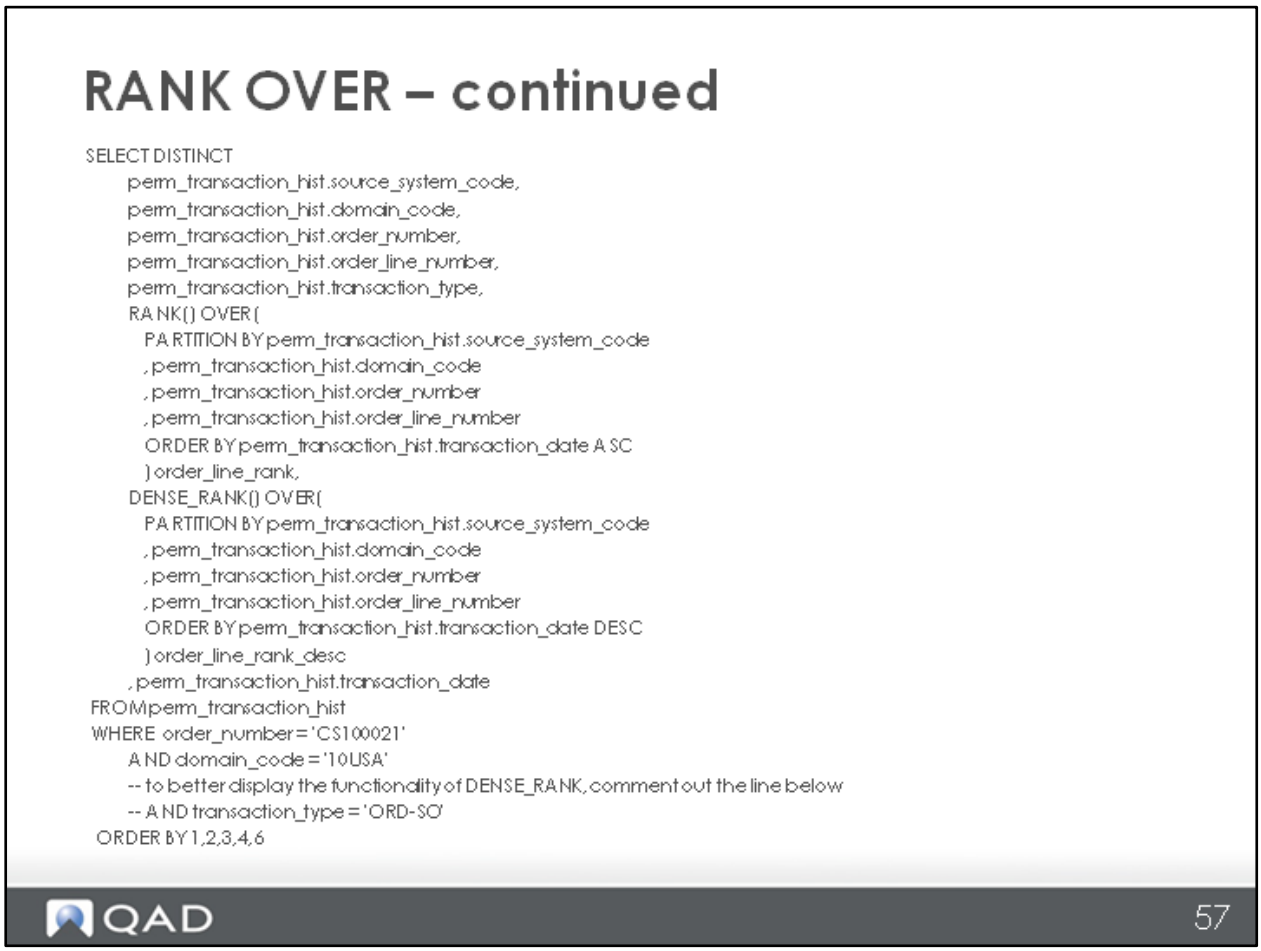
Comparing Two Results Sets – continued



RANK OVER

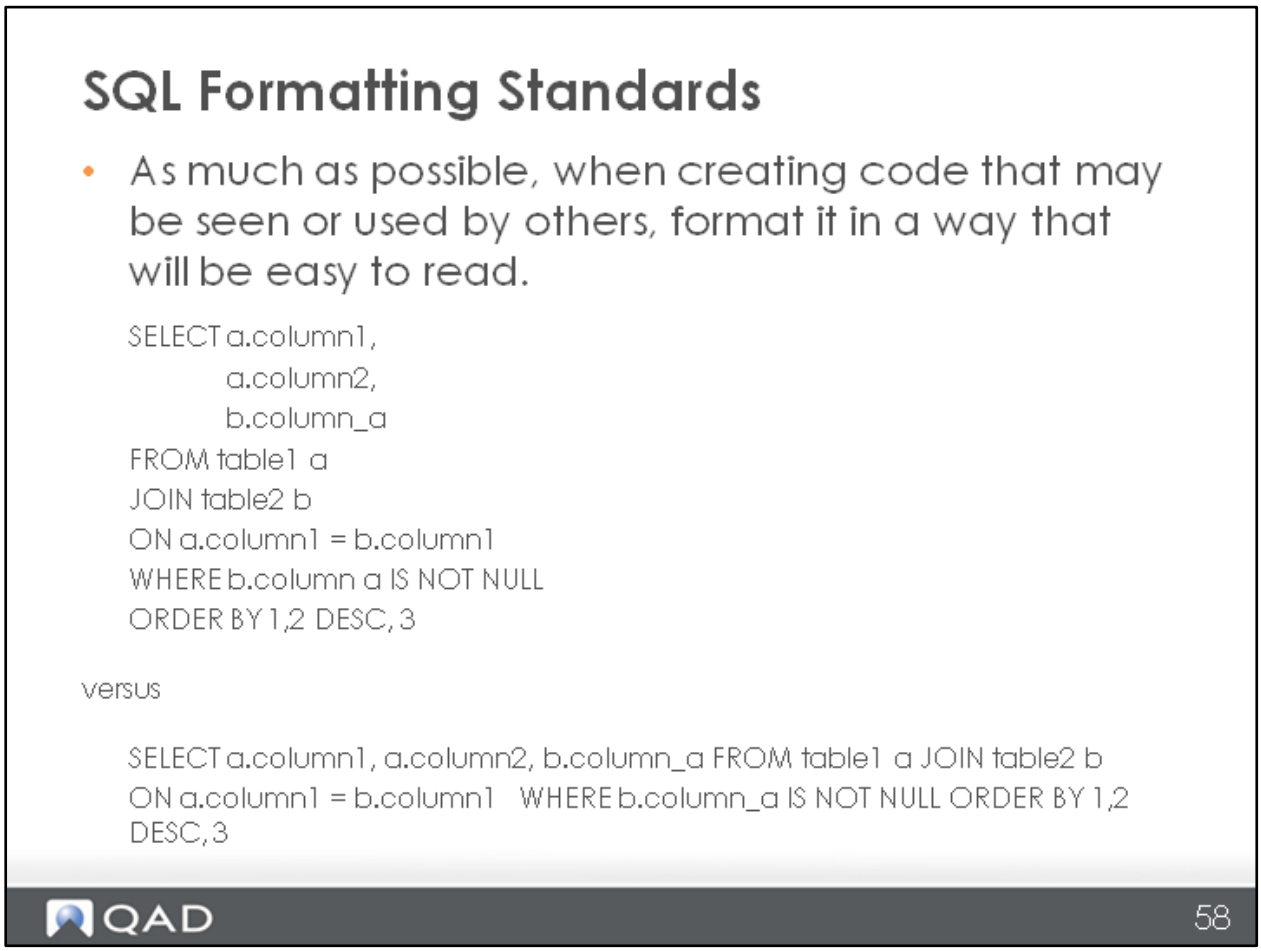

RANK OVER
RANK() OVER (
PARTITION BY dim_credit_term.source_system_code
, dim_credit_term.domain_code
, dim_credit_term.credit_term_code
ORDER BY dim_credit_term.dss_version DESC
, dim_credit_term.credit_term_date_code DESC
, dim_credit_term.credit_term_split_sequence ASC
, dim_credit_term.credit_term_discount_percent DESC
) credit_term_rank,
Also, DENSE_RANK() OVER will increment grouped rows by 1 for each group as opposed to the RANK() OVER which will skip numbers if there are multiple values that had the same rank.
QAD
56

RANK OVER – continued



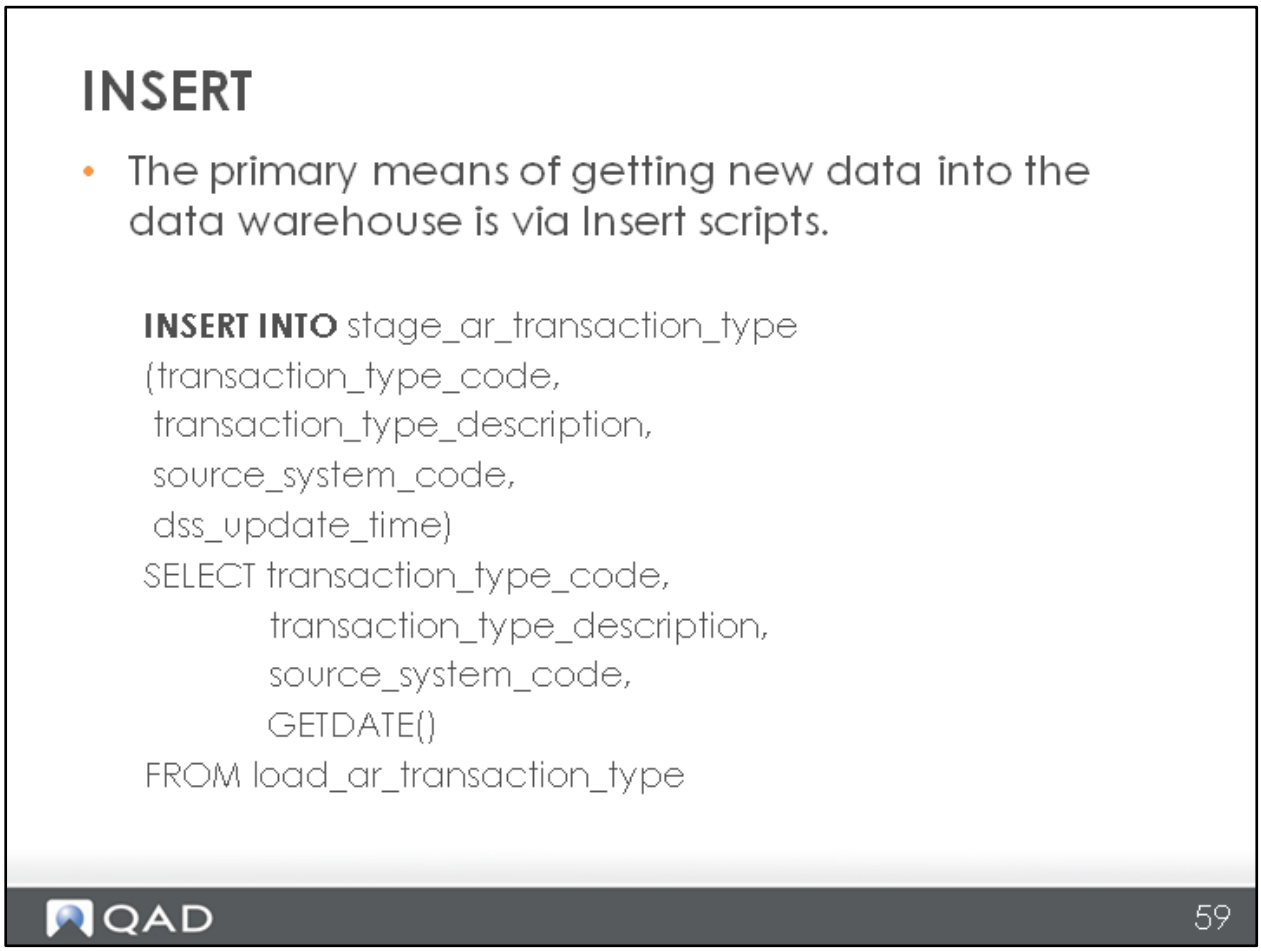
In this example, to note the effects of ranking duplicate records, using rank over and dense rank over, run the query with the second to last line uncommented, then with it commented out. Also try running the query with the second to last line commented out and the 6th line, perm_transaction_hist.transaction_type commented out. Note how the use of RANK() acts like a group by clause, rolling duplicate data up when transaction type is commented out everywhere.

SQL Formatting Standards



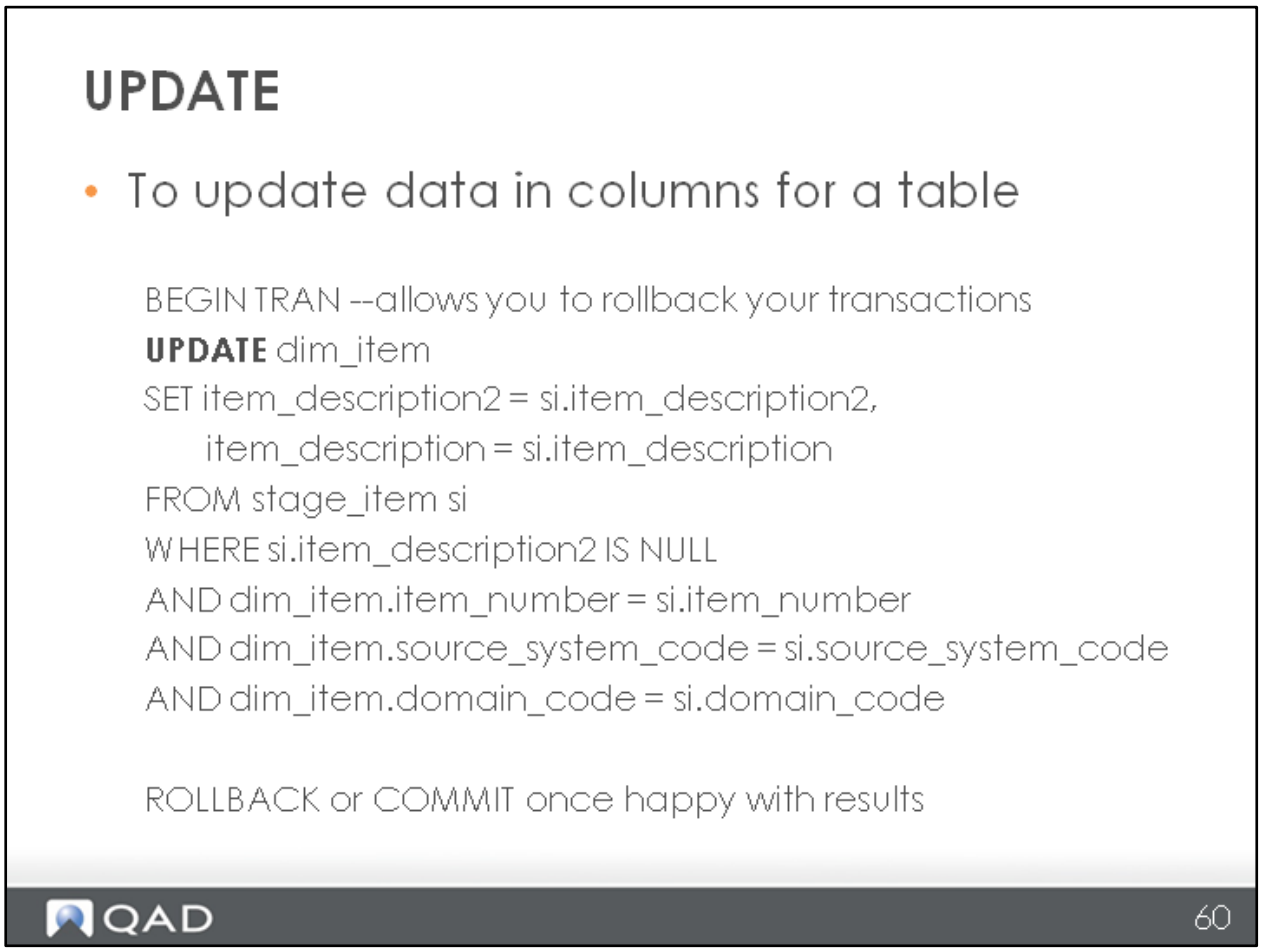
The purpose of this example is to encourage students to not write code where all the columns and FROM clauses are in one line. The code should be easy to follow and easy to see where the FROM and the JOINs are.

INSERT



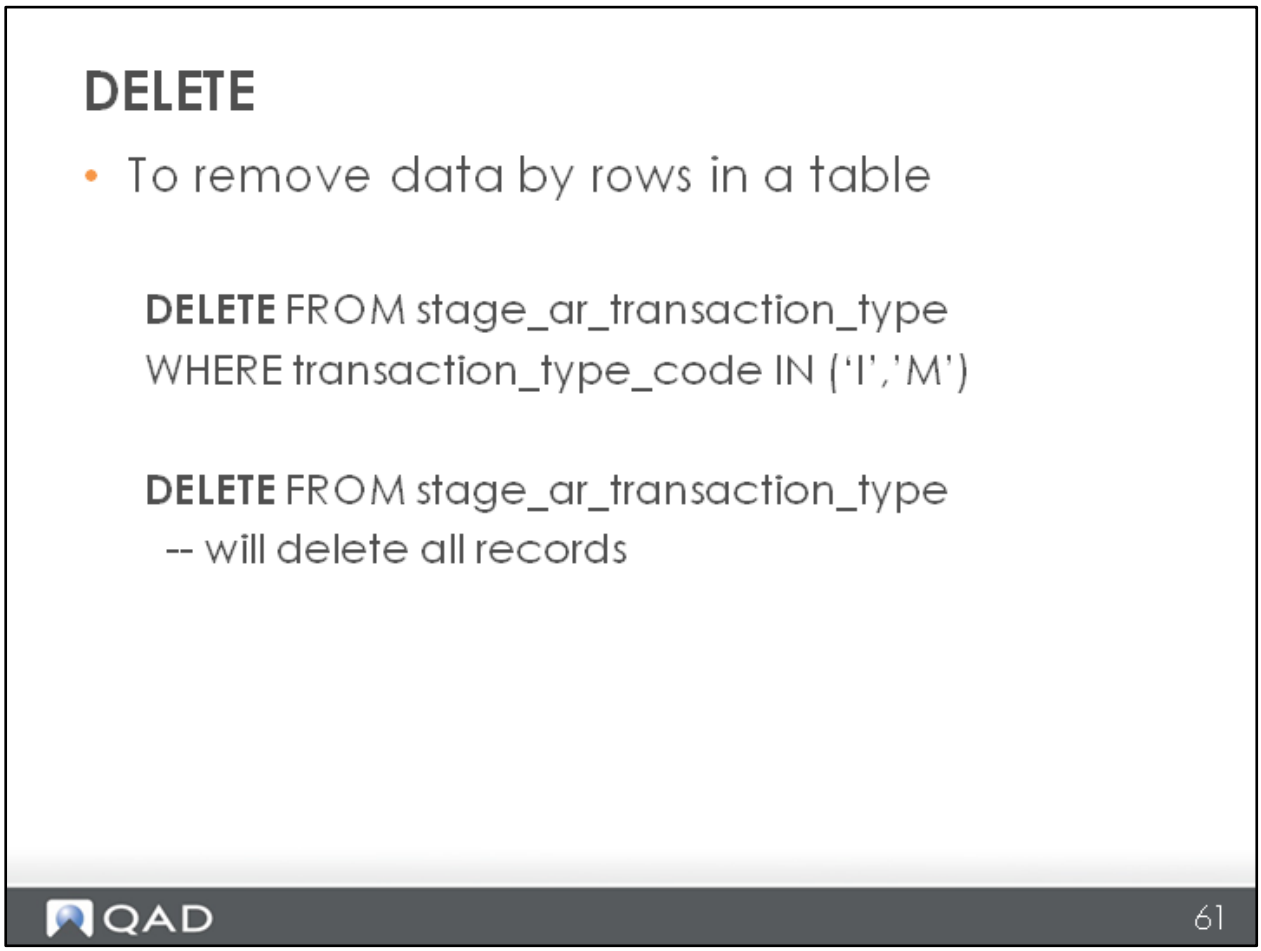
Insert scripts define the table and columns to be inserted into, then selects the columns from the table(s) to be inserted from.

UPDATE



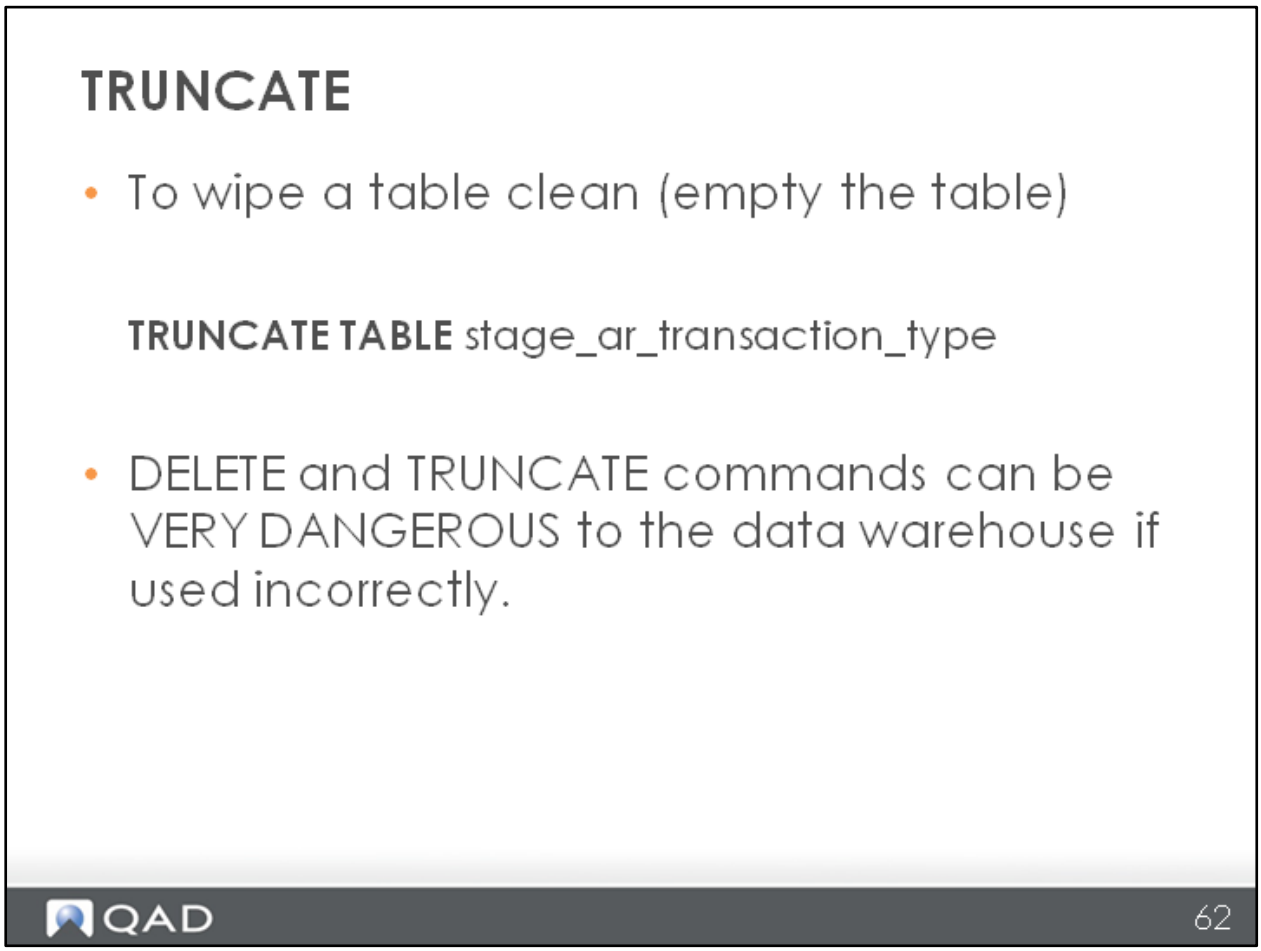
Each update script can only update data in one table, but any one or all column in a table.

DELETE



A delete statement can remove all rows in a table or a single row in a table. For very large tables, it can take a long time to run as the delete script works to find every appropriate row in the table for deletion.

TRUNCATE

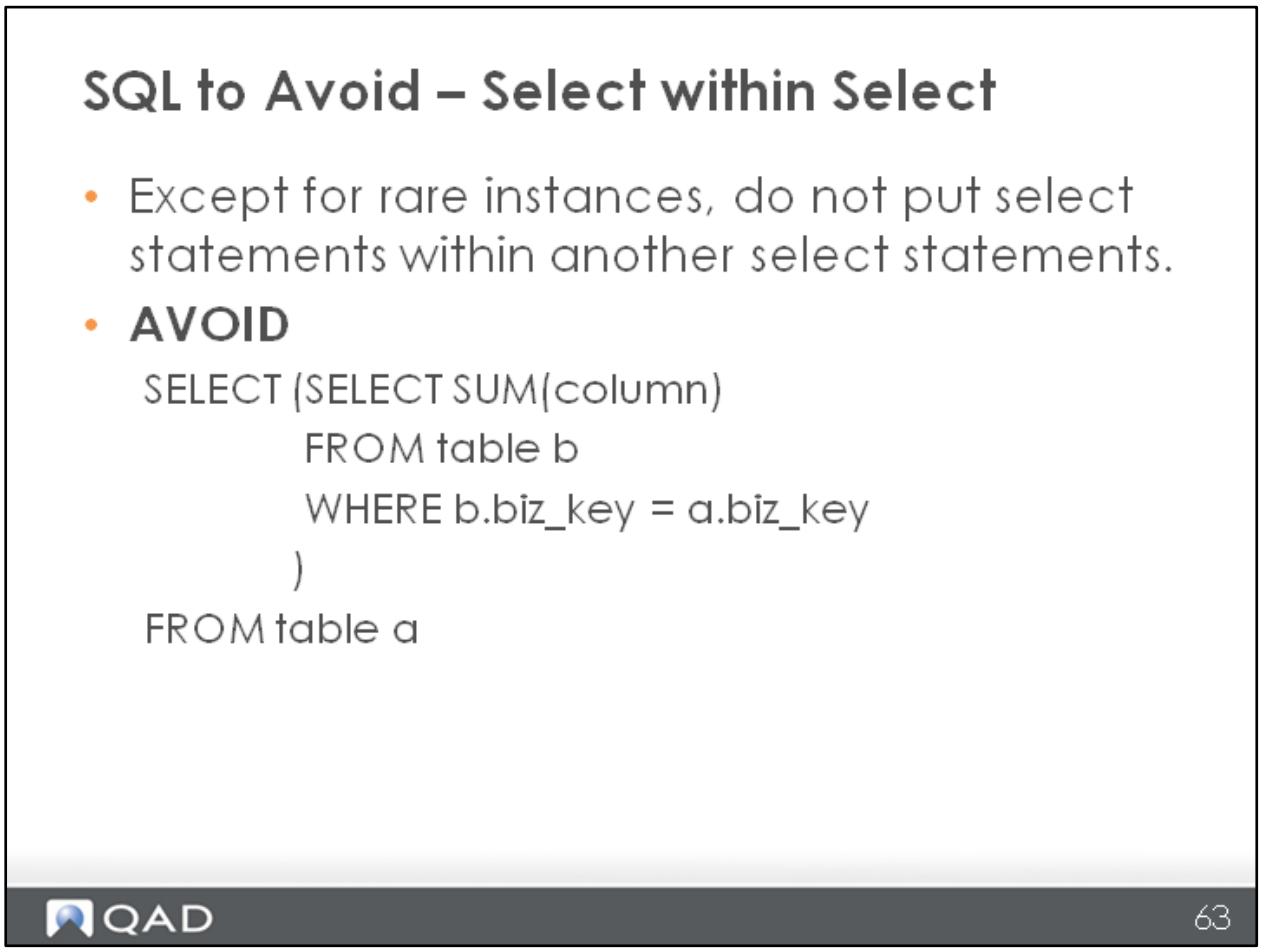


When truncate is used, it is very fast and removes every record in the table. It's important to keep in mind what type of table data is being deleted from and if the table holds data from multiple sources or not. A person can delete or truncate load and stage tables all day with little repercussion. Imagine if a person deletes data from a fact table, what would happen if they have three sources populating a fact table and needed to do a historical reload for the second data source and they used a truncate on that table.

The three types of tables that will be adversely affected by the DELETE or TRUNCATE are:

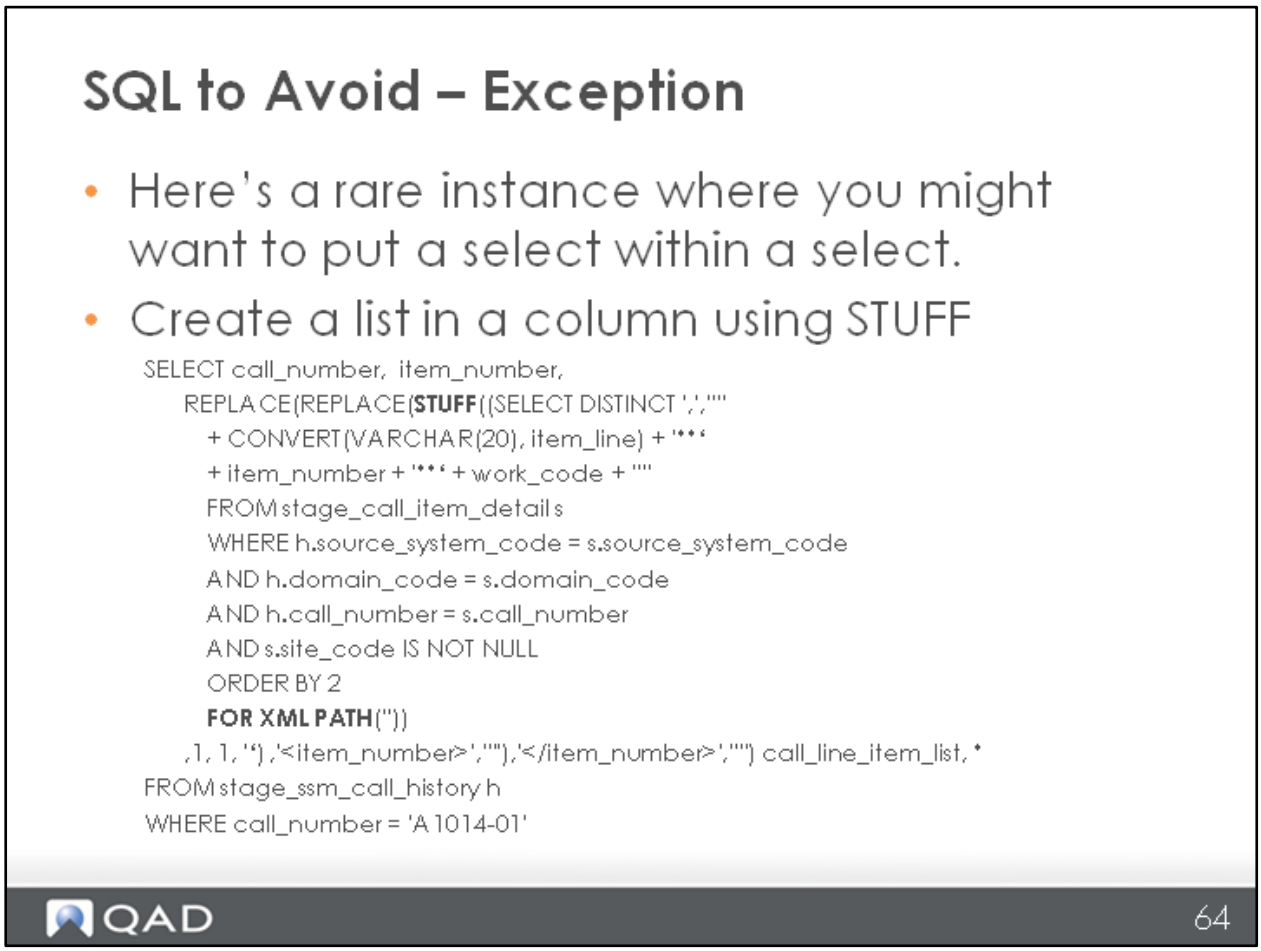
1) Fact tables – will require a historical reload from perm tables. Not too difficult, but can be time consuming.
2) Permanent tables – will require a historical reload, extracting from the ERP source system can be fairly tricky.
3) Dimension tables – deleting one of these is THE BIGGEST DANGER to a data warehouse, because the keys are lost and there is no guarantee that re-populating the table will ensure that the keys get sequenced in the same order. All fact tables that reference those keys will likely need to also be historically reloaded from permanent tables.

SQL to Avoid – Select within Select



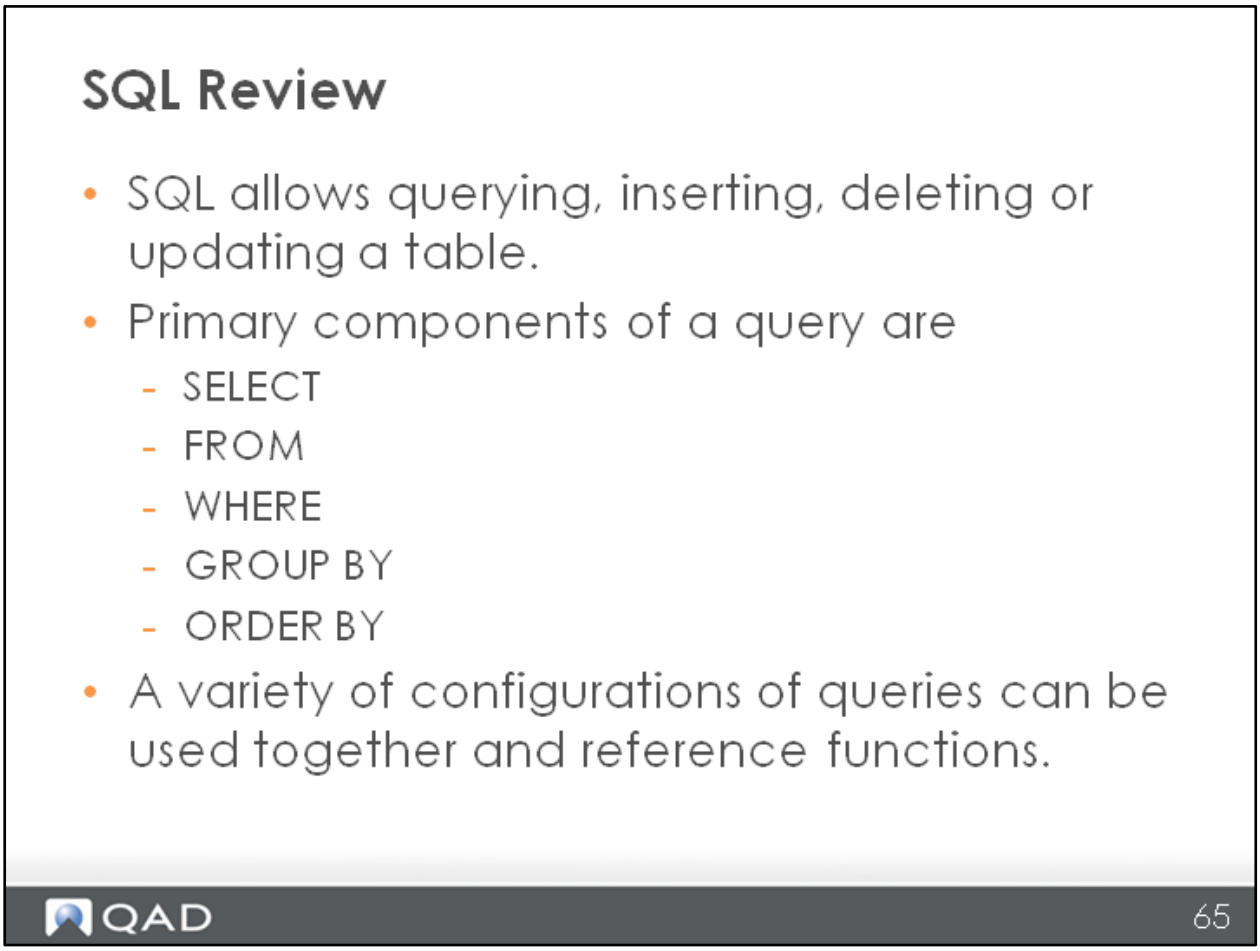
With this, each row would generate another search on the table to make the match in order to do a sum. For something like this it's better to just join the tables in the where clause.

SQL to Avoid – Exception

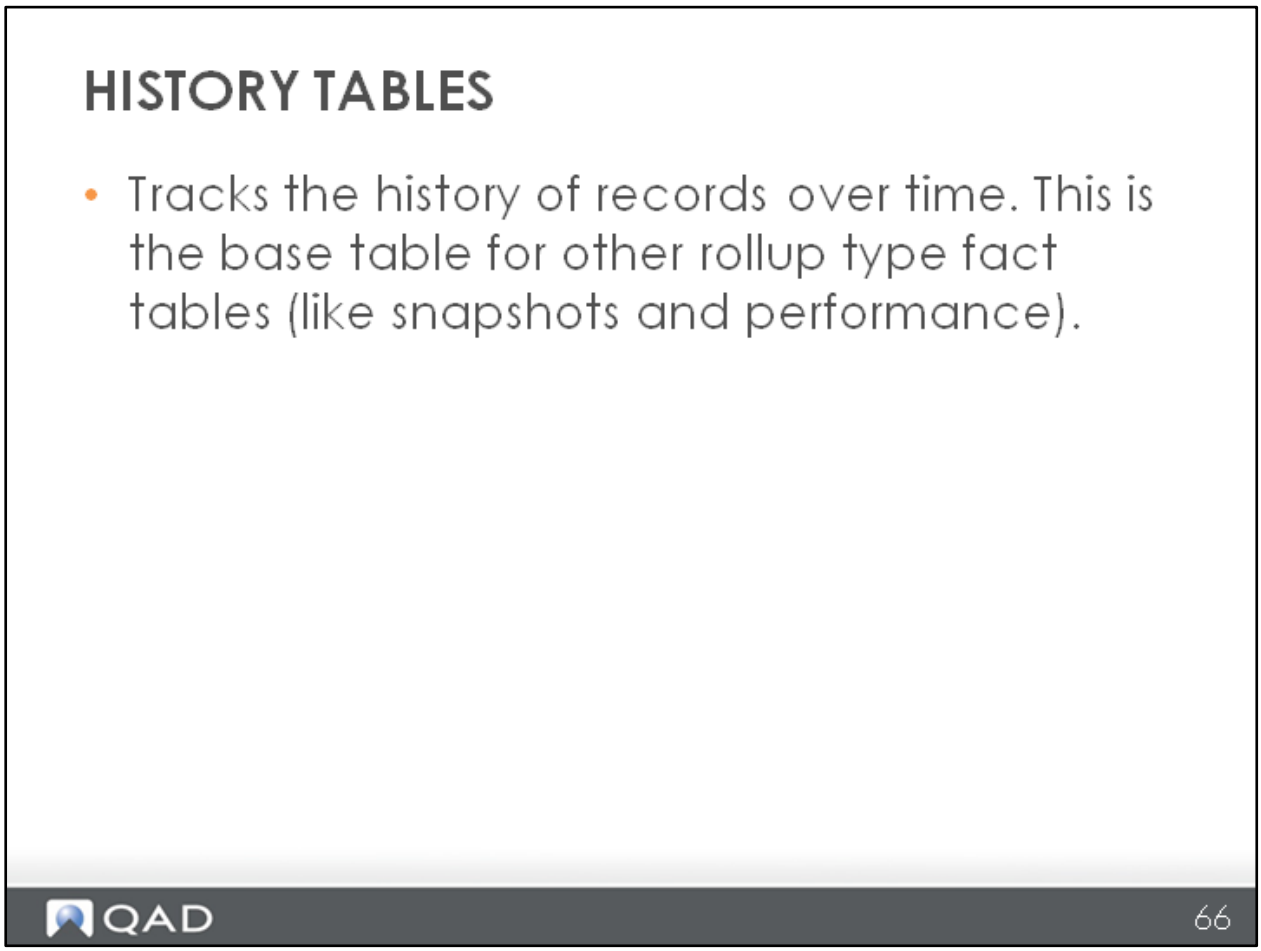


This code will put a concatenated list inside a column, much like we have a list of salespersons in some of the Progress tables.

SQL Review

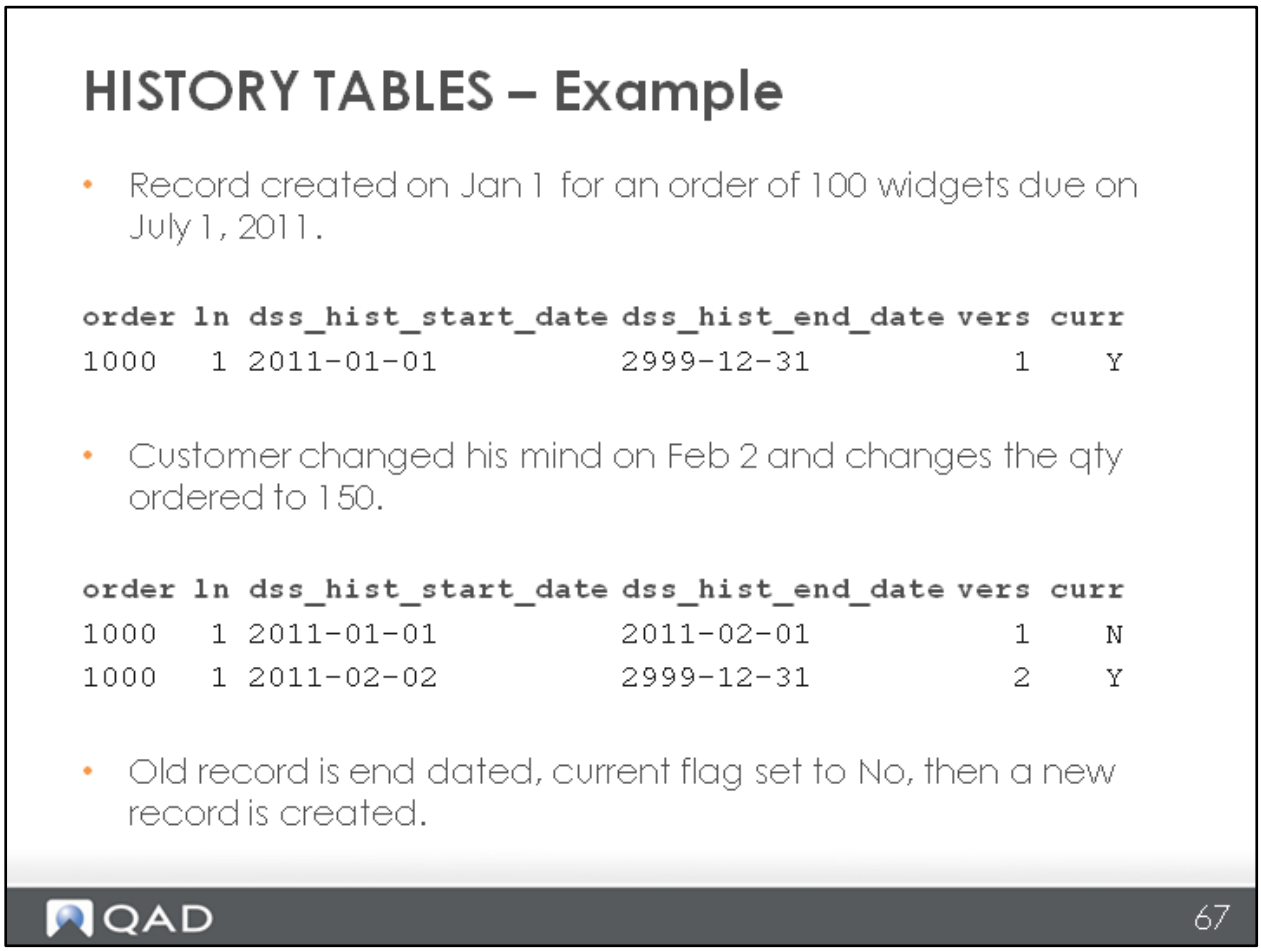

SQL Review
SQL allows querying, inserting, deleting or updating a table.
Primary components of a query are
SELECT
FROM
WHERE
GROUP BY
ORDER BY
A variety of configurations of queries can be used together and reference functions.
QAD
65

HISTORY TABLES

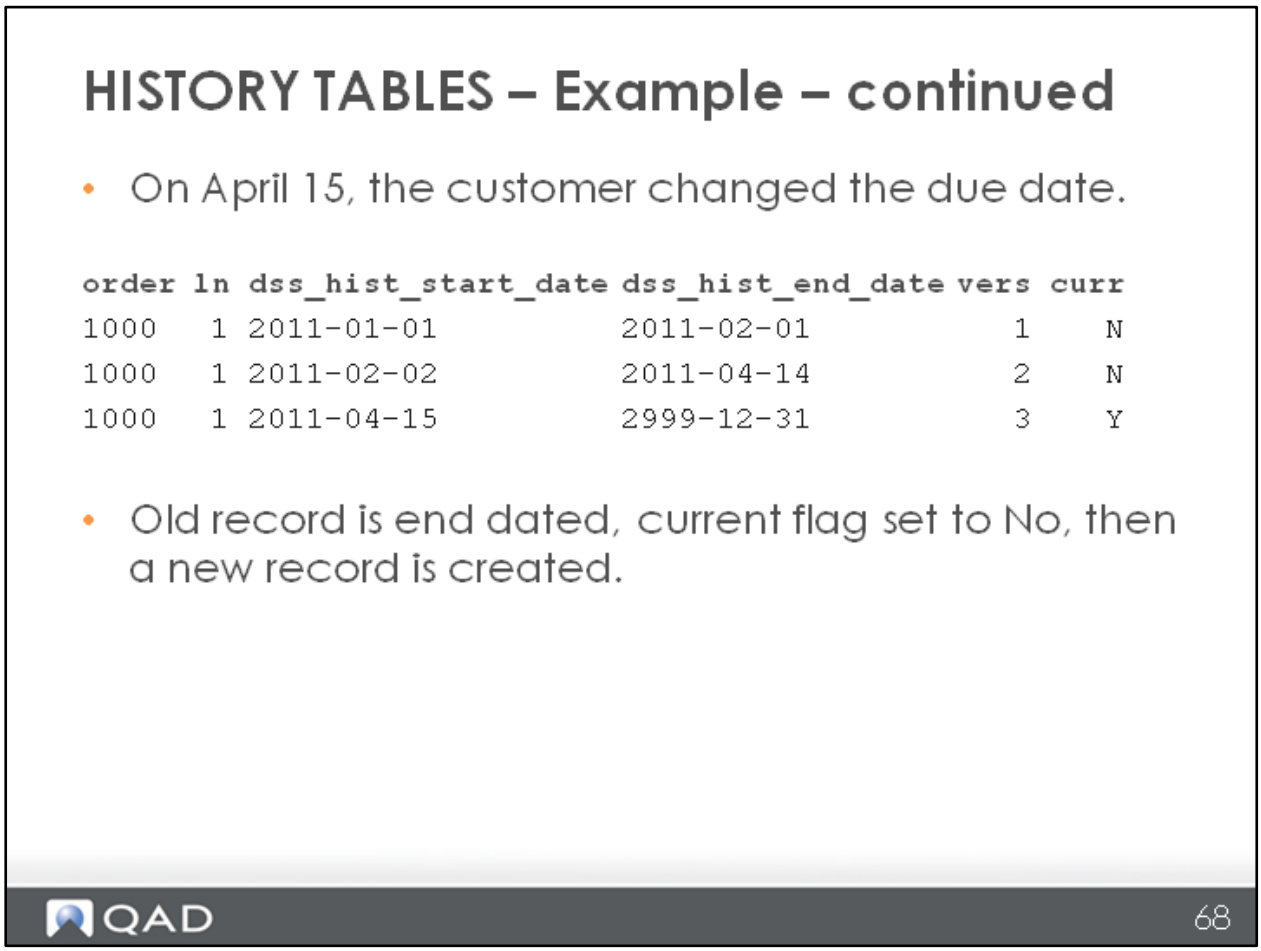


We'll detour here for a moment to look at how history tables work. Our first SQL exercise will work with a history table so it is helpful to understand how they work.

HISTORY TABLES – Example



HISTORY TABLES – Example – continued



SNAPSHOT FROM HISTORY TABLE

SNAPSHOT FROM HISTORY TABLE

- Creates snapshots of the data from certain points in time (daily, weekly, monthly or both monthly and weekly).
- Using prior example …

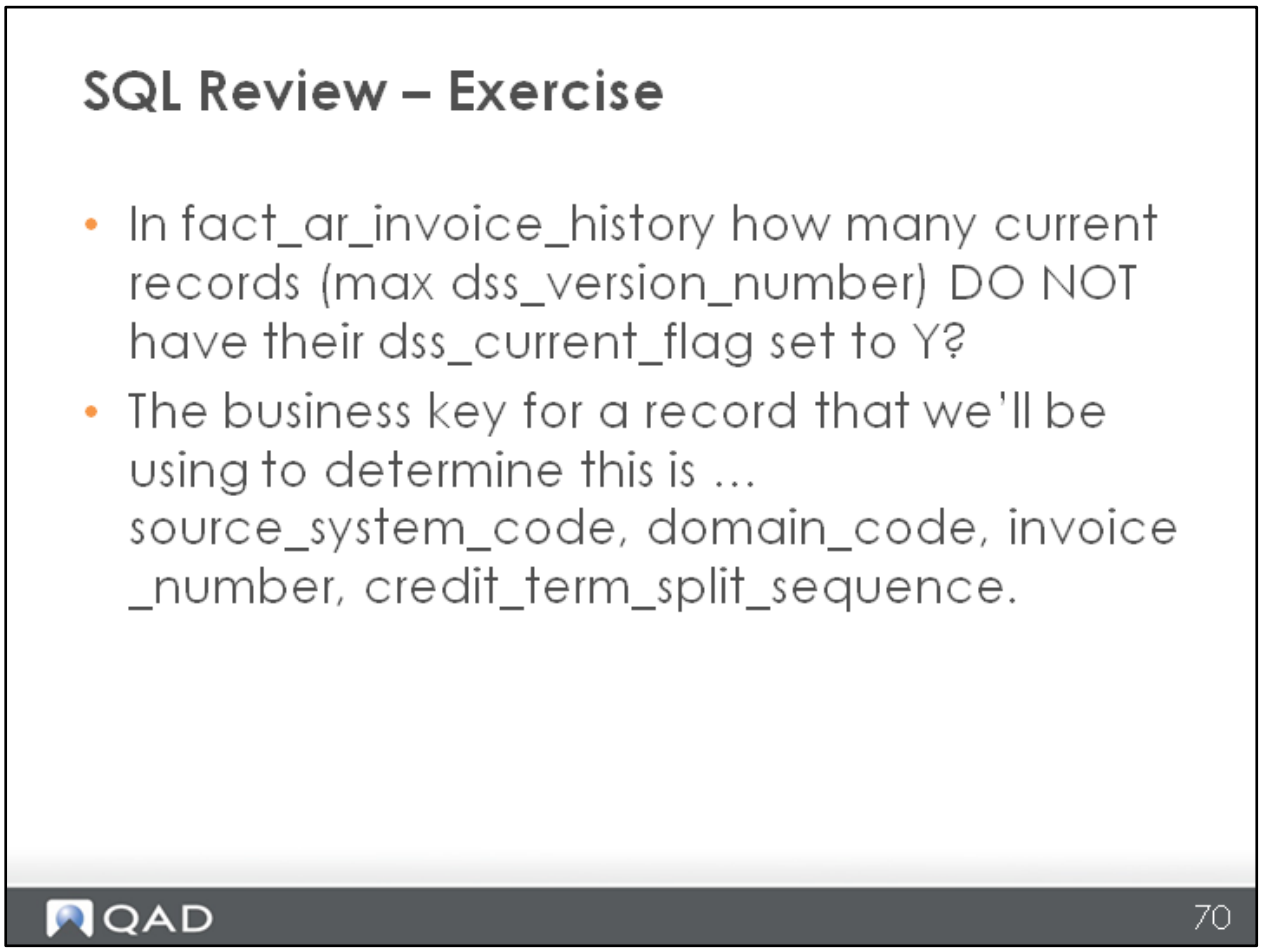
order	ln	dss_hist_start_date	dss_hist_end_date	vers	curr
1000	1	2011-01-01	2011-02-01	1	N
1000	1	2011-02-02	2011-04-14	2	N
1000	1	2011-04-15	2999-12-31	3	Y

- Snapshot would show …

Month	Order	Ln	Effective Date	Version
January	1000	1	2011-01-31	1
February	1000	1	2011-02-28	2
March	1000	1	2011-03-31	2
April	1000	1	2011-04-30	3
May	1000	1	2011-05-31	3
June	1000	1	2011-06-30	3

QAD 69

SQL Review – Exercise



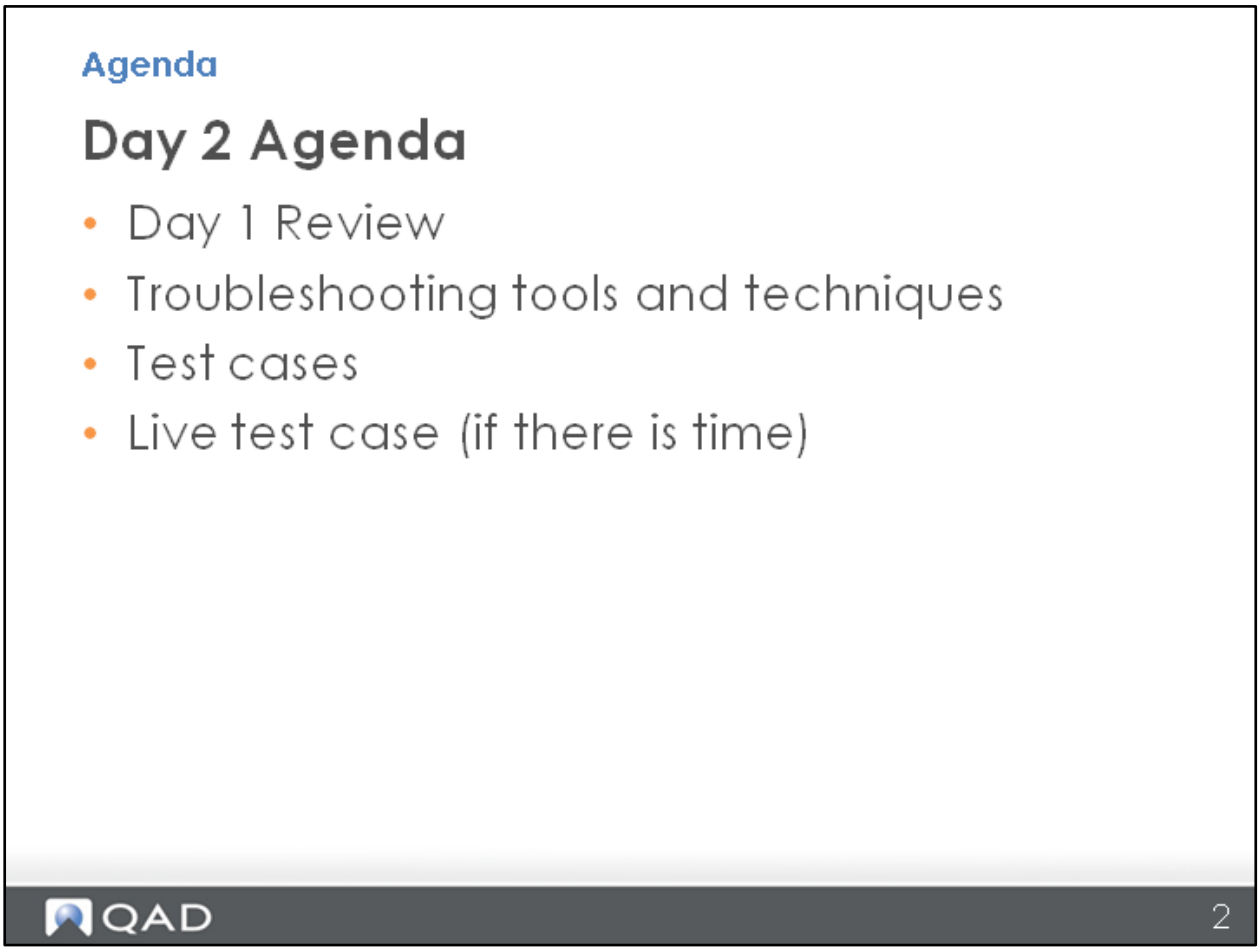
CHAPTER 2

QAD BI v3 – Tech Level 2 Certification - Part 2

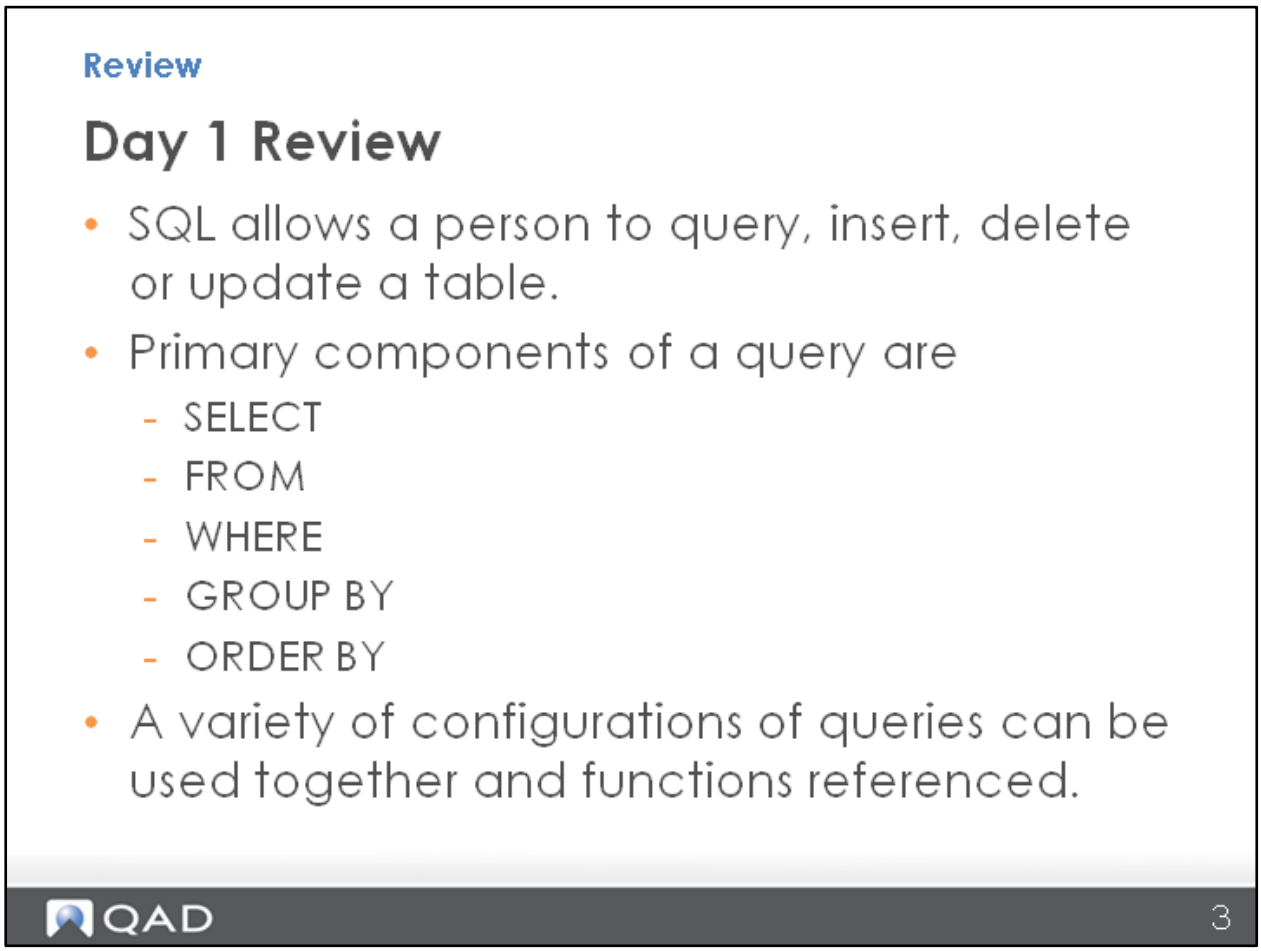
QAD BI v3 – Technical Level II Certification Class – Part 2



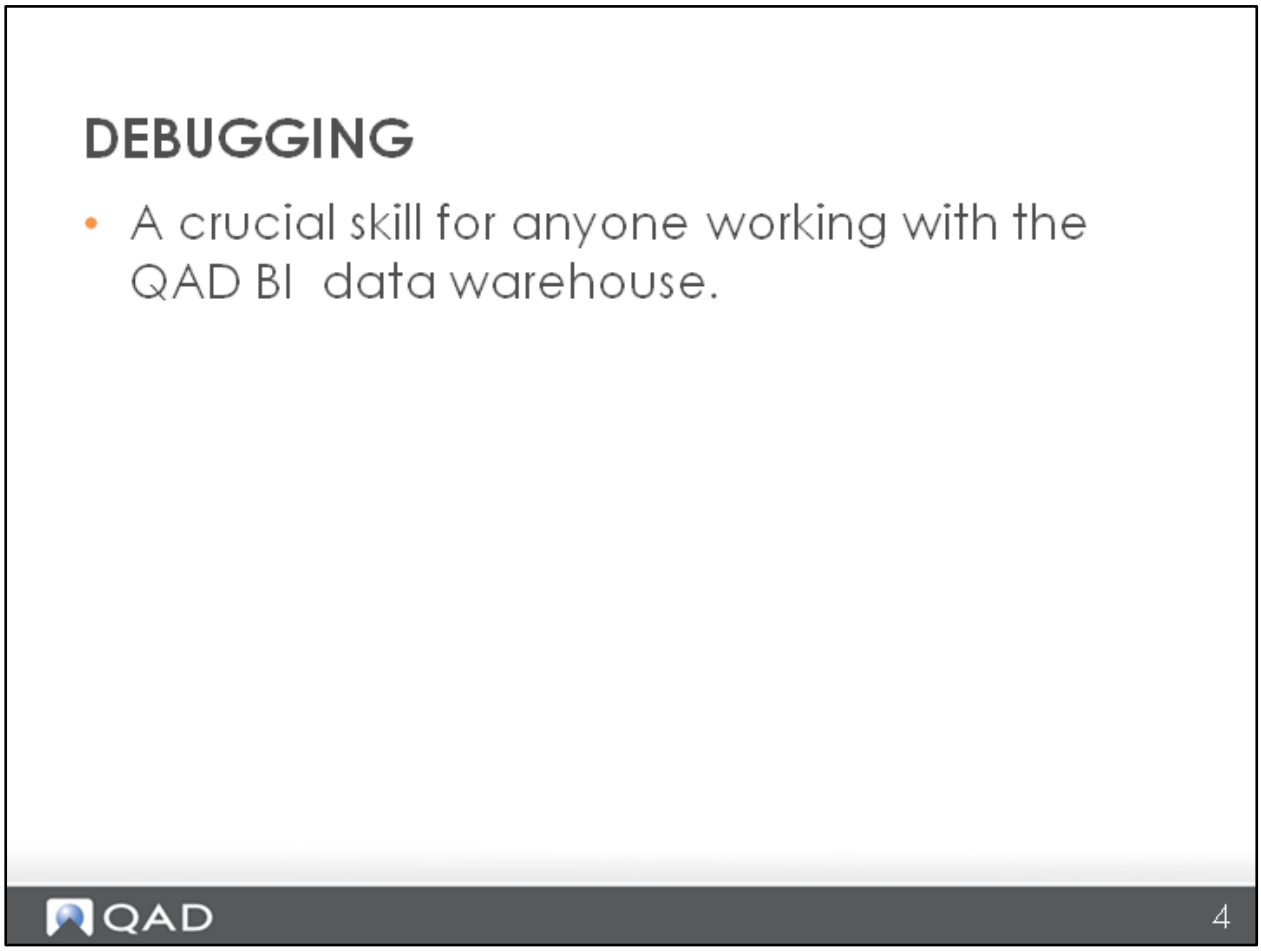
Day 2 Agenda


Agenda
Day 2 Agenda
Day 1 Review
Troubleshooting tools and techniques
Test cases
Live test case (if there is time)
QAD
2

Day 1 Review

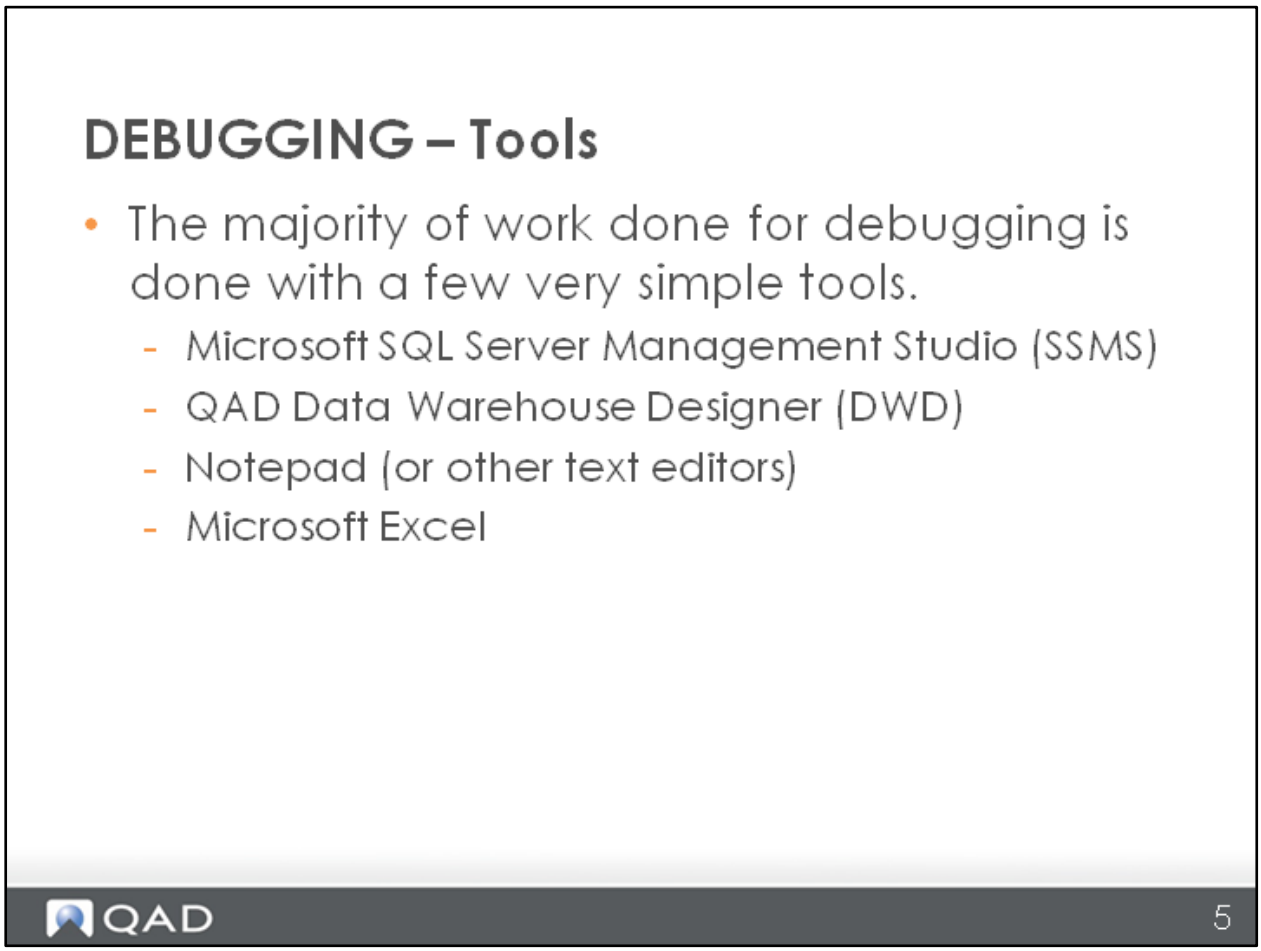

Review
Day 1 Review
• SQL allows a person to query, insert, delete or update a table.
• Primary components of a query are
- SELECT
- FROM
- WHERE
- GROUP BY
- ORDER BY
• A variety of configurations of queries can be used together and functions referenced.
QAD
3

DEBUGGING

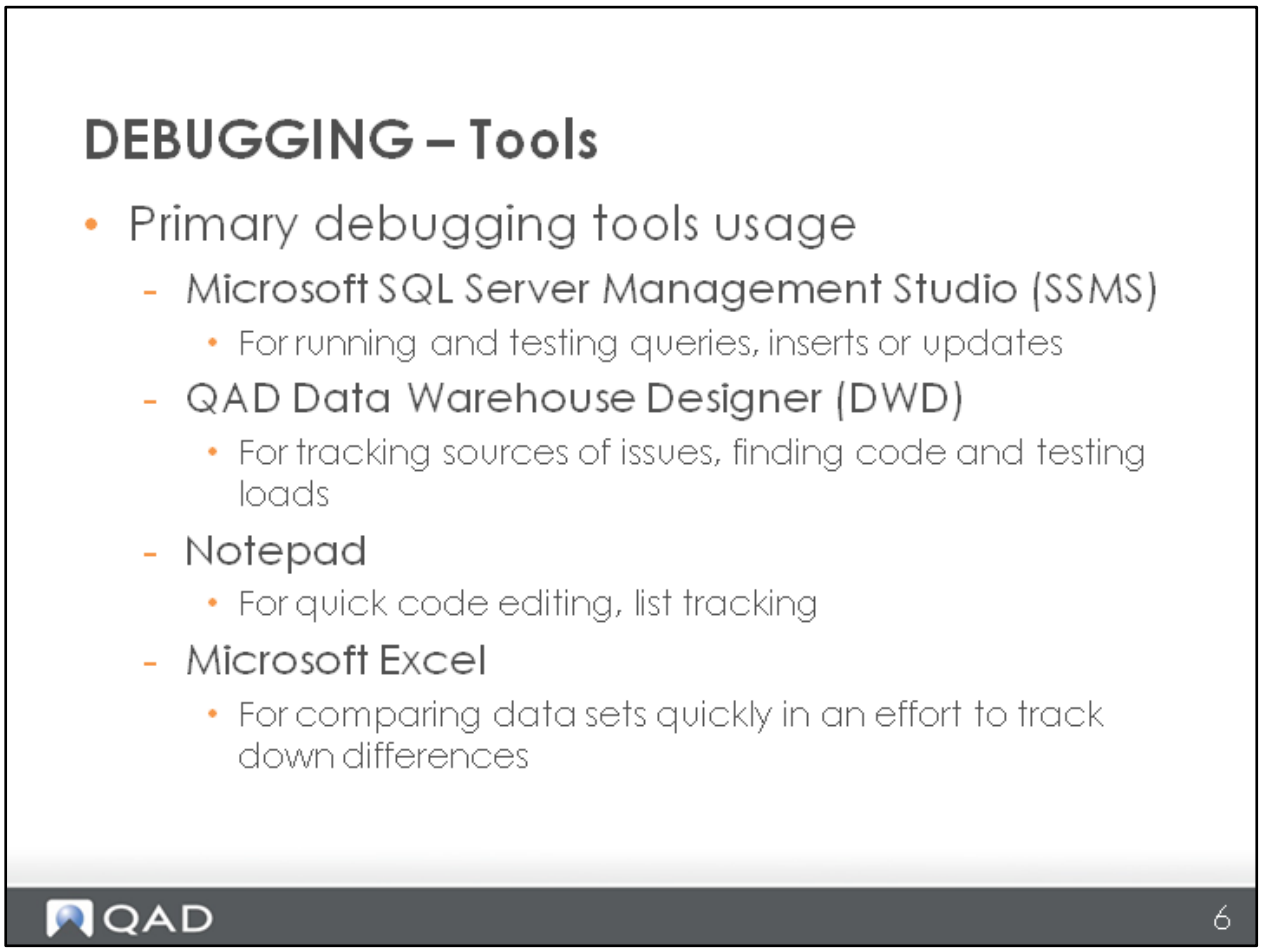


When building a data warehouse, it is nearly impossible to foresee all the possible permutations of data that might enter it. A data warehouse is driven completely by the data that enters it. There are frequently new scenarios that surface (especially with newer, less stable modules) that can cause a load to fail or cause the data in the warehouse to not be what it should be.

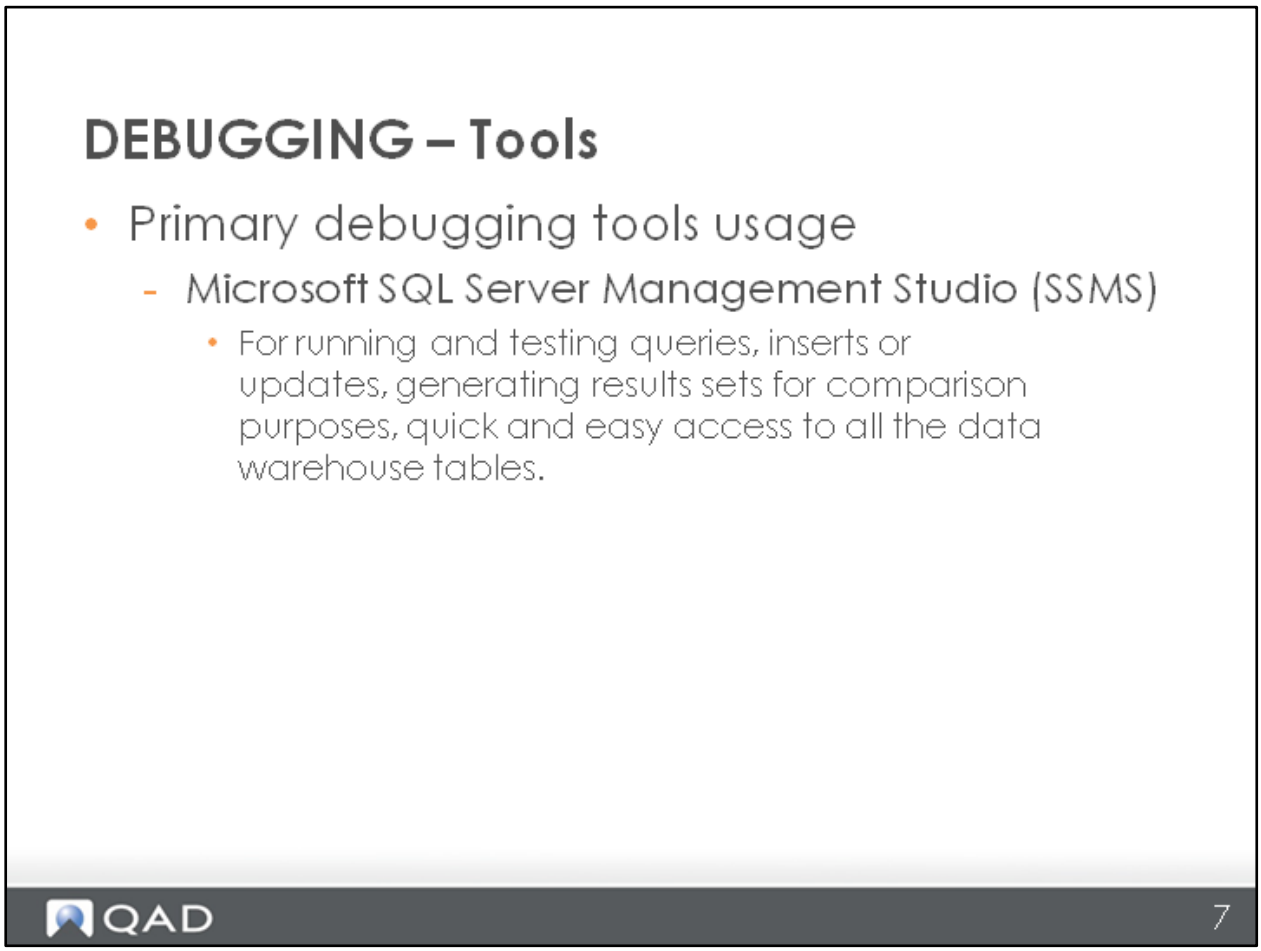
DEBUGGING – Tools



DEBUGGING – Tools

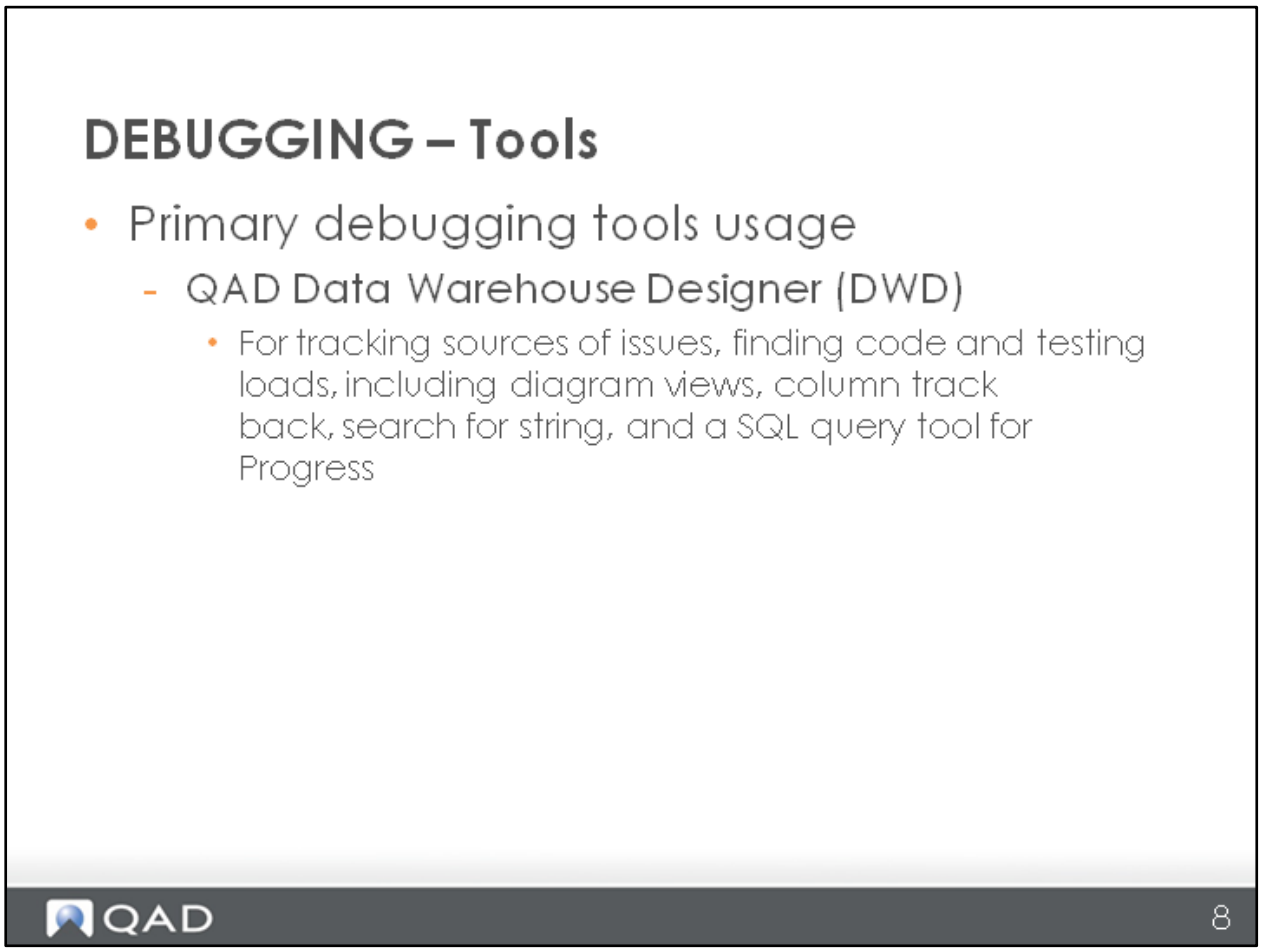


DEBUGGING – Tools



With MS SQL Server, queries, updates, and inserts can be run against various databases, code from the DWD can be dissected and run down to its most essential components, table structures and indexes can be compared against the versions in the DWD, query result sets can be generated for comparison.

DEBUGGING – Tools



DEBUGGING – Tools



DEBUGGING – Tools



Troubleshooting Tools

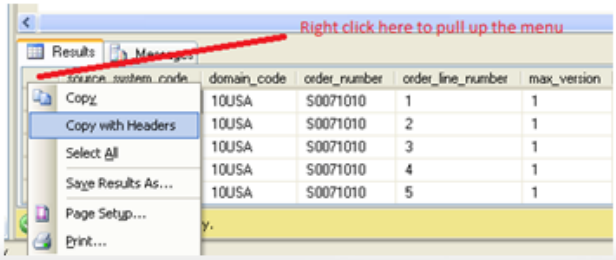


SQL Server (SSMS) for Troubleshooting

Troubleshooting Tools

SQL Server (SSMS) for Troubleshooting

- Primarily used to run queries to check for results that match or counter expectations.
- In SQL Server, result sets can be copied with or without their header by right clicking on the top left hand corner of the results pane. These results can be copied into Microsoft Excel to look for differences.



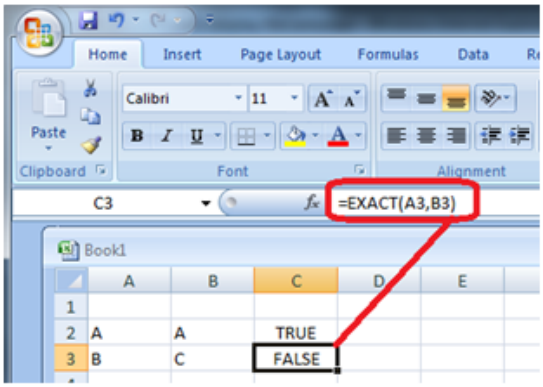
QAD 12

Microsoft Excel for Troubleshooting

Troubleshooting Tools

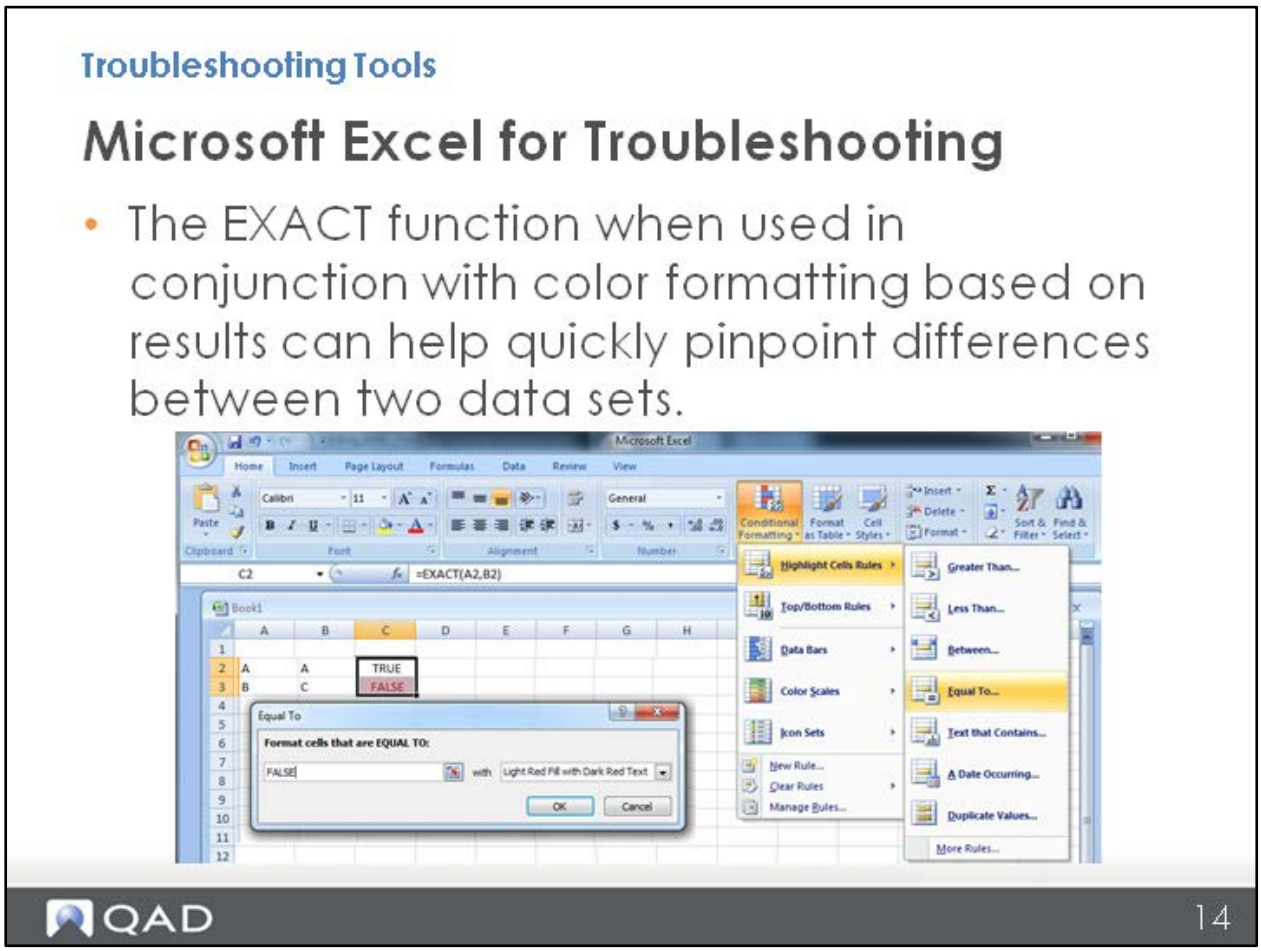
Microsoft Excel for Troubleshooting

- There is a function called =EXACT(Cell1,Cell2) that will provide a TRUE/FALSE result set when comparing cells to see if they are the same or not.



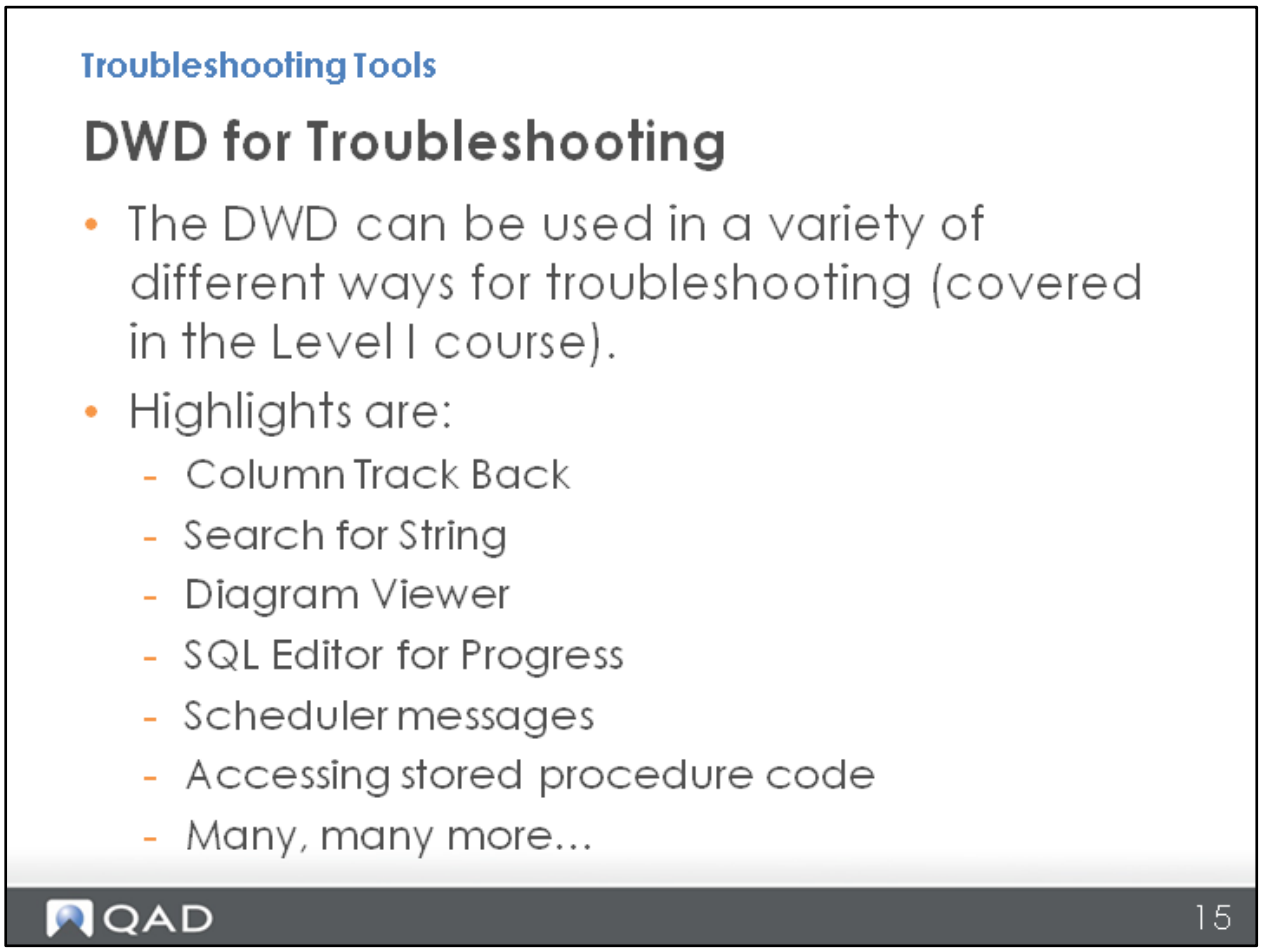
QAD 13

Microsoft Excel for Troubleshooting



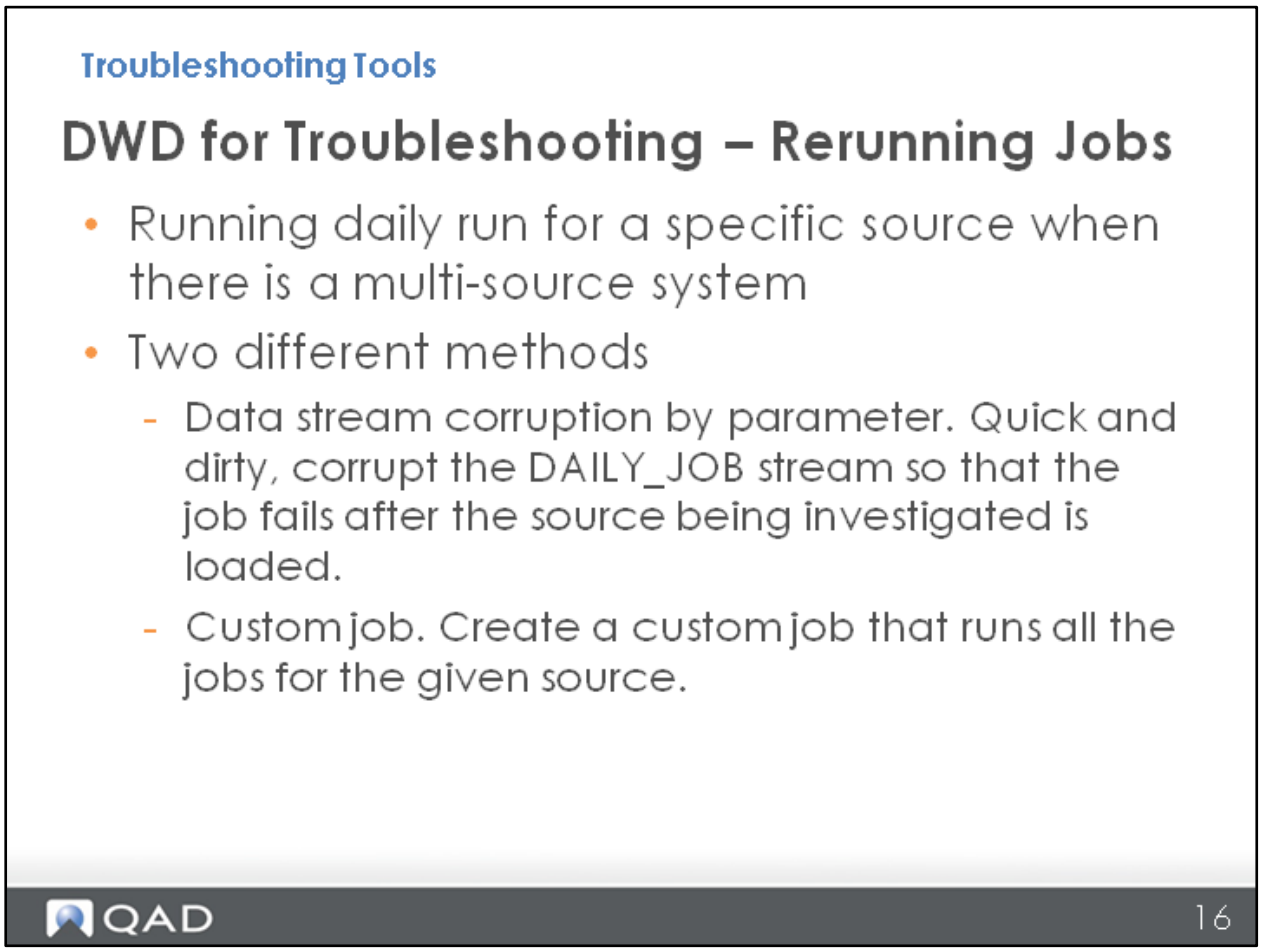
Different versions of Excel have different options for highlighting results with colors based on the results. Most versions have some type of conditional formatting option, then a choice of how the chosen cell results will get highlighted (we'll be using the Equal To option). Set the Equal To value to FALSE and pick the color scheme to be displayed when a match to FALSE is found. The path to get to this varies across different versions of Excel, but the principles are the same.

DWD for Troubleshooting



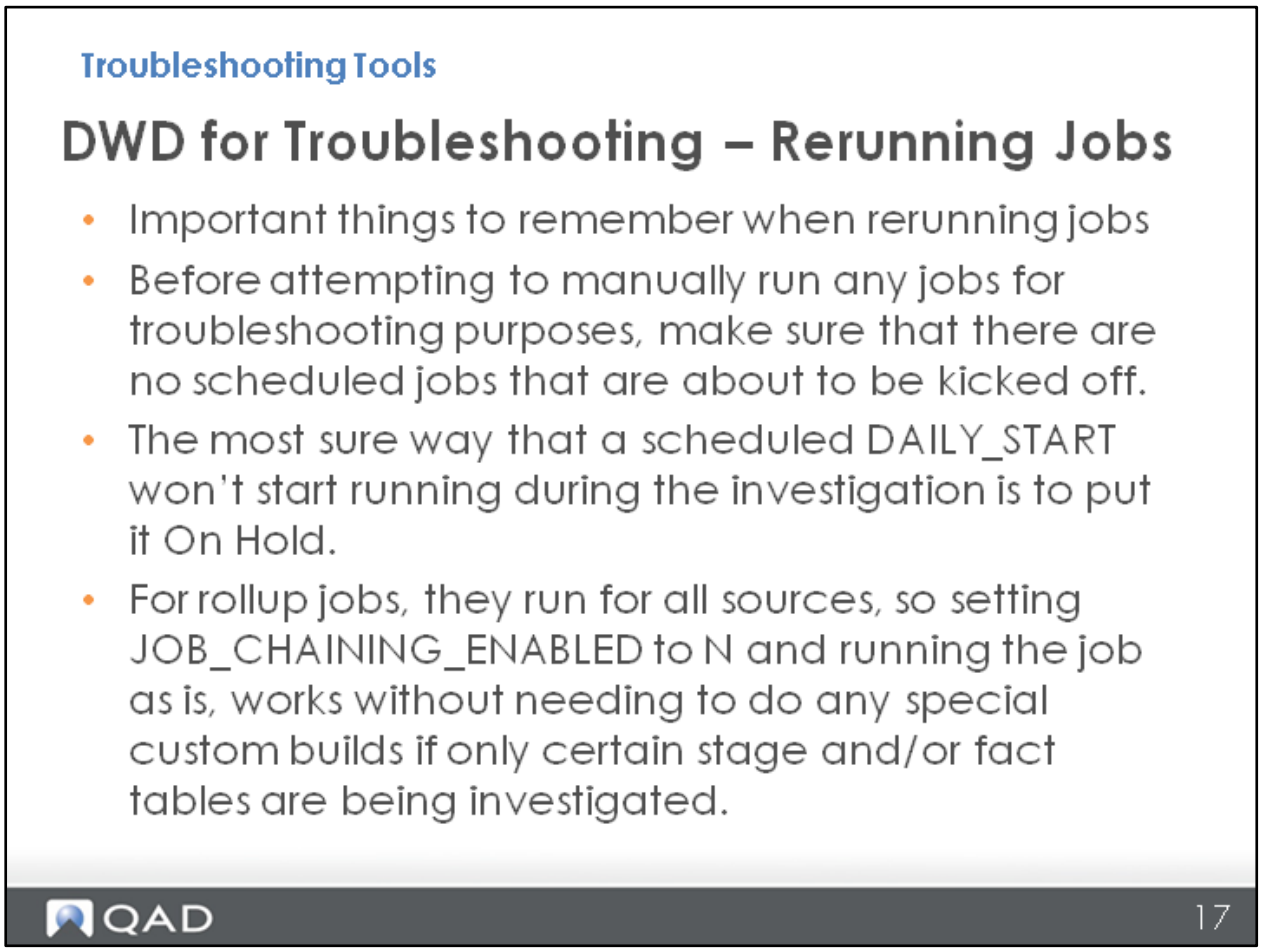
We'll be applying all of these in the list, plus more, such as looking at column sources in table structures.

DWD for Troubleshooting – Rerunning Jobs



When troubleshooting on a multi-source system, sometimes the source system data being investigated is not in the stage tables because another source system ran after the one being investigated in the daily run.

DWD for Troubleshooting – Rerunning Jobs



DWD for Troubleshooting – Rerunning Jobs

Troubleshooting Tools

DWD for Troubleshooting – Rerunning Jobs

- Daily job stream corruption methodology
- In the daily job stream after the source being investigated is populated, find the job that sets the connection for the next data source and add some extra characters to the end of the name, ensuring it's named for a job that doesn't exist.
- In this example SEDemo had _xxxxxx added to it.

DAILY_LOAD_JOB0018	DAILY_OP_INV_PROCESS_CHAINED	Inserted via WsParameterWrite
DAILY_LOAD_JOB0019	DAILY_SSM_PERM_CHAINED	Inserted via WsParameterWrite
DAILY_LOAD_JOB0020	DAILY_SSM_PROCESS_CHAINED	Inserted via WsParameterWrite
DAILY_LOAD_JOB0021	SET_CONNECTION_SEDemo_xxxxxx	Inserted via WsParameterWrite
DAILY_LOAD_JOB0022	PREPROCESS_COMMON_TRUNCATE	Inserted via WsParameterWrite
DAILY_LOAD_JOB0023	PREPROCESS_OP_TRUNCATE	Inserted via WsParameterWrite
DAILY_LOAD_JOB0024	PREPROCESS_SSM_TRUNCATE	Inserted via WsParameterWrite

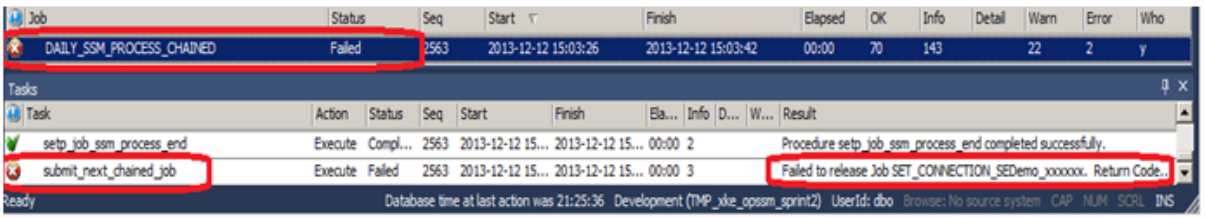
QAD 18

DWD for Troubleshooting – Rerunning Jobs

Troubleshooting Tools

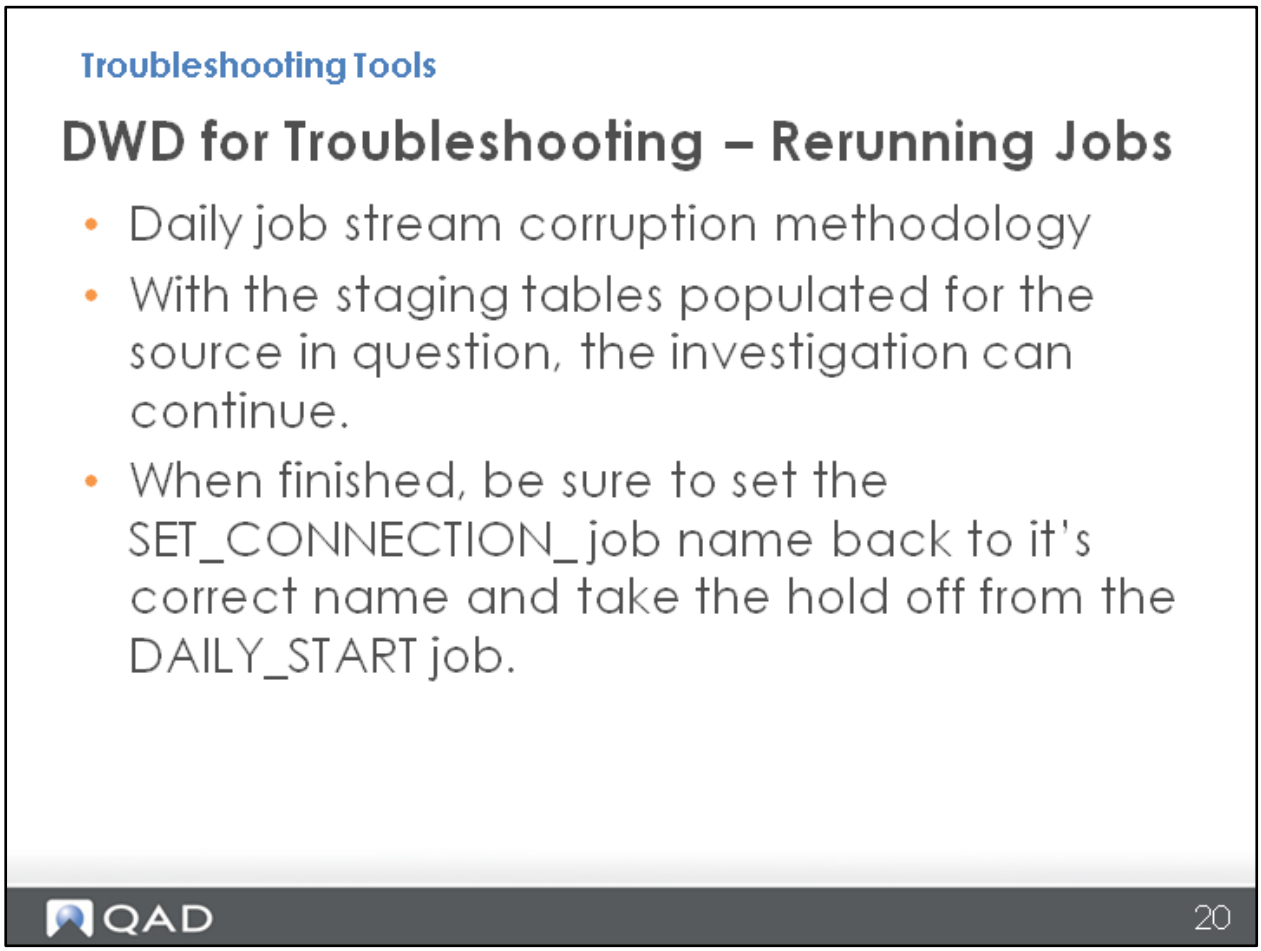
DWD for Troubleshooting – Rerunning Jobs

- Daily job stream corruption methodology
- Kick off DAILY_START (right click, Start the Job). When it finishes with the main part of the source system load and goes to kick off the next SET_CONNECTION_ job, it will fail because the DAILY_JOB parameter we renamed doesn't exist in the scheduler.

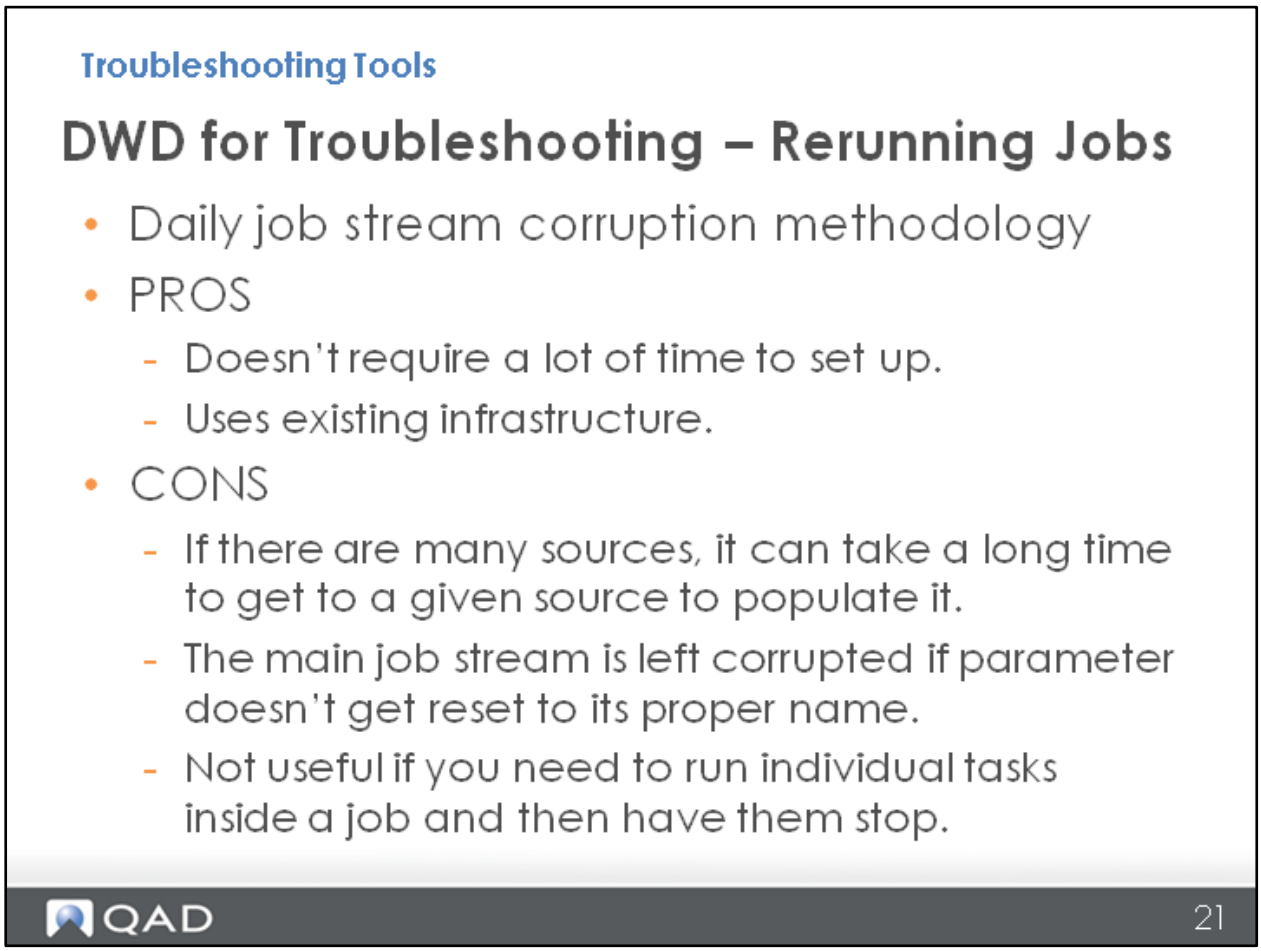


QAD 19

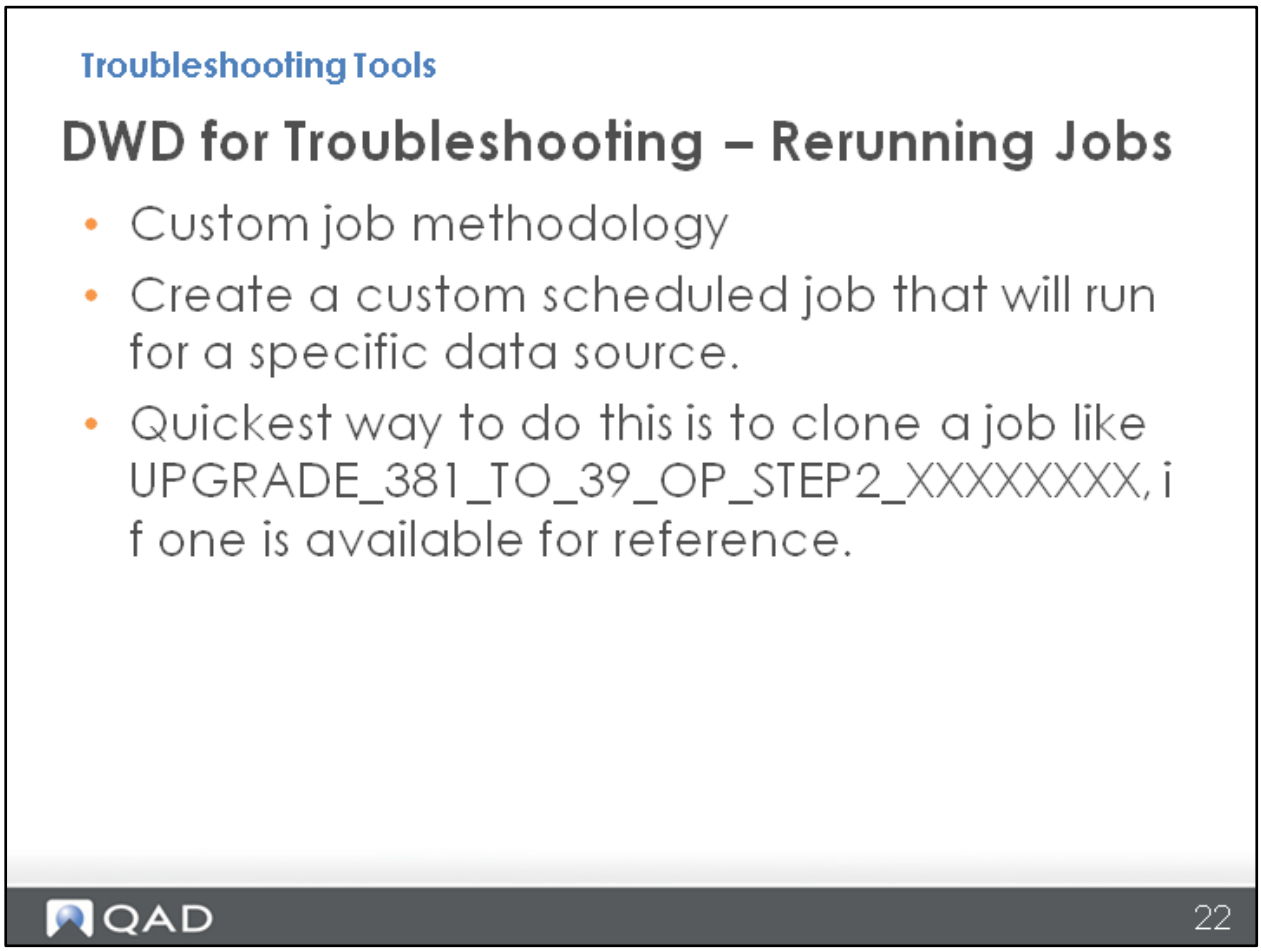
DWD for Troubleshooting – Rerunning Jobs



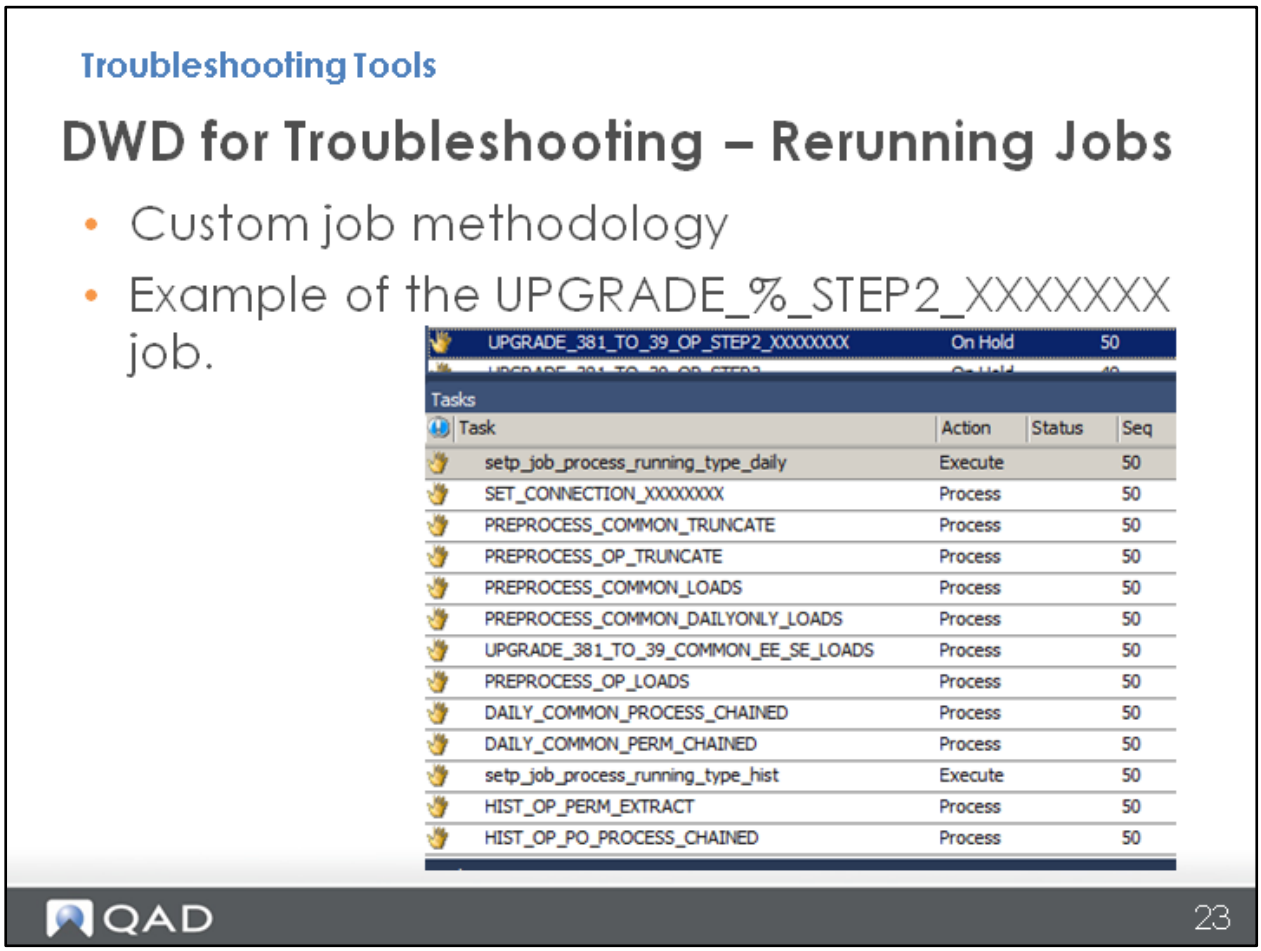
DWD for Troubleshooting – Rerunning Jobs



DWD for Troubleshooting – Rerunning Jobs

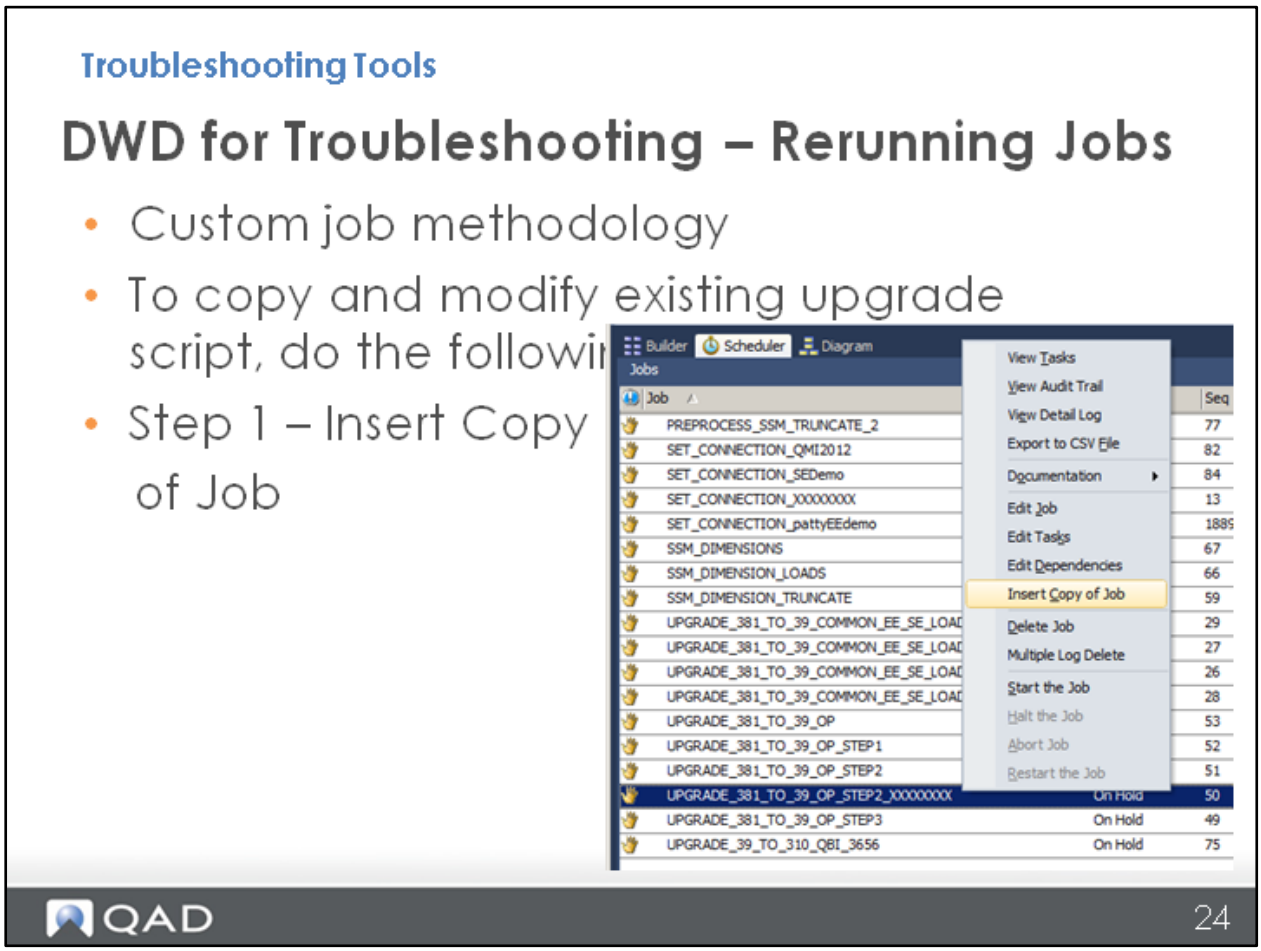


DWD for Troubleshooting – Rerunning Jobs



The UPGRADE_ STEP2_XXXXXXX jobs are used by the upgrade job to generate source specific upgrade jobs. An earlier step in the UPGRADE process, converts the *XXXXXX of the UPGRADE* STEP2_XXXXXXXX job to another source name for each appropriate source. It also does the same with the SET_CONNECTION_XXXXXXX task name inside the upgrade job.

DWD for Troubleshooting – Rerunning Jobs



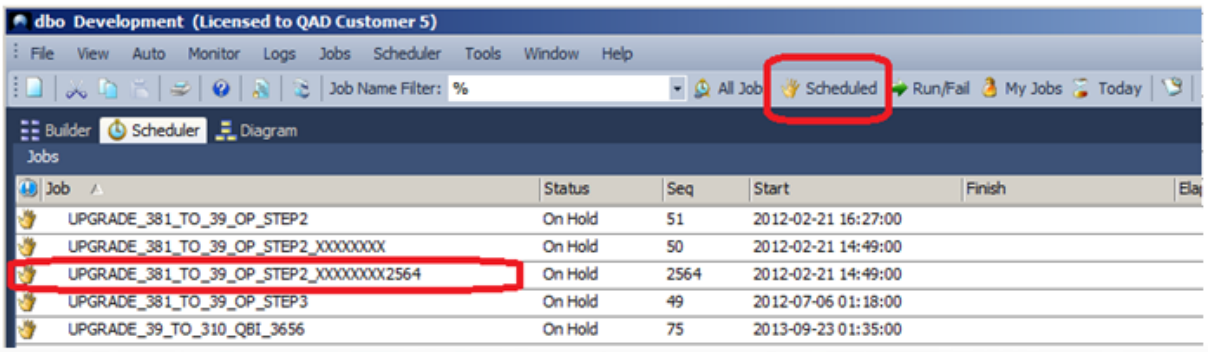
Right click on the UPGRADE_ STEP2_XXXXXXX job and choose Insert Copy of Job.

DWD for Troubleshooting – Rerunning Jobs

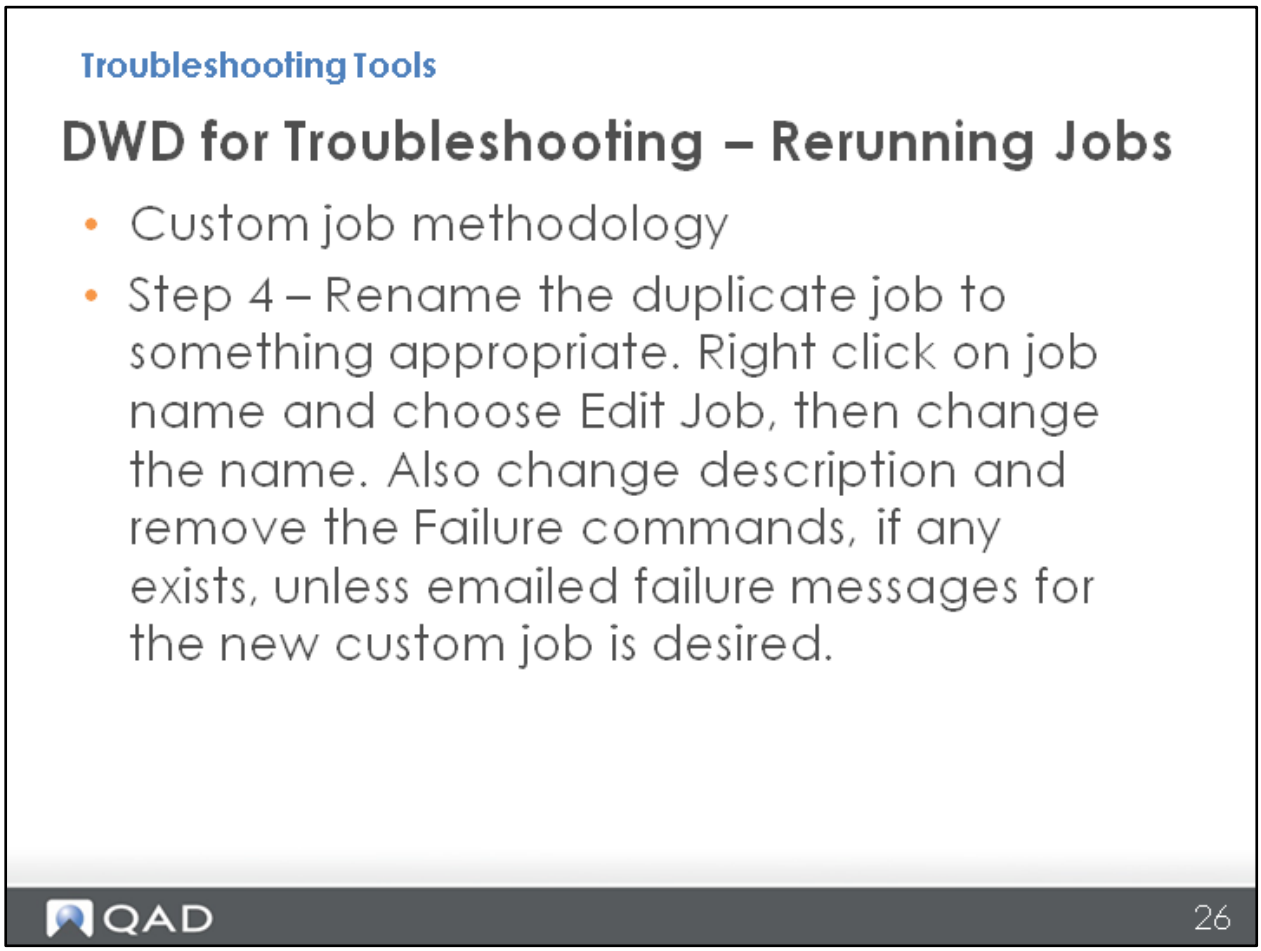
Troubleshooting Tools

DWD for Troubleshooting – Rerunning Jobs

- Custom job methodology
- Step 2 – Refresh the scheduled job list by clicking on the Scheduled button.
- Step 3 – Find copy of job in list.



DWD for Troubleshooting – Rerunning Jobs

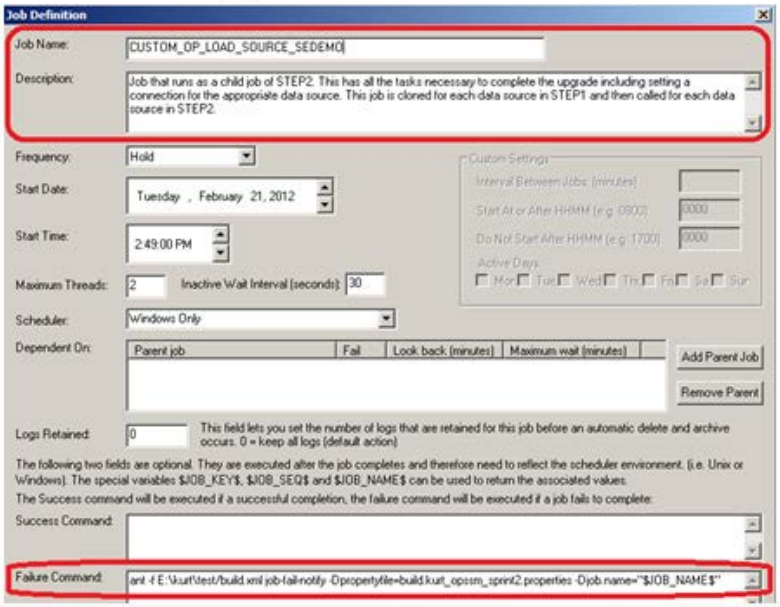


DWD for Troubleshooting – Rerunning Jobs

Troubleshooting Tools

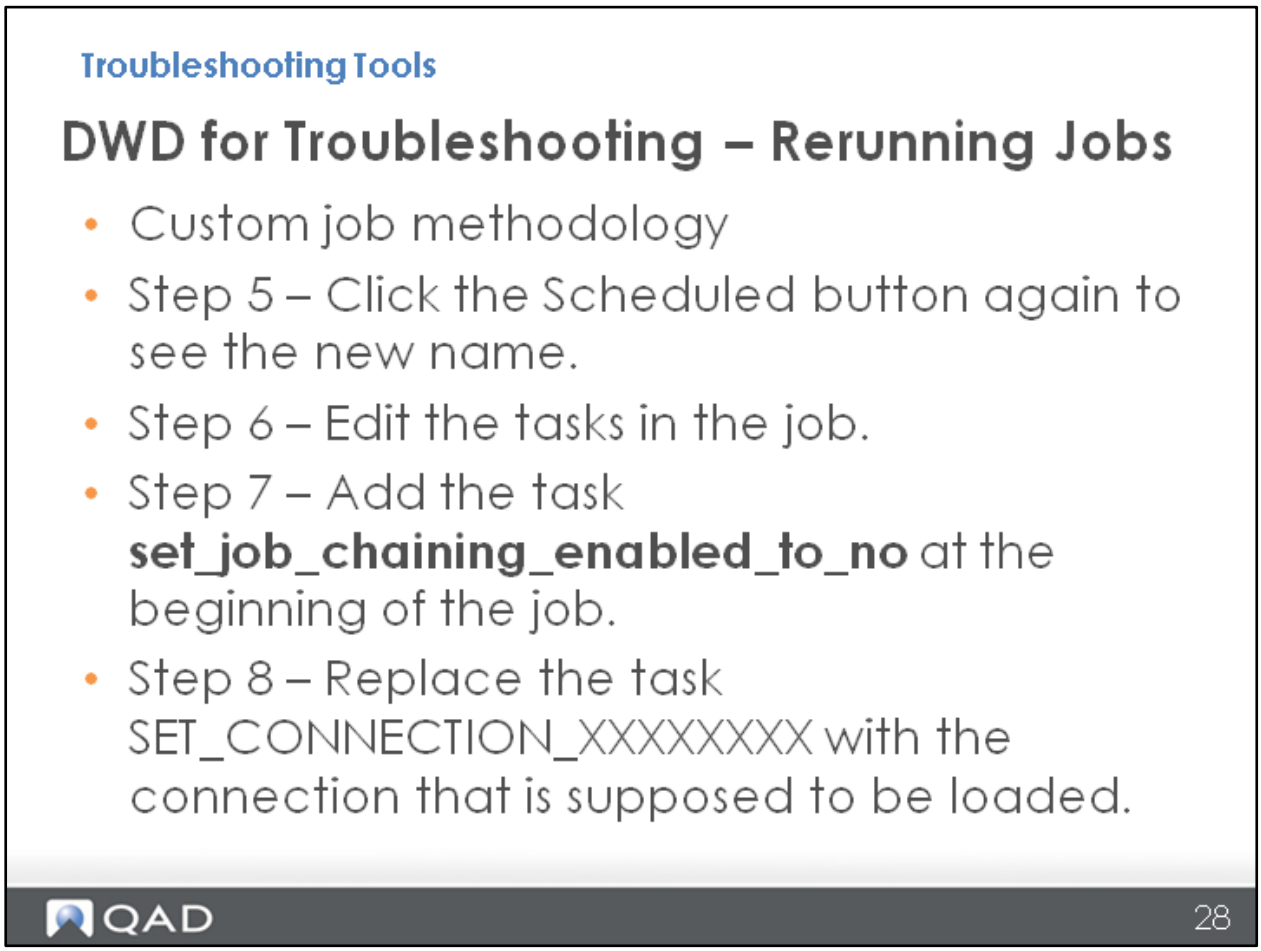
DWD for Troubleshooting – Rerunning Jobs

- Custom job methodology
- Step 4 – cont.



QAD 27

DWD for Troubleshooting – Rerunning Jobs



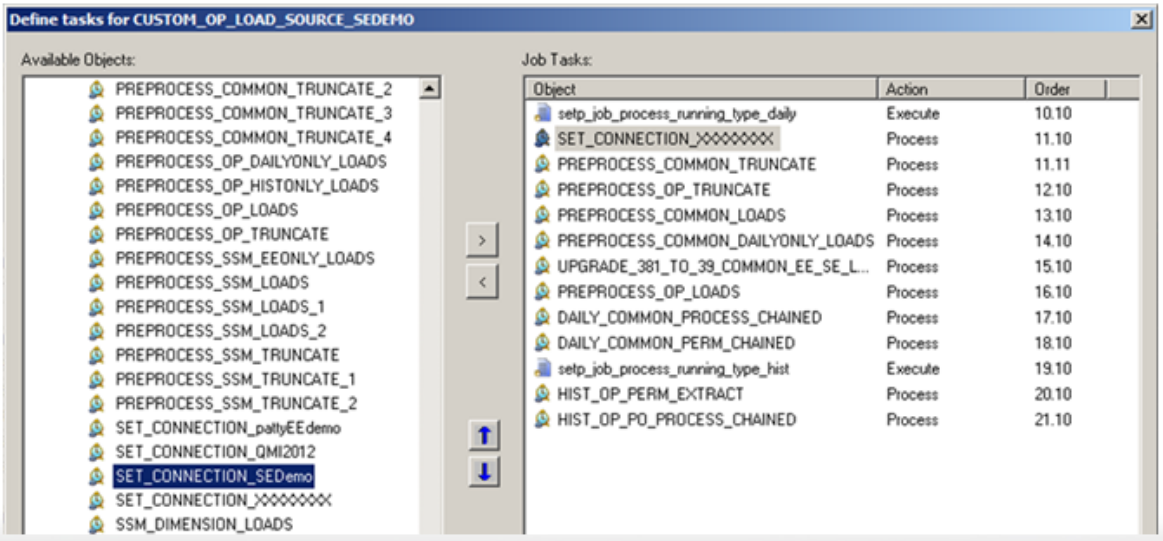
We add the *set_job_chaining_enabled_to_no* task to the beginning of the job because we don't want the jobs that are being run as tasks here to launch other chained jobs via the parameter list. Everything inside this custom job will just run in sequence. Make sure too that the order number is in sequence, i.e. don't have jobs with the same number otherwise they could both try to run at the same time.

DWD for Troubleshooting – Rerunning Jobs

Troubleshooting Tools

DWD for Troubleshooting – Rerunning Jobs

- Custom job methodology
- Steps 7 & 8 – continued from this



DWD for Troubleshooting – Rerunning Jobs

Troubleshooting Tools

DWD for Troubleshooting – Rerunning Jobs

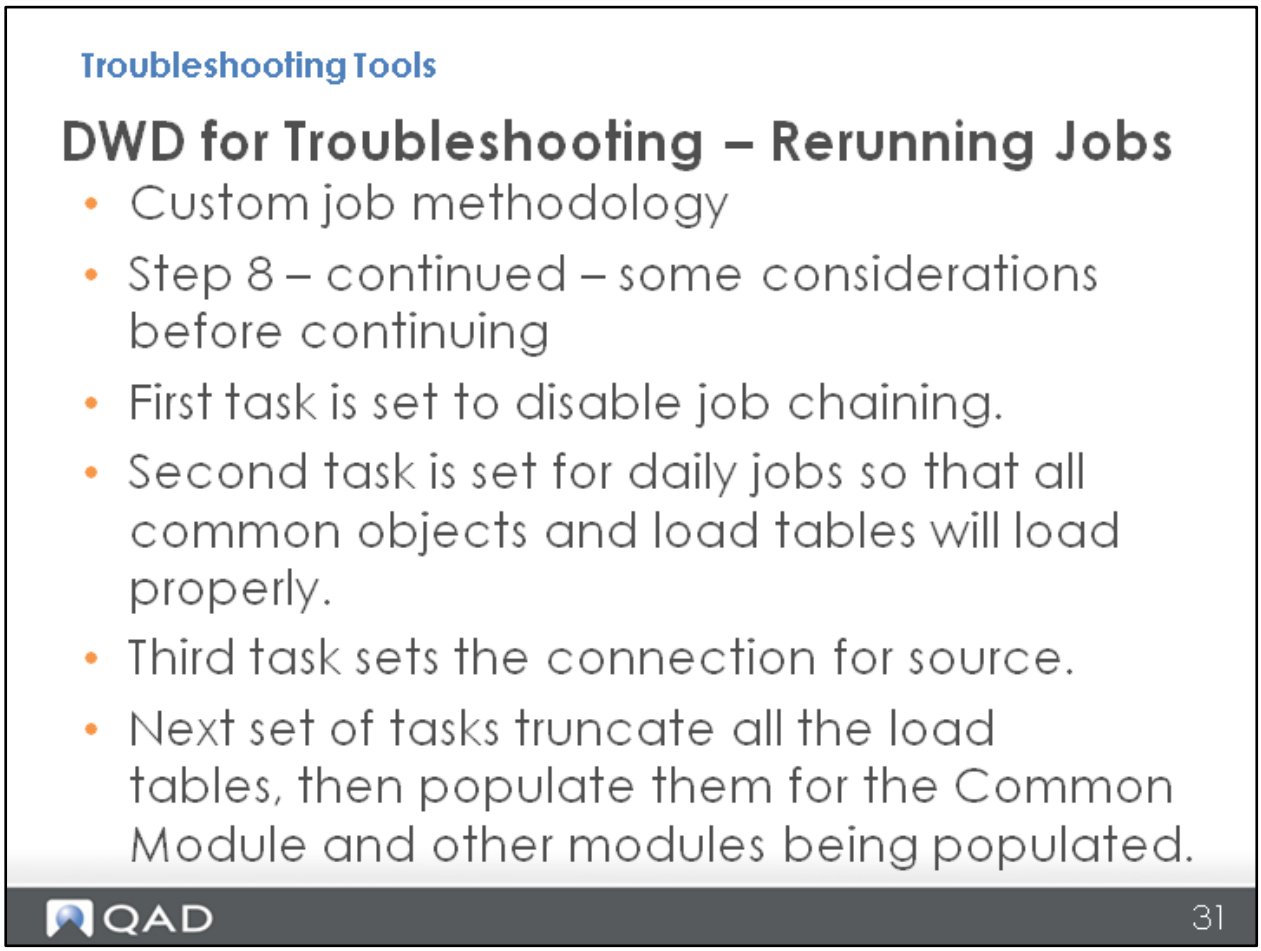
- Custom job methodology
- Steps 7 – continued to this

Job Tasks:

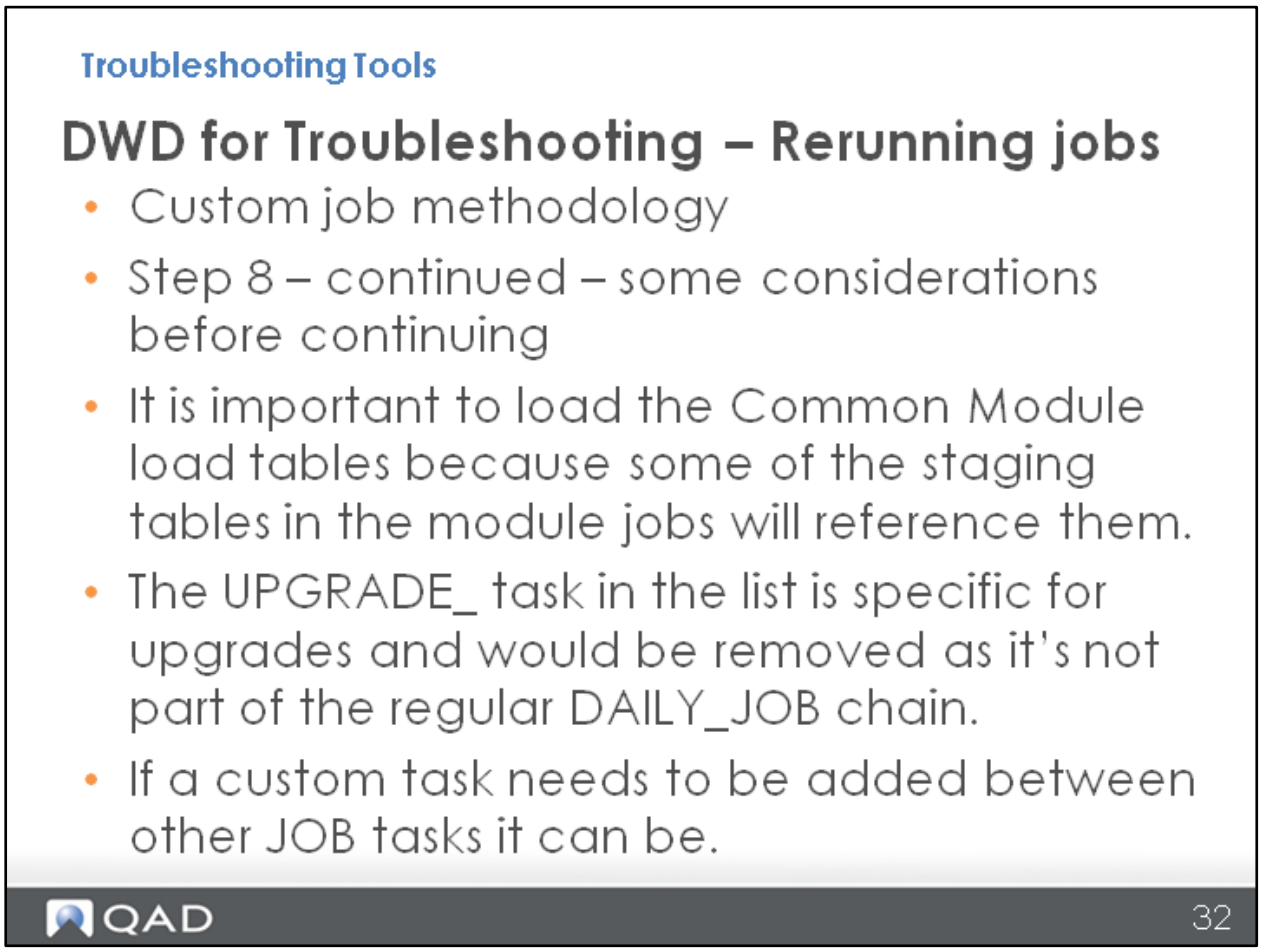
Object	Action	Order
set_job_chaining_enabled_to_no	Execute	10.9
setp_job_process_running_type_daily	Execute	10.10
SET_CONNECTION_SEDemo	Process	11.10
PREPROCESS_COMMON_TRUNCATE	Process	11.11
PREPROCESS_OP_TRUNCATE	Process	12.10
PREPROCESS_COMMON_LOADS	Process	13.10
PREPROCESS_COMMON_DAILYONLY_LOADS	Process	14.10
PREPROCESS_OP_LOADS	Process	16.10
DAILY_COMMON_PROCESS_CHAINED	Process	17.10
DAILY_COMMON_PERM_CHAINED	Process	18.10
setp_job_process_running_type_hist	Execute	19.10
HIST_OP_PERM_EXTRACT	Process	20.10
HIST_OP_PO_PROCESS_CHAINED	Process	21.10

QAD 30

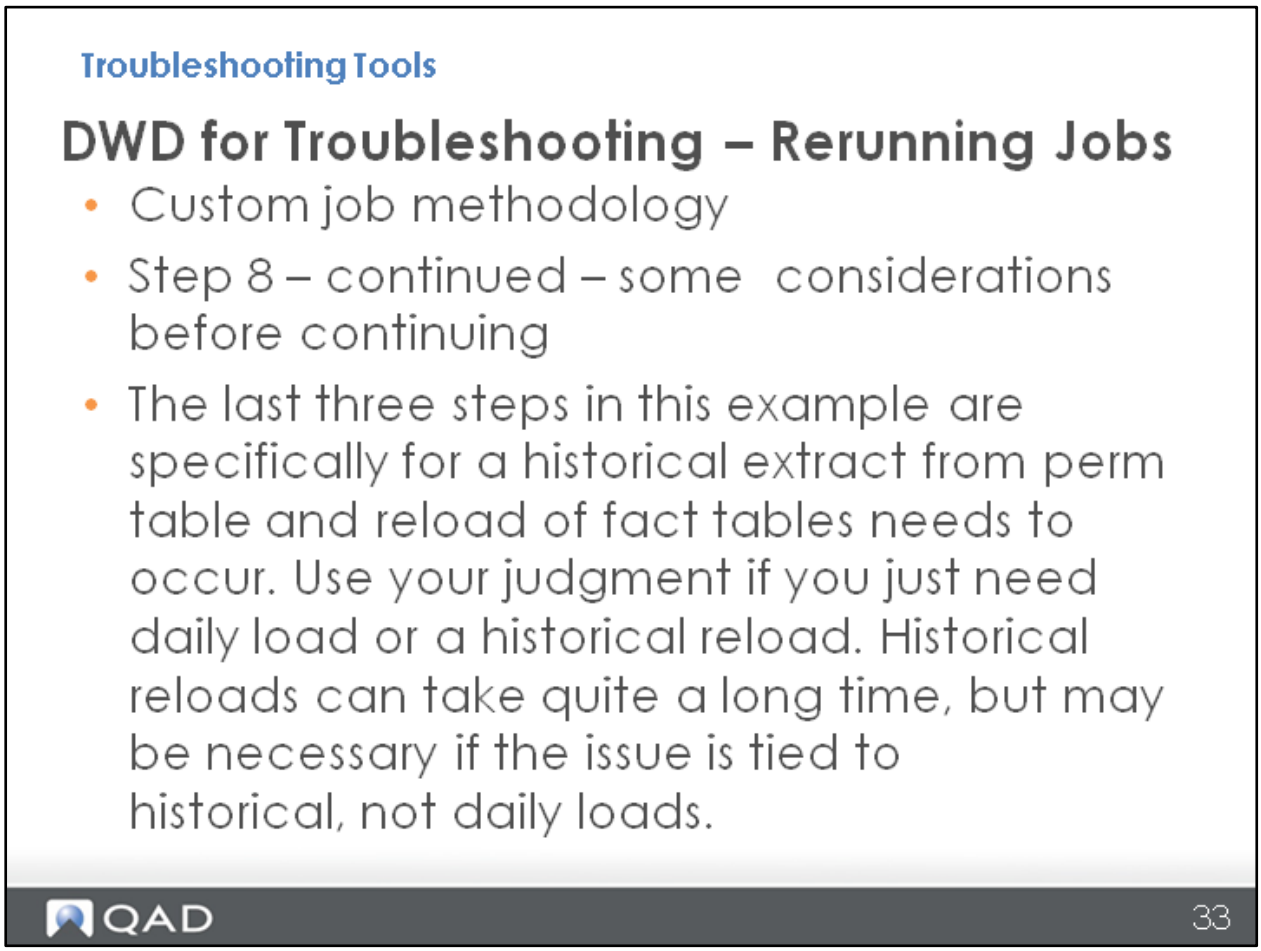
DWD for Troubleshooting – Rerunning Jobs



DWD for Troubleshooting – Rerunning jobs



DWD for Troubleshooting – Rerunning Jobs



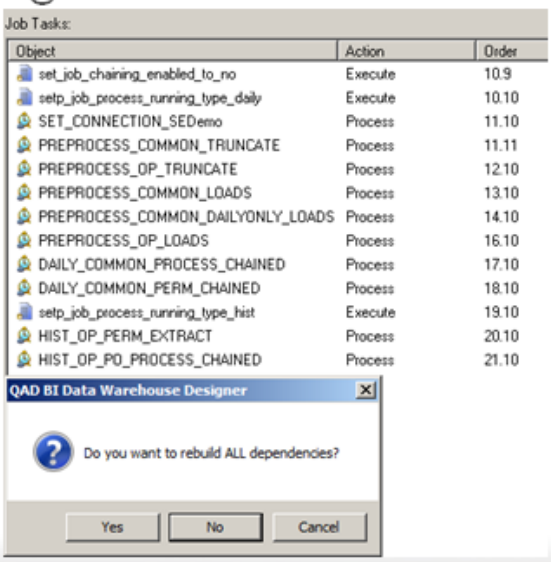
If a straight forward DAILY_PERM_EXTRACT and DAILY_PROCESS is desired, remove the *setp_job_process_running_type_to_hist* task from the list. This task flags the jobs to know they are running in historical mode which we wouldn't want for daily. The daily parameter setting was flagged in the second task and would still apply at this portion of the staging tables.

DWD for Troubleshooting – Rerunning Jobs

Troubleshooting Tools

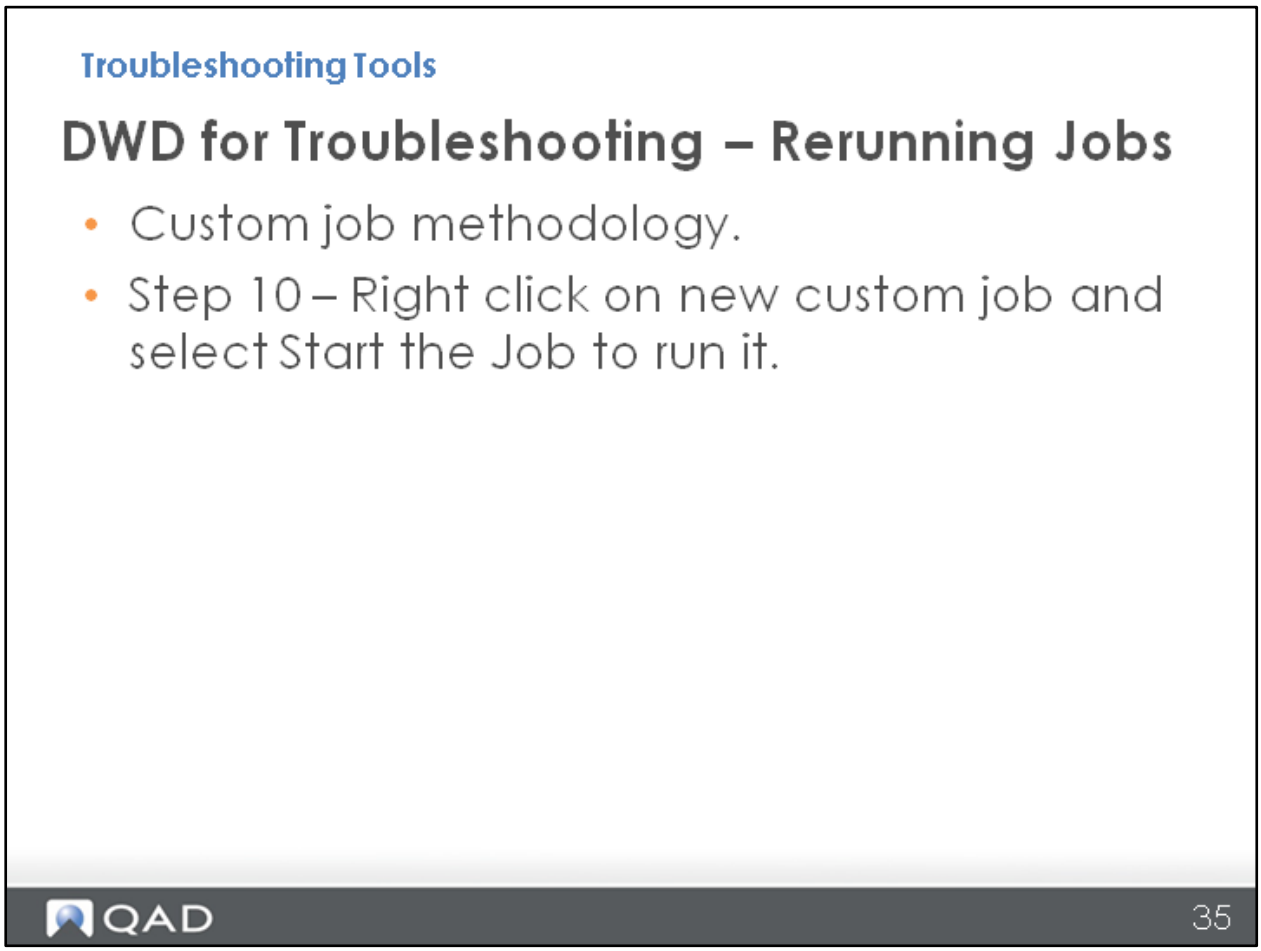
DWD for Troubleshooting – Rerunning Jobs

- Custom job methodology
- Step 9 – After putting together the custom job/task list, rebuild ALL dependencies.

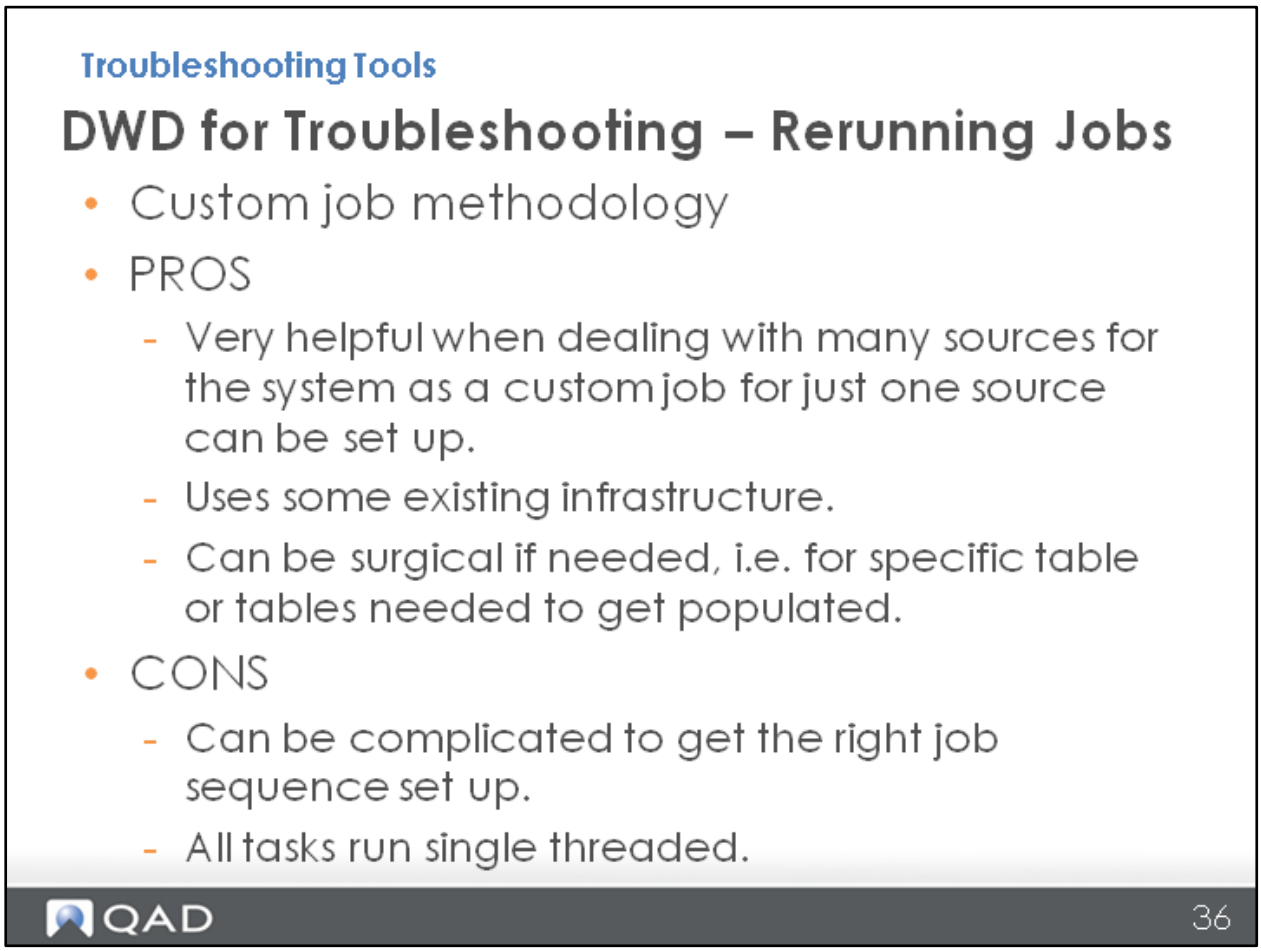


QAD 34

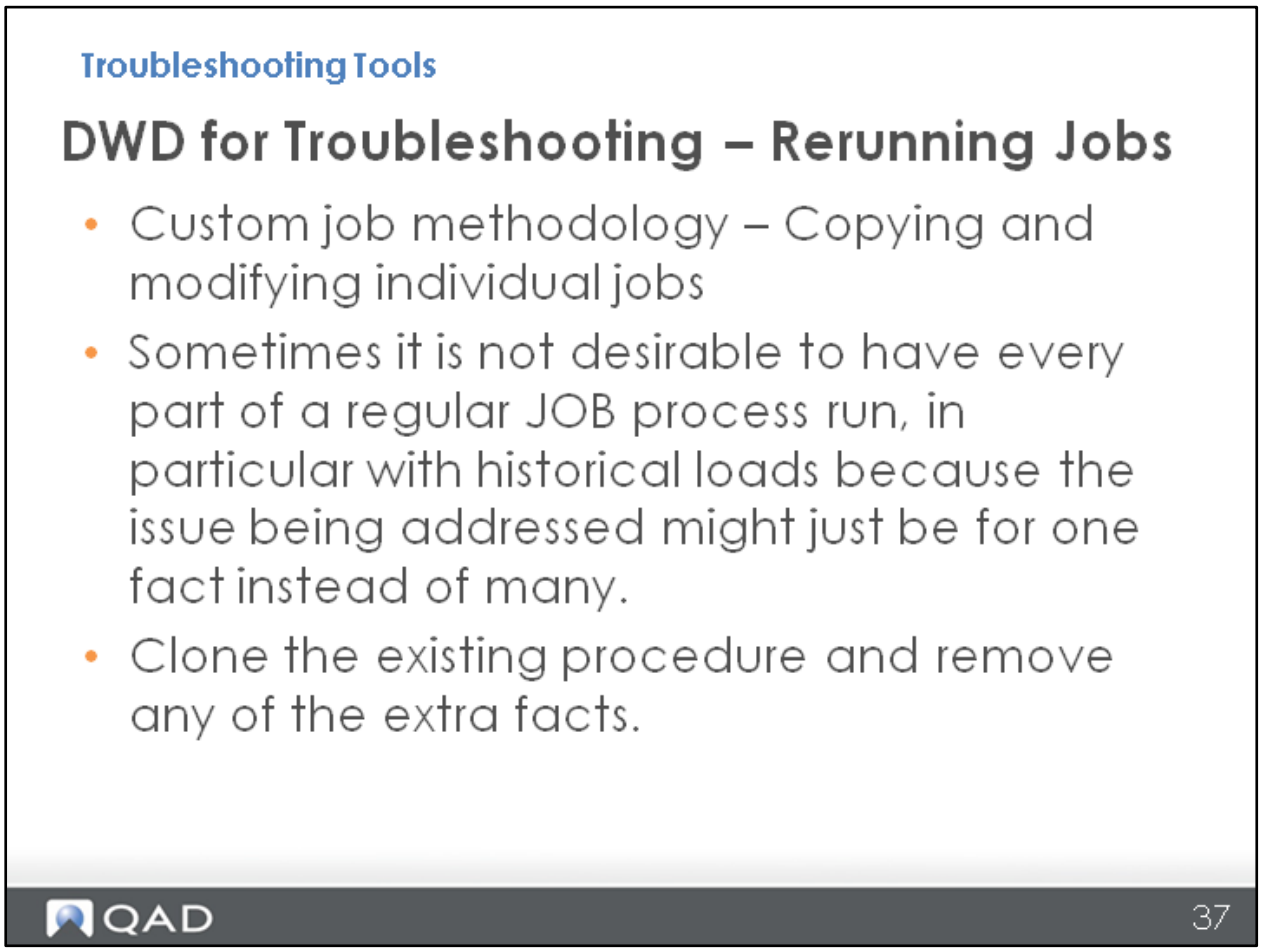
DWD for Troubleshooting – Rerunning Jobs



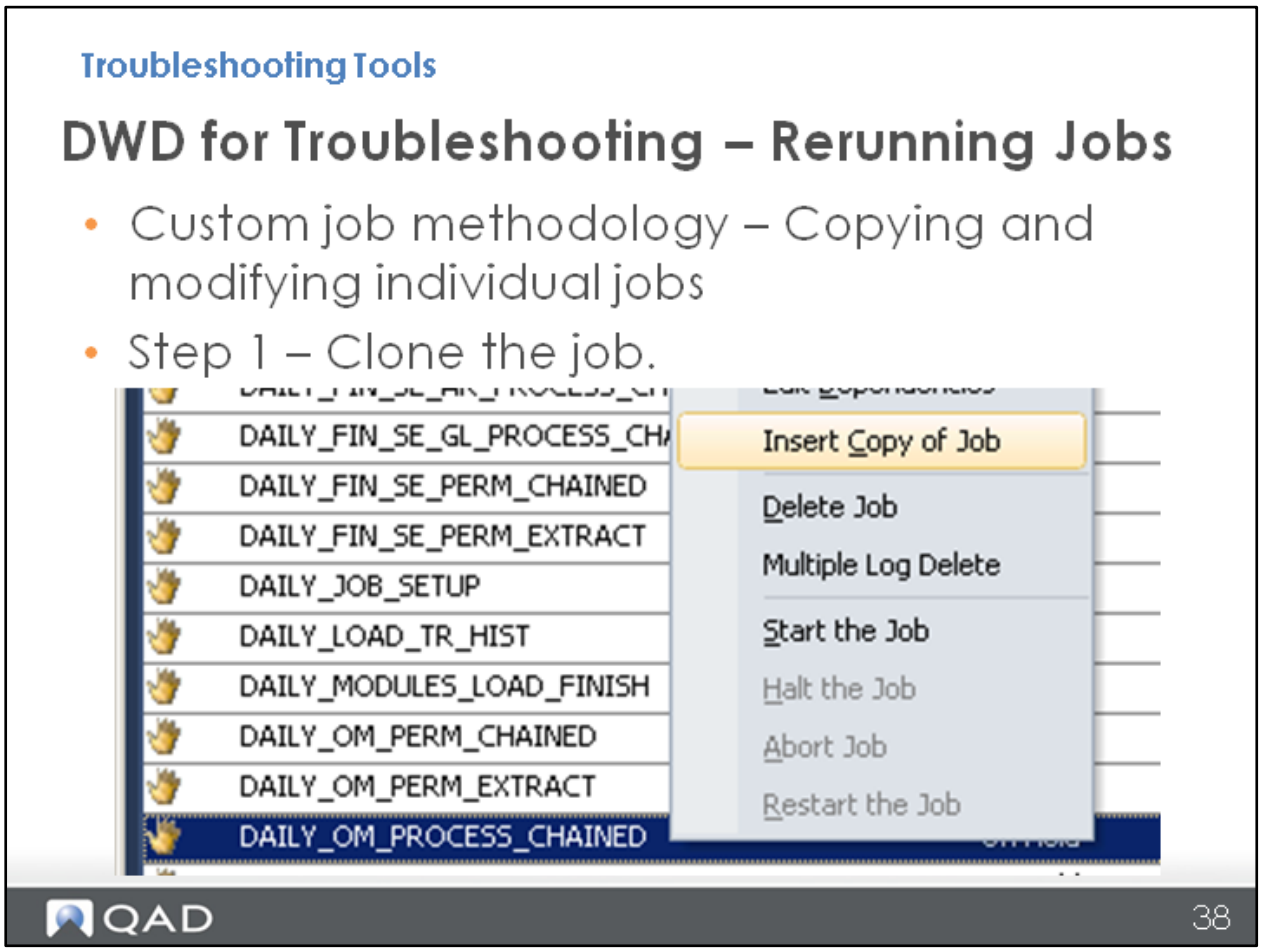
DWD for Troubleshooting – Rerunning Jobs



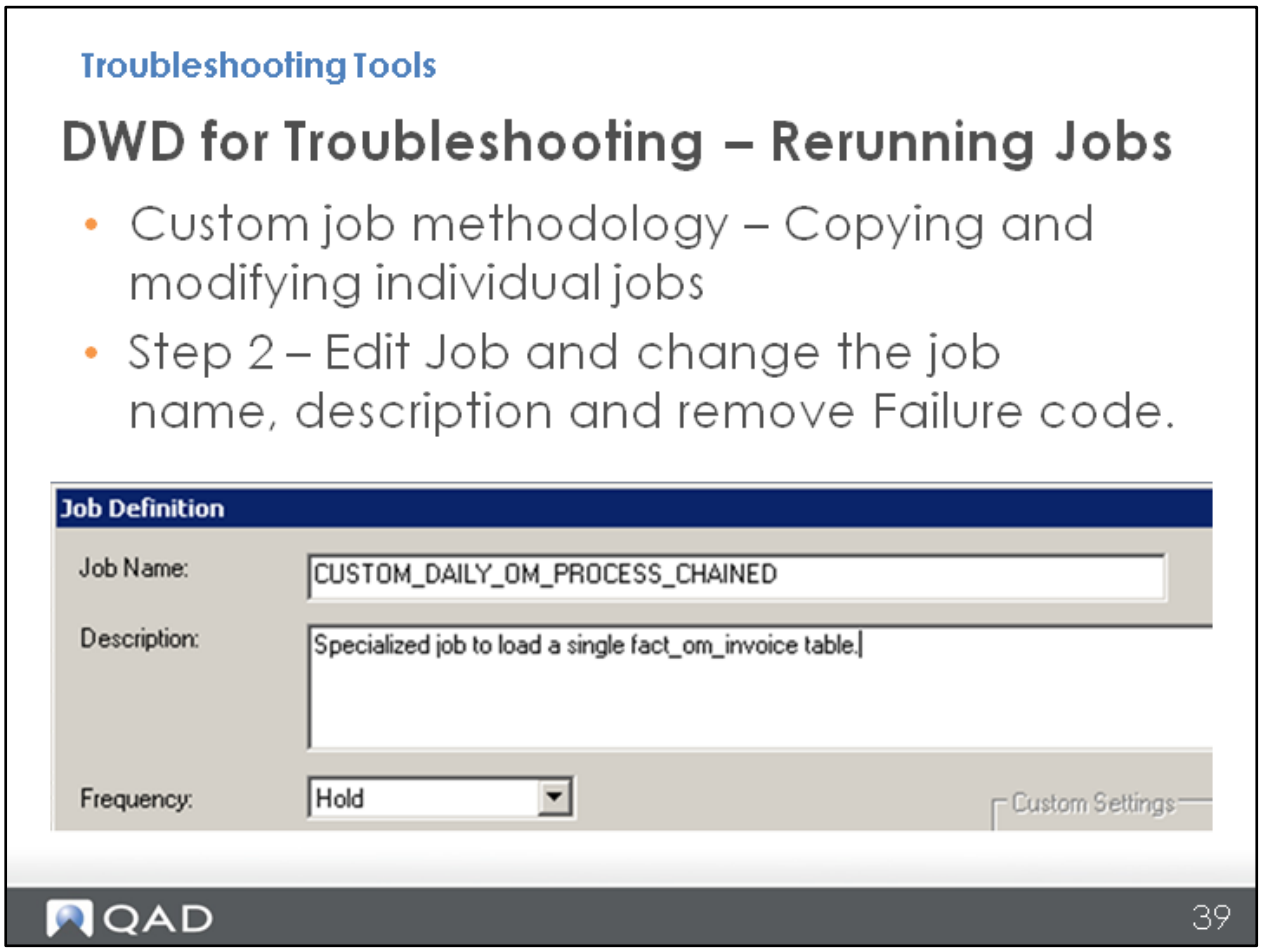
DWD for Troubleshooting – Rerunning Jobs



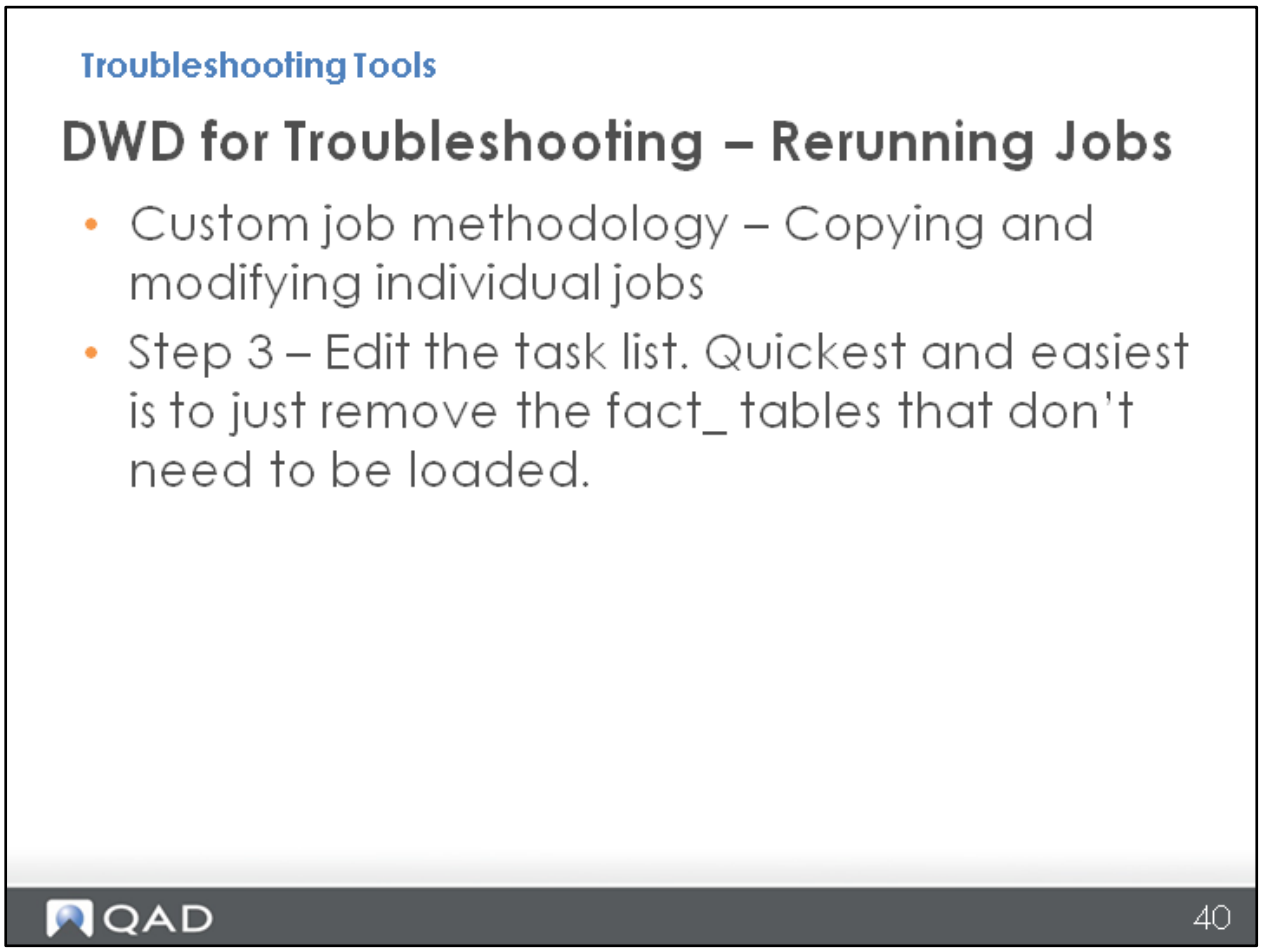
DWD for Troubleshooting – Rerunning Jobs



DWD for Troubleshooting – Rerunning Jobs



DWD for Troubleshooting – Rerunning Jobs



DWD for Troubleshooting – Rerunning Jobs

Troubleshooting Tools

DWD for Troubleshooting – Rerunning Jobs

- Custom job methodology – Copying and modifying individual jobs.

Loading only fact_om_invoice

Before:

Job Tasks:

Object	Action	Order
stage_om_booking2	Process	88.10
stage_om_booking3	Process	89.10
stage_om_booking4	Process	90.10
stage_om_booking_previous	Process	91.10
stage_om_booking	Process	92.10
fact_om_booking	Process	93.10
fact_om_booking	Custom	94.10
stage_om_invoice_transaction1	Process	95.10
stage_om_ar_master_extract	Process	96.10
stage_om_invoice_transaction	Process	97.10
stage_om_invoice_scheduled_transaction	Process	98.10
stage_om_invoice1	Process	99.10
stage_om_invoice2	Process	100.10
stage_om_invoice3	Process	101.10
stage_om_invoice	Process	102.10
fact_om_invoice	Process	103.10
stage_om_shipment_transaction	Process	104.10
stage_om_shipment1	Process	105.10
stage_om_shipment2	Process	106.10
stage_om_shipment3	Process	107.10
stage_om_shipment	Process	108.10
fact_om_shipment	Process	109.10
setp_sales_booking_date	Execute	110.10
setp_job_sales_process_end	Execute	111.10
submit_next_chained_job	Execute	112.10

QAD 41

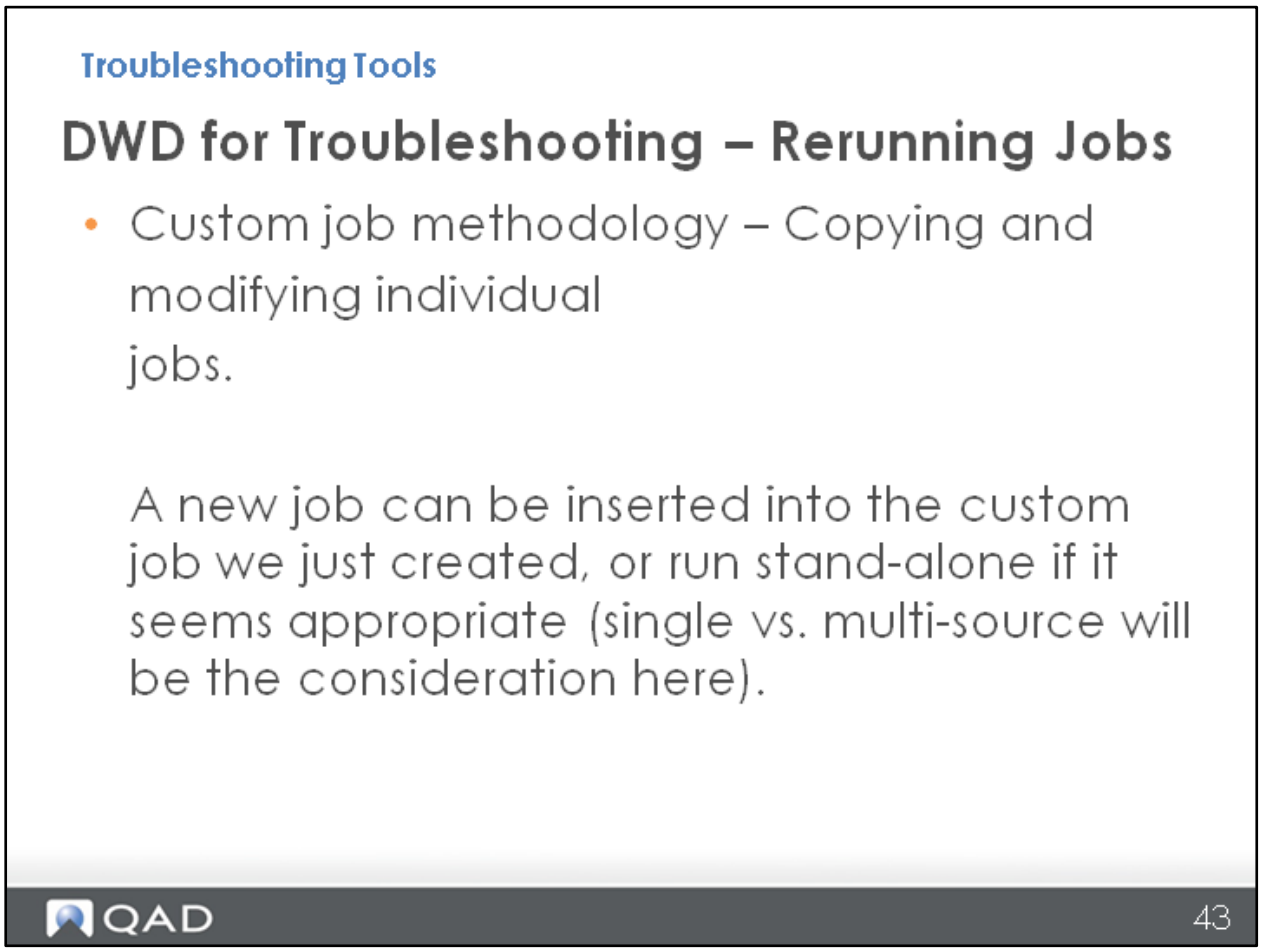
There are various fact tables in the job.

DWD for Troubleshooting – Rerunning Jobs



The fact_om_shipment and shipment stage tables have been removed. Other fact_ tables have been removed and only fact_om_invoice remains. That leaves lots of unnecessary stage tables for other fact tables that are not in the scheduler any more, but ensures that proper stage tables do get populated. This is a quick way to build a custom job without trying to figure out which stage tables are safe to remove and which are not.

DWD for Troubleshooting – Rerunning Jobs

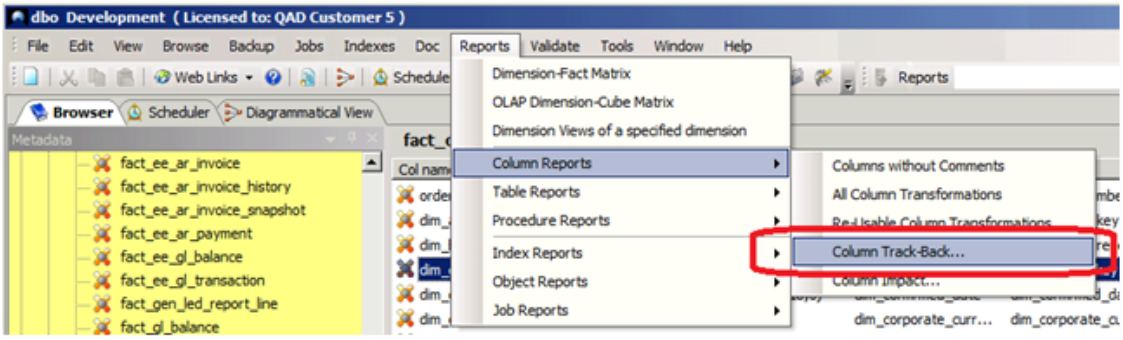


DWD – Features – Column Track Back

DWD – Features – Column Track Back

Very useful reporting option. Displays all tables and columns backwards from the table and column in question to its source.

1. From Builder/Browser tab, Go to Reports/Column Reports/Column Track-Back

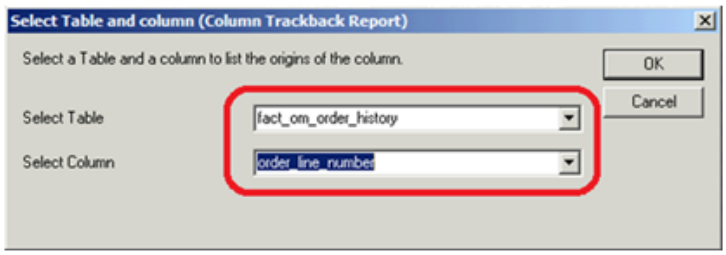


QAD 44

DWD – Features – Column Track Back

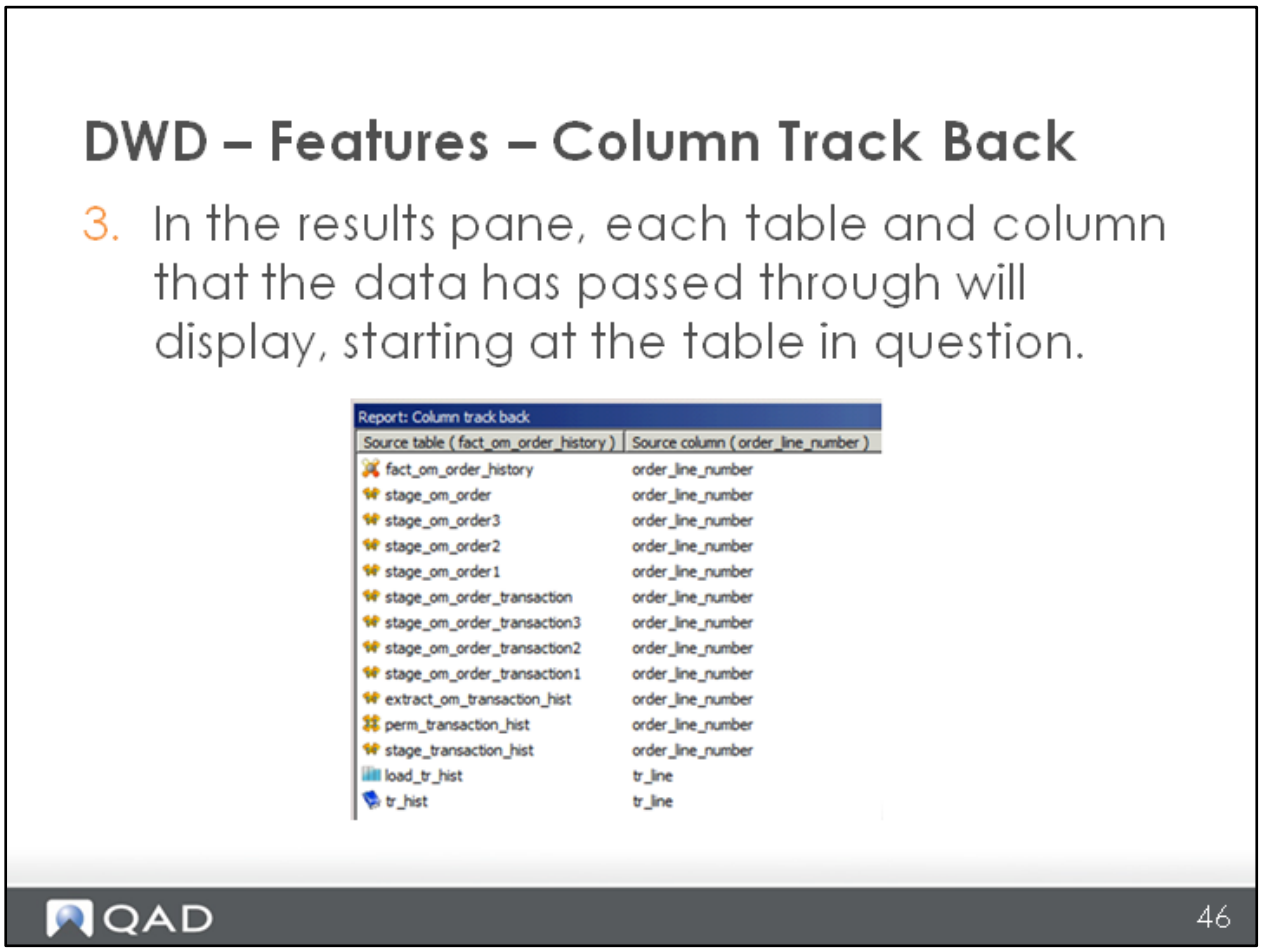
DWD – Features – Column Track Back

2. From the Select Table and Select Column window, pick the table and the column to be tracked back on and click OK.



QAD 45

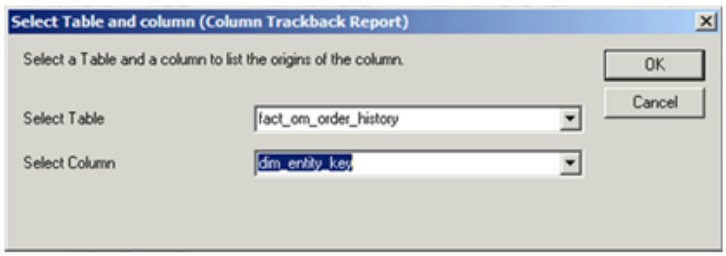
DWD – Features – Column Track Back

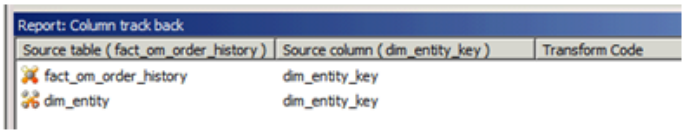


DWD – Features – Column Track Back

DWD – Features – Column Track Back

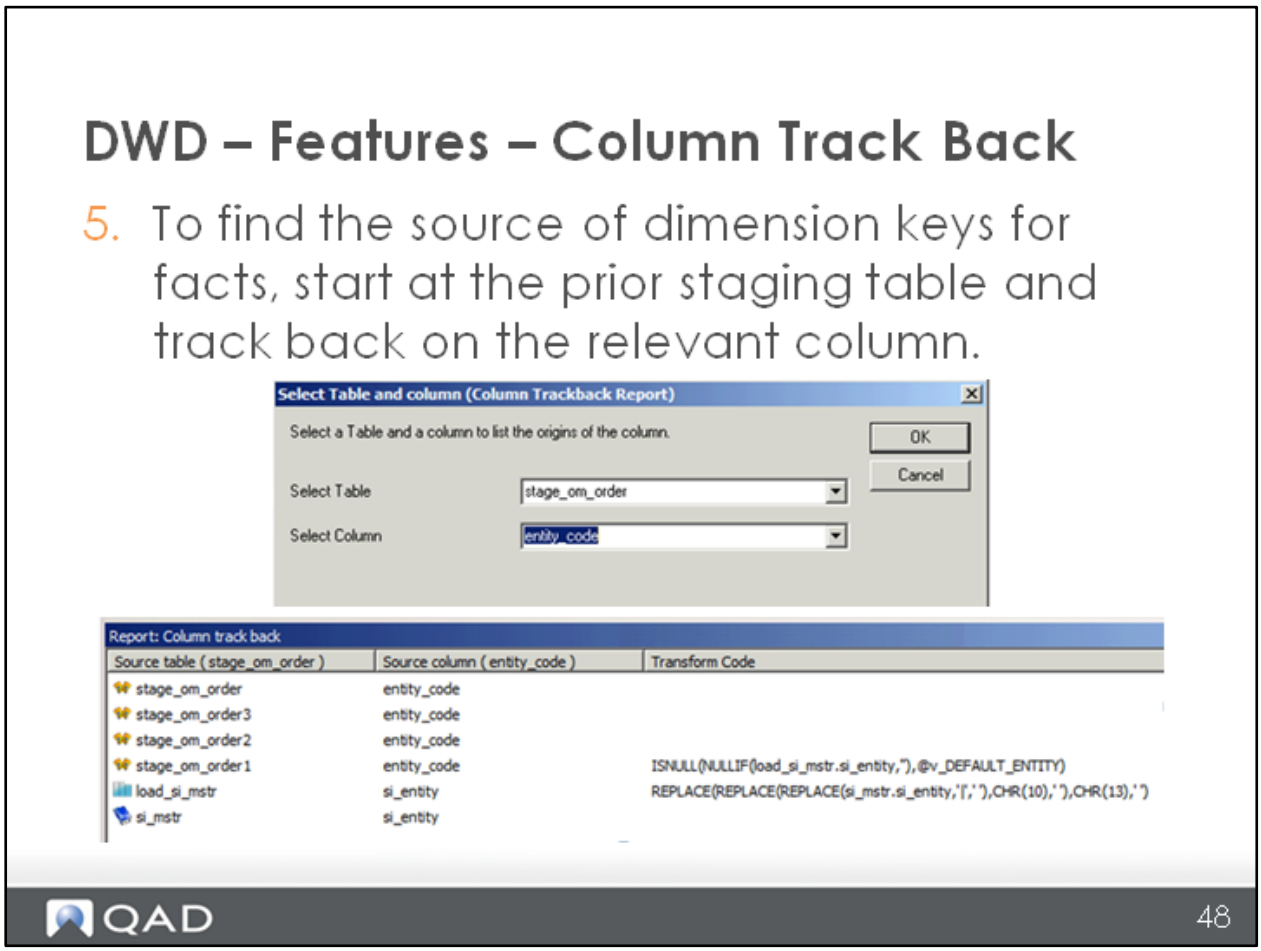
4. However, an attempt to track back on a dimension key, will only bring data back for that dimension.



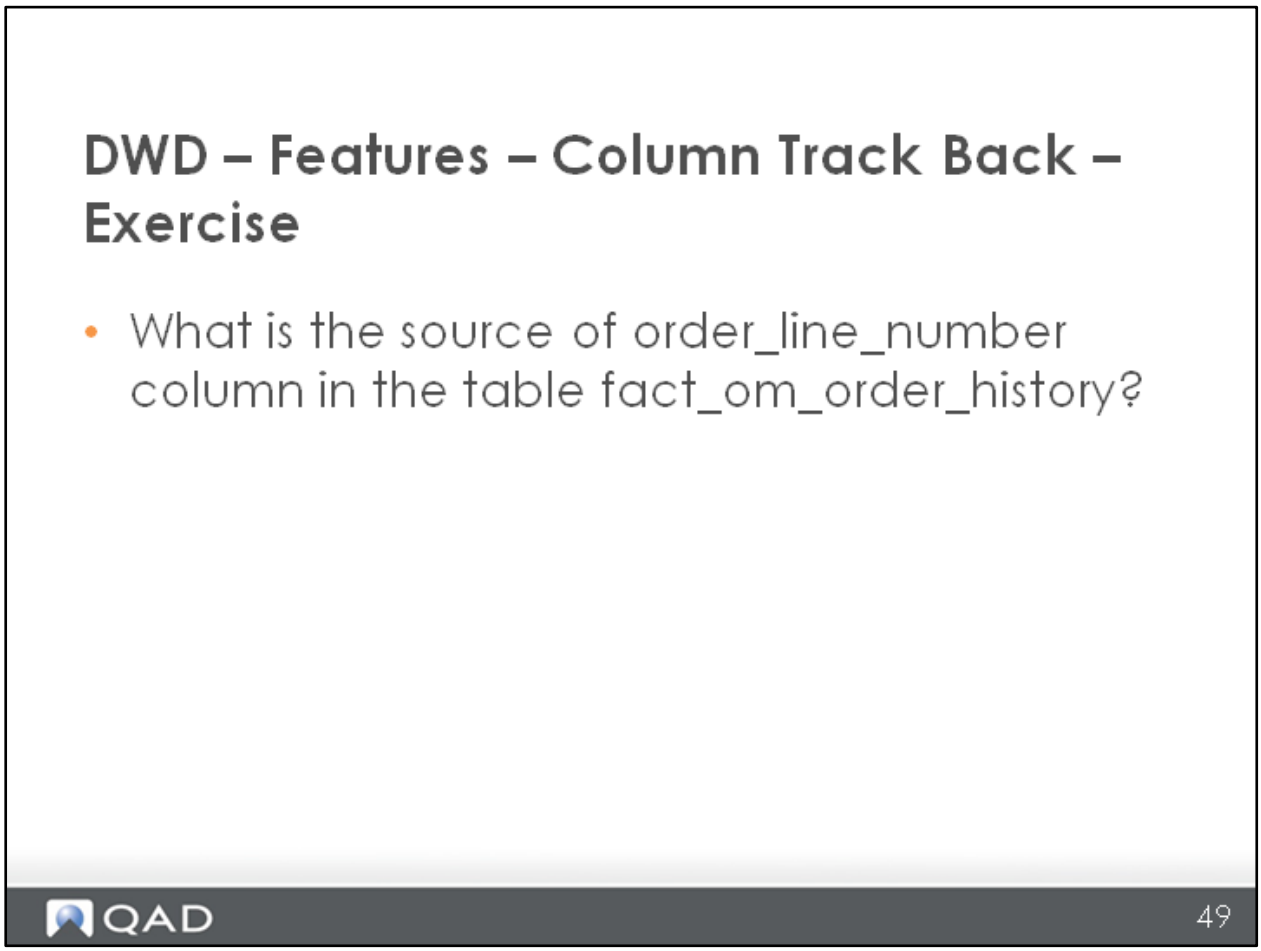


QAD 47

DWD – Features – Column Track Back



DWD – Features – Column Track Back – Exercise

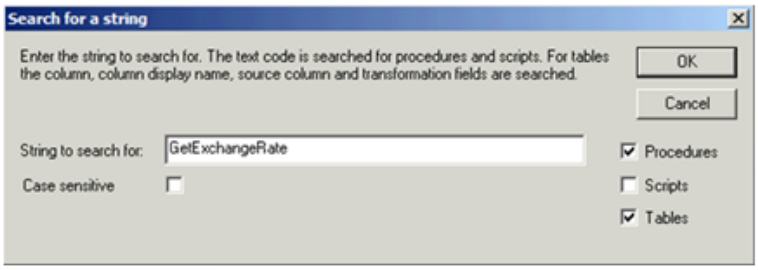


DWD – Features – Search For String

DWD – Features – Search For String

If a certain column or procedure or function or any type of string is sought in a procedure or in a table, this can easily be achieved using the **Search for String** option under Tools.

1. Click the check boxes for where the string will be searched.

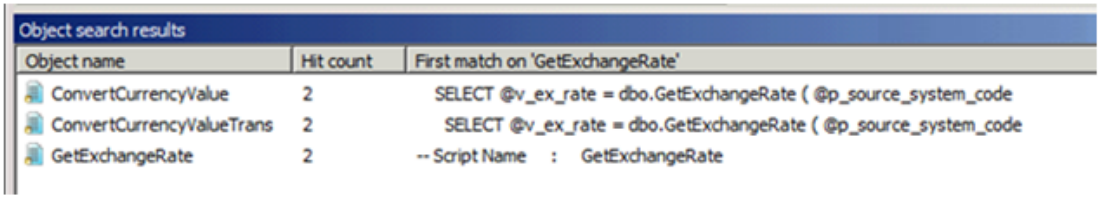


QAD 50

DWD – Features – Search For String

DWD – Features – Search For String

2. The relevant procedures and tables are displayed with the number of times that the string appears in that procedure/table.



QAD 51

DWD – Features – Column Transformations

DWD – Features – Column Transformations

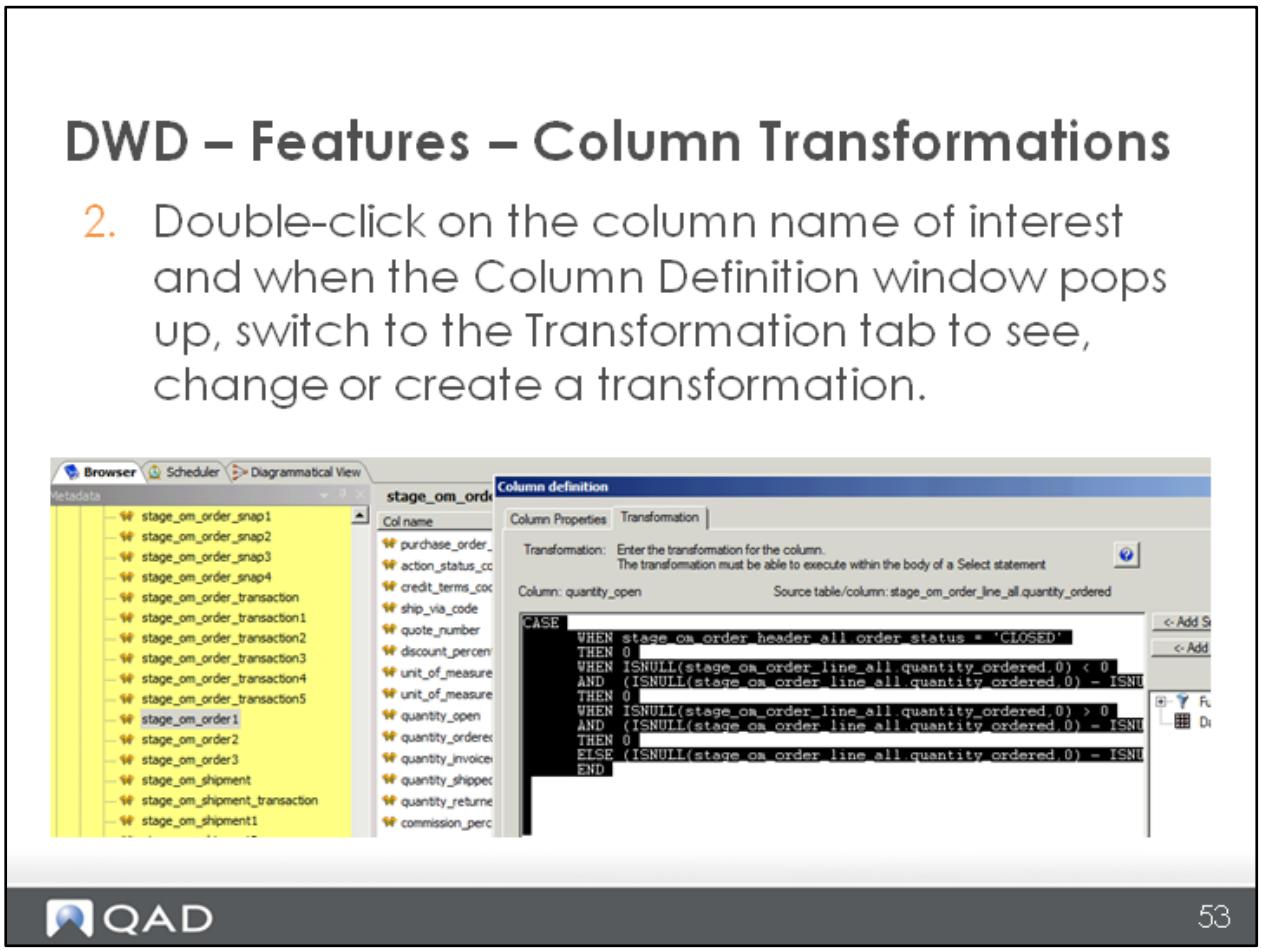
In many instances, columns are transformed to display some other value than the column being referenced.

1. In the Builder tab under the Objects pane, click on a table name to see the columns from the table.



QAD 52

DWD – Features – Column Transformations

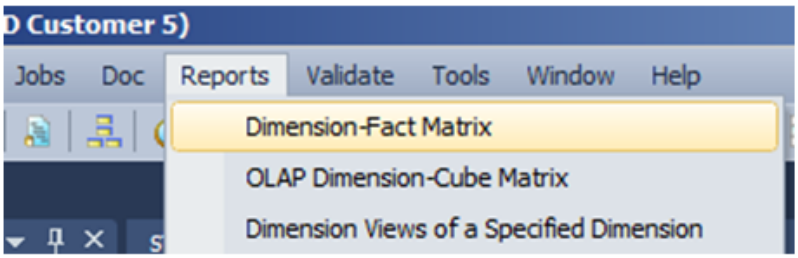


Use stage_om_order1 table and quantity_open column as a good example.

DWD – Features – Fact-Dimension Matrix

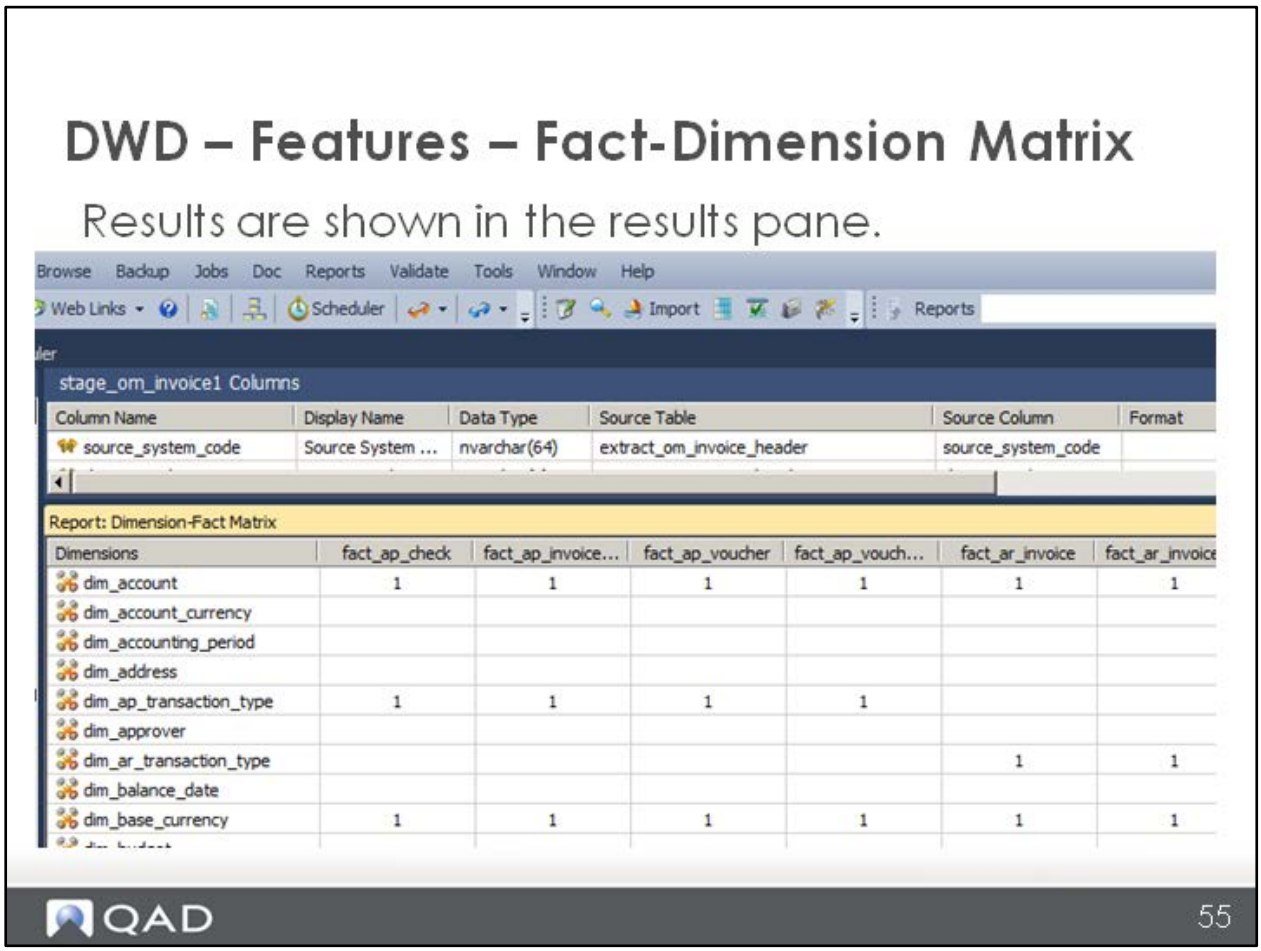
DWD – Features – Fact-Dimension Matrix

- When trying to see which facts use which dimensions or which dimensions are used by which facts, the quickest way to see all of them at once is the fact/dimension grid.
- Go to Reports/Dimension-Fact Matrix

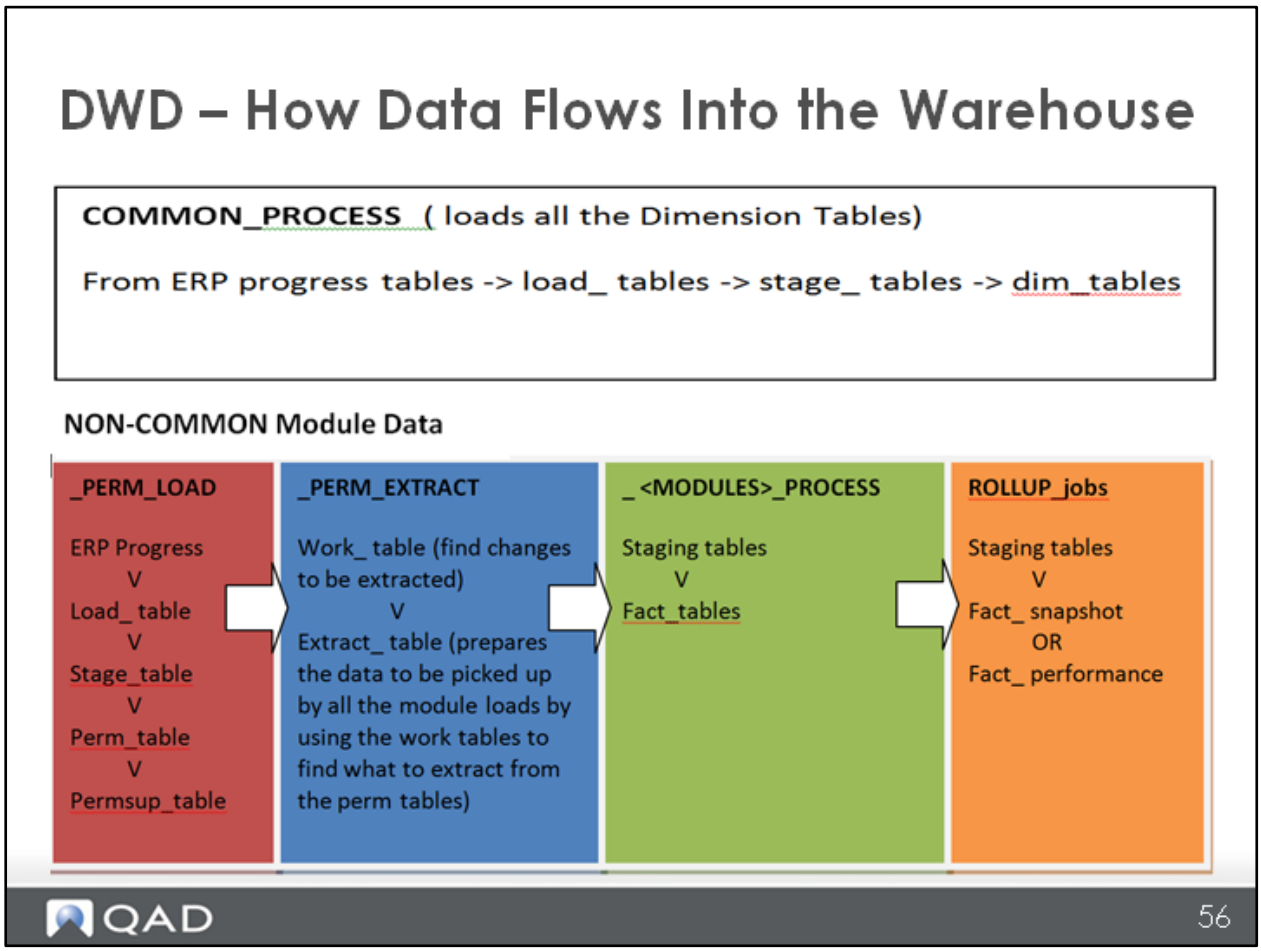


QAD 54

DWD – Features – Fact-Dimension Matrix



DWD – How Data Flows Into the Warehouse



Common Process plays a VERY important role of building the dimension tables which have all the keys, the fact tables will refer to when they are built.

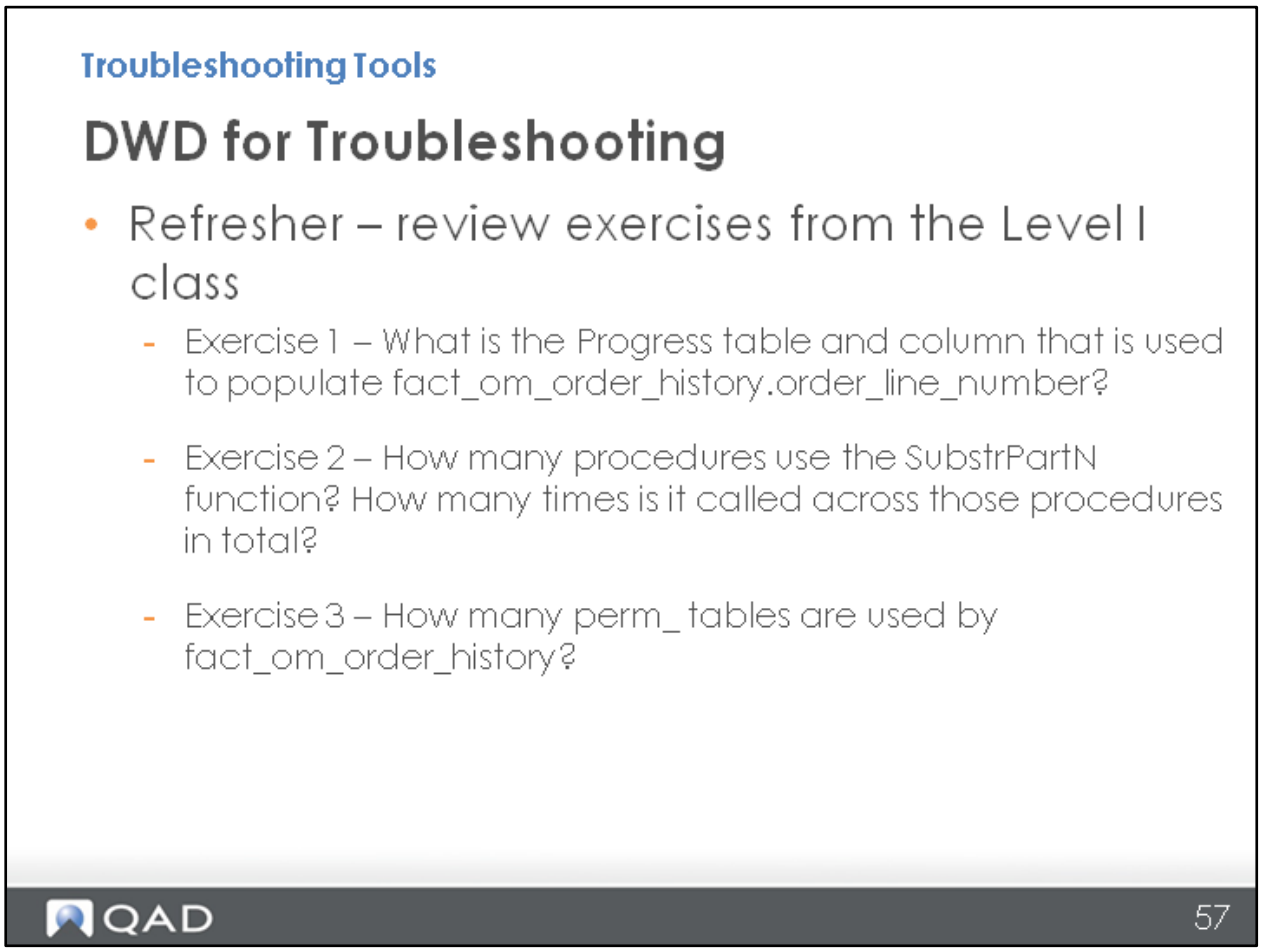
Perm_load job loads any tables that store permanent data (large tables that shouldn't change once created).

Perm_extract job "primes the pump" for the module loads by looking for changes to perm_tables and preparing the data to be loaded in extract tables.

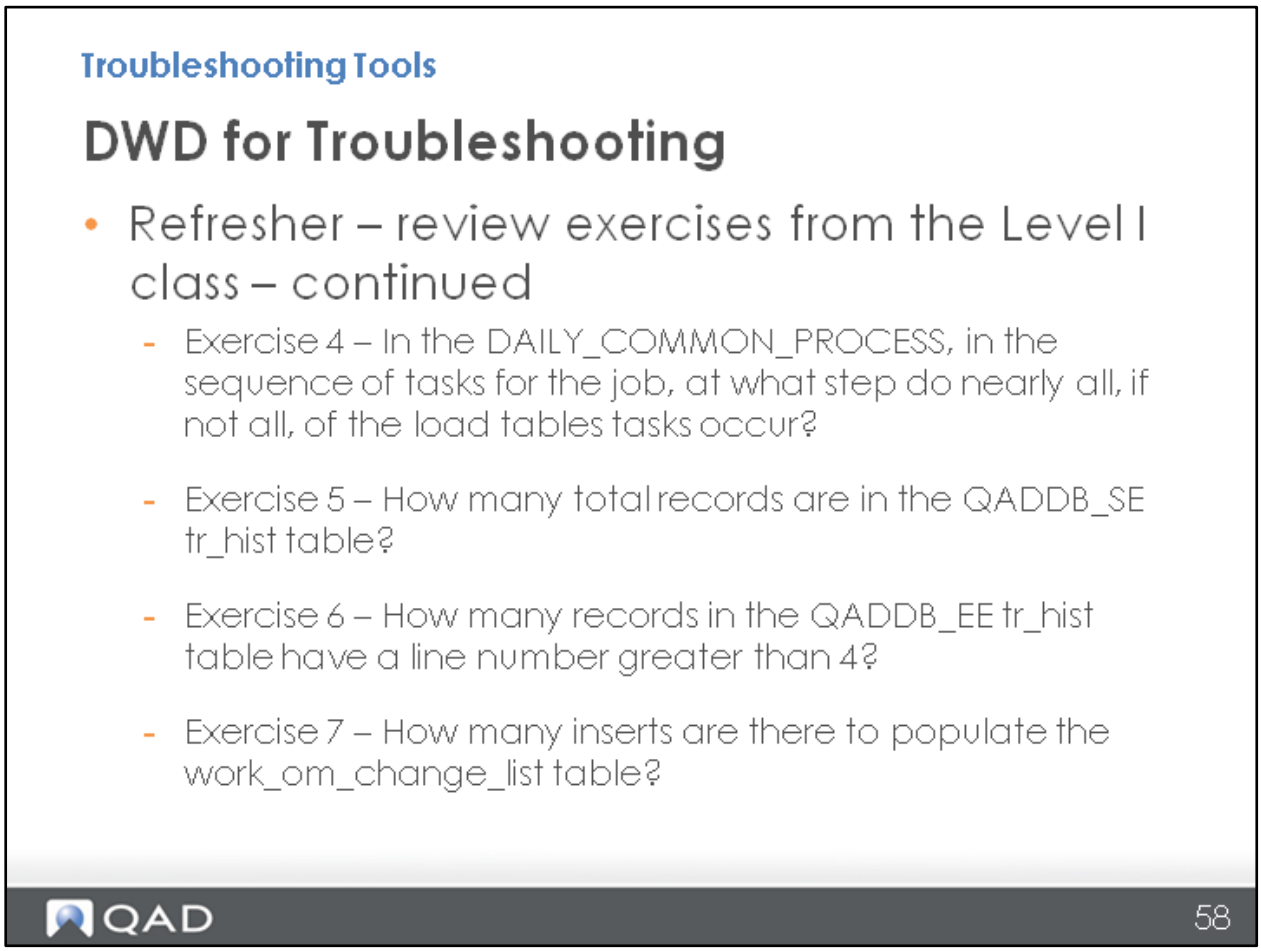
The various modules process jobs load for their respective modules.

The Rollup jobs populate the snapshots and the performance tables.

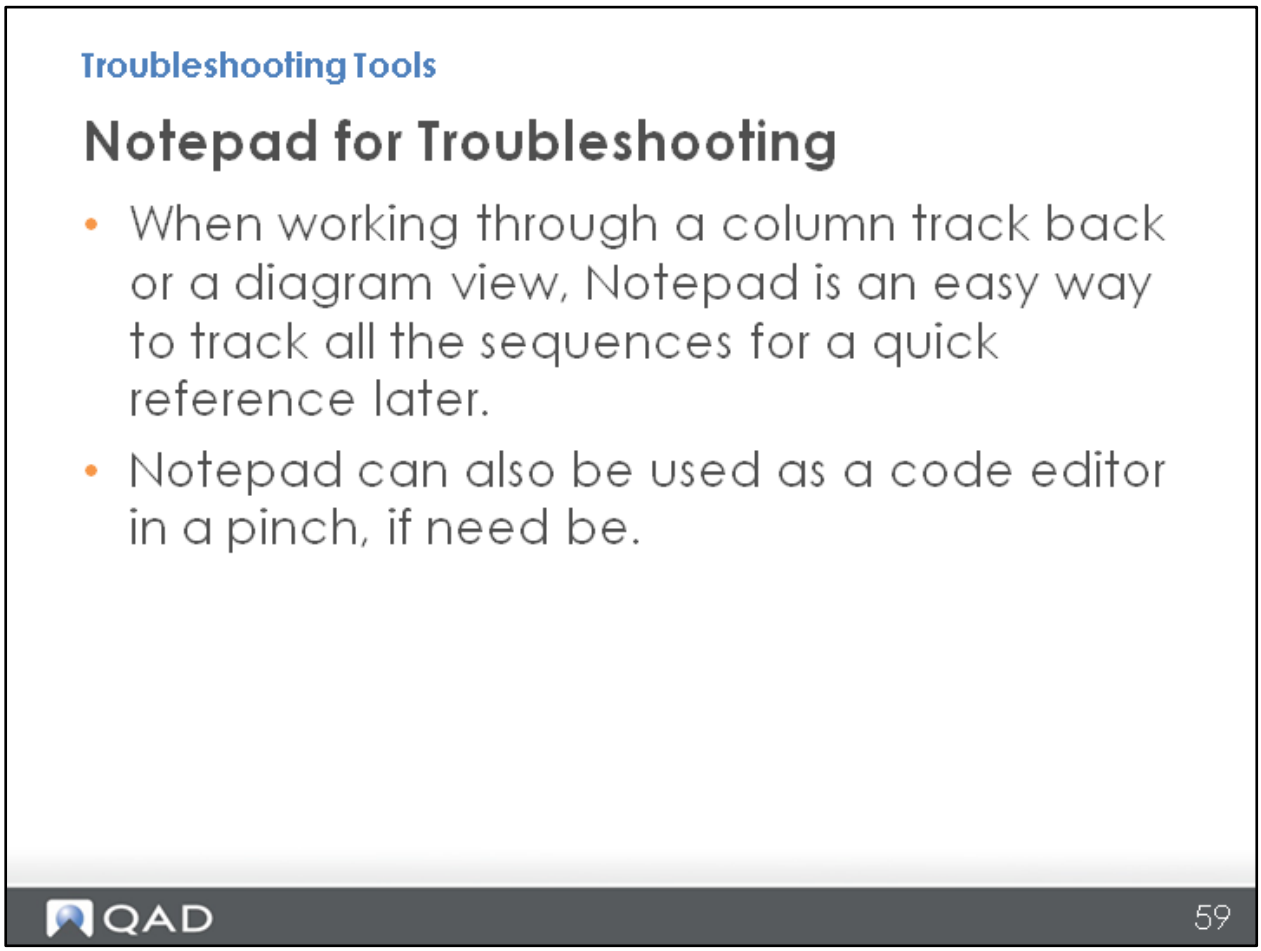
DWD for Troubleshooting



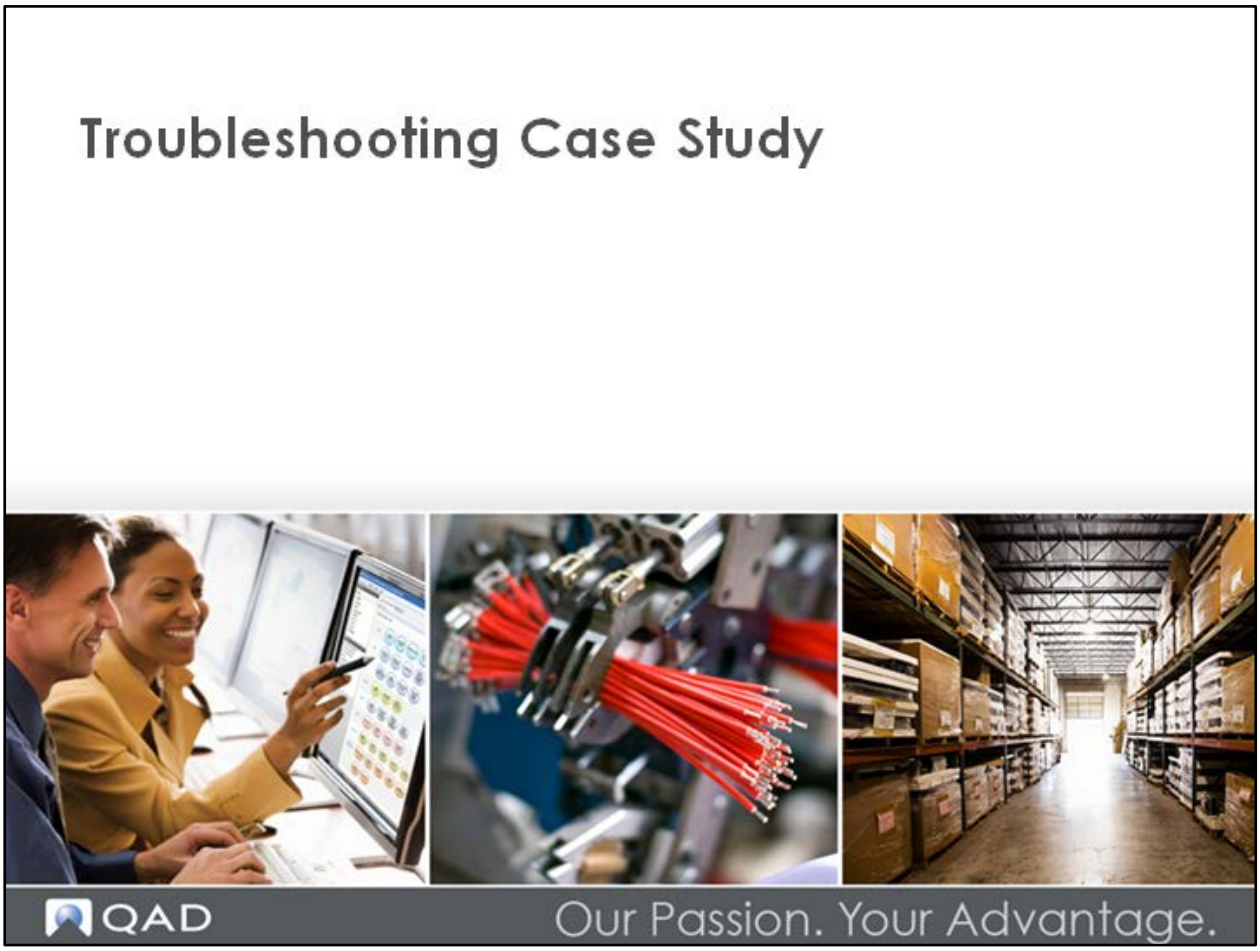
DWD for Troubleshooting



Notepad for Troubleshooting



Troubleshooting Case Study

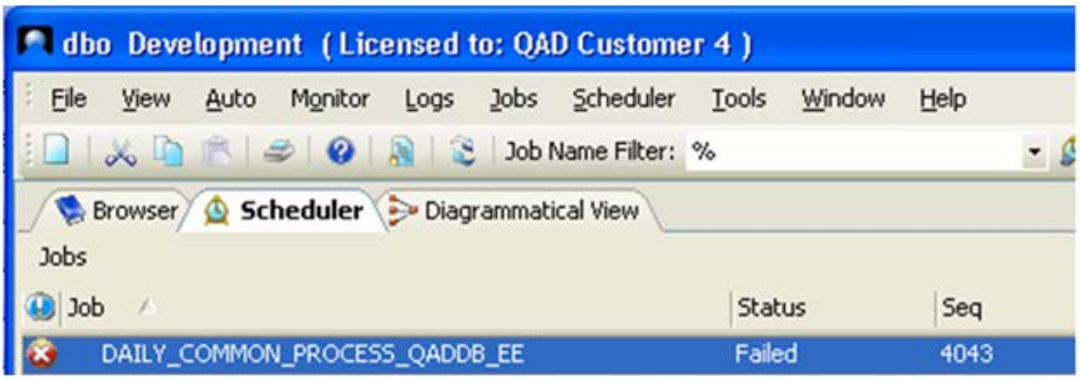


Troubleshooting – Case 1

Case Study

Troubleshooting – Case 1

- Customer keeps getting a job failure message in the scheduler on DAILY_COMMON_PROCESS_QADDB_EE and nothing else runs after it.



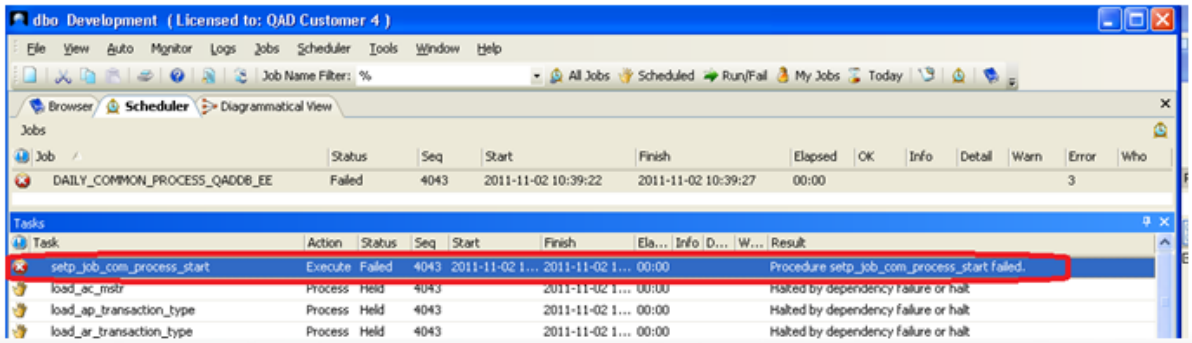
QAD 61

Troubleshooting – Case 1

Case Study

Troubleshooting – Case 1

- To troubleshoot failed jobs
- Step 1 – Open up the failed job by double clicking on it and see which task the job failed on. In this case, it's the first task setp_job_com_process_start.



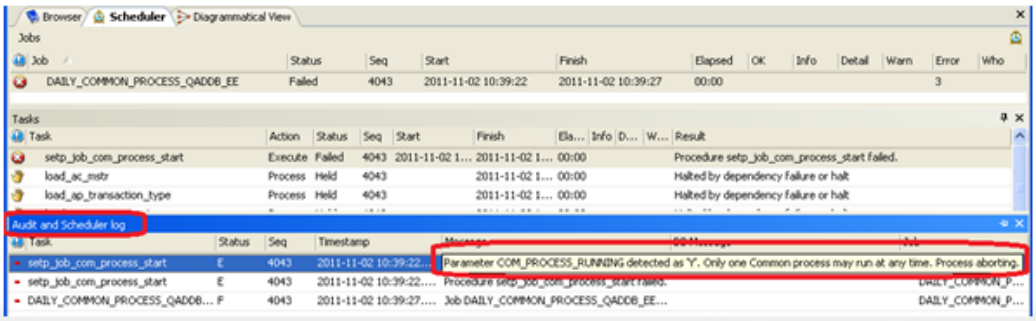
QAD 62

Troubleshooting – Case 1

Case Study

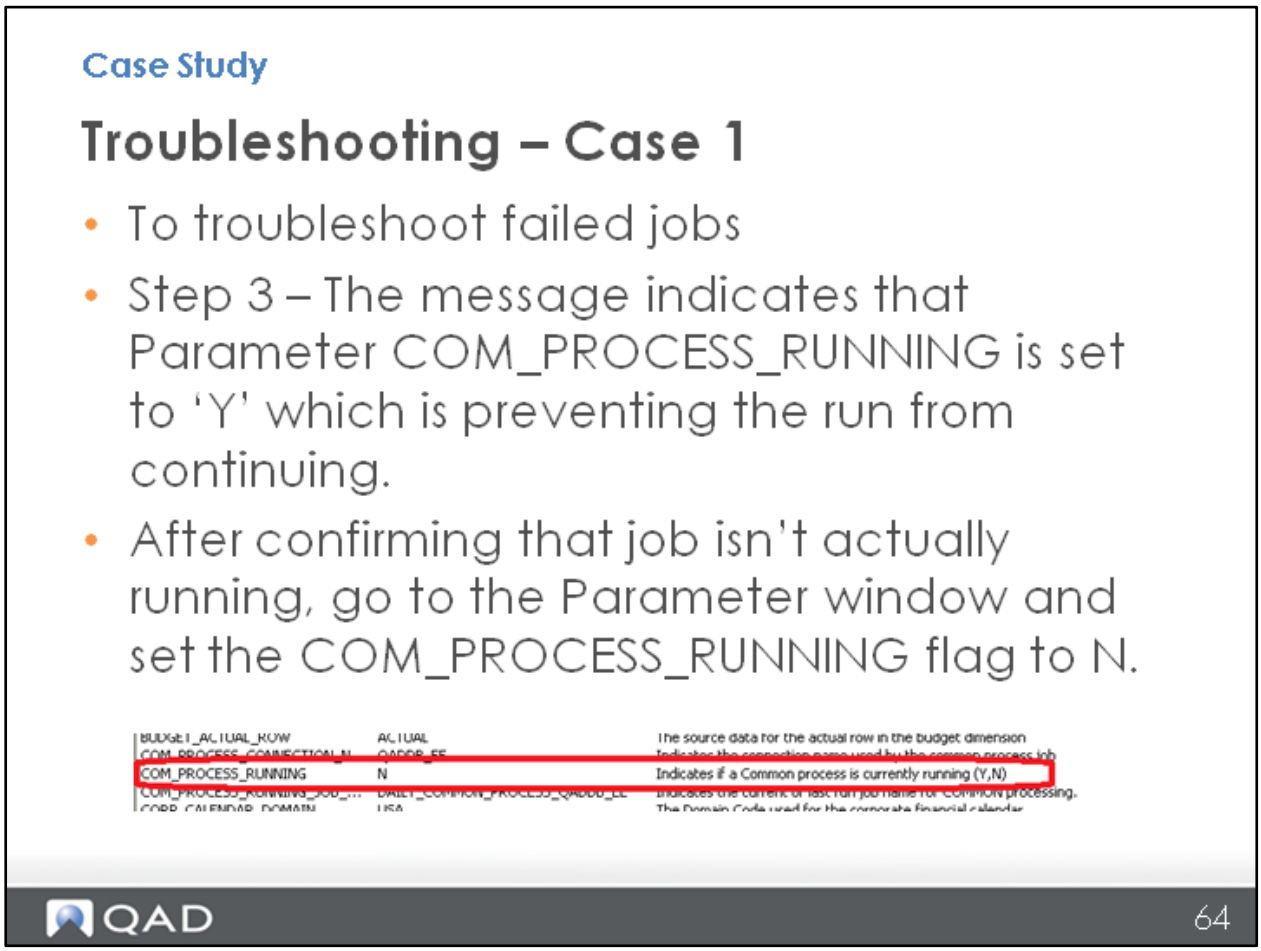
Troubleshooting – Case 1

- To troubleshoot failed jobs
- Step 2 – Double click on the failed task to see the audit log message affiliated with the failure. When it opens, hover over the message to see the full message.



QAD 63

Troubleshooting – Case 1



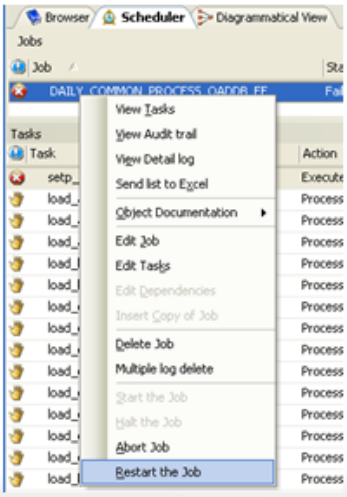
The setp_...._start jobs always check to make sure that other same jobs are not running, to prevent jobs from stepping on themselves. If a job fails and a new job is restarted, the parameter will still be set to Y until someone goes in and manually sets that value back to N. Make sure though that another same job isn't actually running at the same time, when setting this parameter back to N.

Troubleshooting – Case 1

Case Study

Troubleshooting – Case 1

- To troubleshoot failed jobs
- Step 4 – With the parameter now set to N, restart the failed job in the Scheduler.



QAD 65

Troubleshooting – Case 1

Case Study

Troubleshooting – Case 1

- To troubleshoot failed jobs
- Step 5 – Confirm that the job now runs successfully.

Jobs

Job	Status	Seq	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
DAILY_COMMON_PROCESS_QADDB_EE	On Hold	72	2011-11-03 10:39:10								
DAILY_COMMON_PROCESS_QADDB_EE	Running	4045	2011-11-03 10:39:10		00:54	6	42			3	
DAILY_START	Waiting	27	2011-11-04 10:39:00								
DAILY_START	Completed	4044	2011-11-03 10:39:10	2011-11-03 10:39:12	00:00	3	3				

Tasks

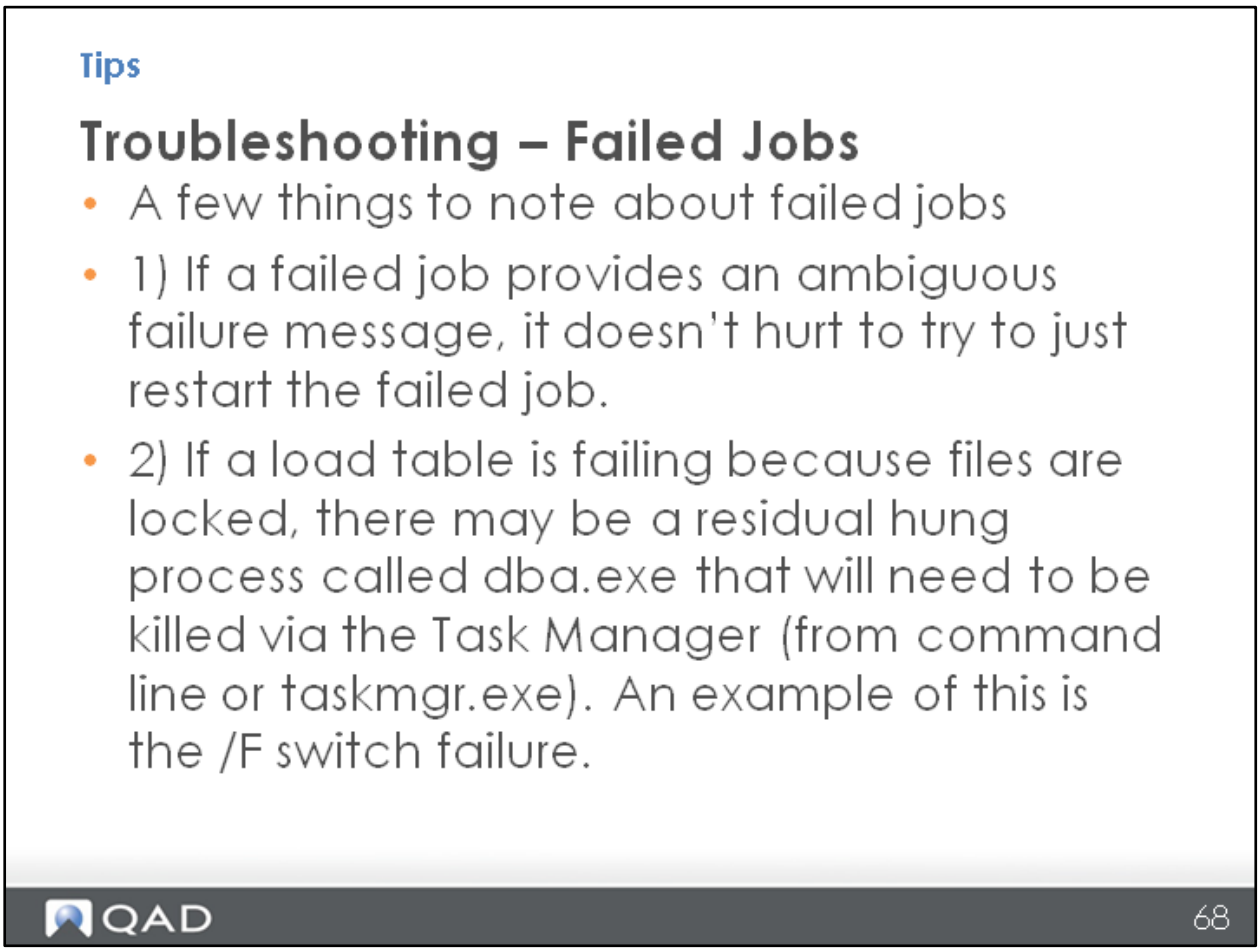
Task	Action	Status	Seq	Start	Finish	Ela...	Info	D...	W...	Result
setp_job_com_process_start	Execute	Compl...	4045	2011-11-03 1...	2011-11-03 1...	00:00	7			Procedure setp_job_com_process_start completed successfully.
load_ac_mstr	Process	Compl...	4045	2011-11-03 1...	2011-11-03 1...	00:00	7			3466 rows loaded via Native ODBC into load_ac_mstr
load_ap_transaction_type	Process	Compl...	4045	2011-11-03 1...	2011-11-03 1...	00:00	6			0 rows loaded into load_ap_transaction_type
load_ar_transaction_type	Process	Compl...	4045	2011-11-03 1...	2011-11-03 1...	00:00	6			0 rows loaded into load_ar_transaction_type
load_bg_mstr	Process	Compl...	4045	2011-11-03 1...	2011-11-03 1...	00:00	6			SQL Admin: No data returned from source system. No data loaded...
load_budget_actual	Process	Compl...	4045	2011-11-03 1...	2011-11-03 1...	00:00	4			1 rows loaded into load_budget_actual
load_cc_mstr	Process	Running	4045	2011-11-03 1...		00:00	3			
load_code_mstr	Process	Running	4045	2011-11-03 1...		00:00	2			

QAD 66

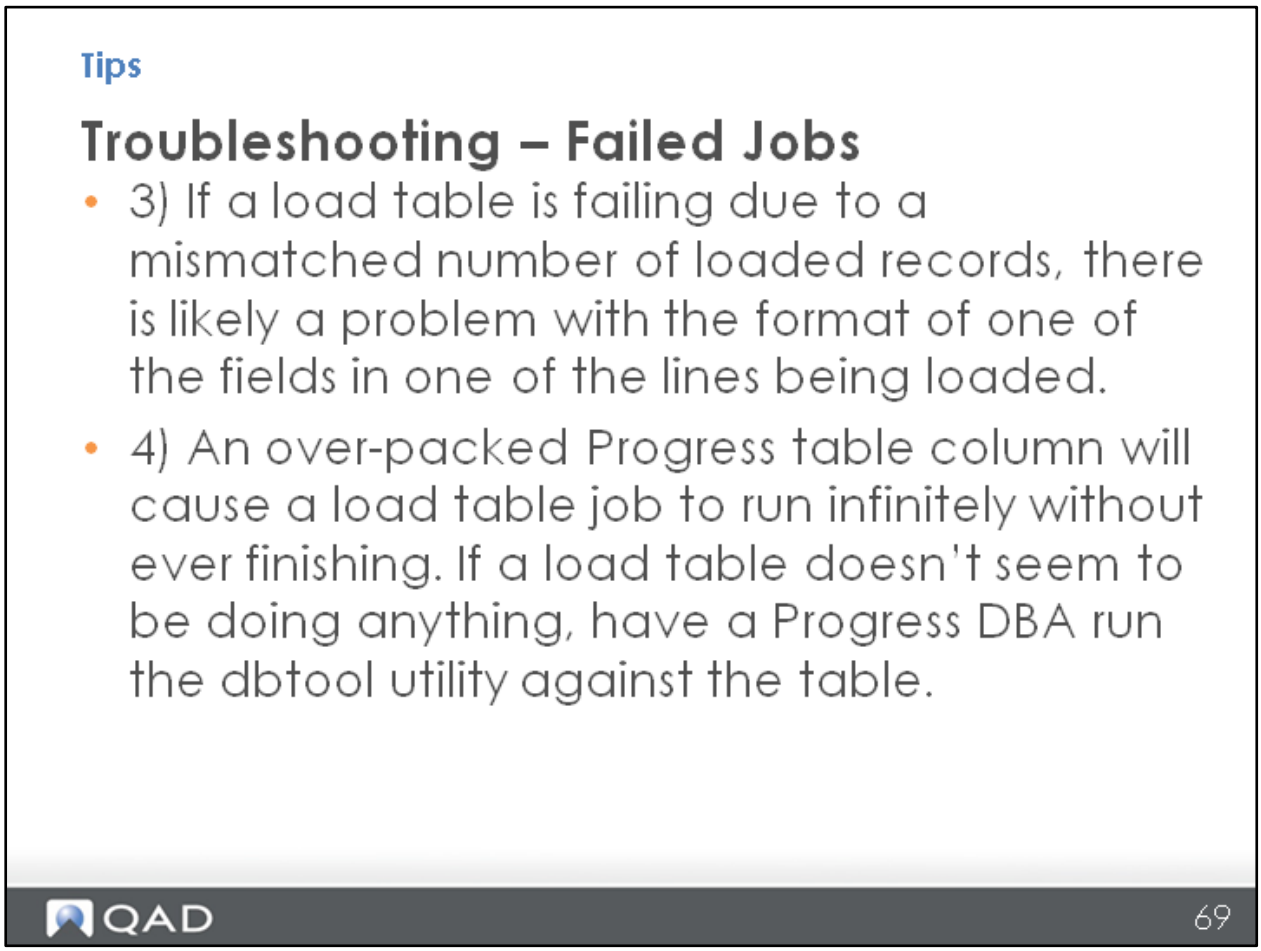
Troubleshooting Tips



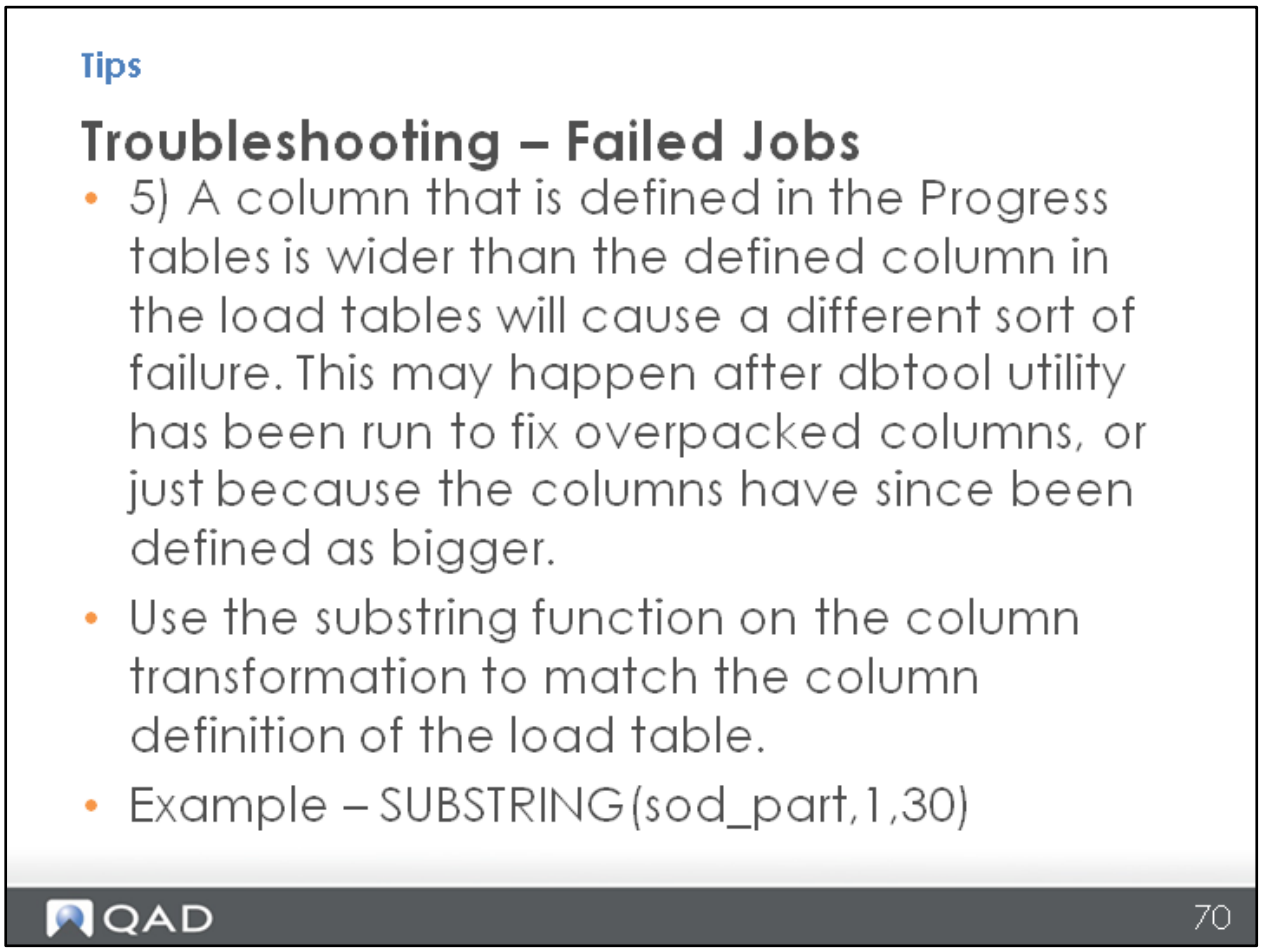
Troubleshooting – Failed Jobs



Troubleshooting – Failed Jobs



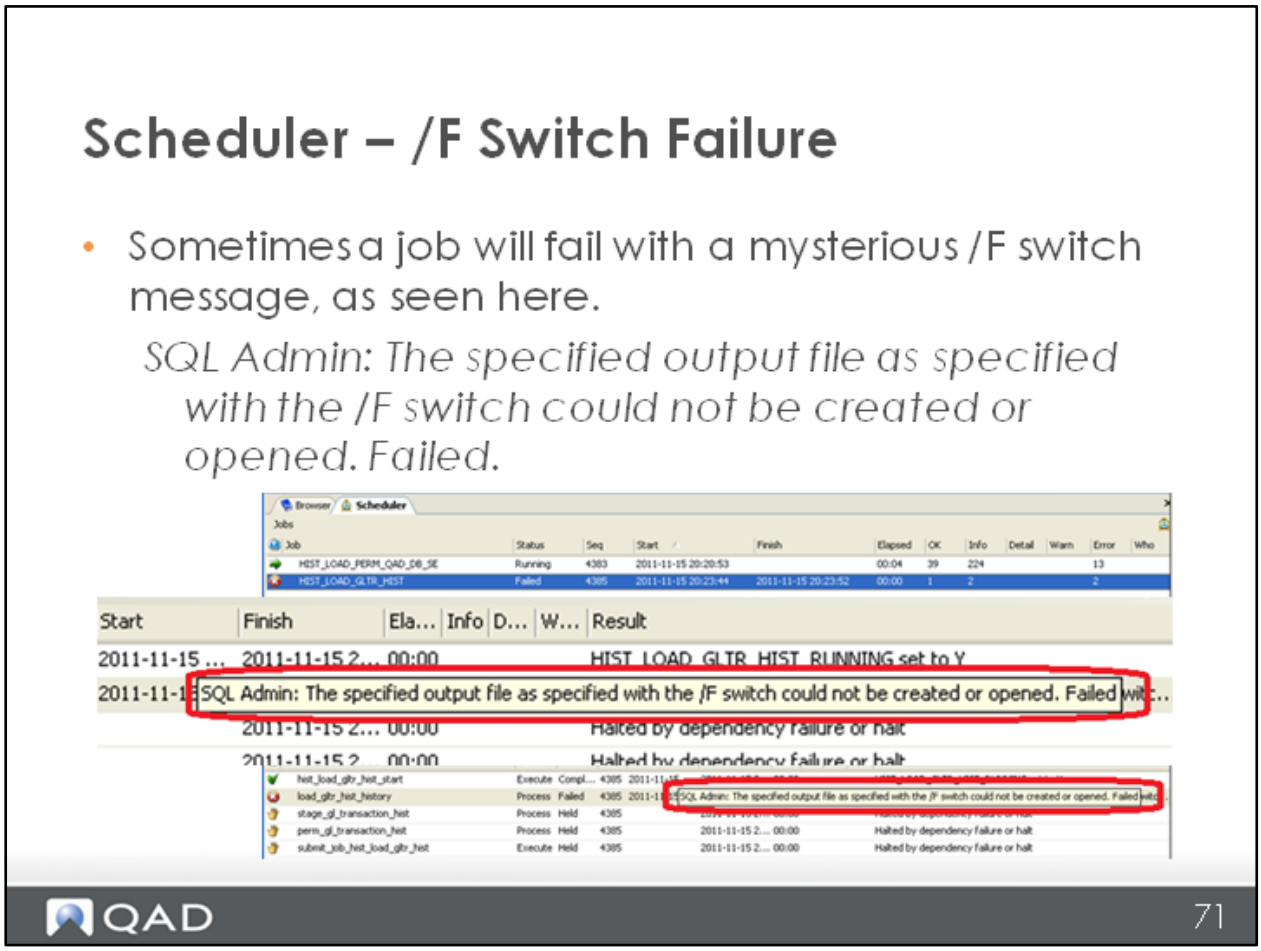
Troubleshooting – Failed Jobs



Instead of forcing the customer to have shorter names in their column definitions or changing the column widths in the load tables that may run into problems later if for some reason the load table gets upgraded, simply put a substring on the column to only bring over as much data as the column is defined. This is presuming that the customer doesn't need whatever extra data is being truncated in that column.

If we keep having to do this for certain columns, definitely let the BI R&D team know and we'll increase the column sizes in the load tables for future releases. We've done this already with some of the address fields.

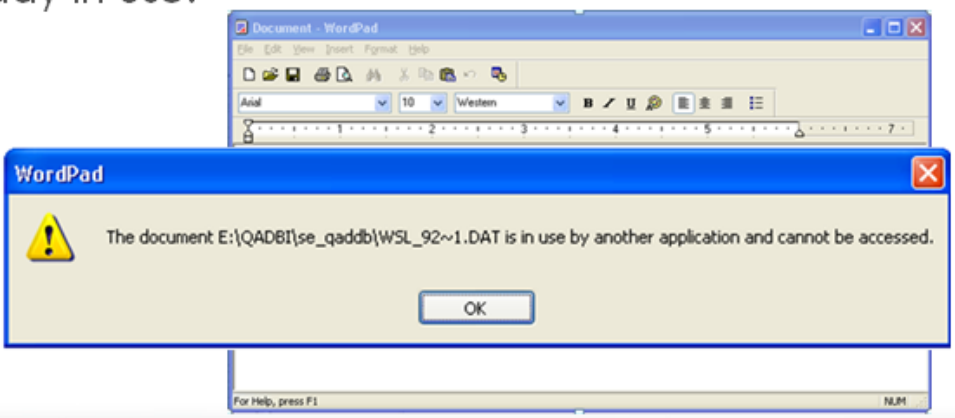
Scheduler – /F Switch Failure



Scheduler – /F Switch Failure

Scheduler – /F Switch Failure

- This is due to a job failure where the there is a hung job that locks a .dat file.
- If you try to look at the .dat file under the database connection name folder, it will tell you that the file is already in use.

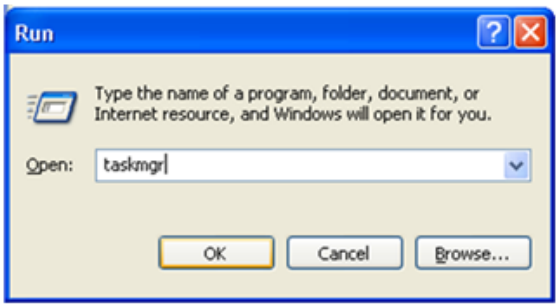


QAD 72

Scheduler – /F Switch Failure

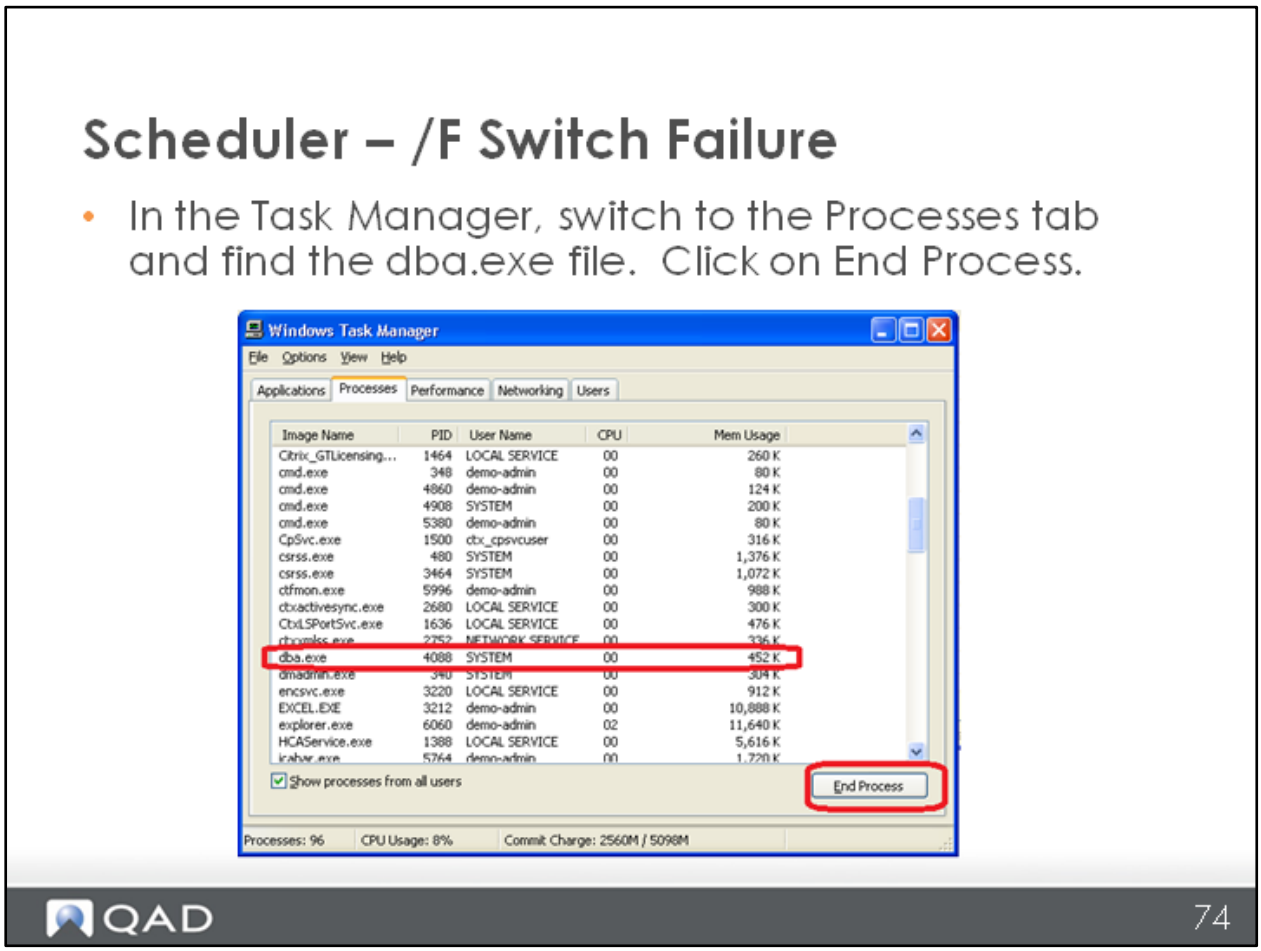
Scheduler – /F Switch Failure

- To unlock the .dat file to be able to get the job to run again, do the following.
- Launch the Task Manager (this can be done via Start/Run).



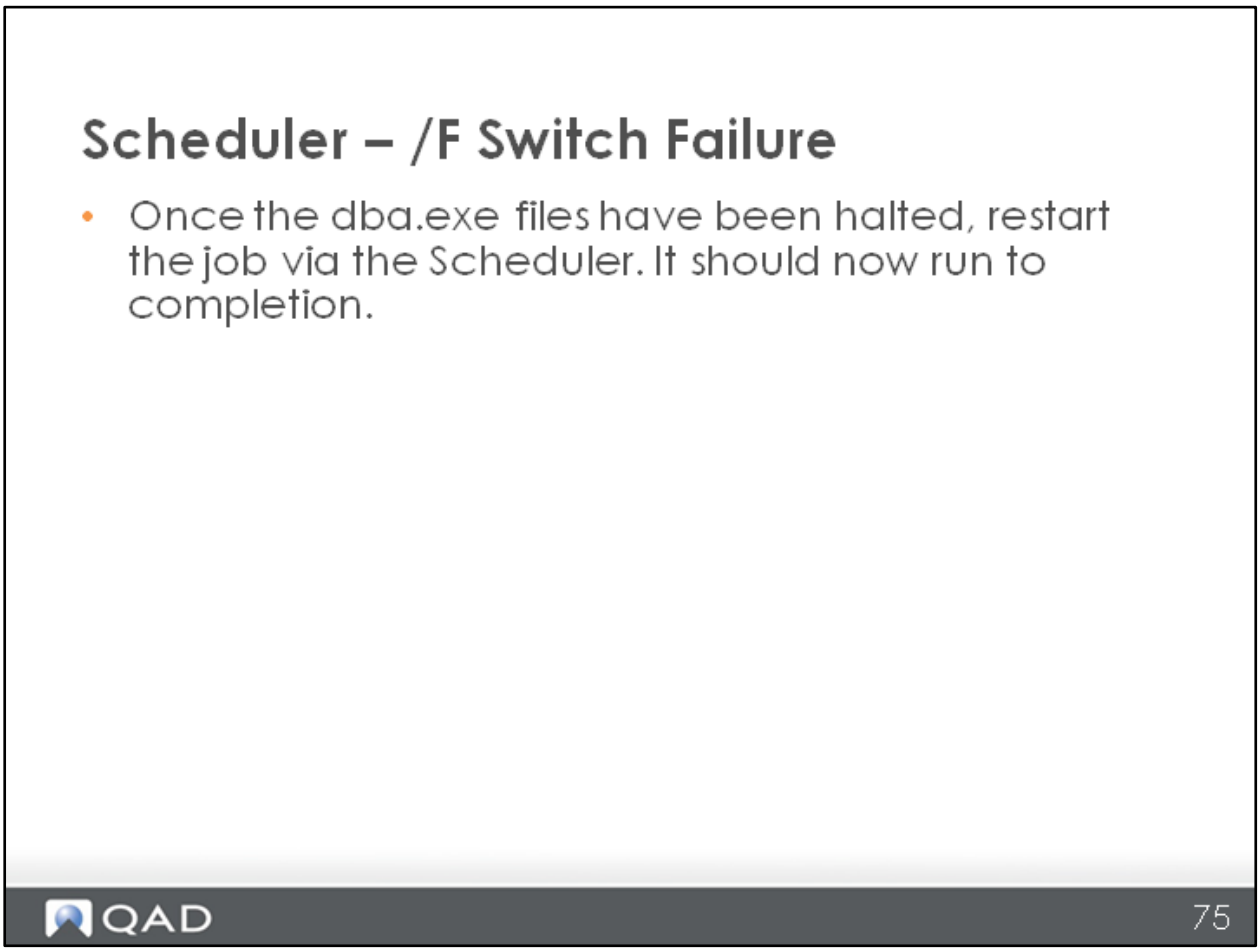
QAD 73

Scheduler – /F Switch Failure

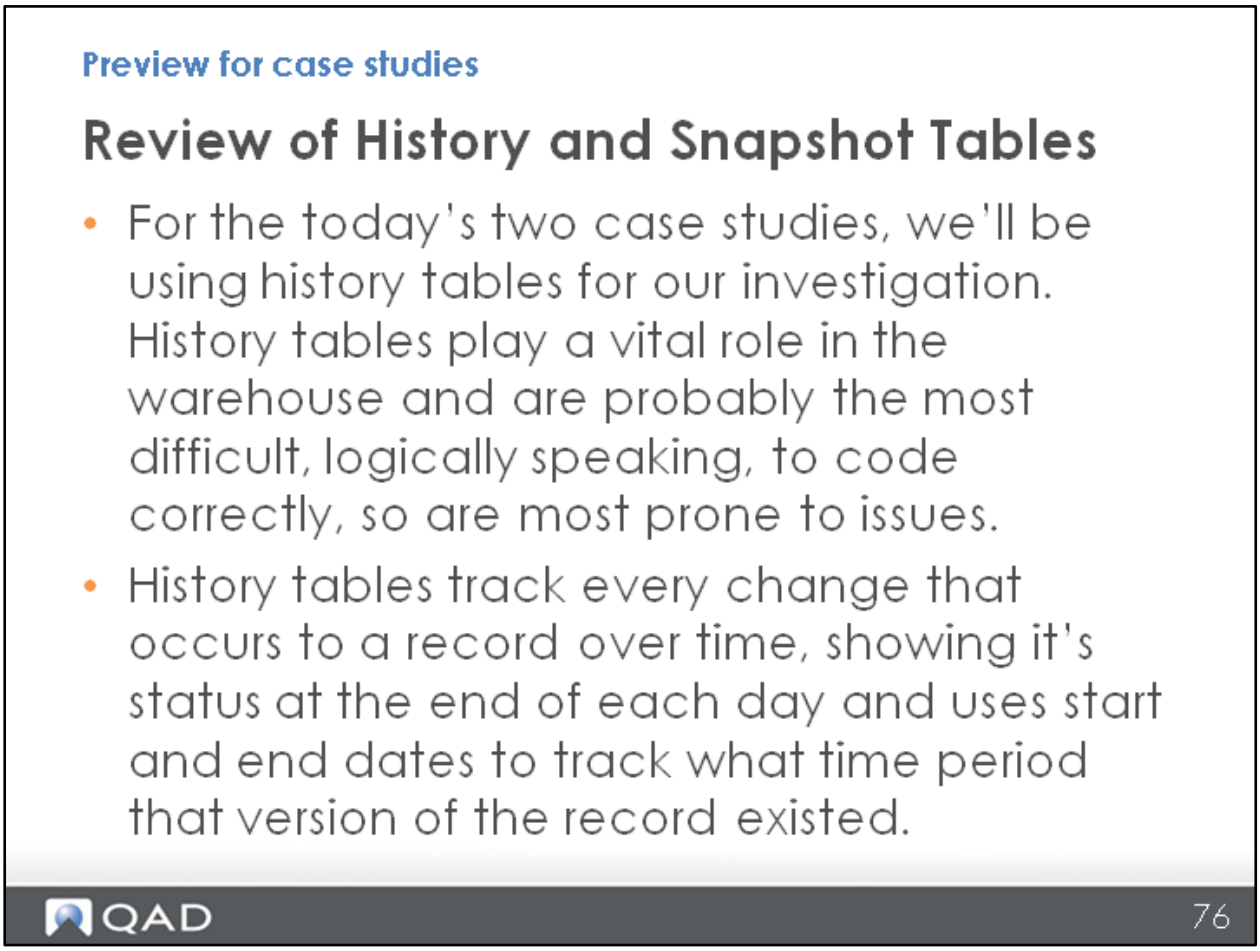


Click on Image Name to sort by name. Look for a job called dba.exe. It's possible there may be multiple running (if there are, make sure that other load tables aren't actually loading before proceeding). This is the job that is holding the file. Click on the job, then click End Process.

Scheduler – /F Switch Failure



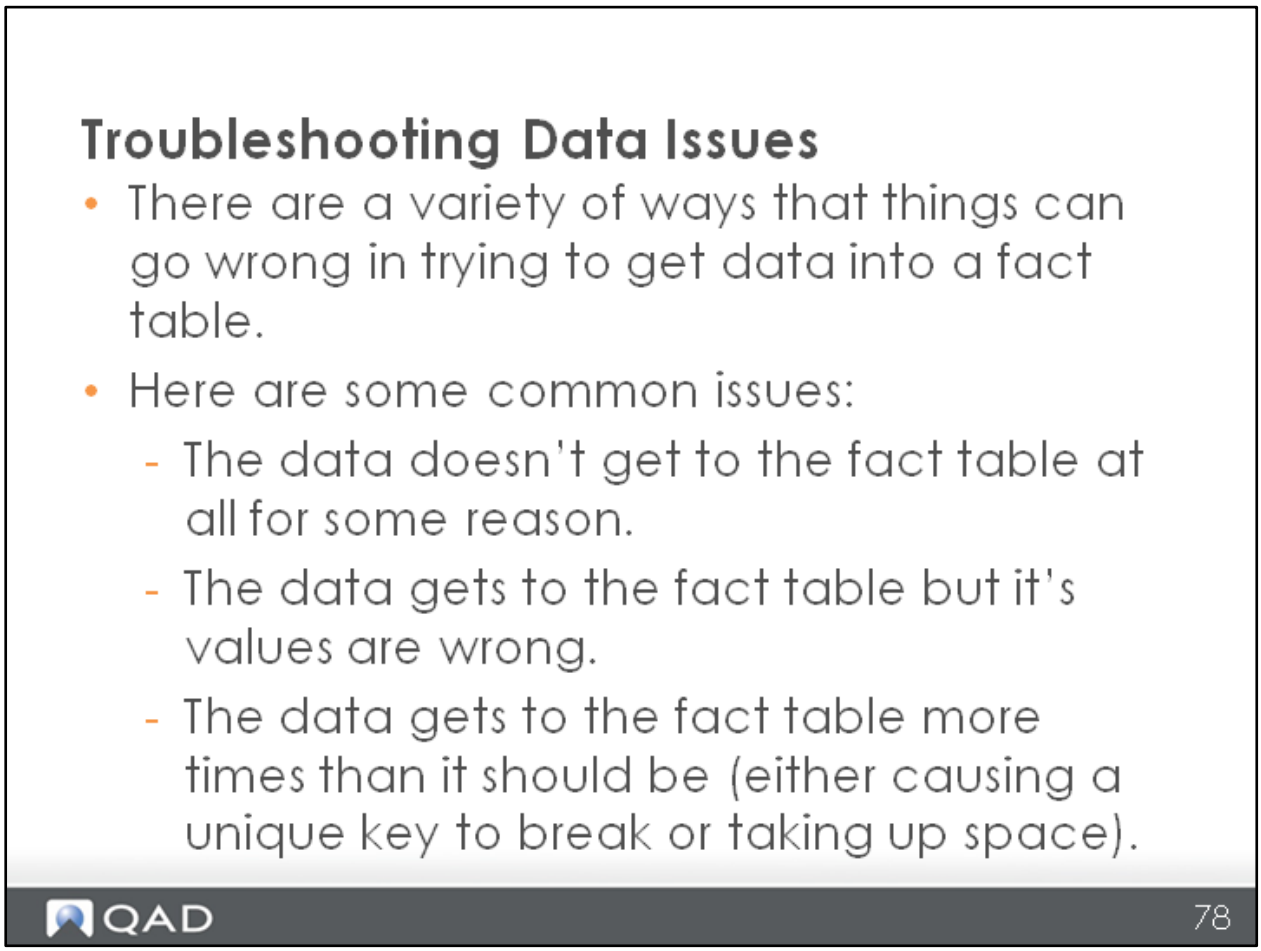
Review of History and Snapshot Tables



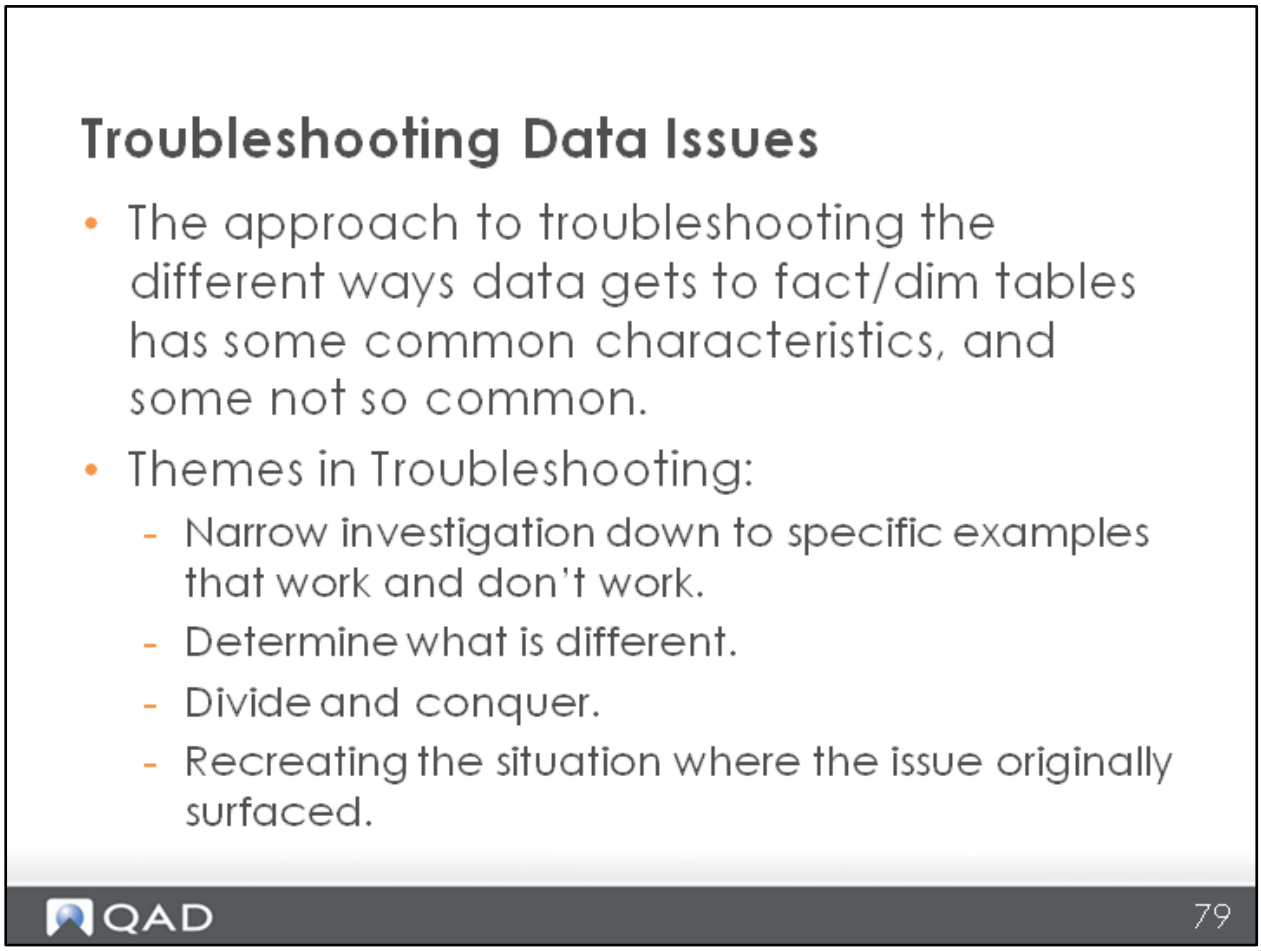
Troubleshooting



Troubleshooting Data Issues

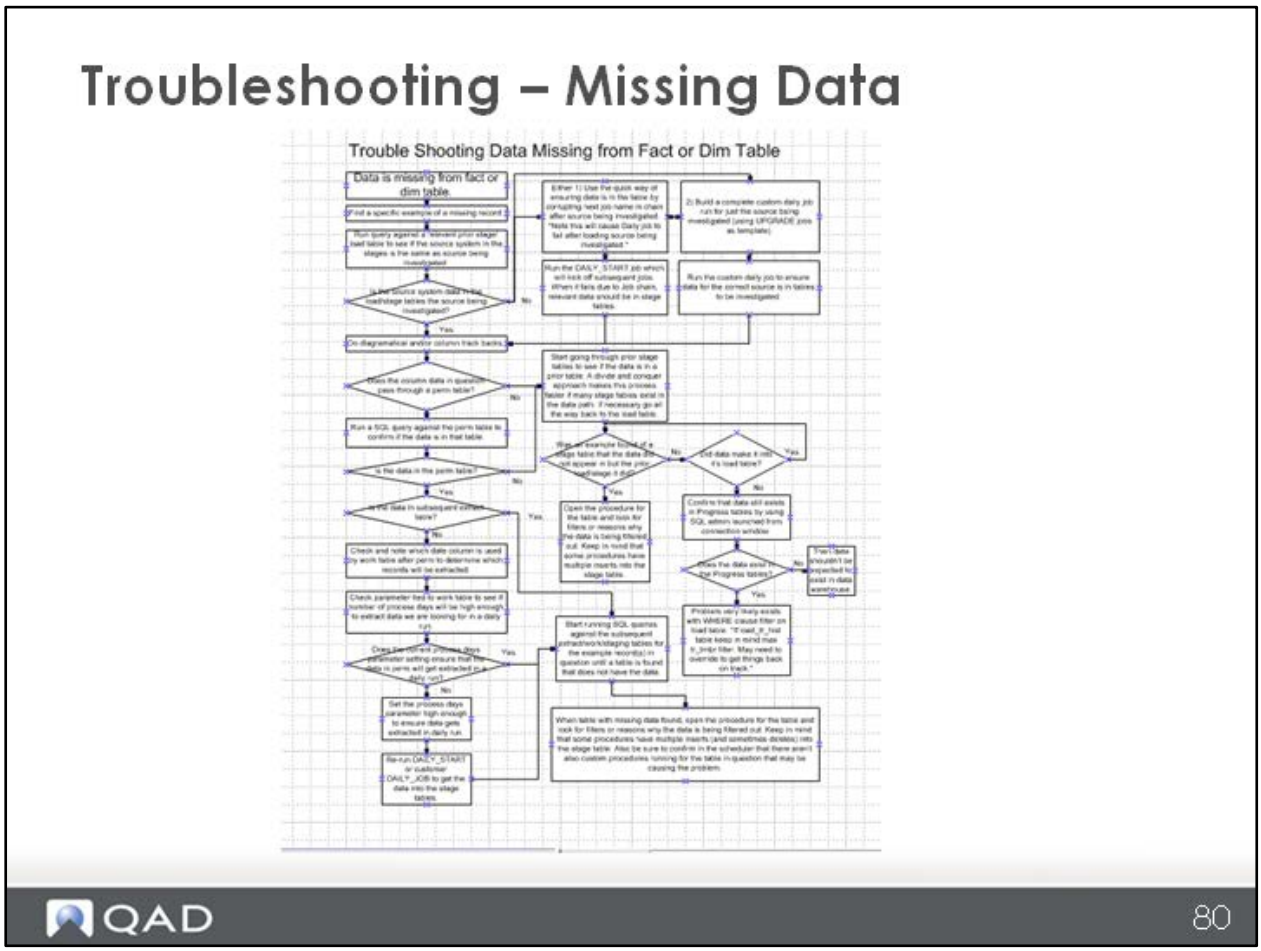


Troubleshooting Data Issues

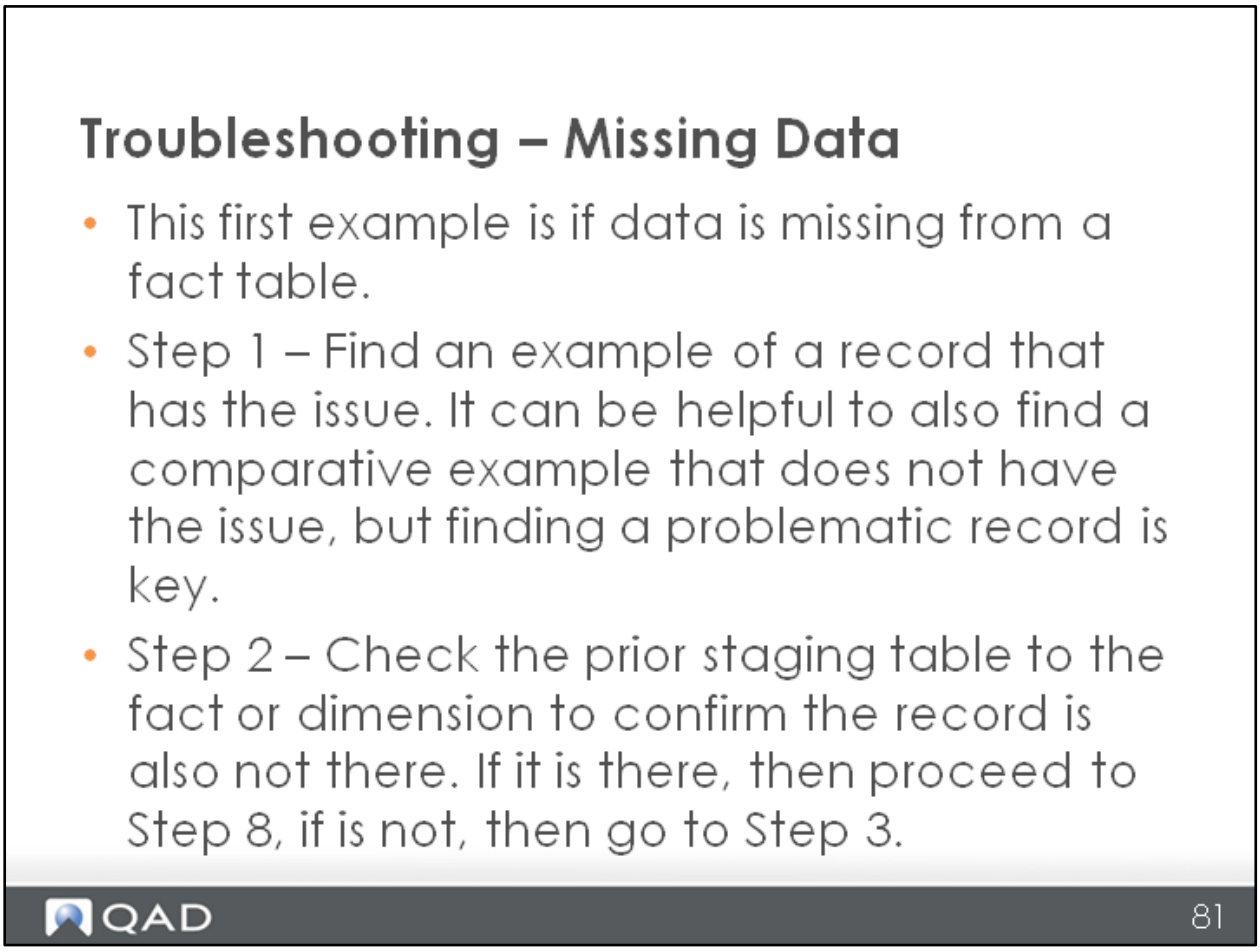


It can be helpful also to find an example that does not have the issue as a baseline to compare results to, but finding a problematic record is key.

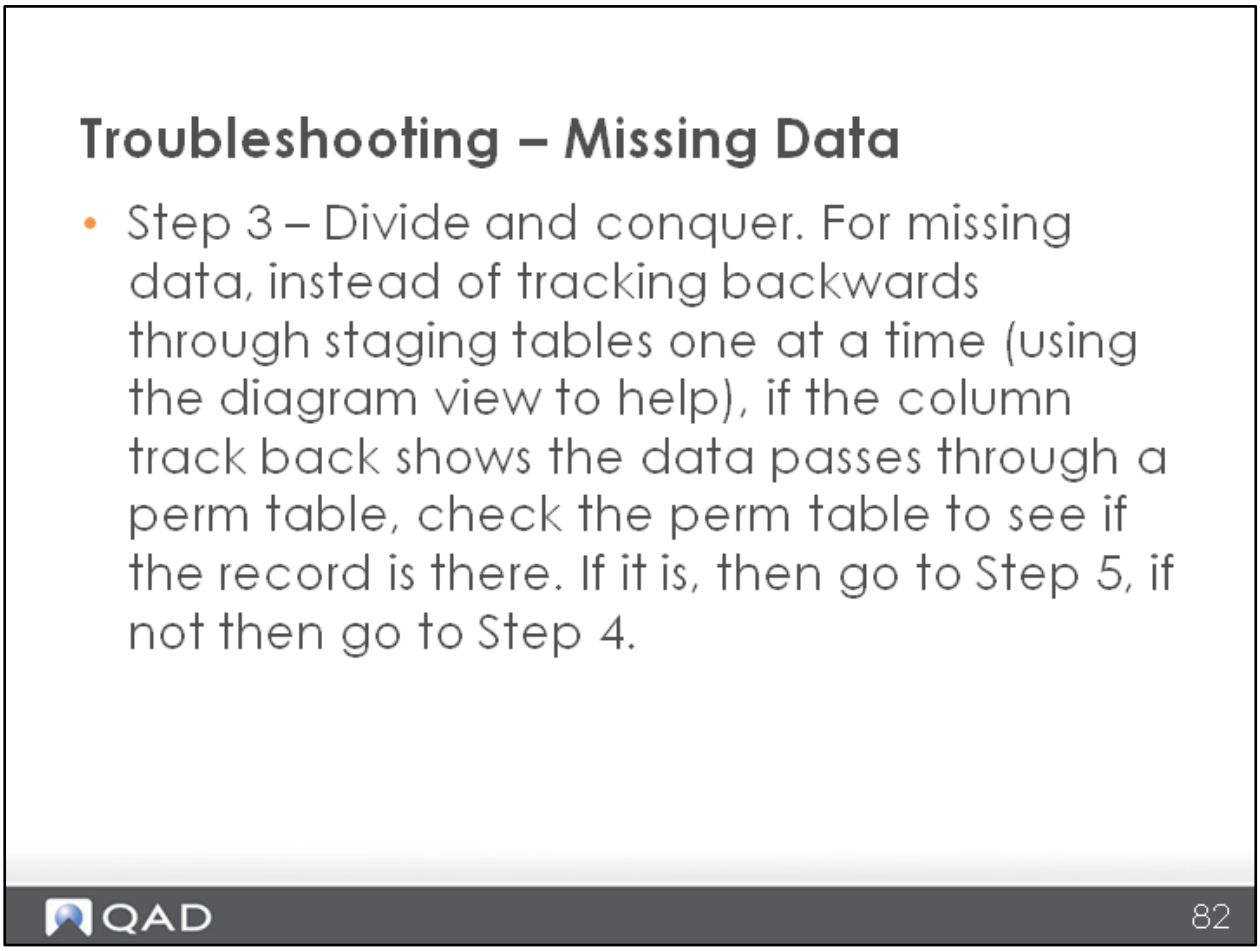
Troubleshooting – Missing Data



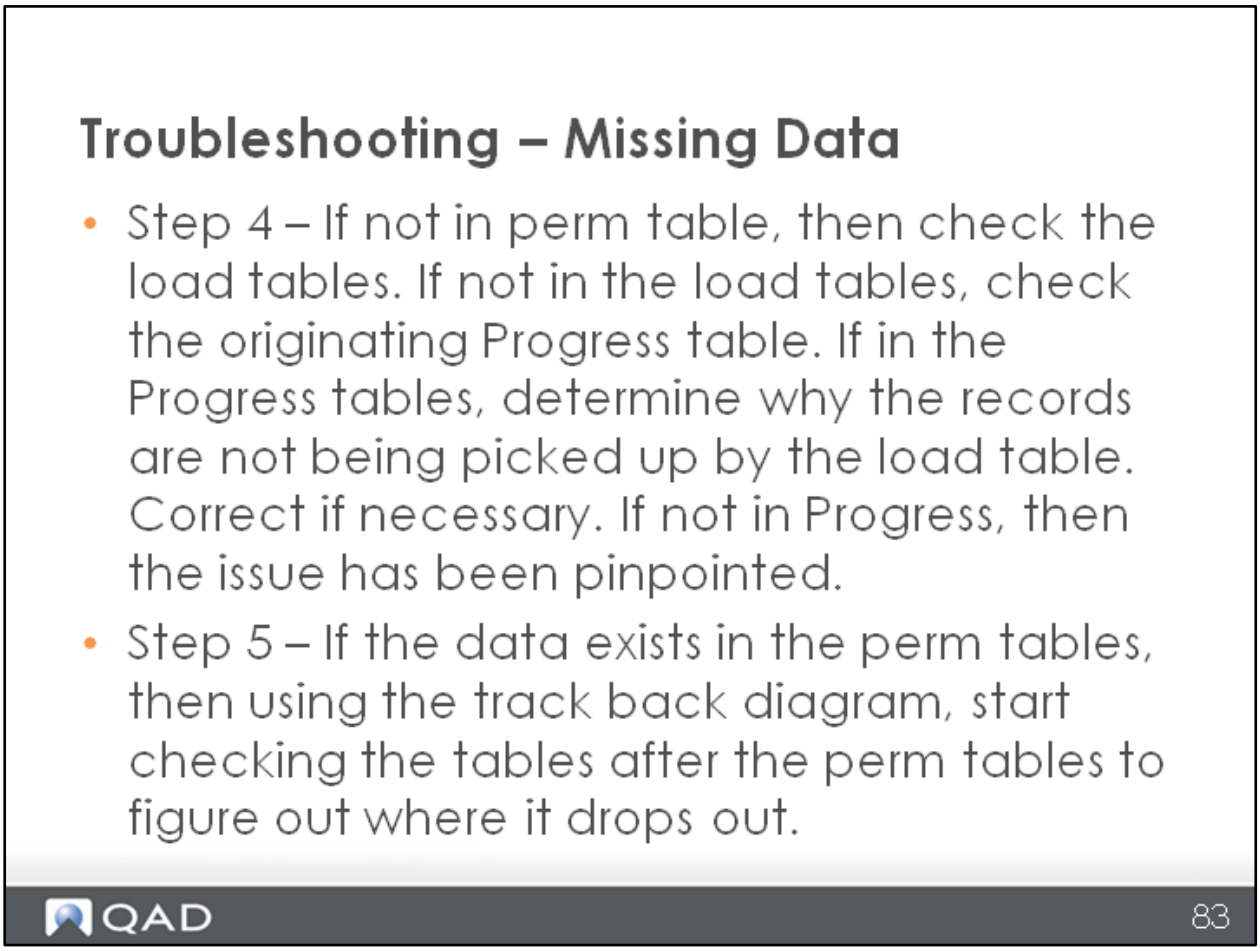
Troubleshooting – Missing Data



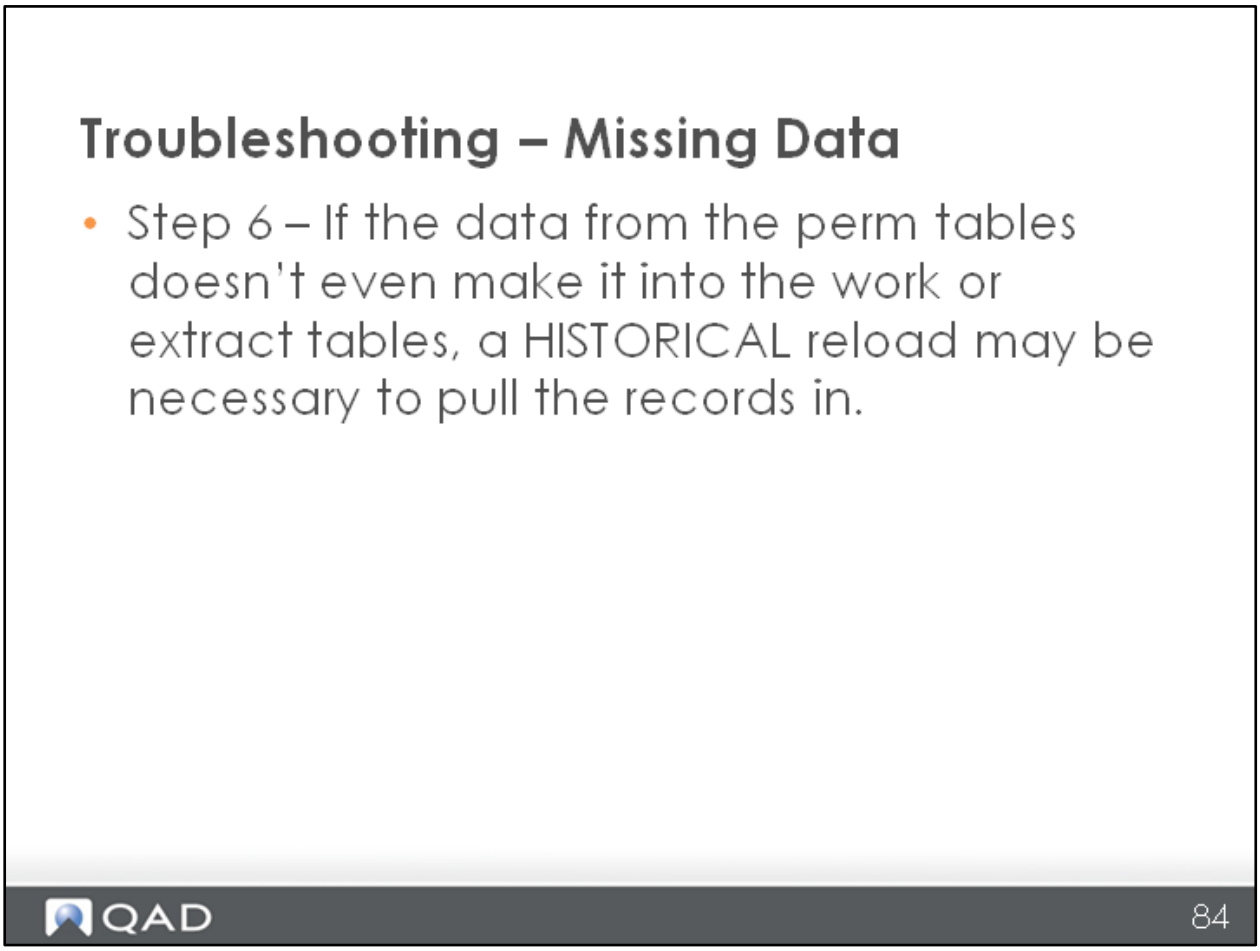
Troubleshooting – Missing Data



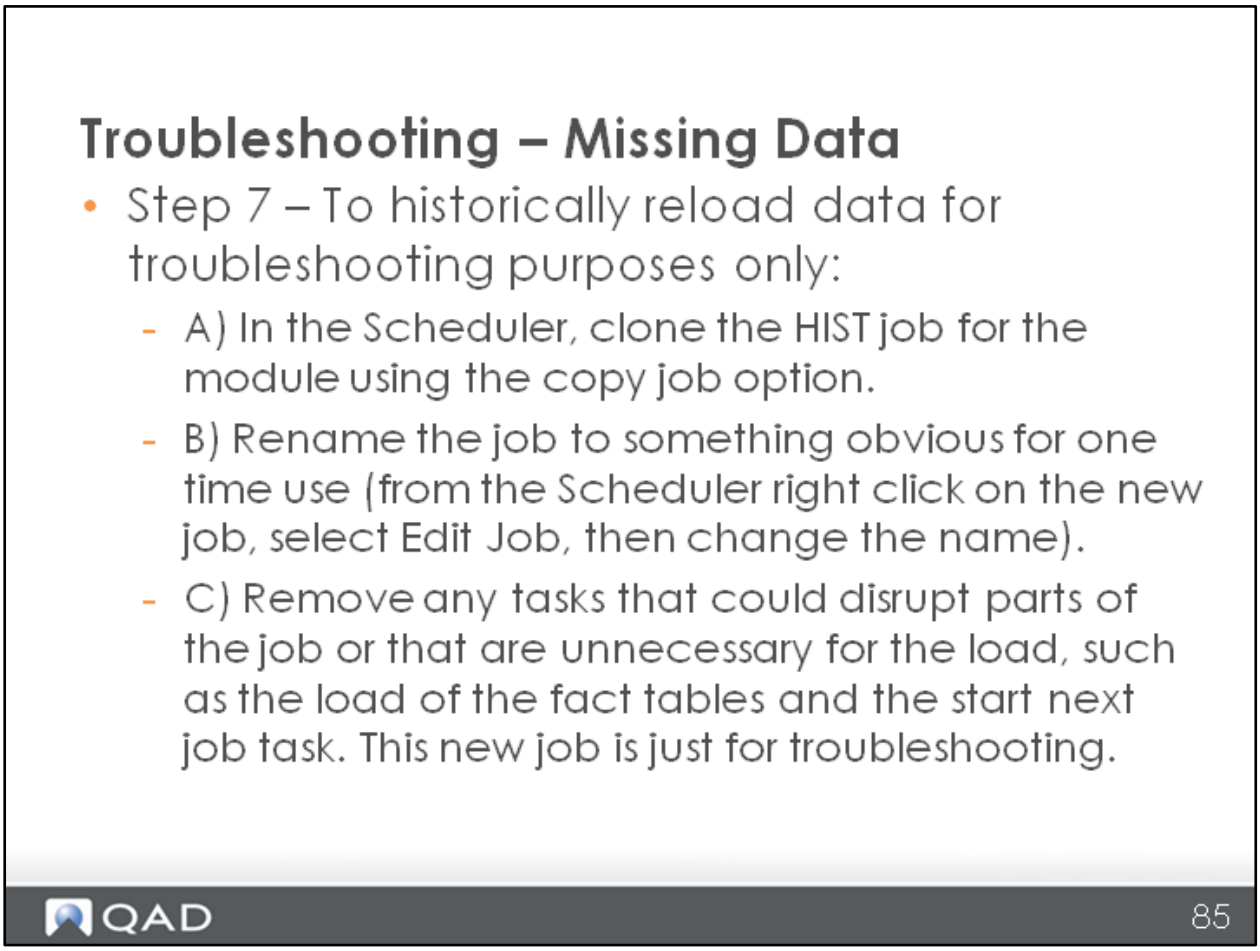
Troubleshooting – Missing Data



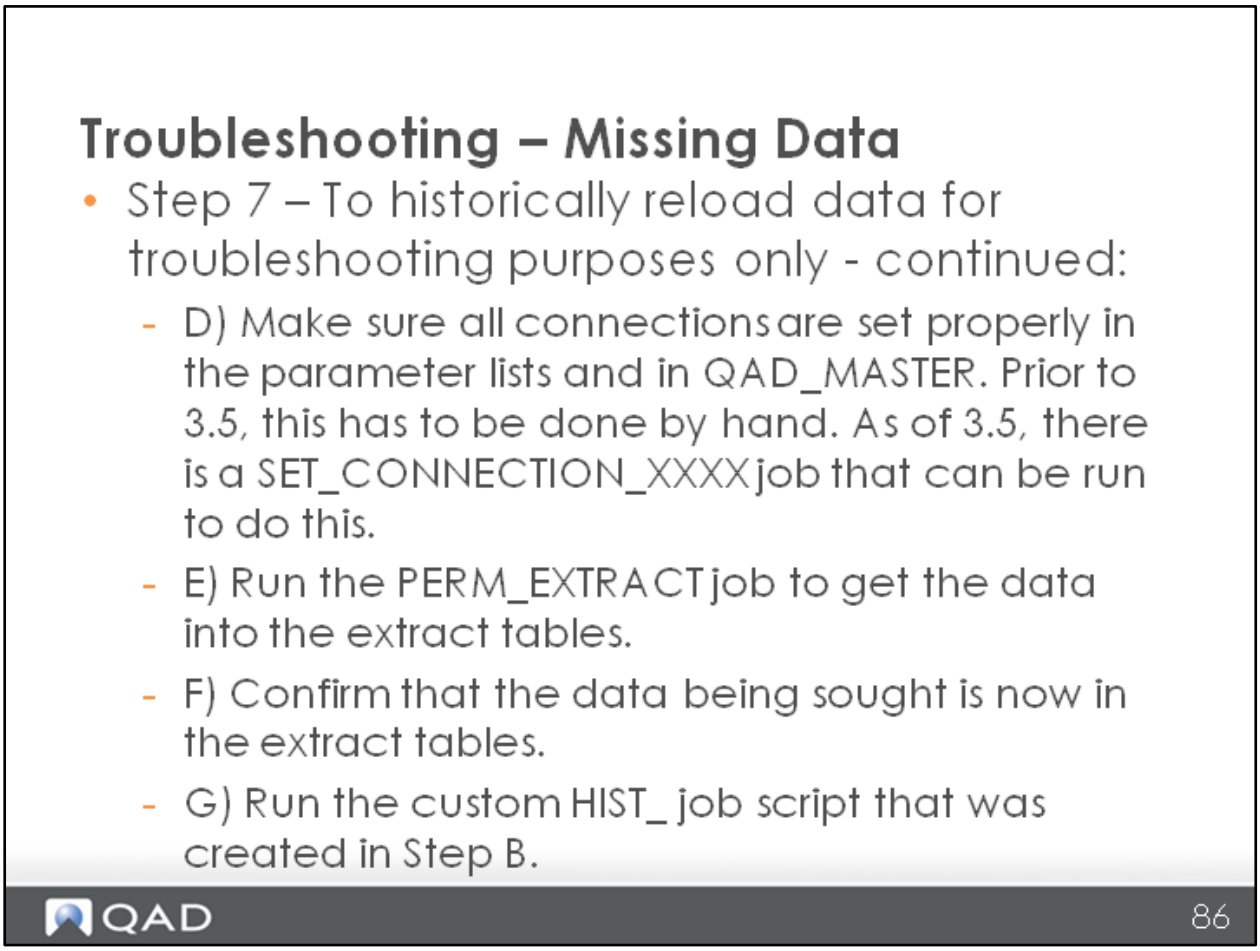
Troubleshooting – Missing Data



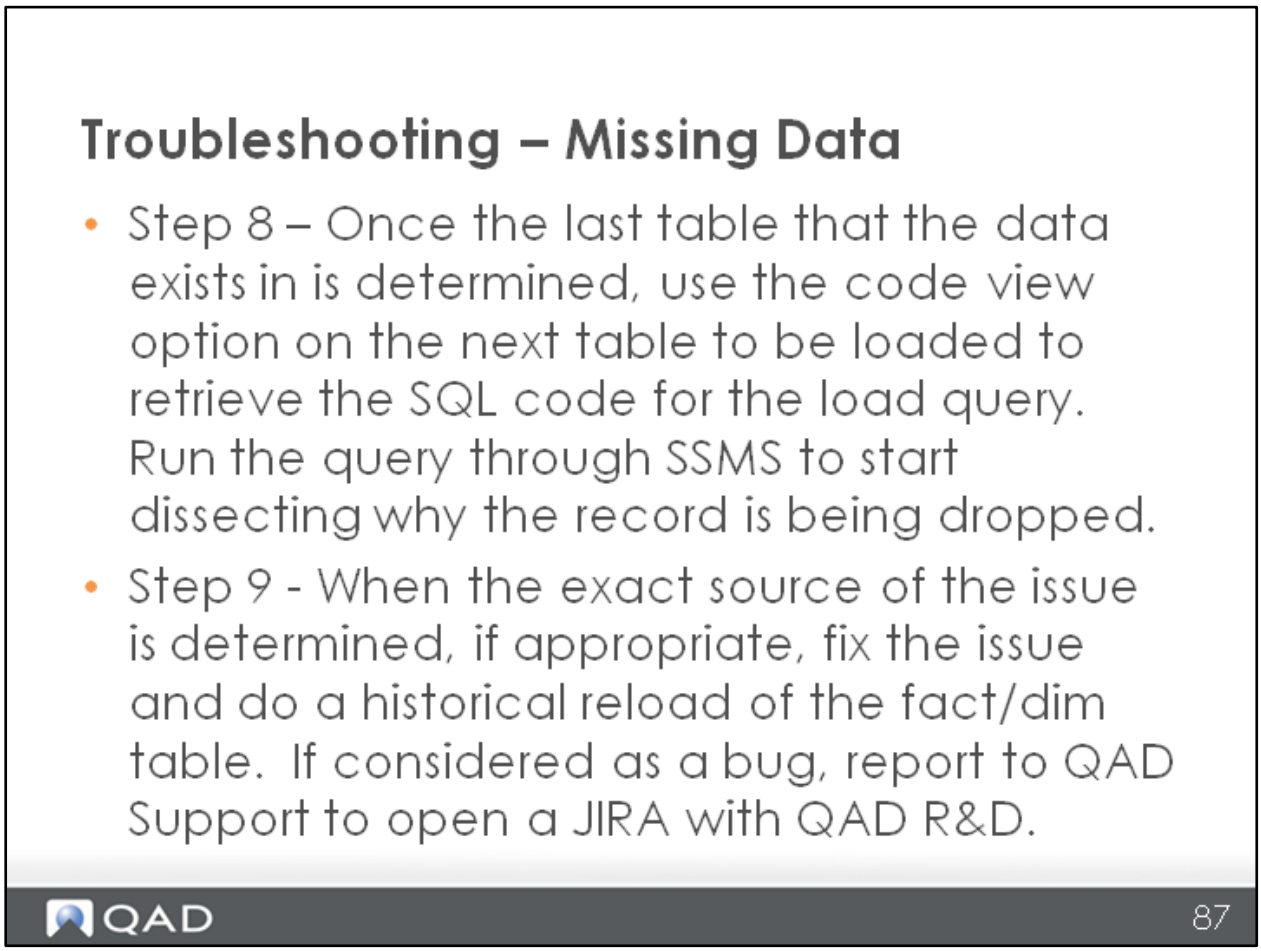
Troubleshooting – Missing Data



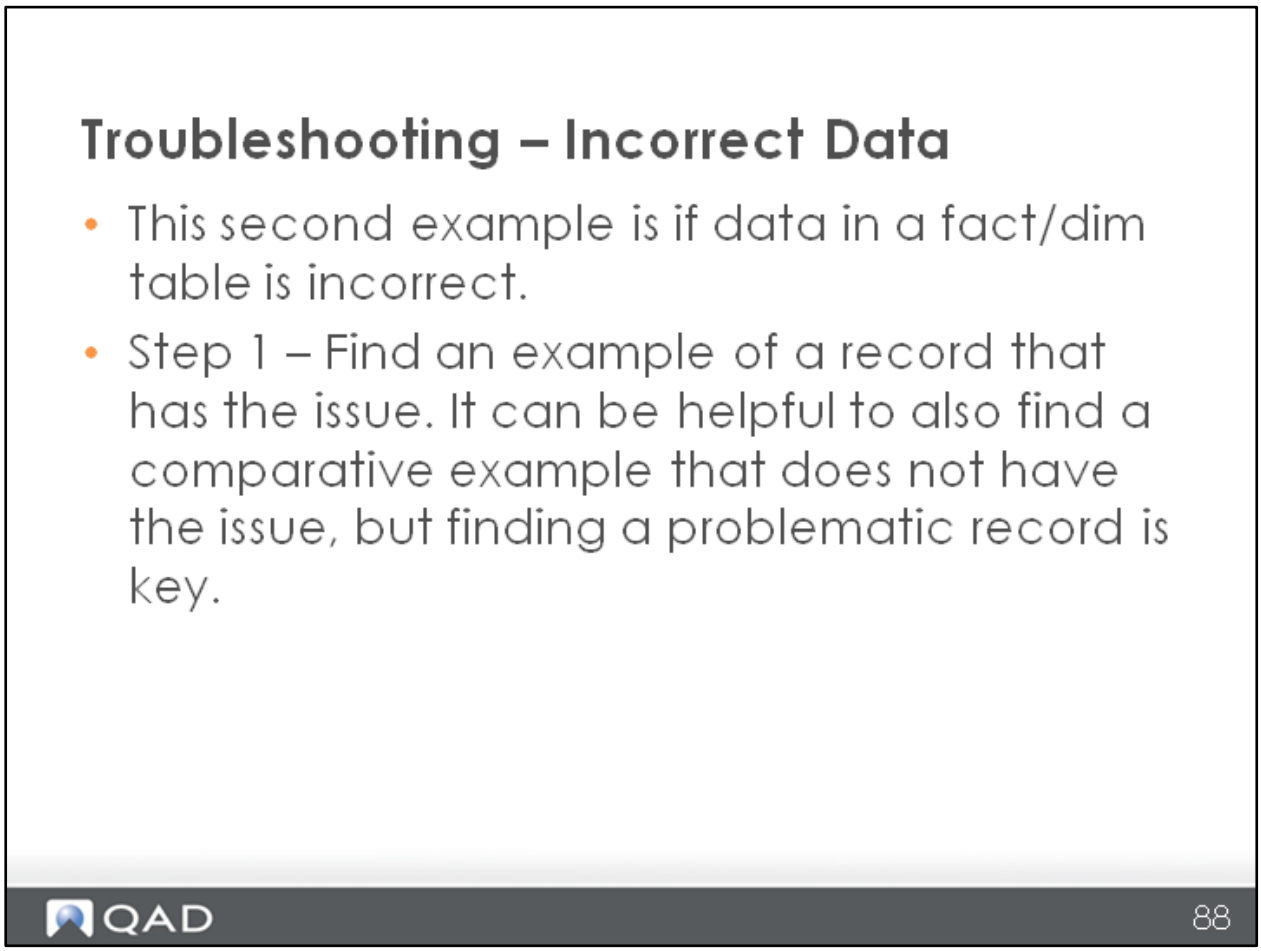
Troubleshooting – Missing Data



Troubleshooting – Missing Data

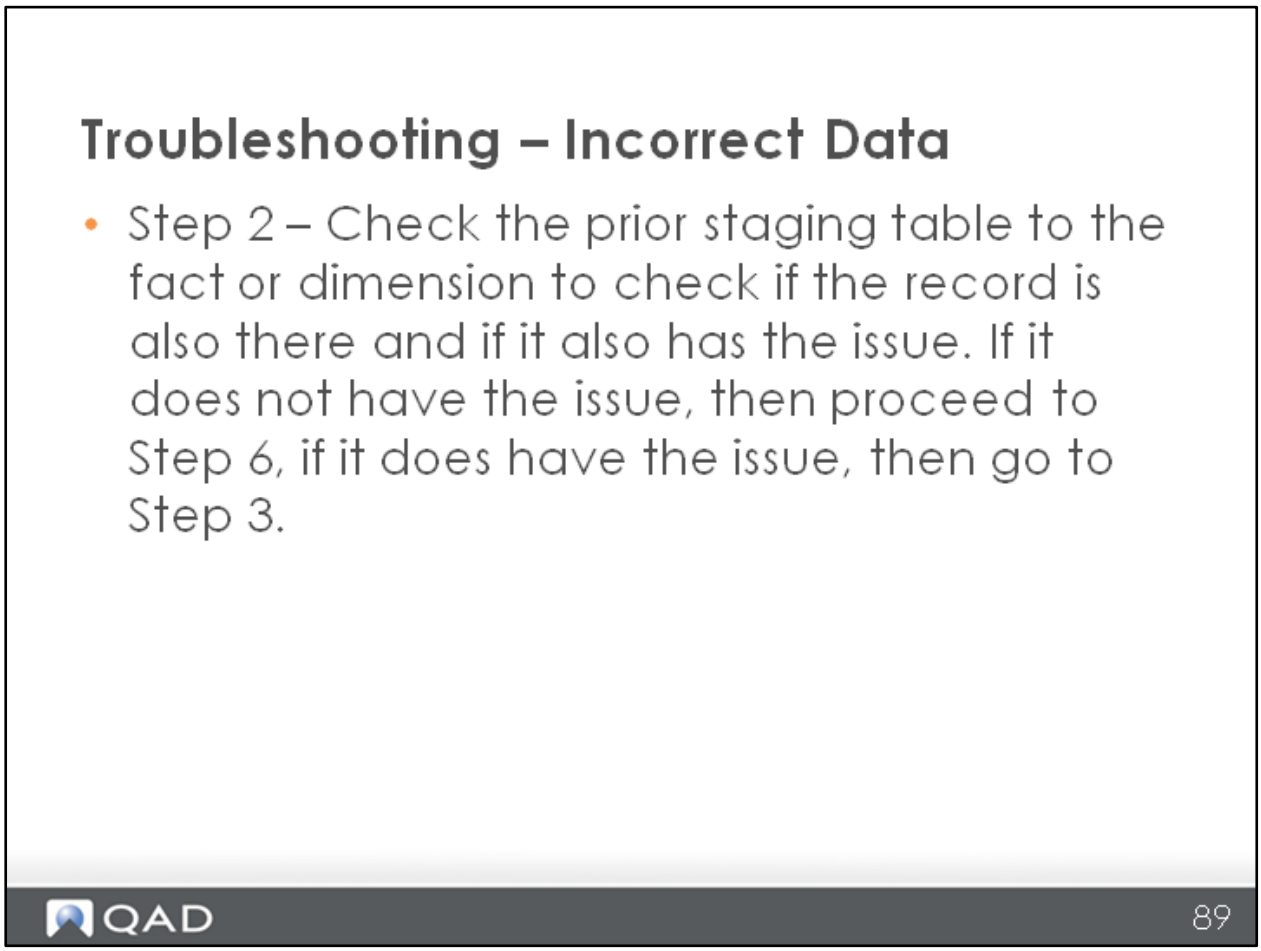


Troubleshooting – Incorrect Data

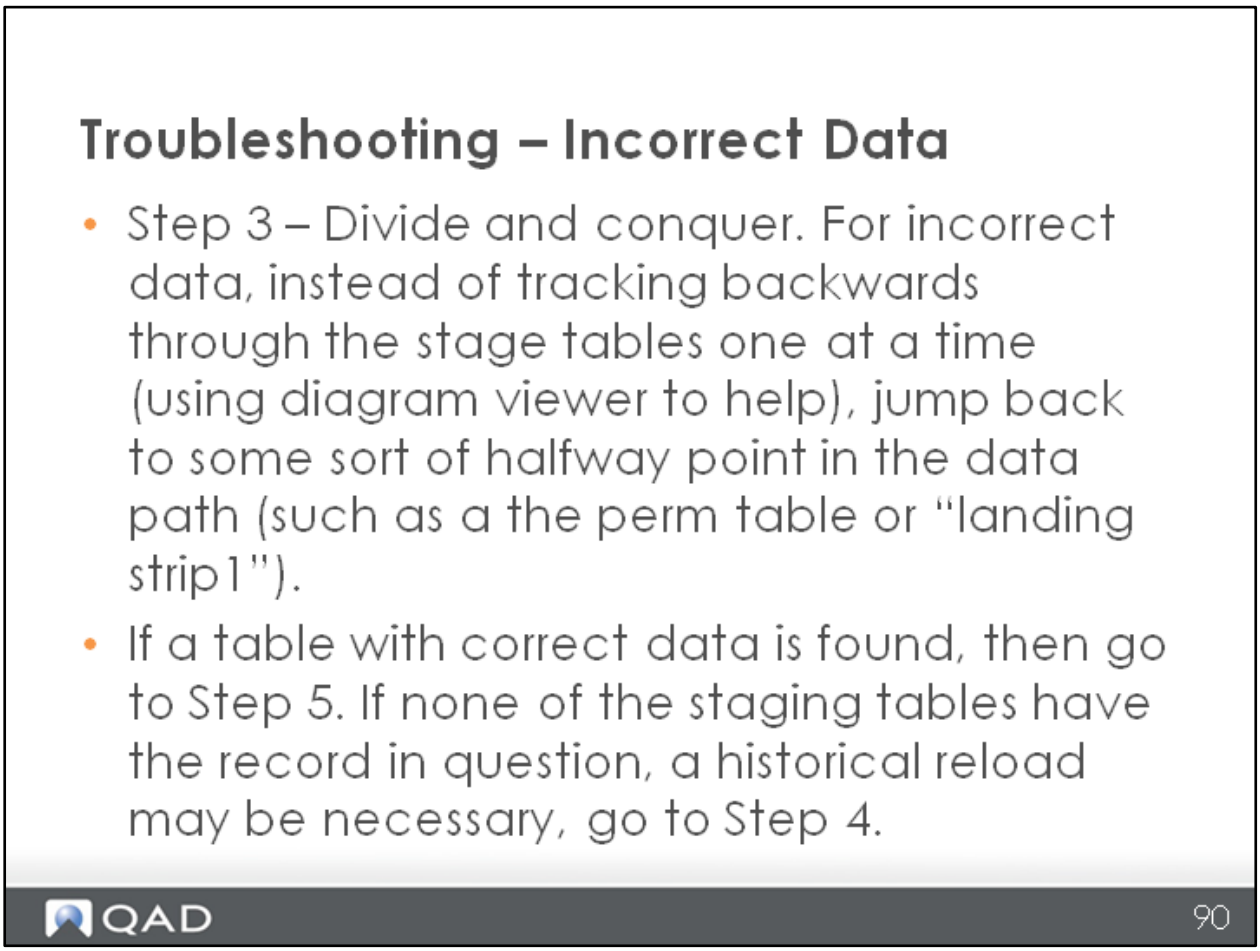


It can be helpful to also find an example that does not have the issue as a baseline to compare results to, but finding a problematic record is key. If the issue is being raised by a customer, ask them for a specific example, if they are willing to provide one. Some customers will just look at the total from 3 million records and say that the result is wrong. That provide much information to go on. If at all possible enlist their aid in finding a specific record that can be pointed at to say, “That record is wrong.”

Troubleshooting – Incorrect Data

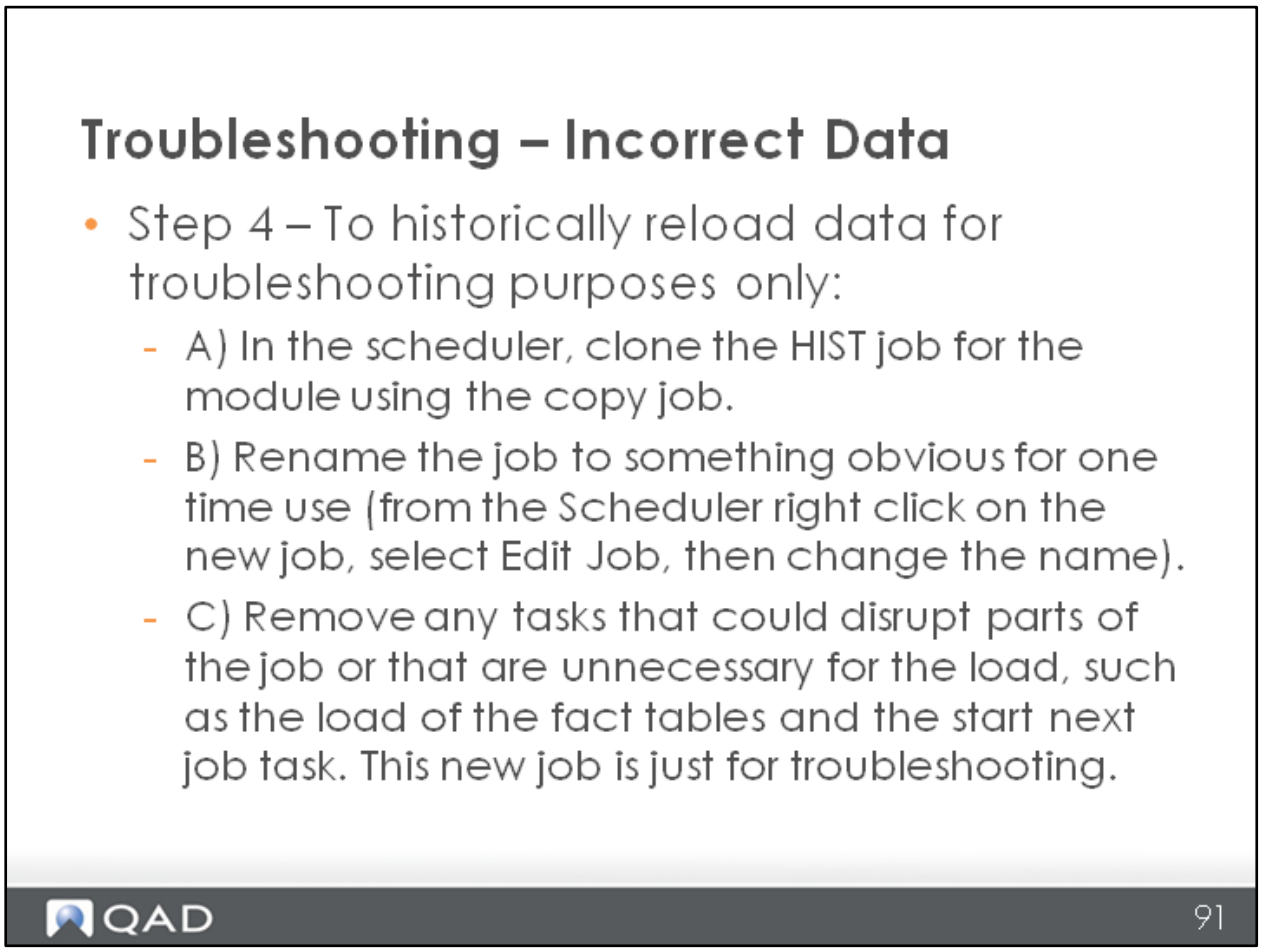


Troubleshooting – Incorrect Data

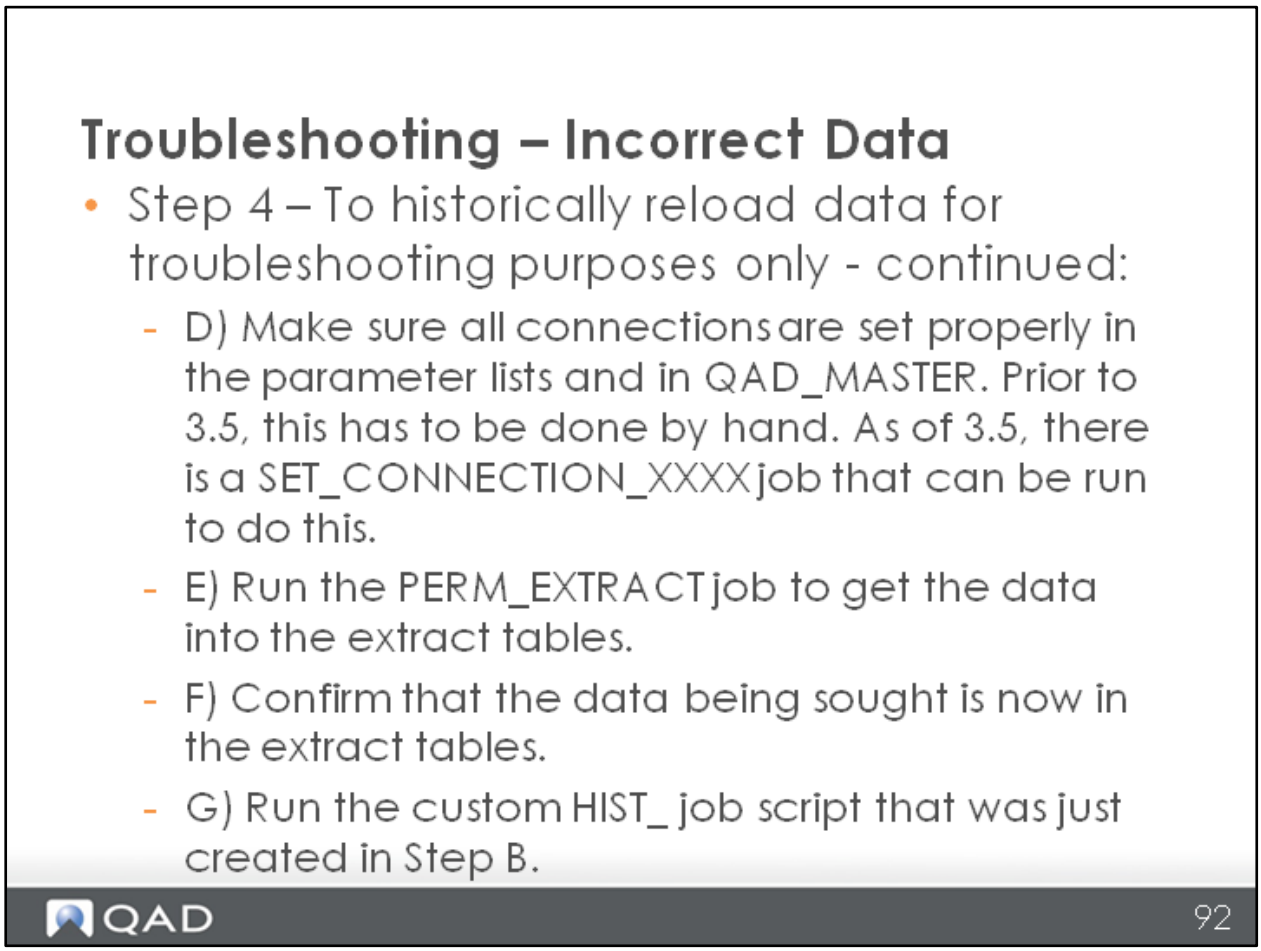


If the column track back shows the data passes through a perm table, check the perm table to see if the record is there and if it is correct or not. Or at least jump back to the first table in the "landing strip" to the fact table. The "landing strip" tables are the final stage tables prior to a fact table that have all the records headed to the fact collated, but still need to calculate currency conversion rates and get dimension keys, etc. Examples would be stage_om_order1, stage_om_order2, stage_om_order3, stage_om_order.

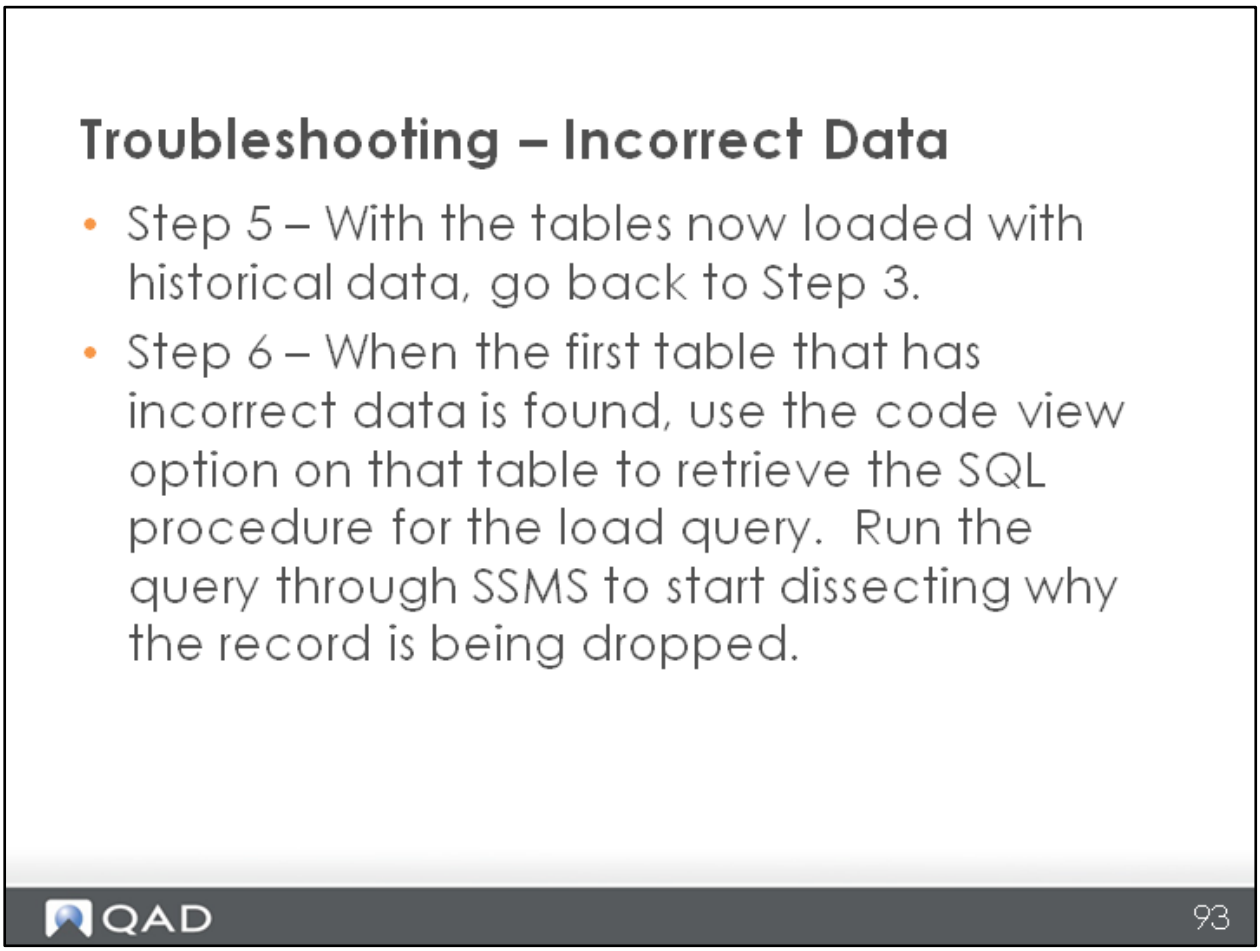
Troubleshooting – Incorrect Data



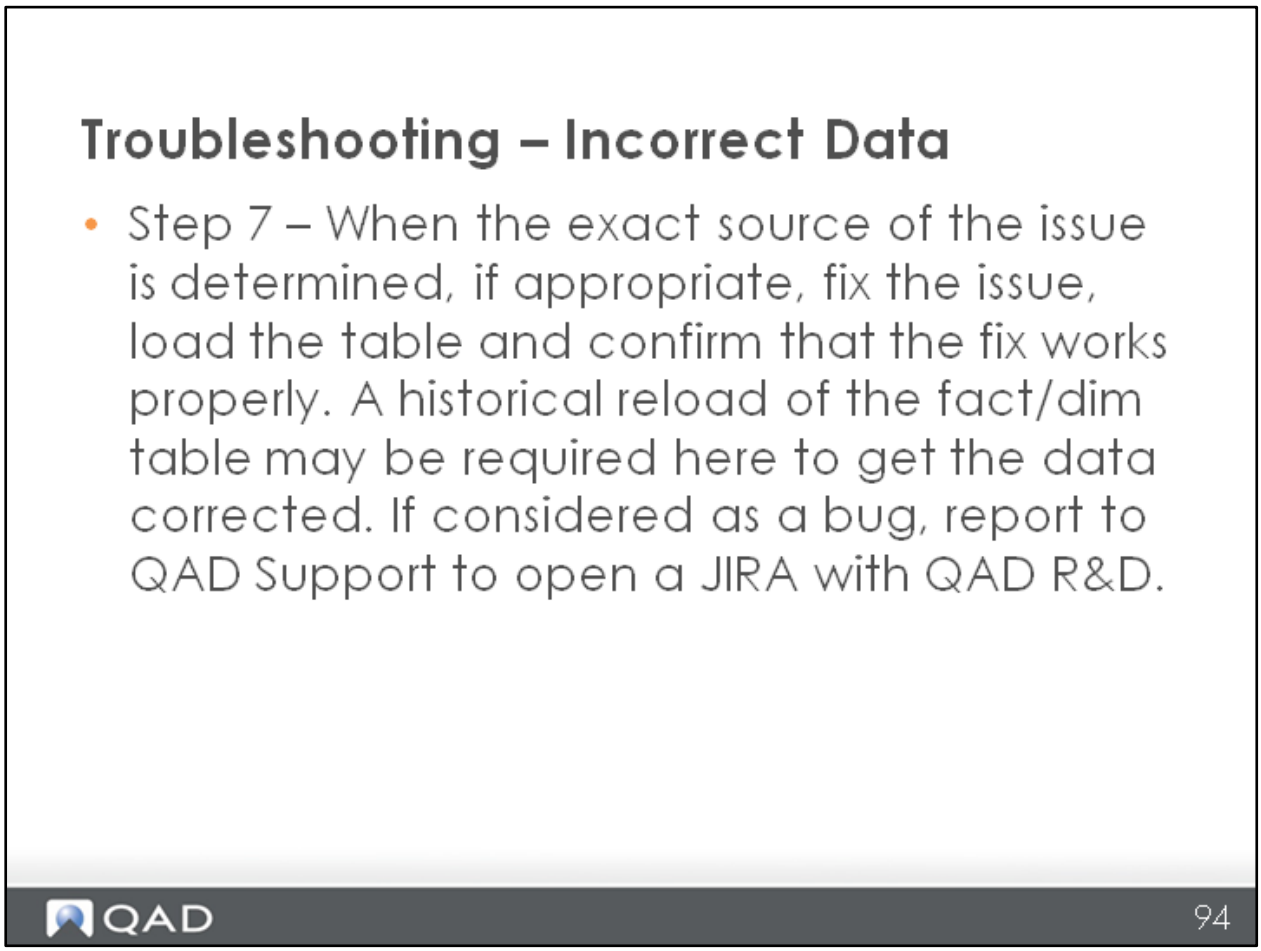
Troubleshooting – Incorrect Data



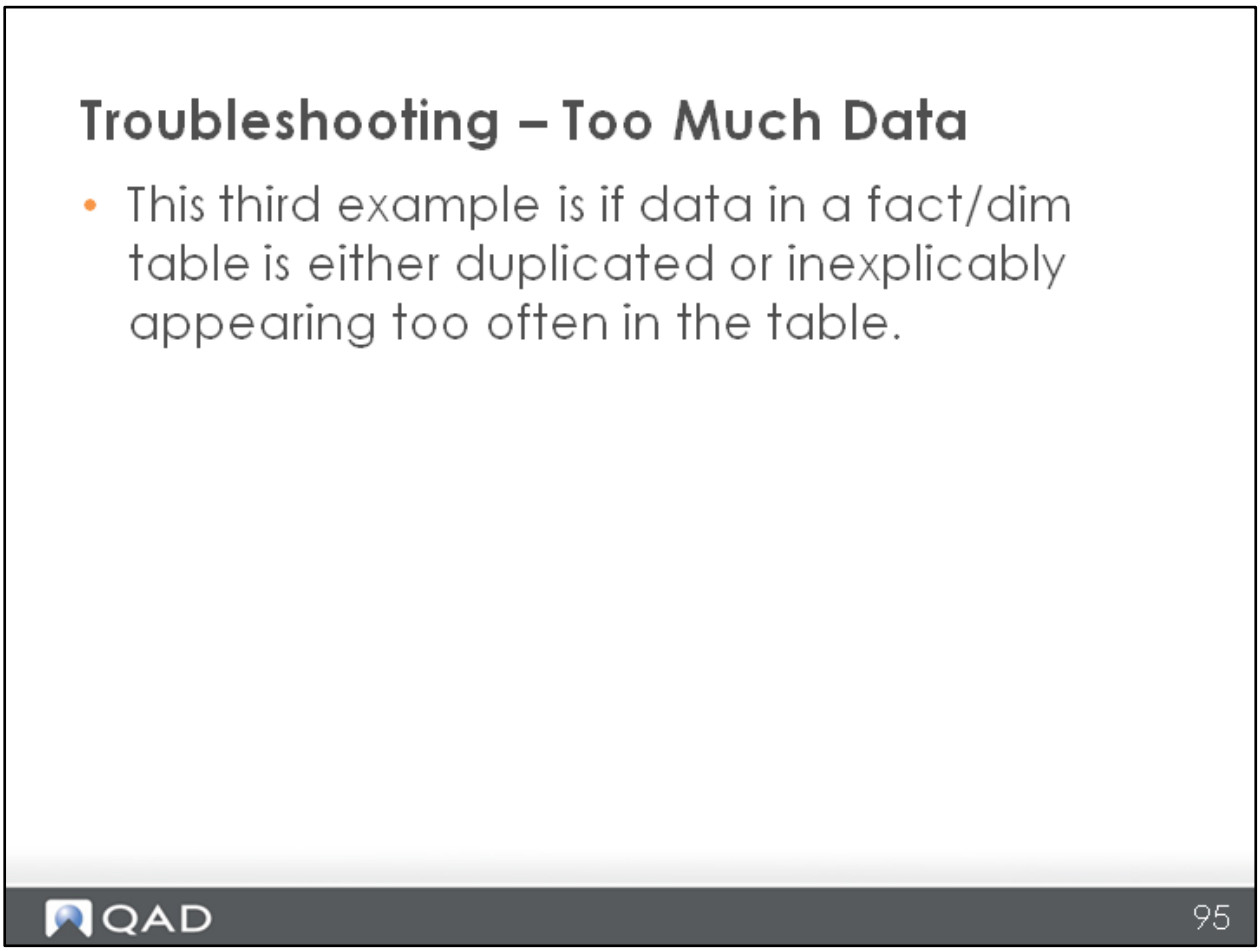
Troubleshooting – Incorrect Data



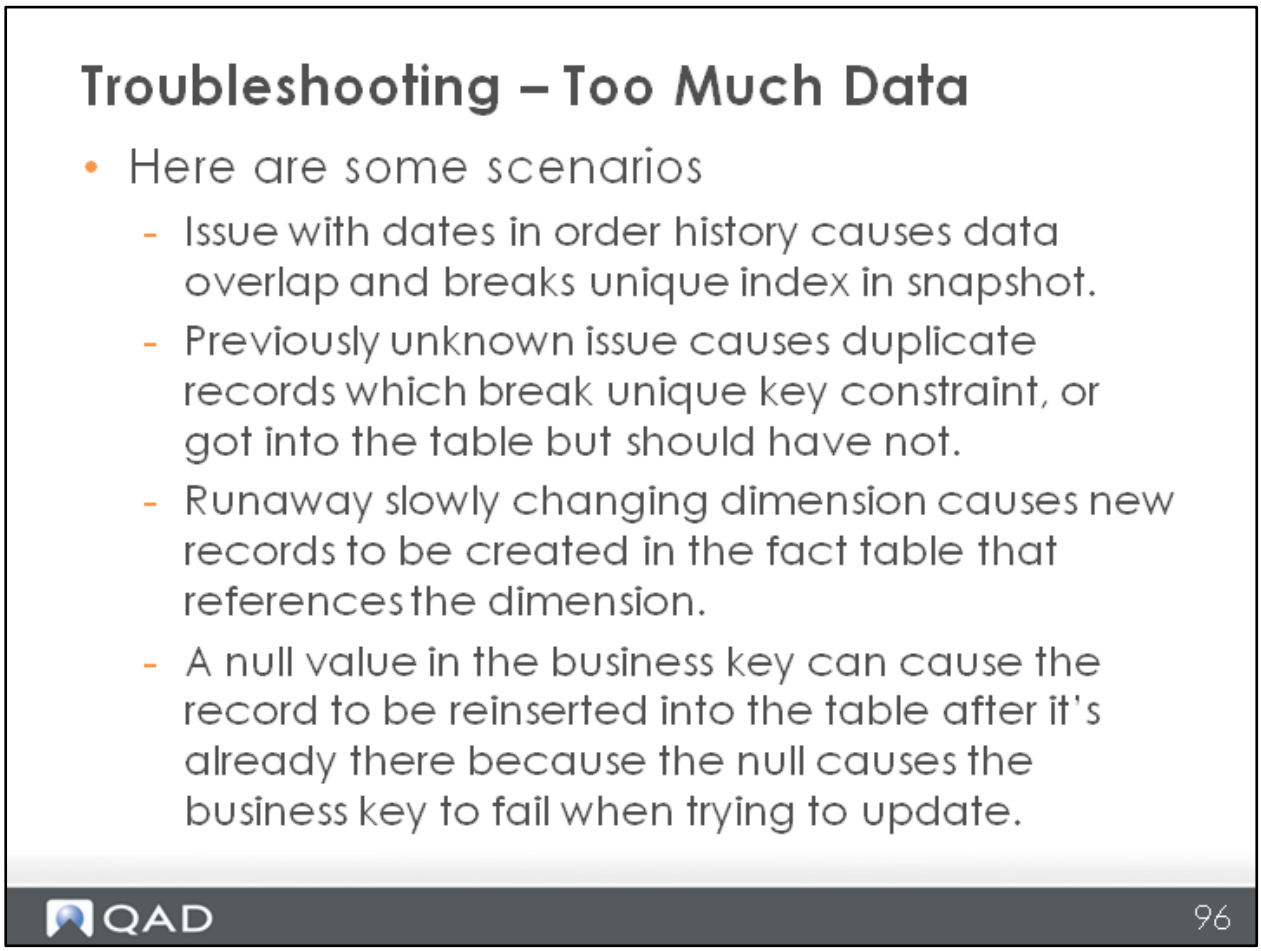
Troubleshooting – Incorrect Data



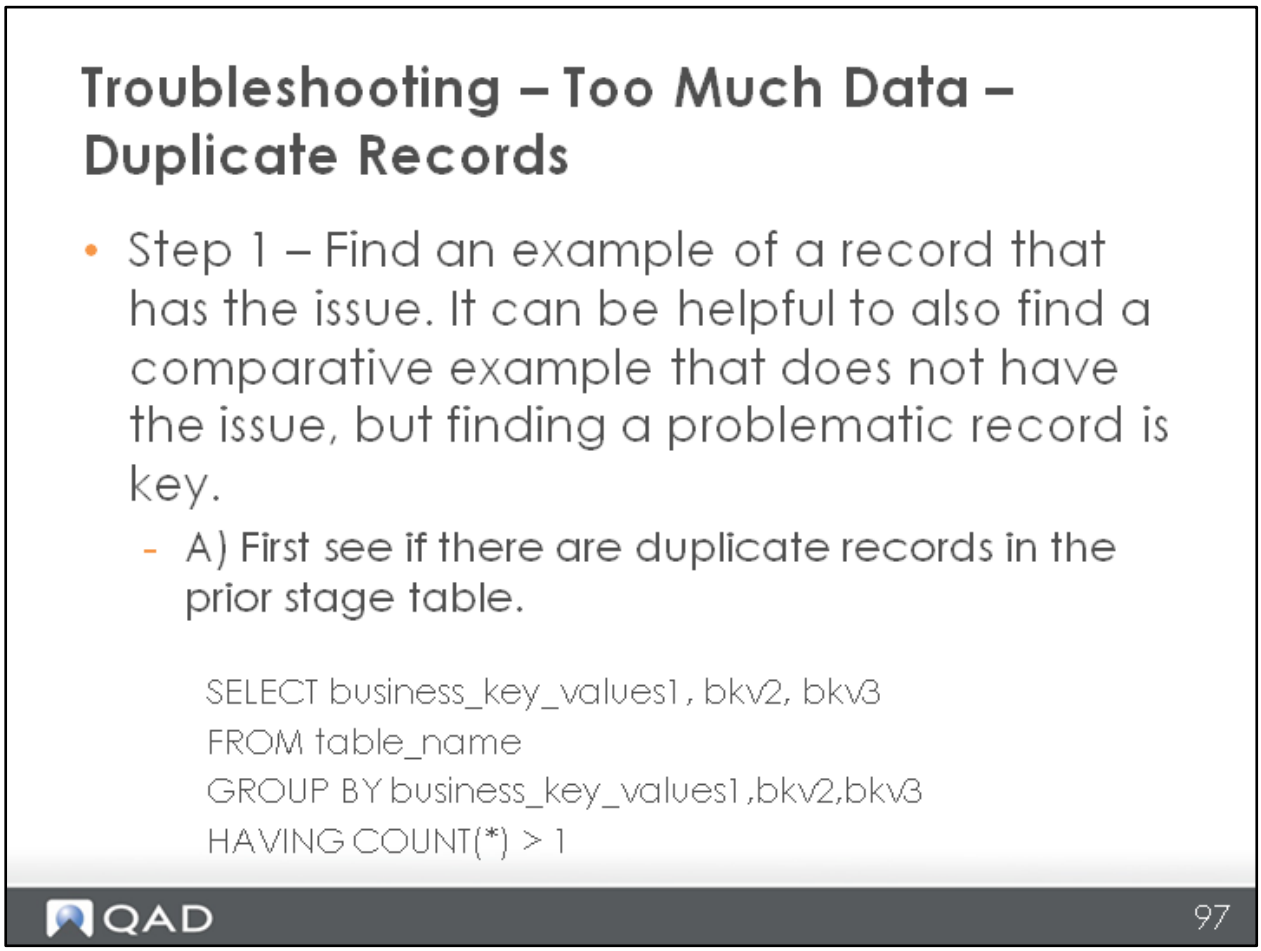
Troubleshooting – Too Much Data



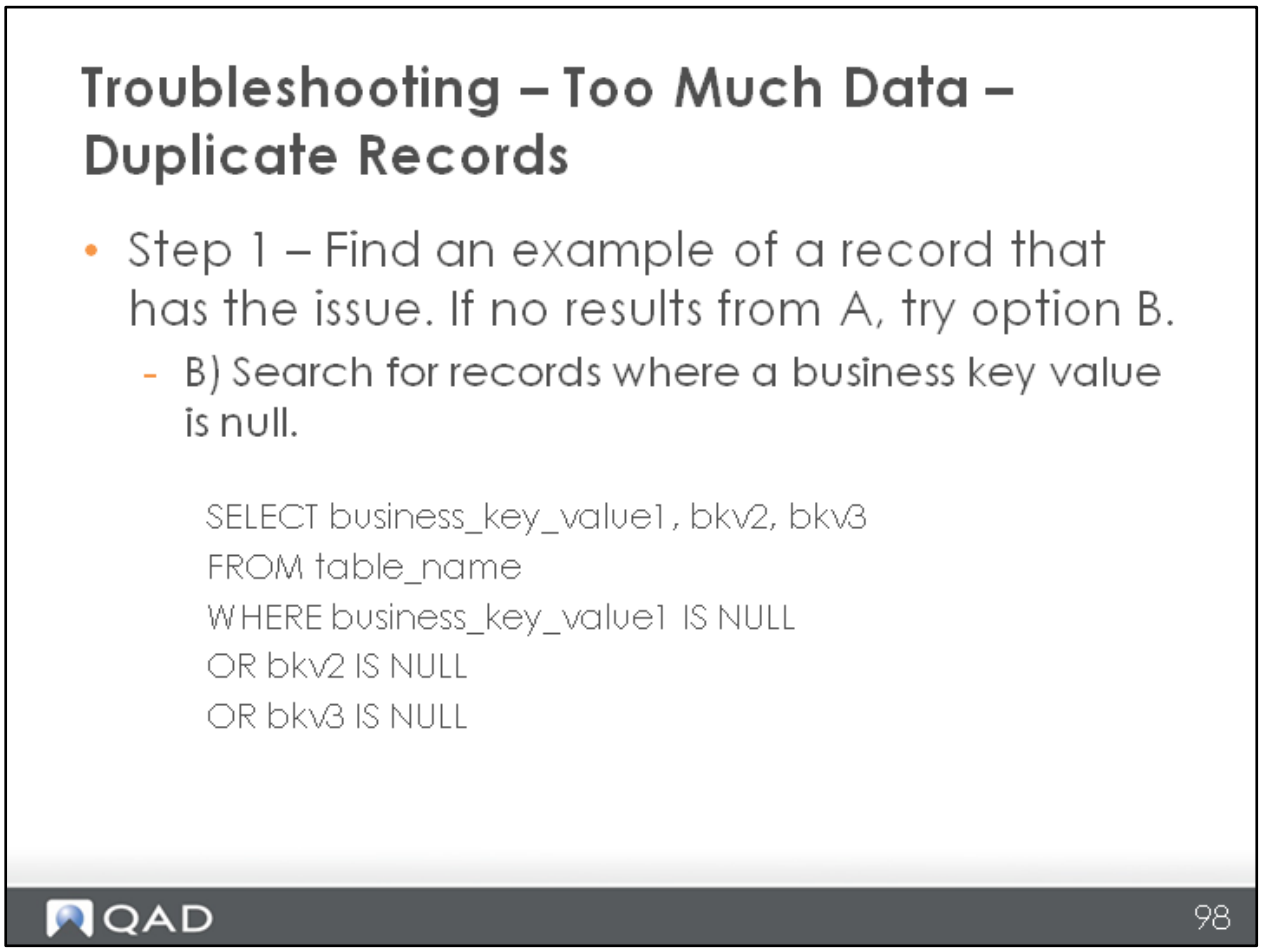
Troubleshooting – Too Much Data



Troubleshooting – Too Much Data – Duplicate Records

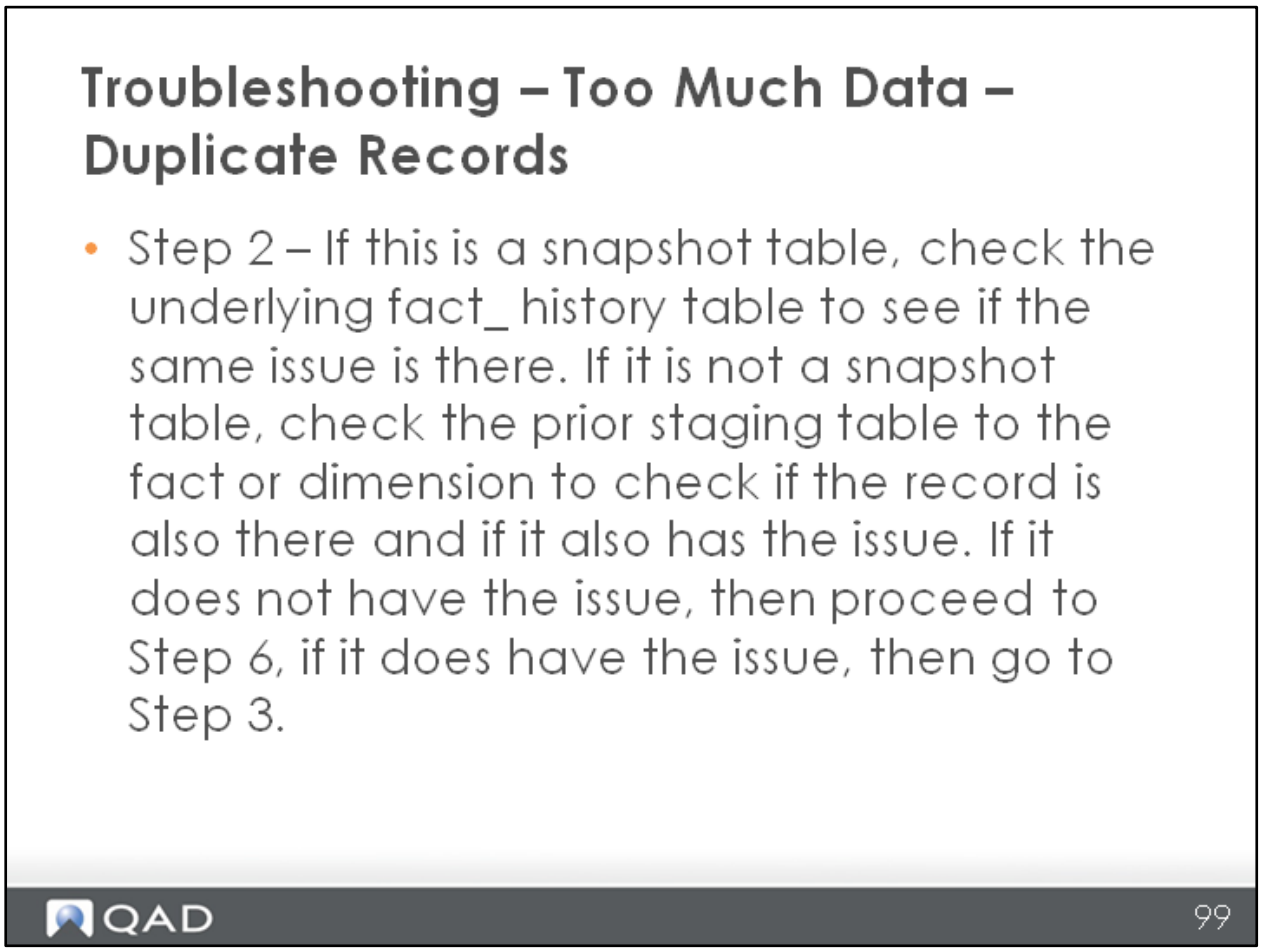


Troubleshooting – Too Much Data – Duplicate Records



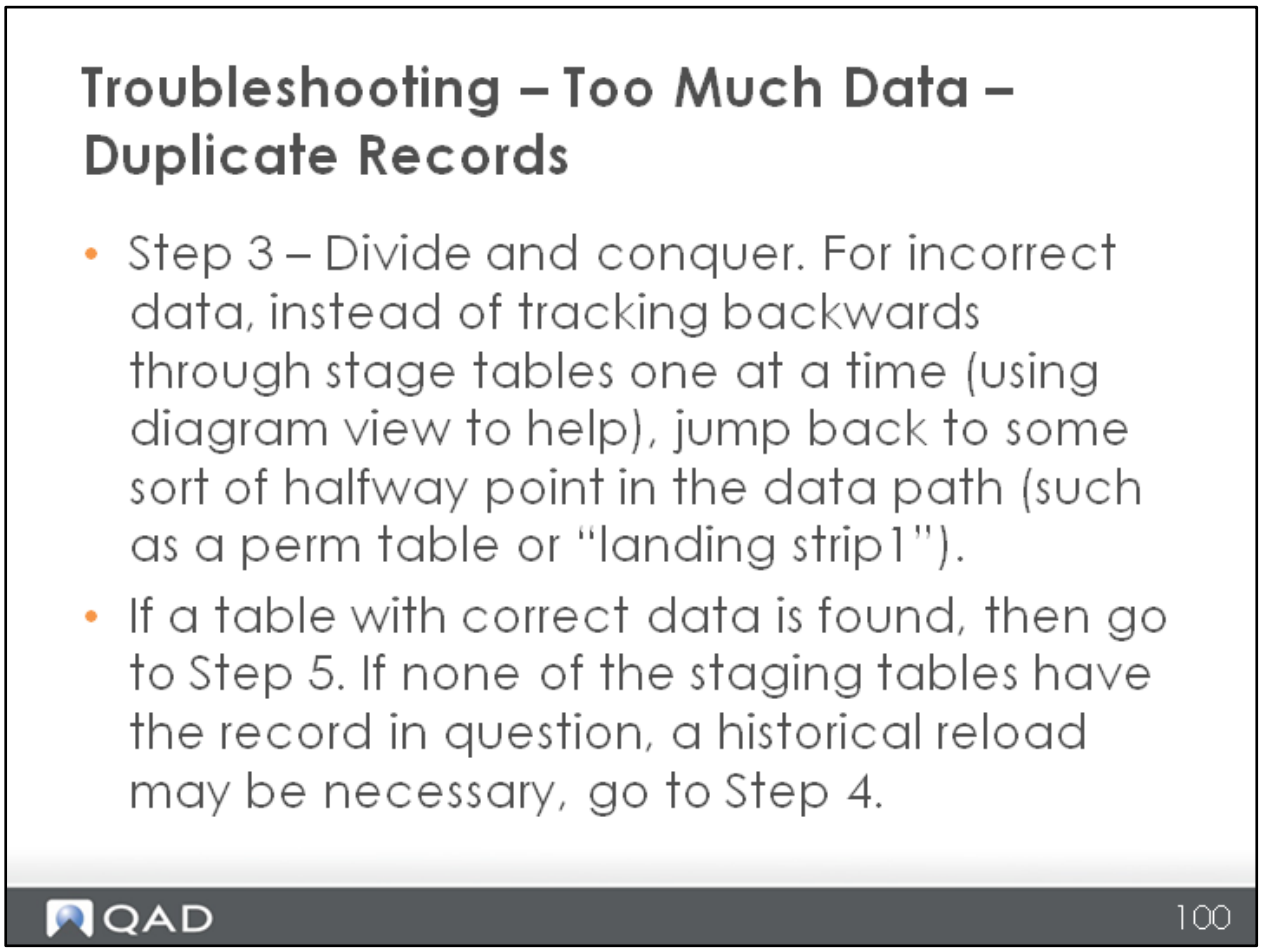
Business key values should never be null. If they are, there is a problem with either the data or the code.

Troubleshooting – Too Much Data – Duplicate Records



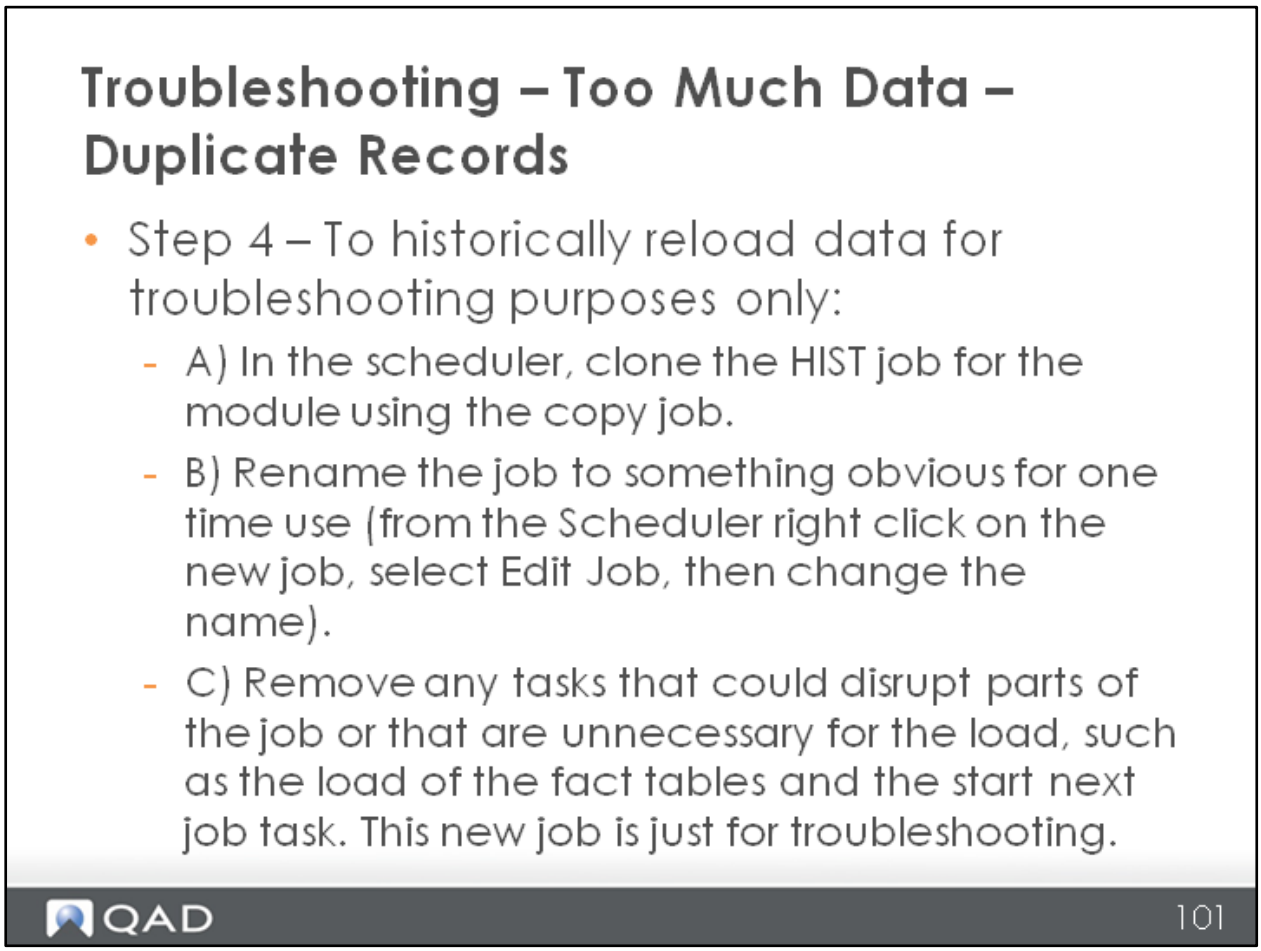
In earlier versions of BI (pre 3.5), there were quite a few problems with sliding historical date parameters and overlapping begin and end dates in the OM and PO order history tables. These would inevitably cause snapshot tables to fail when two same records with overlapping begin and end dates would get picked up by the snapshot table. The solution for these was to either delete one of the records or better still, set the begin and end dates of the records so they were no longer overlapping and wouldn't get picked up by the snapshot for the same end of month/week date.

Troubleshooting – Too Much Data – Duplicate Records

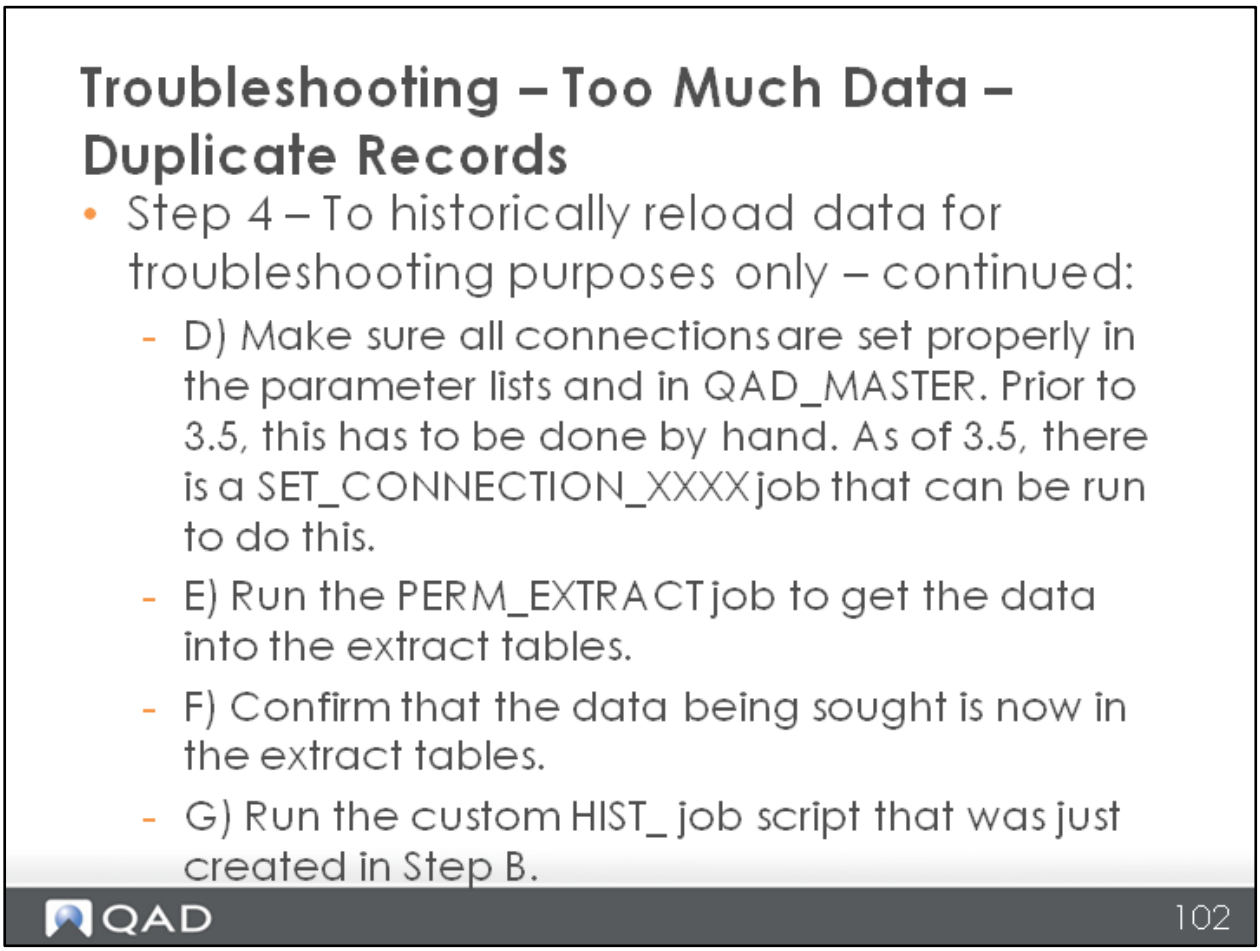


If the column track back shows the data passes through a perm table, check the perm table to see if the record is there and if it is correct or not. Or at least jump back to the first table in the "landing strip" to the fact table. The "landing strip" tables are the final stage tables prior to a fact table that have all the records headed to the fact collated, but still need to calculate currency conversion rates and get dimension keys, etc. Examples would be stage_om_order1, stage_om_order2, stage_om_order3, stage_om_order.

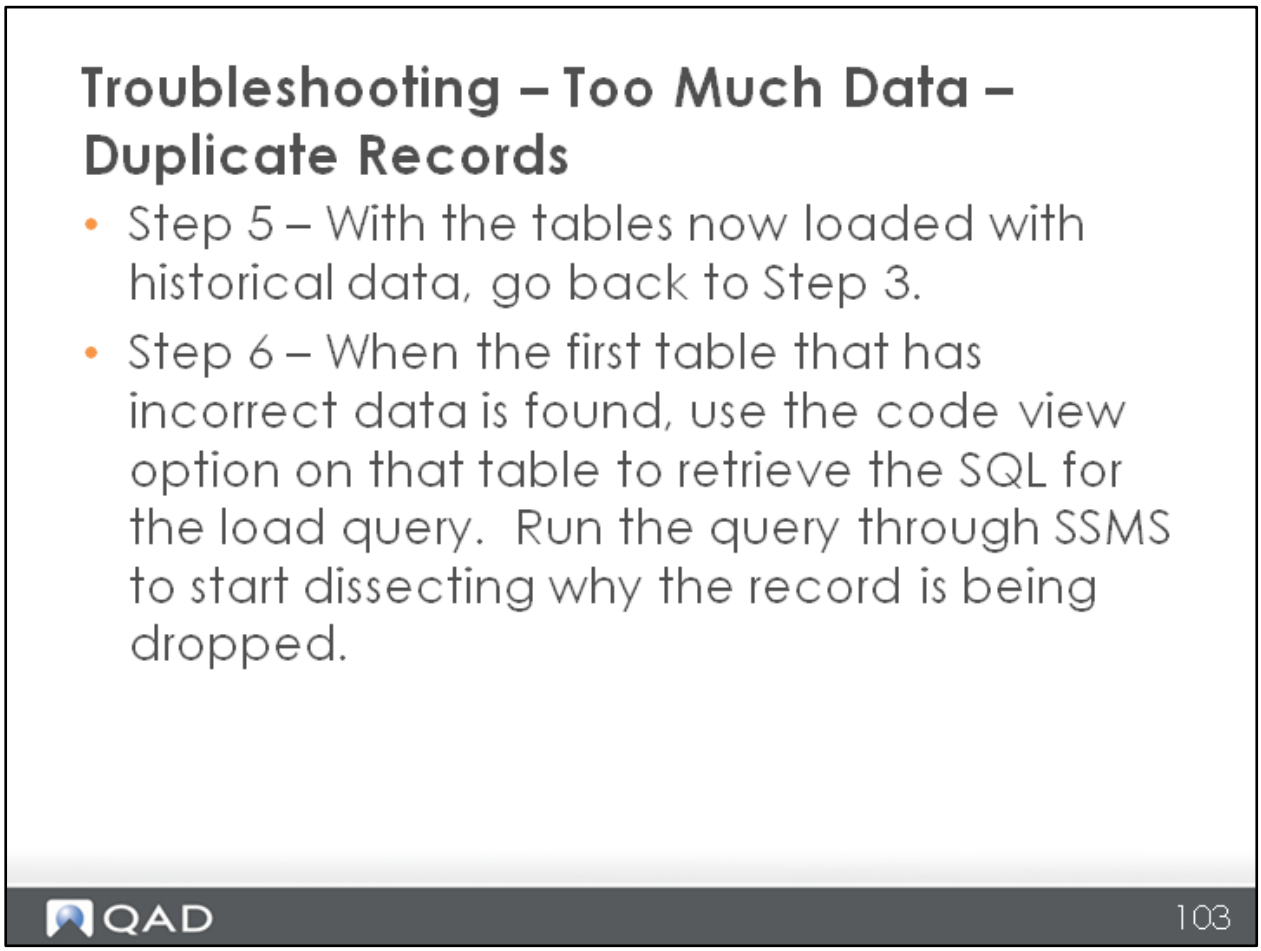
Troubleshooting – Too Much Data – Duplicate Records



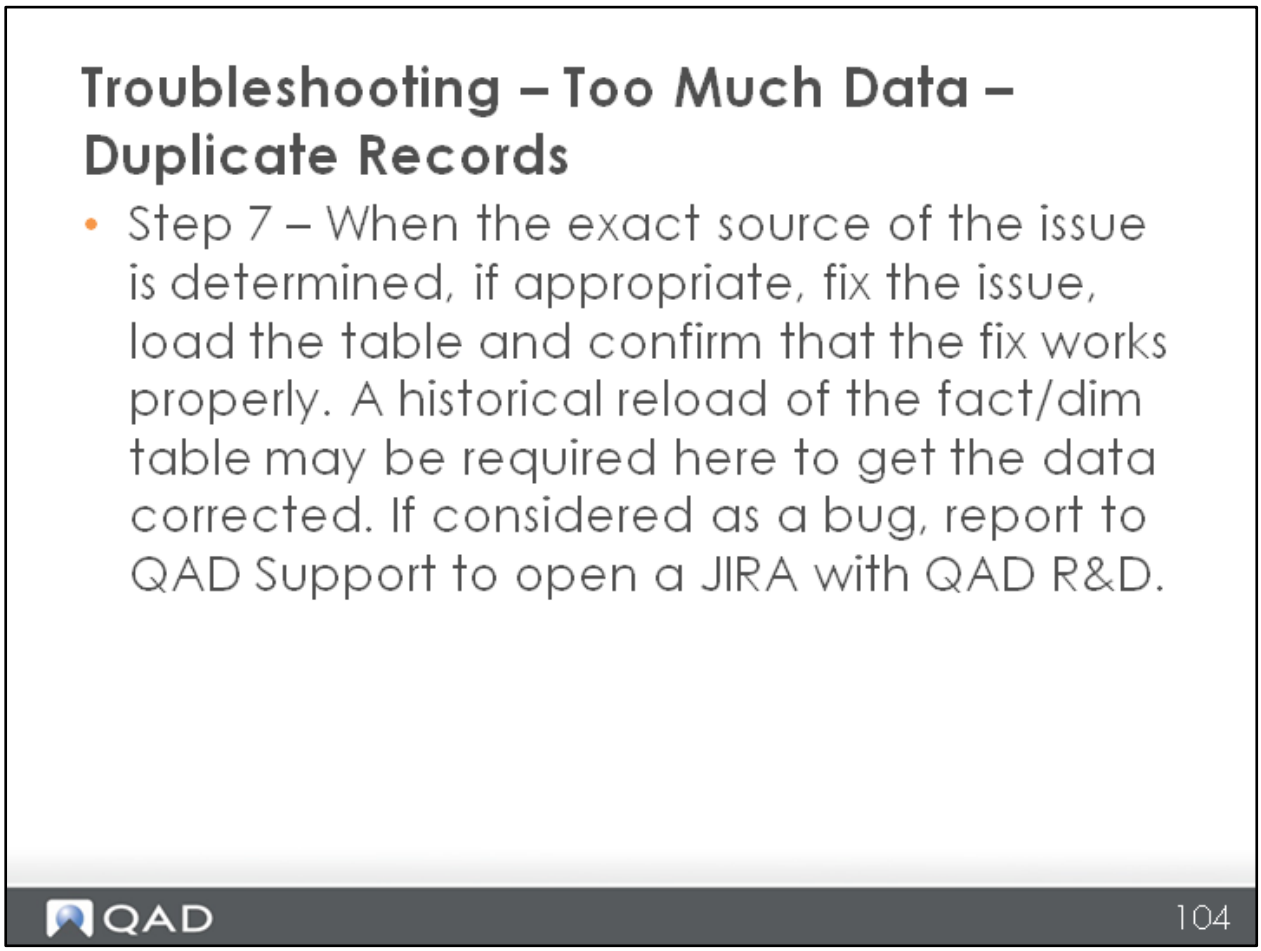
Troubleshooting – Too Much Data – Duplicate Records



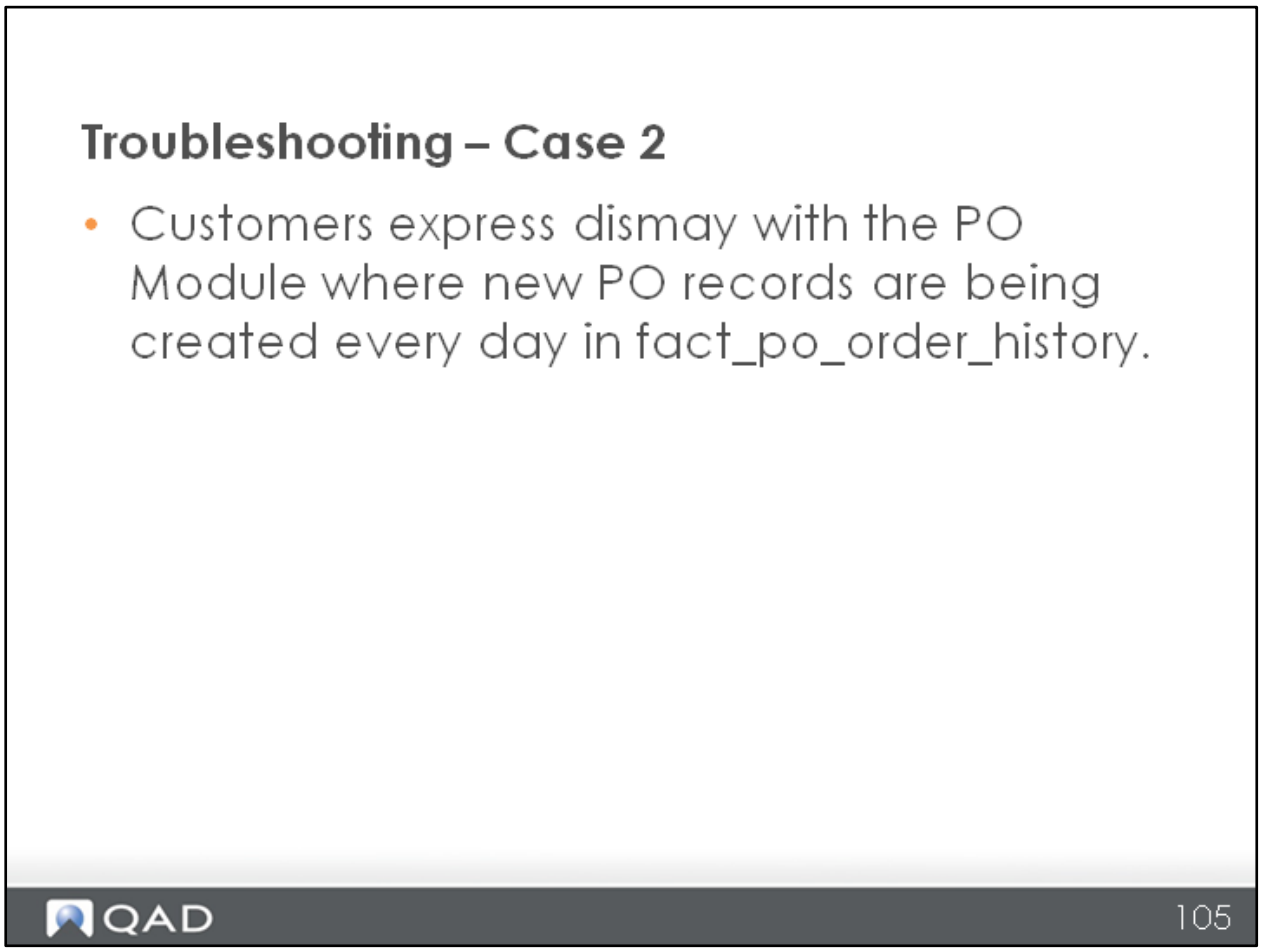
Troubleshooting – Too Much Data – Duplicate Records



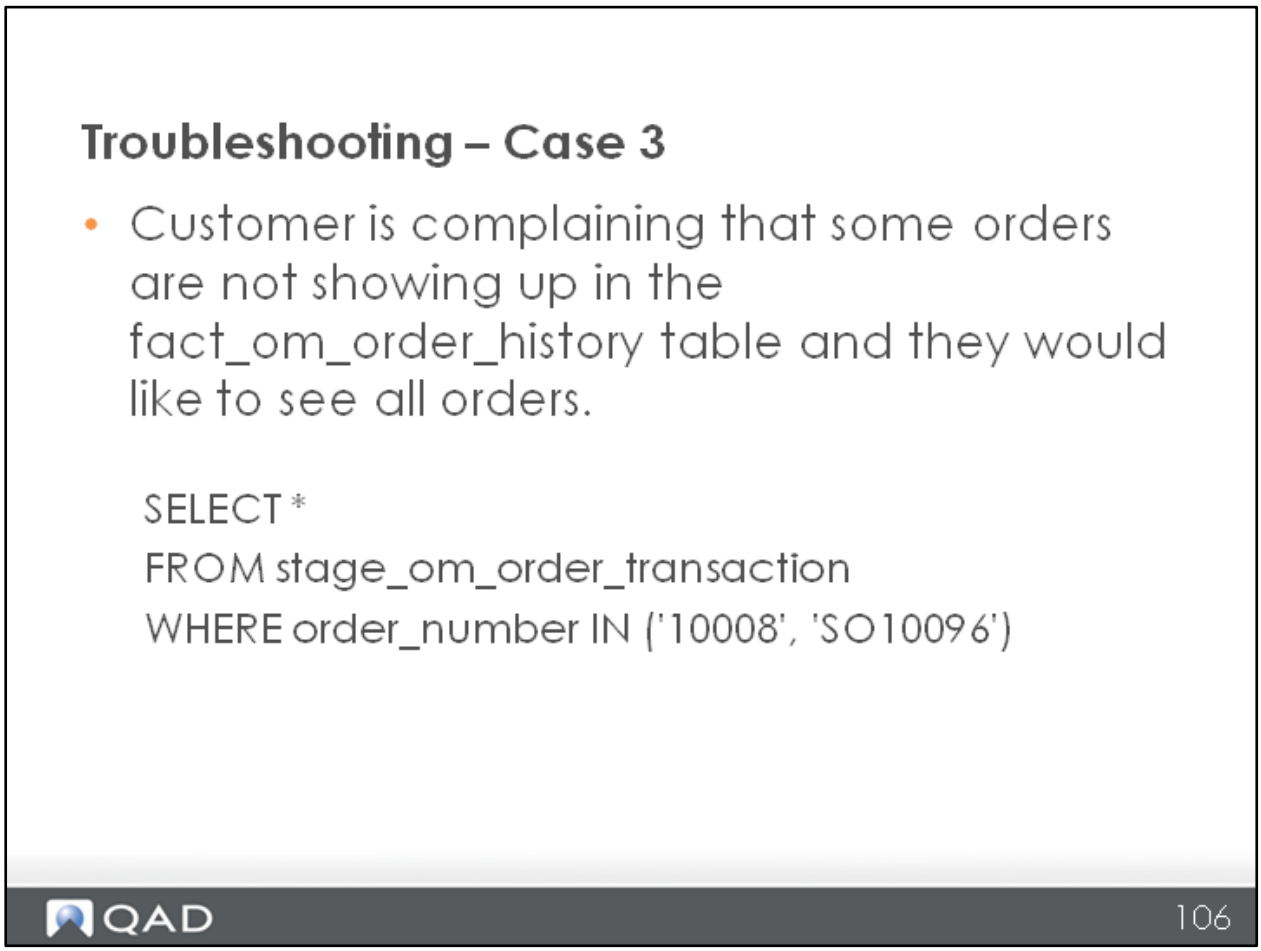
Troubleshooting – Too Much Data – Duplicate Records



Troubleshooting – Case 2

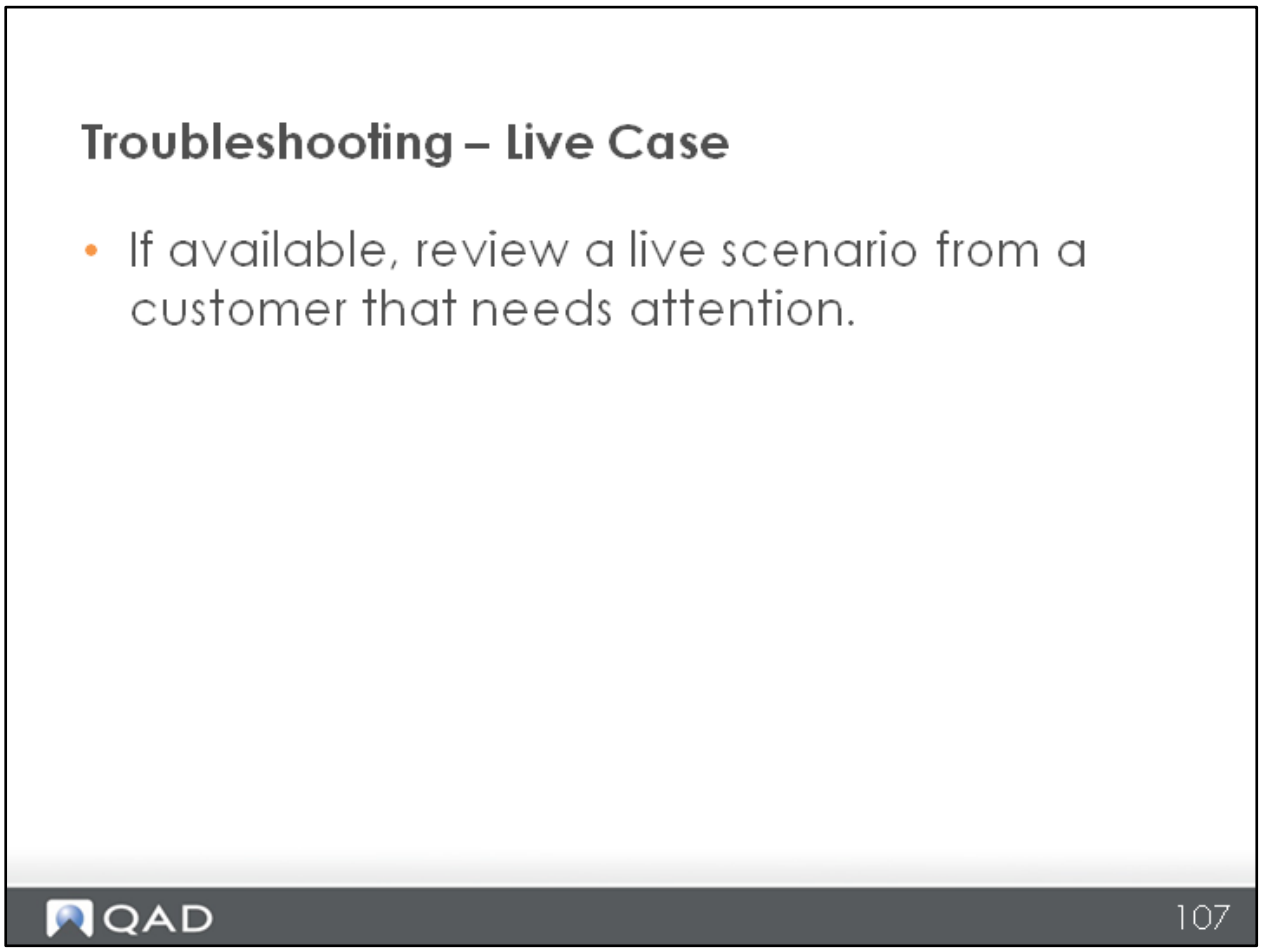


Troubleshooting – Case 3



In this case, we need to figure out why some orders are not showing up in the fact_om_order_history table, while others are.

Troubleshooting – Live Case



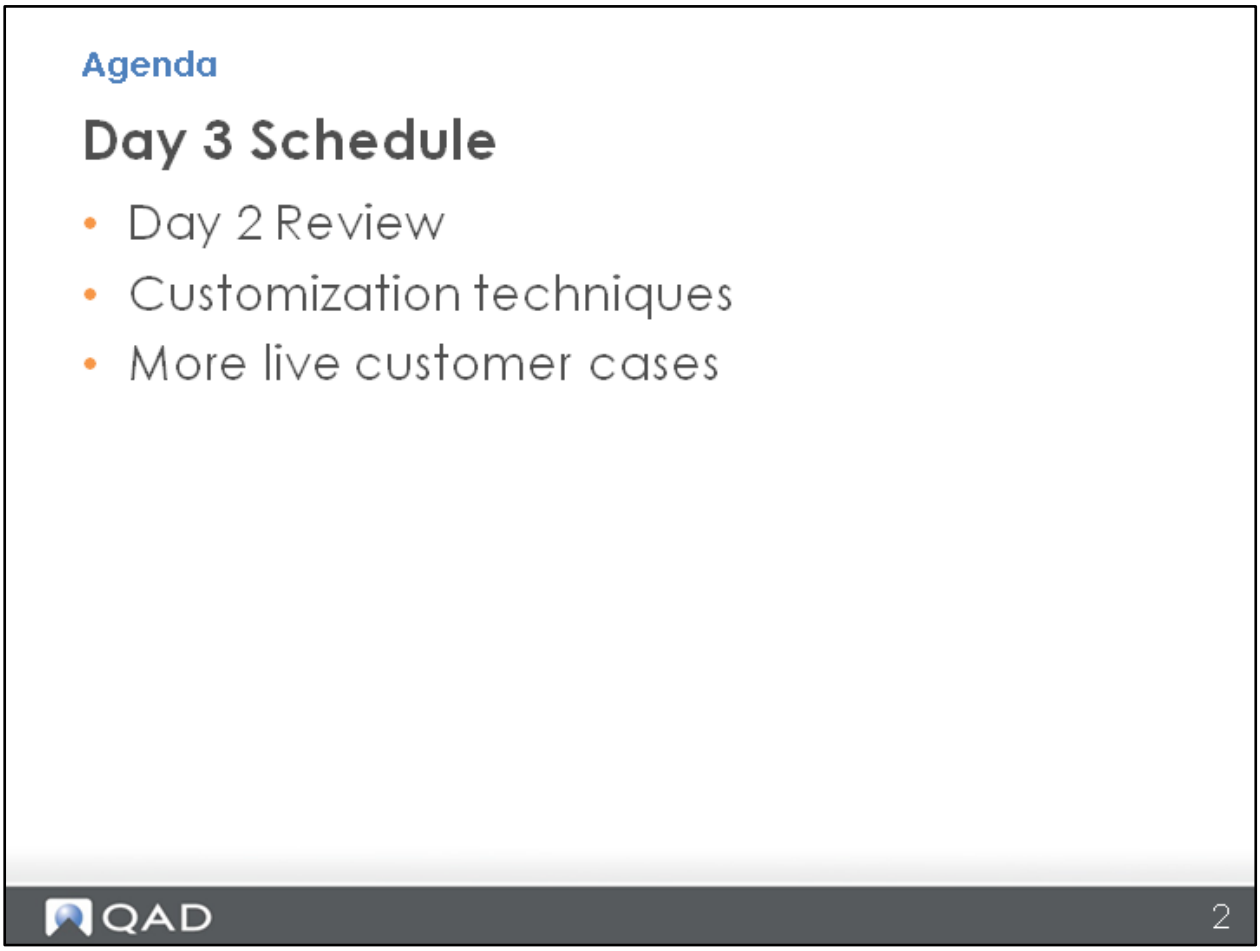
CHAPTER 3

QAD BI v3 - Tech Level 2 Certification - Part 3

QAD BI v3 – Technical Level II Certification Class – Part 3



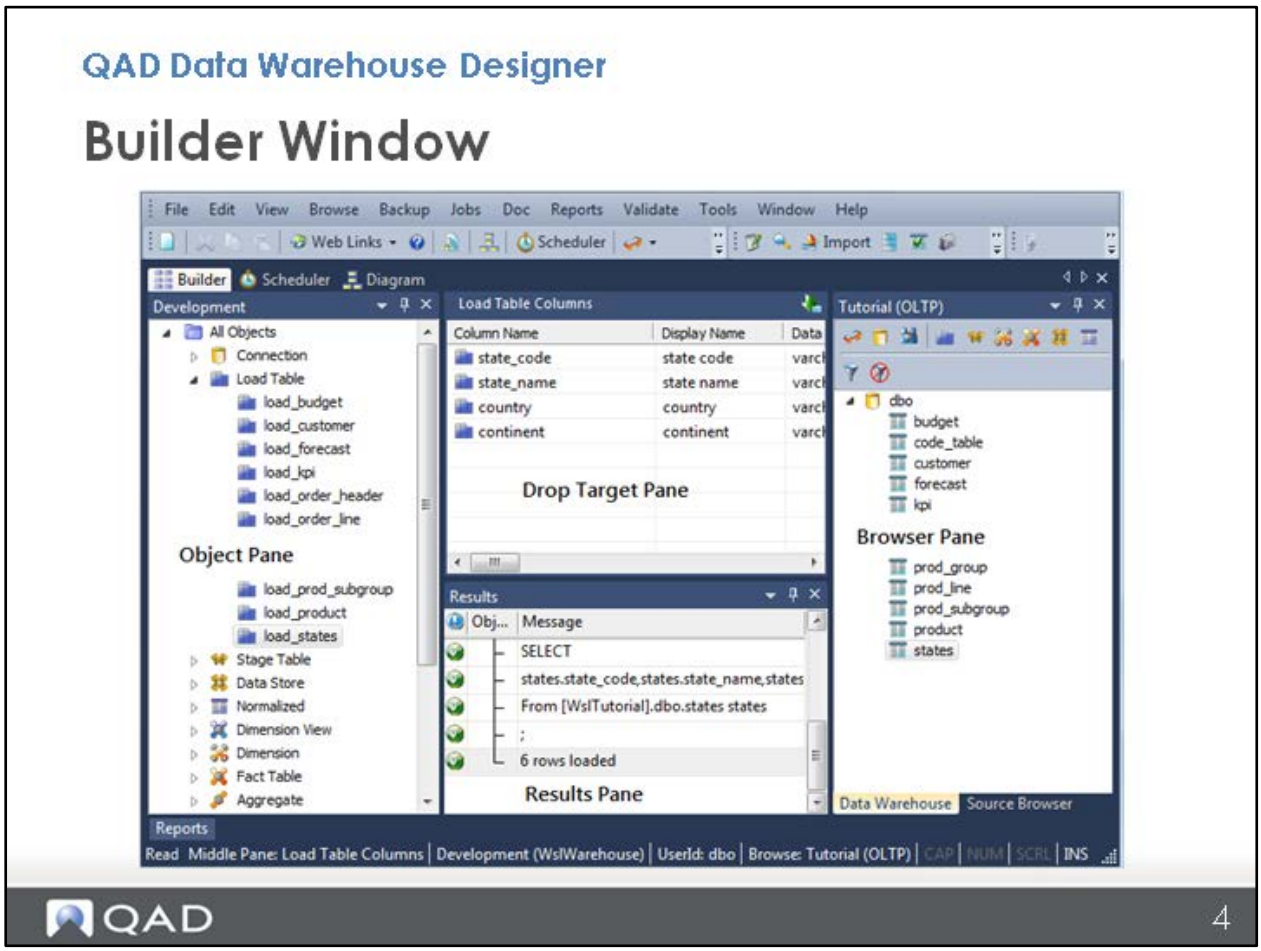
Day 3 Schedule



QAD Data Warehouse Designer



Builder Window



In earlier versions of the QAD Designer (DWD), the Builder window is called the Browser window.

Also referred to as:

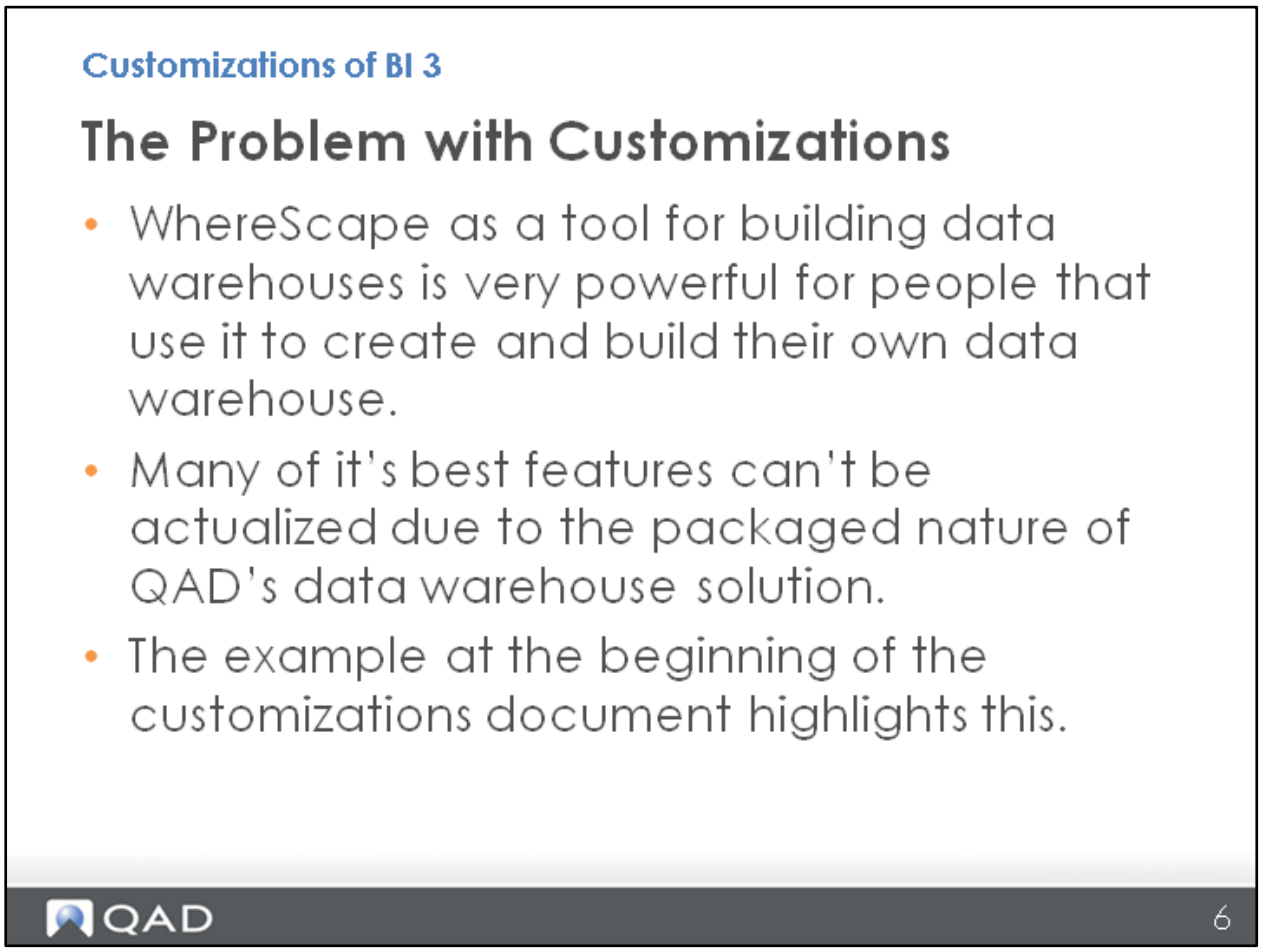
Object Pane = Object Browser Pane

Browser Pane = Source Browser Pane

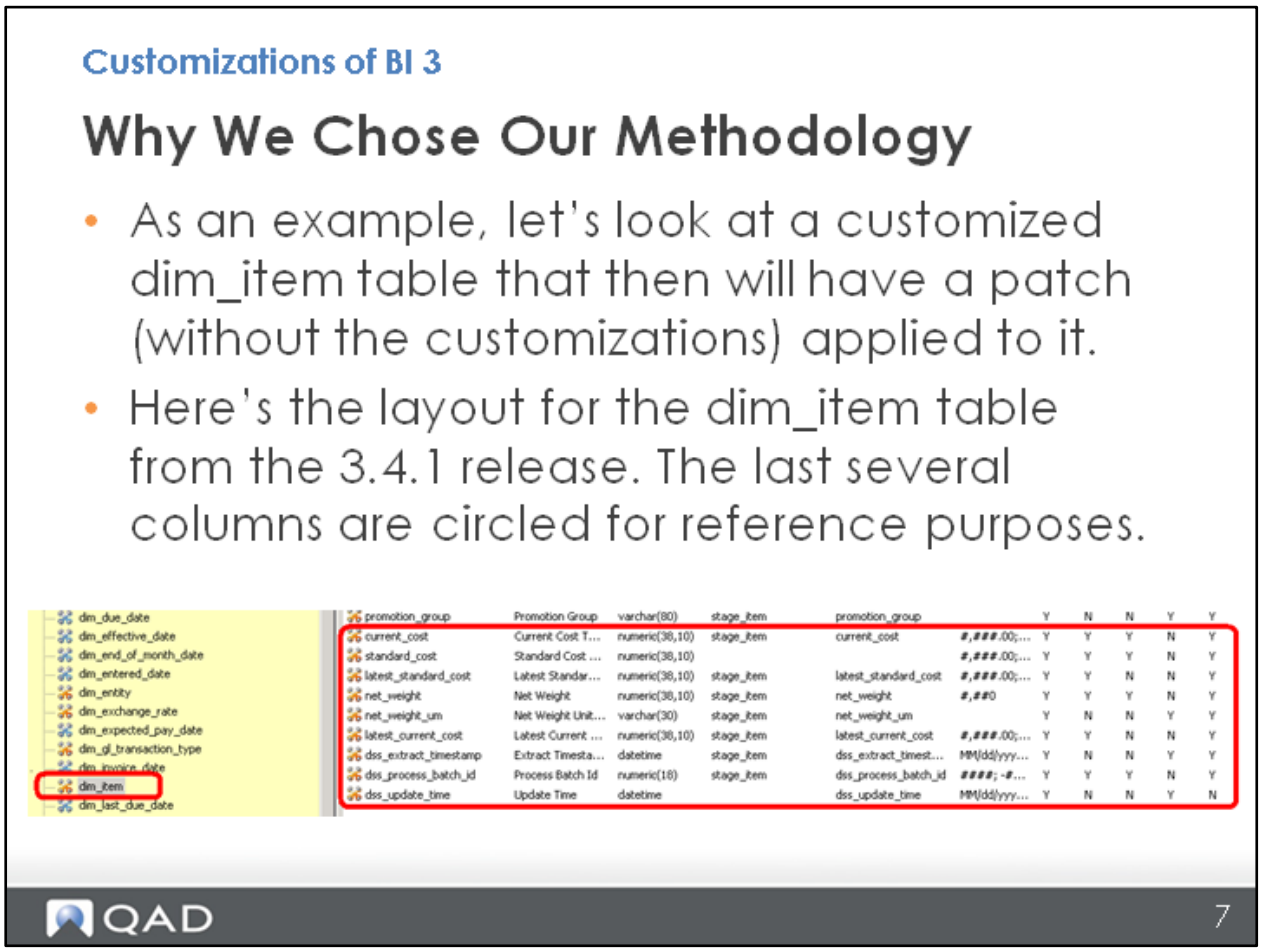
Customizations of BI v3



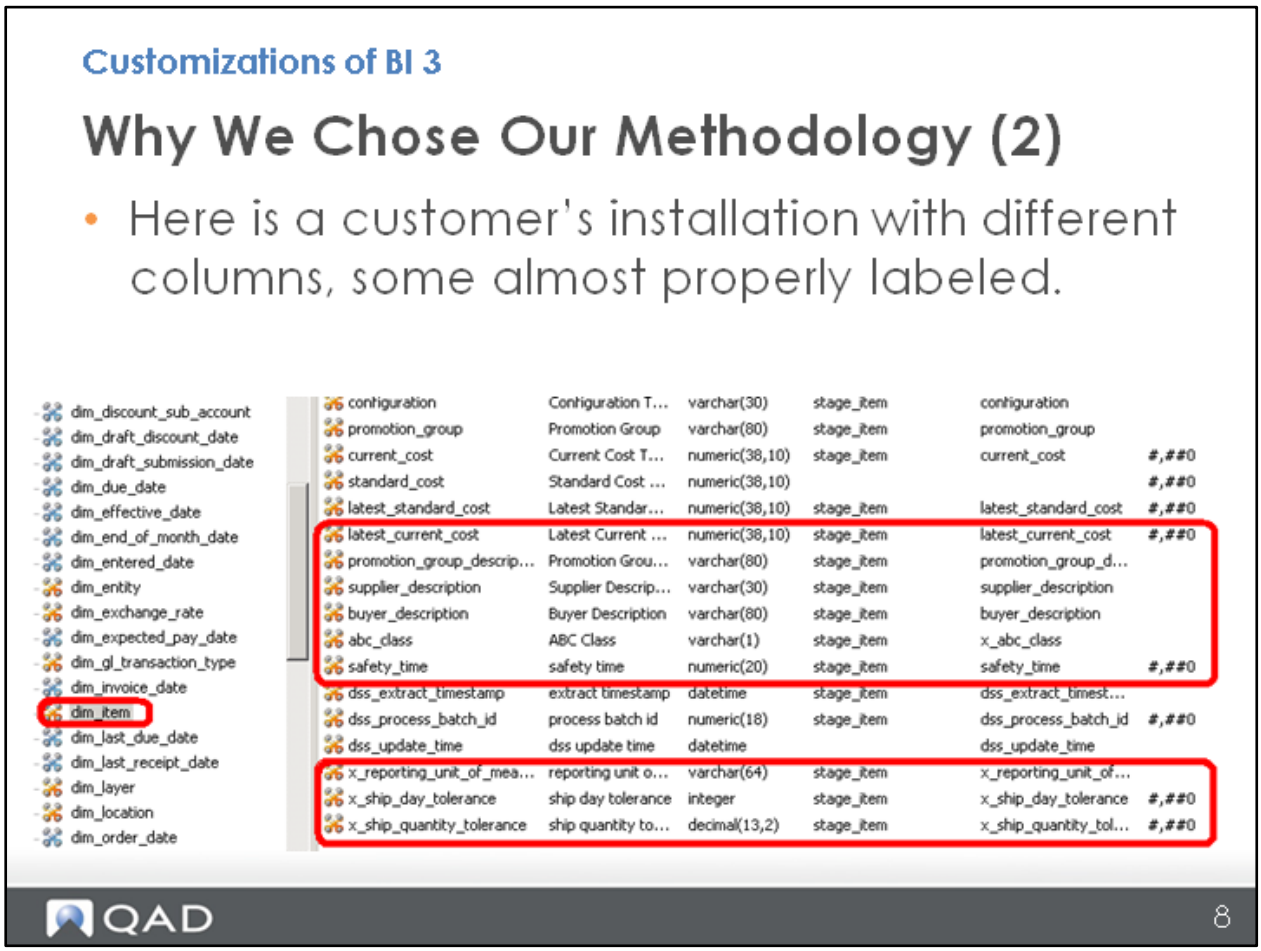
The Problem with Customizations



Why We Chose Our Methodology

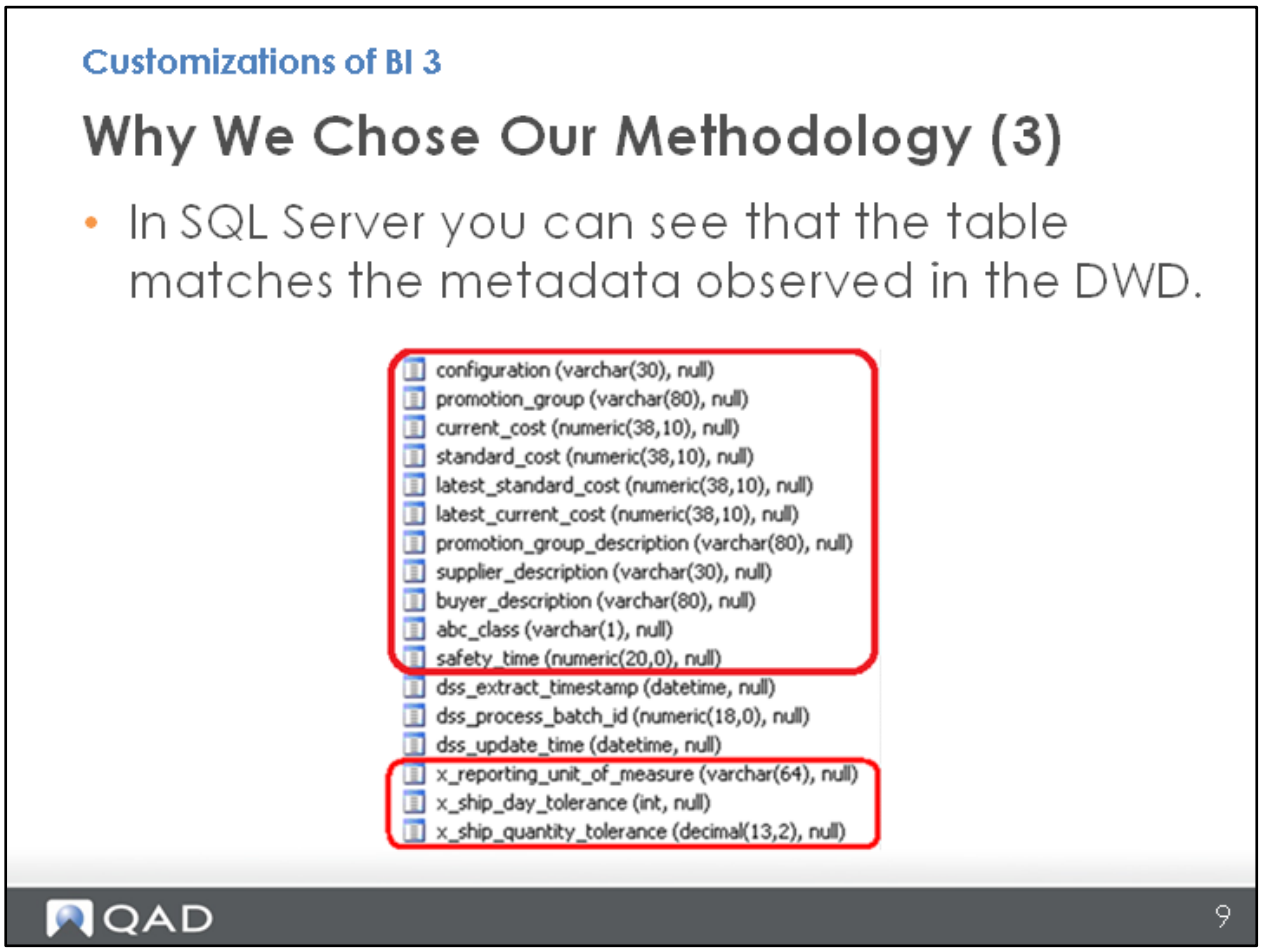


Why We Chose Our Methodology (2)



Here's the layout for the dim_item table for a customer that has a 3.3 customized version of the modules that not only has the easily identifiable custom columns that begin with an x_ prefix (our preferred prefix for custom columns now is xx_), but also note that they have some other columns that don't match our 3.4.1 dim_item table. These had been added at time of installation with the belief that these same columns would be added to the code base. That ended up not yet happening, so an upgrade to 3.4.1 is likely going to require even more work and investigation. Unless columns are already in the code base, treat any additions as custom.

Why We Chose Our Methodology (3)



Why We Chose Our Methodology (4)

Customizations of BI 3

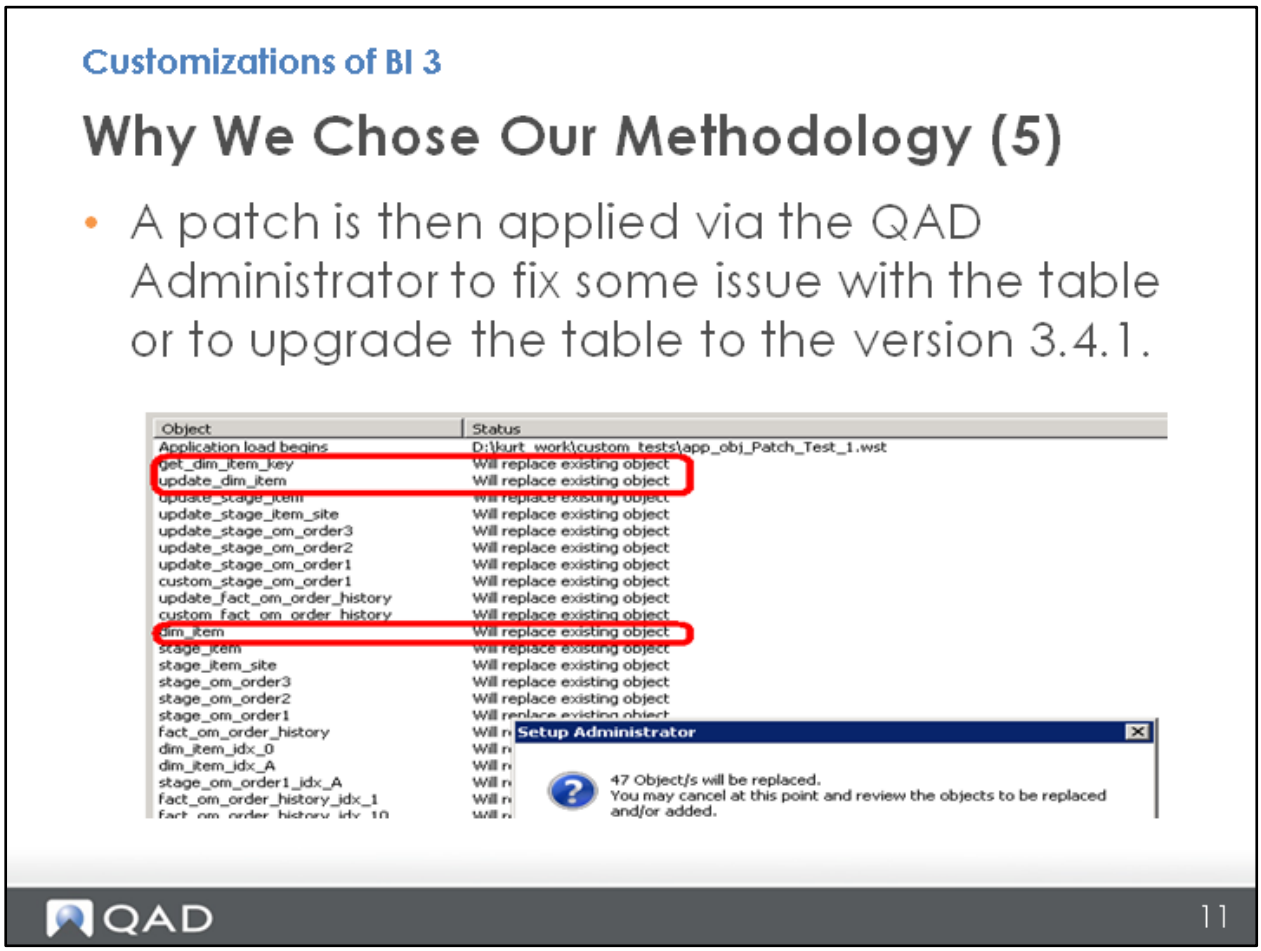
Why We Chose Our Methodology (4)

- And if we open the affiliated procedure, in the DWD, we can see what columns are being inserted into and that they match the columns listed in the DWD.

```
, current_cost                          current_cost
, 0                                     standard_cost
, latest_standard_cost                  latest_standard_cost
, latest_current_cost                   latest_current_cost
, promotion_group_description           promotion_group_description
, supplier_description                  supplier_description
, buyer_description                     buyer_description
, x_abc_class                           abc_class
, safety_time                           safety_time
, dss_extract_timestamp                 dss_extract_timestamp
, dss_process_batch_id                  dss_process_batch_id
, x_reporting_unit_of_measure           x_reporting_unit_of_measure
, x_ship_day_tolerance                  x_ship_day_tolerance
, x_ship_quantity_tolerance             x_ship_quantity_tolerance
FROM
  stage_item
WHERE NOT EXISTS (
  SELECT 1 FROM dim_item
  WHERE ( stage_item.source_system_code = dim_item.source_system_code
    OR ( stage_item.source_system_code IS NULL AND dim_item.source_system_code IS NULL ))
  AND ( stage_item.domain_code = dim_item.domain_code
    OR ( stage_item.domain_code IS NULL AND dim_item.domain_code IS NULL ))
```

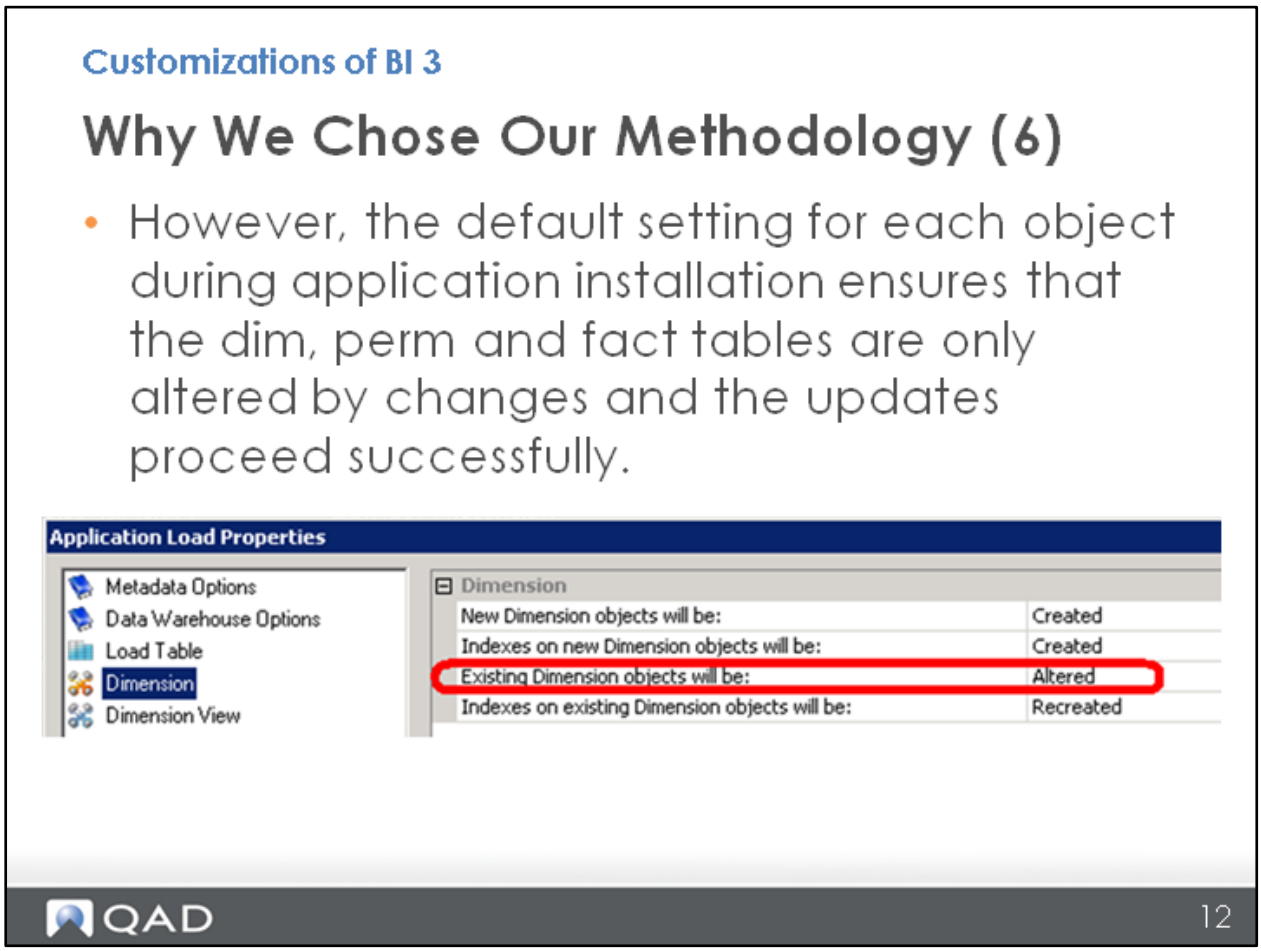
QAD 10

Why We Chose Our Methodology (5)



You can see that it initially says that existing objects will be replaced.

Why We Chose Our Methodology (6)



Why We Chose Our Methodology (7)

Customizations of BI 3

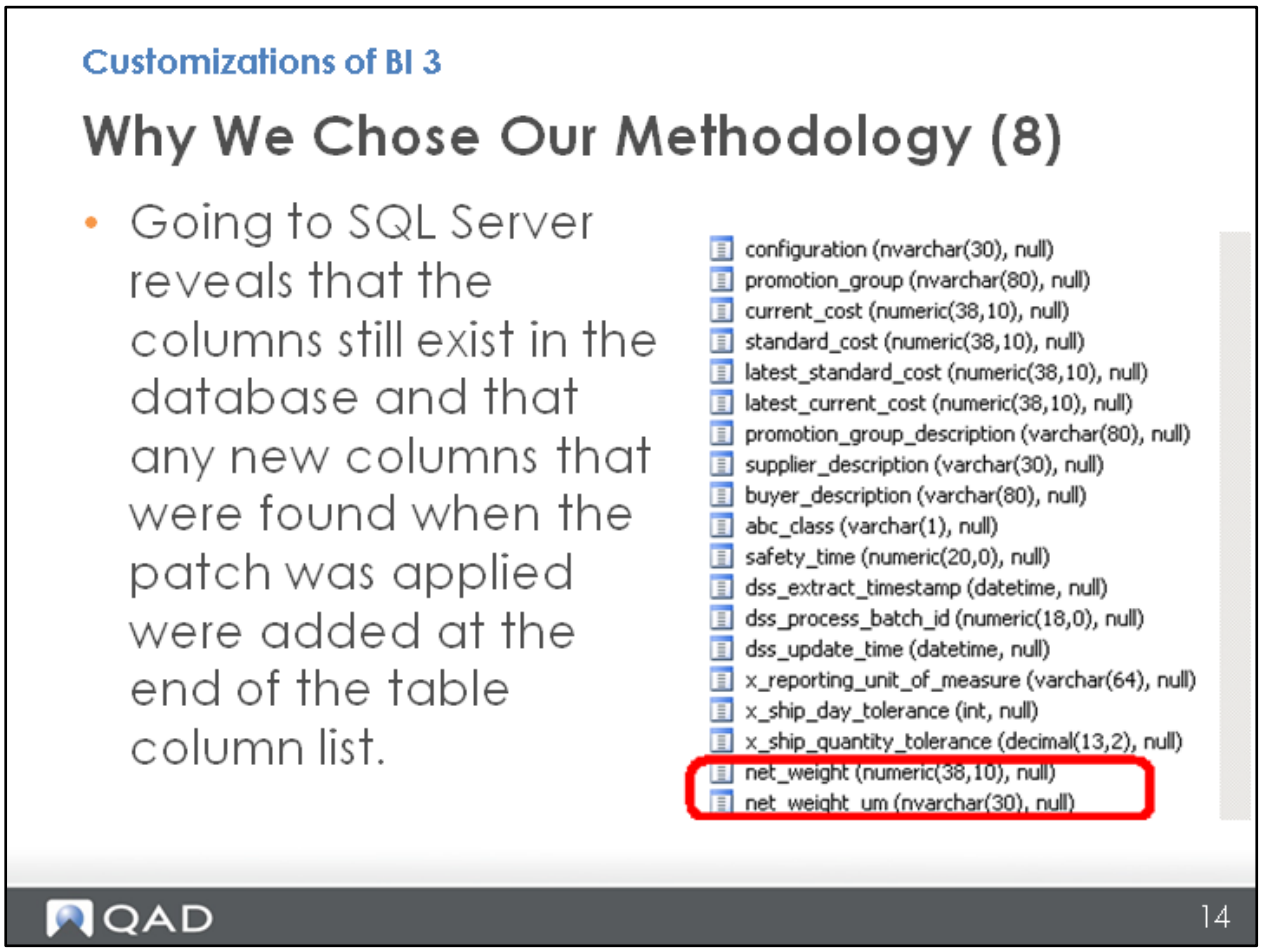
Why We Chose Our Methodology (7)

- A refresh of the DWD Builder Pane to review the changes to dim_item then reveals that the data in the tables looks exactly like 3.4.1 version of dim_item. Any other columns that had existed for this table in the DWD's metadata are no longer shown.



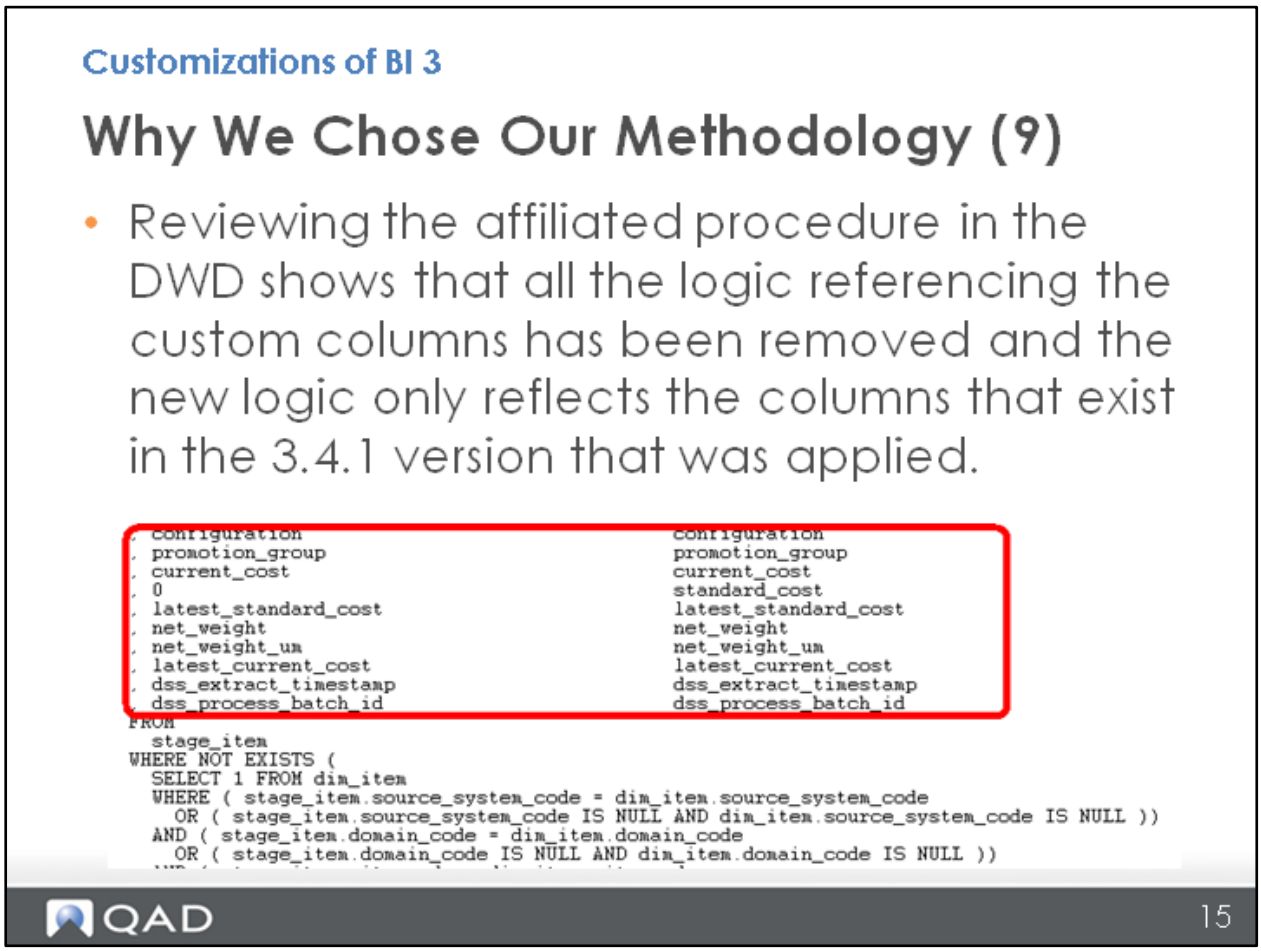
QAD 13

Why We Chose Our Methodology (8)



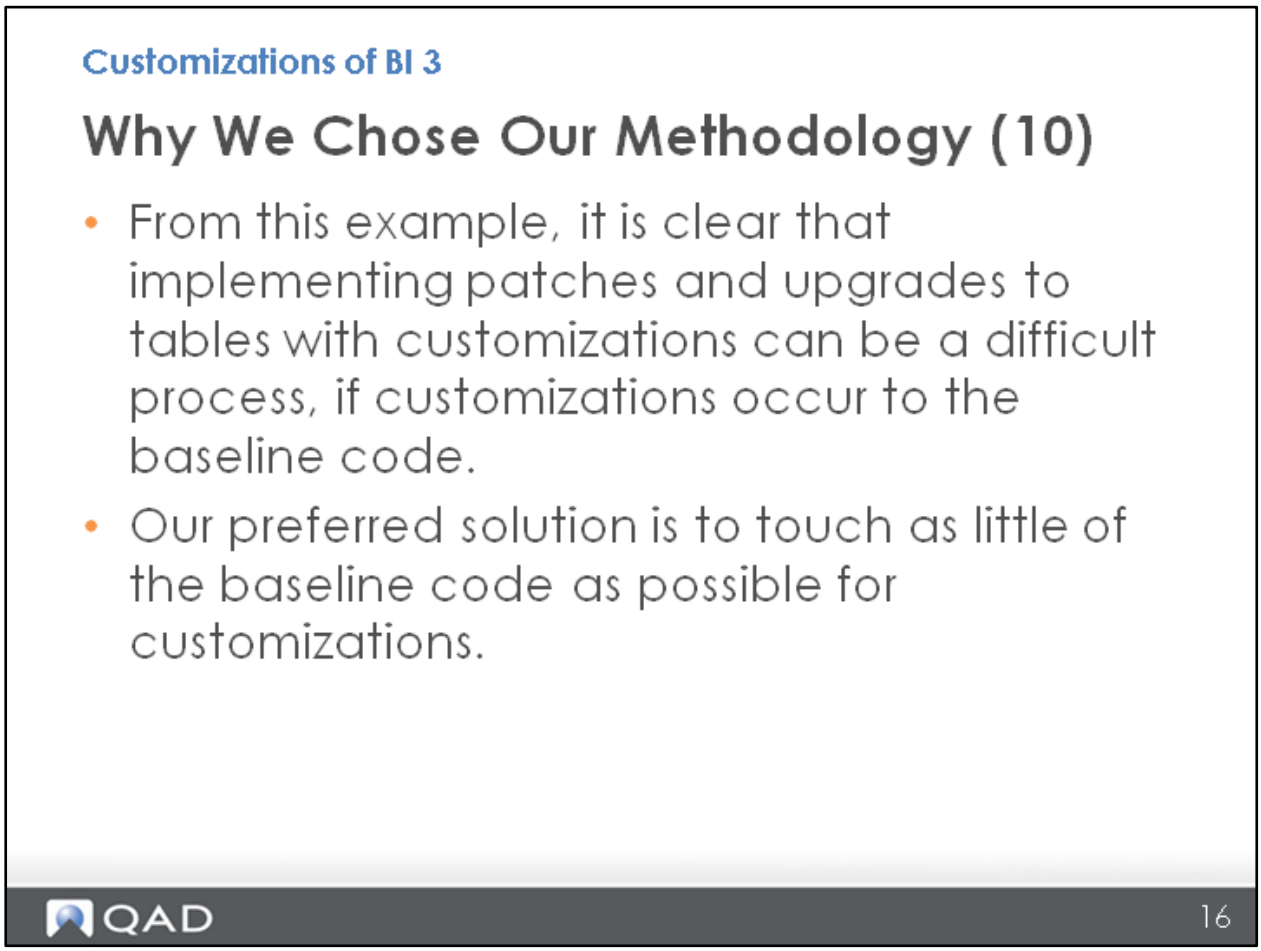
When the patch was applied, we had selected to not change existing table, only alter them. Why are the other columns gone?

Why We Chose Our Methodology (9)



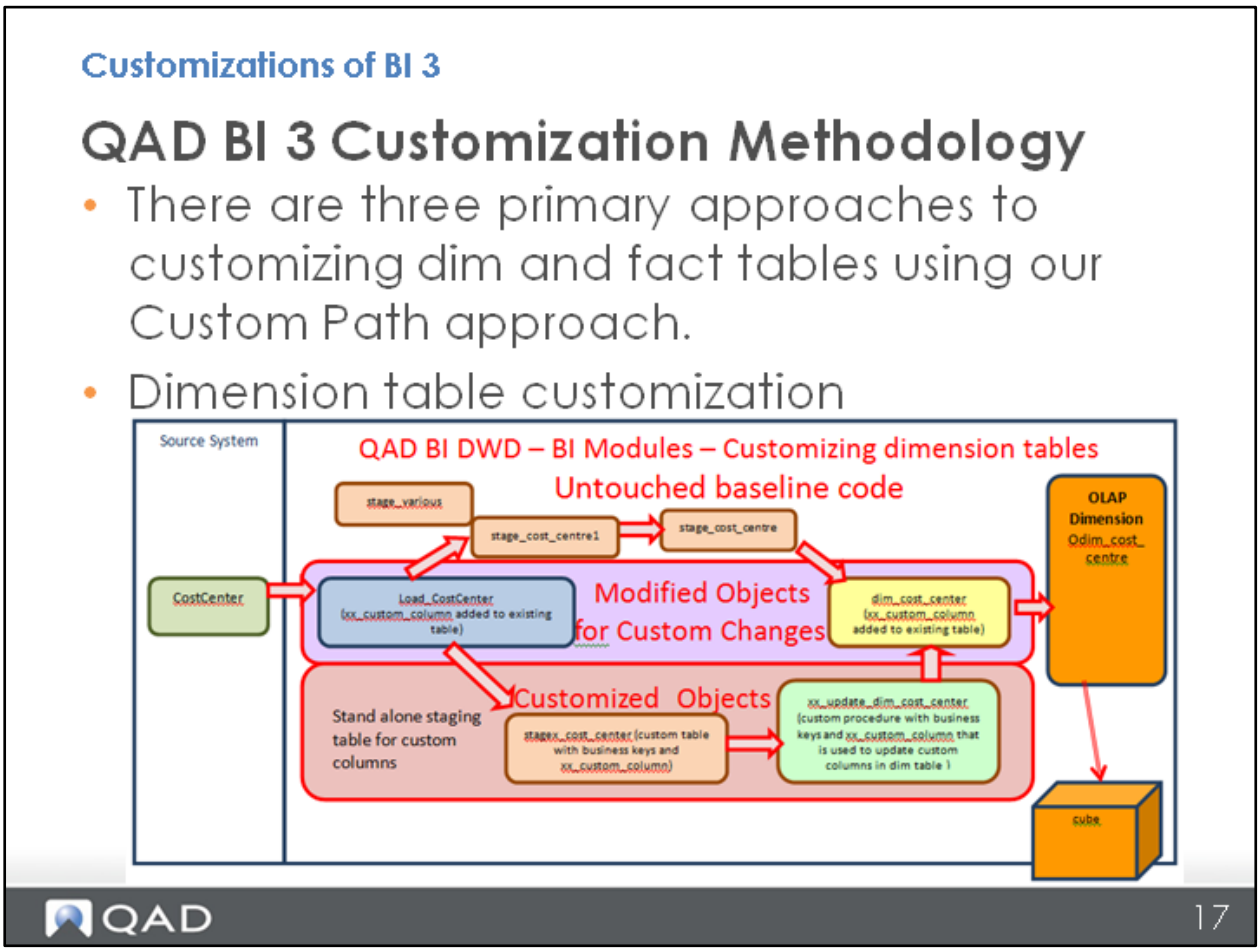
Since the new columns were added to the table and the table was not dropped, the custom columns still would contain their data, but would no longer be updated by the new procedure. A SQL Query against the table would confirm this.

Why We Chose Our Methodology (10)



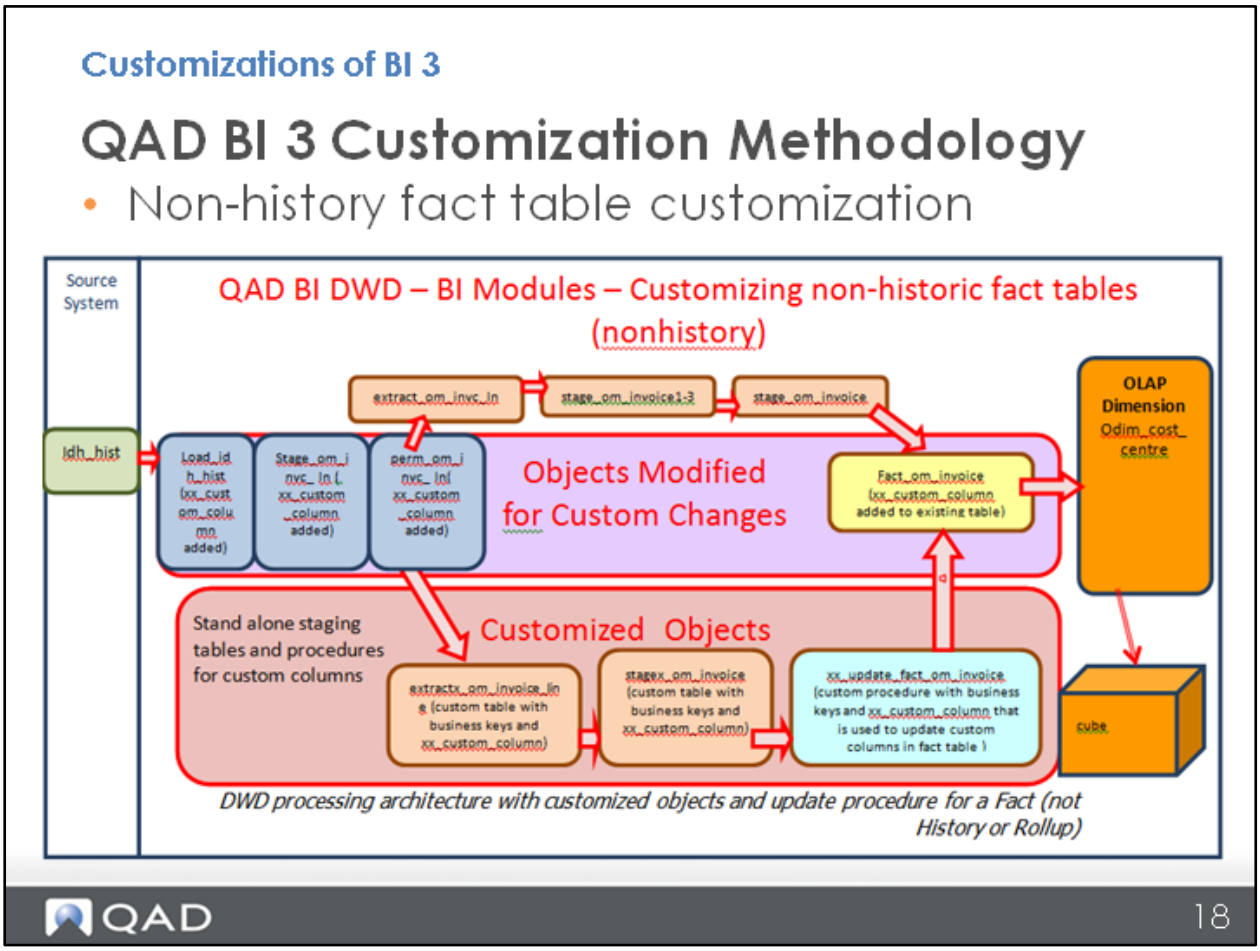
As one can see, implementing patches and upgrades to tables that have any custom work done to them is not a straight forward process and likely requires a fair amount of follow up work to get the customizations back in place in the DWD metadata (the underlying SQL table can remain unchanged). It is for this reason that **ideally** custom work will not be done directly on standard release tables in the data warehouse or at the very least, the footprint of the customized work will mostly fall outside of the baseline code.

QAD BI 3 Customization Methodology



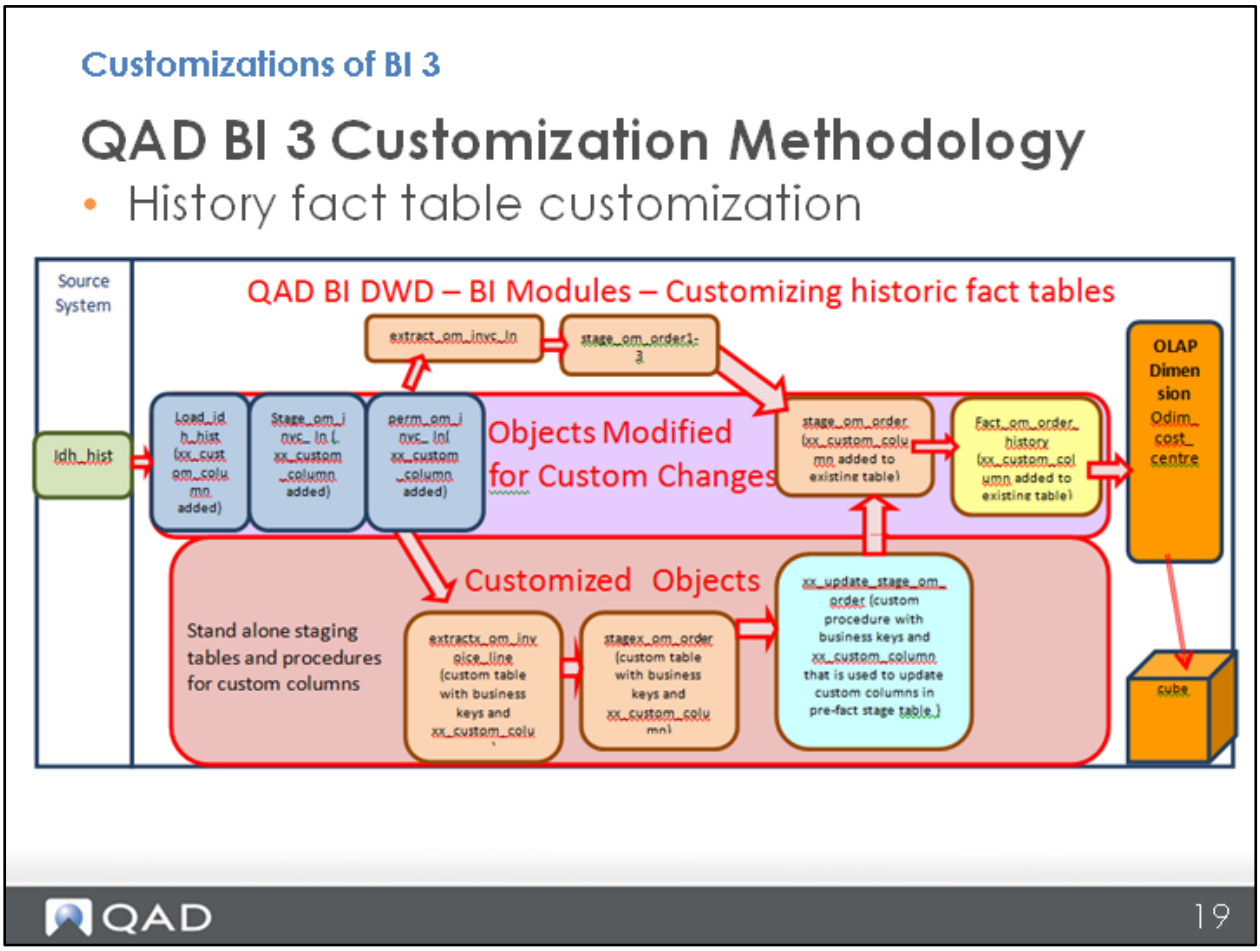
For dimension tables, the only tables that should get modified are load tables and dimension tables. Load tables can have additional columns added and so can dimension tables, but the new column in a dimension should only be updated by an outside procedure that is run after the table is loaded, not the code built by the wizard.

QAD BI 3 Customization Methodology



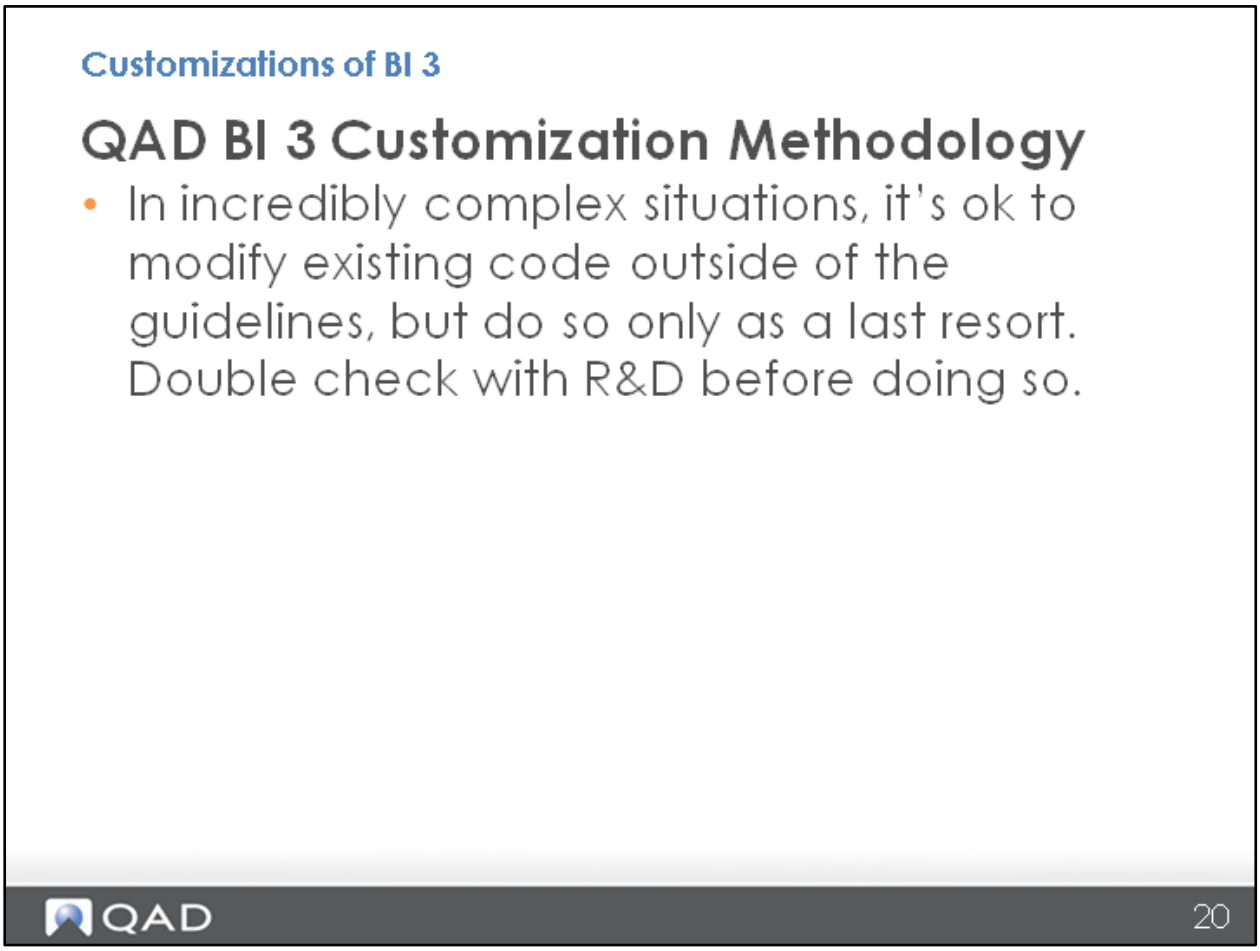
For non-history fact tables, the only tables that should get modified are load tables, the stage table prior to a perm table, perm tables and the fact tables. Any new columns in the fact table should only be updated by an outside procedure that is run after the table is loaded, not the code built by the wizard.

QAD BI 3 Customization Methodology

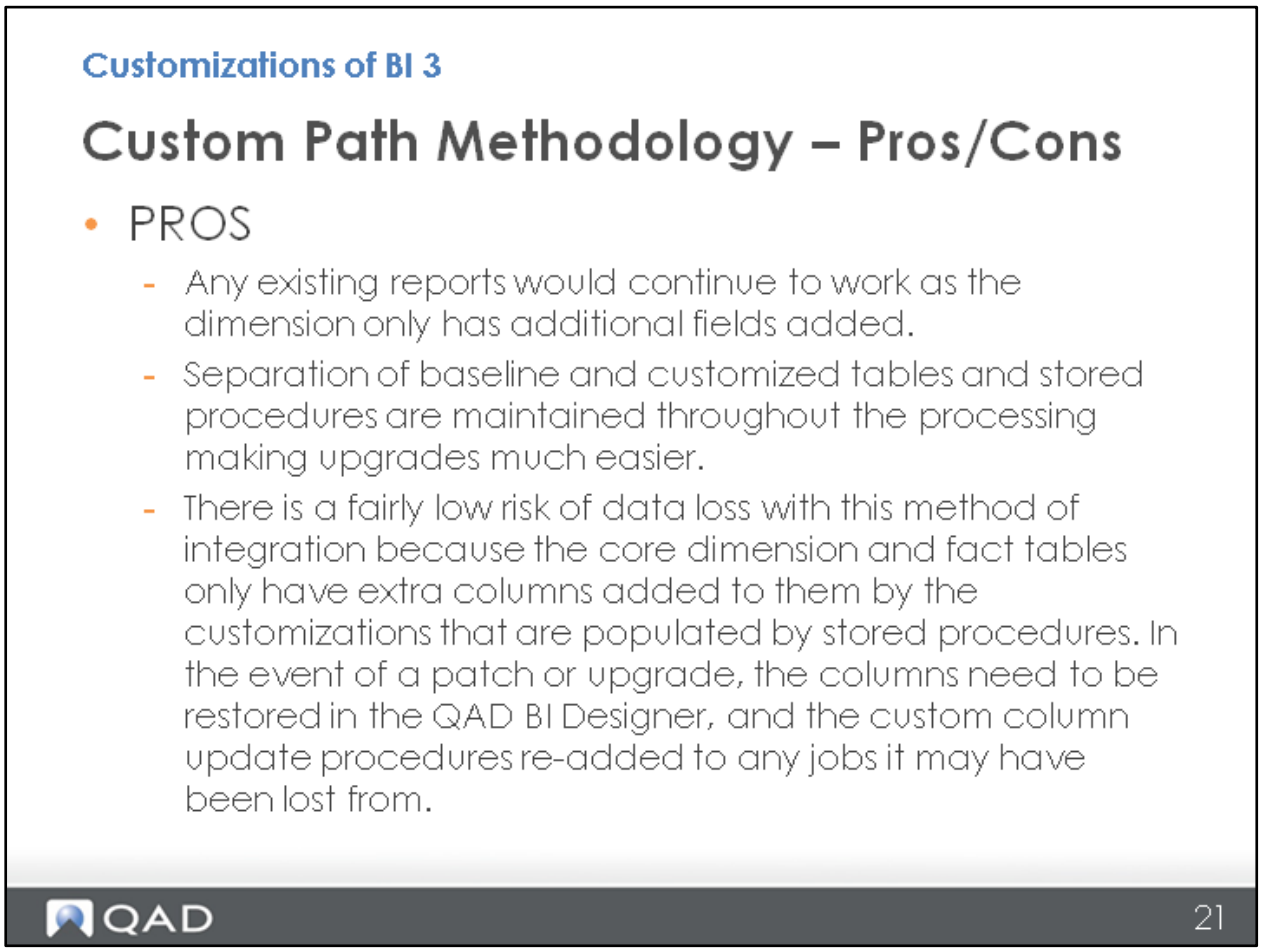


For history fact tables, the only tables that should get modified are load tables, the stage table prior to a perm table, perm tables, the last stage table prior to the fact and the fact table. Any new columns in the fact table should only be updated by the prior staging table. The new column in the prior staging table should be updated by an outside procedure that is run after the table is loaded, not the code built by the wizard. The reason we allow this is that the logic involved in getting data right in the fact_om_order_history table is incredibly complex and by inserting it into the staging table prior, we can insure that the logic for the fact table does its thing as it should and people building customizations don't have to attempt to emulate all that logic with an update procedure for a customization.

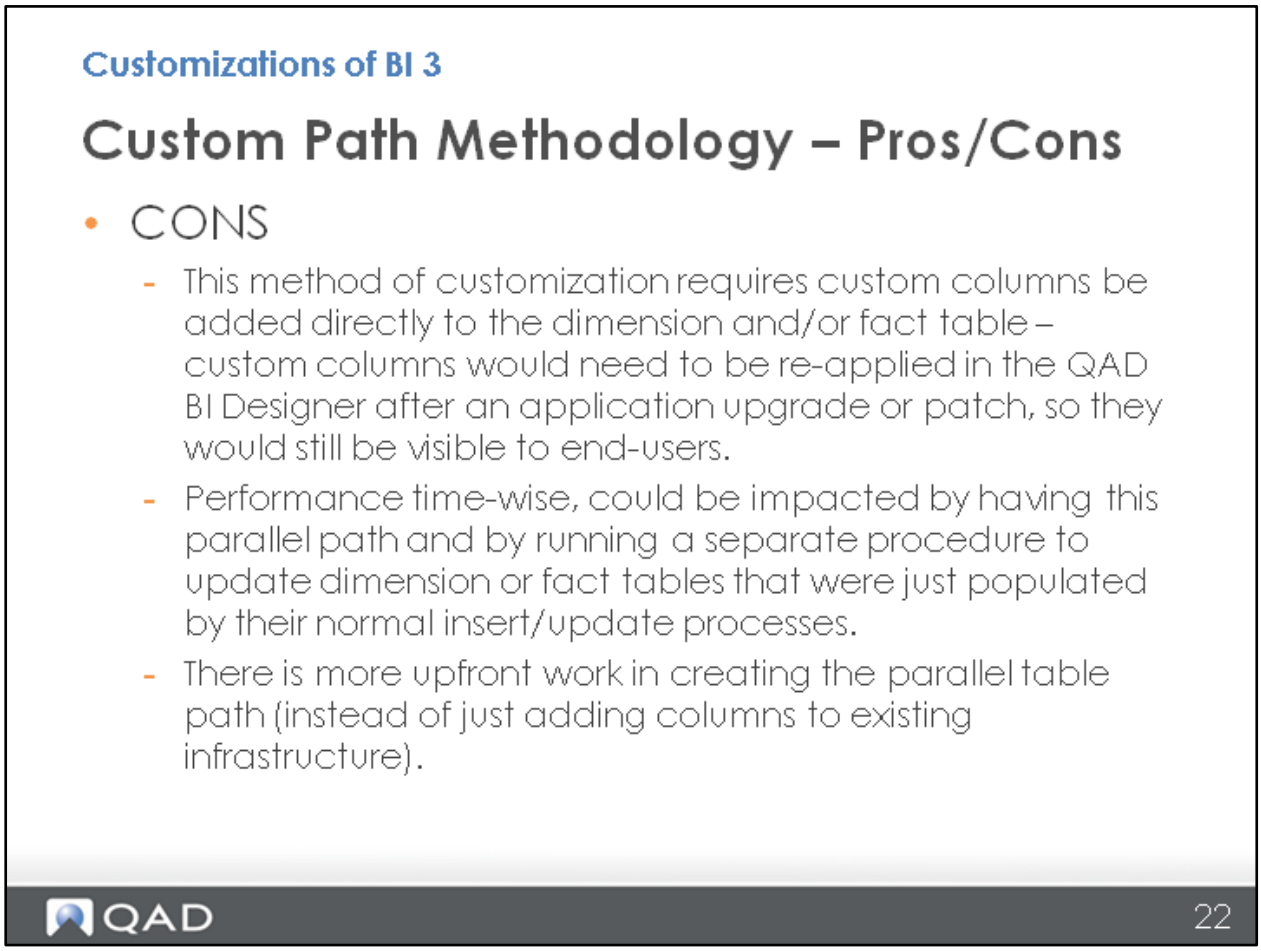
QAD BI 3 Customization Methodology



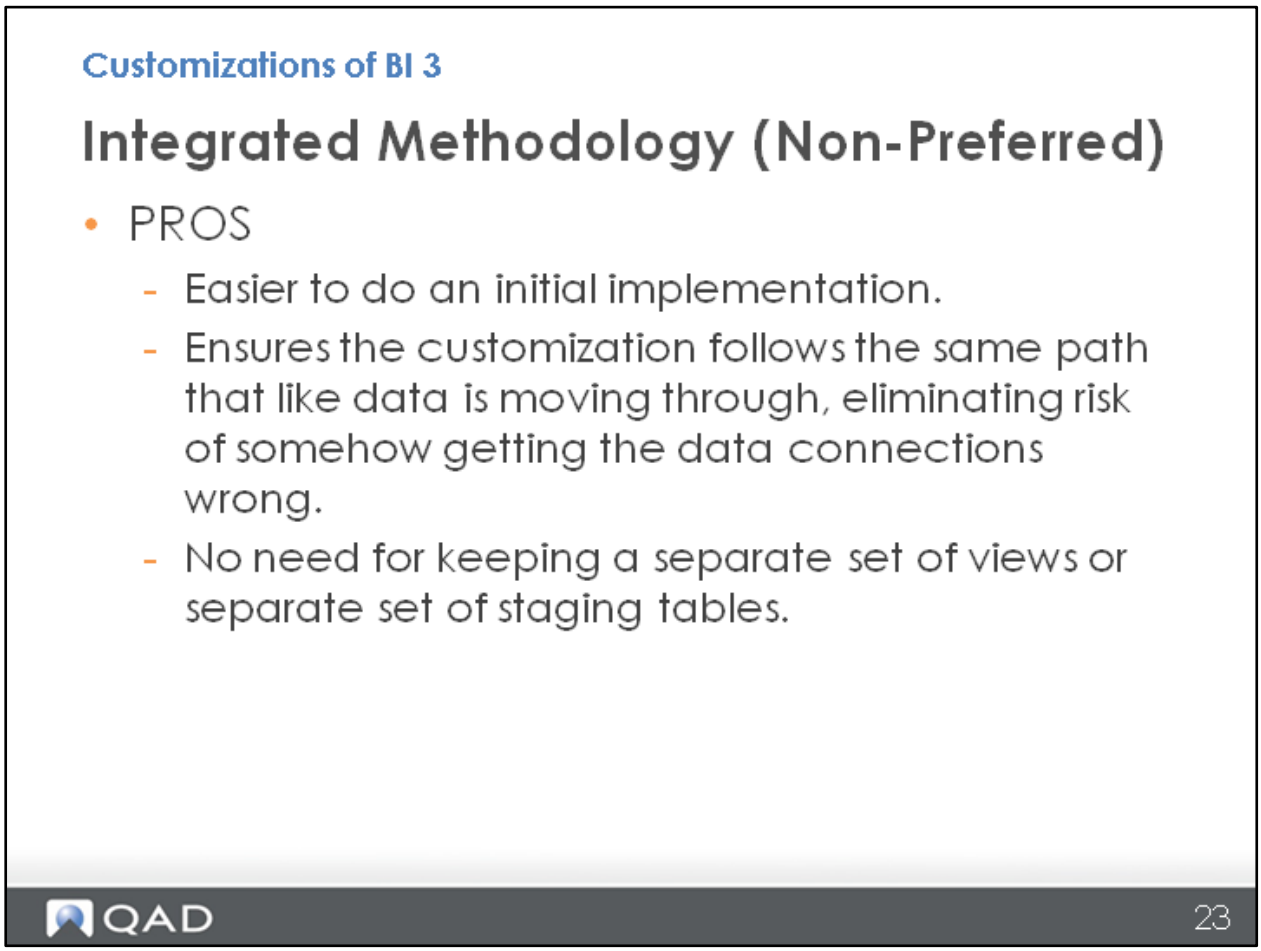
Custom Path Methodology – Pros/Cons



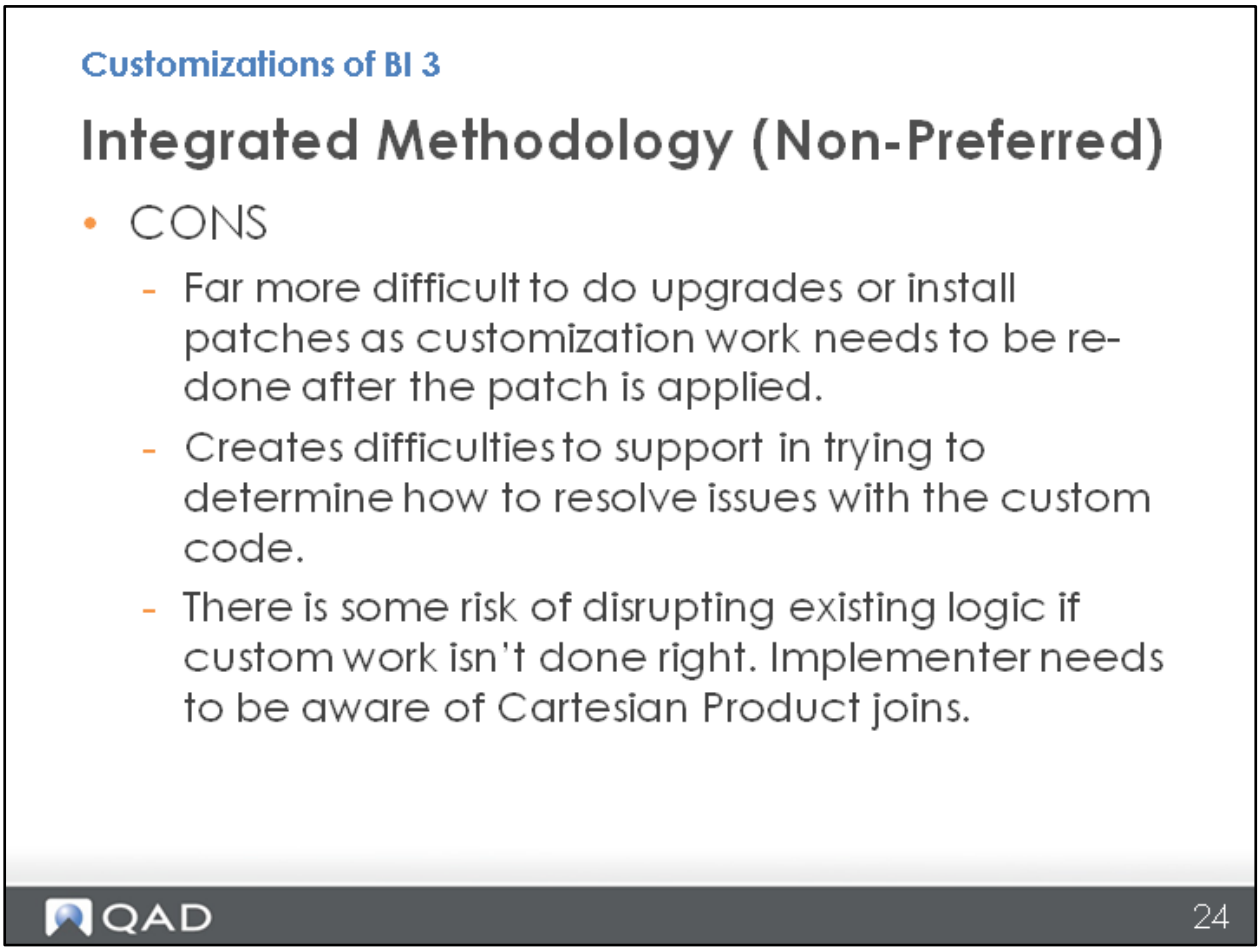
Custom Path Methodology – Pros/Cons



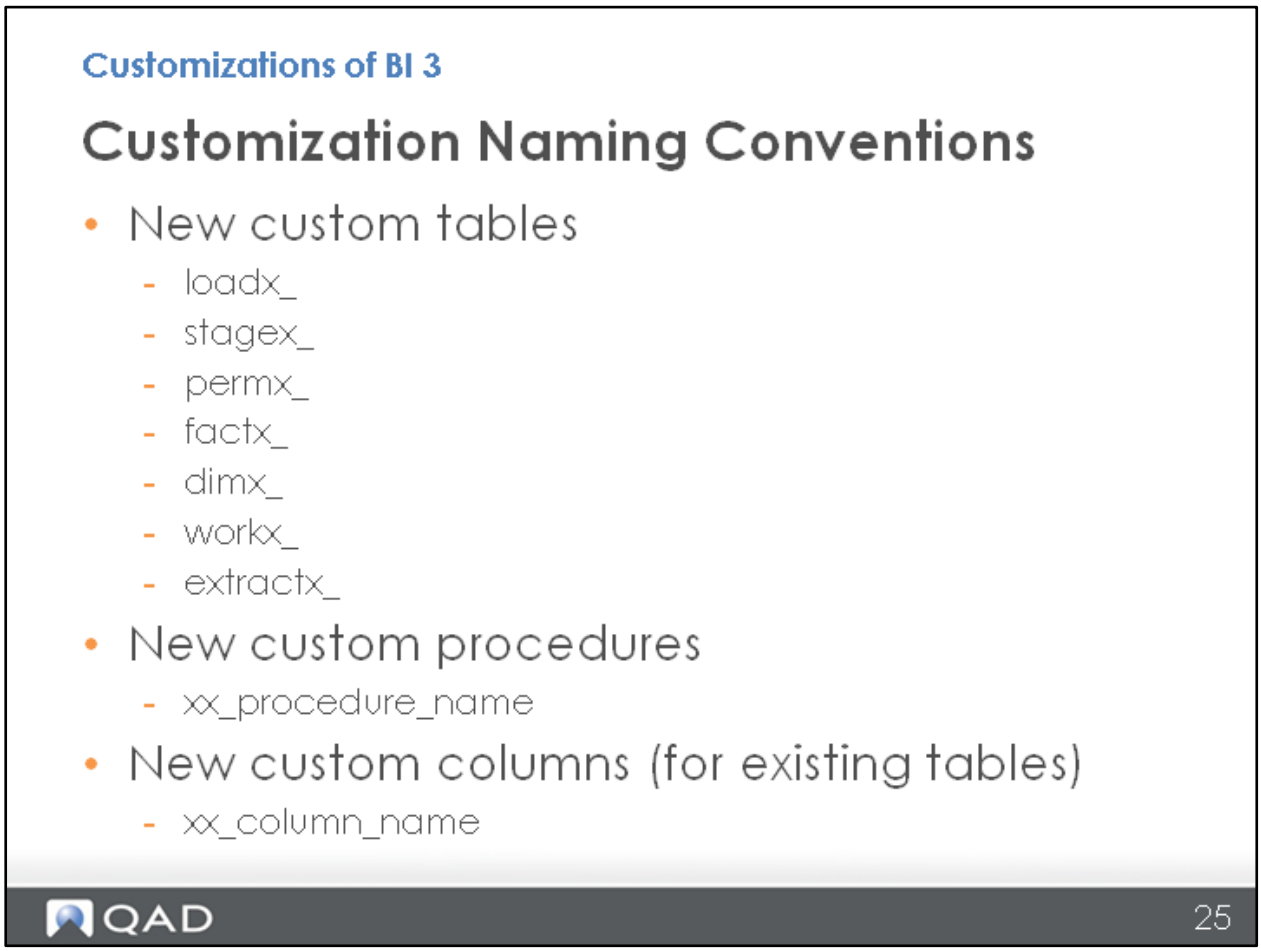
Integrated Methodology (Non-Preferred)



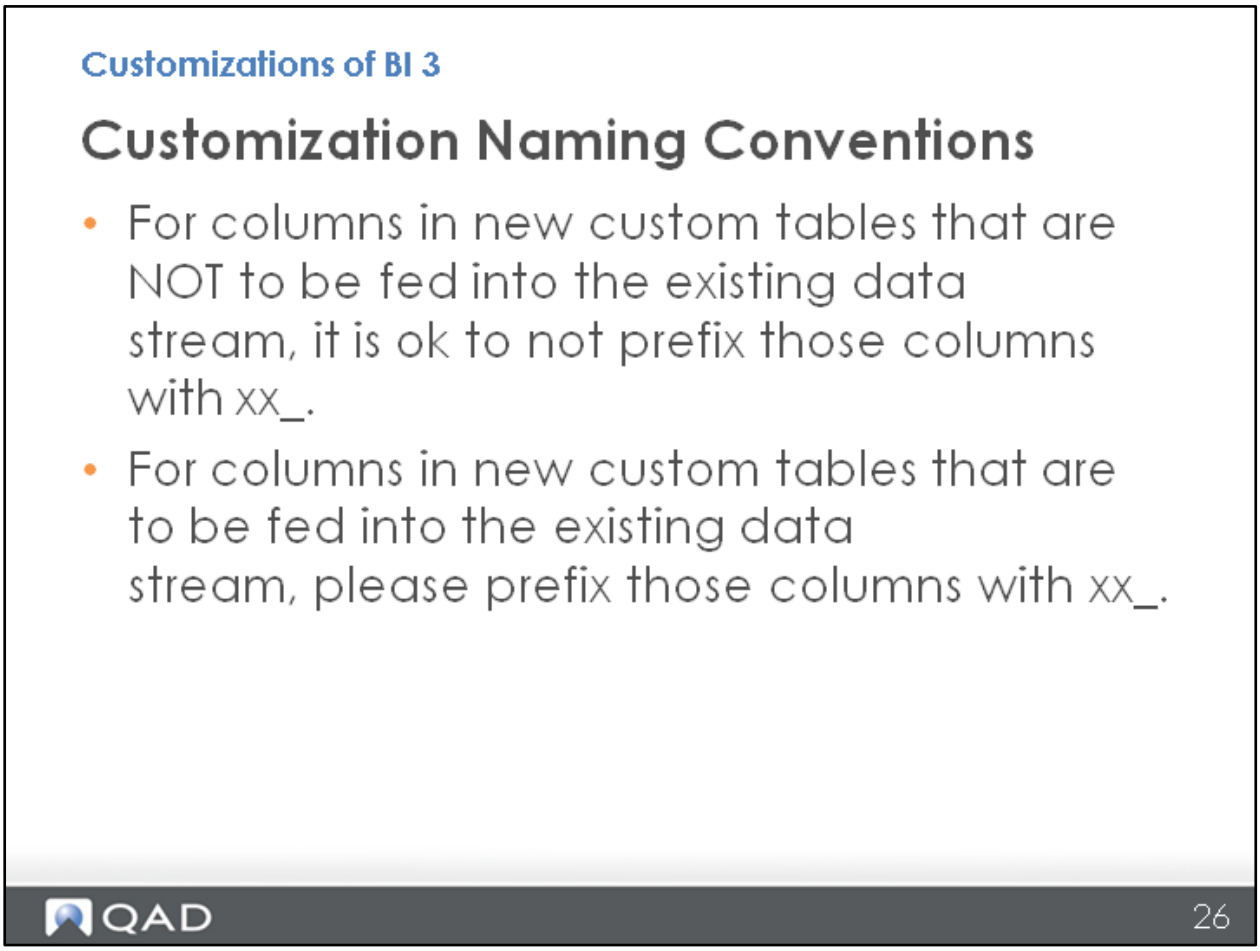
Integrated Methodology (Non-Preferred)



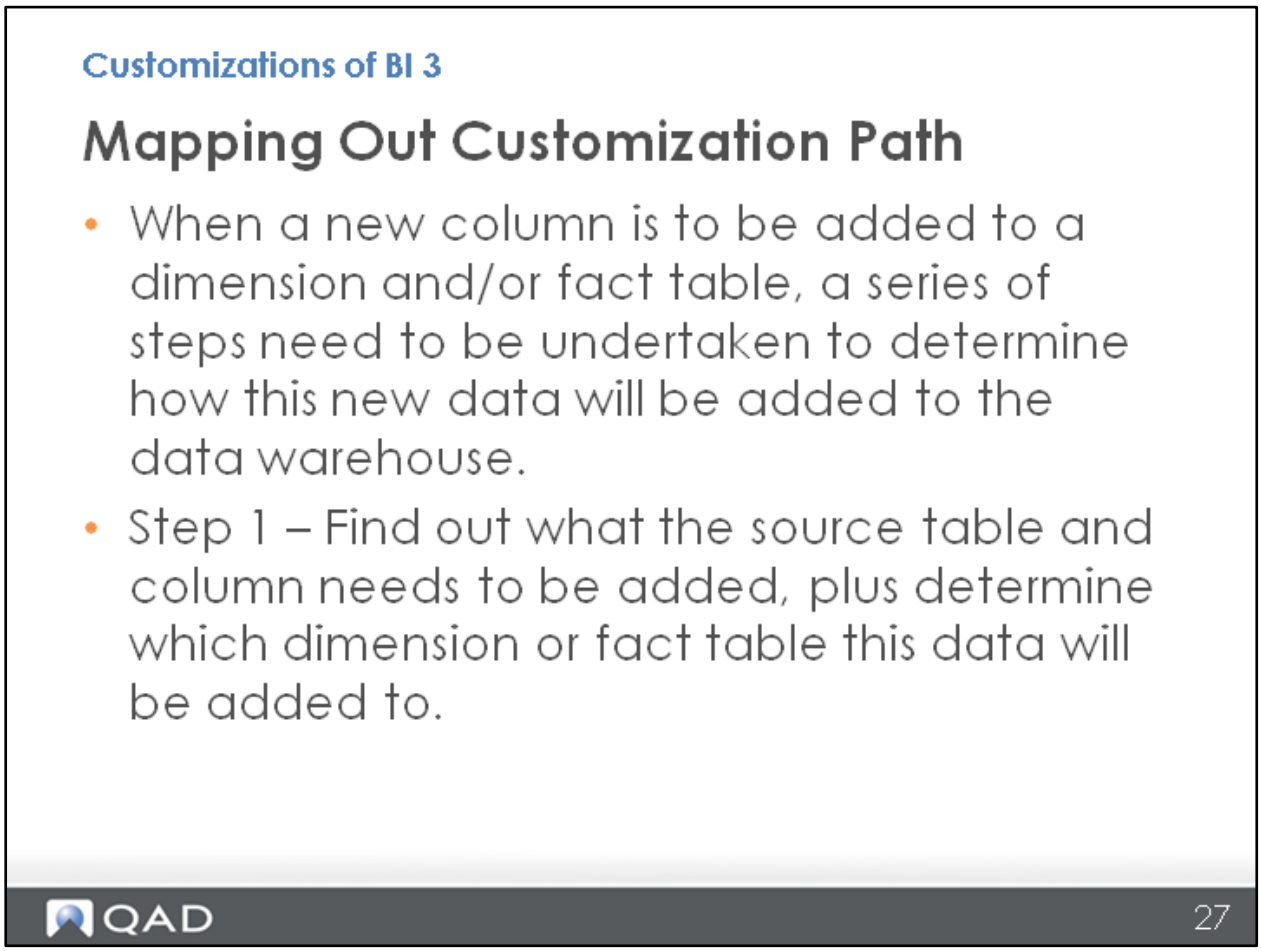
Customization Naming Conventions



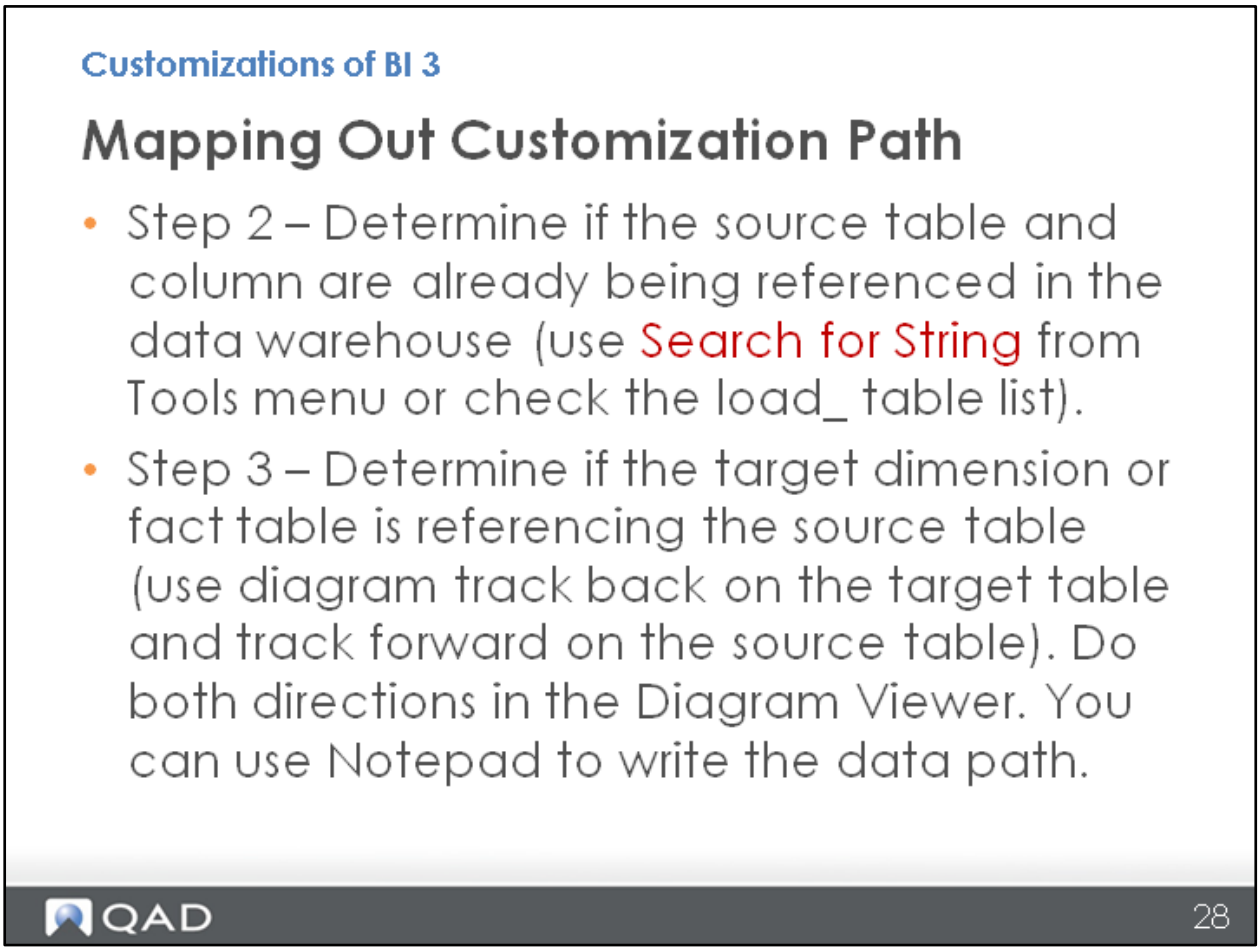
Customization Naming Conventions



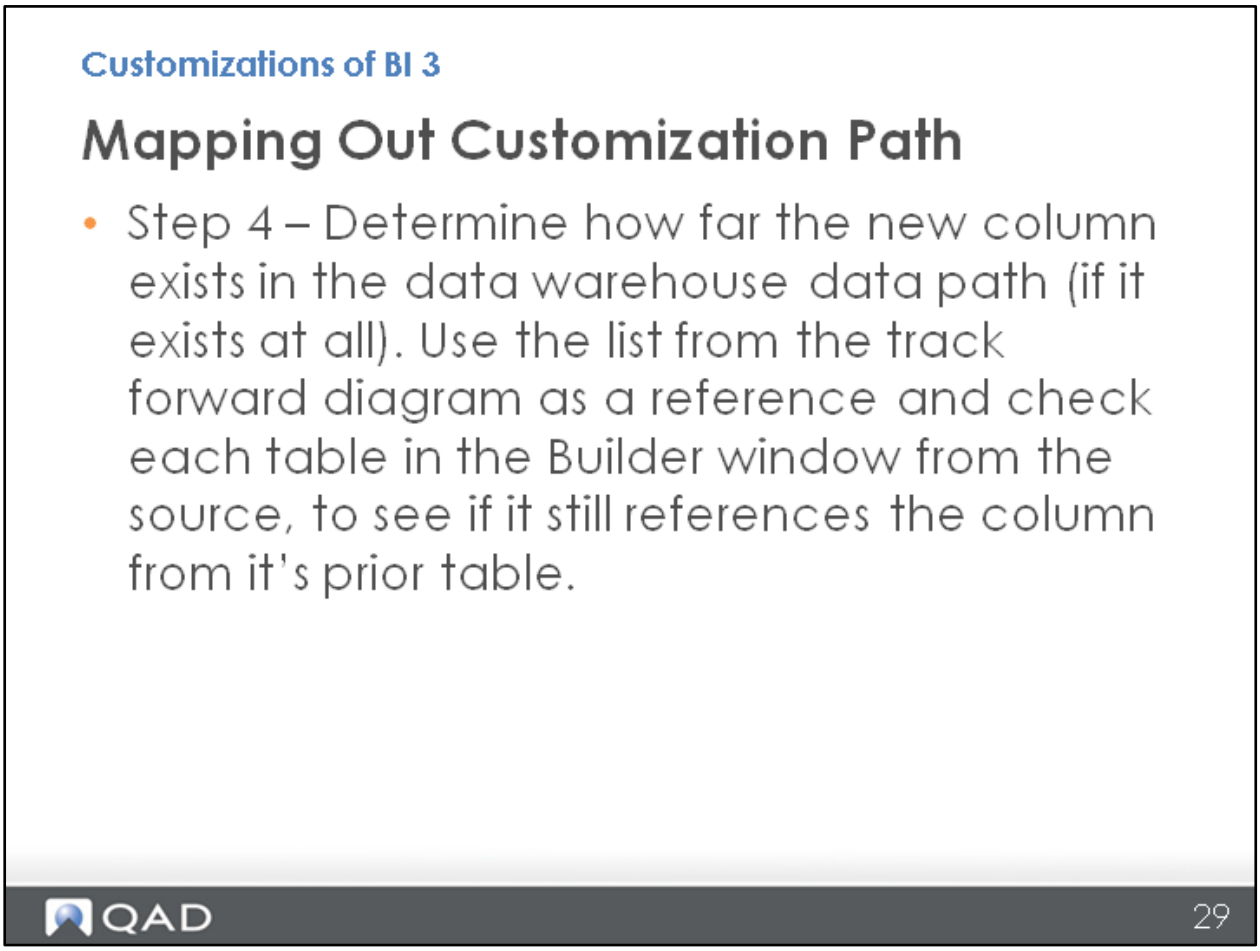
Mapping Out Customization Path



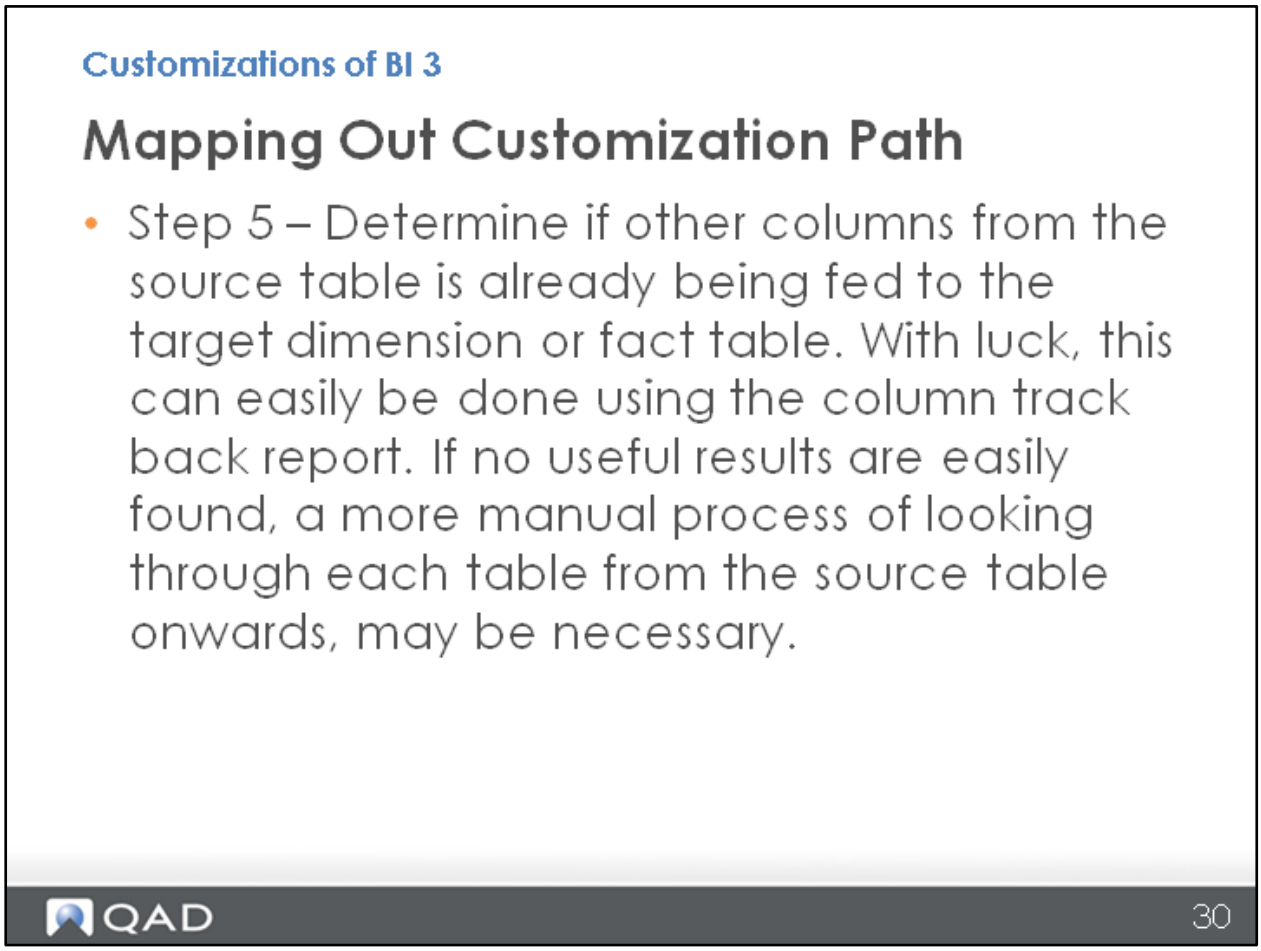
Mapping Out Customization Path



Mapping Out Customization Path

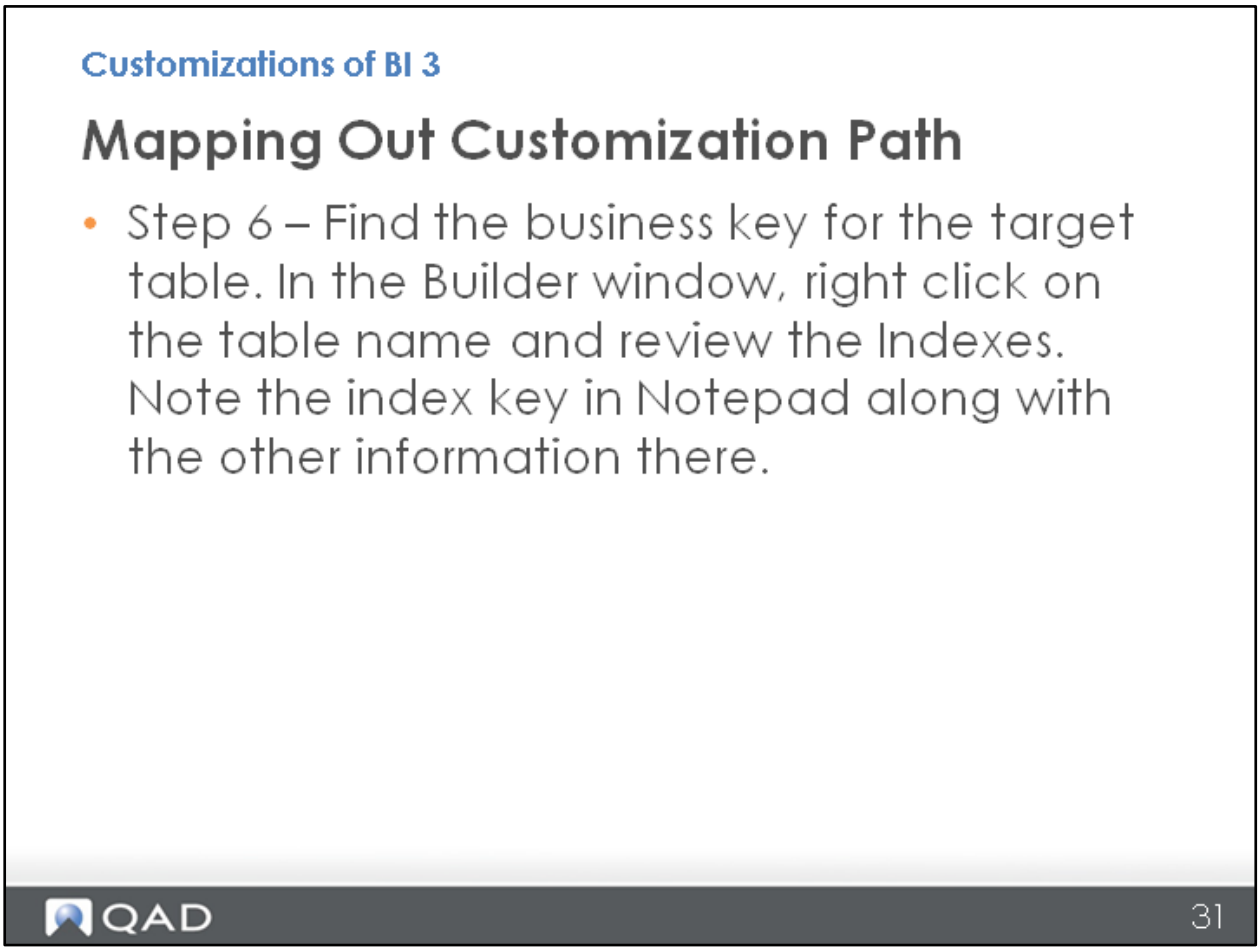


Mapping Out Customization Path

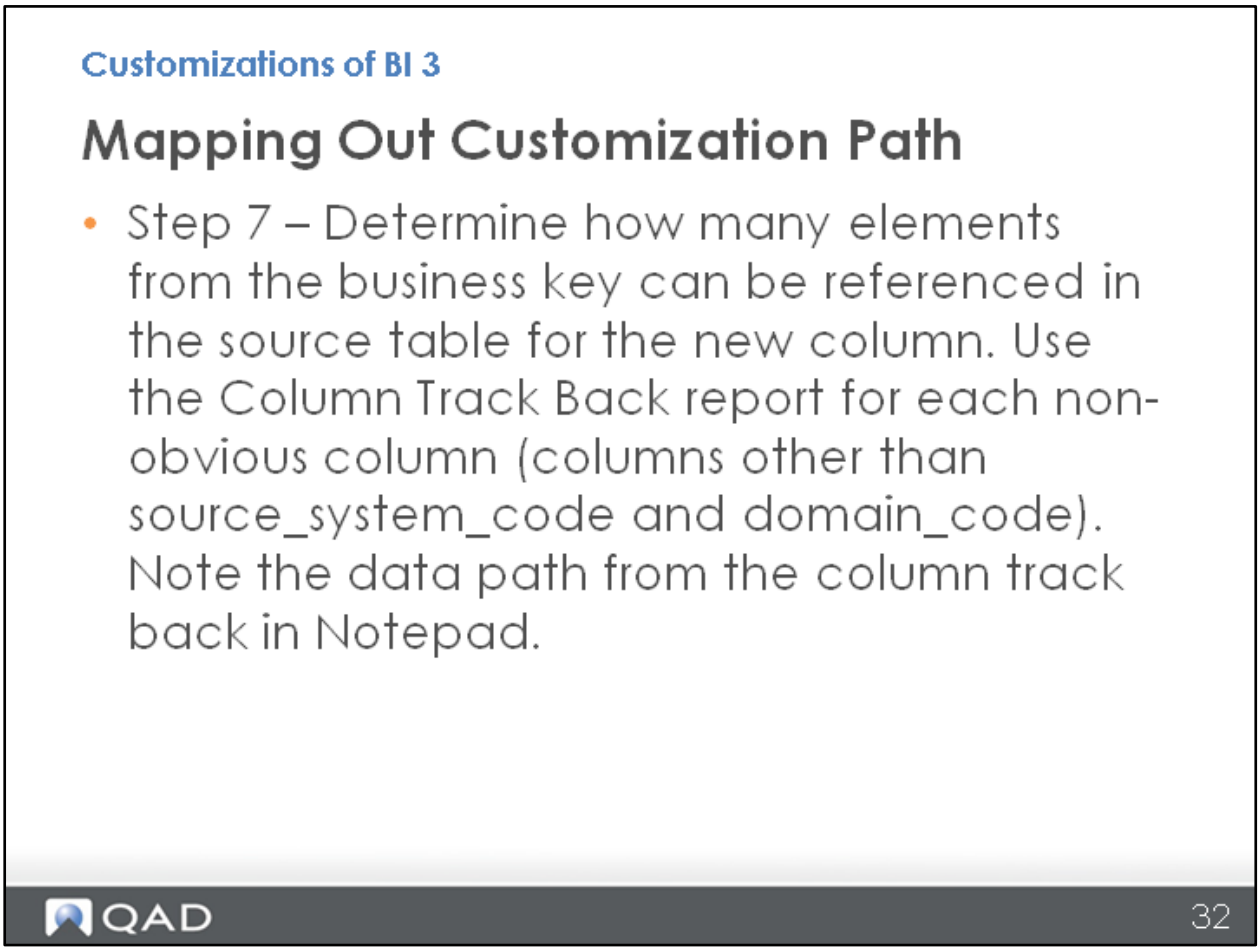


To determine if the column is fed further into the data stream and maybe renamed, we find the next table in the track back/track forward diagram and then look at that table in the Builder/Browser pane, and look for the source column list and confirm whether it exists or not. You can click on the source column header to sort to make finding it easier.

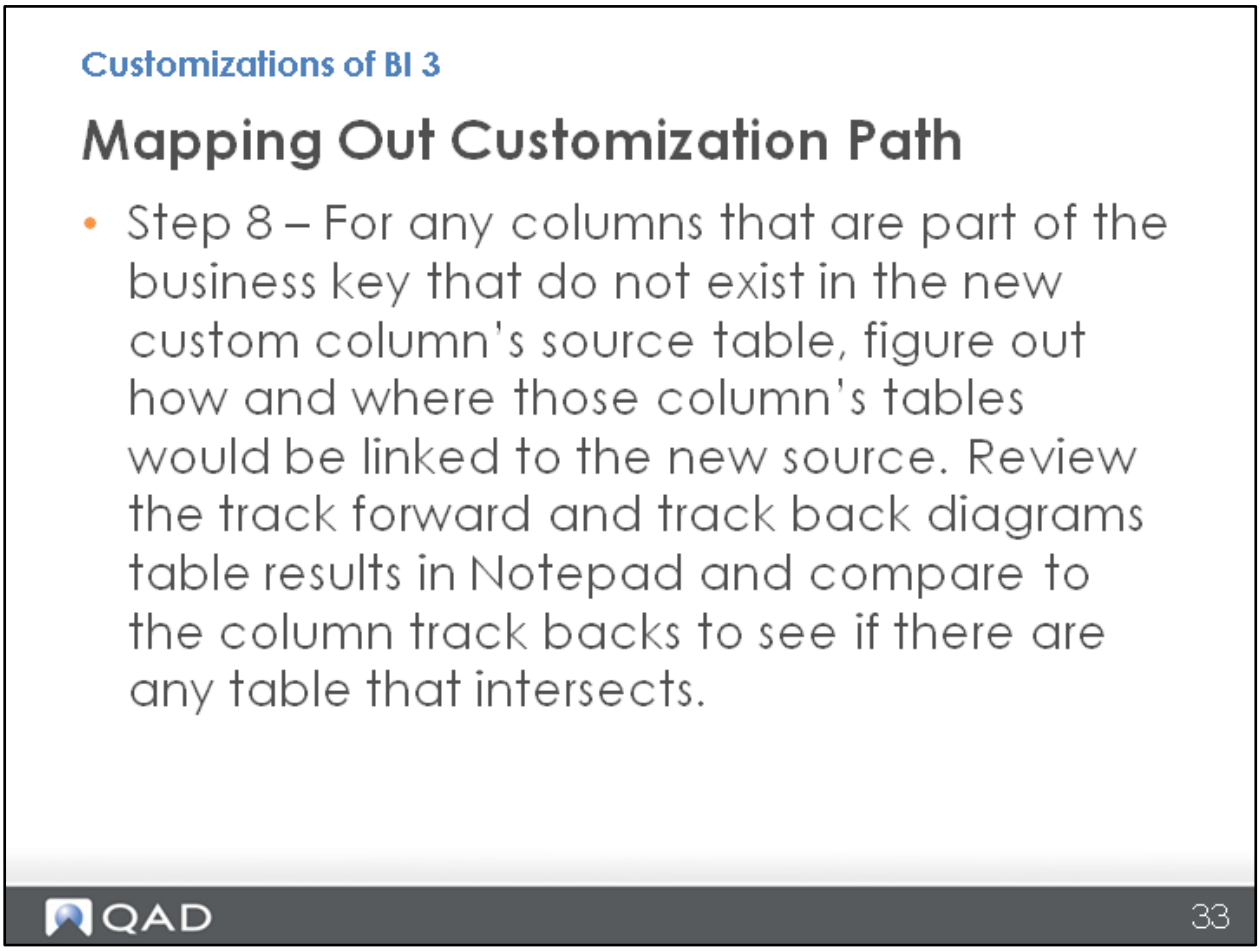
Mapping Out Customization Path



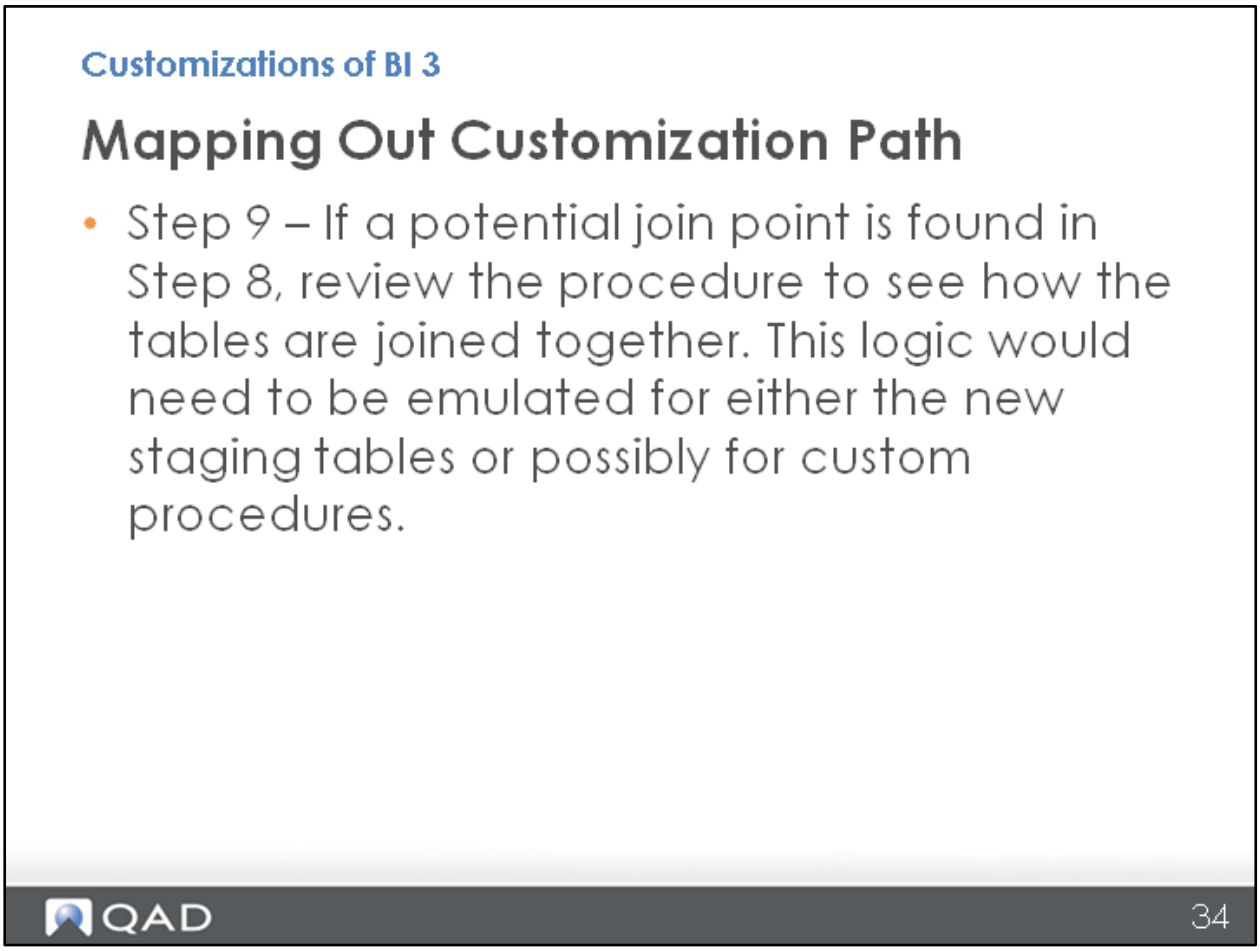
Mapping Out Customization Path



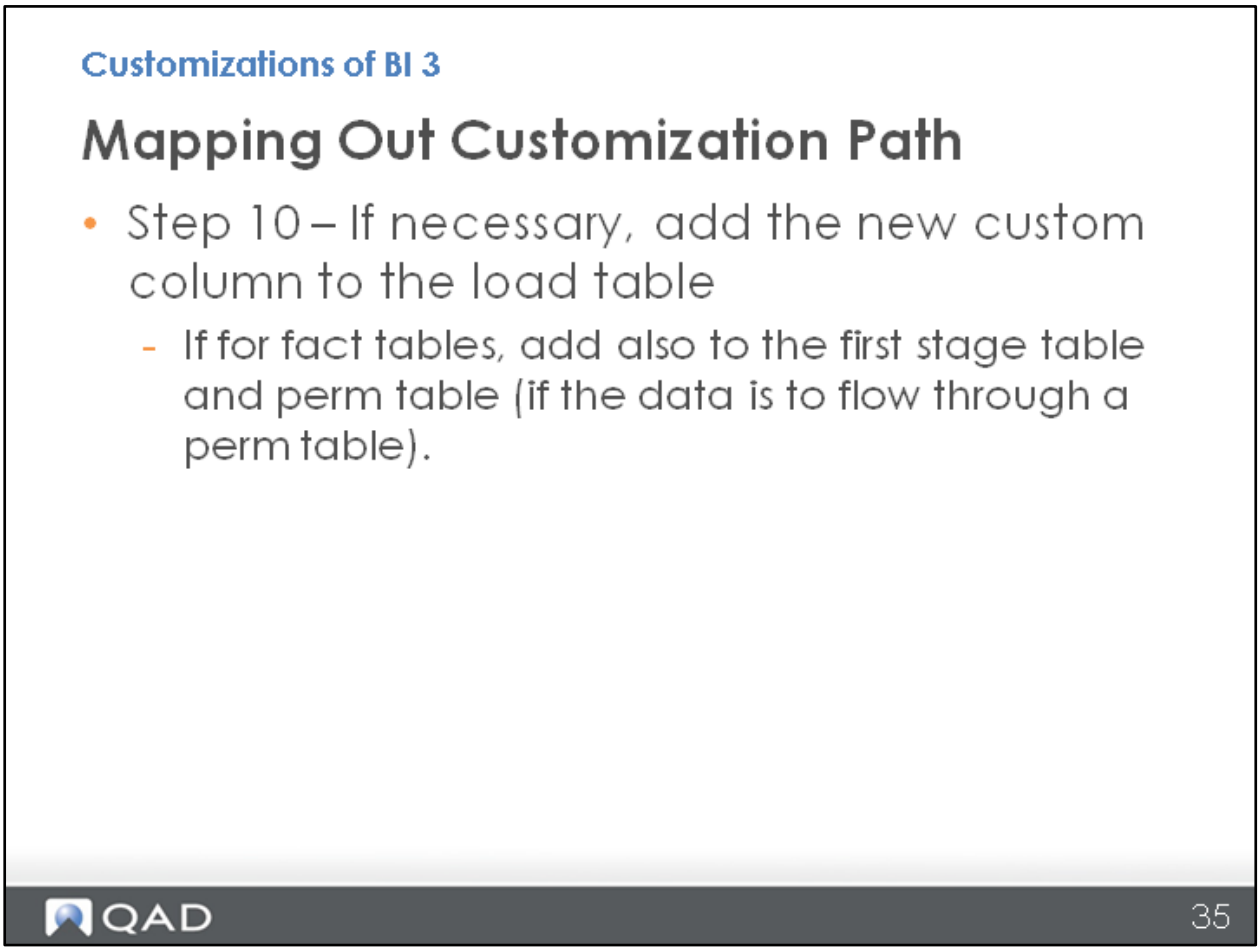
Mapping Out Customization Path



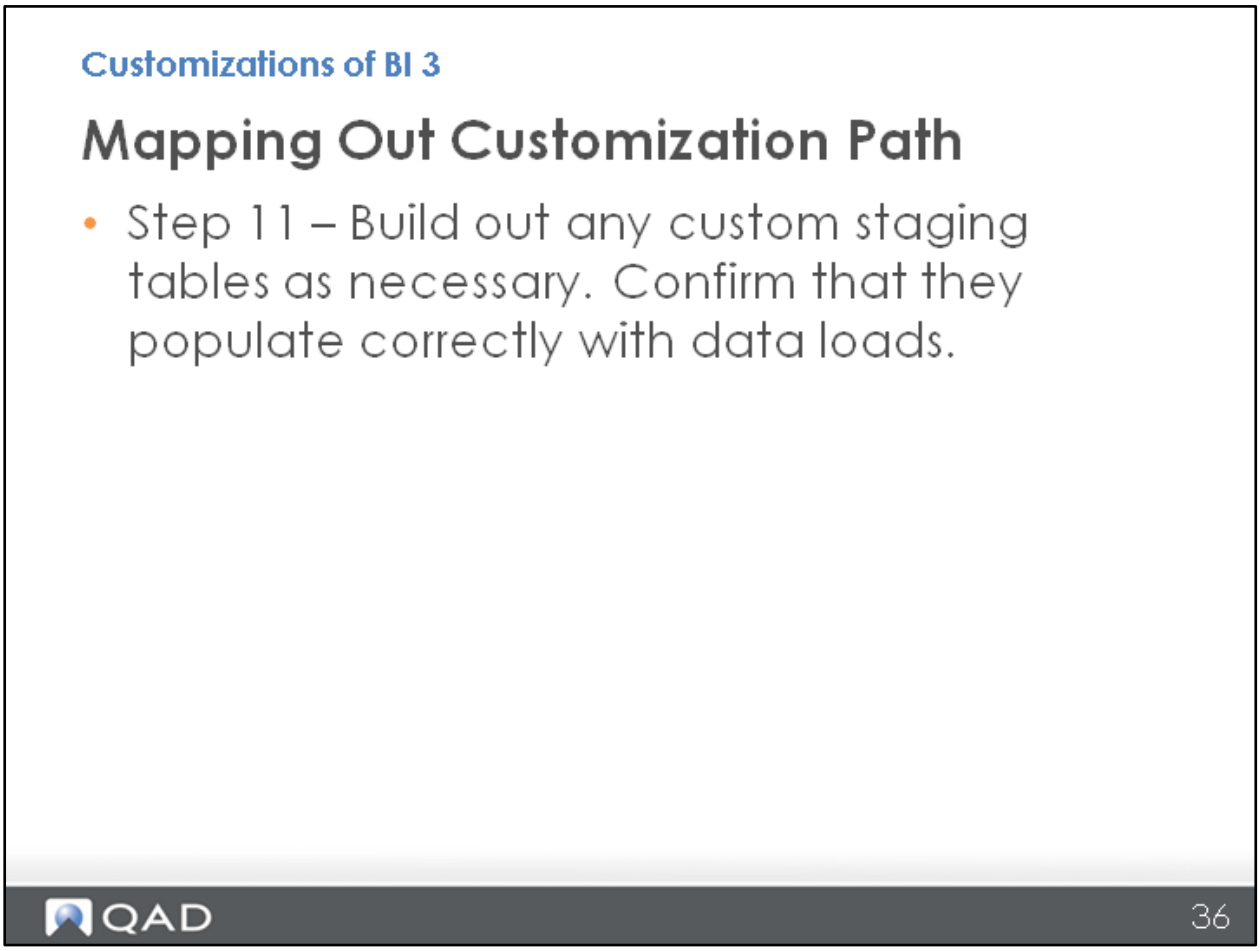
Mapping Out Customization Path



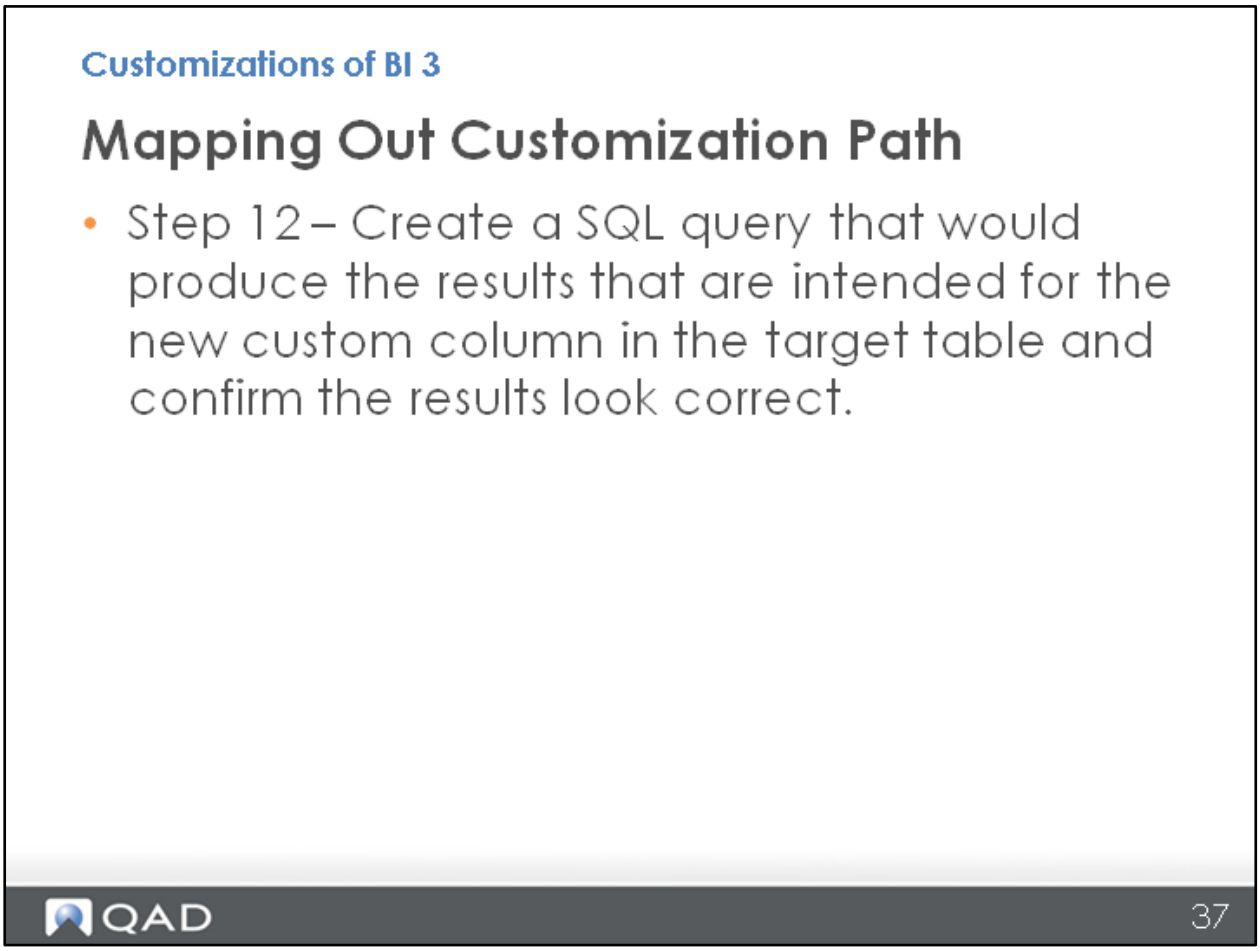
Mapping Out Customization Path



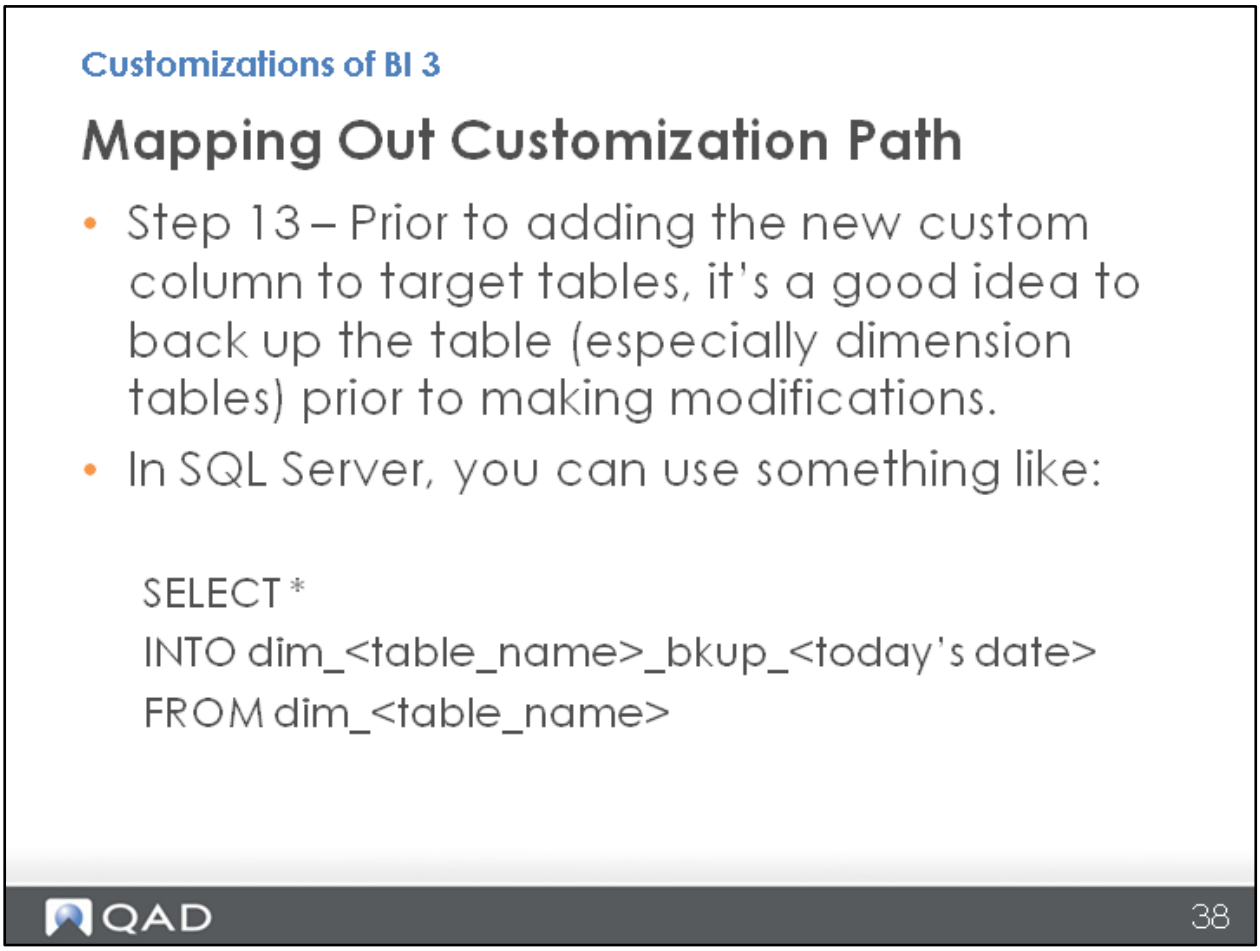
Mapping Out Customization Path



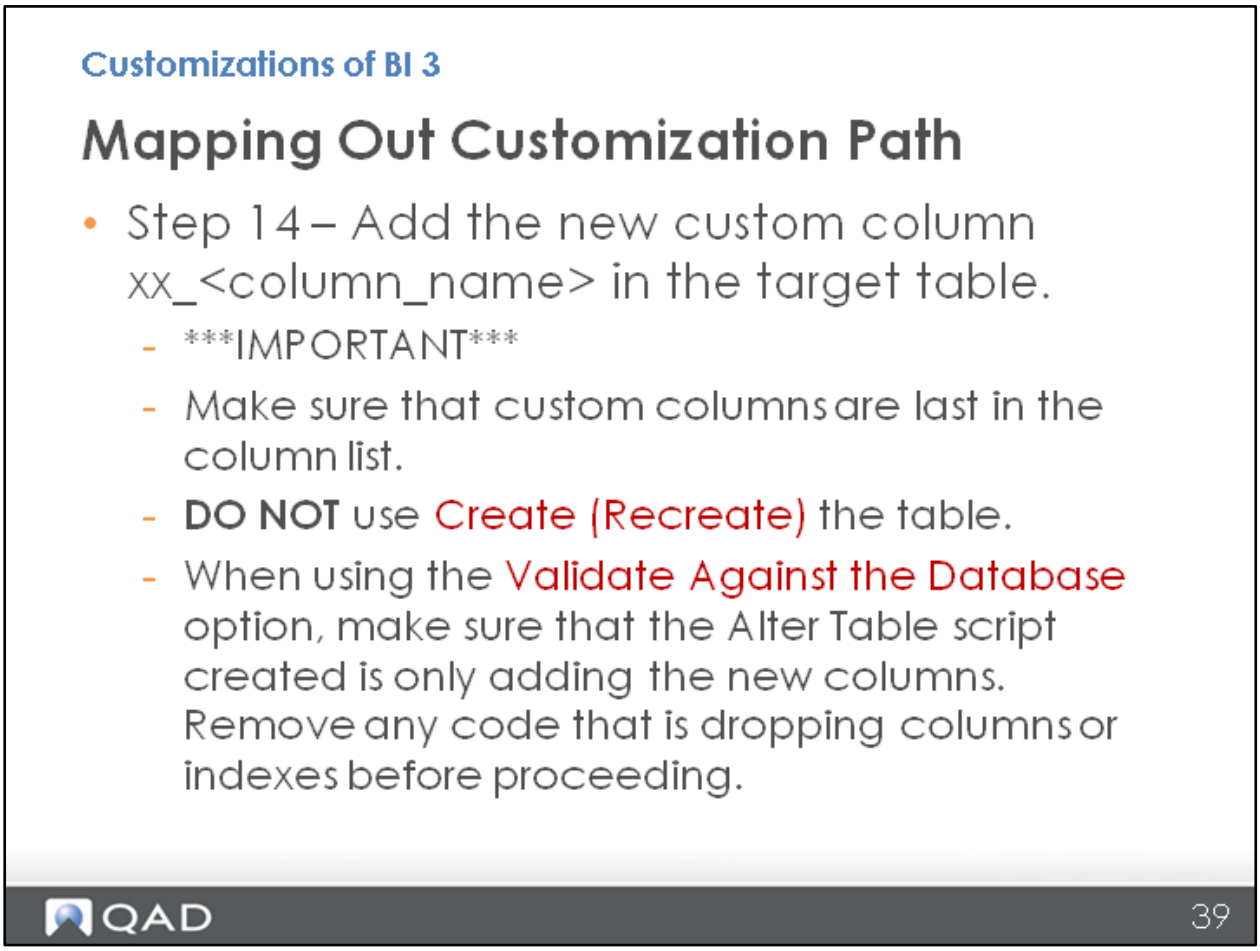
Mapping Out Customization Path



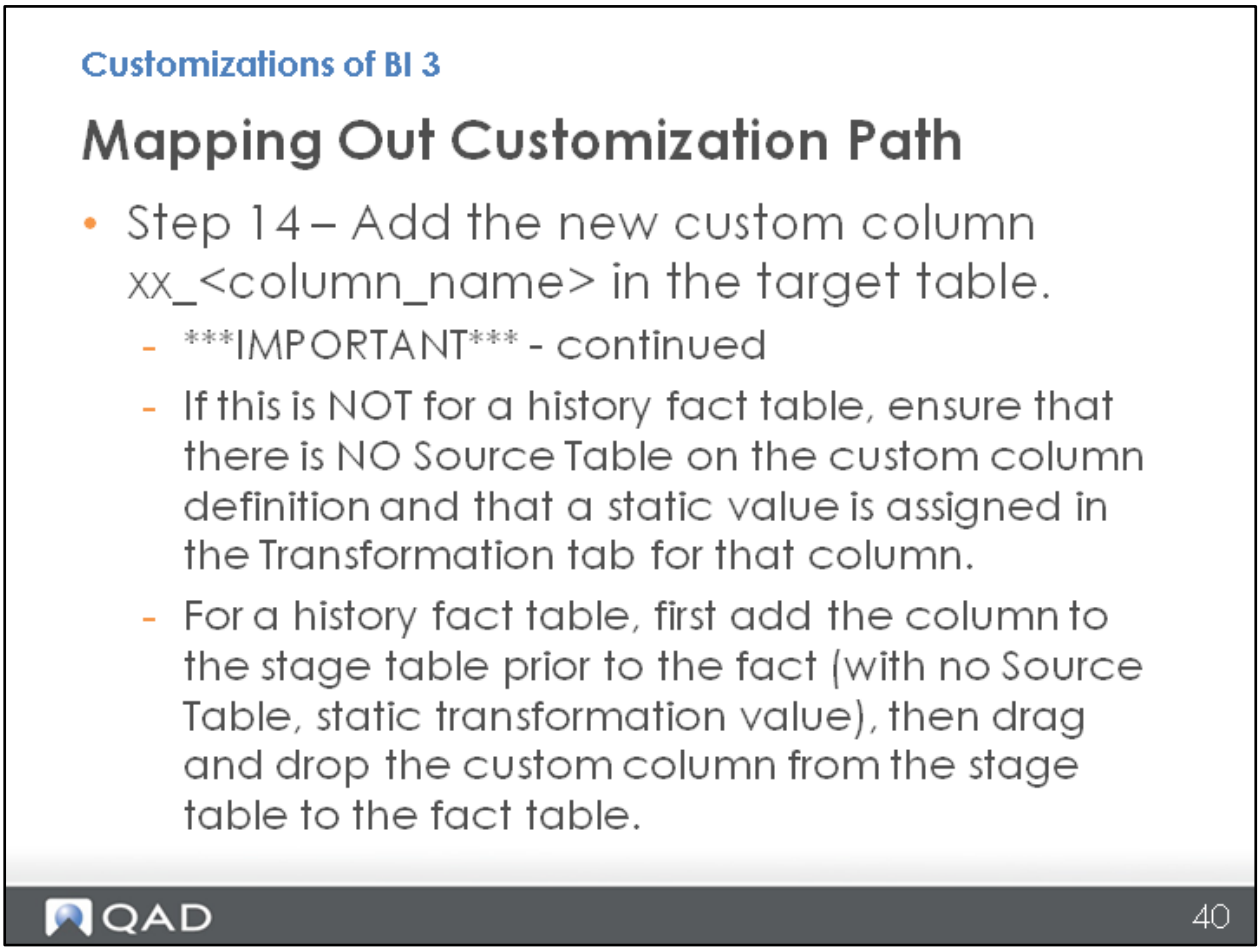
Mapping Out Customization Path



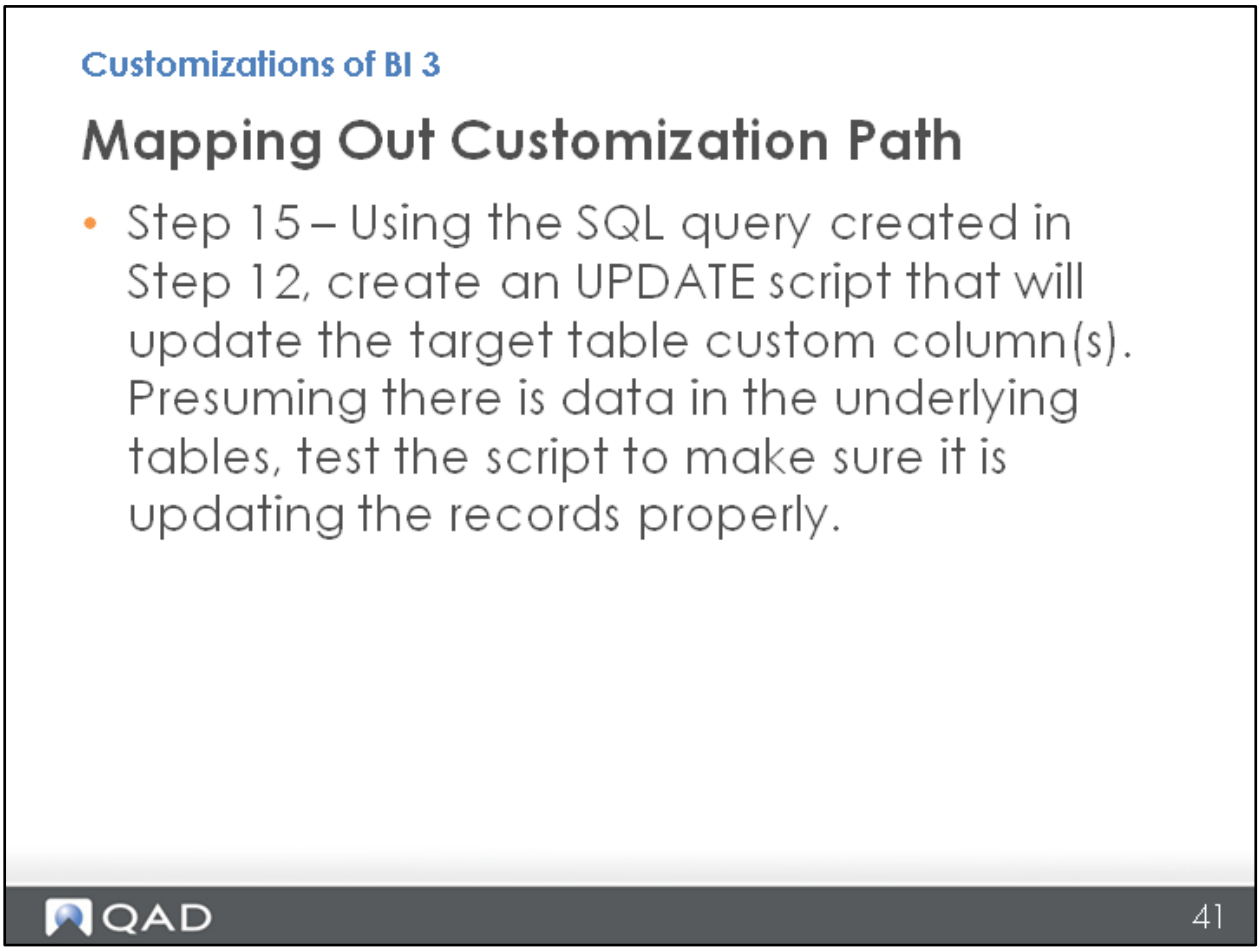
Mapping Out Customization Path



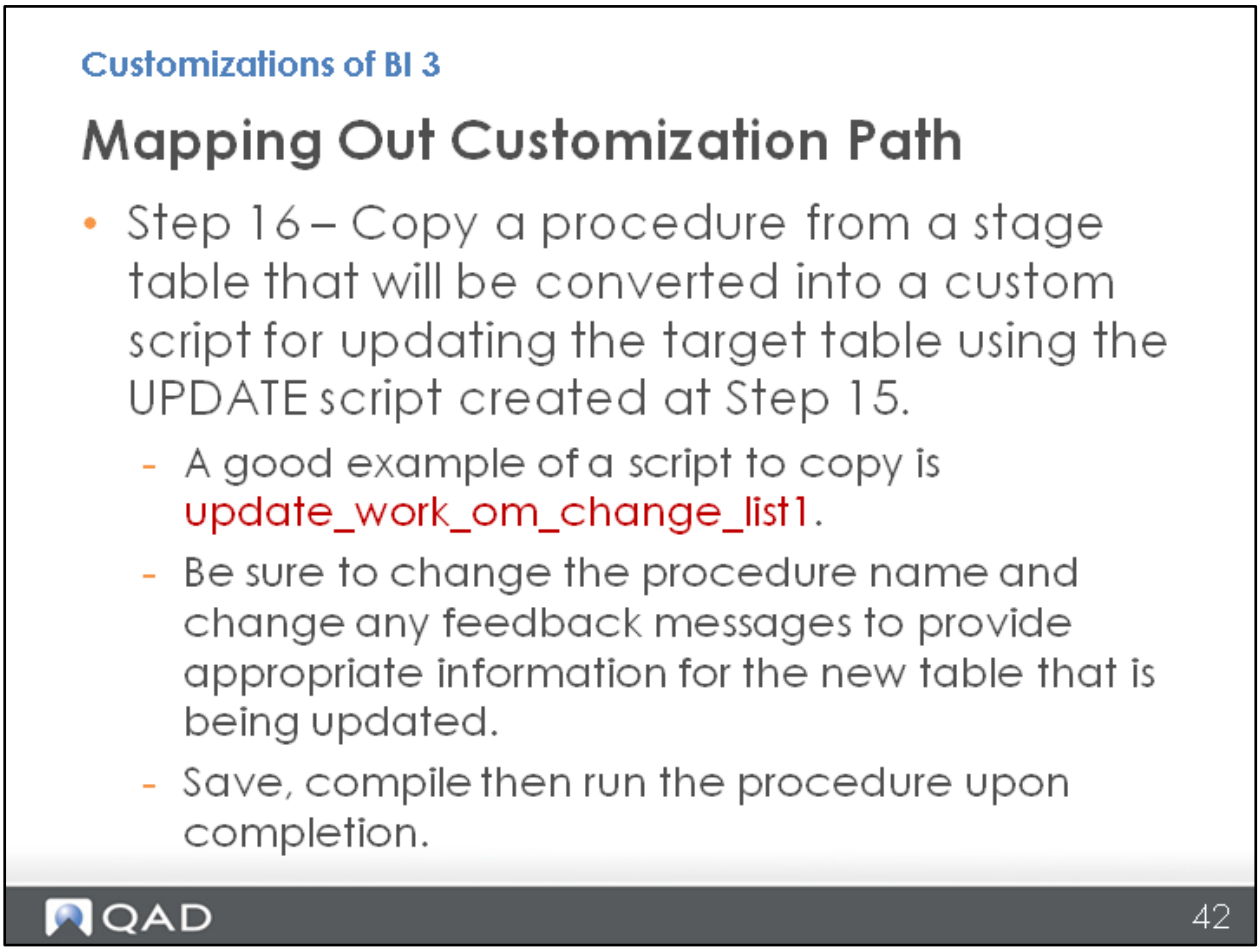
Mapping Out Customization Path



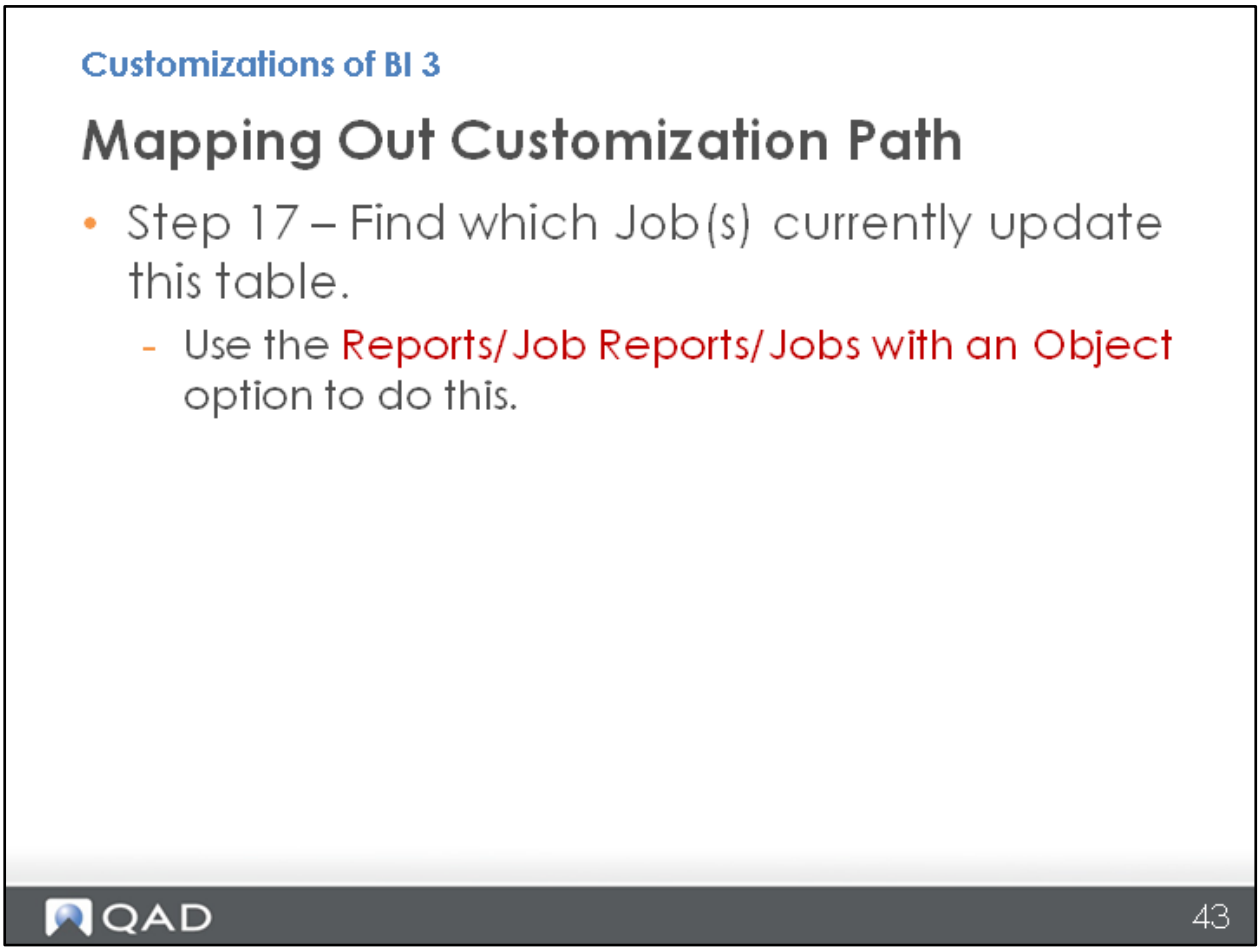
Mapping Out Customization Path



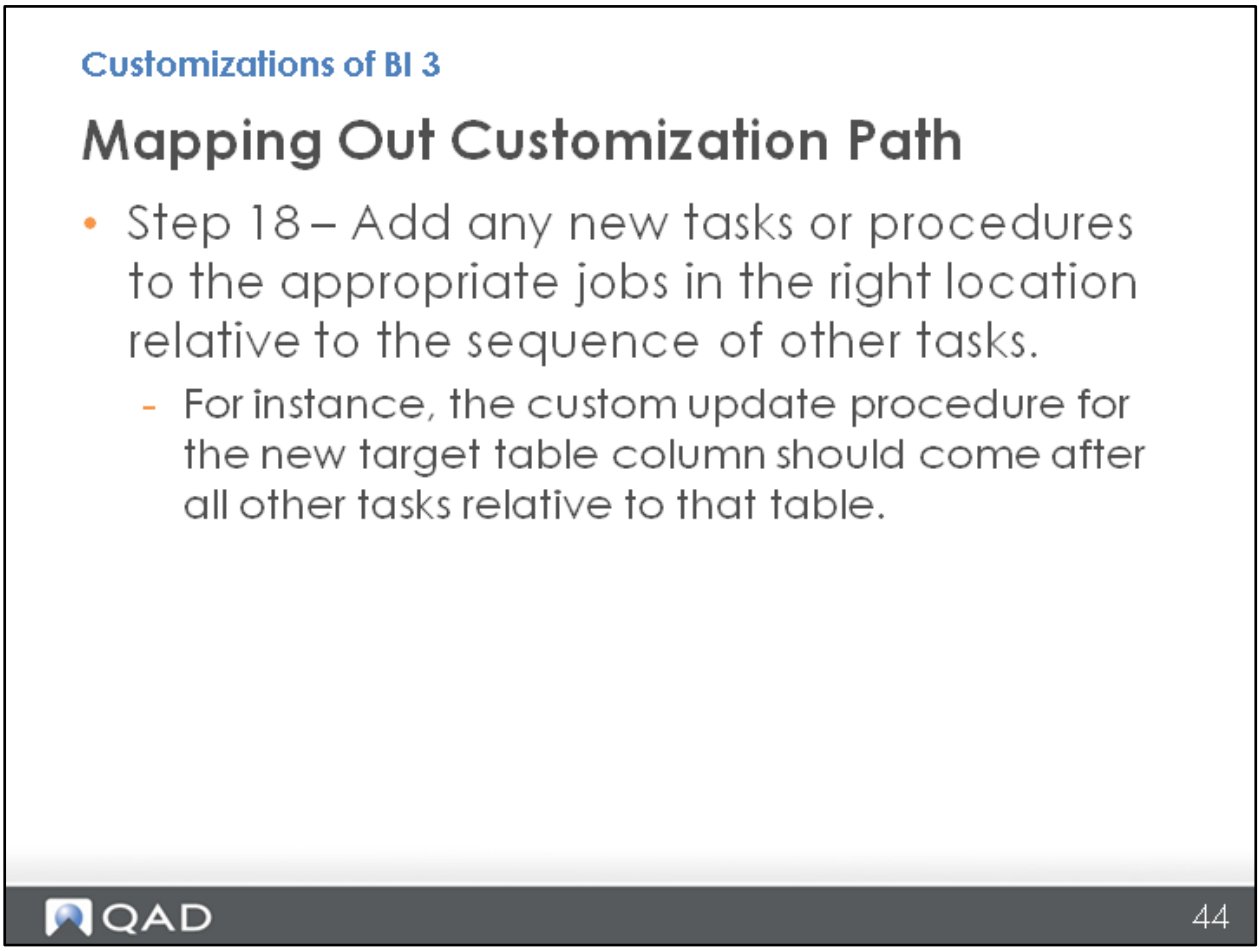
Mapping Out Customization Path



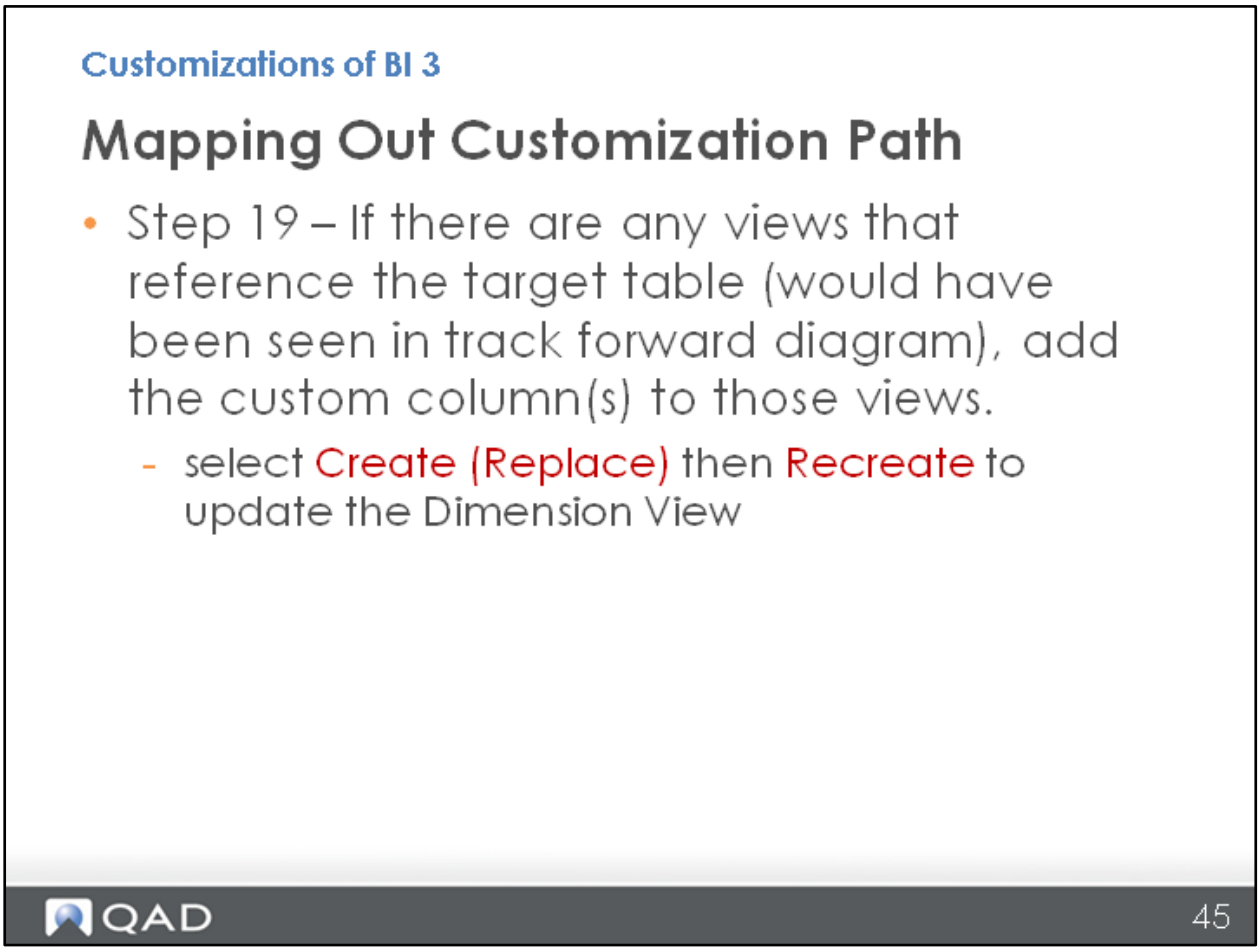
Mapping Out Customization Path



Mapping Out Customization Path



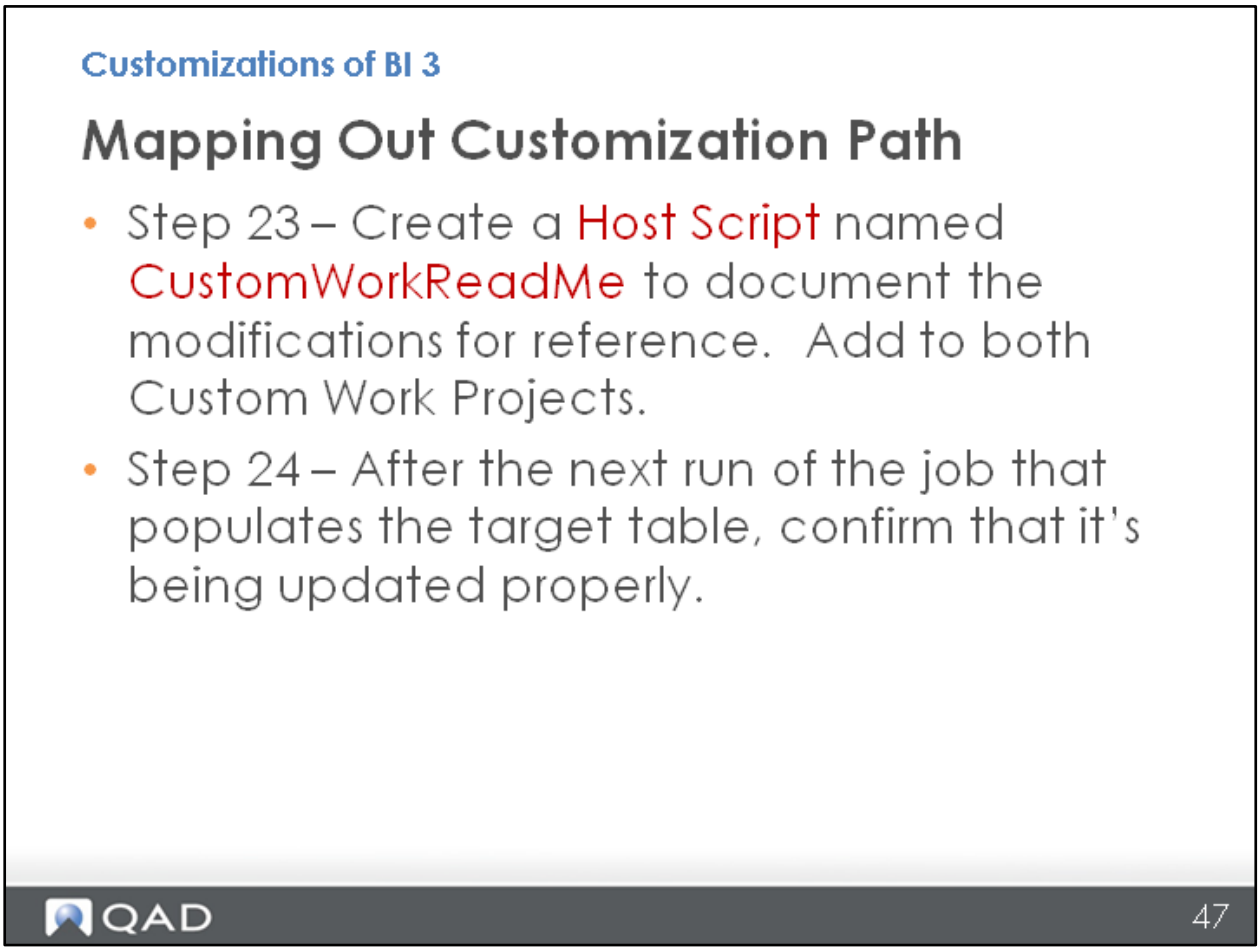
Mapping Out Customization Path



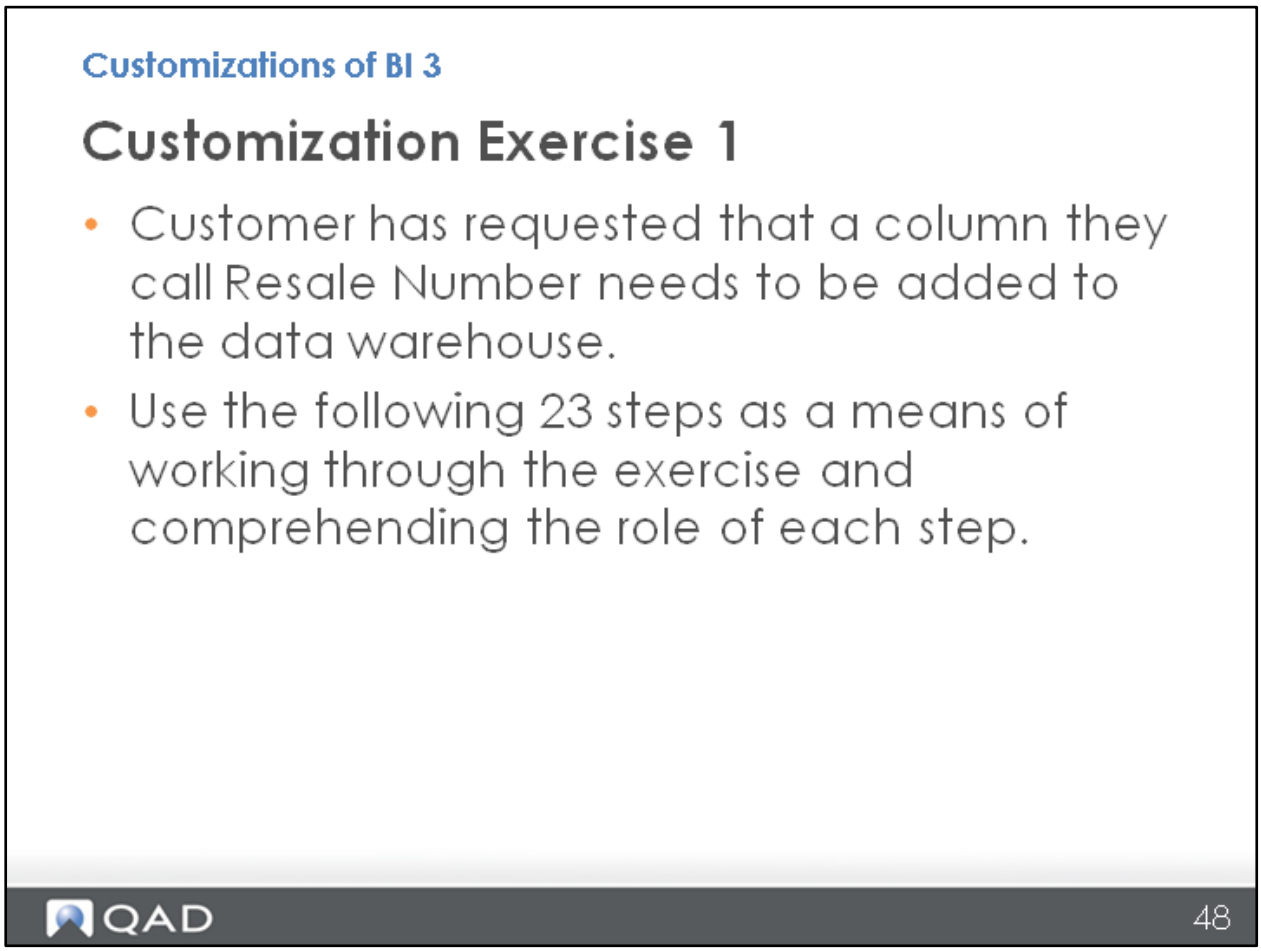
Mapping Out Customization Path



Mapping Out Customization Path

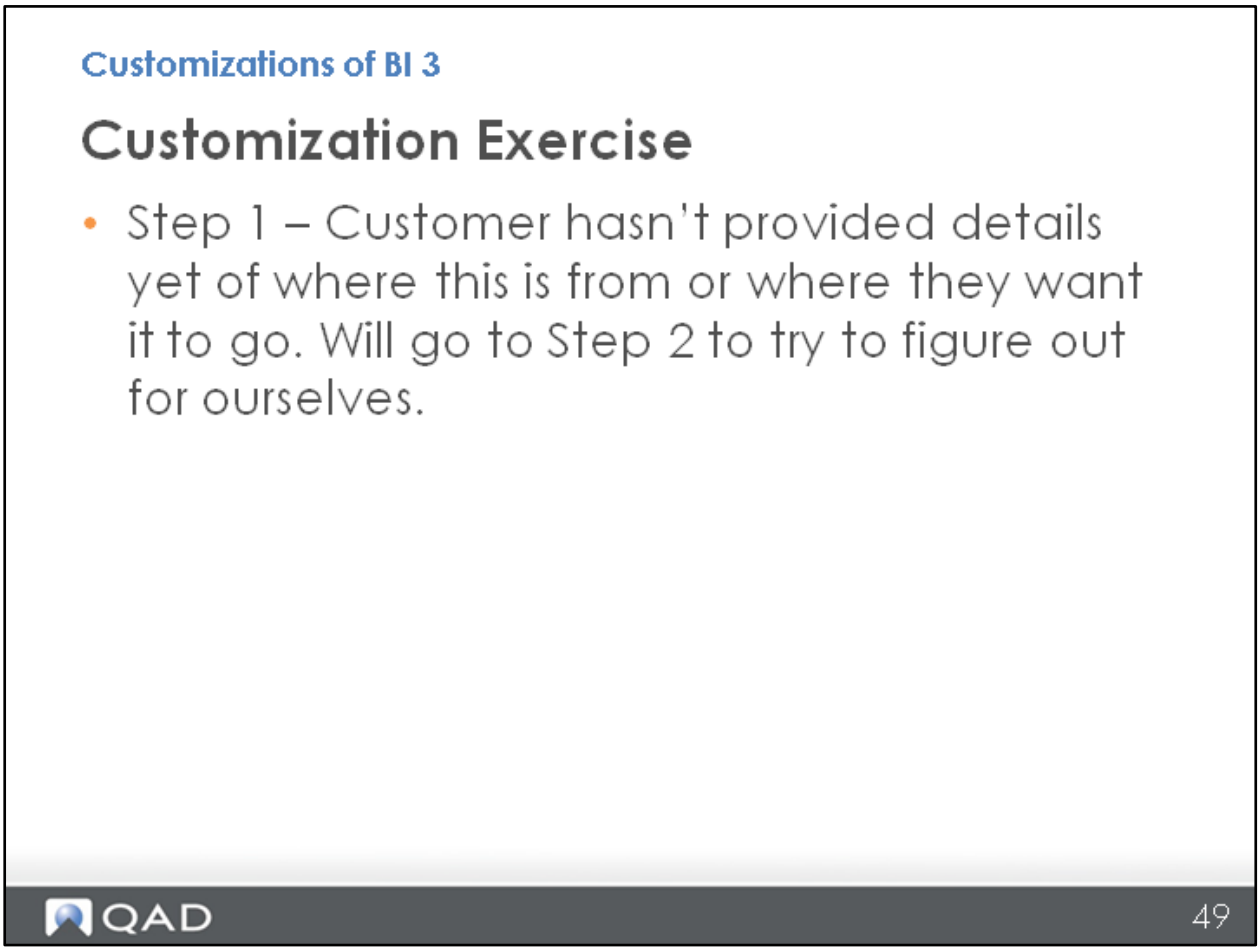


Customization Exercise 1



Use the steps from the prior slides to add this column to the data warehouse.

Customization Exercise

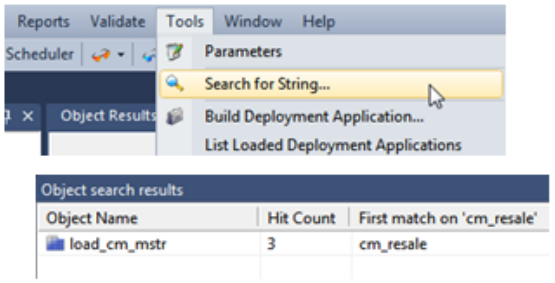


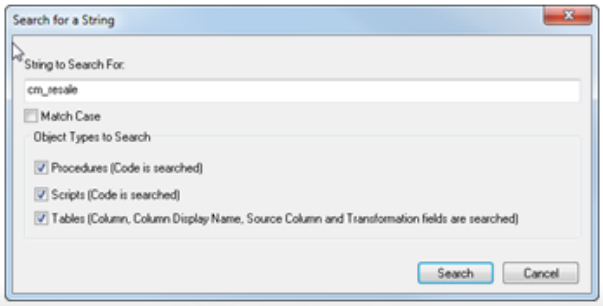
Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 2 – Determine if we already have the table and column in the data warehouse.
- In this case using Search for String reveals a cm_resale column in the load_cm_mstr table. Customer confirms that it is the column and table they were referring to.





QAD 50

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 3 – Try to determine the destination of the load_cm_mstr table since it already exists in our data warehouse. Do a track forward on load_cm_mstr and note the path in Notepad. Also do track back for later review.



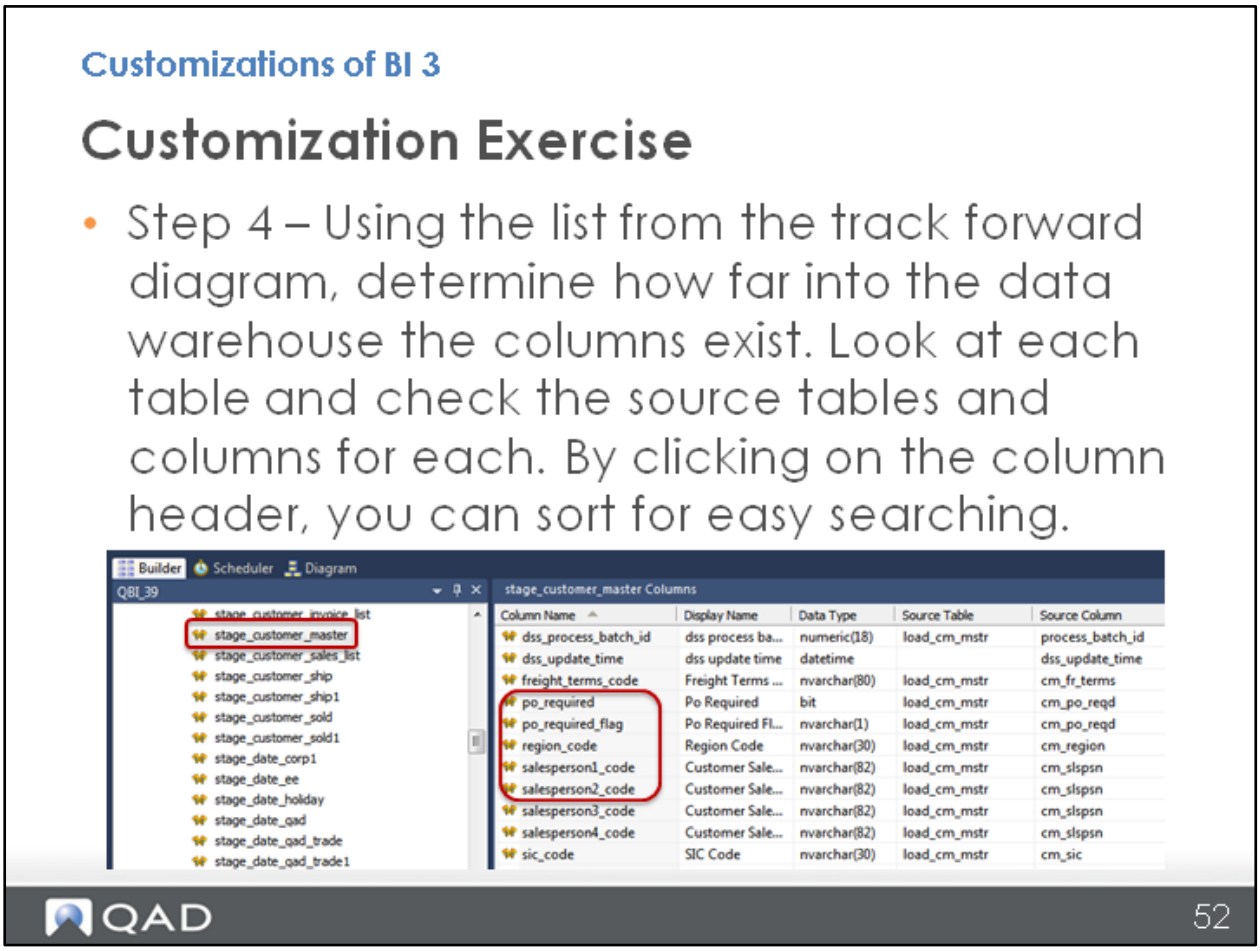
cm_resale.txt - Notepad

File Edit Format View Help

```
load_cm_mstr
stage_customer_master
stage_se_customer
stage_customer
dim_customer
  views:
    dim_customer_bill_to
    dim_customer_ship_to
    dim_customer_sold_to
```

QAD 51

Customization Exercise



In this instance the column only appears as far as load_cm_mstr. It does not appear in the subsequent stage_customer_master table.

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 5 – Use column track back for another column in dim_customer that tracks back to load_cm_mstr to double check the data path.

Report: Column Track Back

Source Table (dim_customer)	Source Column (salesperson1_co...	Transform Code
dim_customer	salesperson1_code	
stage_customer	salesperson1_code	
stage_se_customer	salesperson1_code	
stage_customer_master	salesperson1_code	SUBSTRING(cm_slspsn,1,CHARINDEX(';', cm_slspsn)-1)
load_cm_mstr	cm_slspsn	REPLACE(REPLACE(REPLACE(cm_mstr."cm_slspsn",'|',' '),CHR(10),' '),CHR(13),' ')
cm_mstr	cm_slspsn	

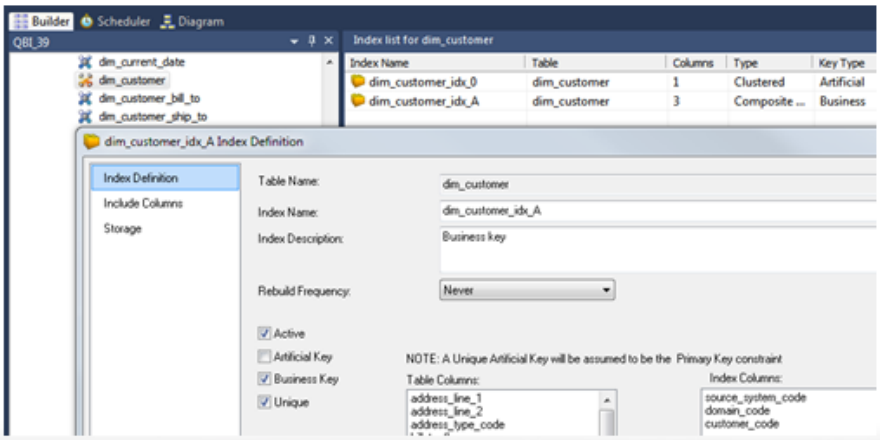
QAD 53

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 6 – To find the business key of the target dim_customer table, in the Builder window, right click on the table and choose Display Indexes. Note the columns in No



QAD 54

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 7 – Determine how many columns in the dim_customer table business key can be referenced in the load_cm_mstr table. We know we already have source_system_code and domain_code. Doing a column track back on customer_code reveals it is sourced from ad_mstr.ad_addr. Note the column data path in your Notepad.

Report: Column Track Back

Source Table (dim_customer)	Source Column (customer_code)	Transform Code
dim_customer	customer_code	
stage_customer	customer_code	
stage_se_customer	customer_code	ISNULL(stage_address_master.customer_code,' ')
stage_address_master	customer_code	
load_ad_mstr	ad_addr	REPLACE(REPLACE(REPLACE(ad_mstr."ad_addr",'[',' '),CHR(10),' '),CHR(13),' ')
ad_mstr	ad_addr	

QAD 55

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 8 – Looking at the track forward diagram for load_cm_mstr , column track back on dim_customer.customer_code which tracks back to load_ad_mstr and also looking at the track back diagram on dim_customer, it's clear that the first place that the cm_mstr and ad_mstr data may be joined is in stage_se_customer.



QAD 56

Customization Exercise

Customizations of BI 3

Customization Exercise

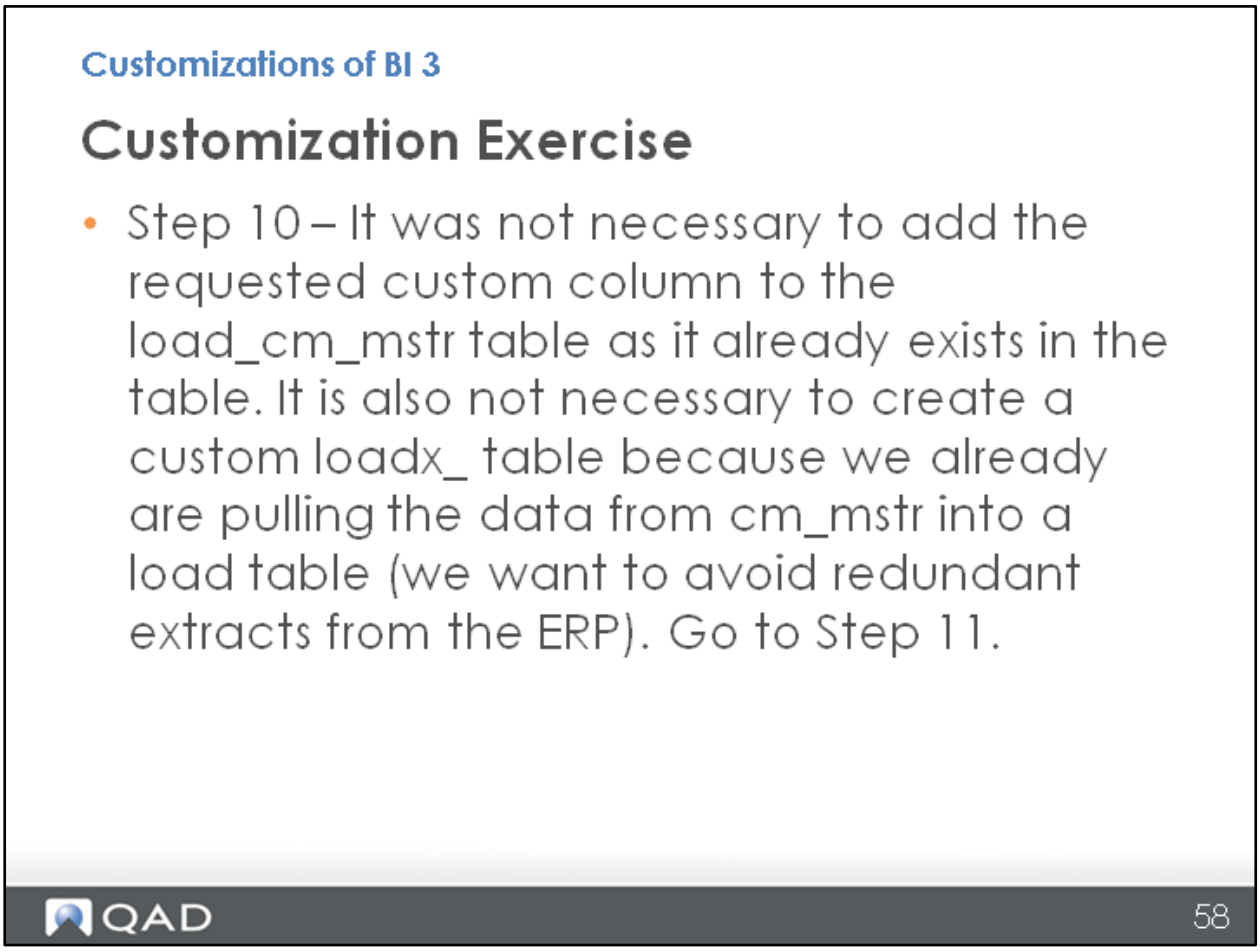
- Step 9 – Having found the potential join point, check the stored procedure for that table to confirm how the two tables are joined. Copy the code from here into your Notepad.

Builder | Scheduler | Diagram | update_stage_se_customer

```
    stage_address_master.dss_extract_timestamp,
    stage_address_master.dss_process_batch_id,
    @v_dss_update_time
FROM stage_address_master
LEFT OUTER JOIN stage_customer_master
ON stage_address_master.source_system_code = stage_customer_master.source_system_code
AND stage_address_master.domain_code = stage_customer_master.domain_code
AND stage_address_master.customer_code = stage_customer_master.customer_code
LEFT OUTER JOIN stage_customer_bill
ON stage_address_master.source_system_code = stage_customer_bill.source_system_code
AND stage_address_master.domain_code = stage_customer_bill.domain_code
AND stage_address_master.customer_code = stage_customer_bill.bill_to_code
LEFT OUTER JOIN stage_customer_sold
```

QAD 57

Customization Exercise

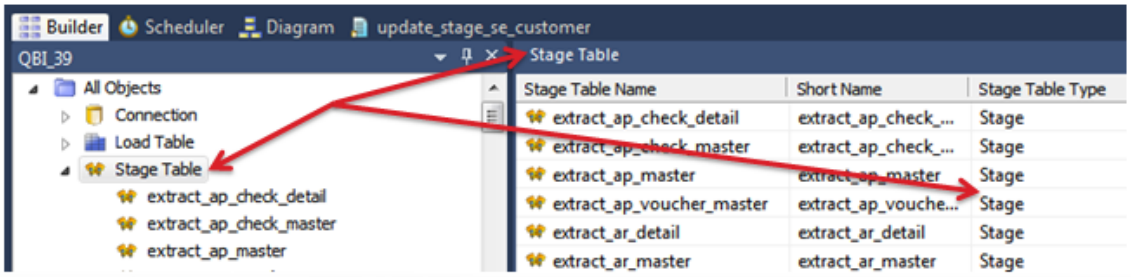


Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build the necessary custom staging tables for getting our data to the point it can be referenced by our soon to be created custom update procedure.
  - In the Builder window, double click on Stage Table to get the stage table list in the Drop Target Pane.



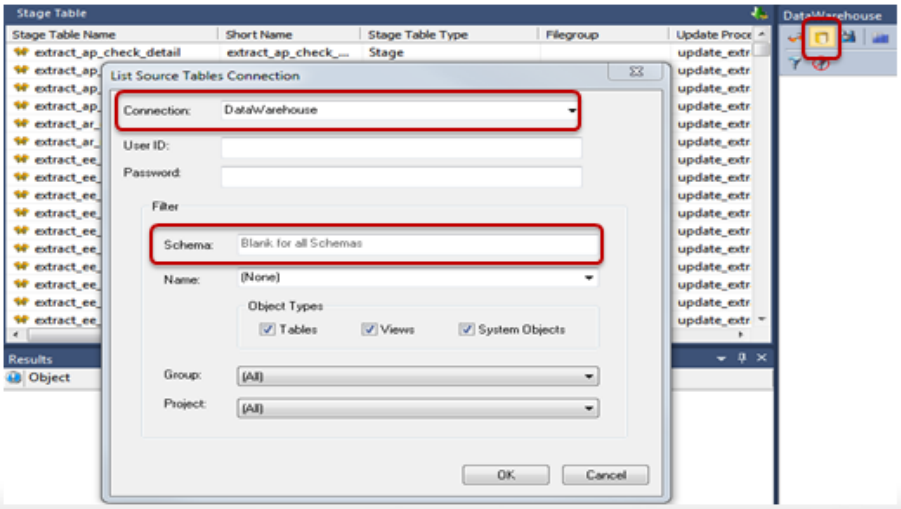
QAD 59

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build staging tables – continued
  - In the source Browser Pane, set the Connection to DataWarehouse (make sure the Schema is blank).



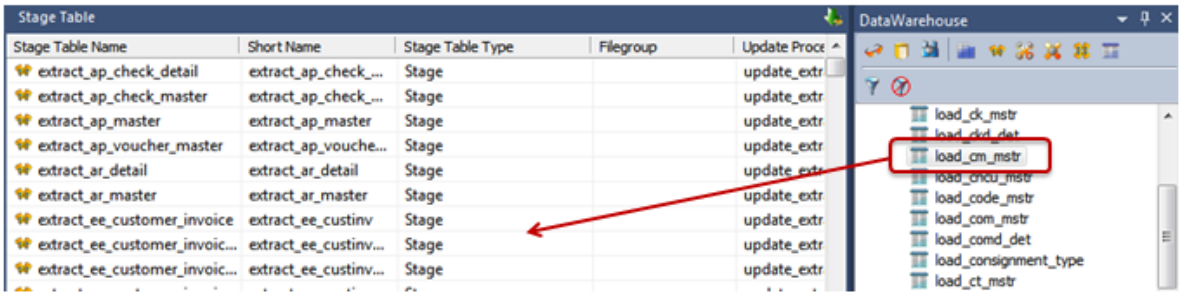
QAD 60

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build staging tables – continued
  - Find the load_cm_mstr table in the Browser Pane and drag and drop it into the Drop Target Pane.



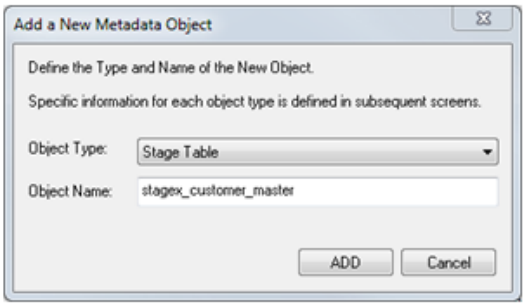
QAD 61

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build staging tables – continued
  - When prompted for the name of the new staging table, name it stagex_customer_master.



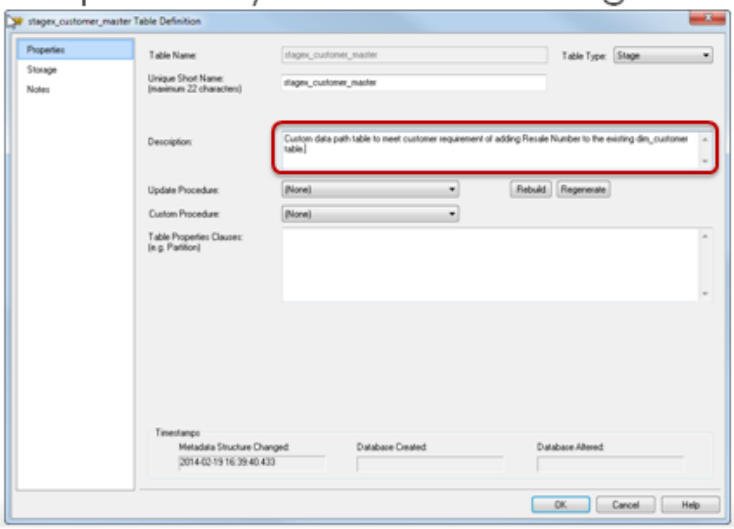
QAD 62

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build staging tables – continued
  - In the Table Definition window, enter a description to explain why this table is being created.



QAD 63

Customization Exercise

Customizations of BI 3

Customization Exercise

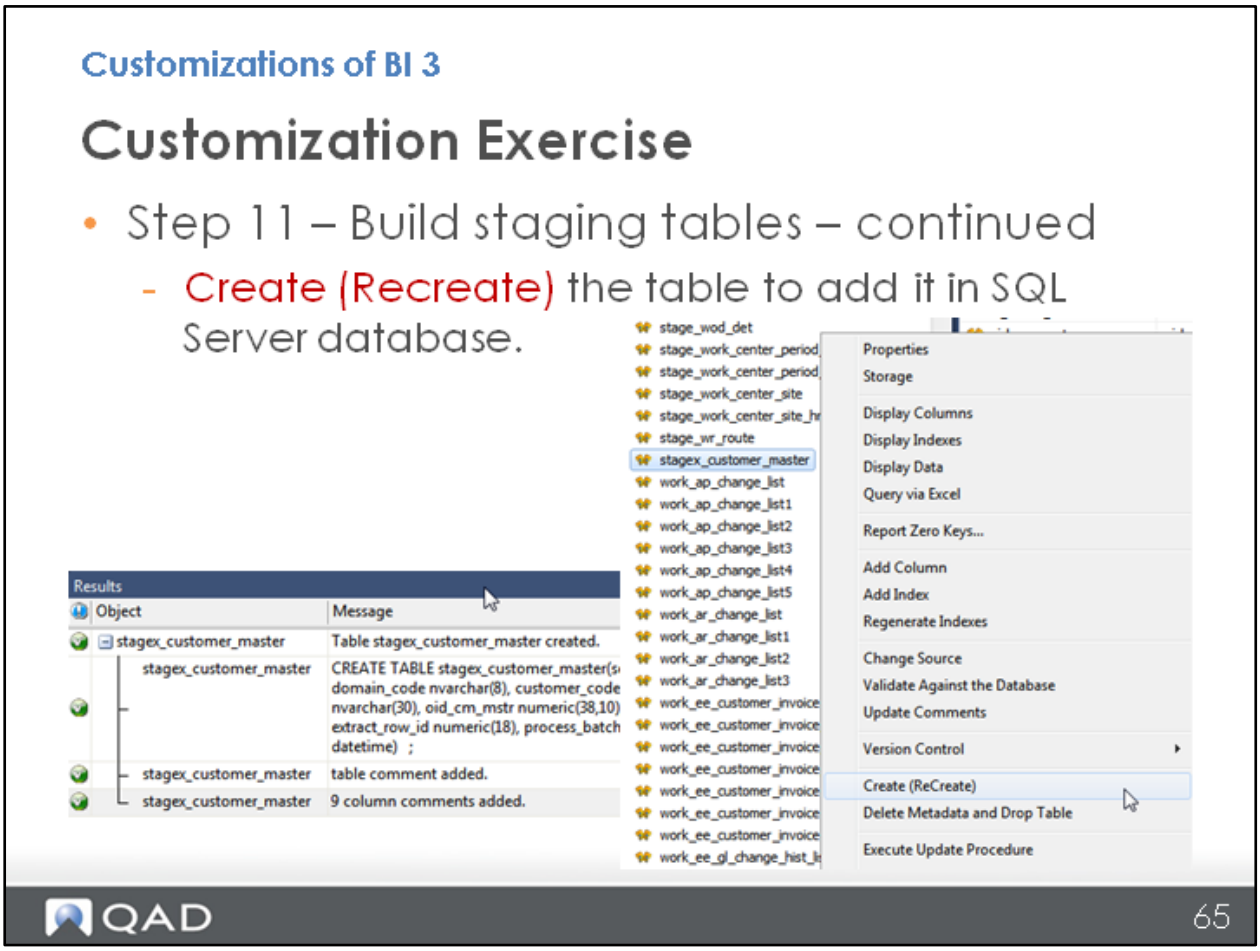
- Step 11 – Build staging tables – continued
  - When the resulting list of columns for the new stagex_customer_master table appears, delete any unnecessary columns, leaving the business key columns, the new xx_resale_number column and the handful of timestamp related columns at the end of the table.

stagex_customer_master Columns

Column Name	Display Name	Data Type	Source Table	Source Column	Format
source_system_code	source system ...	nvarchar(64)	load_cm_mstr	source_system_code	
domain_code	cm domain	nvarchar(8)	load_cm_mstr	cm_domain	
customer_code	cm addr	nvarchar(80)	load_cm_mstr	cm_addr	
xx_resale_number	cm resale	nvarchar(30)	load_cm_mstr	cm_resale	
dss_extract_timestamp	extract timesta...	datetime	load_cm_mstr	extract_timestamp	
dss_process_batch_id	process batch id	numeric(18)	load_cm_mstr	process_batch_id	#,##0
dss_update_time	dss update time	datetime		dss_update_time	

QAD 64

Customization Exercise

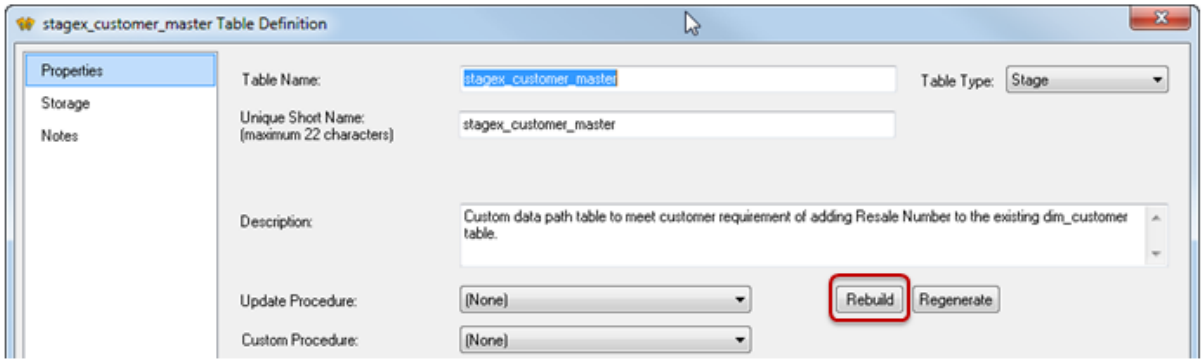


Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build staging tables – continued
  - Build the procedure for the table. In the Builder window, double click on the new table name and click Rebuild.



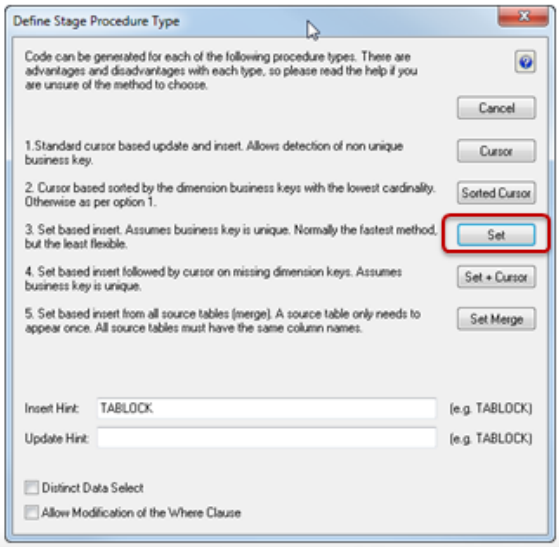
QAD 66

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build staging tables – continued
  - Build the procedure for the table – continued. Choose the Set option (unless you are certain you should be using a different option).



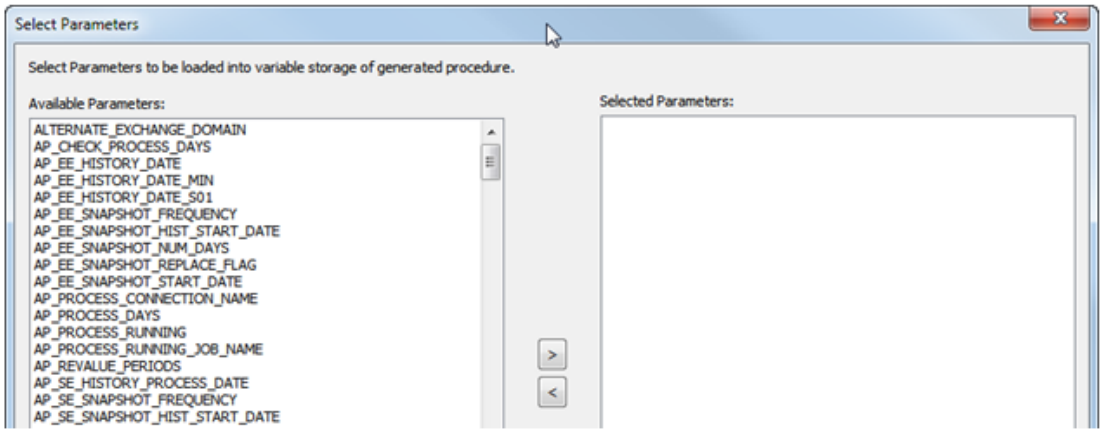
QAD 67

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build staging tables – continued
  - Build the procedure for the table – continued. Add any appropriate parameters, if any.



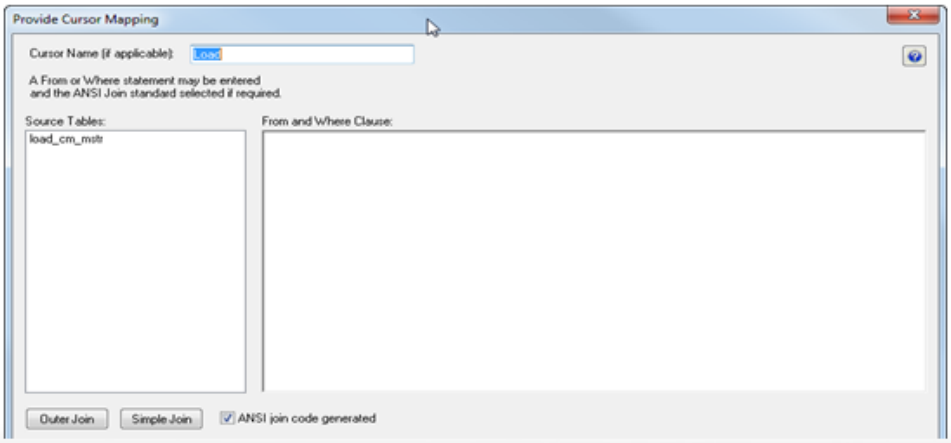
QAD 68

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build staging tables – continued
  - Build the procedure for the table – continued. Add any table JOIN(s) or WHERE clause constraints. In this case there weren't any to add.



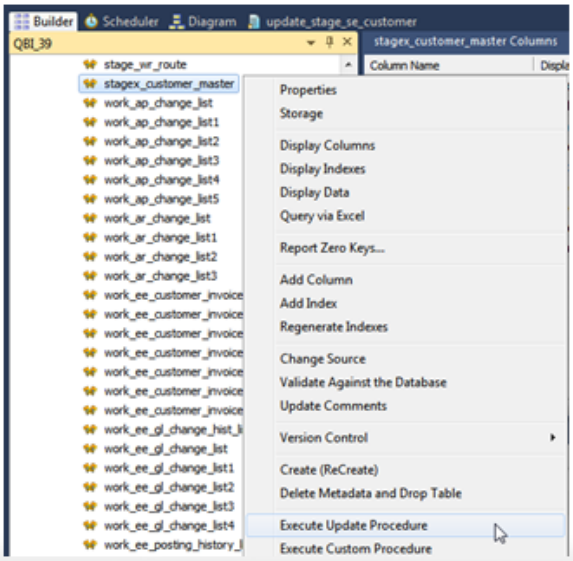
QAD 69

Customization Exercise

Customizations of BI 3

Customization Exercise

- Step 11 – Build staging tables – continued
  - Execute the update procedure to populate the table.



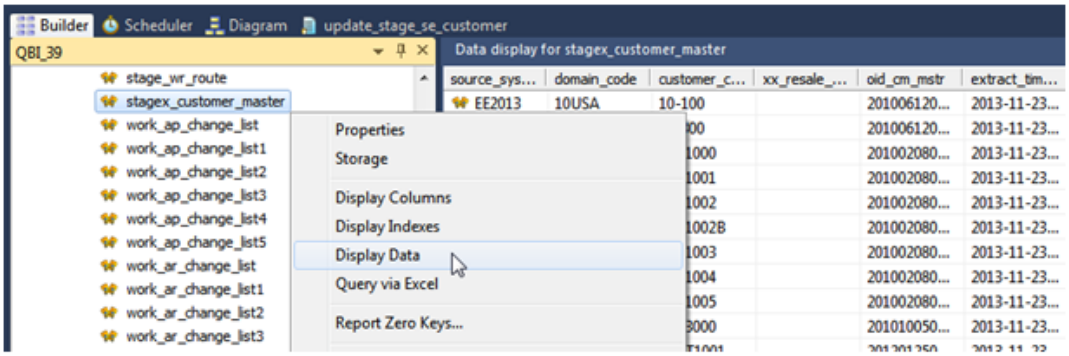
QAD 70

Customization Exercise

Customizations of BI 3

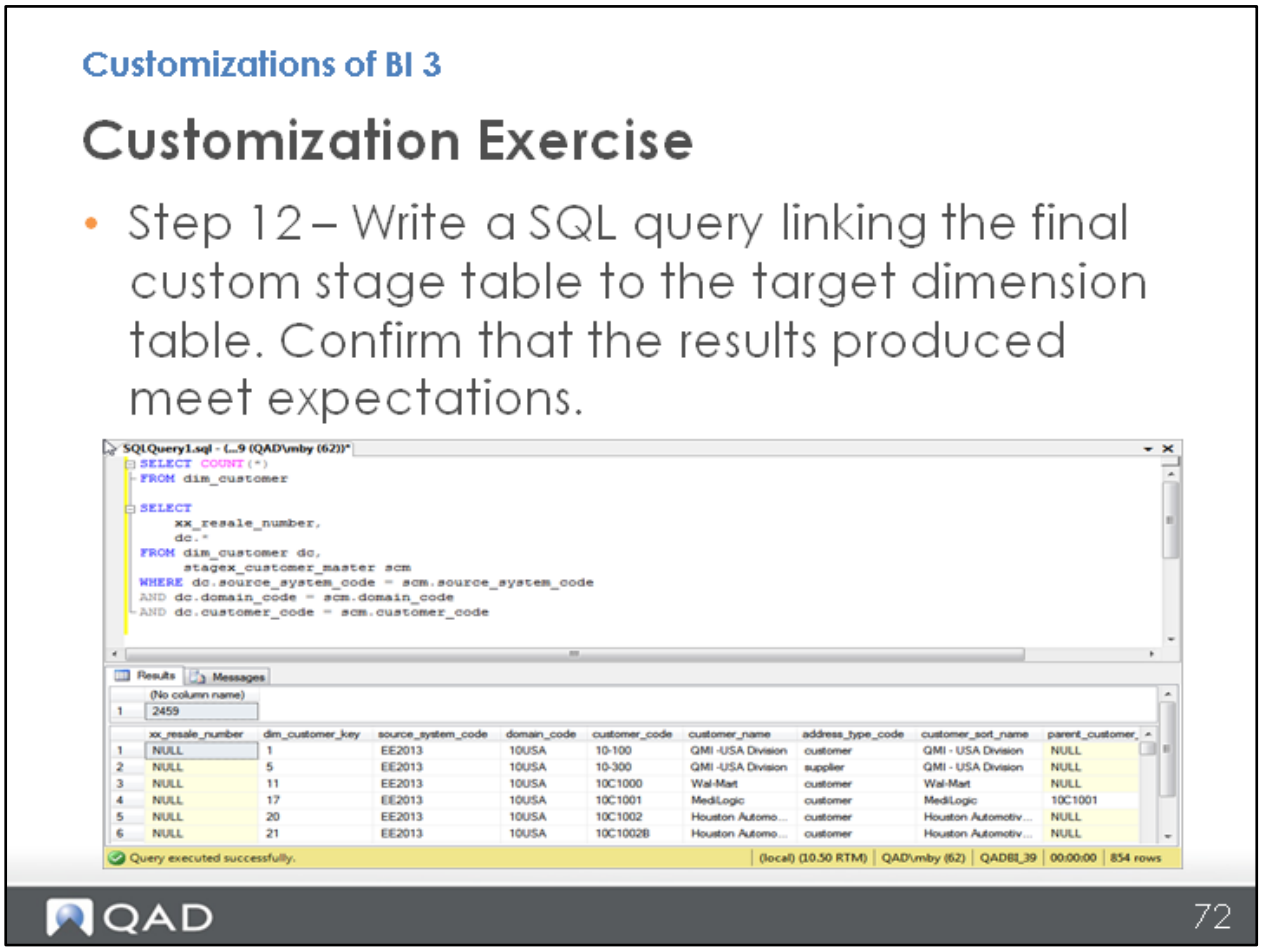
Customization Exercise

- Step 11 – Build staging tables – continued
  - Confirm that data gets into the new staging table, by right clicking on the table name and choosing Display Data.



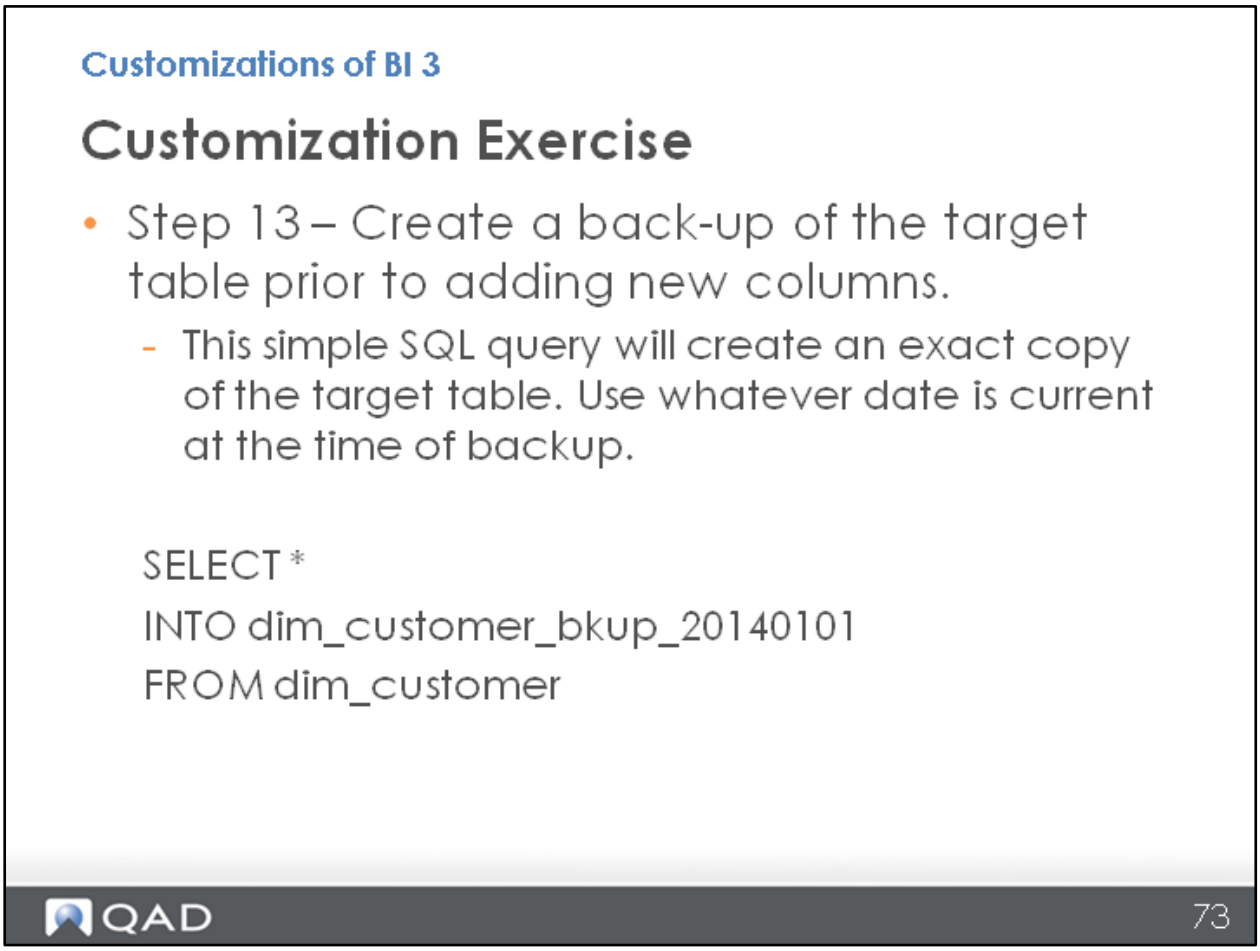
QAD 71

Customization Exercise

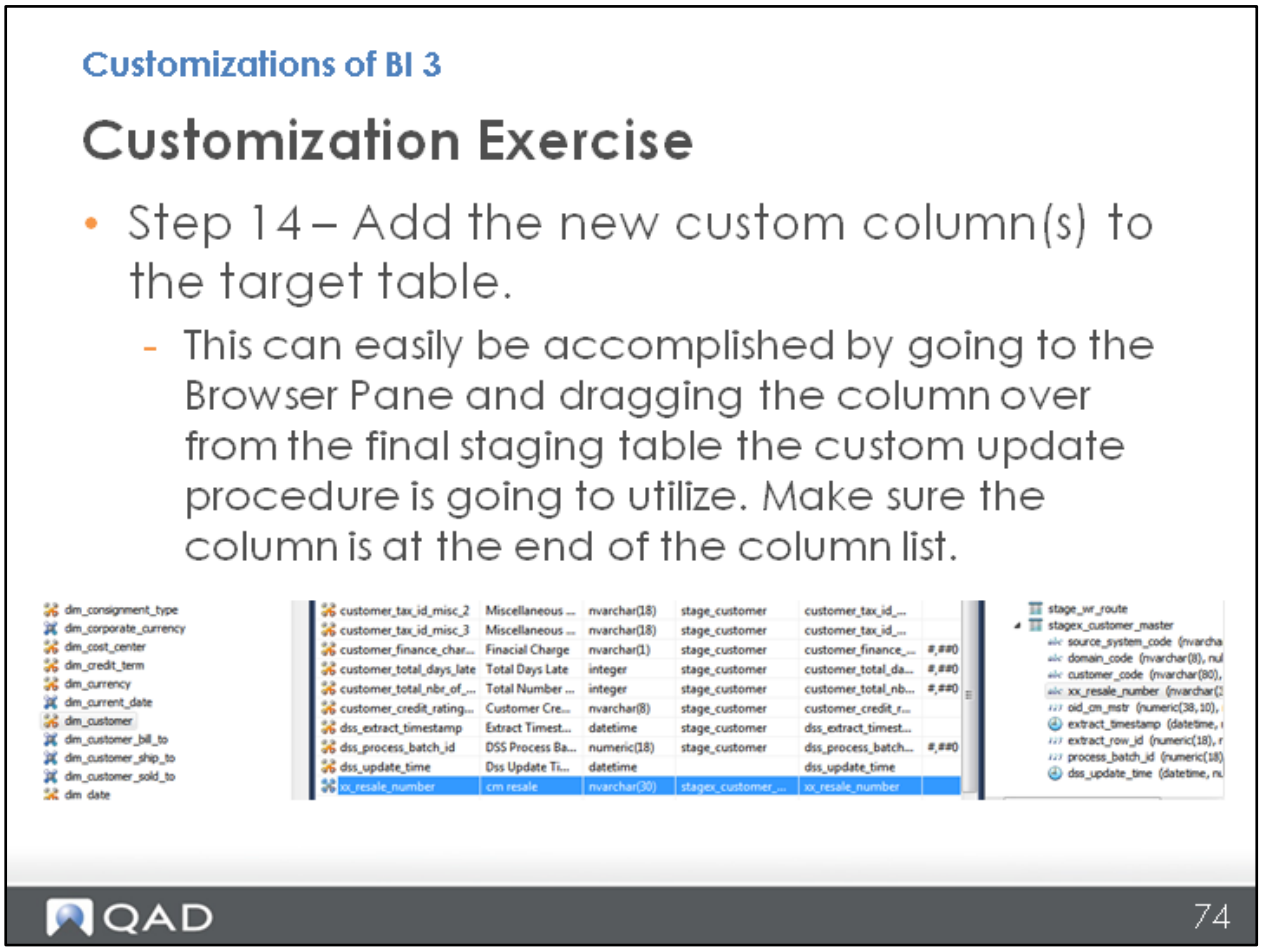


By confirming results, for instance, you shouldn't expect to see more results in your results sets than the count of the records in the target table.

Customization Exercise

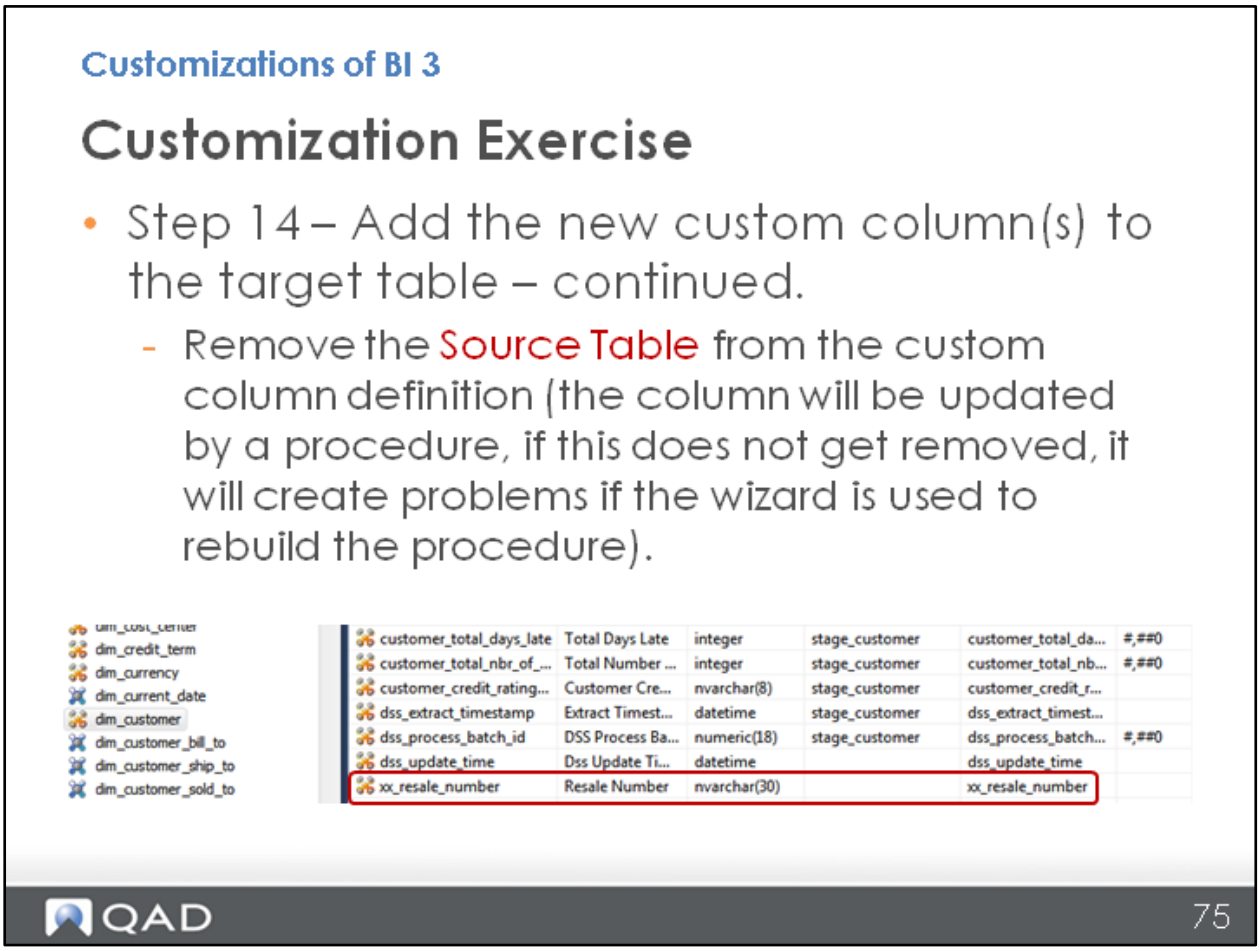


Customization Exercise



If it is difficult to place the new custom column(s) to the end of the column list, an alternate option is to go to the last column, copy the order number of that column, paste it in to the custom column order number and then add a number to it, thereby ensuring that the numbers of the custom column(s) is/are the highest.

Customization Exercise



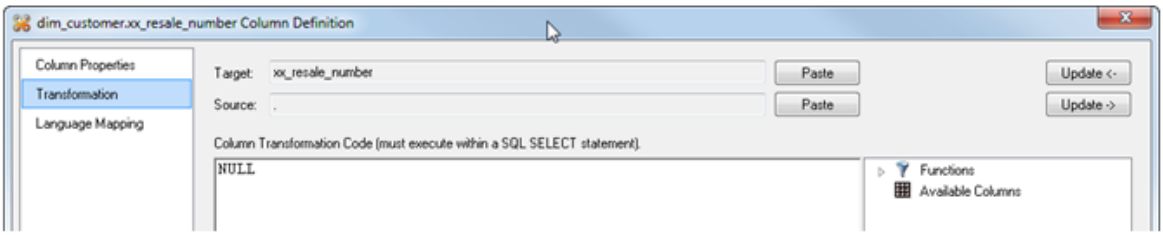
This can easily be accomplished by dragging the column over from the final staging table the custom update procedure is going to utilize.
First find the target table in the Builder/Browser pane and click on it once so it's columns show up in the Target pane. Then in the Source pane, find the final stage table to be referenced by the custom update procedure. If you don't see that last stage table in the Source pane, try Refreshing the Connection from the Connection button. Once the table is found in the Source pane, double click on it to find the columns to be added and drag and drop them into the target pane. Make sure to place them at the end of the column list.

Customization Exercise

Customizations of BI 3

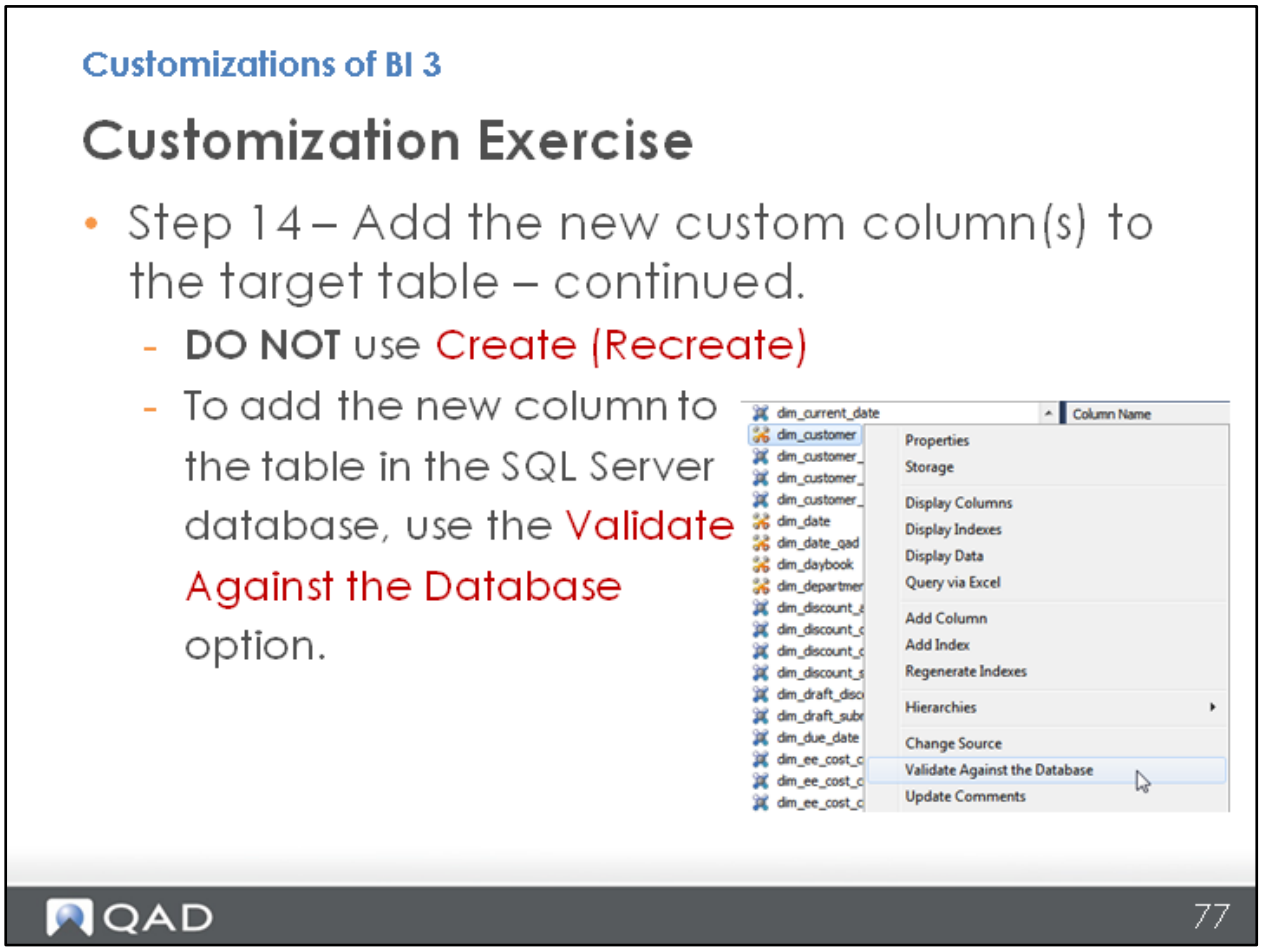
Customization Exercise

- Step 14 – Add the new custom column(s) to the target table – continued.
  - In the new custom column's Transformation tab, set the value to NULL.



QAD 76

Customization Exercise



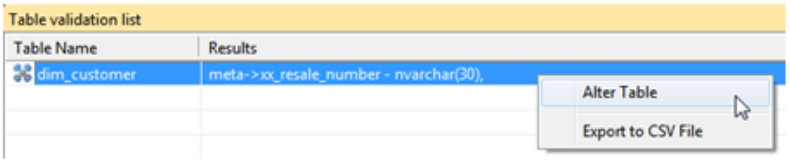
Create(Recreate) on fact/dim/perm tables in systems that are already populated with data will create all sorts of headaches as the data in the table gets truncated and the table rebuilt. Dimensions are the worst since they are referenced by multiple fact tables and may not populate the same way, so all the keys will be broken.

Customization Exercise

Customizations of BI 3

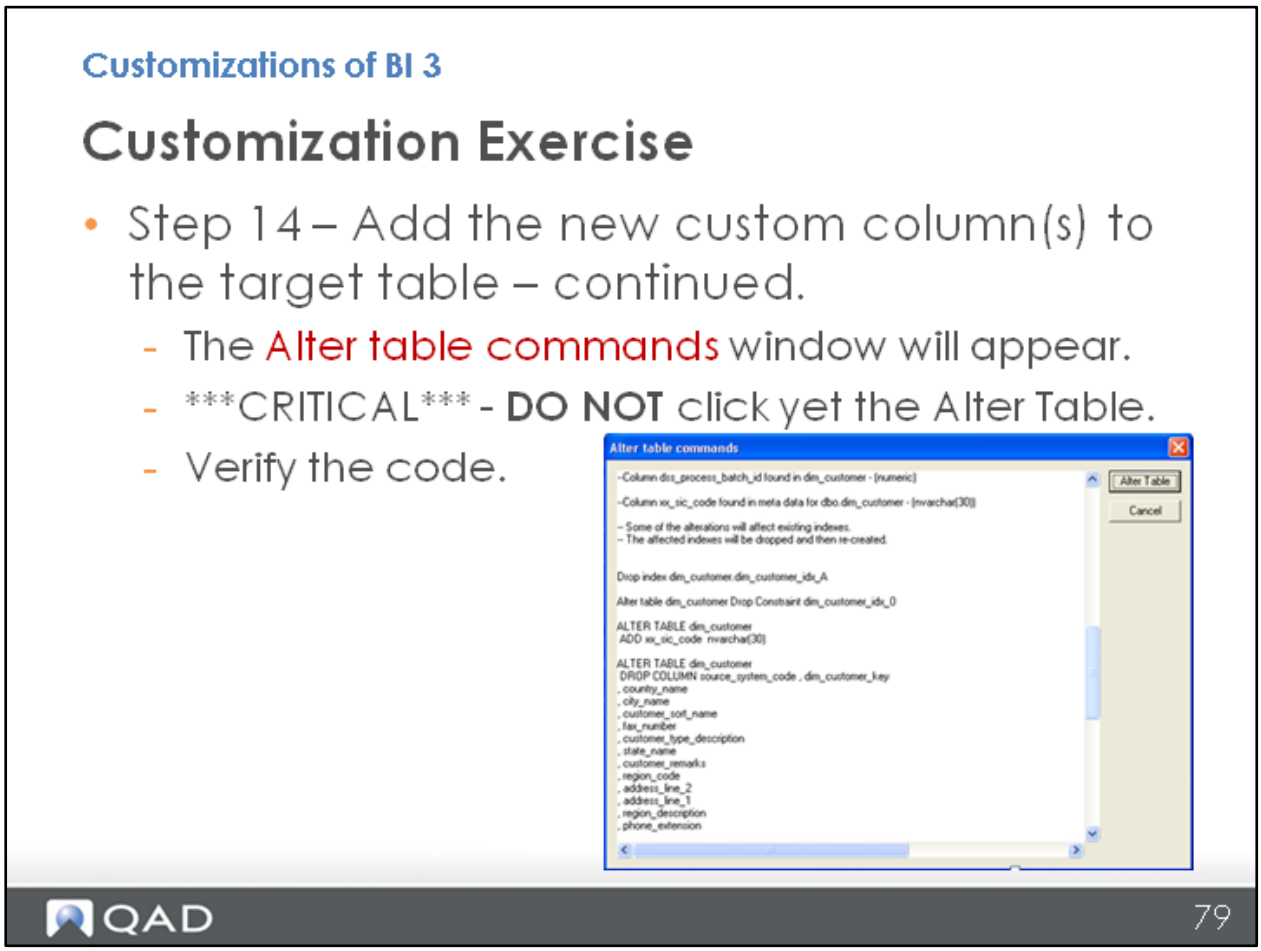
Customization Exercise

- Step 14 – Add the new custom column(s) to the target table – continued.
  - When the Table validation list appears, it will display any columns that are in the DWD's metadata, but not in the SQL Server database.
  - Right click on the table name and pick the Alter Table option.



QAD 78

Customization Exercise



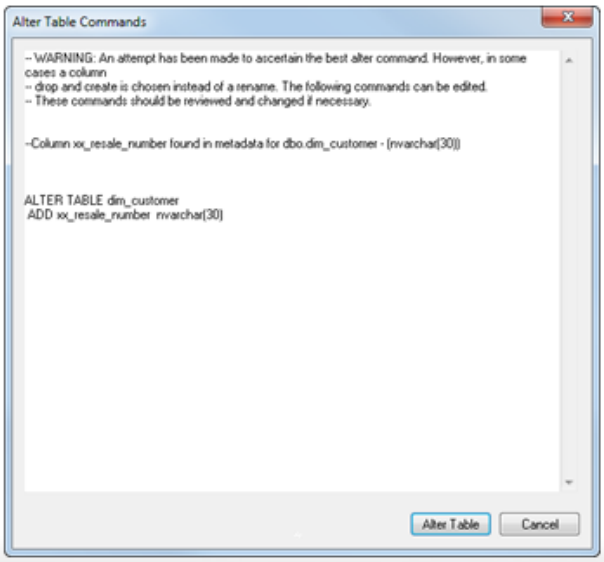
Note that when adding columns to some types of tables like dim_tables using the Validate against database option, the Alter table commands window will sometimes include problematic command logic like dropping all the columns except for the new custom column. This would be completely disastrous for the dim as we would then have a dim table with only one column. Remove all the other logic here so that all that remains is the ALTER TABLE ADD column command.

Customization Exercise

Customizations of BI 3

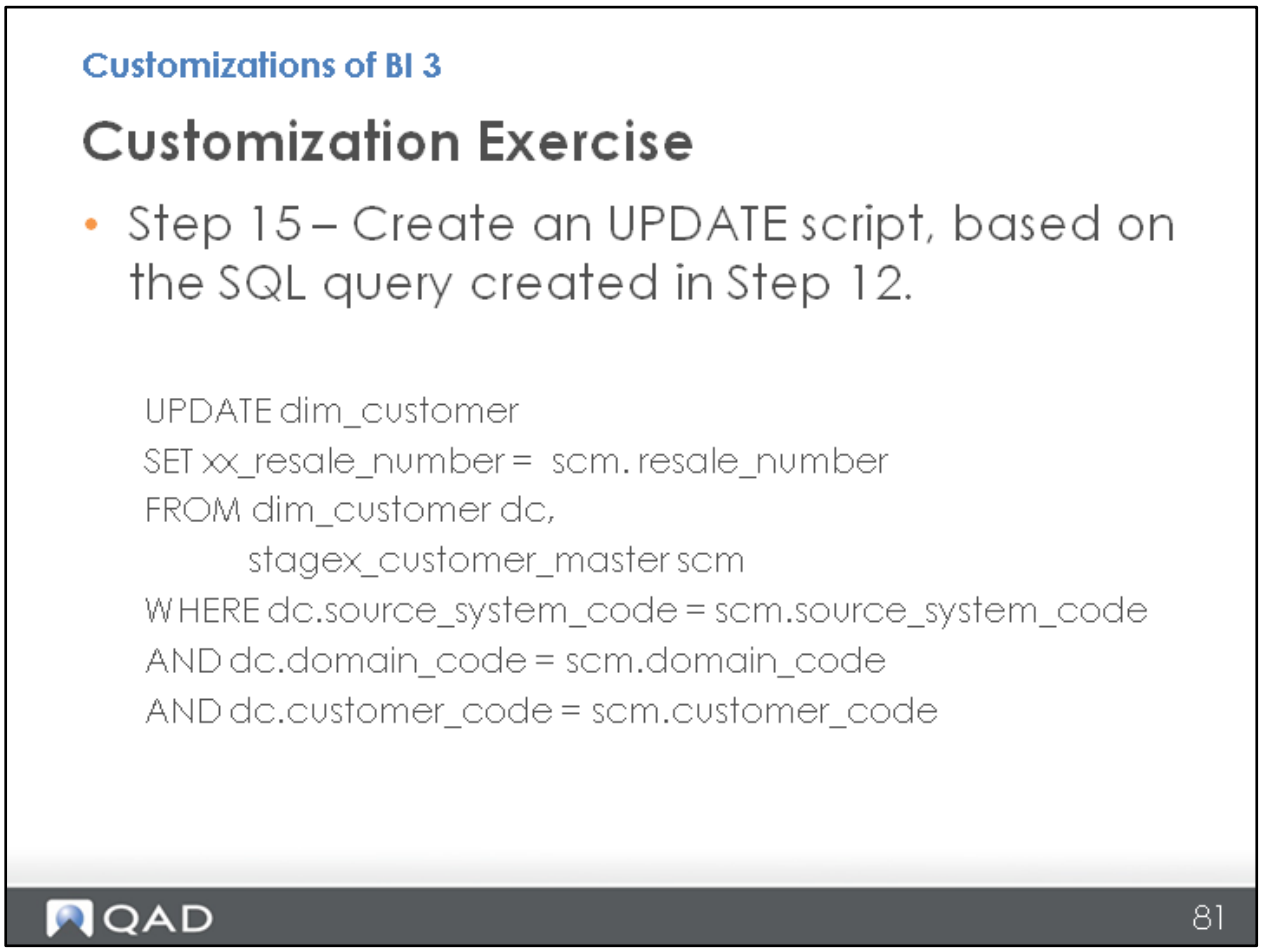
Customization Exercise

- Step 14 – Add the new custom column(s) to the target table – continued.
  - With the commands verified, click the Alter Table button. The new custom column now also resides in the SQL Server database.

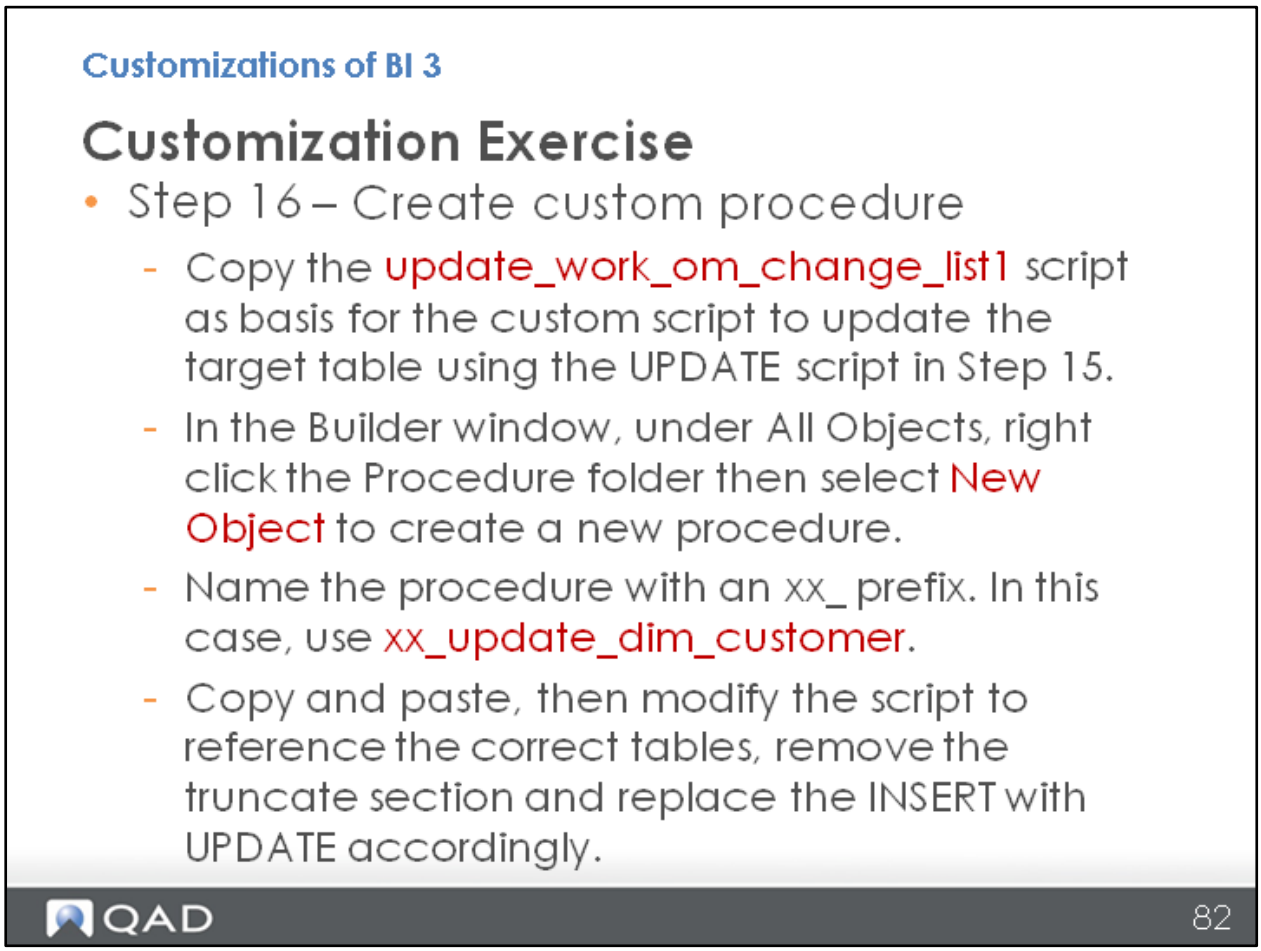


QAD 80

Customization Exercise

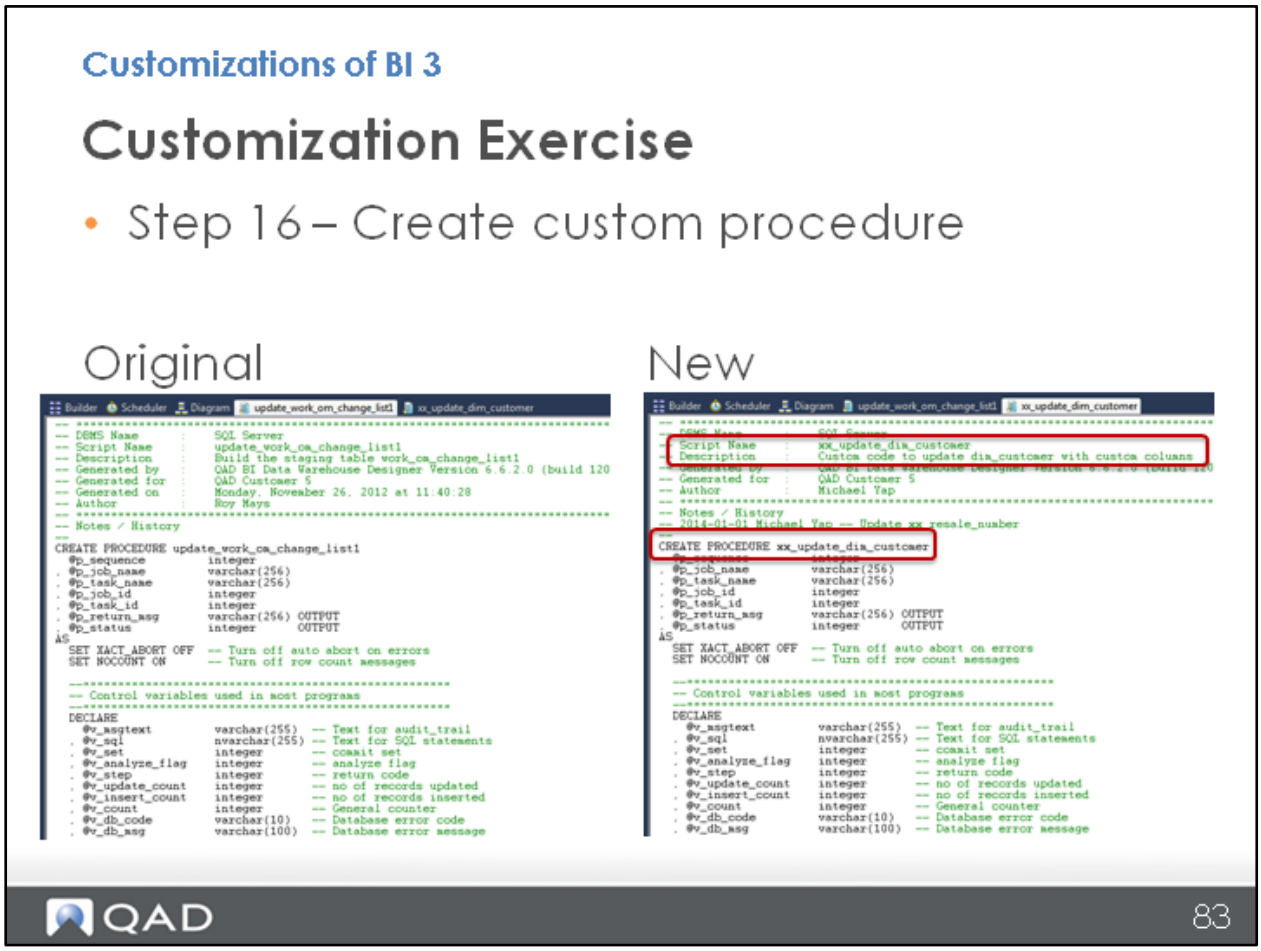


Customization Exercise



Things to be replaced in the procedure includes the comments at the top of the page, the procedure name, the table truncate portion, the insert script (replace with the update script), the insert_count variable which needs to be replaced with the update_count variable, and any comments so they properly refer to the dim_customer table (instead of work_om_change_list1).

Customization Exercise



Be sure as you go through the old code that was copied to change the procedure name, any reference to that procedure or table to point to the correct procedure name or table. Also remove any truncates if this is an update script and point the post update record count (after error check) to point at the update variable, not the insert variable. Clean up any other update notes to make sure everything is relevant (for instance, there is a warning message for updates on the update_work_om_change_list1 procedure that can be completely removed for our new update procedure).

Customization Exercise

Customizations of BI 3

Customization Exercise

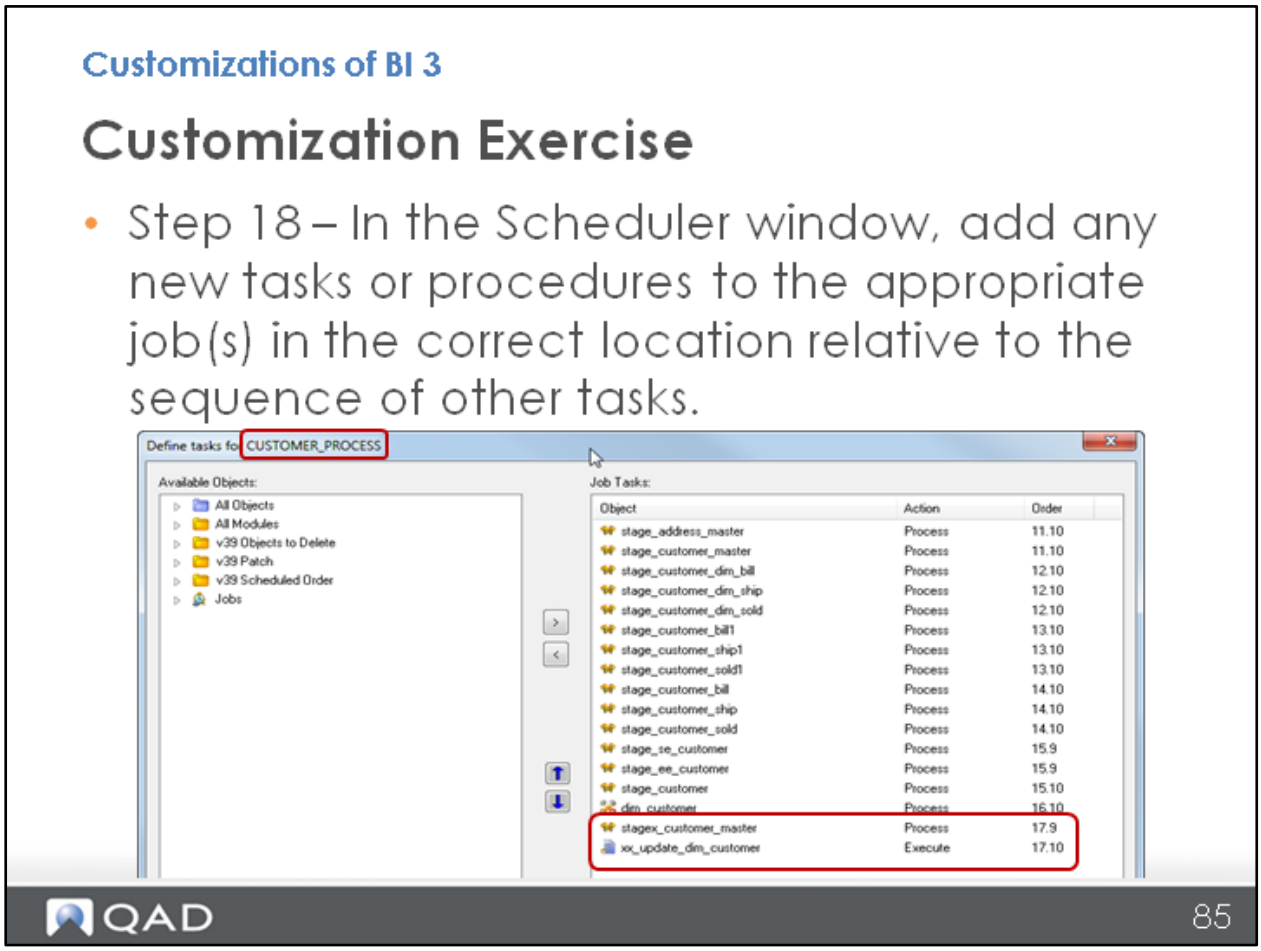
- Step 17 – Find which Job(s) currently update this table.
  - Use the Reports/ Job Reports/ Jobs with an Object option to do this.

Report: Jobs With an Object

Jobs including dim_customer	Action
CUSTOMER_PROCESS	Process
HIST_COMMON_PROCESS_CHAINED	Process
HIST_COMMON_PROCESS_CHAINED	Process

QAD 84

Customization Exercise



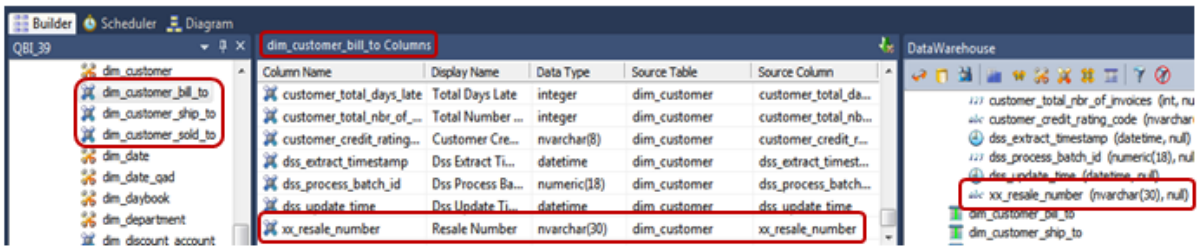
For instance the custom update procedure for the new target table column should come after all other tasks relative to that table.

Customization Exercise

Customizations of BI 3

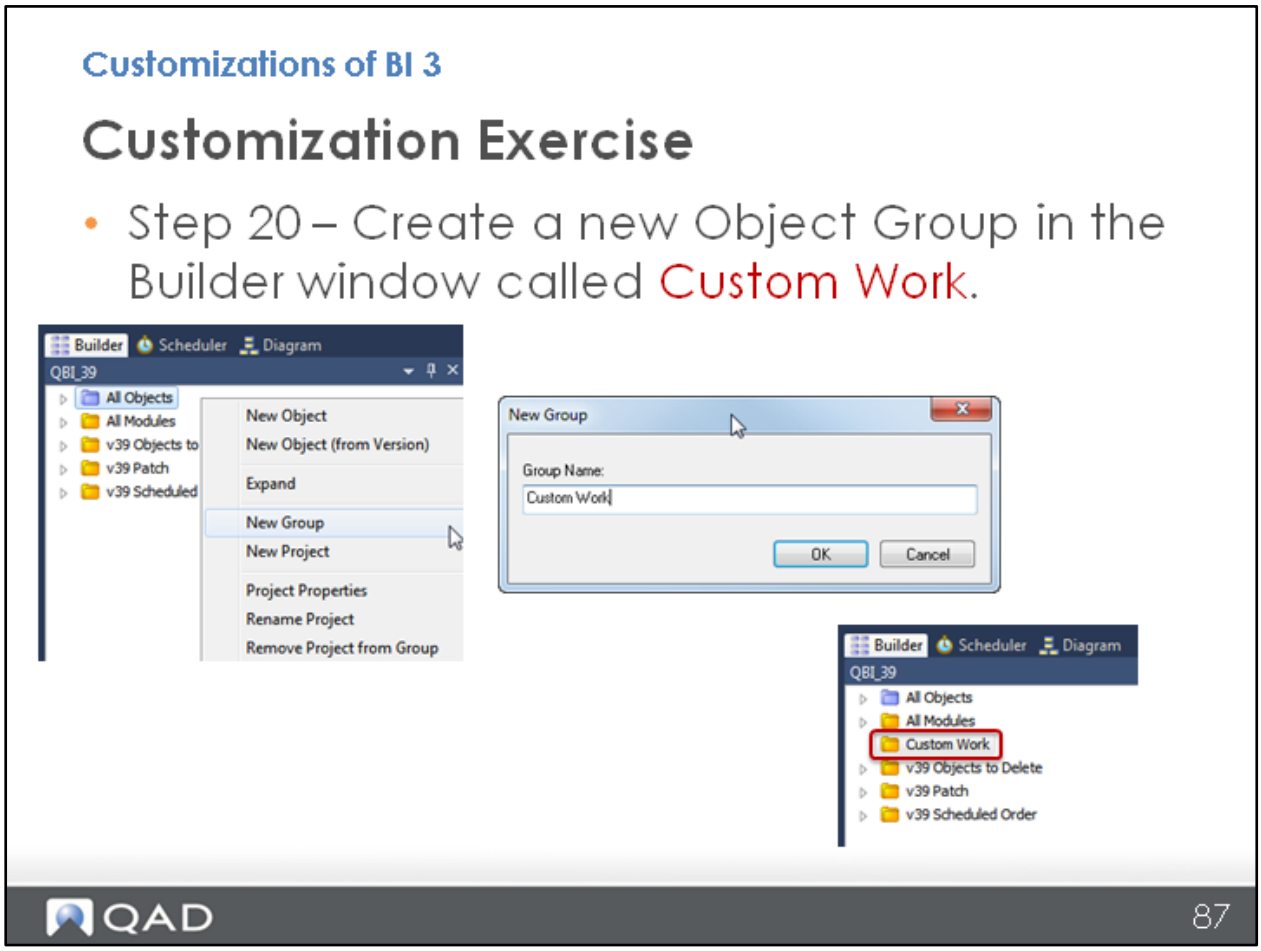
Customization Exercise

- Step 19 – If there are any views that reference the target table (would have been seen in track forward diagram), add the custom column(s) to those views by dragging and dropping the custom column from the target table to the dimension view and invoking the Create (Replace).

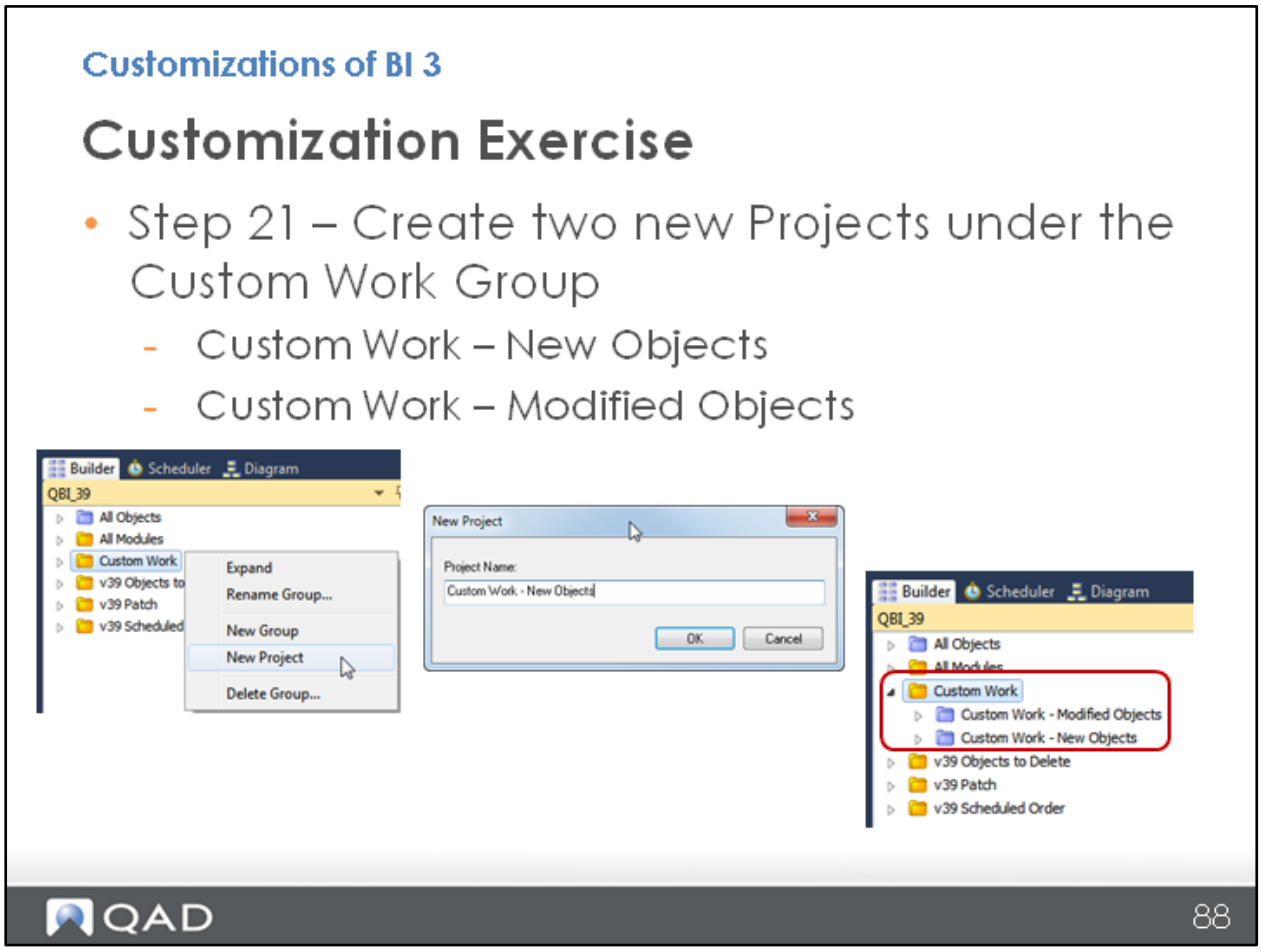


QAD 86

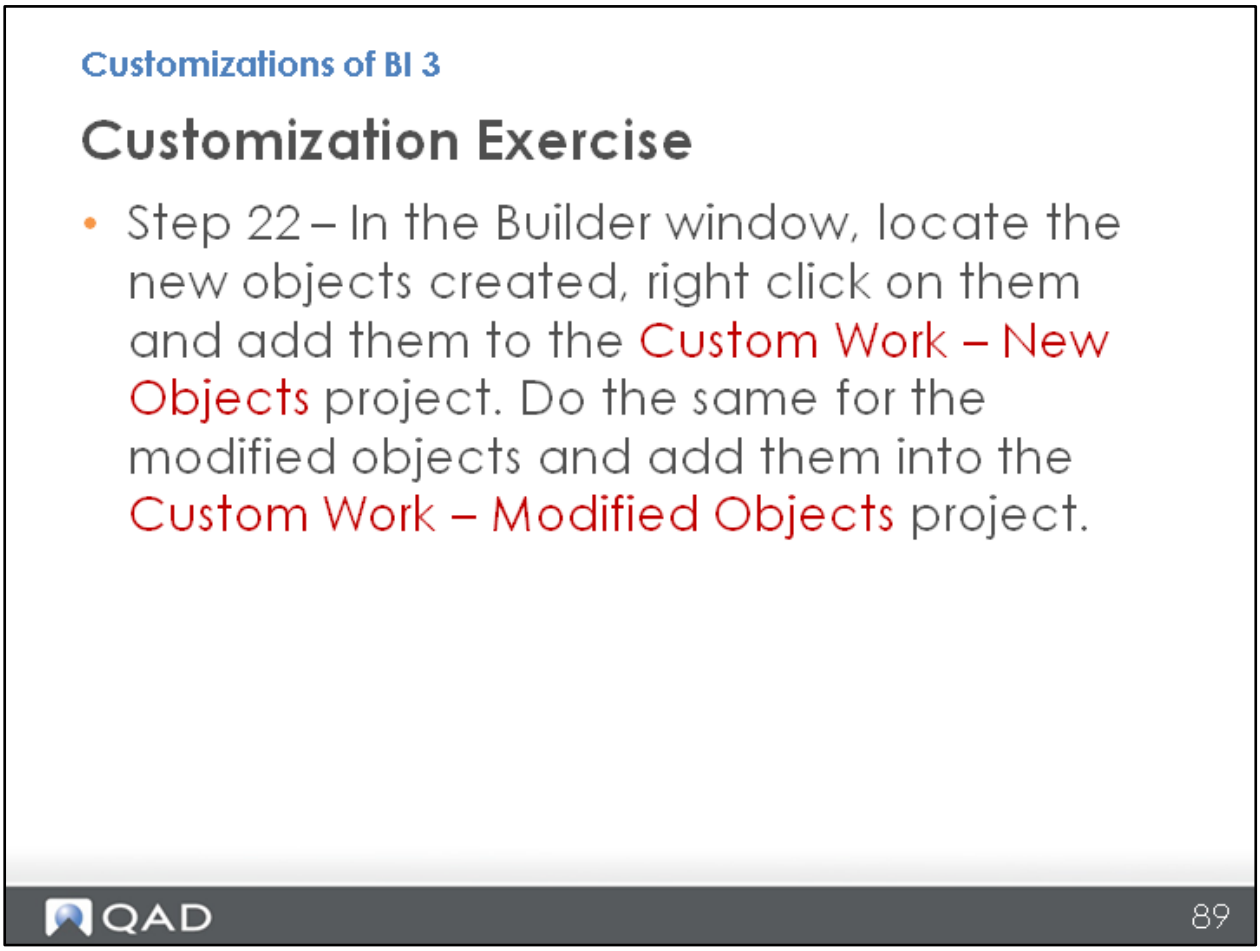
Customization Exercise



Customization Exercise



Customization Exercise

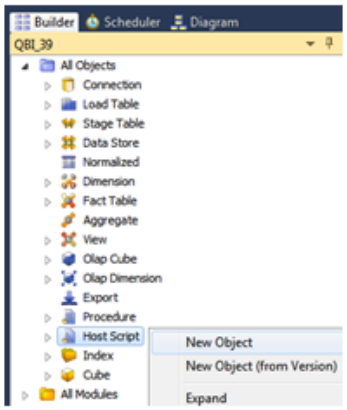


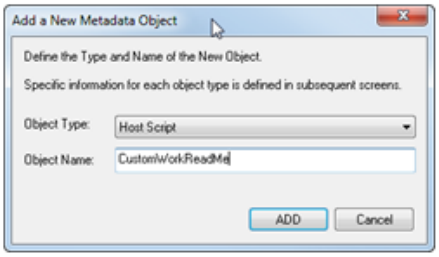
Customization Exercise

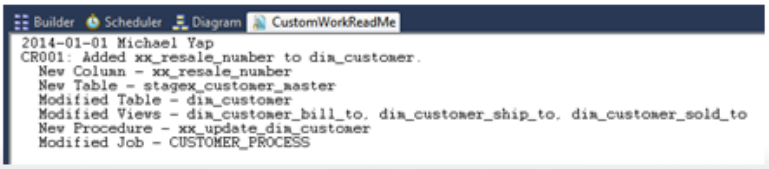
Customizations of BI 3

Customization Exercise

- Step 23 – Create a Host Script named CustomWorkReadMe. Document the modifications for reference. Add it to both Custom Work Projects.

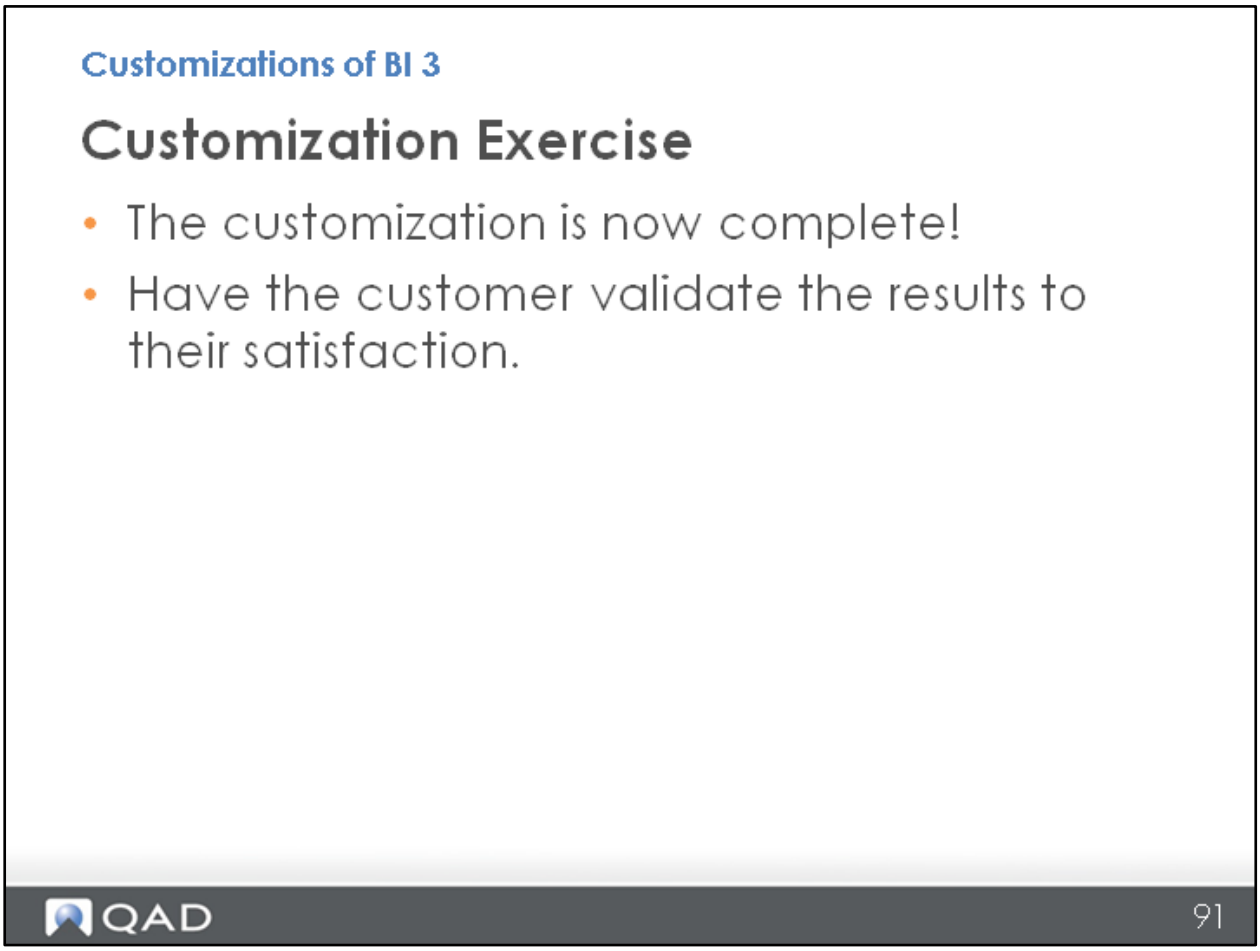




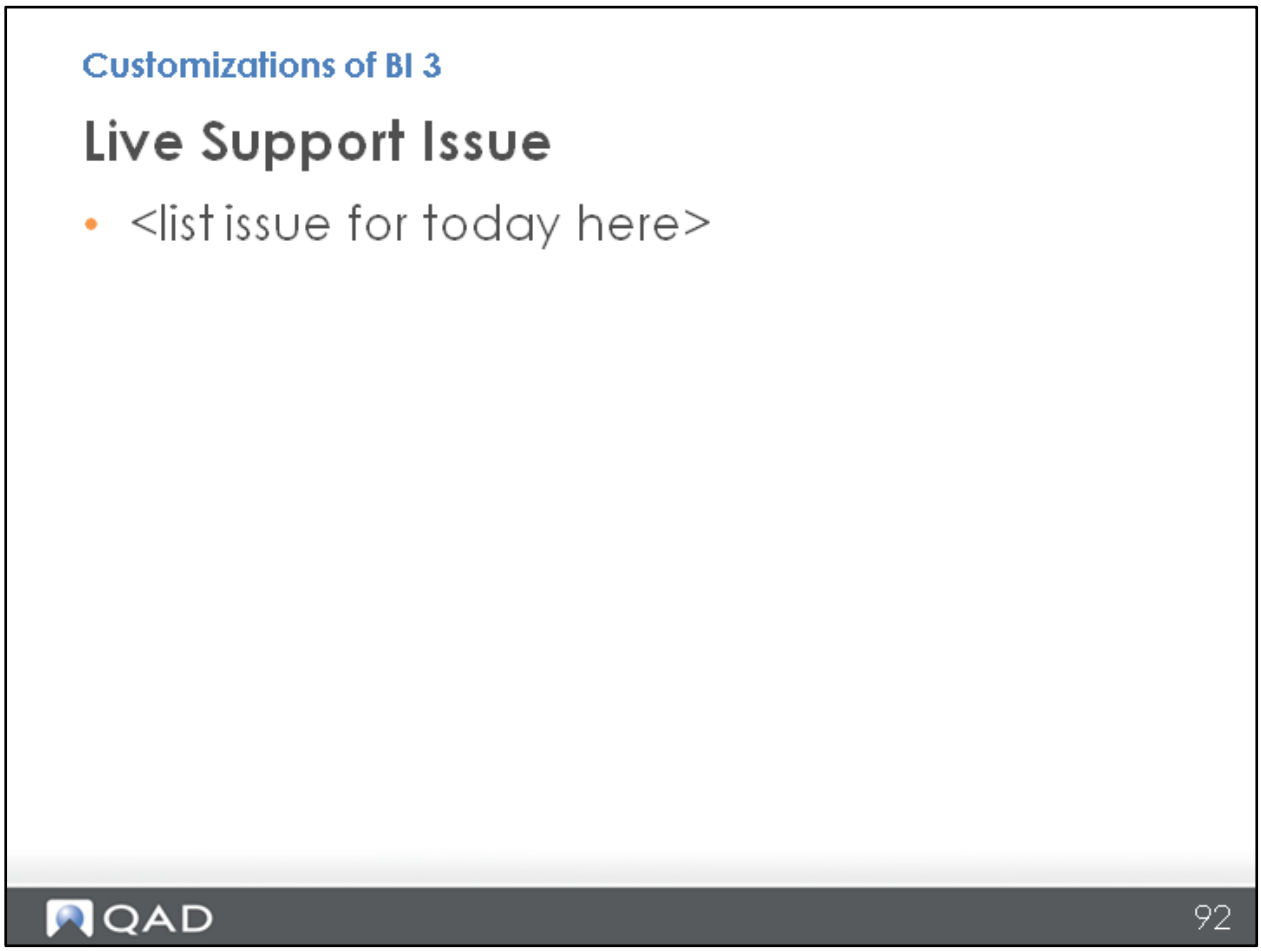


QAD 90

Customization Exercise



Live Support Issue



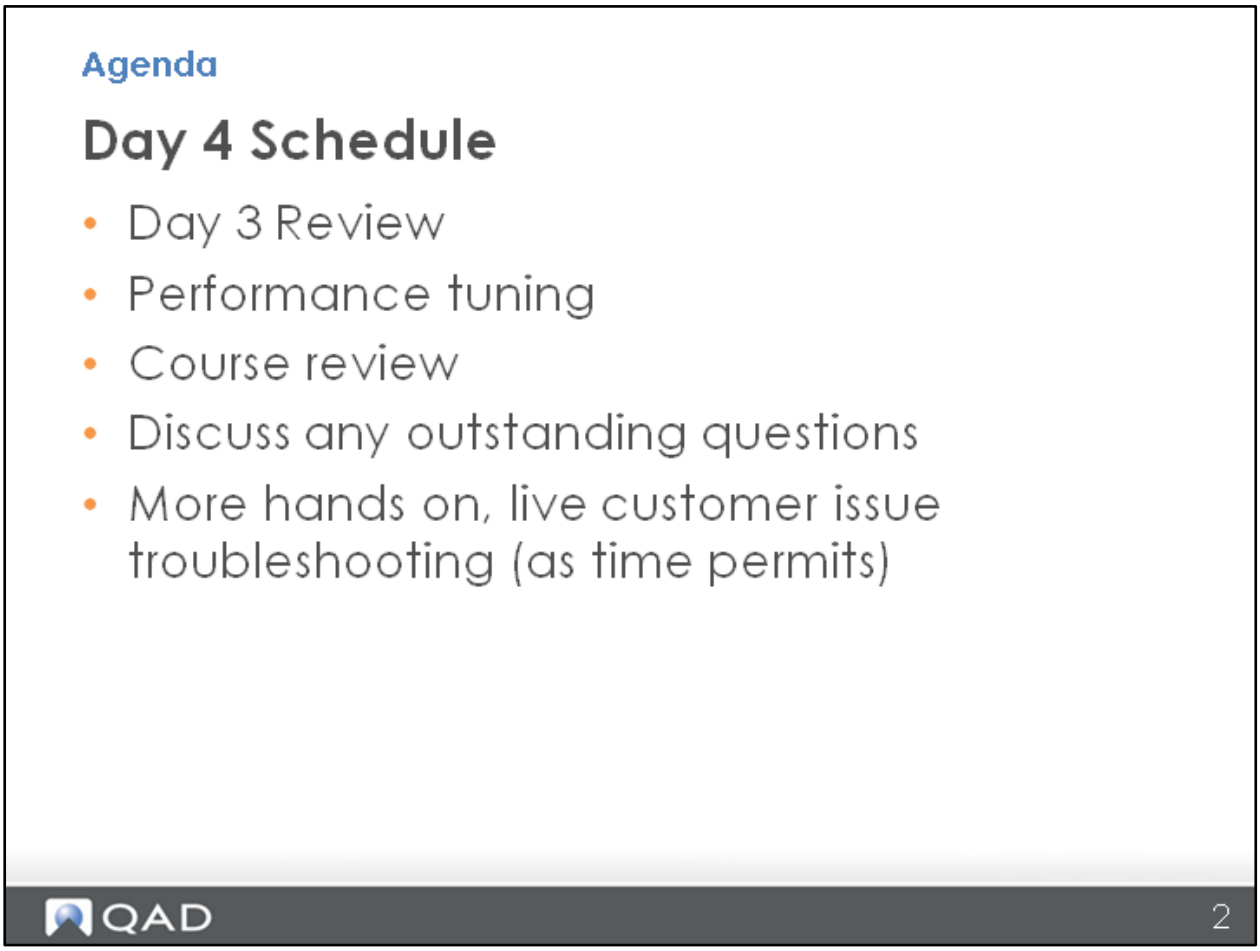
CHAPTER 4

QAD BI v3 - Tech Level 2 Certification - Part 4

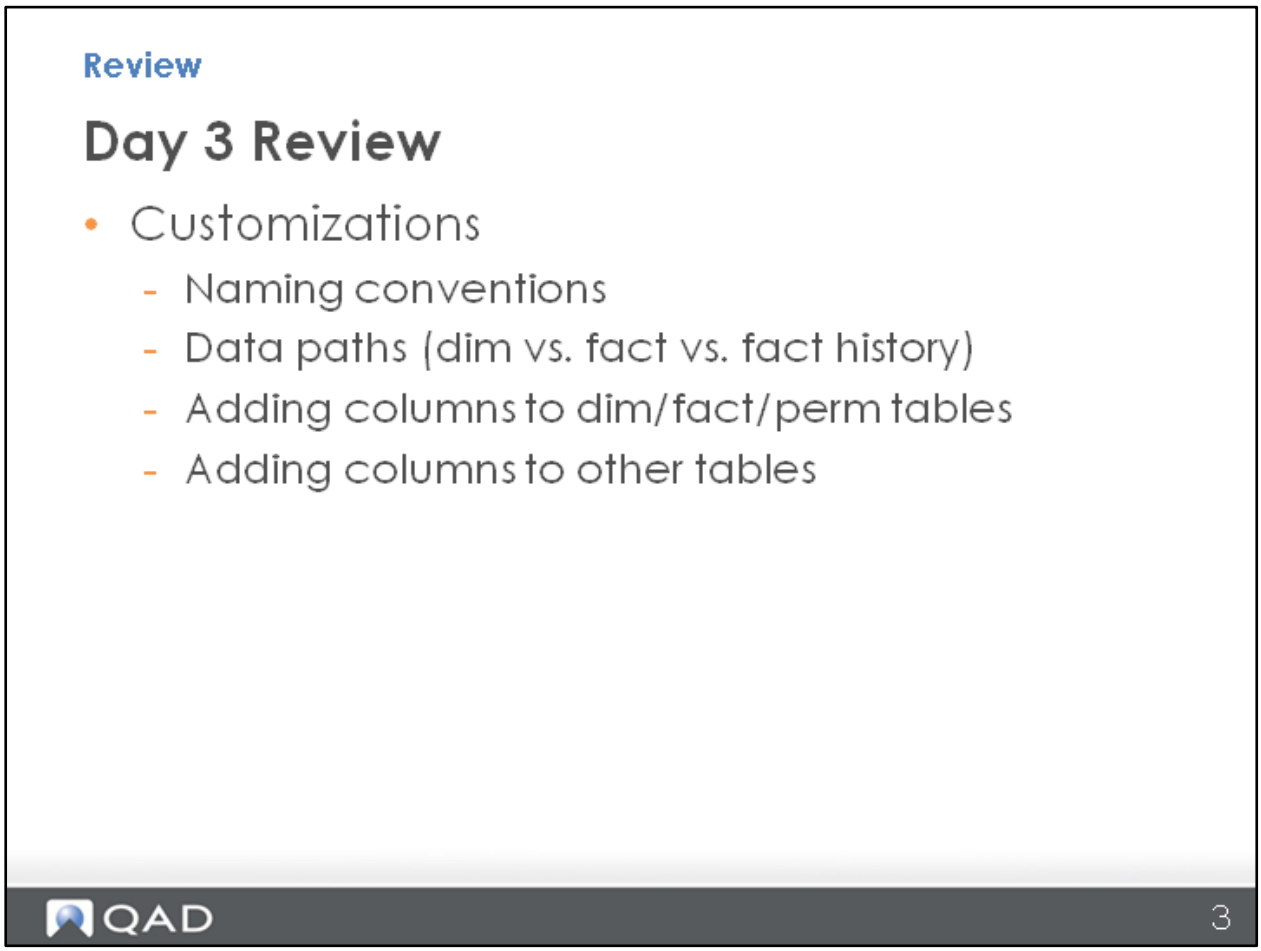
QAD BI v3 – Technical Level II Certification Class – Part 4



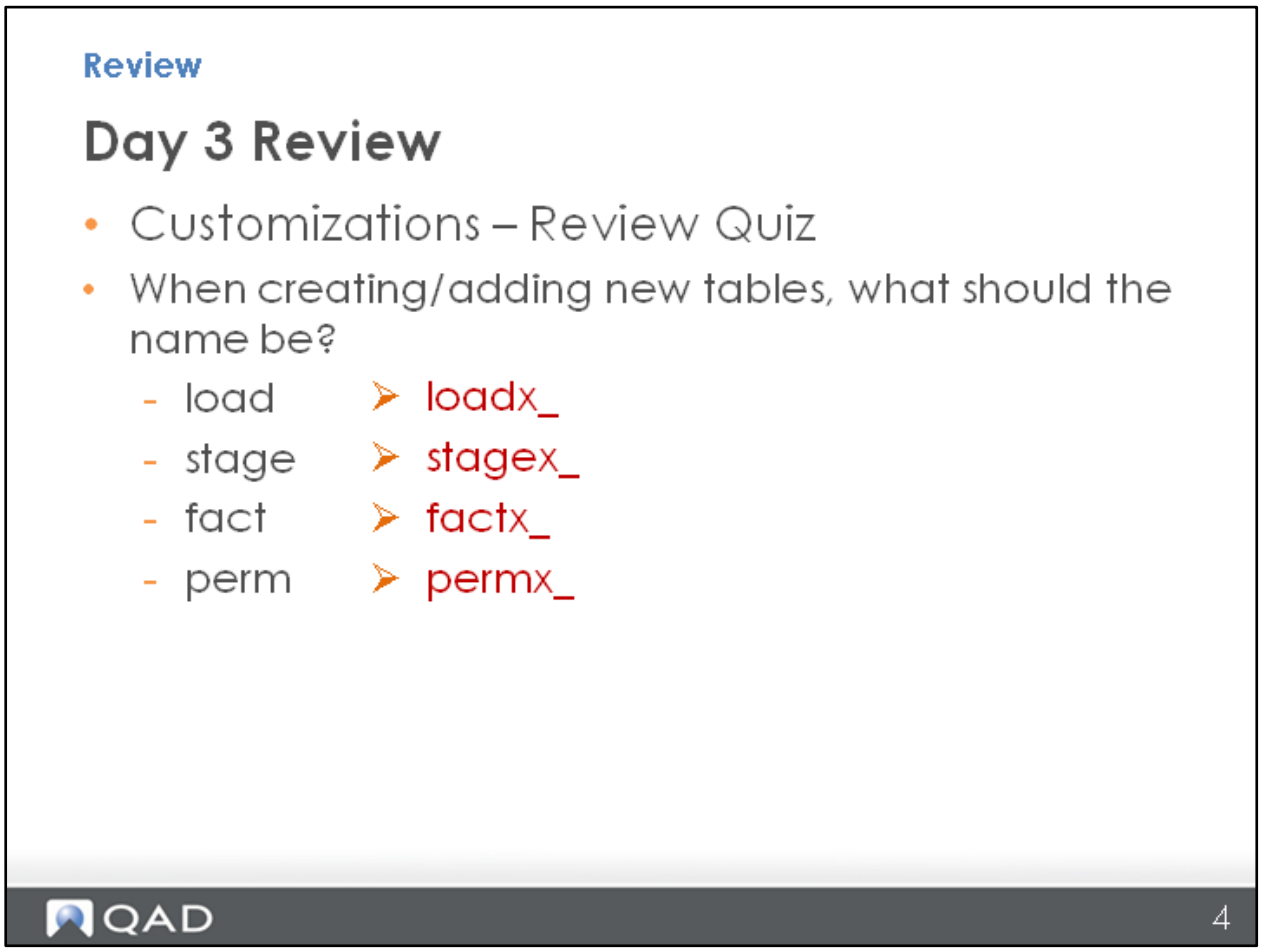
Day 4 Schedule



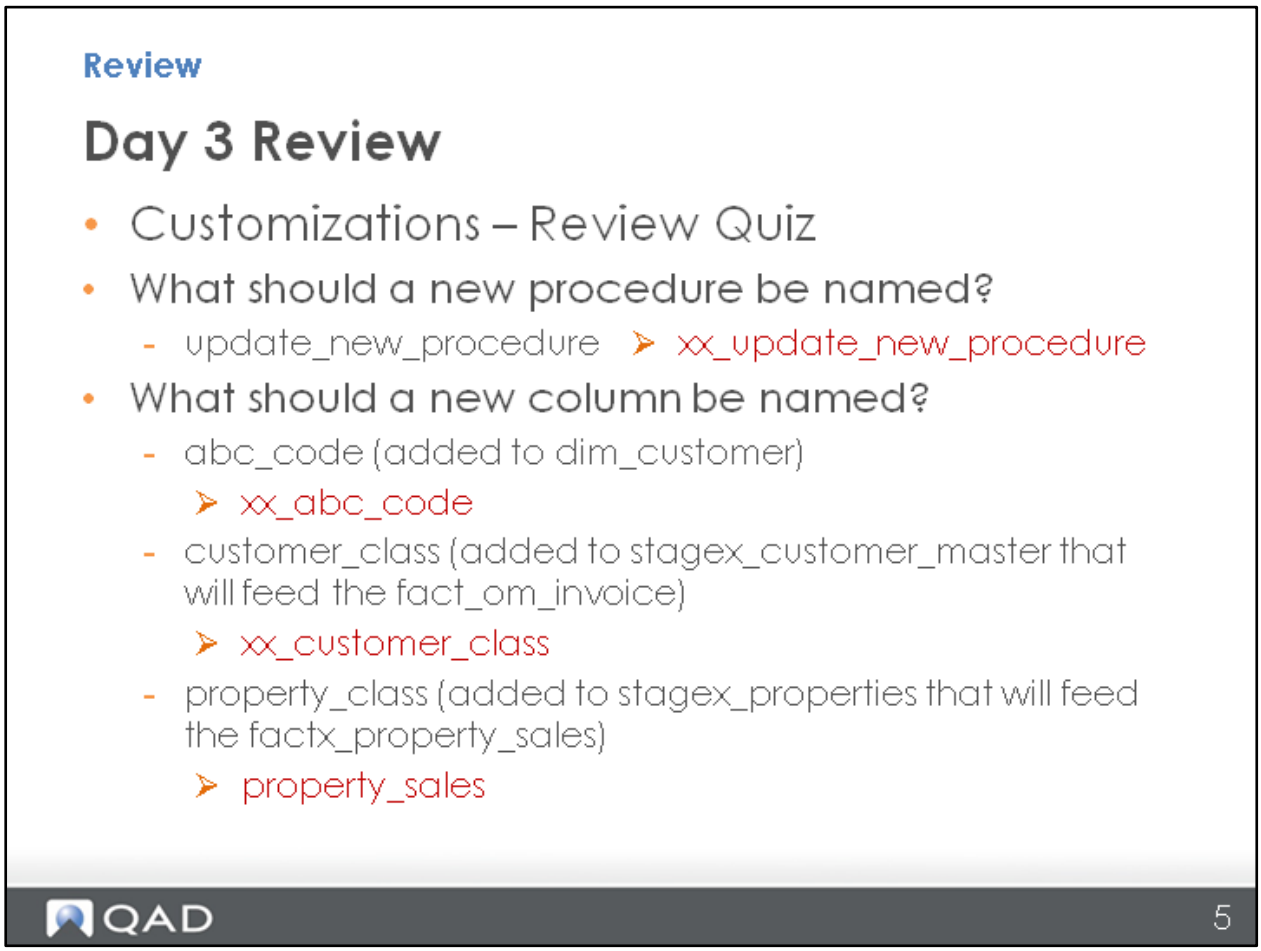
Day 3 Review


Review
Day 3 Review
Customizations
Naming conventions
Data paths (dim vs. fact vs. fact history)
Adding columns to dim/fact/perm tables
Adding columns to other tables
QAD
3

Day 3 Review


Review
Day 3 Review
Customizations – Review Quiz
When creating/adding new tables, what should the name be?
load ➢ loadx_
stage ➢ stagex_
fact ➢ factx_
perm ➢ permx_
QAD
4

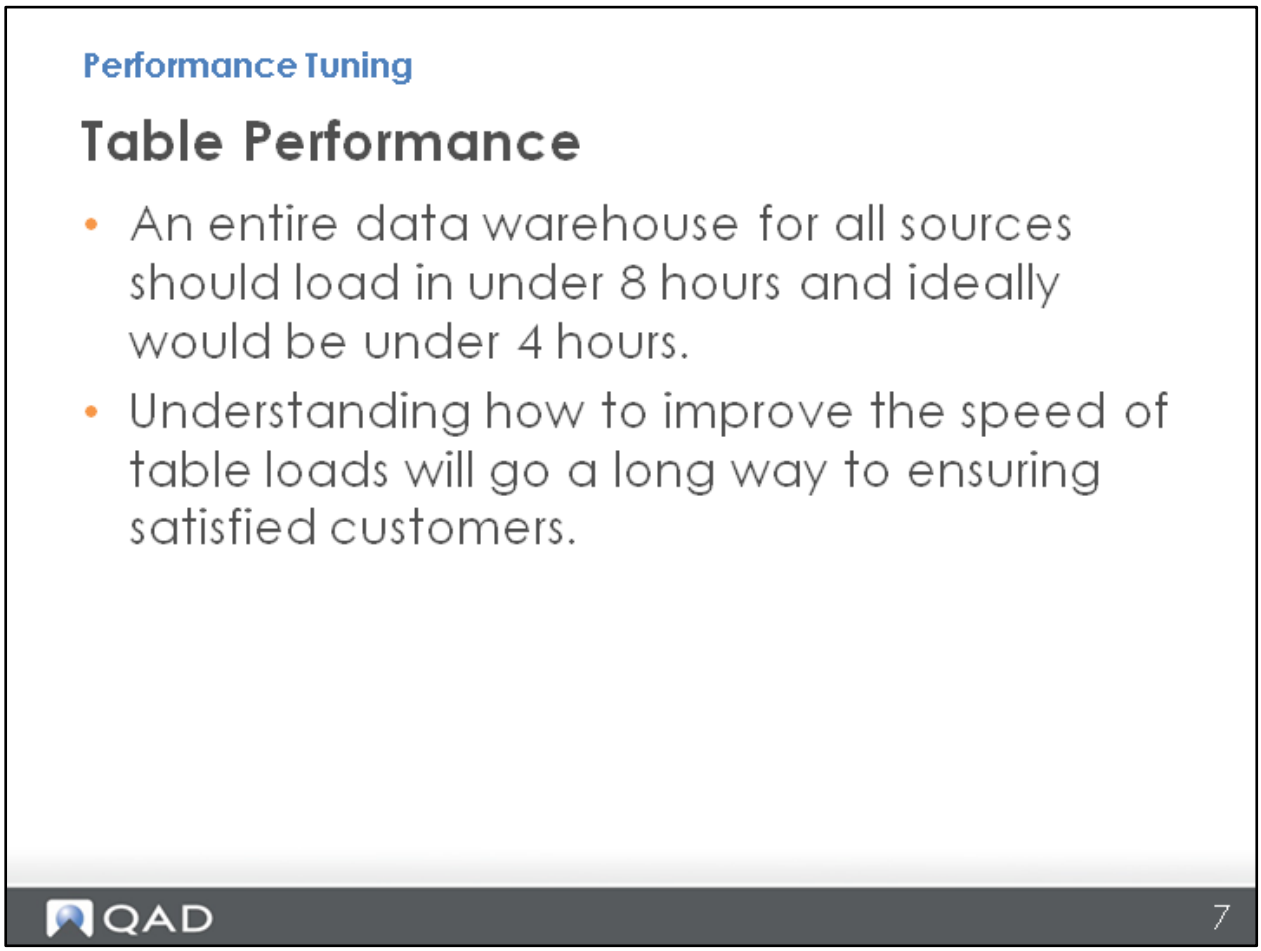
Day 3 Review


Review
Day 3 Review
Customizations – Review Quiz
What should a new procedure be named?
update_new_procedure ➢ xx_update_new_procedure
What should a new column be named?
abc_code (added to dim_customer)
➢ xx_abc_code
customer_class (added to stagex_customer_master that will feed the fact_om_invoice)
➢ xx_customer_class
property_class (added to stagex_properties that will feed the factx_property_sales)
➢ property_sales
QAD
5

Performance Tuning



Table Performance



It is critical for a data warehouse to load quickly. As a rule of thumb a data warehouse should take less than 8 hours to load.

Table Performance

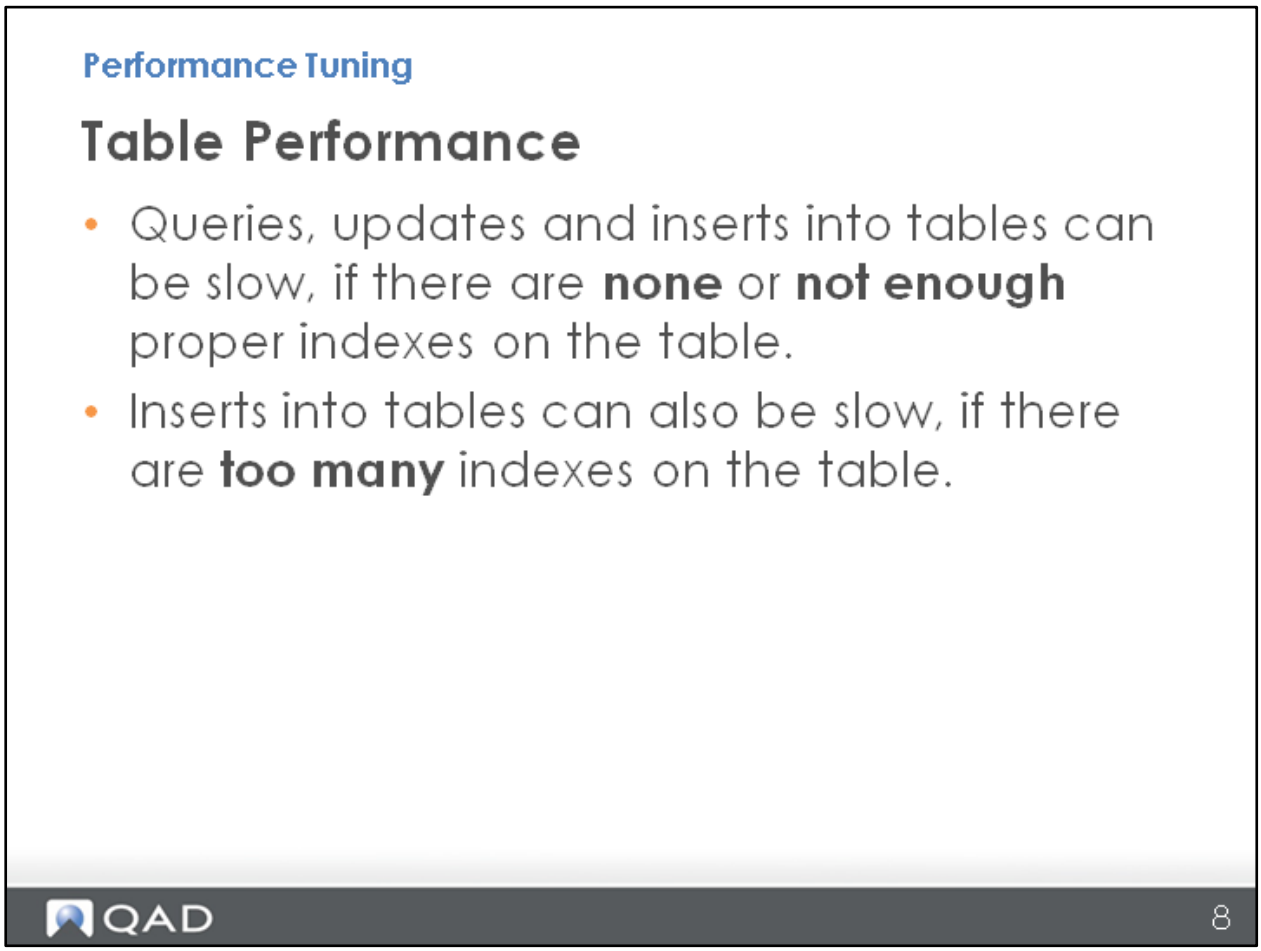
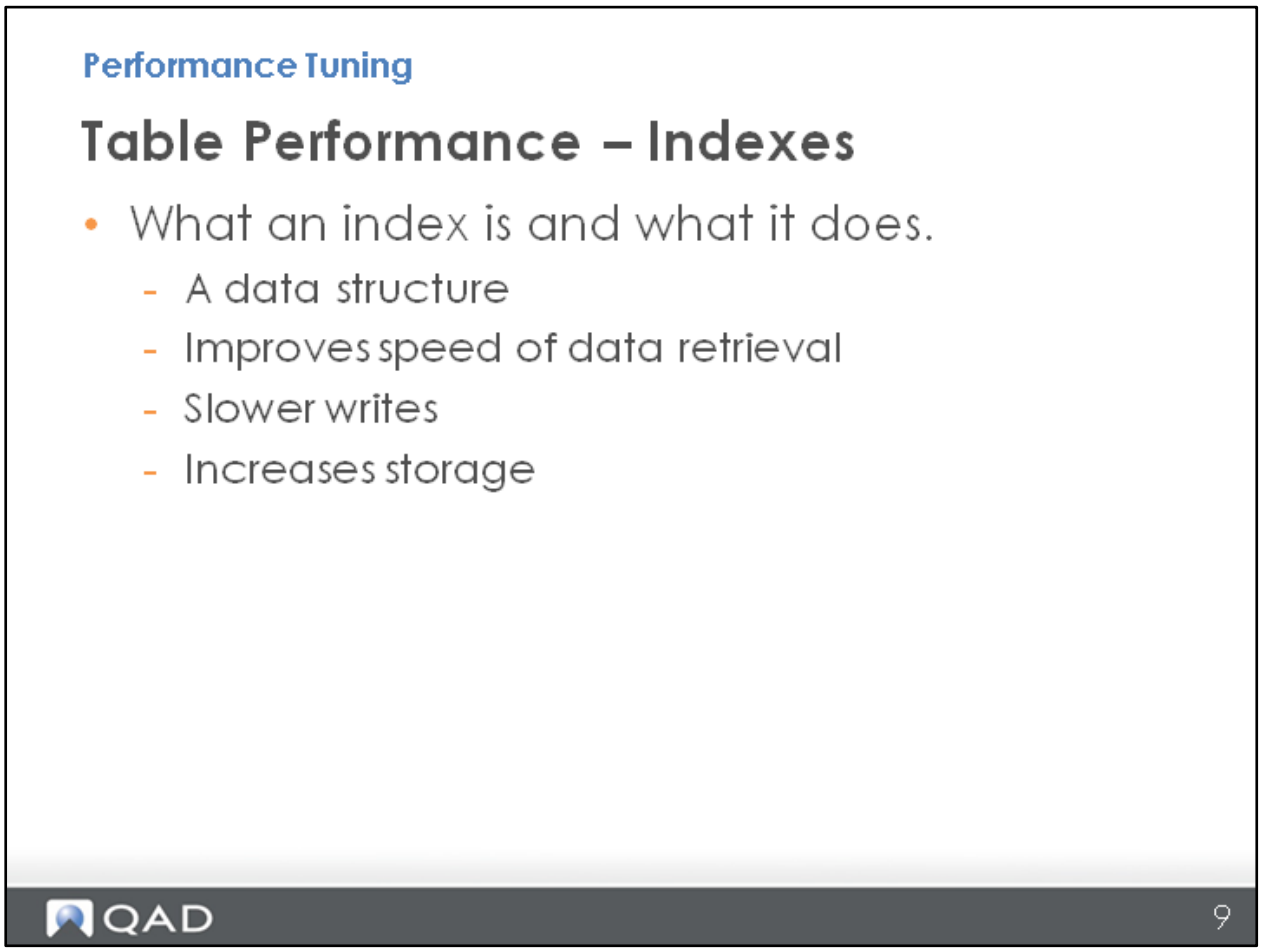


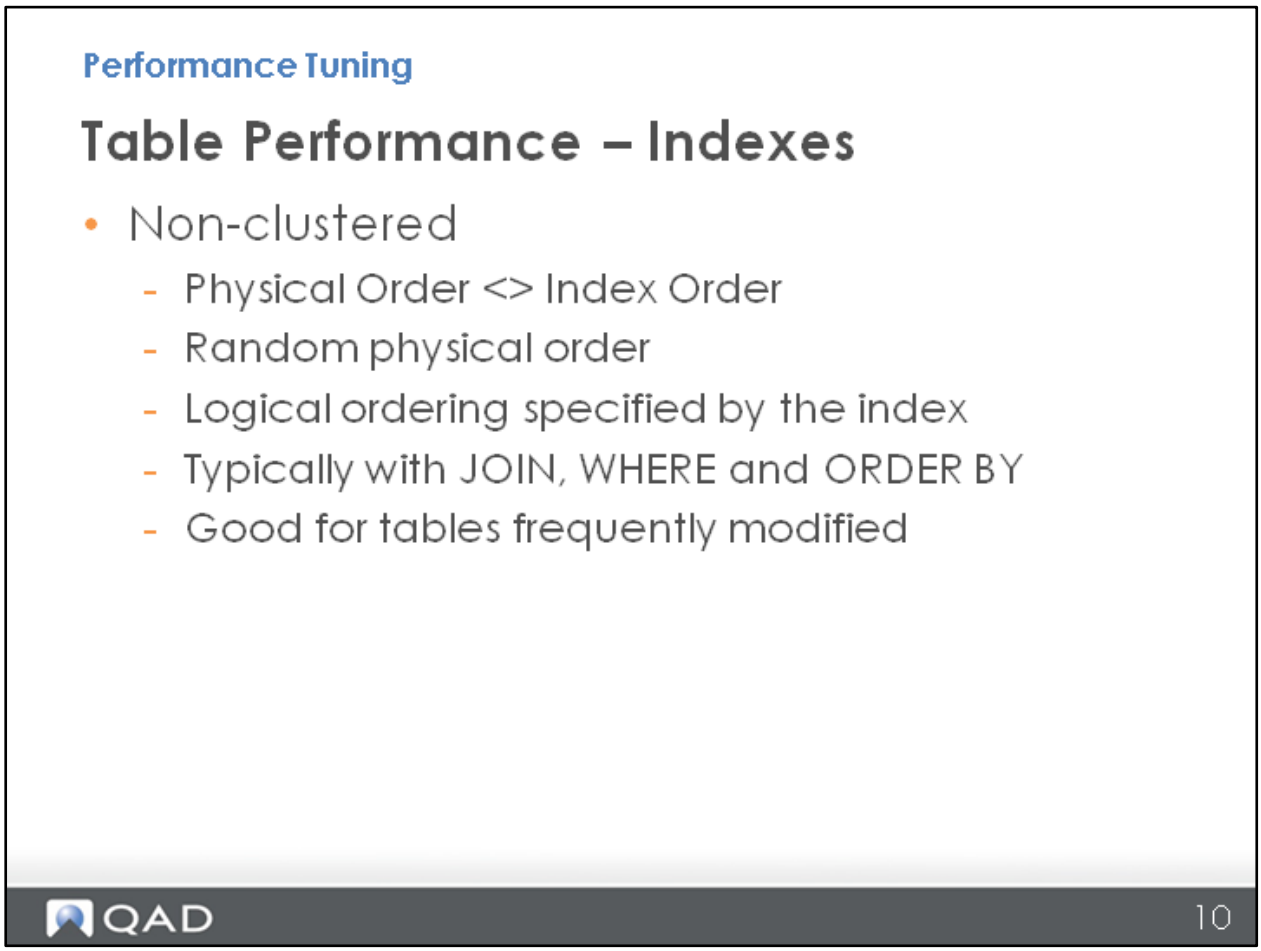
Table Performance – Indexes



A **database index** is a data structure that improves the speed of data retrieval operations on a database table at the cost of slower writes and increased storage space. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records. The disk space required to store the index is typically less than that required by the table (since indices usually contain only the key-fields according to which the table is to be arranged, and exclude all the other details in the table), yielding the possibility to store indices in memory for a table whose data is too large to store in memory.

Wikipedia - http://en.wikipedia.org/wiki/Index_(database)

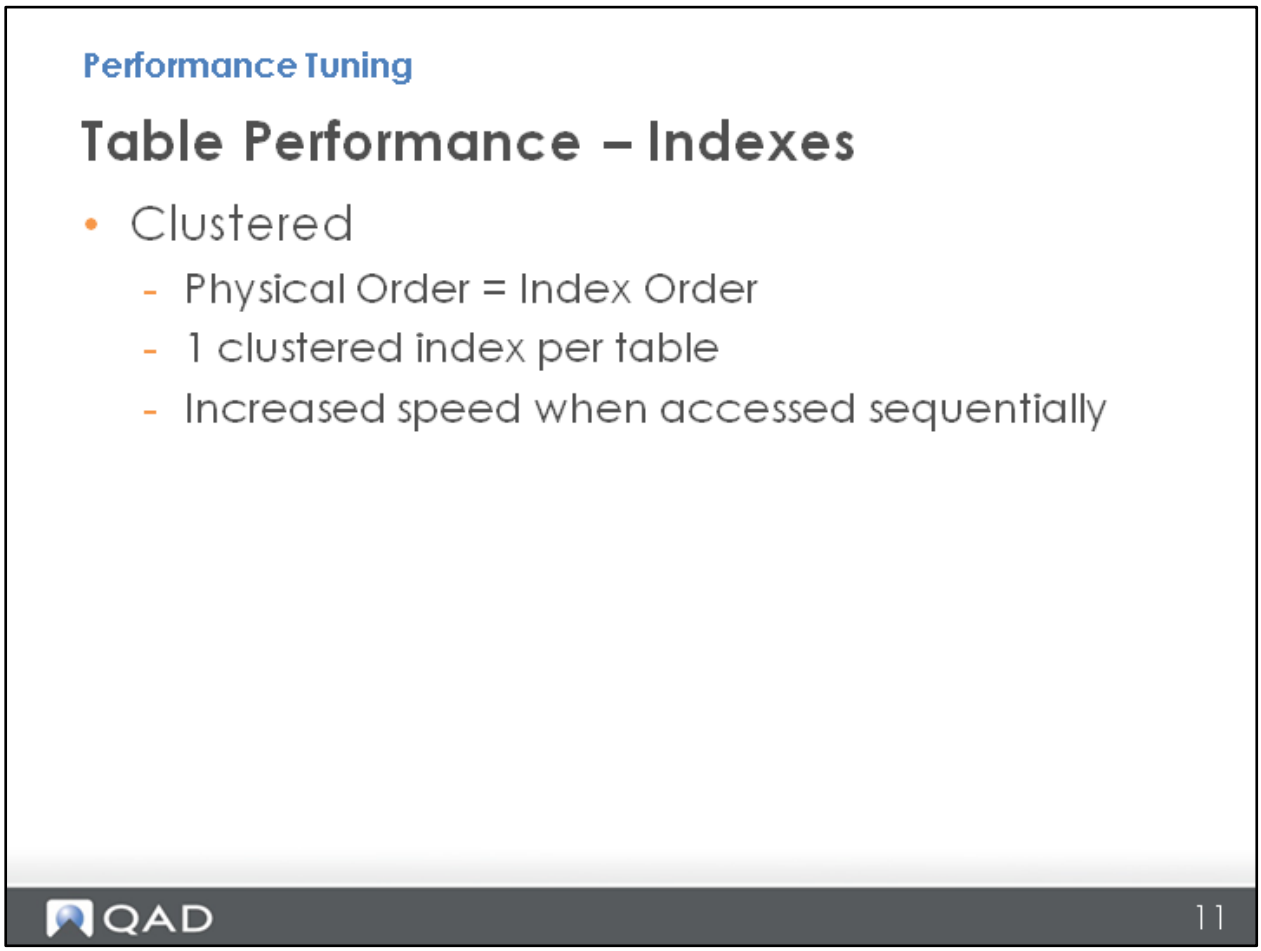
Table Performance – Indexes



The data is present in random order, but the **logical ordering** is specified by the index. The data rows may be randomly spread throughout the table. The non-clustered index tree contains the index keys in sorted order, with the leaf level of the index containing the pointer to the page and the row number in the data page.

Microsoft SQL Server creates non-clustered indexes by default when CREATE INDEX command is given. There can be more than one non-clustered index on a database table. There can be as many as 249 non-clustered indexes per table in SQL Server 2005 and 999 non-clustered indexes per table in SQL Server 2008. It also creates a clustered index on a primary key by default.

Table Performance – Indexes

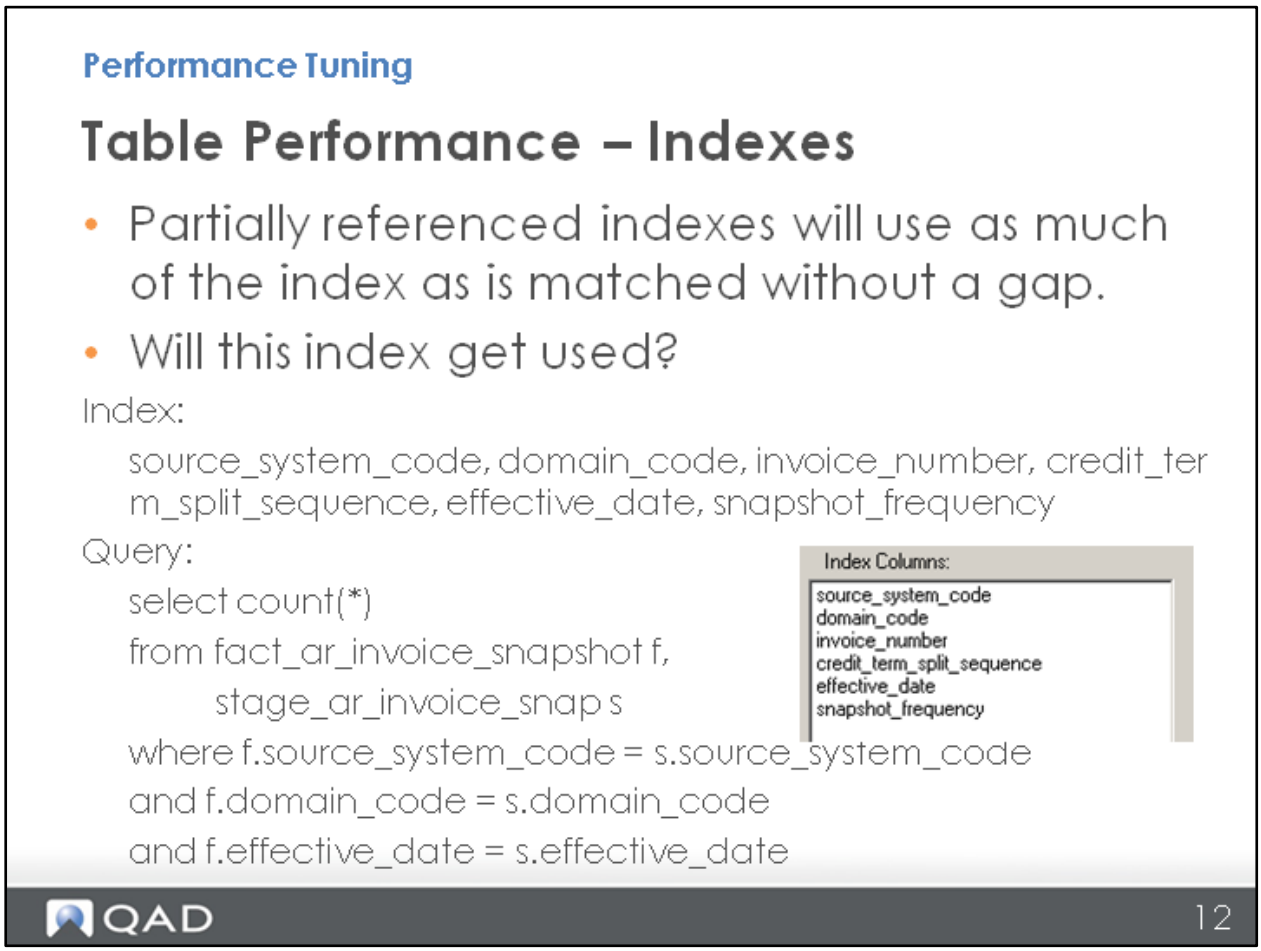


Clustering alters the data block into a certain distinct order to match the index, resulting in the row data being stored in order. Therefore, only one clustered index can be created on a given database table. Clustered indices can greatly increase overall speed of retrieval, but usually only where the data is accessed sequentially in the same or reverse order of the clustered index, or when a range of items is selected.

Since the physical records are in this sort order on disk, the next row item in the sequence is immediately before or after the last one, and so fewer data block reads are required. The primary feature of a clustered index is therefore the ordering of the physical data rows in accordance with the index blocks that point to them. Some databases separate the data and index blocks into separate files, others put two completely different data blocks within the same physical file(s). Create an object where the physical order of rows is same as the index order of the rows and the bottom(leaf) level of clustered index contains the actual data rows.

They are known as “index organized tables” under Oracle database.

Table Performance – Indexes



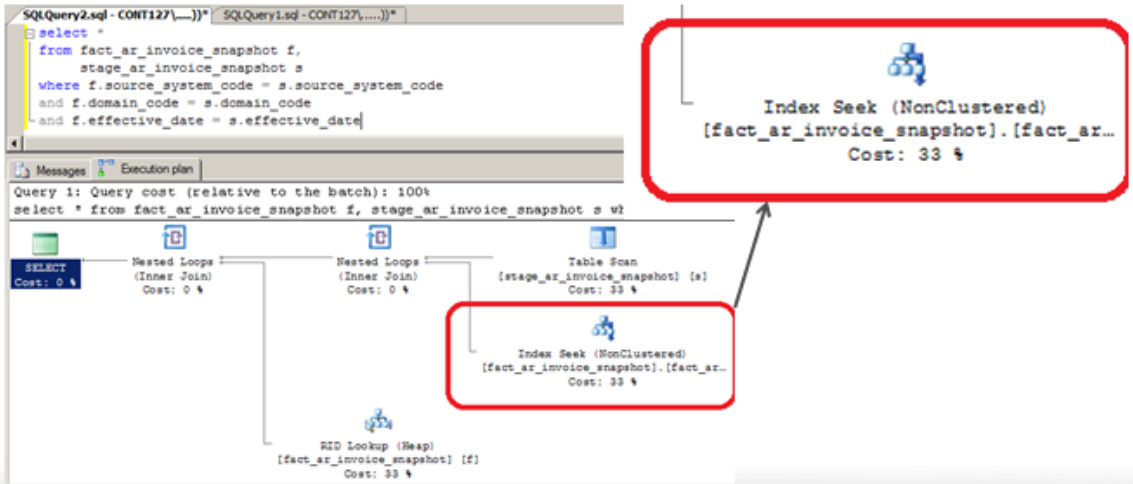
In this example, the query references the first, second and fifth columns in the index. Will the index be used?

Table Performance – Indexes

Performance Tuning

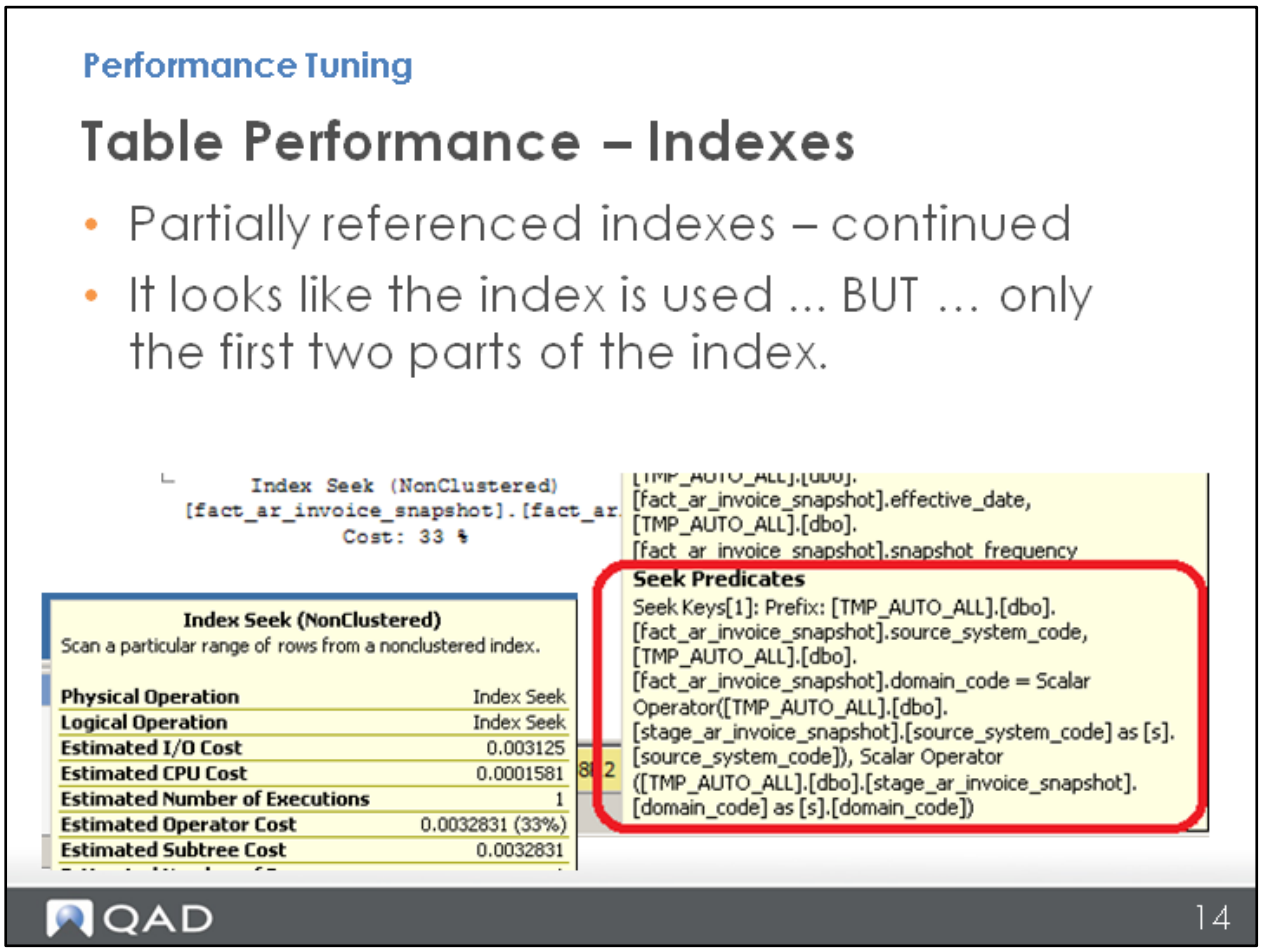
Table Performance – Indexes

- Partially referenced indexes – continued
- It looks like the index is used ... BUT ...



QAD 13

Table Performance – Indexes



If we do an Execution plan on the query, it looks like the Index is working but a closer look is warranted.

Table Performance – Indexes

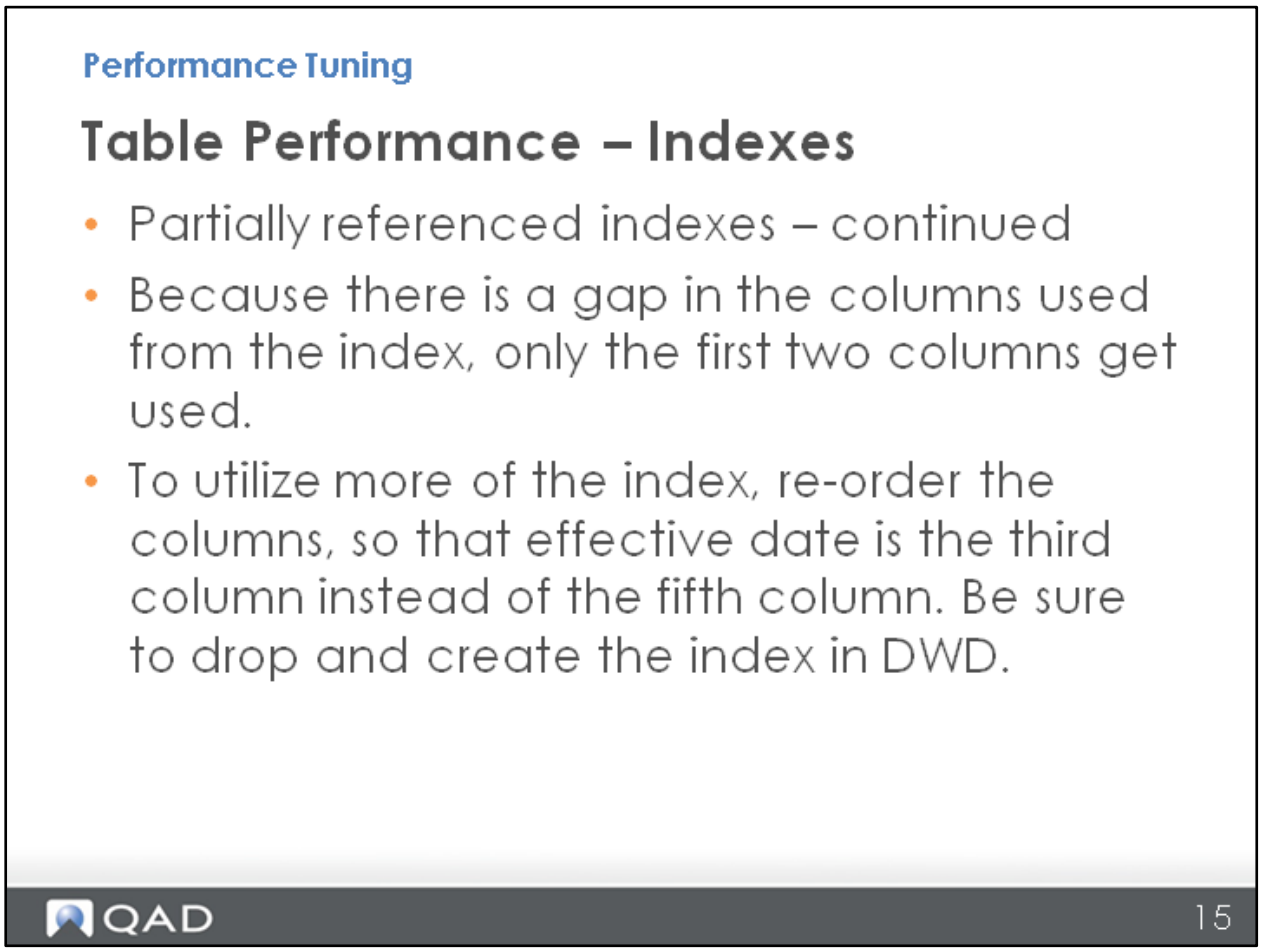
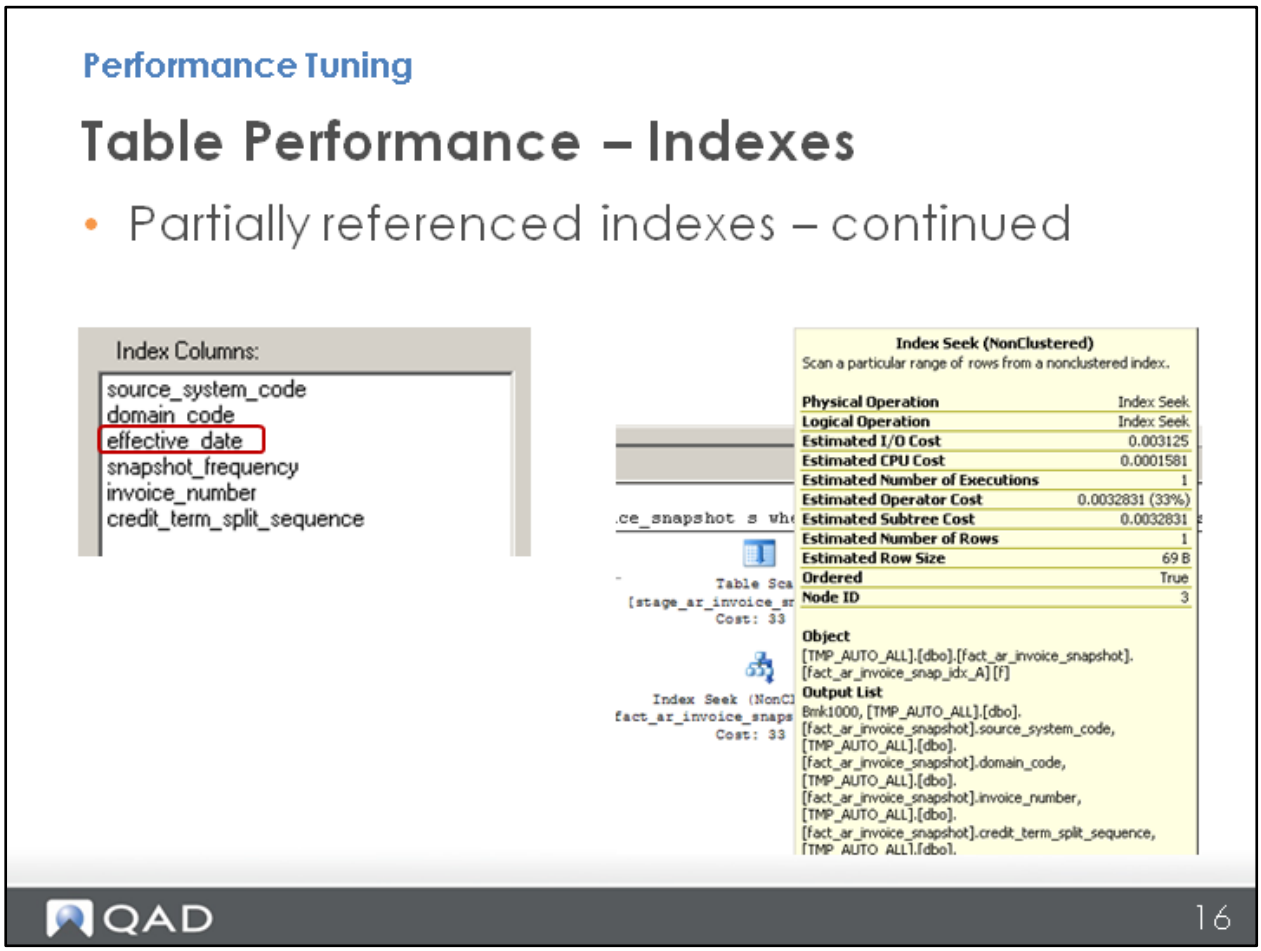
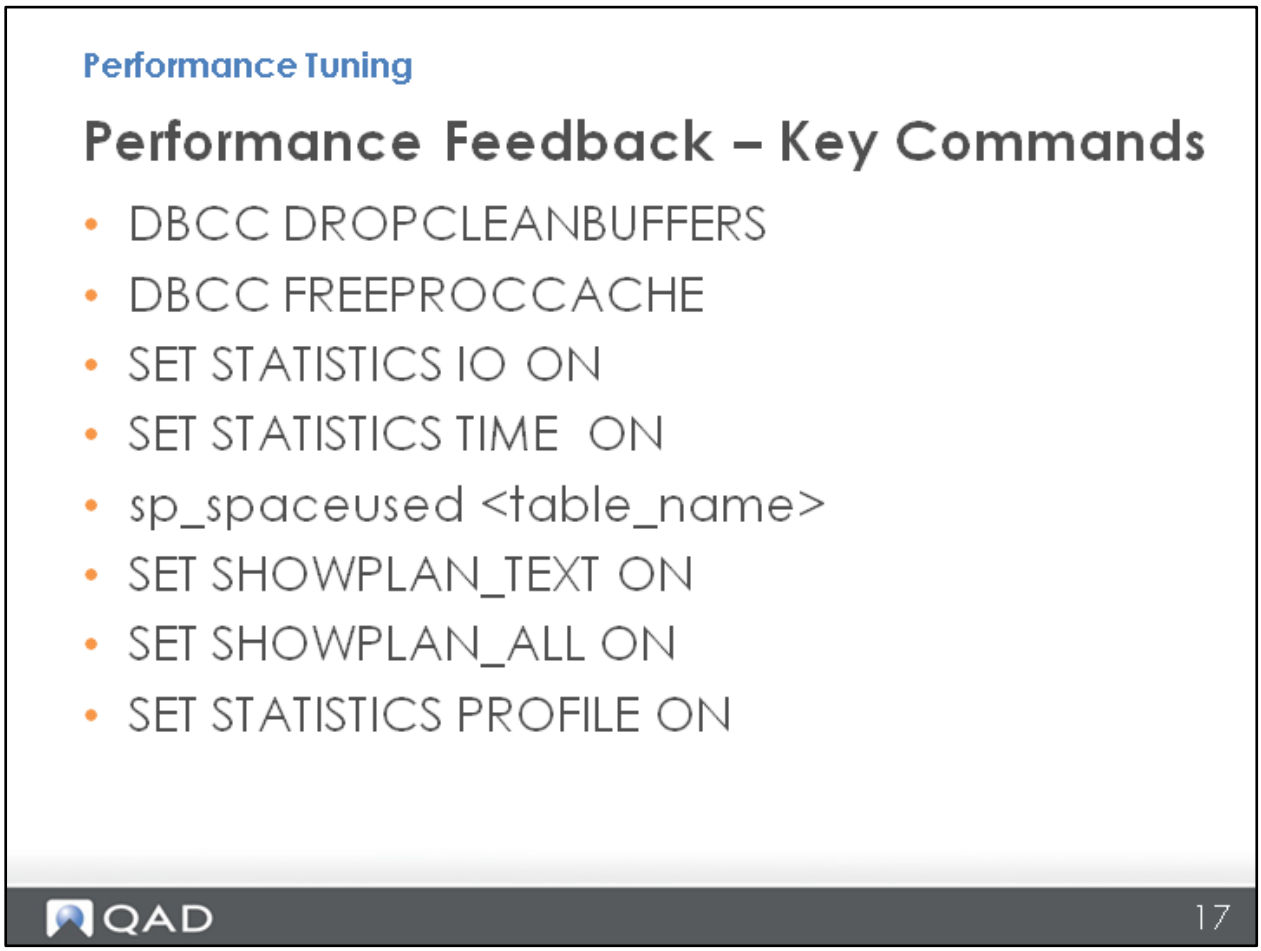


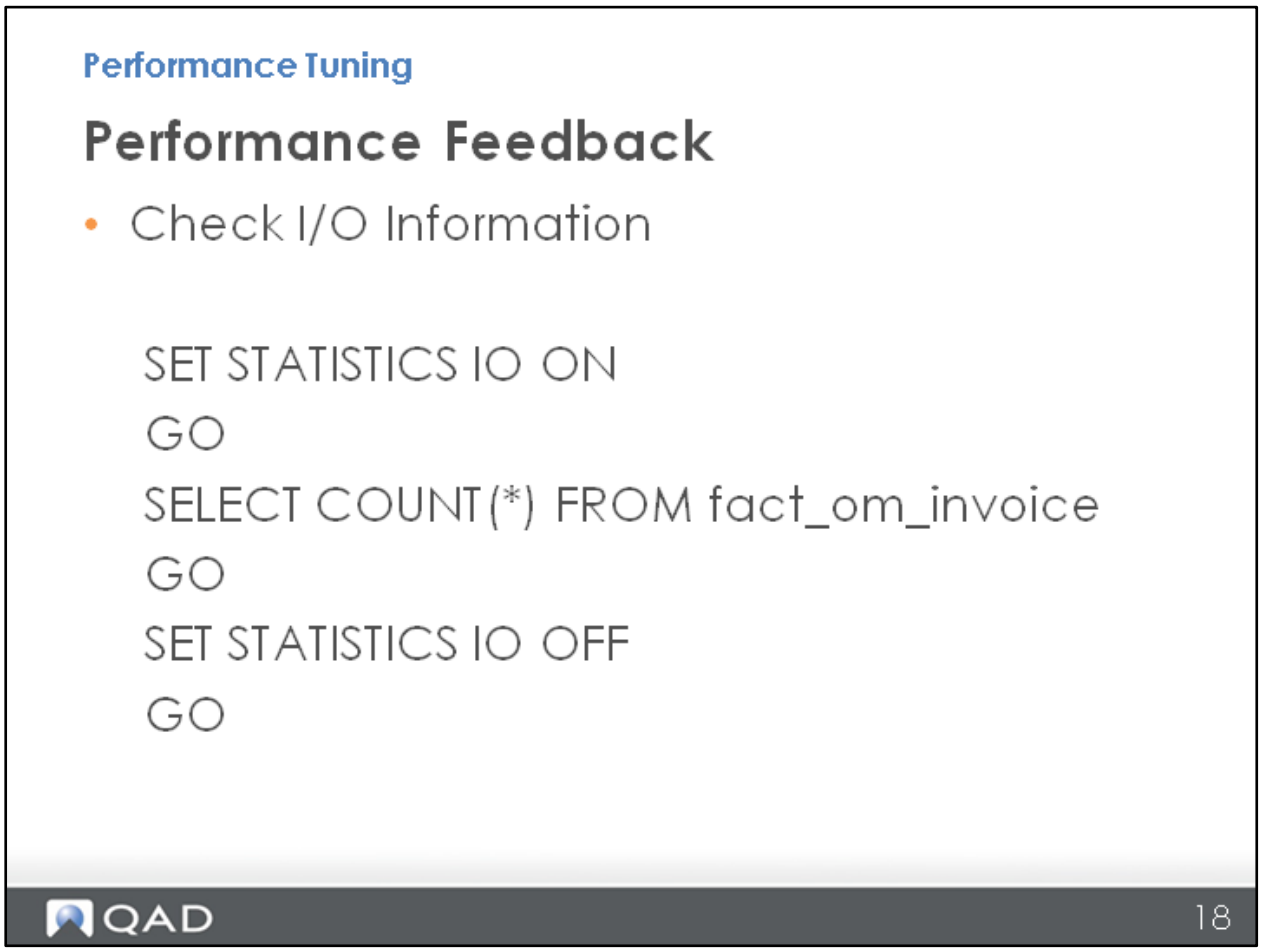
Table Performance – Indexes



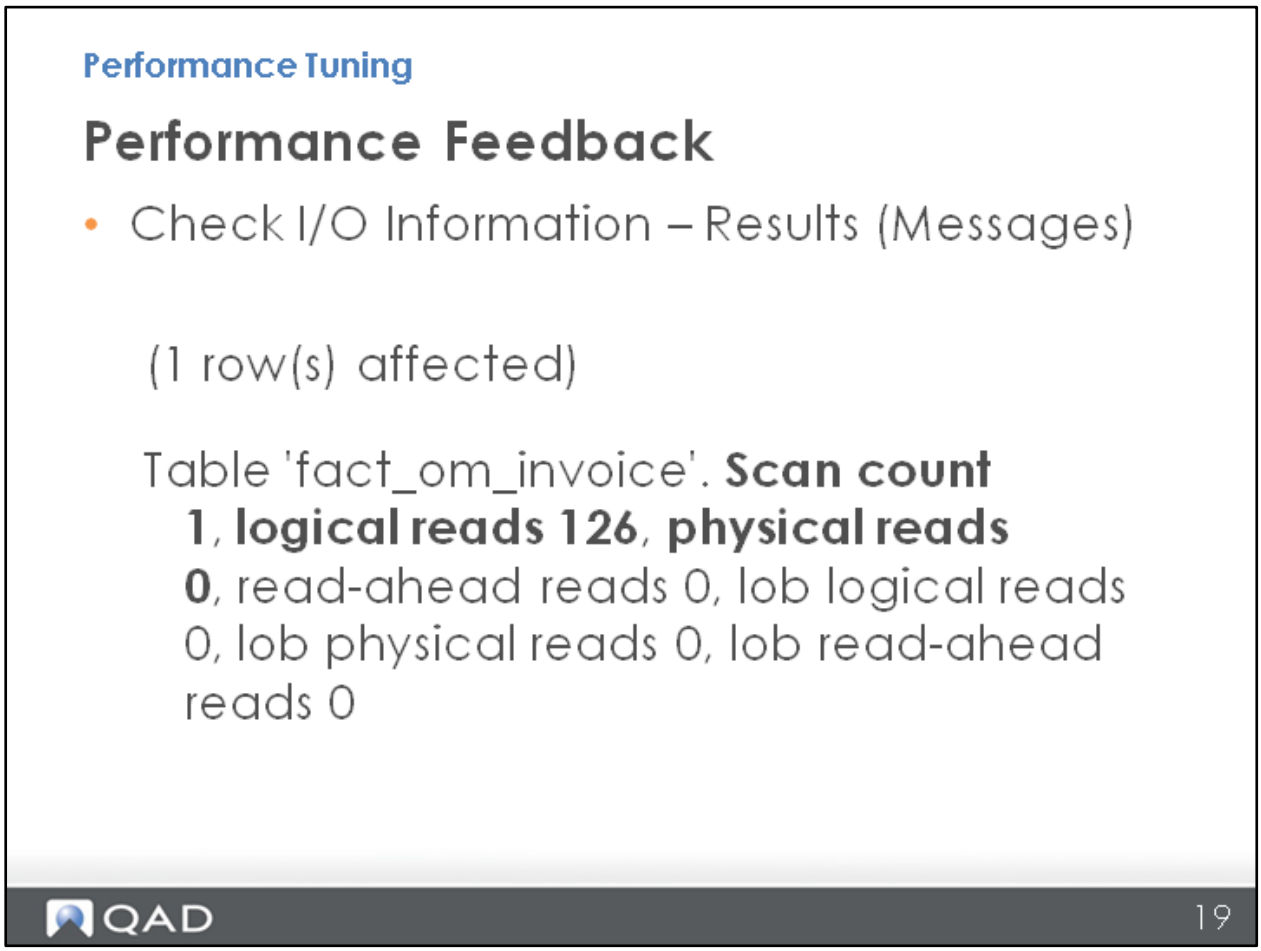
Performance Feedback – Key Commands



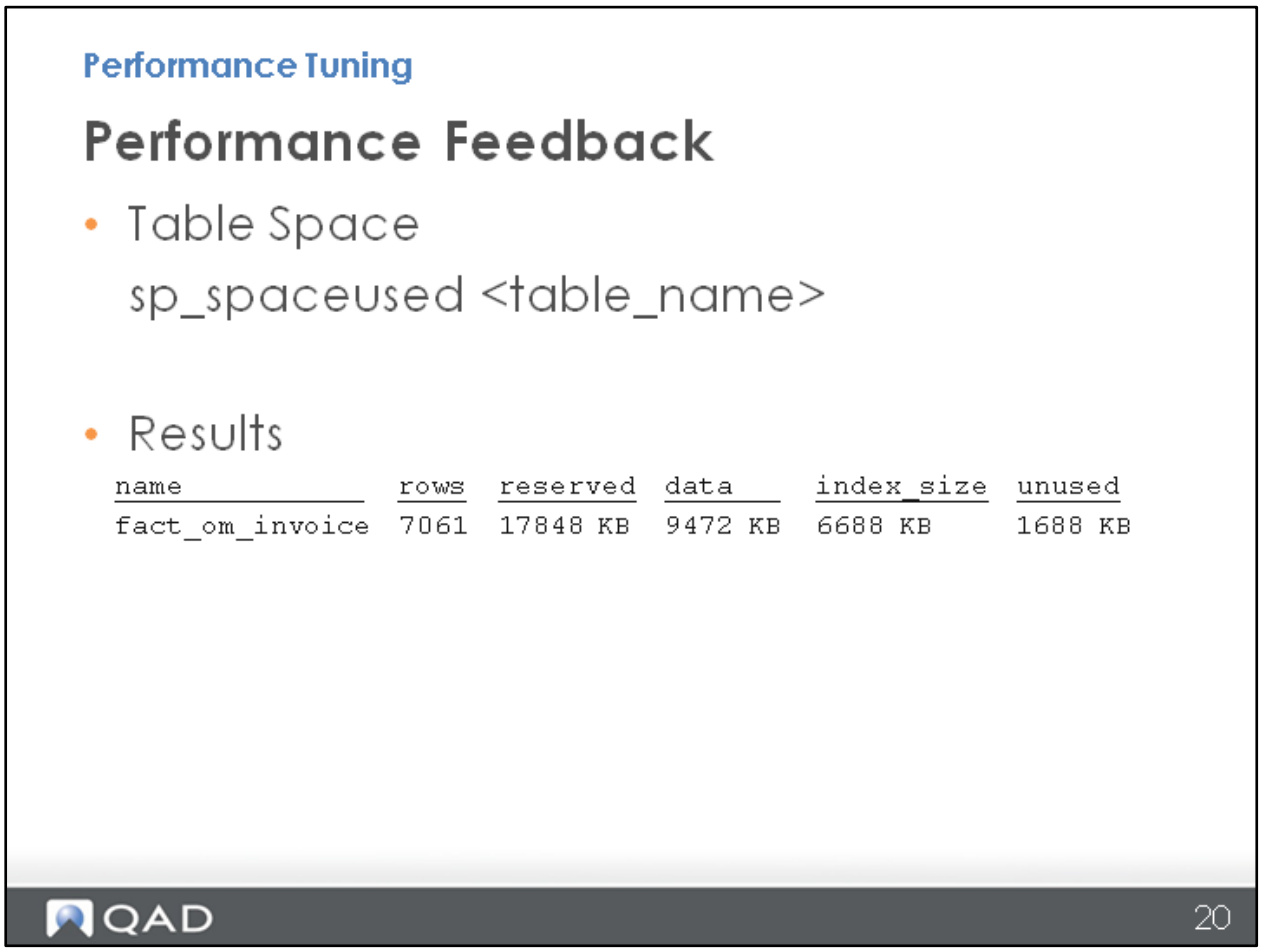
Performance Feedback



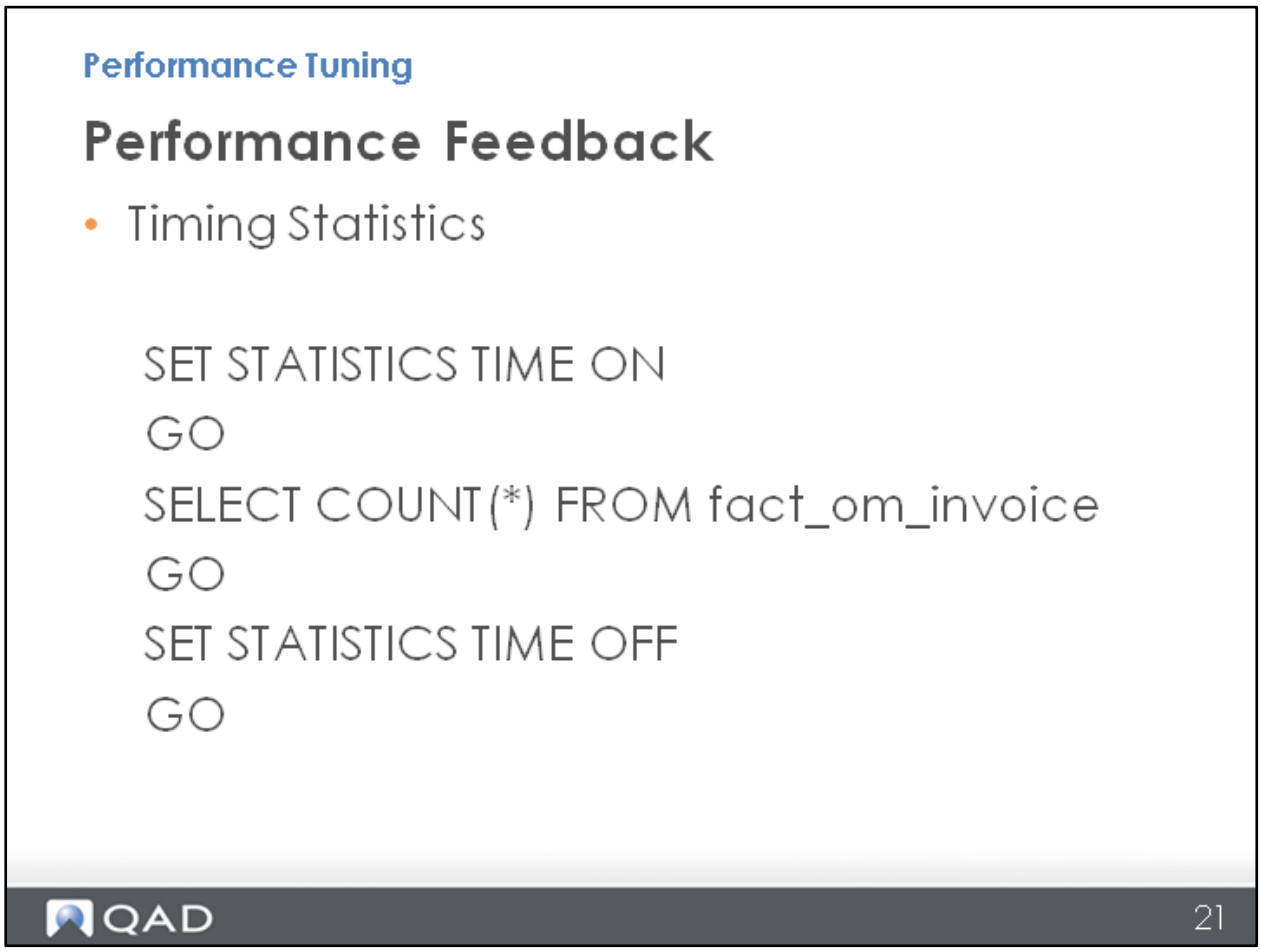
Performance Feedback



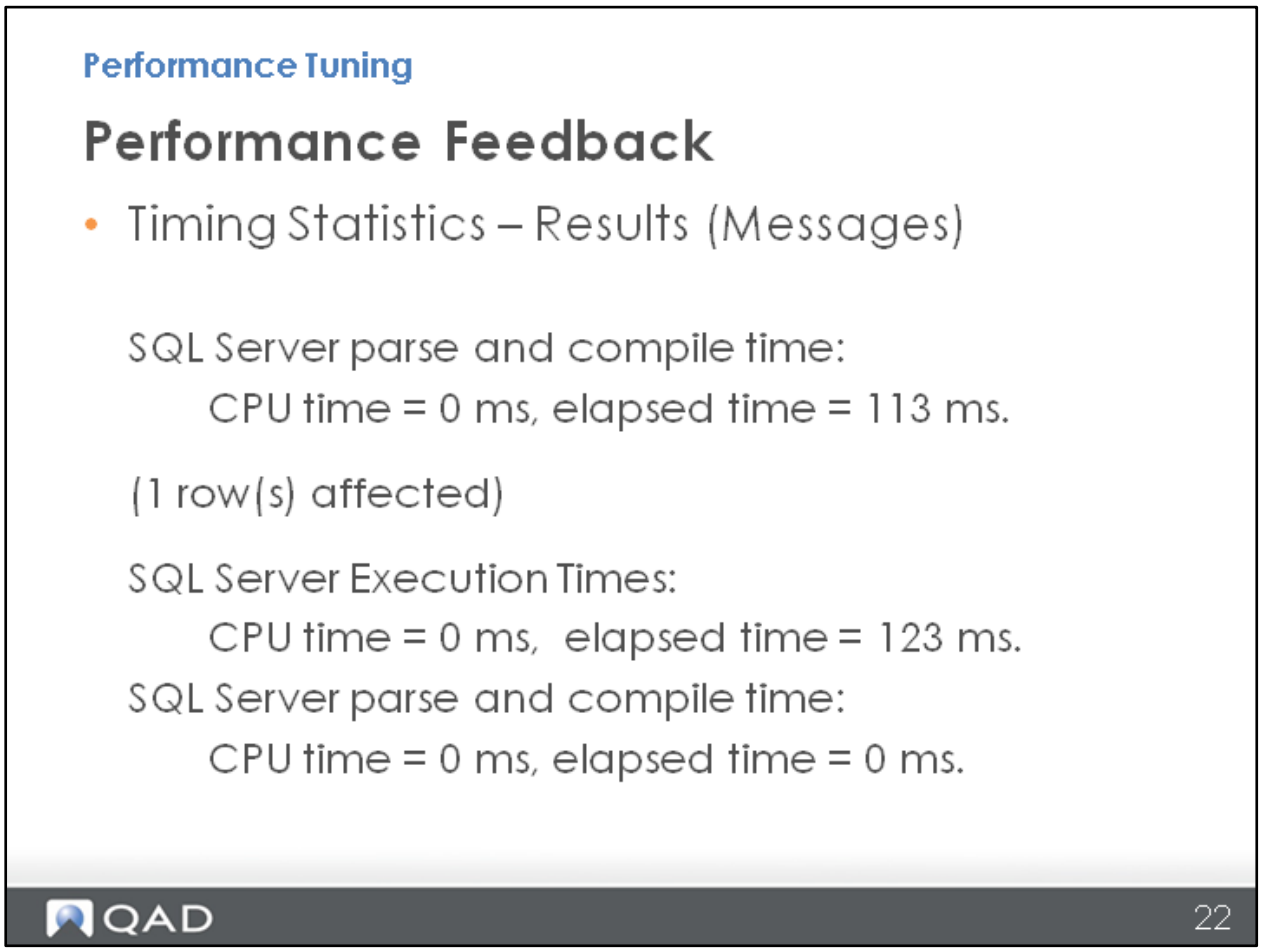
Performance Feedback



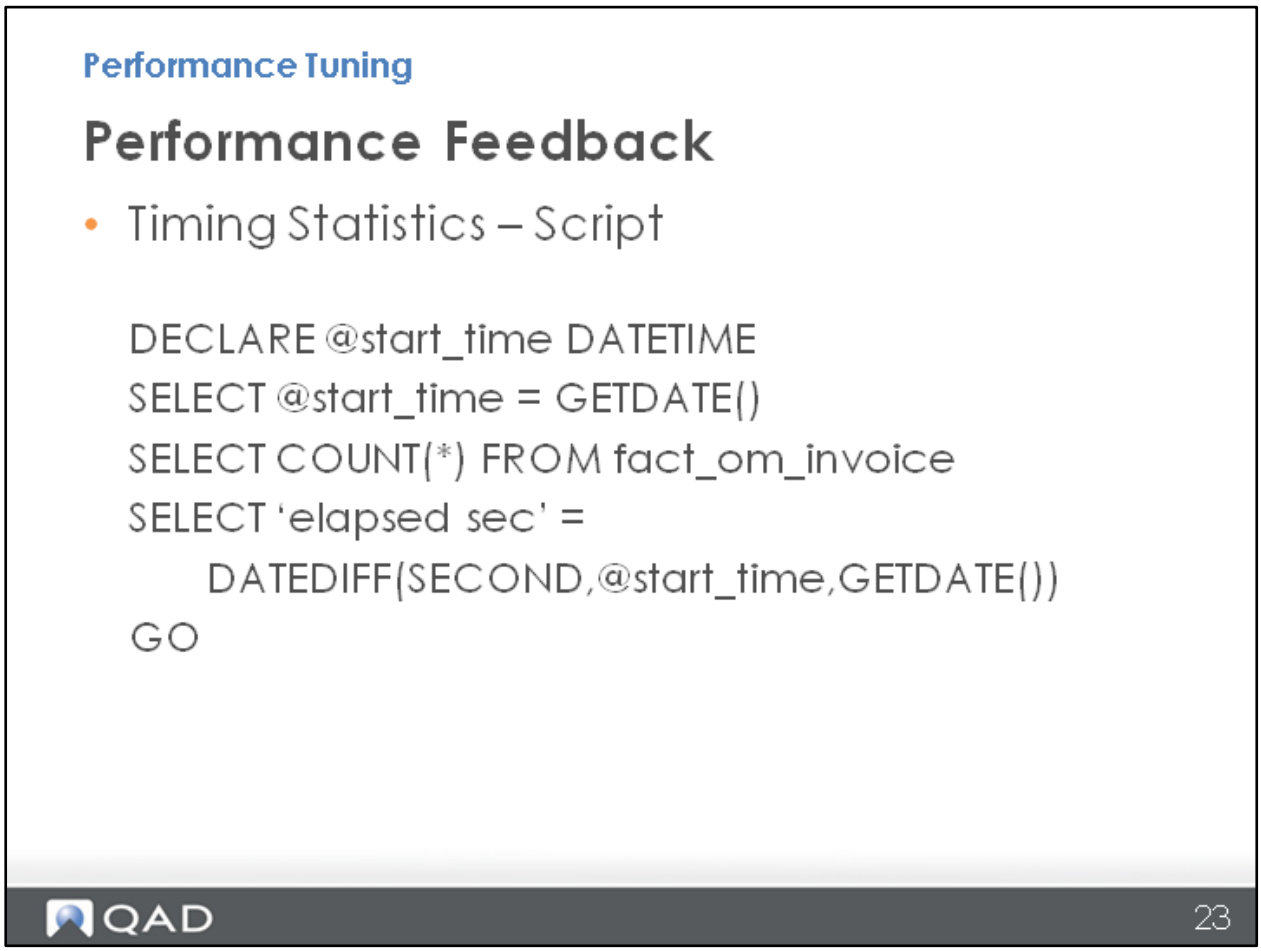
Performance Feedback



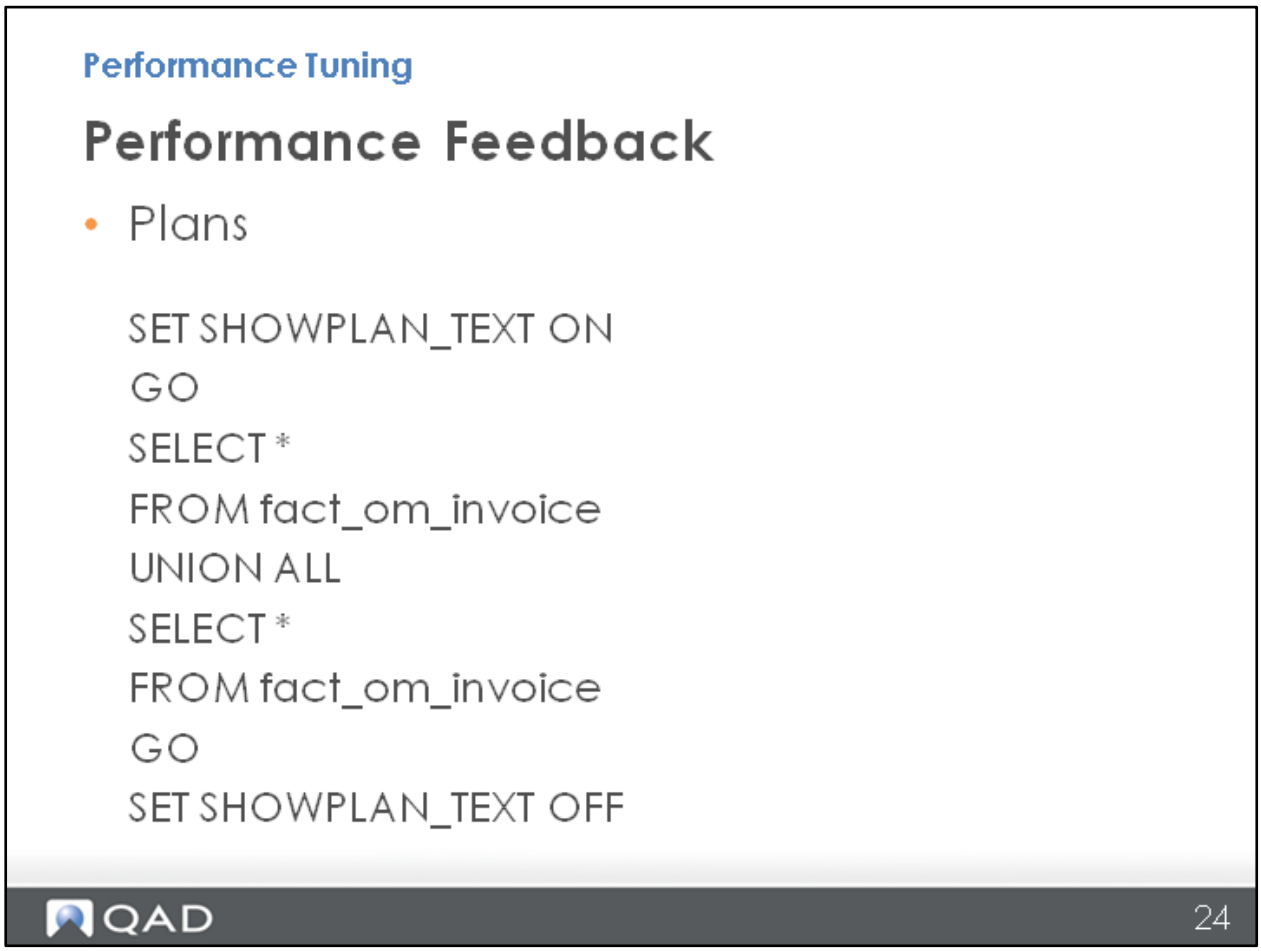
Performance Feedback



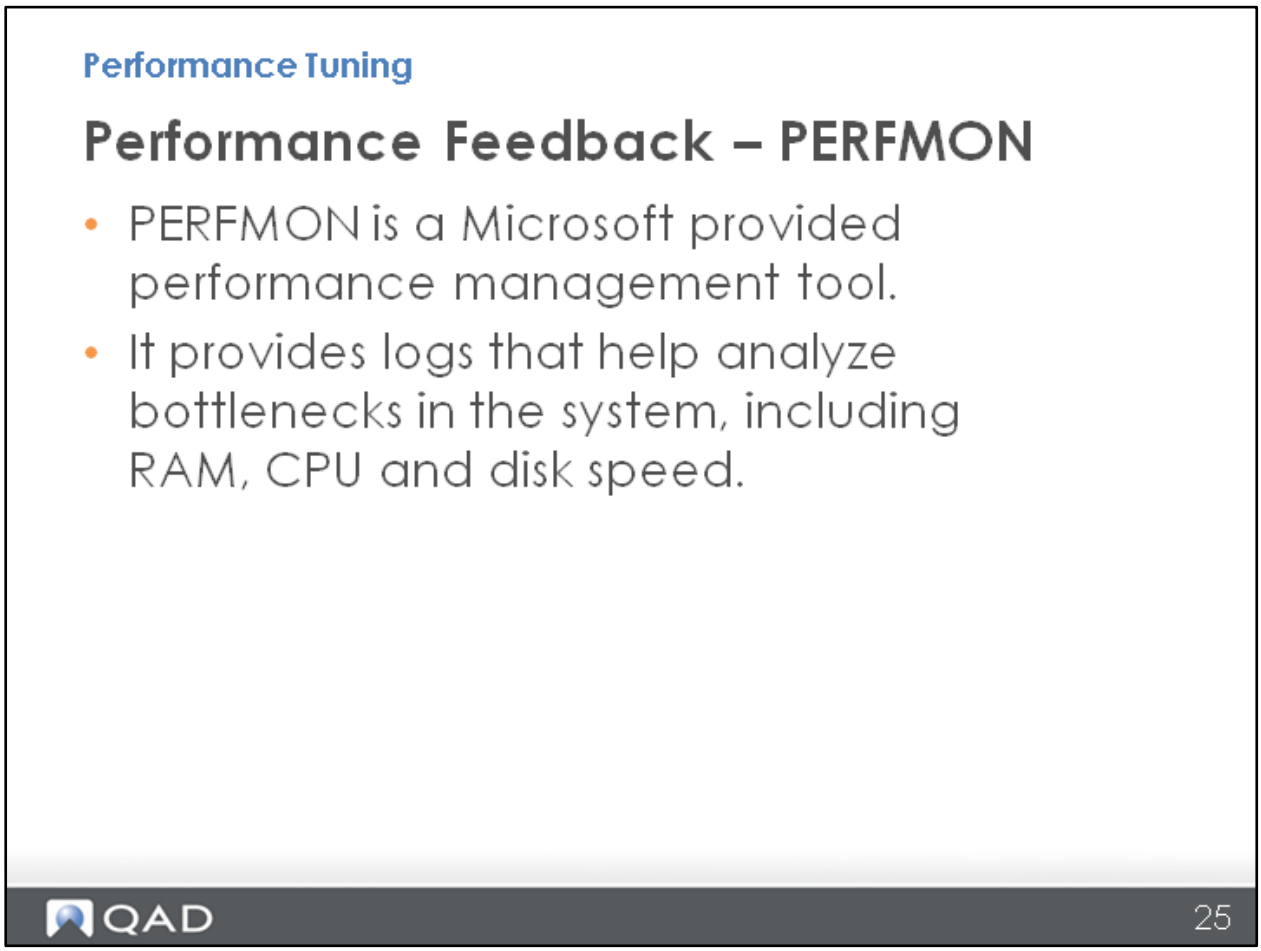
Performance Feedback



Performance Feedback



Performance Feedback – PERFMON

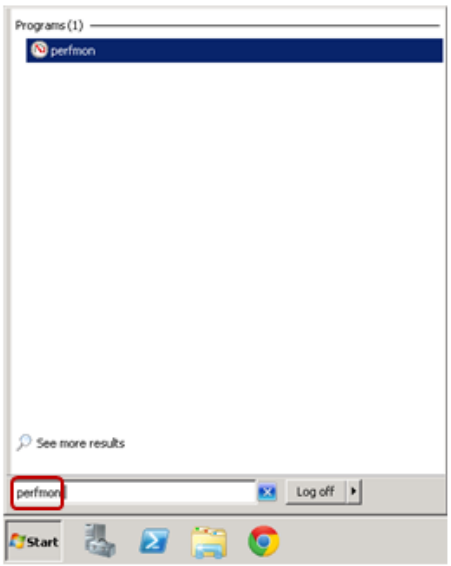


Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 1) Launch PERFMON (perfmon.exe).



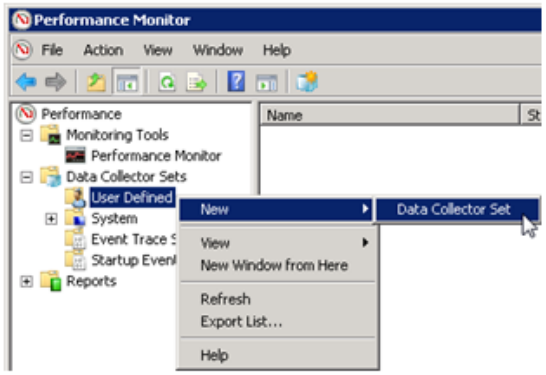
QAD 26

Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 2) Go to Performance / Data Collector Sets / User Defined. Right click on User Defined then select New / Data Collector Set



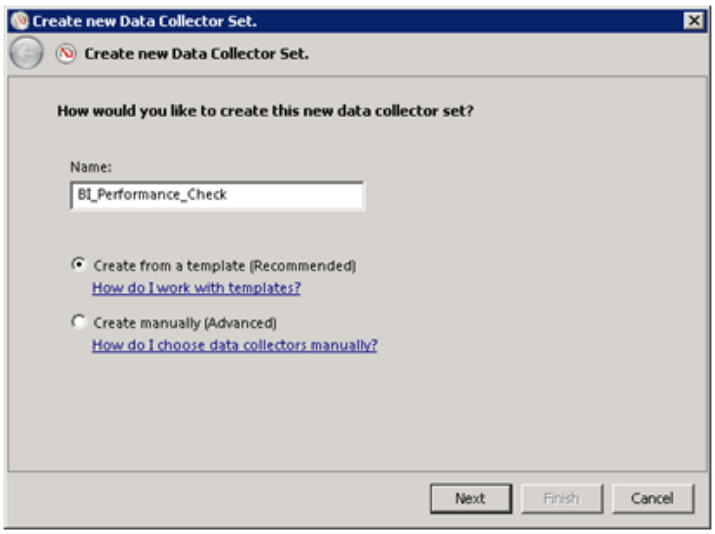
QAD 27

Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 3) Enter a file name then click Next.



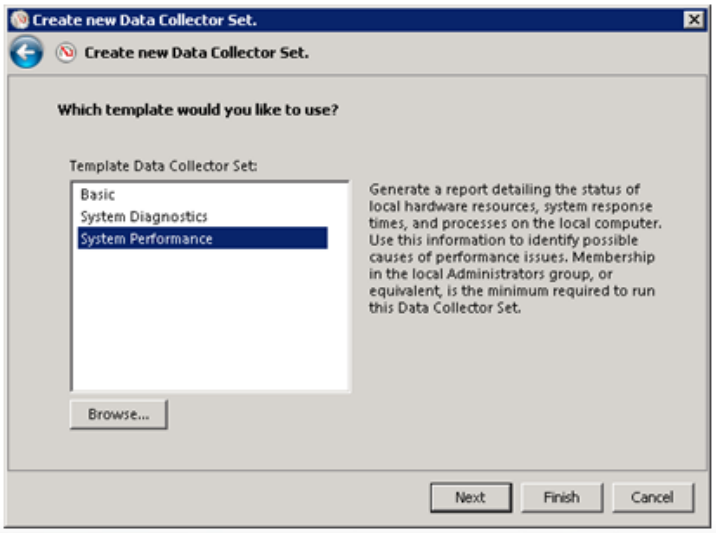
QAD 28

Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 4) Select System Performance then click Next.



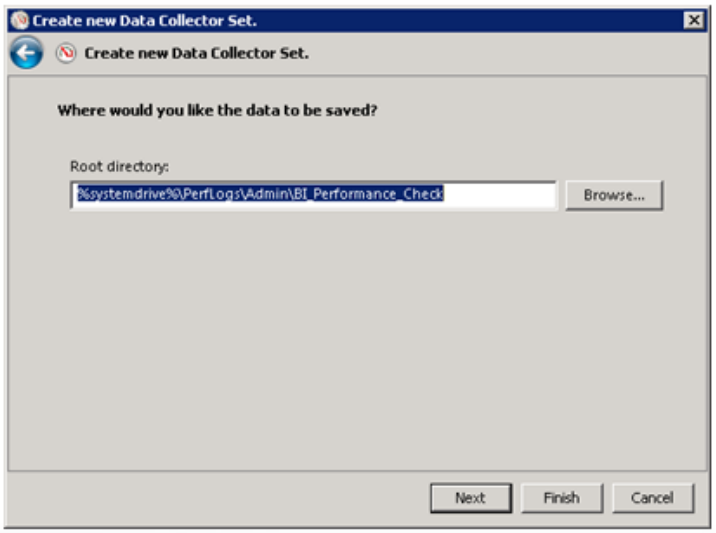
QAD 29

Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 5) Confirm directory where the log will be saved at, then click Next.



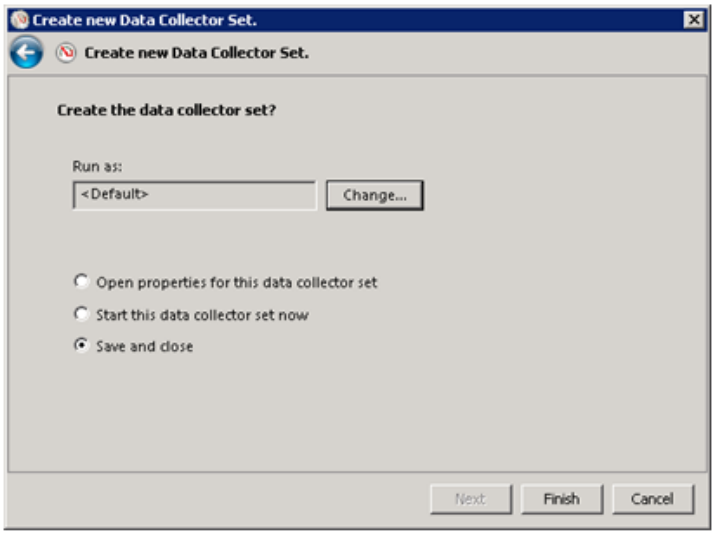
QAD 30

Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 6) Choose Save and close, then click Finish



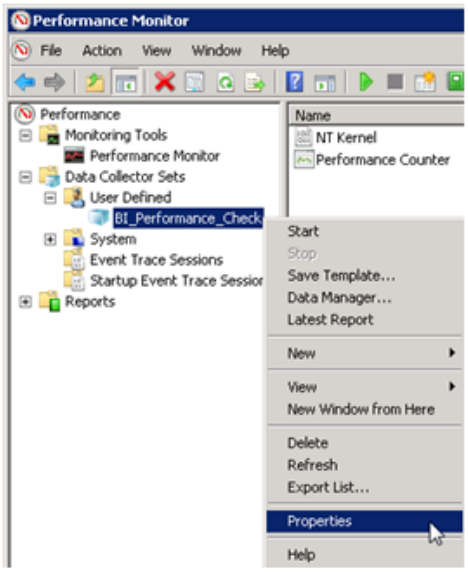
QAD 31

Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 7) Right click on the new Data Collector Set and select Properties.



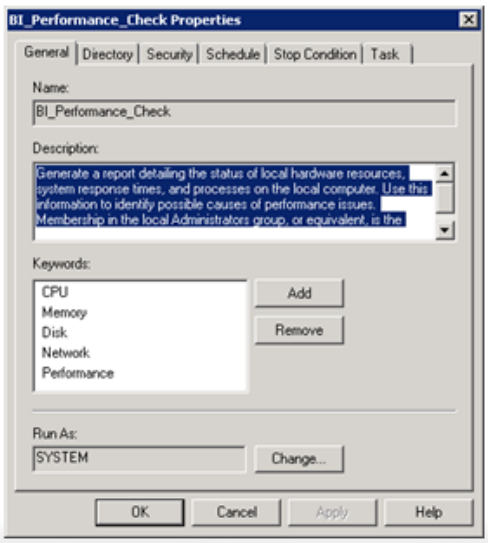
QAD 32

Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 8) Review the settings then click OK. All the default values are fine.



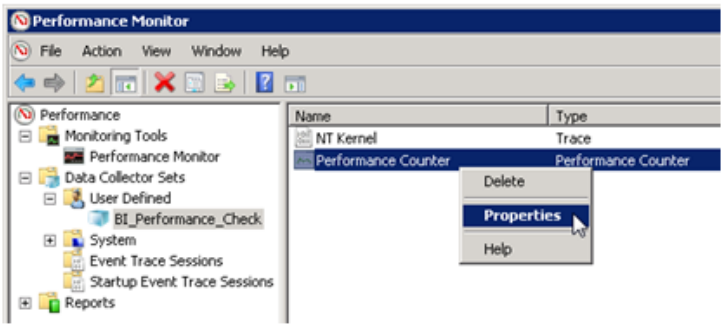
QAD 33

Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 9) Right click on Performance Counter then select Properties.



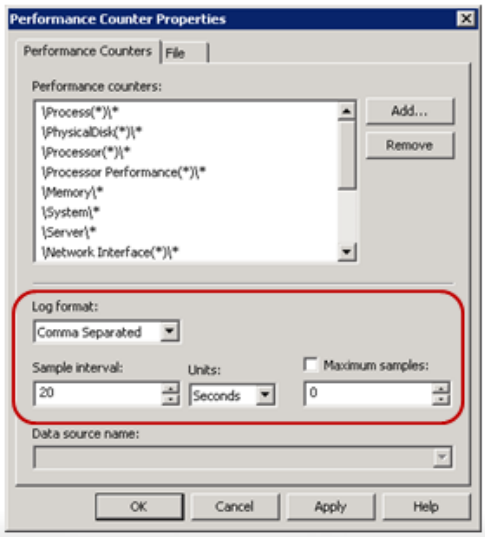
QAD 34

Performance Feedback – PERFMON

Performance Tuning

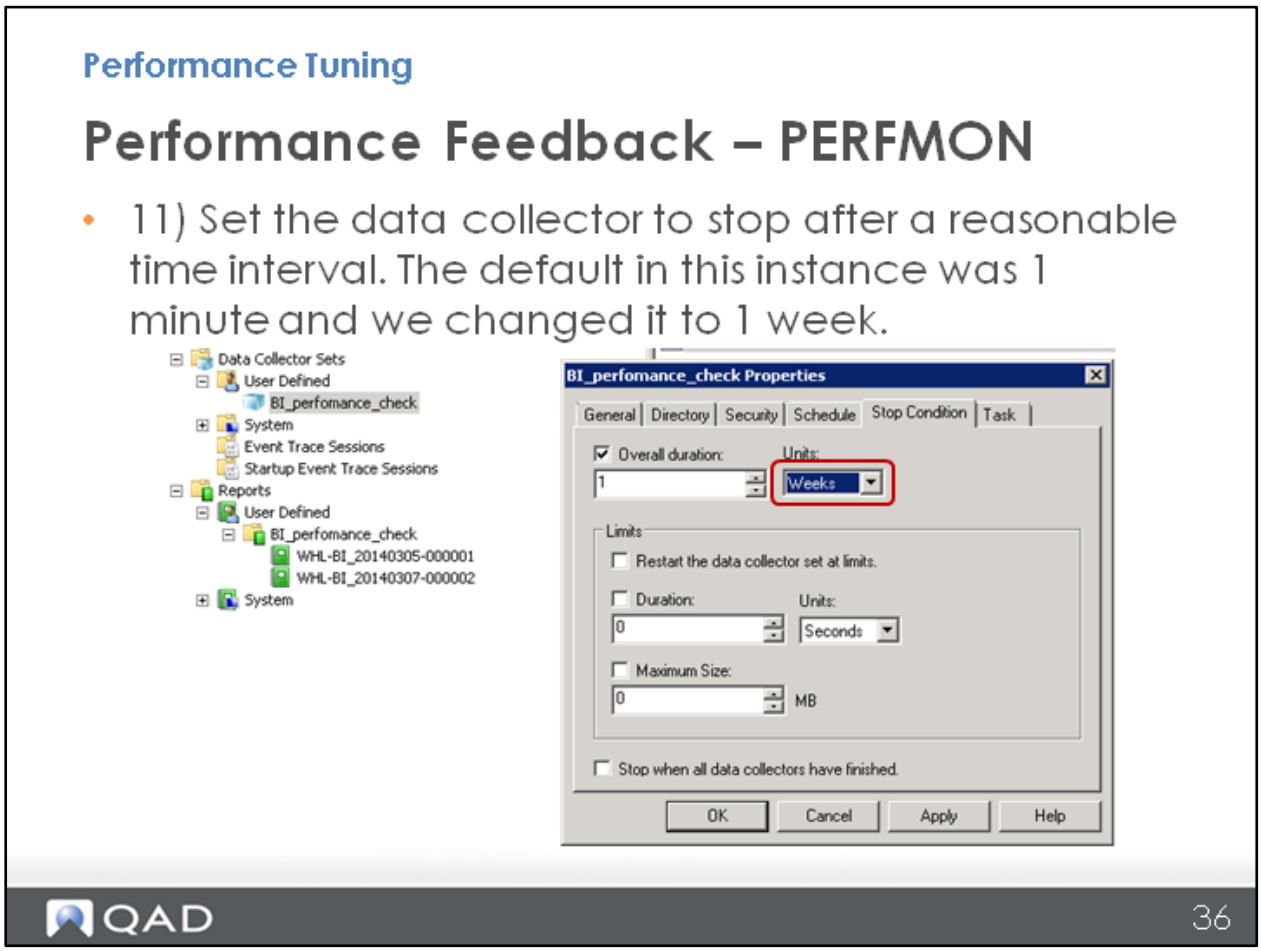
Performance Feedback – PERFMON

- 10) Set the Log format = Comma Separated, Sample interval = 20 Seconds then click OK.



QAD 35

Performance Feedback – PERFMON



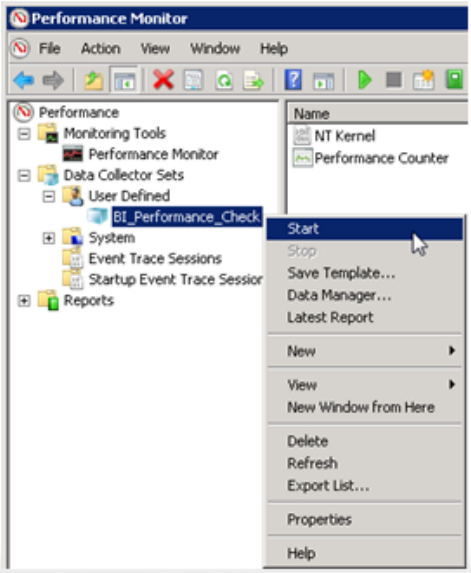
Right click on the User Defined Data Collector Set, in this case BI_performance_check and choose Properties. Go to the Stop Condition tab and make sure that the Overall duration makes sense.

Performance Feedback – PERFMON

Performance Tuning

Performance Feedback – PERFMON

- 12) With length of run time parameter set, start the Data Collector set.



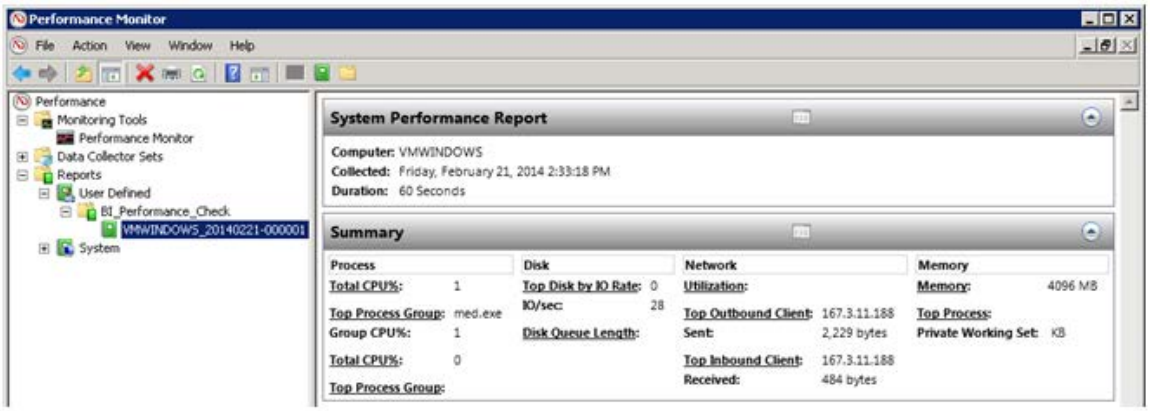
QAD 37

Performance Feedback – PERFMON

Performance Tuning

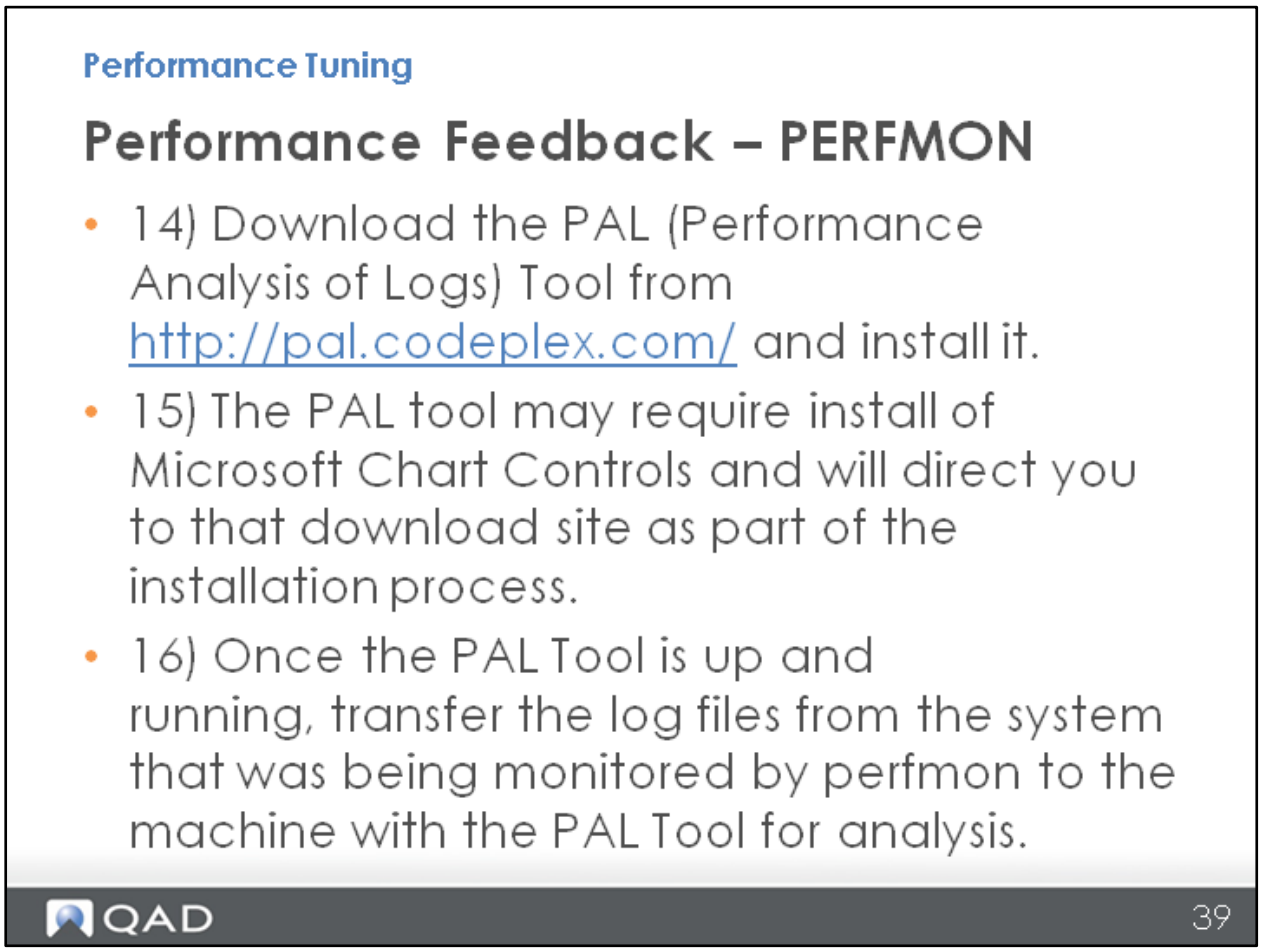
Performance Feedback – PERFMON

- 13) Review the report.

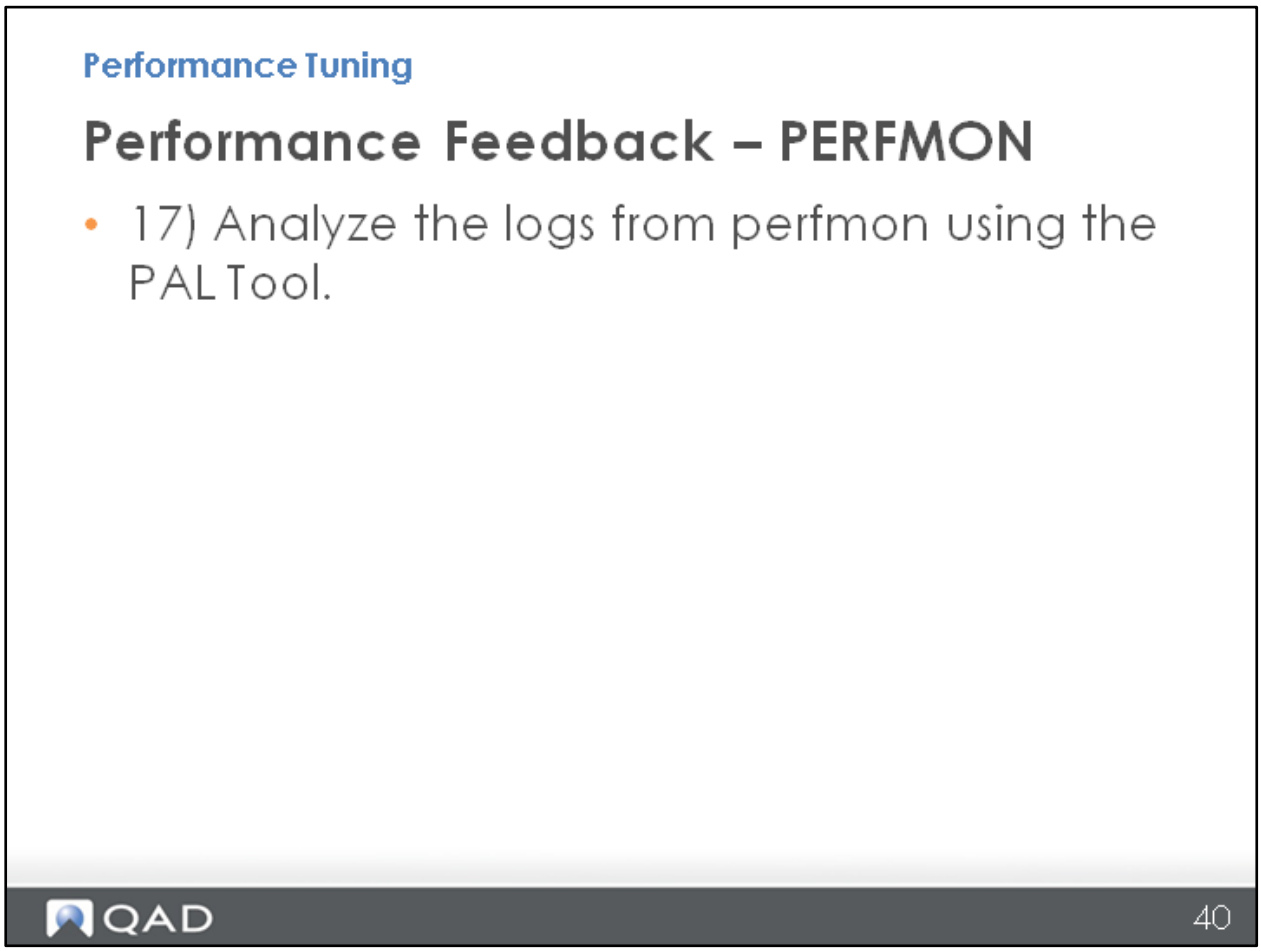


QAD 38

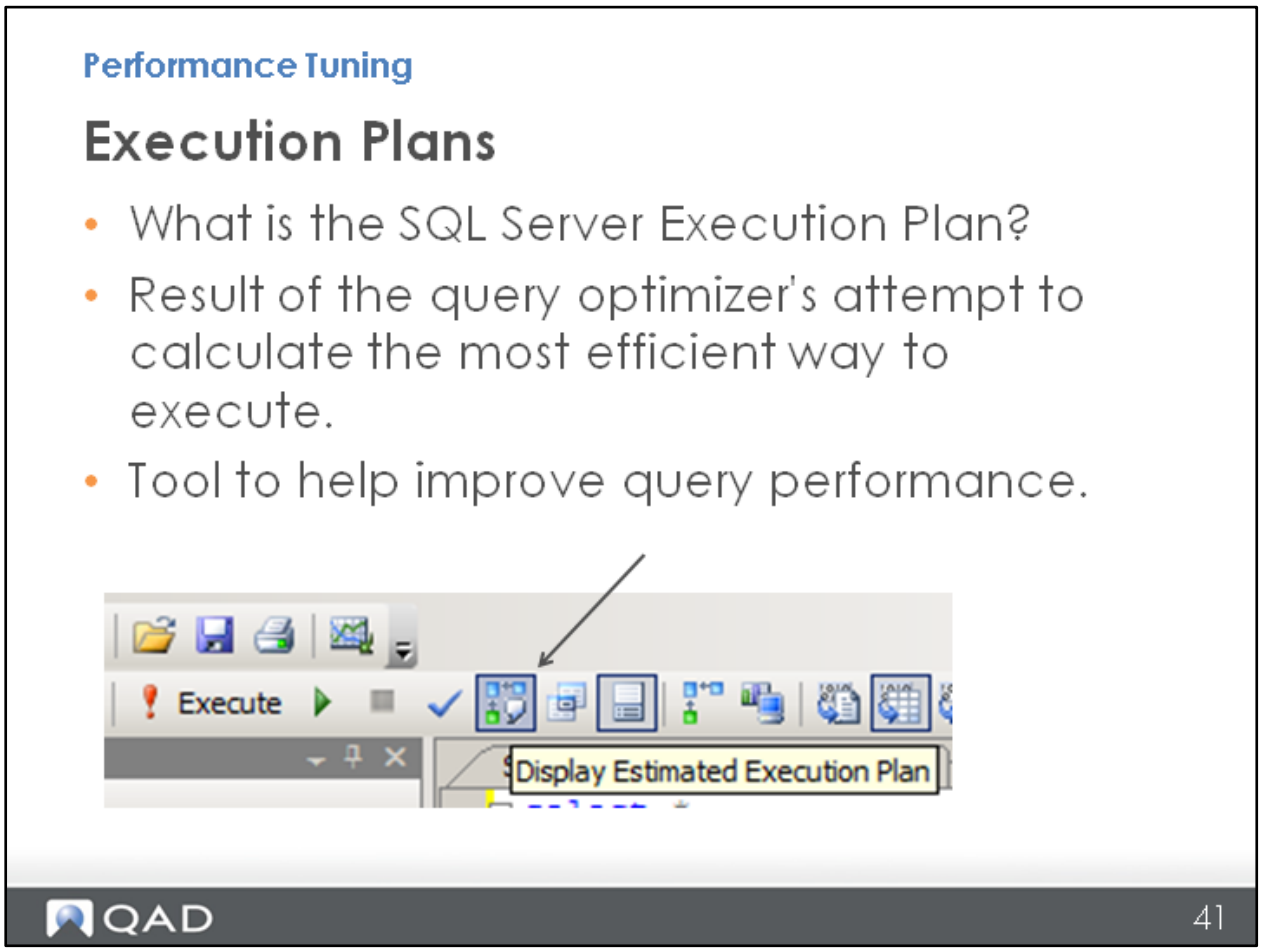
Performance Feedback – PERFMON



Performance Feedback – PERFMON

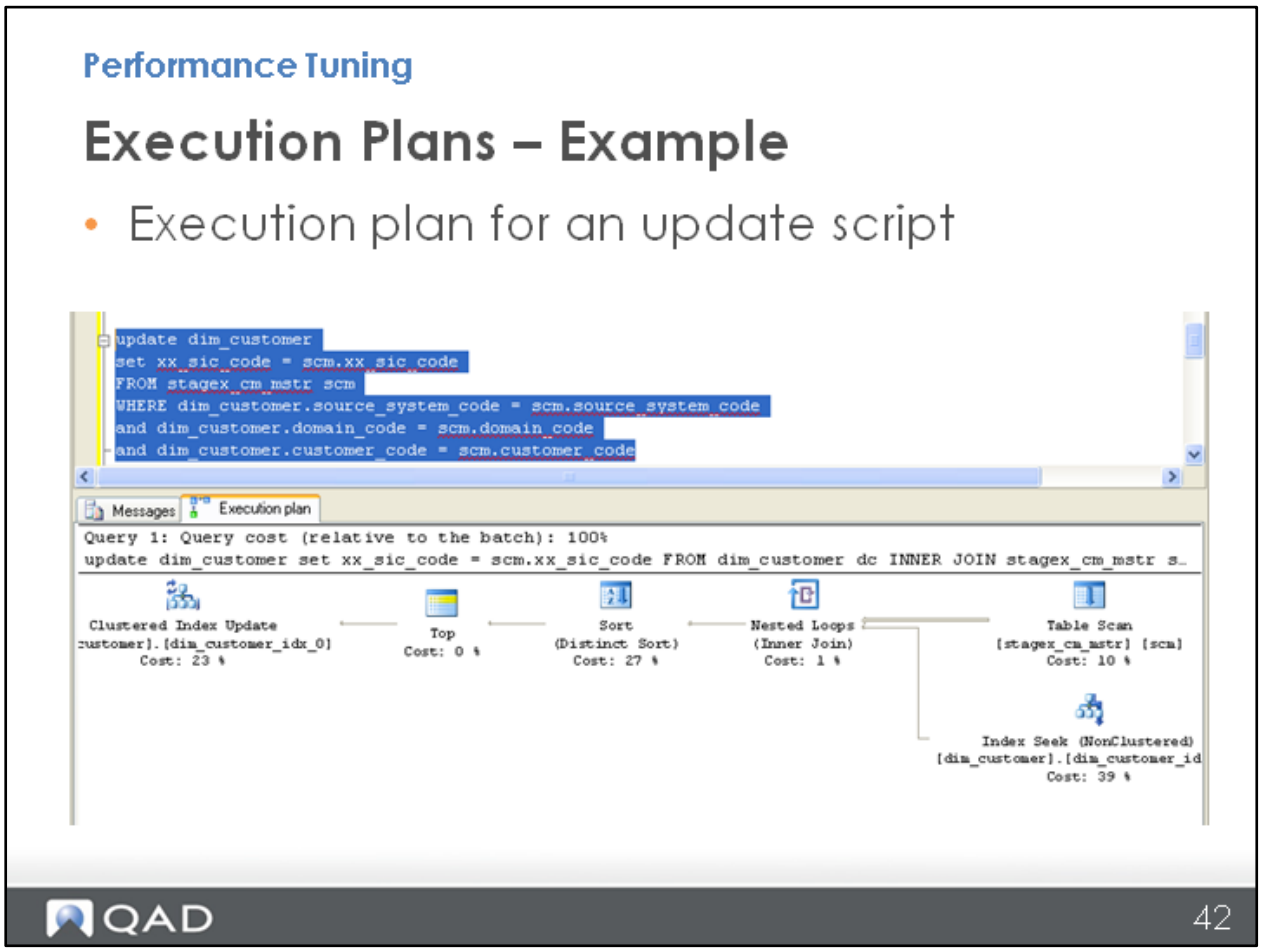


Execution Plans



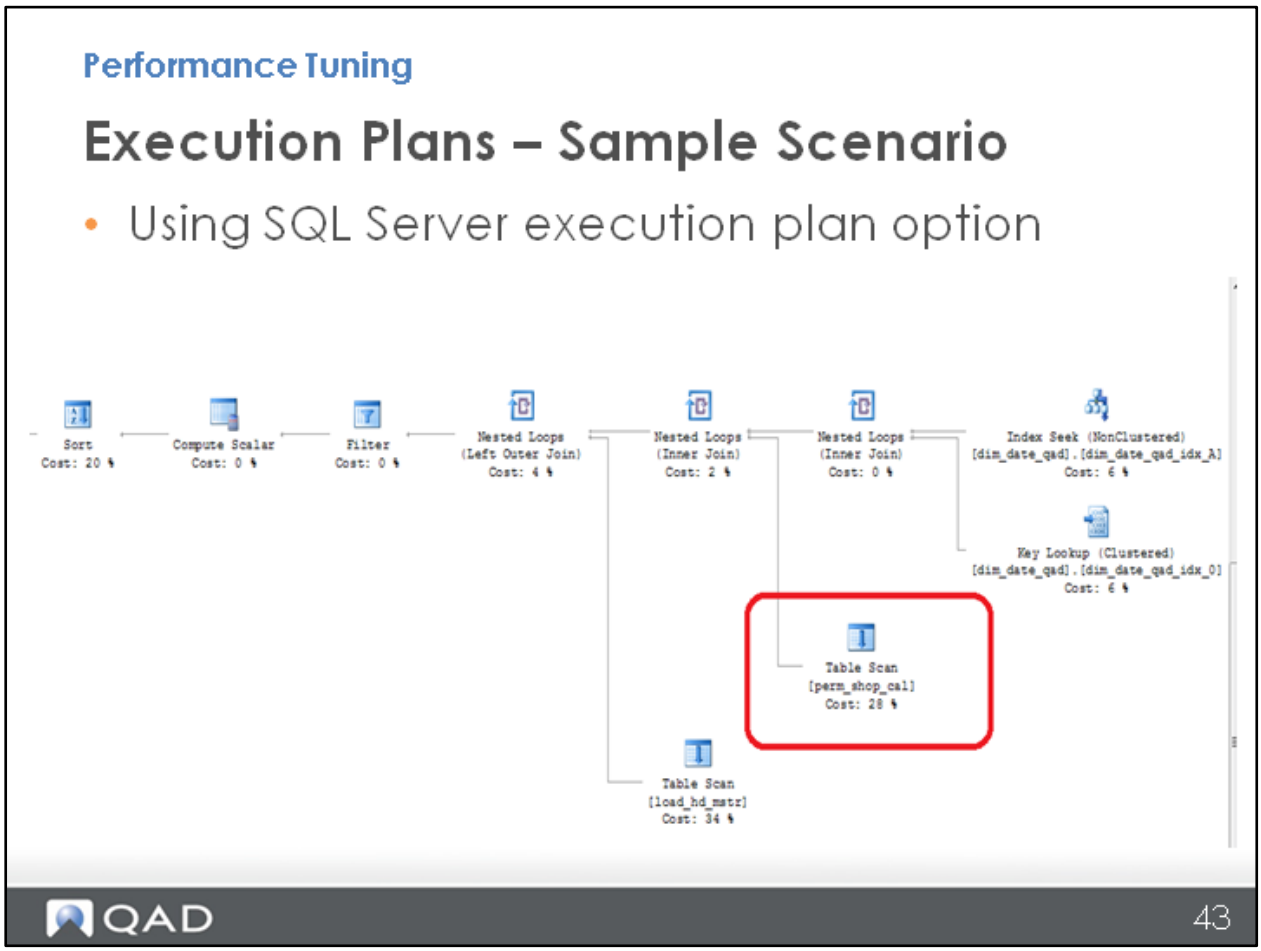
An execution plan documents the estimated cost of a query, the indexes used, and the operations performed. All of this information is vitally important in attempting to speed up a slow performing query.

Execution Plans – Example



To read the plan chronologically, start with the right-most operations and work left. Also, the relative thickness of the arrows is an indication of how much data is being passed from one operation to the next.

Execution Plans – Sample Scenario



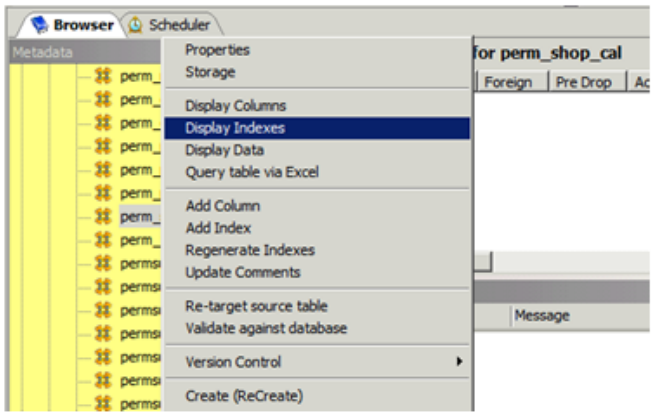
A job in purchasing was taking quite a long time to load for a fairly small record set. It was taking around 45 minutes to run a job for 28 thousand records. An execution plan was run on the query, and on the function call in the query. What you see here is part of the execution plan for the function call. The call to the perm table was one of the biggest resource offenders and we're supposed to have indexes on perm tables. Instead of referencing indexes though, a Full Table Scan is occurring on that perm table.

Execution Plans – Sample Scenario

Performance Tuning

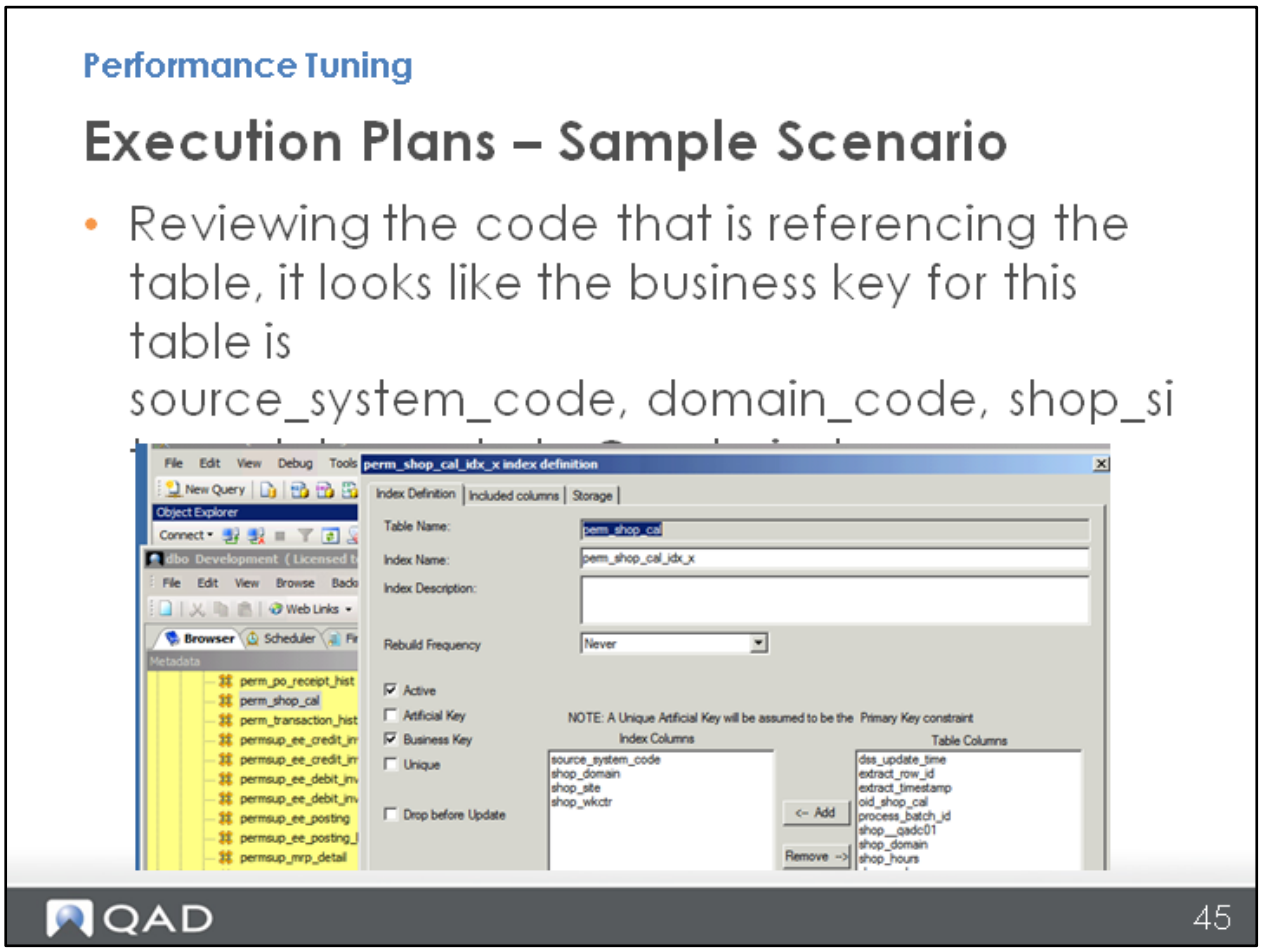
Execution Plans – Sample Scenario

- Checking the indexes for perm_shop_cal reveals that none exist for it.



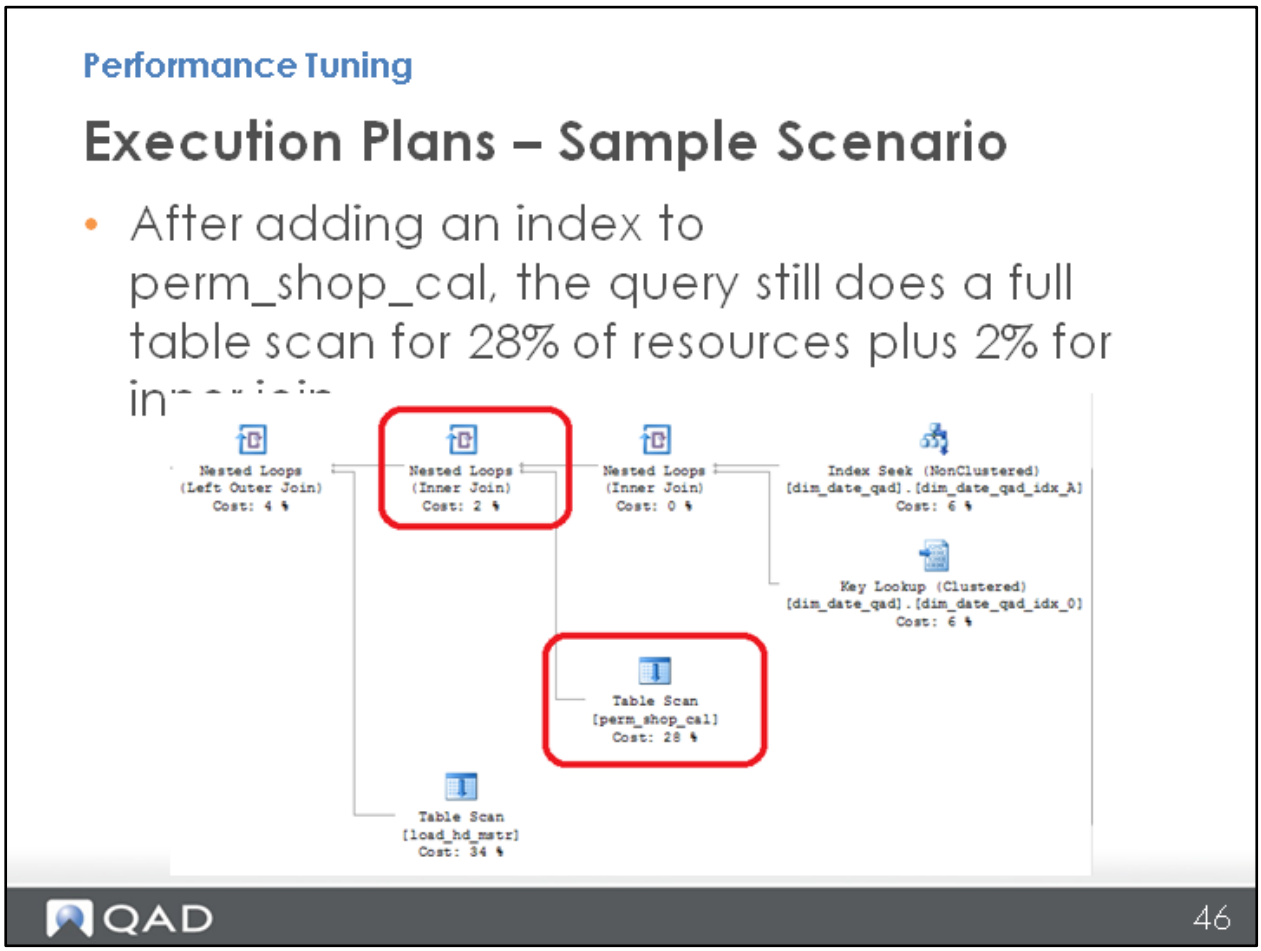
QAD 44

Execution Plans – Sample Scenario



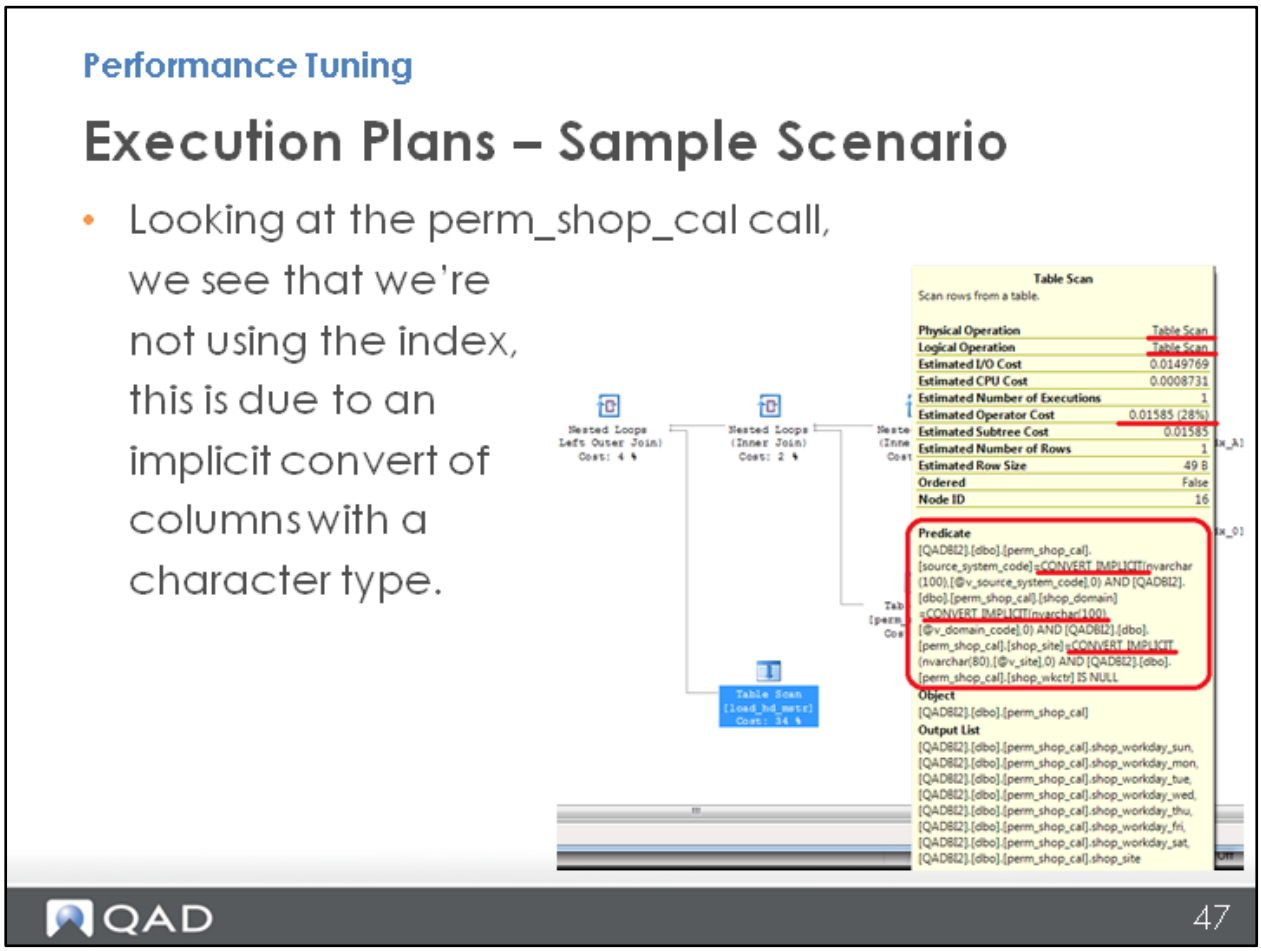
Create an index using the elements that make up the business key. These also tied to the WHERE clause in the function that is being called.

Execution Plans – Sample Scenario



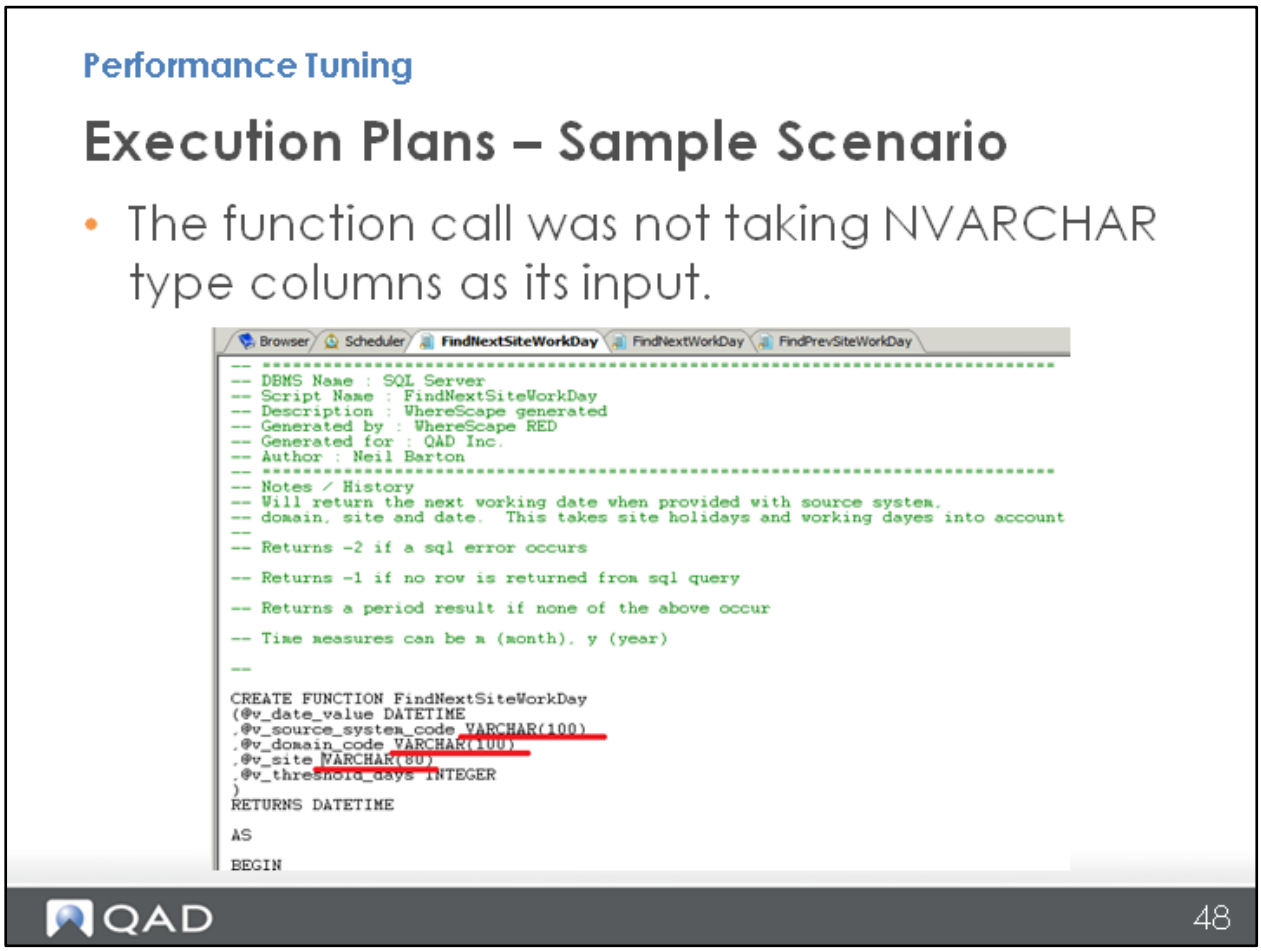
After the indexes were put on the perm_shop_cal table, the performance of the job for 28 thousand records improved from 44 minutes to 4 minutes. However, it looks like the index was still not being properly used. A Table Scan is being done on that table for 28% of the resources. Why aren't the indexes being utilized?

Execution Plans – Sample Scenario



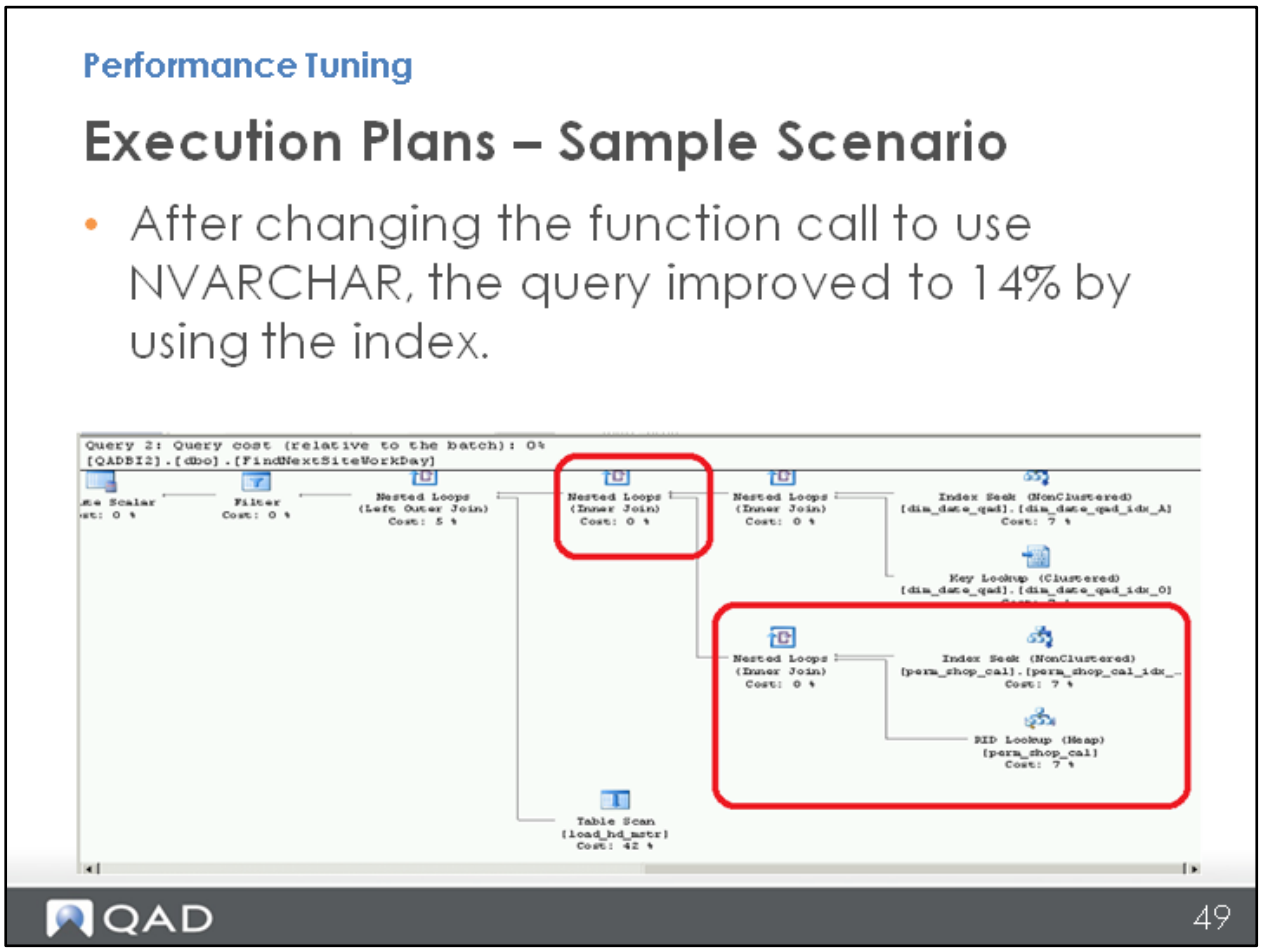
Looking at the node in the execution plan that referenced this table, we can see that all of our joins are doing a CONVERT_IMPLICIT on each of the character type column.

Execution Plans – Sample Scenario



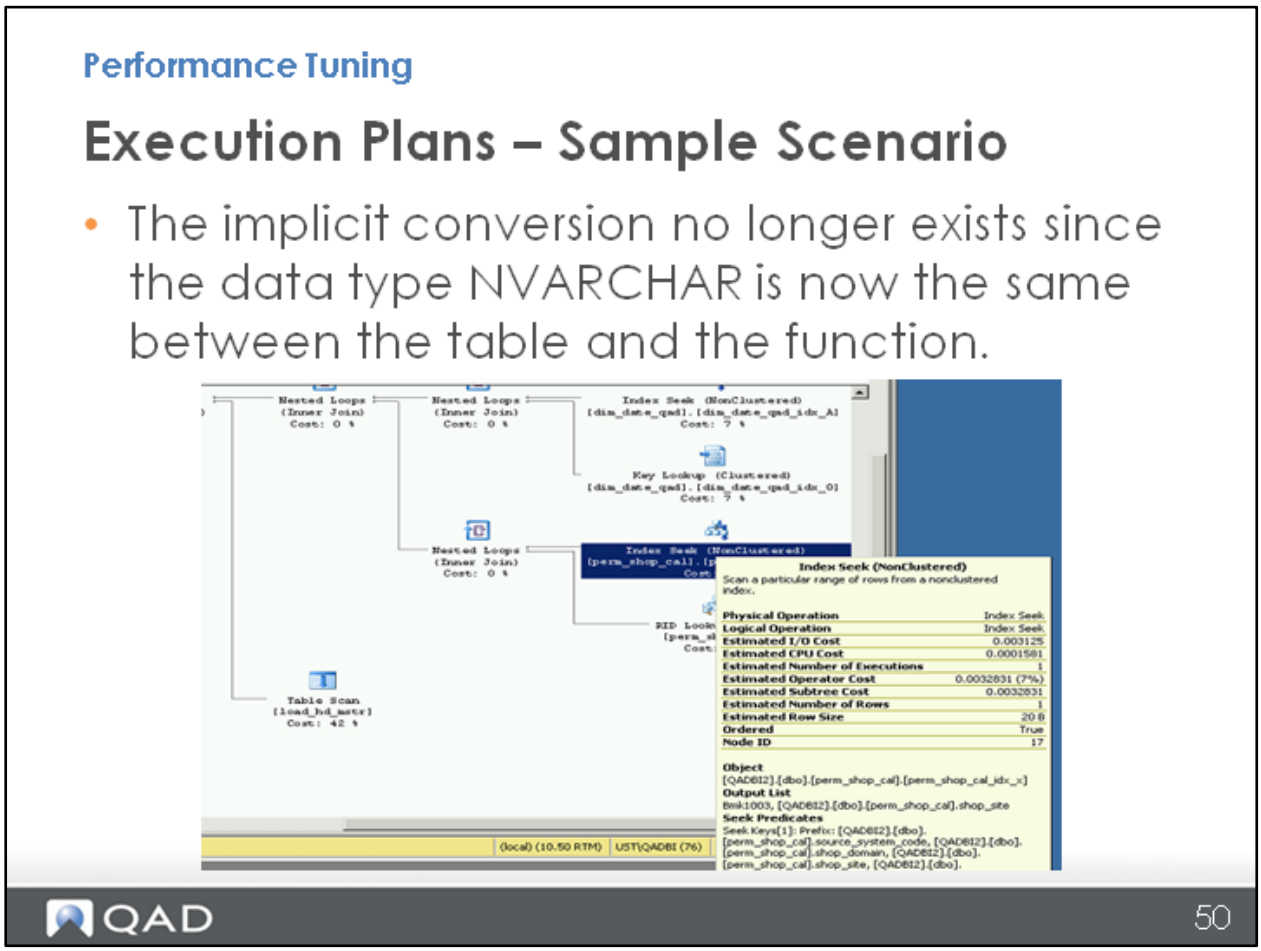
Reviewing the function call that this execution plan was run for, we can see that when it was written, it was created with varchar inputs instead of nvarchar inputs (either because it was written prior to us converting all of our character columns to nvarchar or as a simple oversight). Because the values being passed in were nvarchar, each join parameter value had to be converted to nvarchar. As a result, the index on the table was not being utilized.

Execution Plans – Sample Scenario



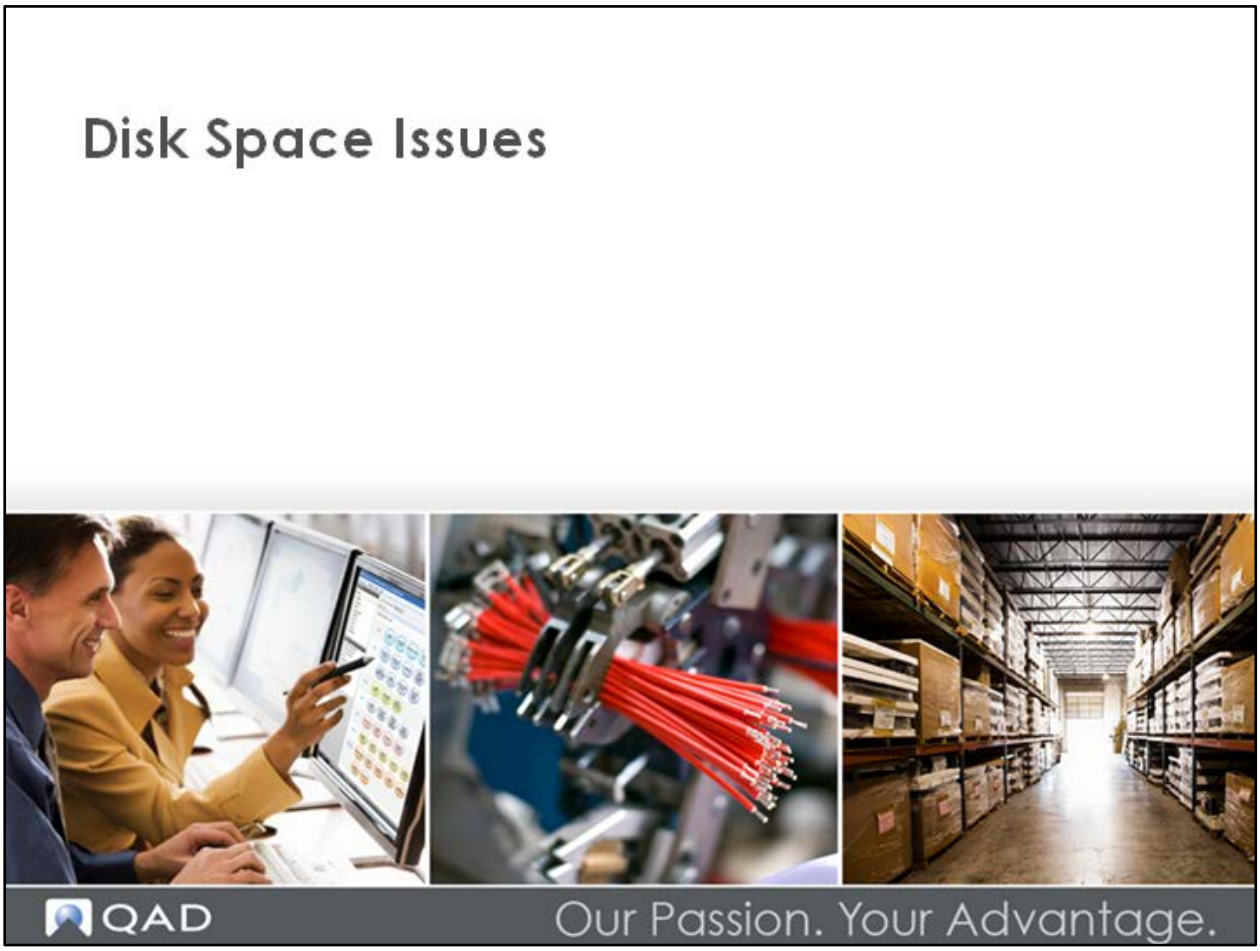
The improvement in calling the query in part was due to it properly using the index now. Before, when the nvarchar vs. varchar issue existed, the implicit conversion of the varchar column to nvarchar was causing the query to not use the index.

Execution Plans – Sample Scenario

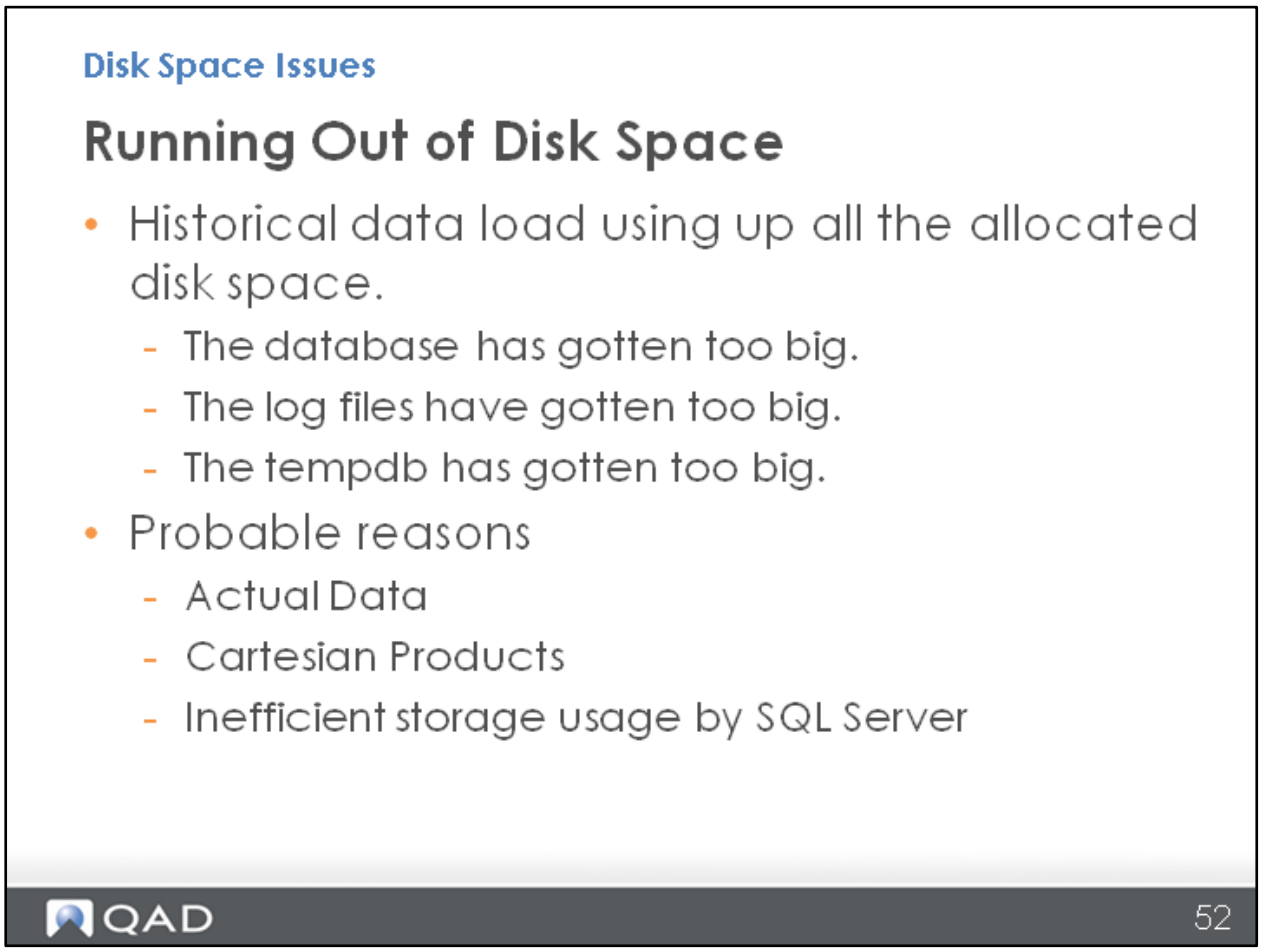


Examining the node again shows that the CONVERT_IMPLICIT are no longer there.

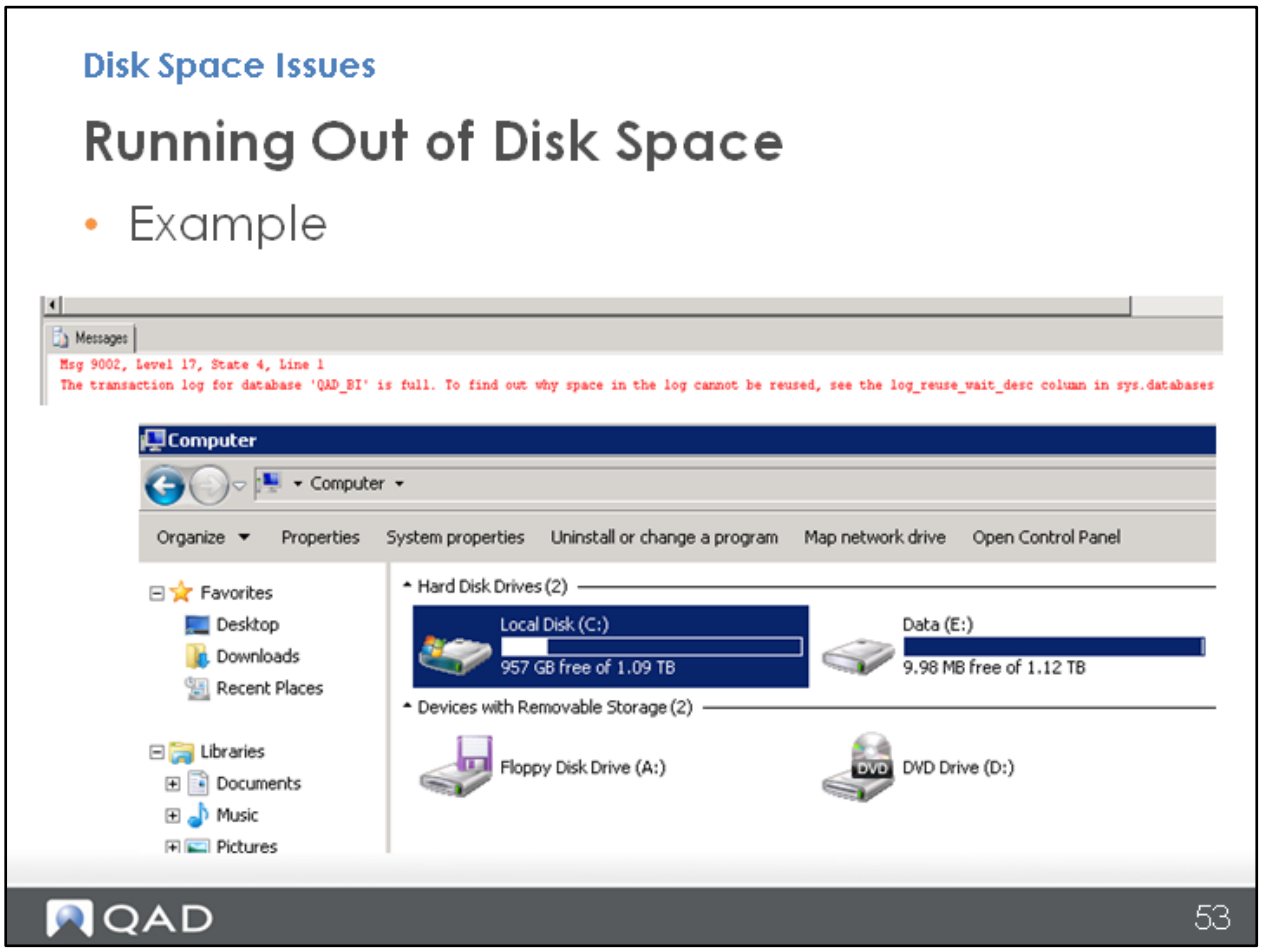
Disk Space Issues



Running Out of Disk Space

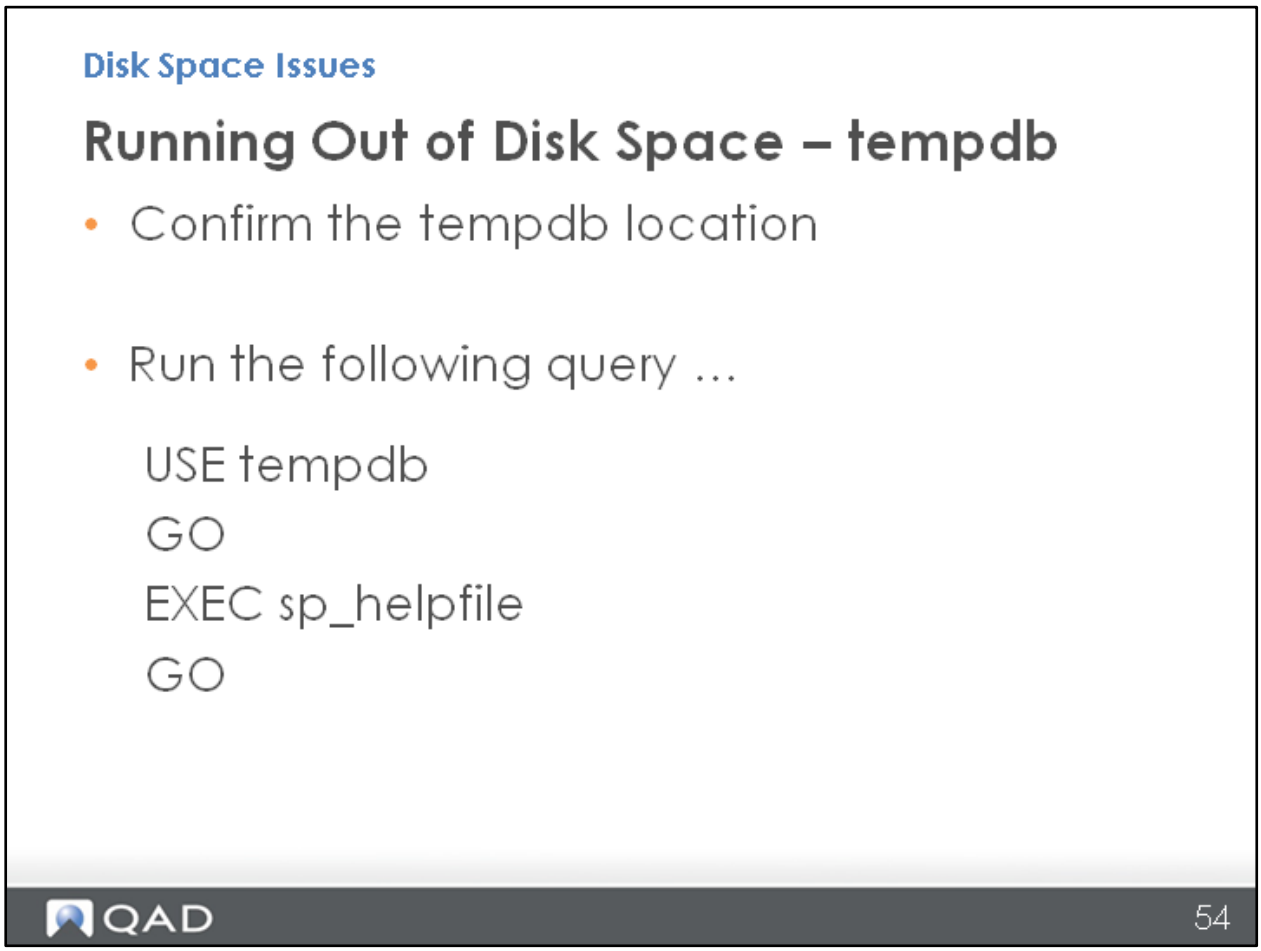


Running Out of Disk Space



Note that the C drive is fairly large in this instance and hardly used and the E drive is nearly the same size and almost completely full.

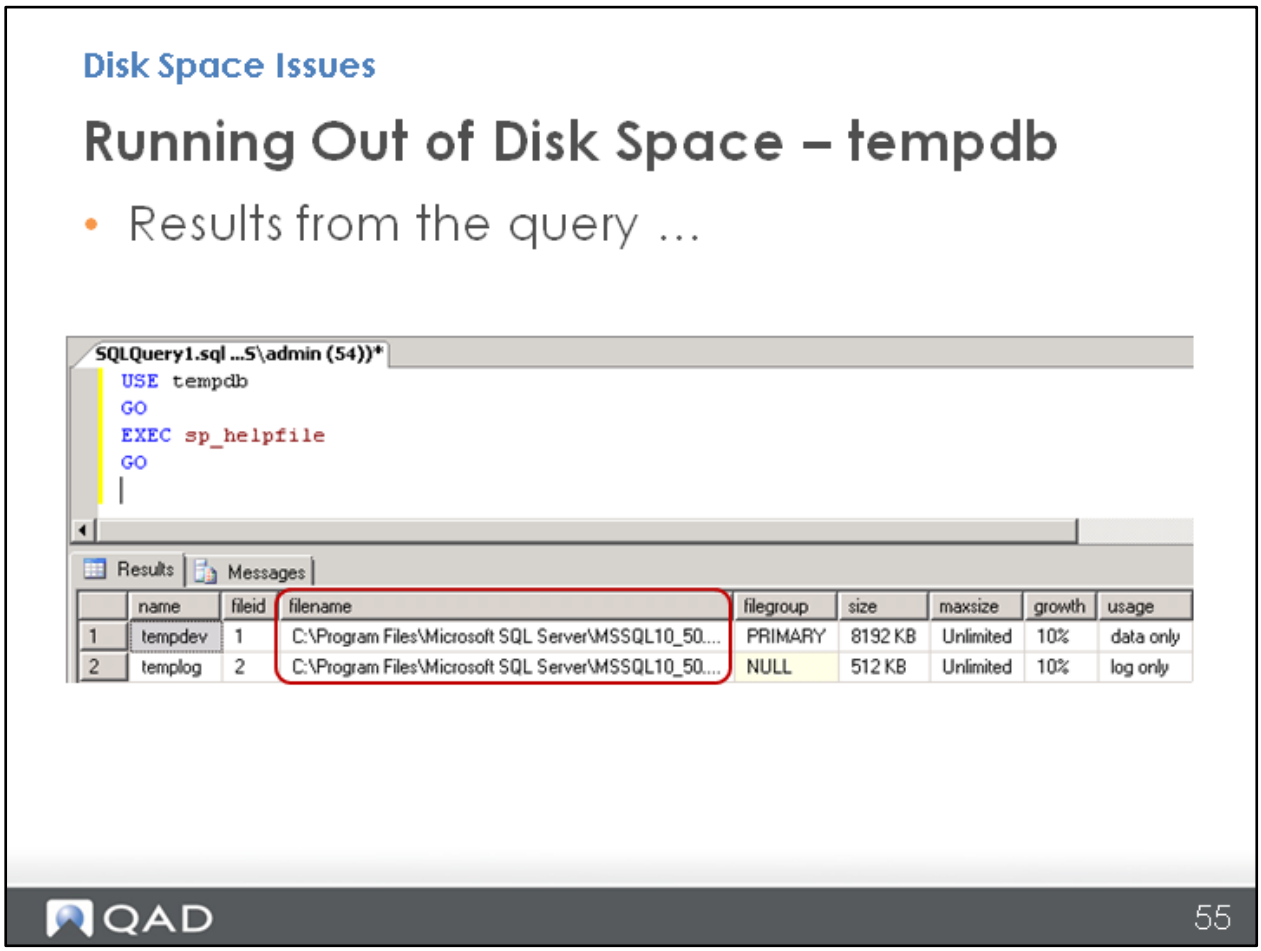
Running Out of Disk Space – tempdb



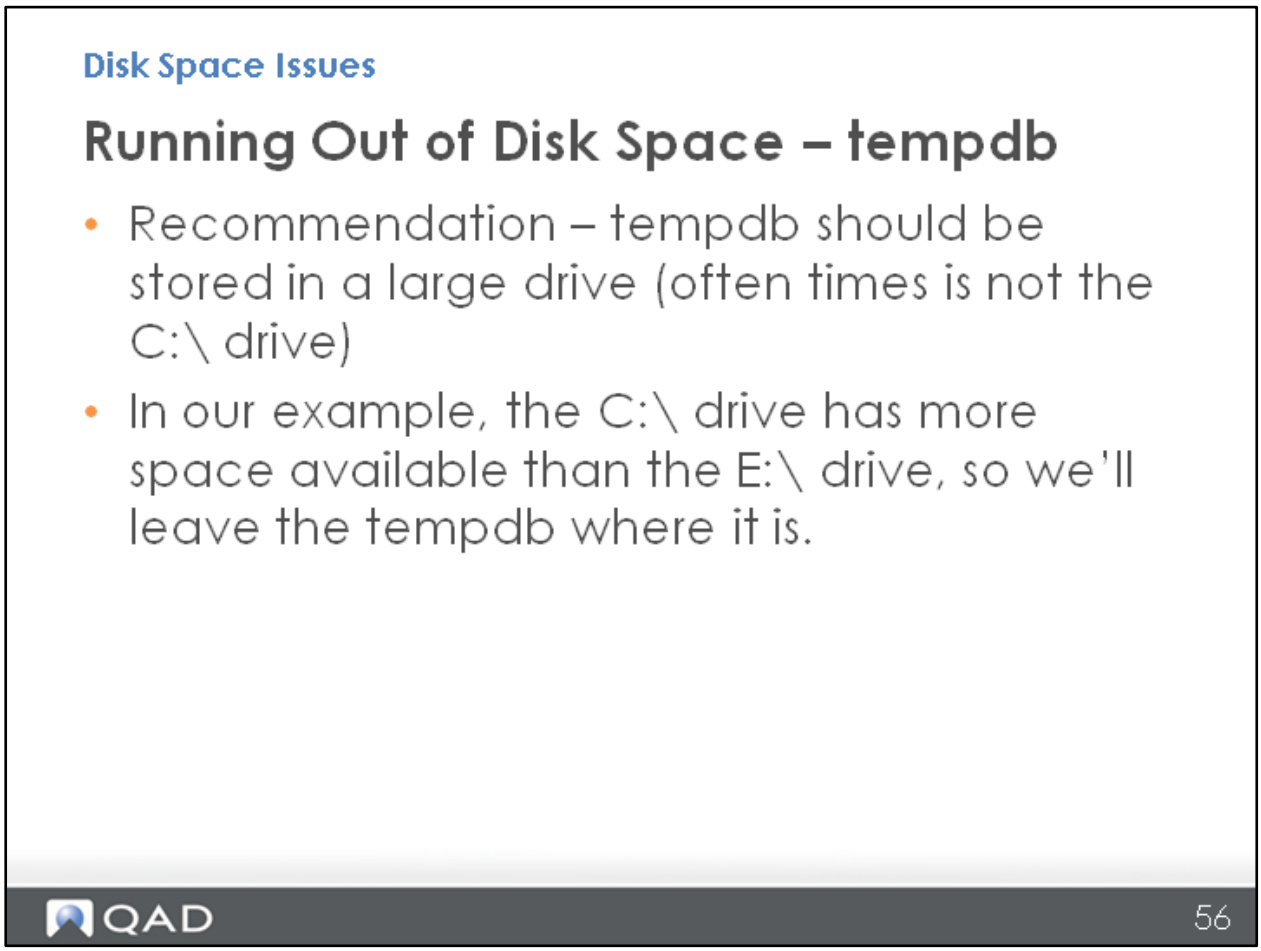
This is typically part of the installation process, but it definitely is wise to check.

The tempdb database is a global resource that is available to all users connected to the instance of SQL Server and is used to hold the following: temporary user objects that are explicitly created, such as global or local temporary tables, temporary stored procedures, table variables, and cursors.

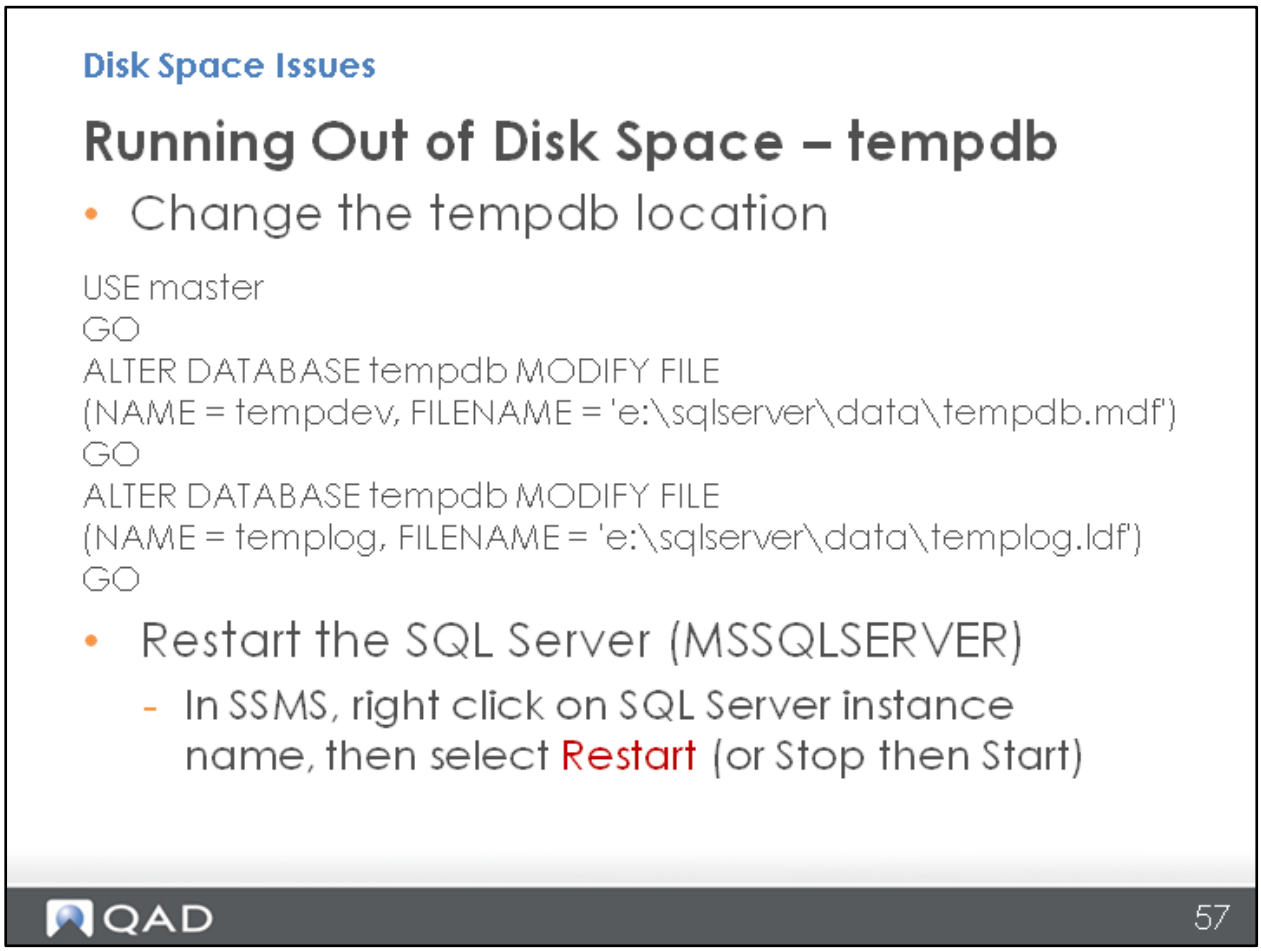
Running Out of Disk Space – tempdb



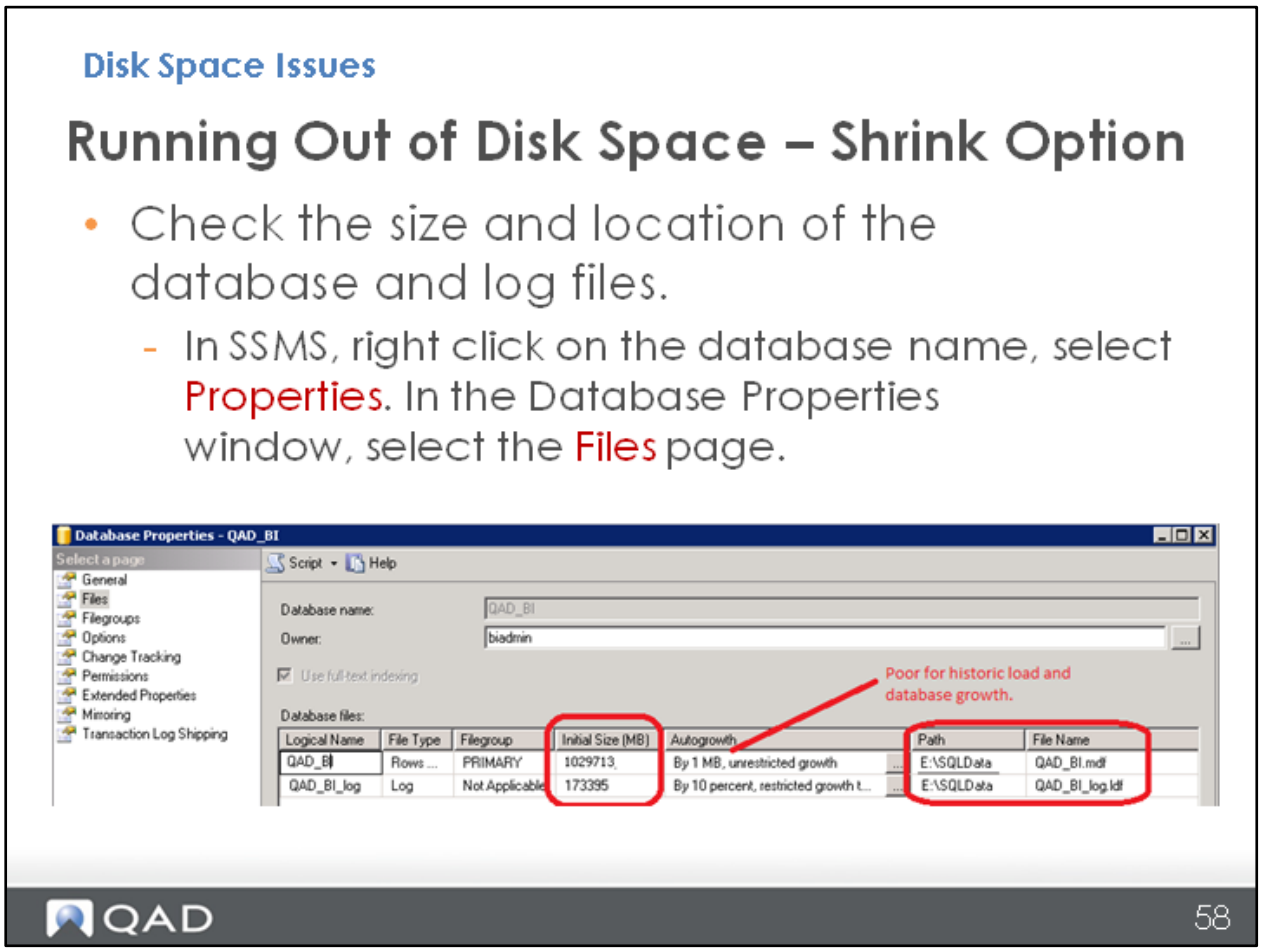
Running Out of Disk Space – tempdb



Running Out of Disk Space – tempdb



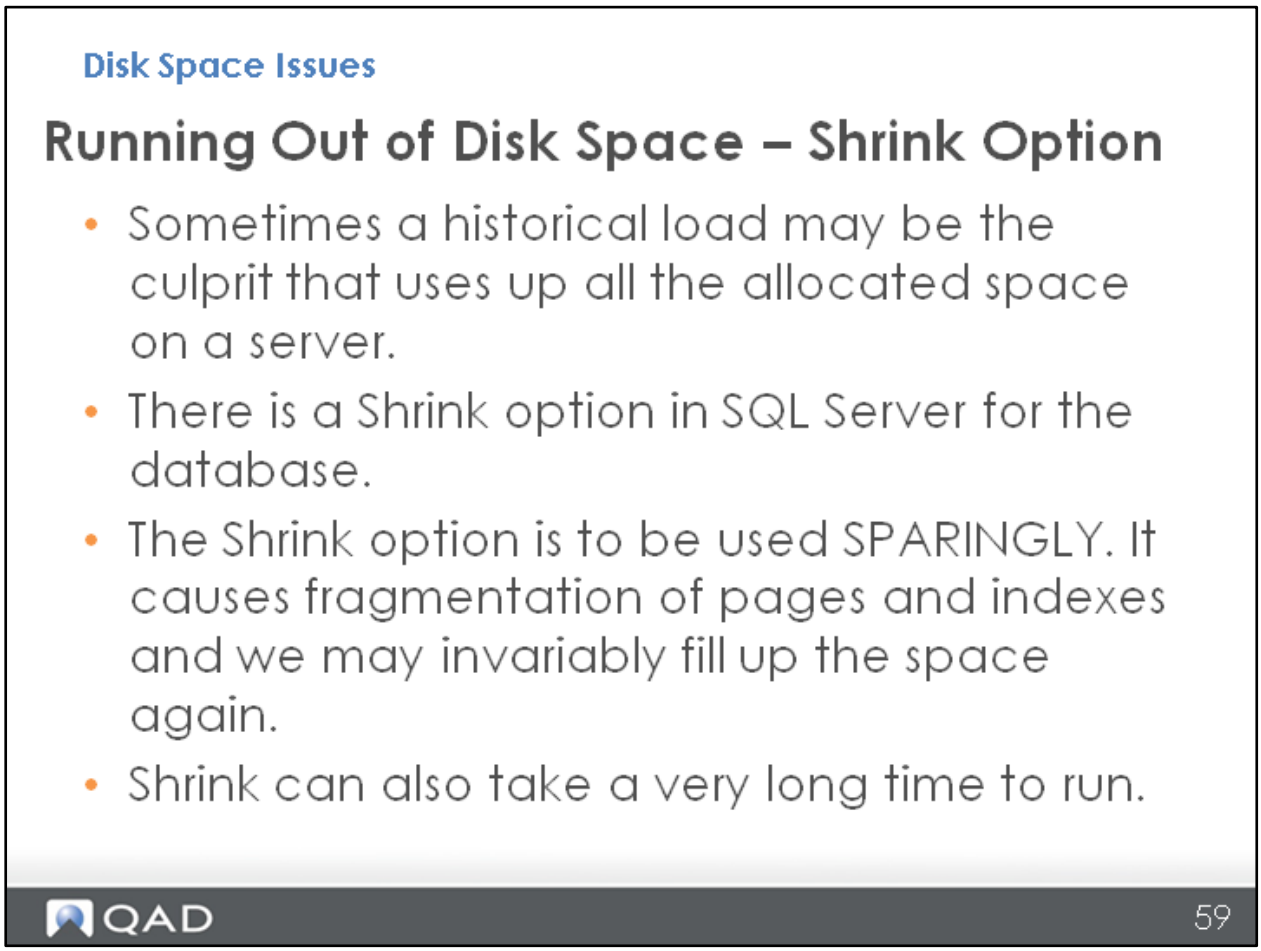
Running Out of Disk Space – Shrink Option



Files reside in the E:\SQLData directory.

Note that the Autogrowth was set at 1 MB, which means that if the database grew from 1 GB to 1100 GB, it had to Autogrow a million times. This should be set appropriately, like 10% or 100 GB.

Running Out of Disk Space – Shrink Option

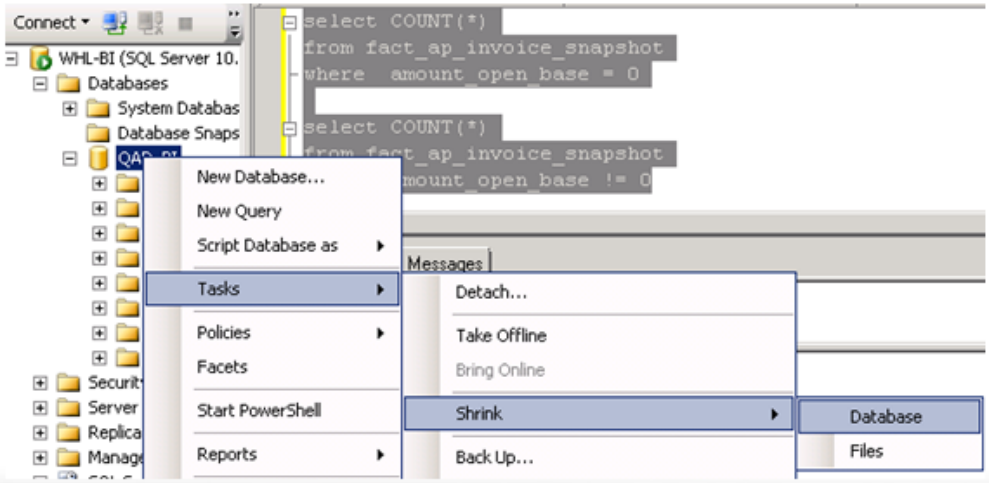


Running Out of Disk Space – Shrink Option

Disk Space Issues

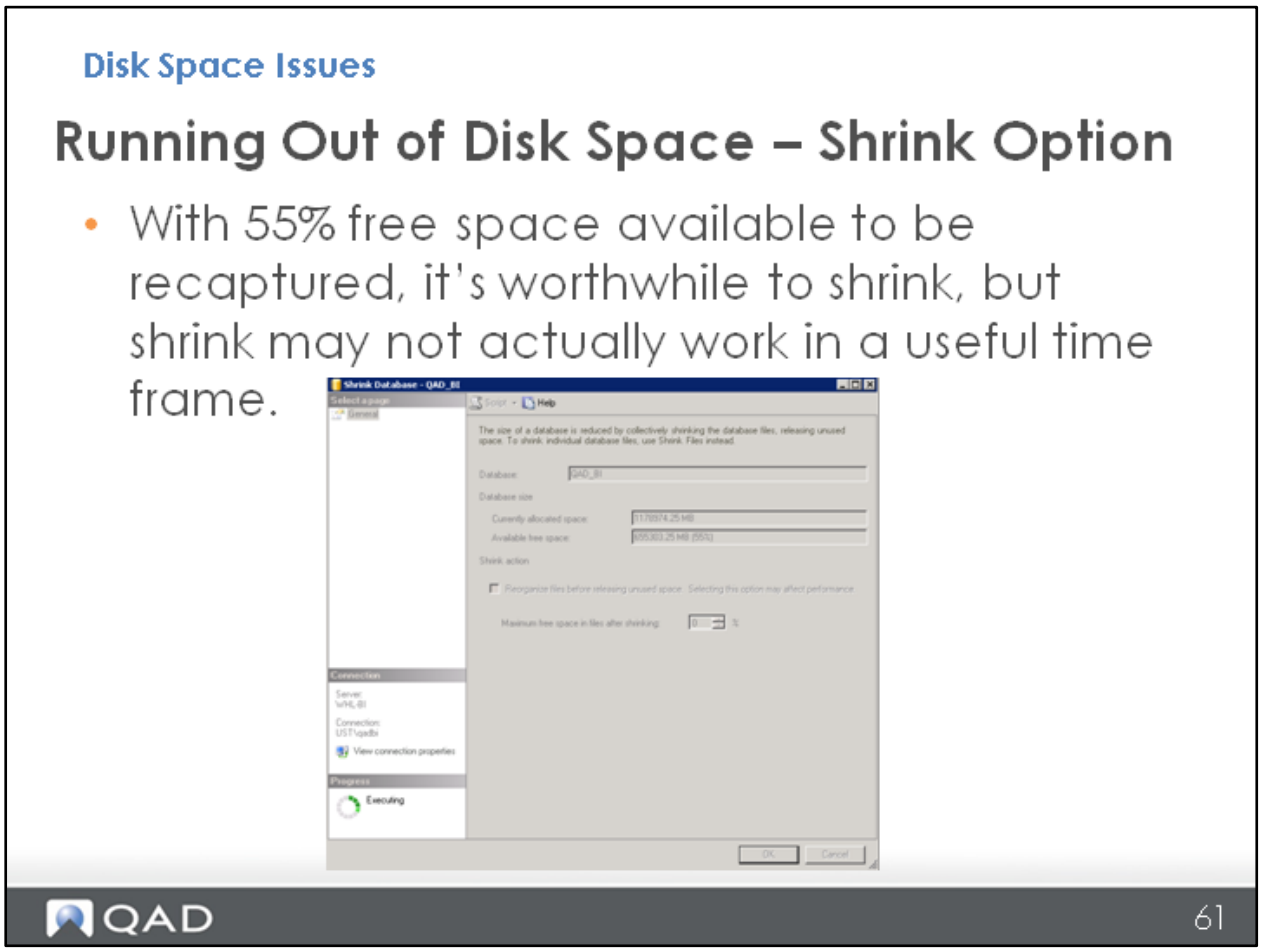
Running Out of Disk Space – Shrink Option

- To shrink a database, right click on the database name, select Tasks / Shrink / Database.



QAD 60

Running Out of Disk Space – Shrink Option

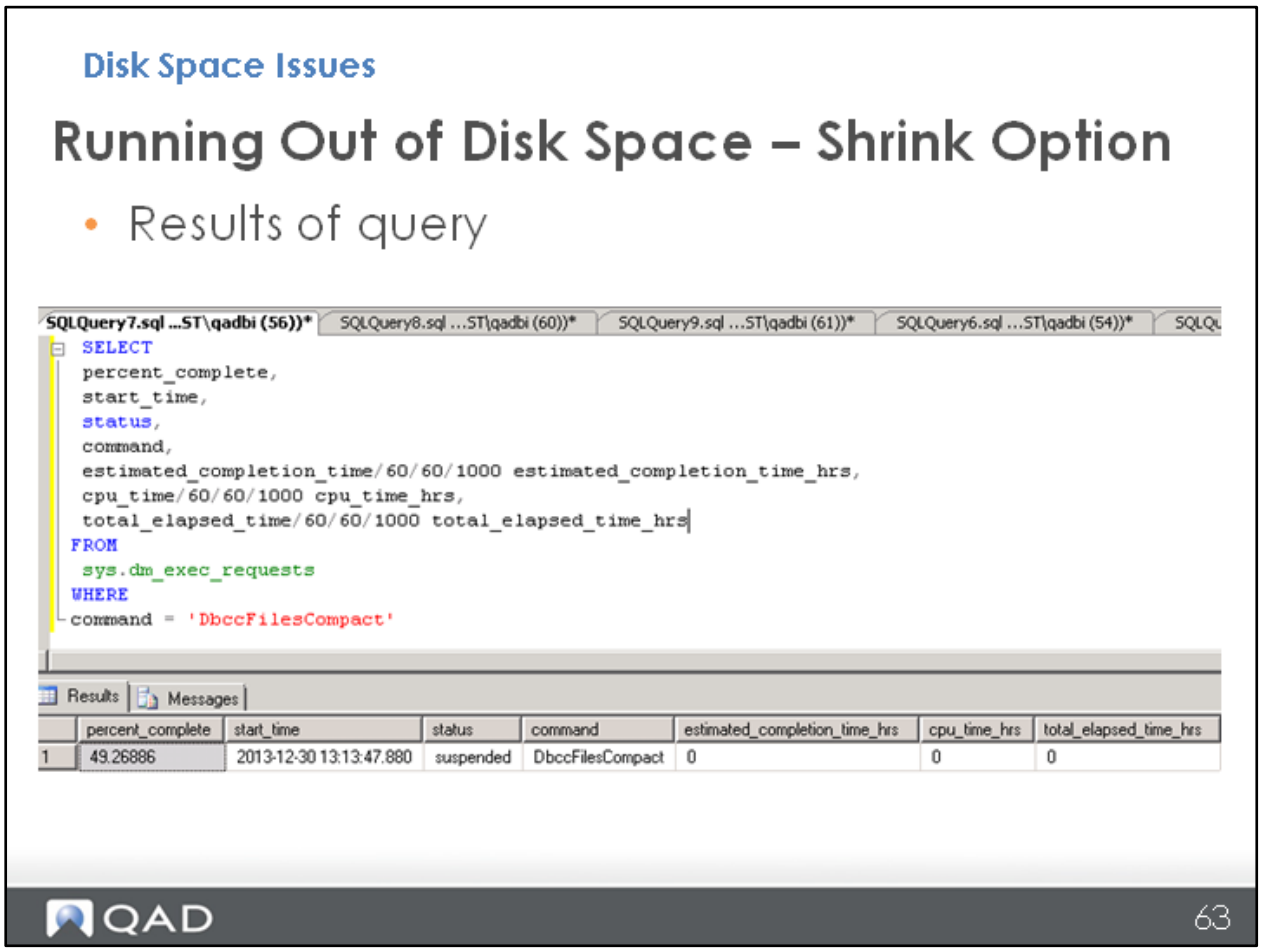


You can see how much space the database is taking up, how much available free space there is and what percent of the total amount of allocated space it is. Click ok to accept default settings to continue.

Running Out of Disk Space – Shrink Option



Running Out of Disk Space – Shrink Option

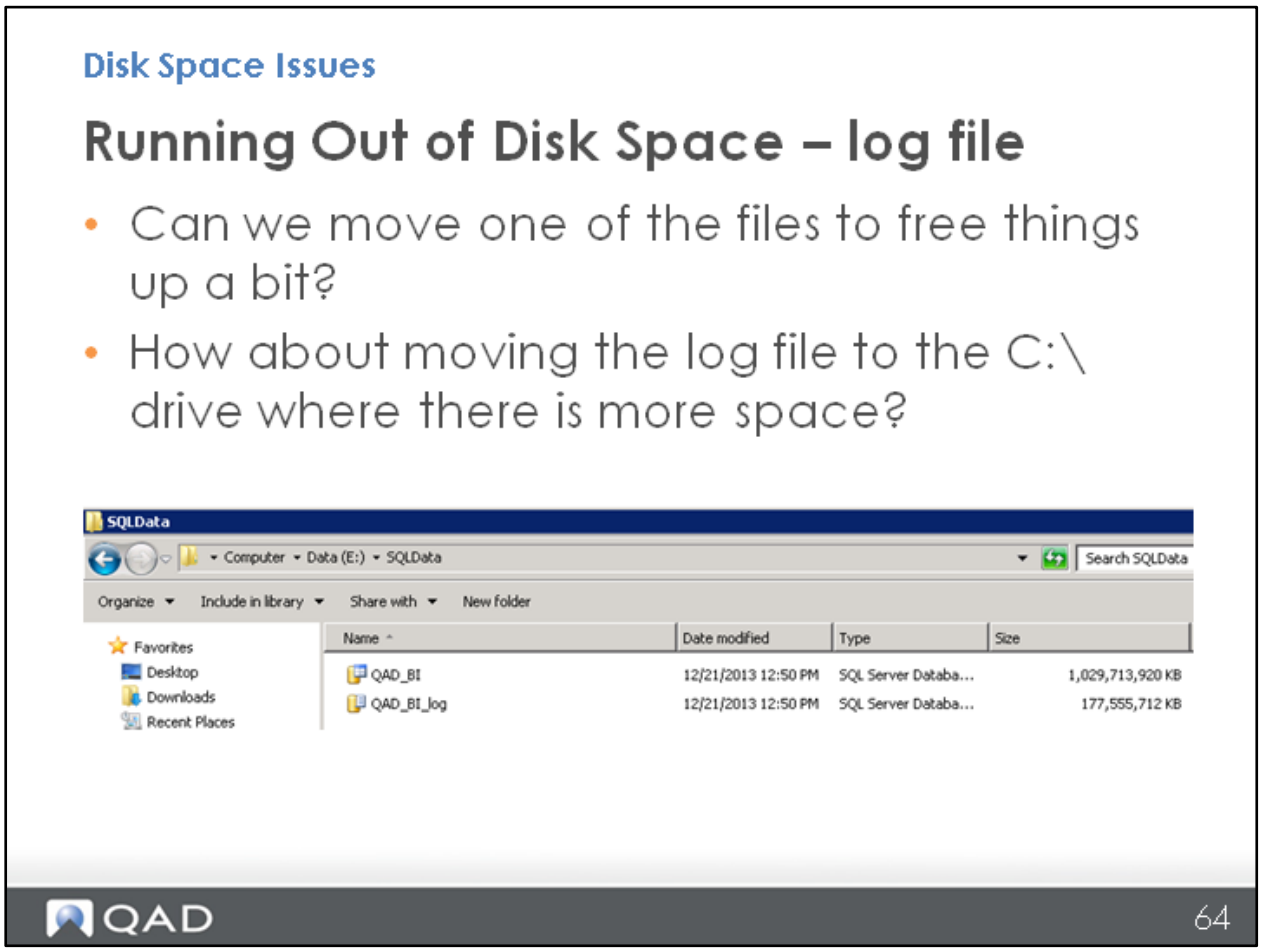


Using this query, you can see how long it will take to finish.

Note that the status frequently will be displayed as suspended. Re-executing the query multiple times may show that the status is running.

These results were produced after a different shrink methodology was run. The actual shrink database results for this customer only completed about 30% of after 4 days, it started to look like it would take months to finish the shrink database option.

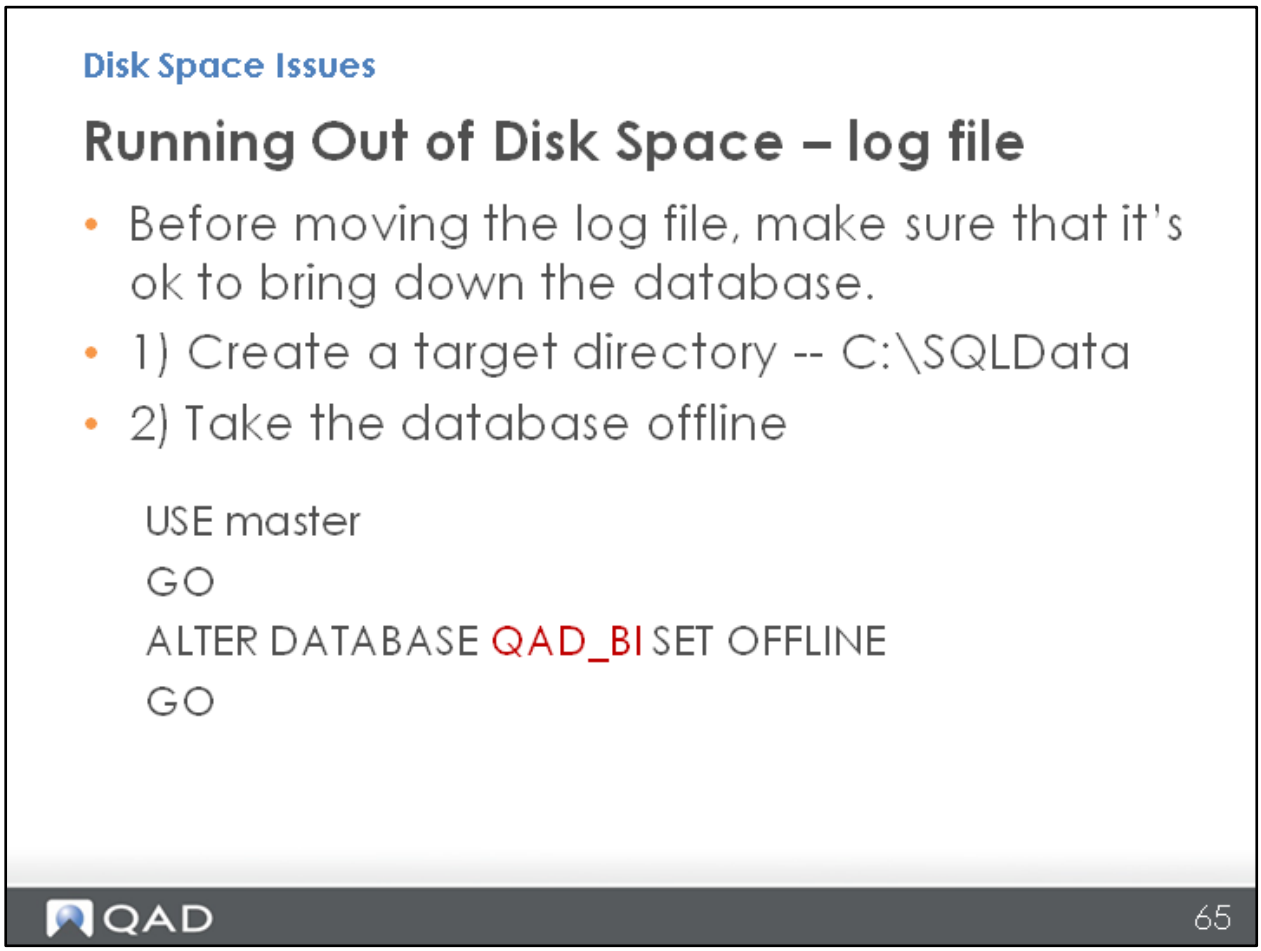
Running Out of Disk Space – log file



Shrink can run for a very long time, potentially even months in this instance. In this case, it's even more difficult because of the lack of disk space.

The actual database is too big to move, but the log file could be moved.

Running Out of Disk Space – log file



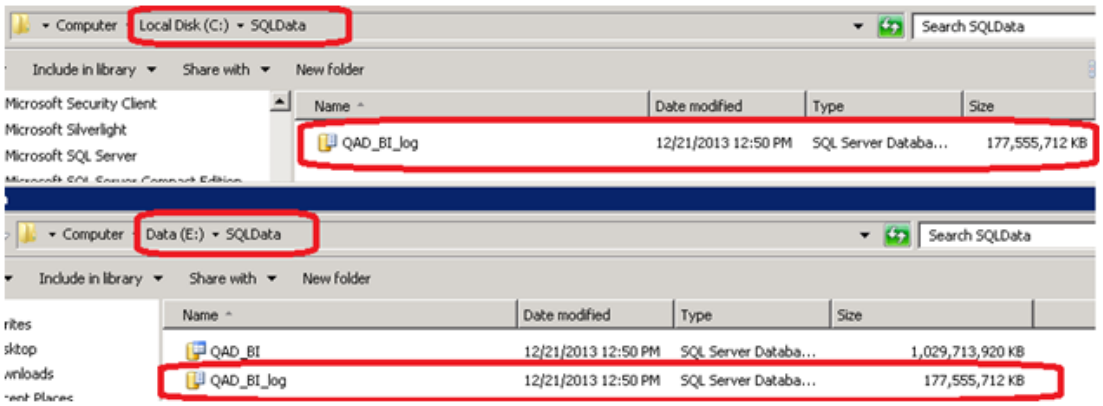
In SSMS, you can also right-click on the database name and select **Tasks** then **Take Offline** option.

Running Out of Disk Space – log file

Disk Space Issues

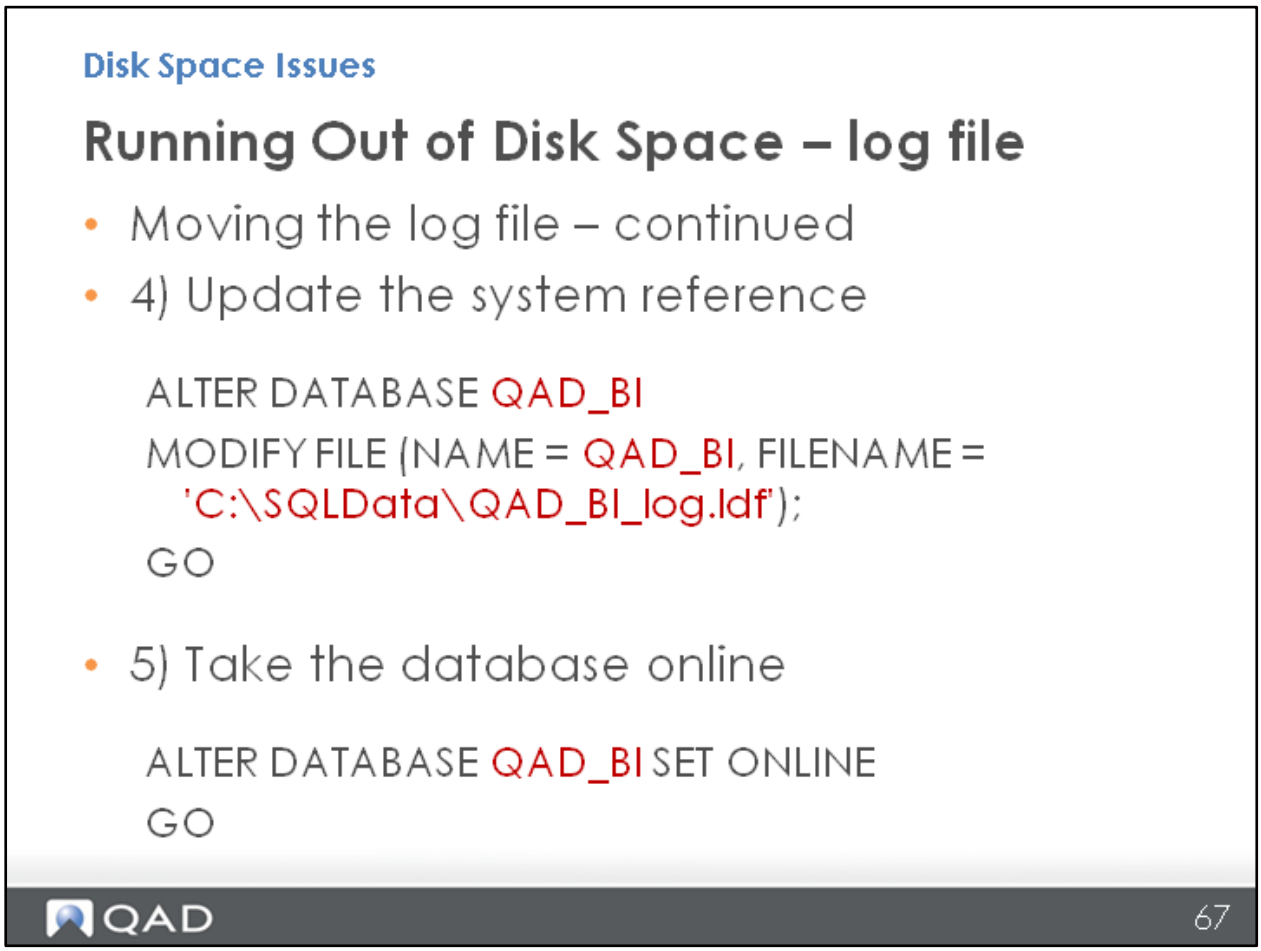
Running Out of Disk Space – log file

- Moving the log file – continued
- 3) In Windows Explorer, use cut and paste



QAD 66

Running Out of Disk Space – log file



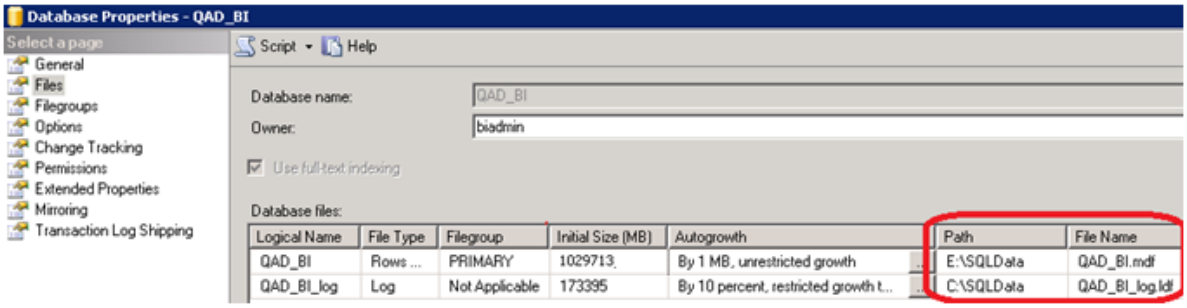
In SSMS, you can also right-click on the database name and select **Tasks** then **Bring Online** option.

Running Out of Disk Space – log file

Disk Space Issues

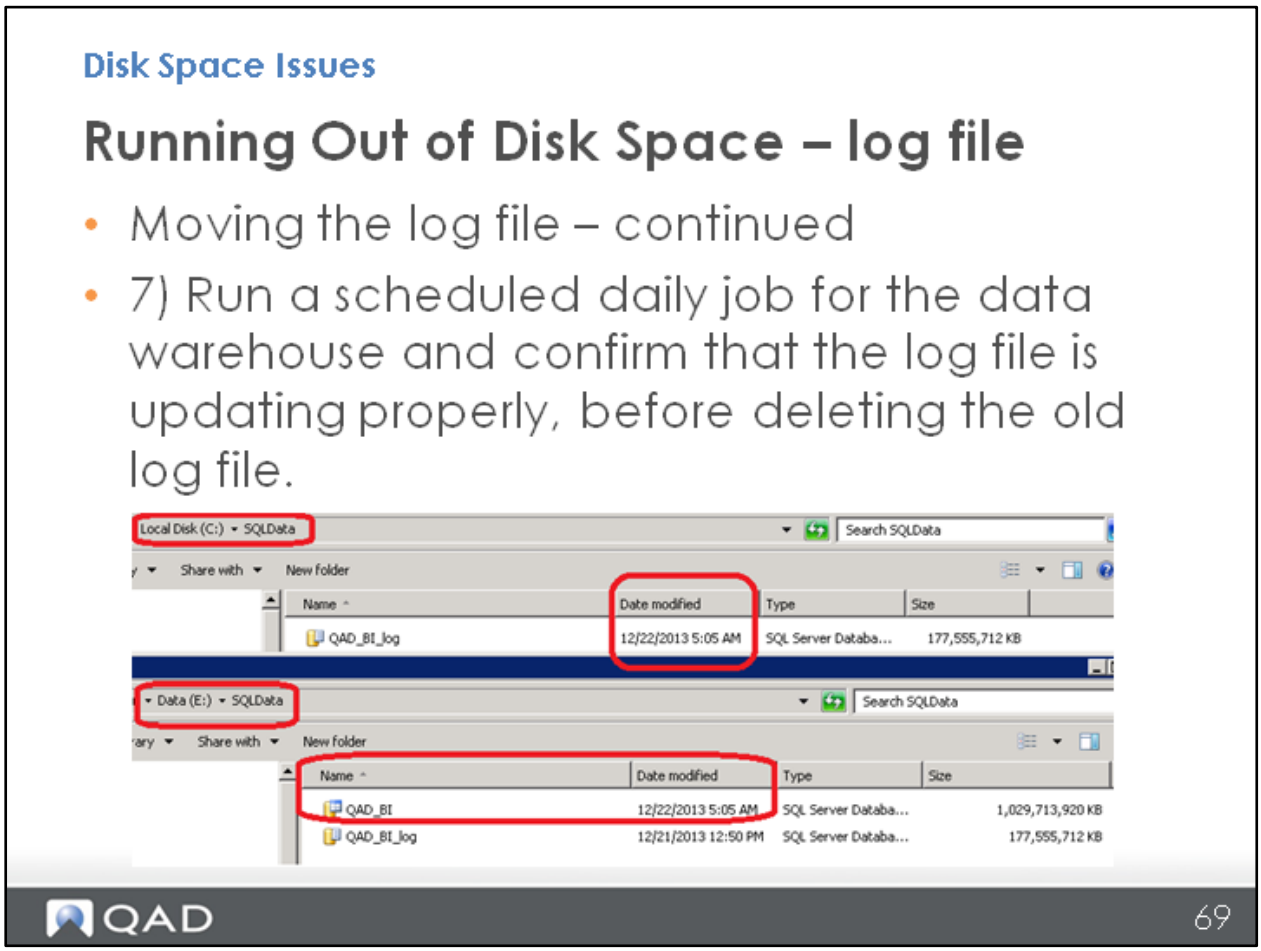
Running Out of Disk Space – log file

- Moving the log file – continued
- 6) Confirm the new Path of the database log file.



QAD 68

Running Out of Disk Space – log file



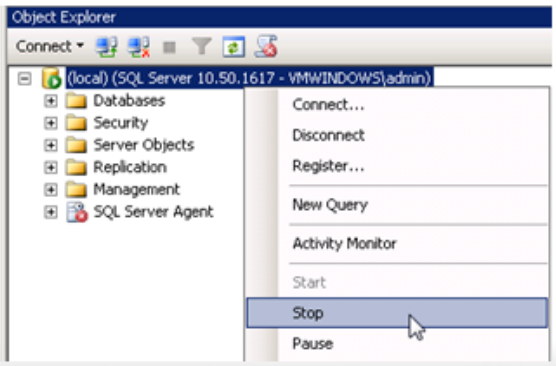
You can see that the date stamp on the log file on the C:\ drive is now the same as the data stamp of the database on the E:\ drive. It is now safe to delete the log file from E:\ drive and free up nearly 178 GB of disk space.

Running Out of Disk Space – log file

Disk Space Issues

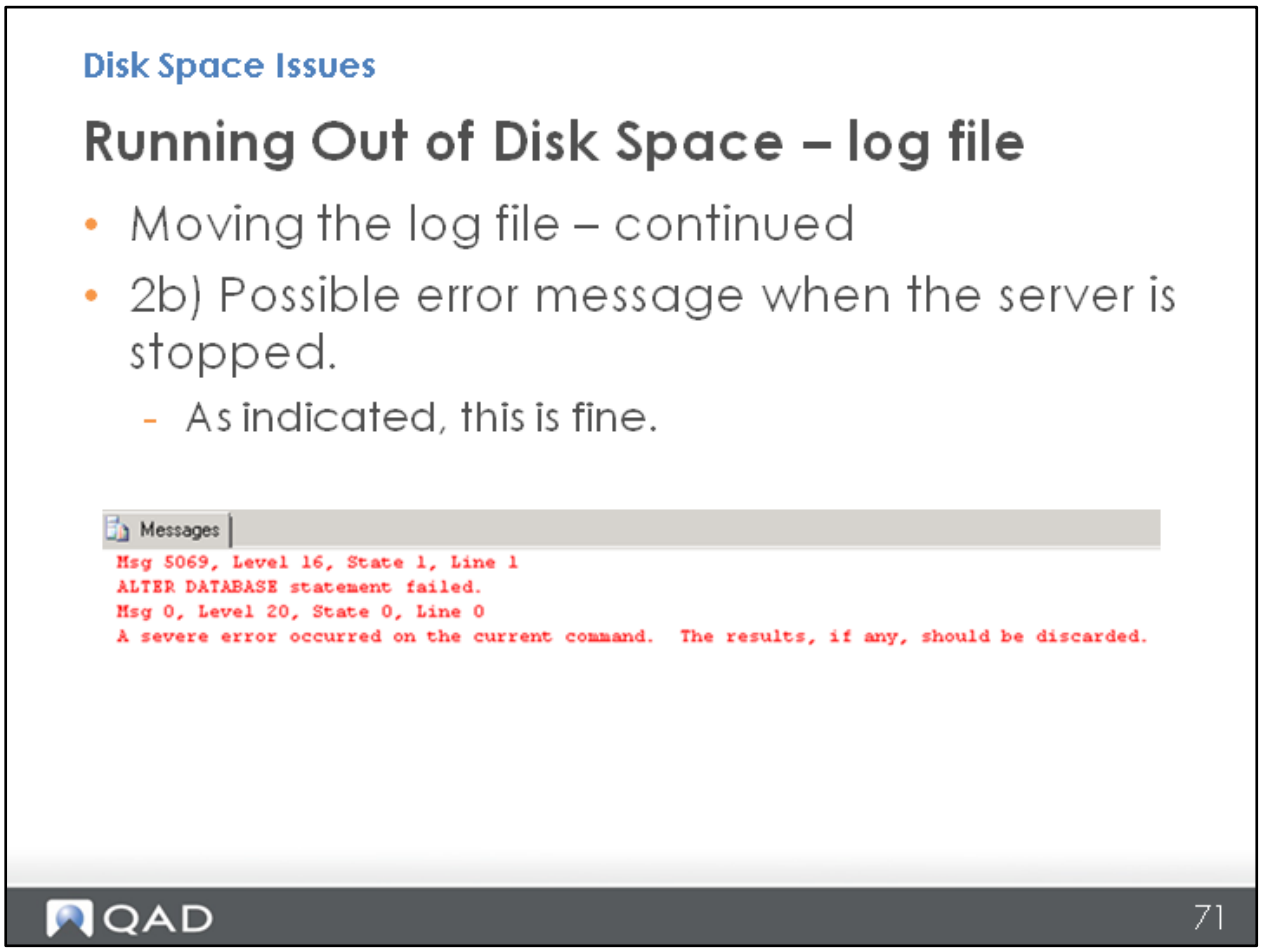
Running Out of Disk Space – log file

- Moving the log file – continued
- 2a) Sometimes, setting the database offline does not work. Instead, the database server must be brought down.
  - Right click on server instance and choose Stop.

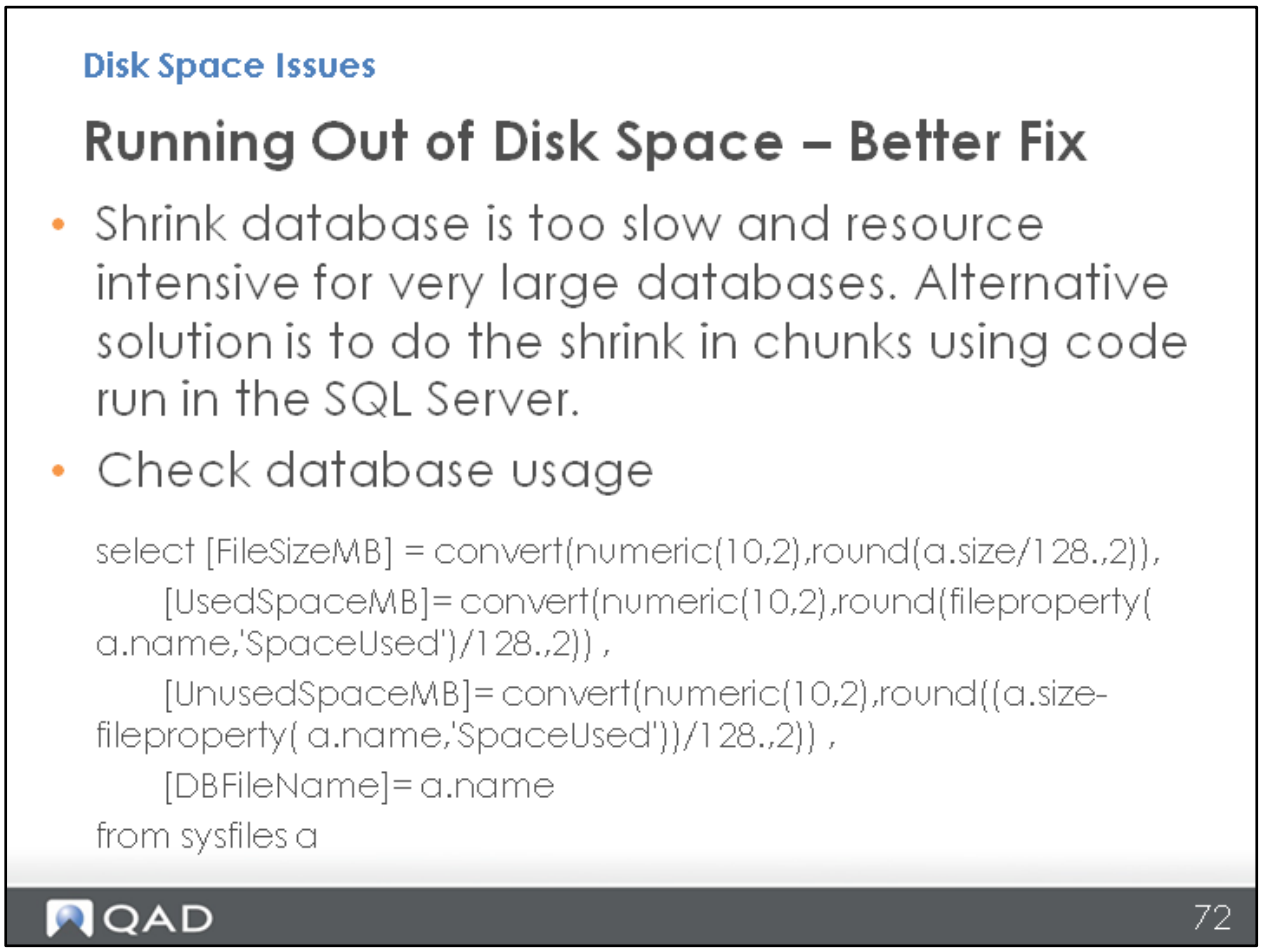


QAD 70

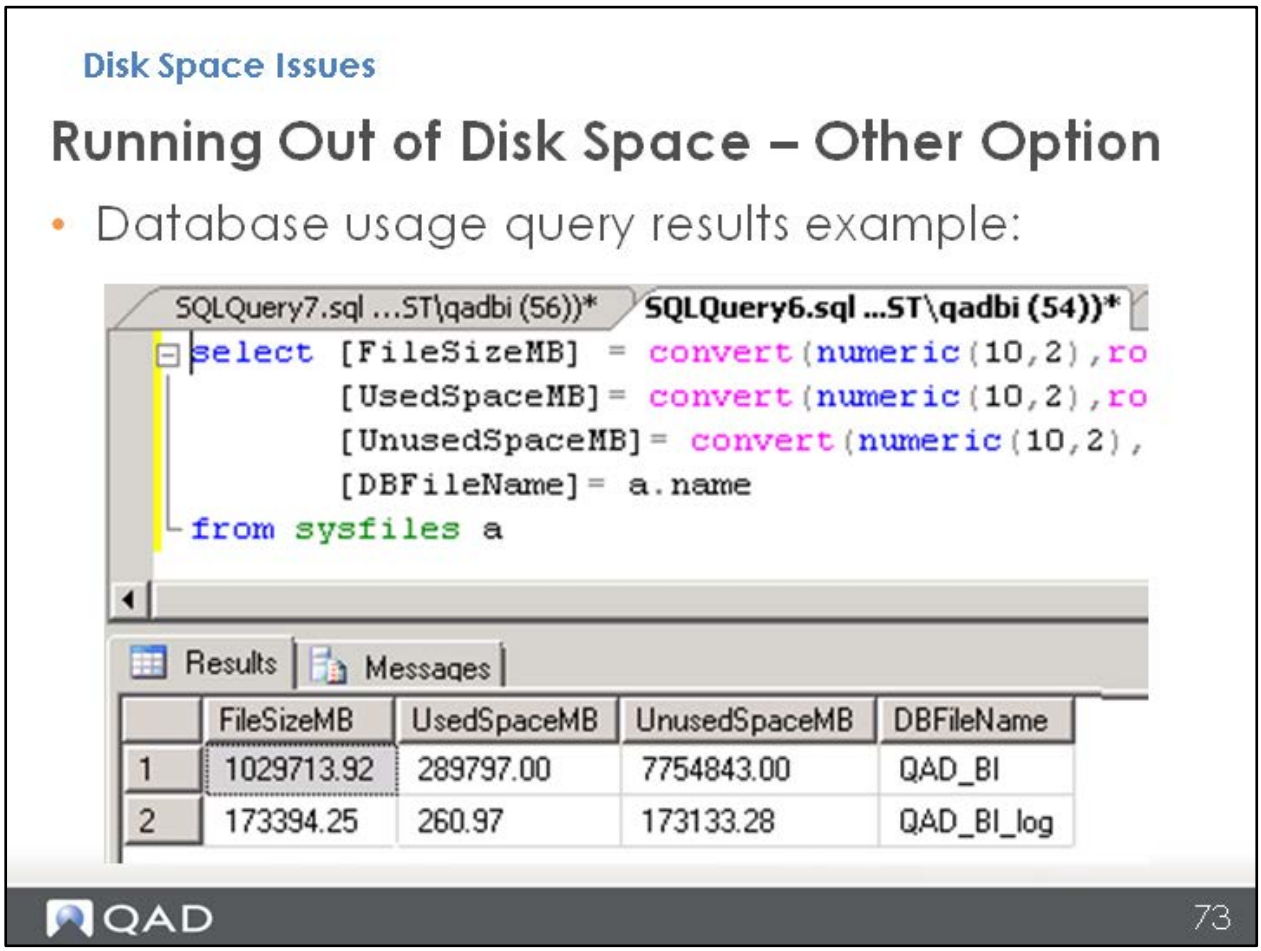
Running Out of Disk Space – log file



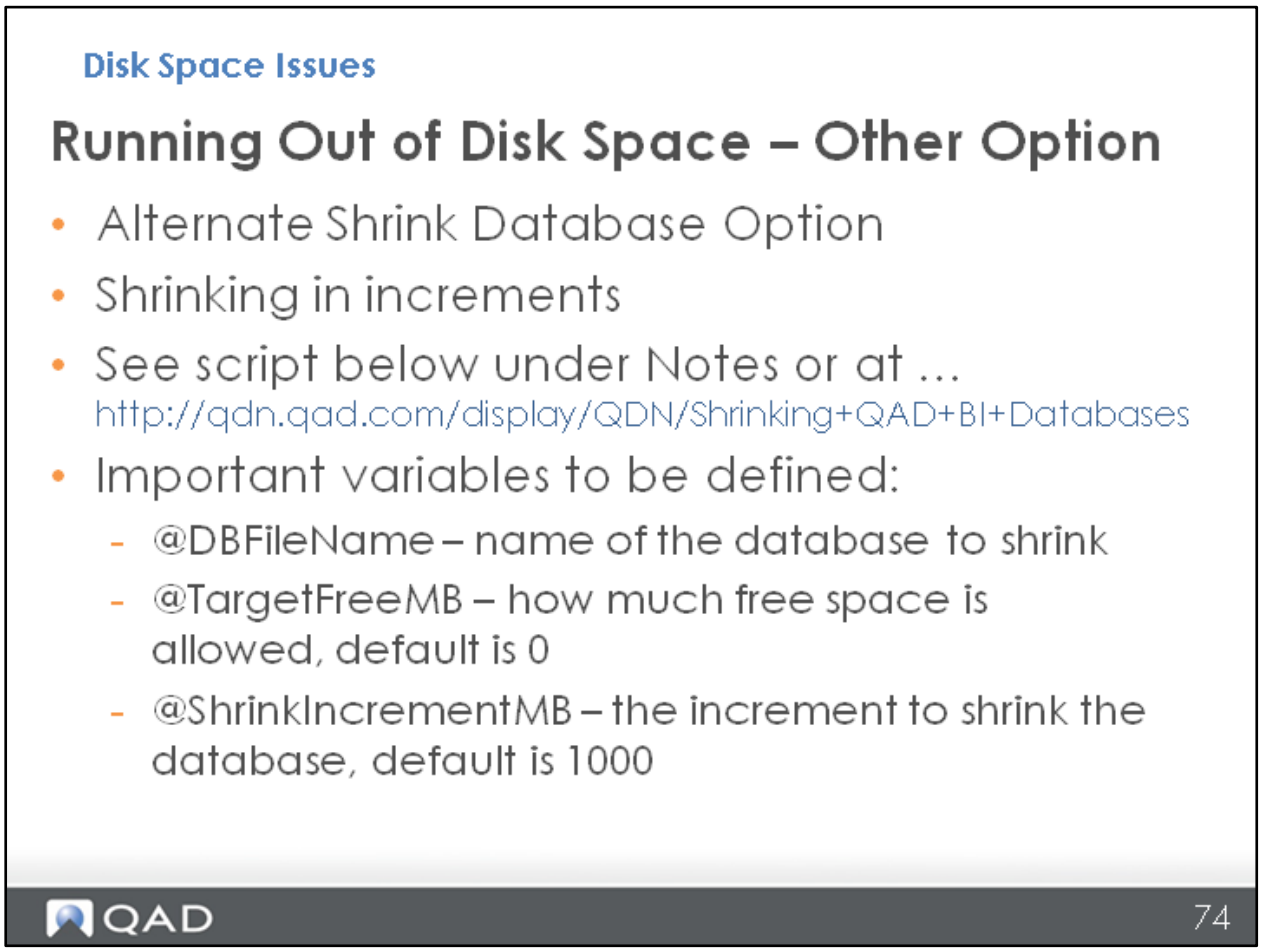
Running Out of Disk Space – Better Fix



Running Out of Disk Space – Other Option



Running Out of Disk Space – Other Option



/*

This script is used to shrink a database file in

increments until it reaches a target free space limit.

Run this script in the database with the file to be shrunk.

1. Set @DBFileName to the name of database file to shrink.

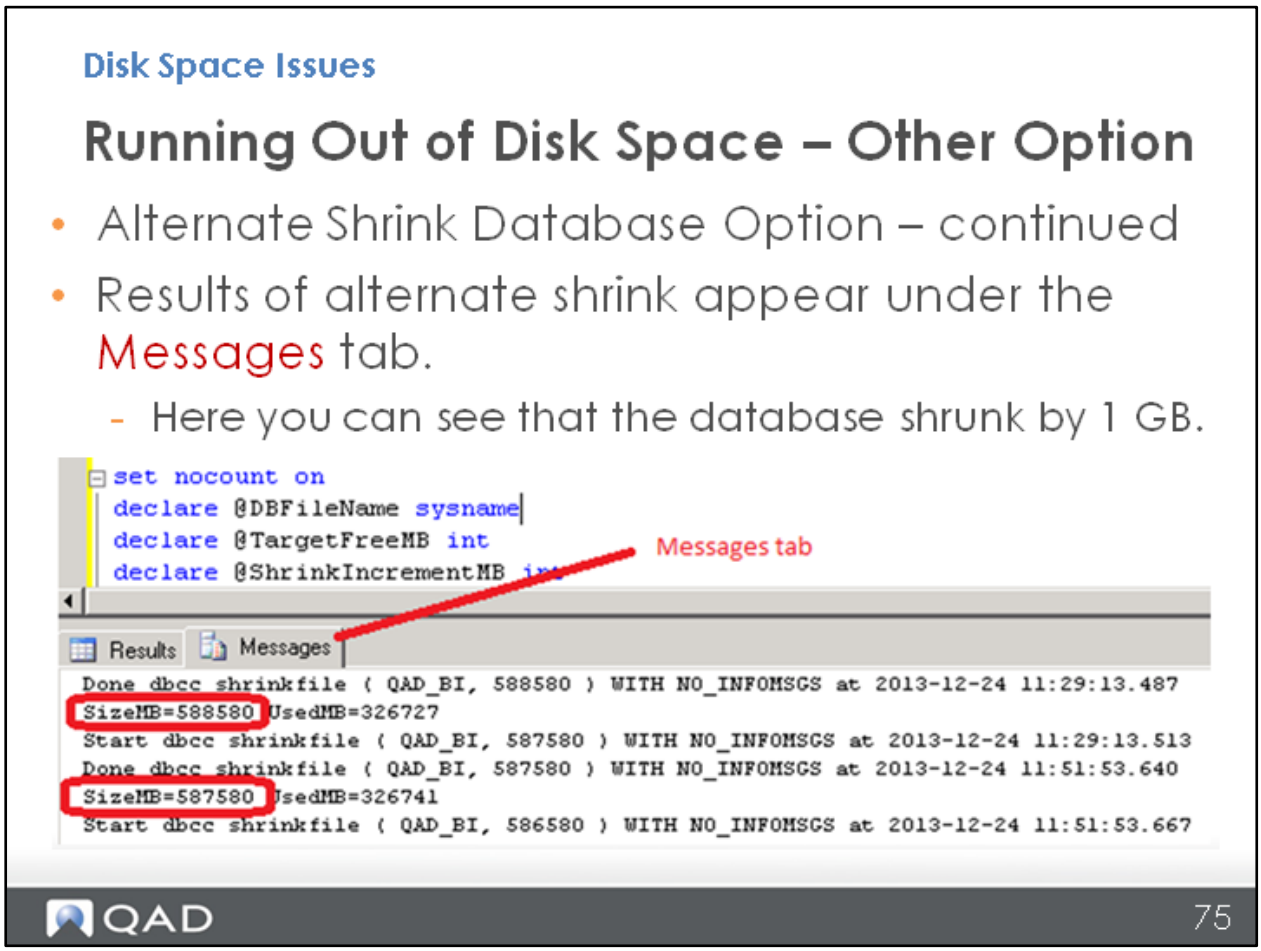
2. Set @TargetFreeMB to the desired file free space in MB after shrink.

3. Set @ShrinkIncrementMB to the increment to shrink file by in MB

4. Run the script

```
*/
set nocount on
declare @DBFileName sysname
declare @TargetFreeMB int
declare @ShrinkIncrementMB int
-- Set Name of Database file to shrink
set @DBFileName = 'QAD_BI'
-- Set Desired file free space in MB after shrink
set @TargetFreeMB = 0
-- Set Increment to shrink file by in MB
set @ShrinkIncrementMB= 1000
-- Show Size, Space Used, Unused Space, and Name of all database files select
[FileSizeMB] = convert(numeric(10,2),round(a.size/128.,2)),
       [UsedSpaceMB]= convert(numeric(10,2),round(fileproperty(
a.name,'SpaceUsed')/128.,2)) ,
       [UnusedSpaceMB]= convert(numeric(10,2),round((a.size-fileproperty(
a.name,'SpaceUsed'))/128.,2)) ,
       [DBFileName]= a.name
from sysfiles a
declare @sql varchar(8000)
declare @SizeMB float
declare @UsedMB float
-- Get current file size in MB
select @SizeMB = size/128. from sysfiles where name = @DBFileName
-- Get current space used in MB
select @UsedMB = fileproperty( @DBFileName,'SpaceUsed')/128.0
-- Loop until file at desired size
Print 'SizeMB = '+convert(varchar(40),@SizeMB)+' UsedMB =
'+convert(varchar(40),@UsedMB)+' TargetFreeMB = ' +
convert(varchar(40),@TargetFreeMB) + ' ShrinkIncrementMB = '+
convert(varchar(40),@ShrinkIncrementMB)
while  @SizeMB > @UsedMB+@TargetFreeMB+@ShrinkIncrementMB
begin
 set @sql = 'dbcc shrinkfile ( '+@DBFileName+', ' +
convert(varchar(20),convert(numeric(15,0),@SizeMB-@ShrinkIncrementMB))+' ) WITH
NO_INFOMSGS'
 --set @sql = 'dbcc shrinkfile ( "QAD_BI", 1006580 ) WITH NO_INFOMSGS' print
'Start ' + @sql + ' at ' + convert(varchar(30),getdate(),121)
 exec ( @sql )
 print 'Done ' + @sql + ' at '+convert(varchar(30),getdate(),121)
 -- Get current file size in MB
 select @SizeMB = size/128. from sysfiles where name = @DBFileName
 -- Get current space used in MB
 select @UsedMB = fileproperty( @DBFileName,'SpaceUsed')/128.0
 print 'SizeMB=' + convert(varchar(20),@SizeMB) + ' UsedMB=' +
convert(varchar(20),@UsedMB)
end
select [EndFileSize] = @SizeMB, [EndUsedSpace] = @UsedMB, [DBFileName] =
@DBFileName
```

Running Out of Disk Space – Other Option



It sometimes takes a little while for things to get started with the messages.

We've tried shrinking by 10 GB, but it was very slow.

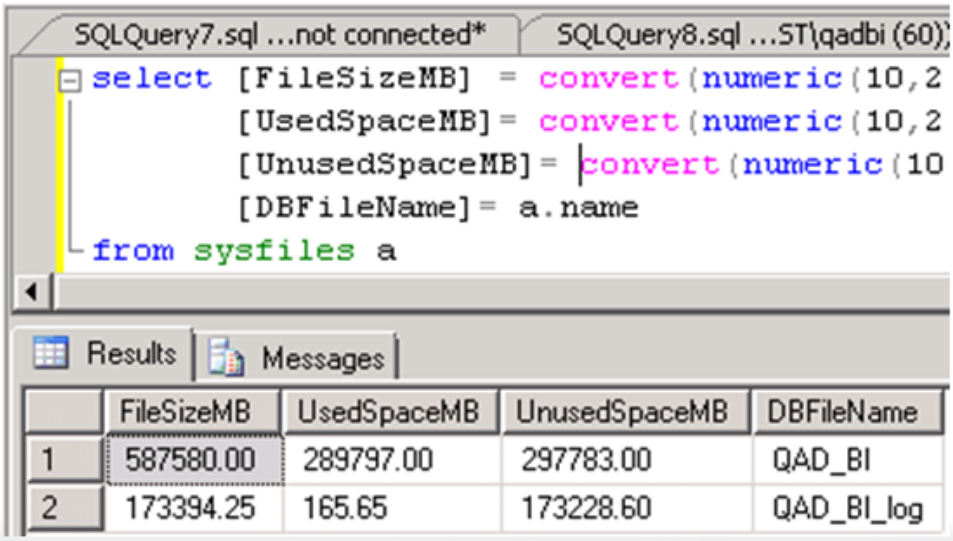
You can stop the query at any time by cancelling in the SQL Server window.

Running Out of Space and Shrink Option.

Disk Space Issues

Running Out of Space and Shrink Option.

- Database usage query results example, after alternate shrink option has run for a while:



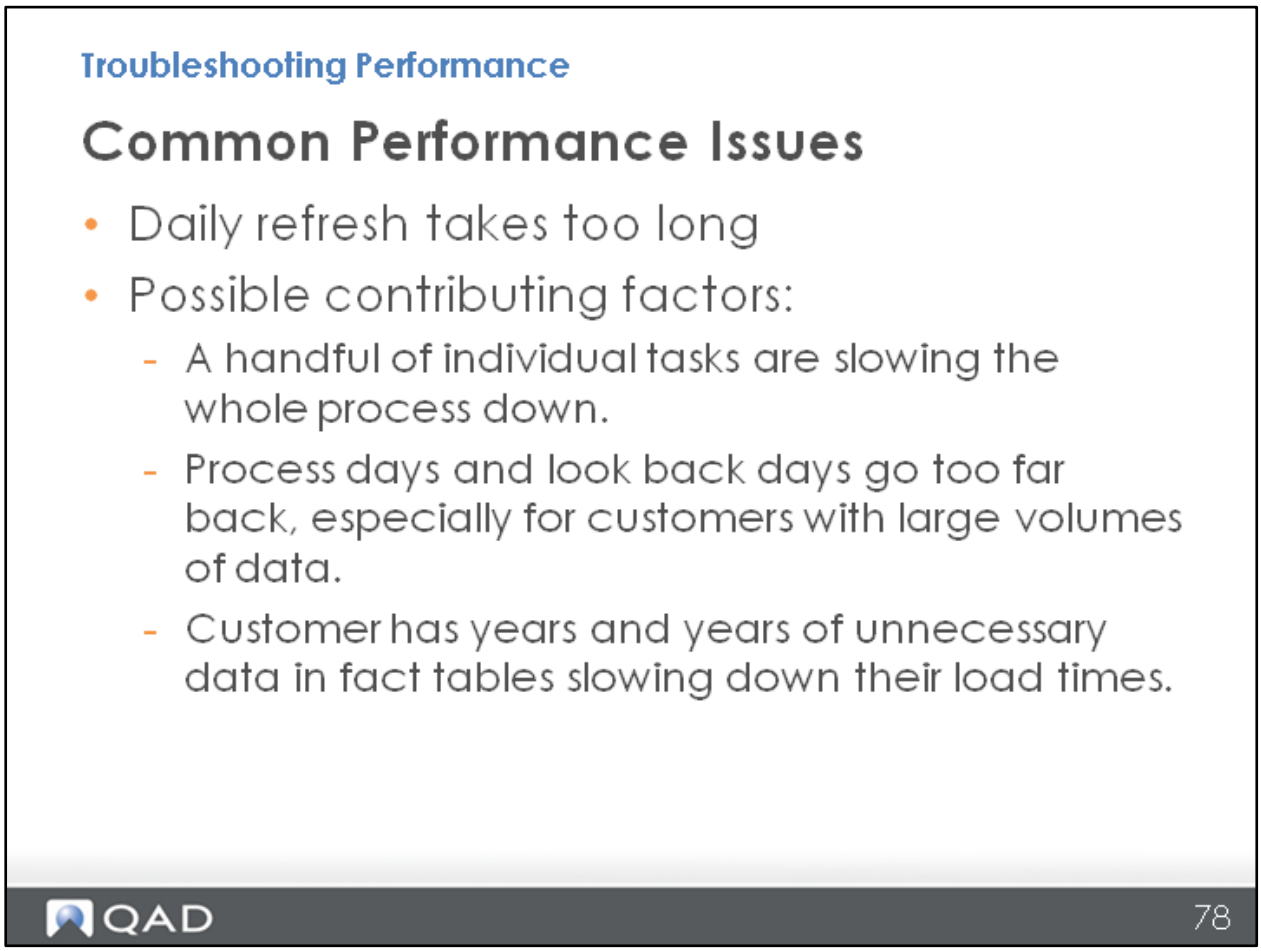
	FileSizeMB	UsedSpaceMB	UnusedSpaceMB	DBFileName
1	587580.00	289797.00	297783.00	QAD_BI
2	173394.25	165.65	173228.60	QAD_BI_log

QAD 76

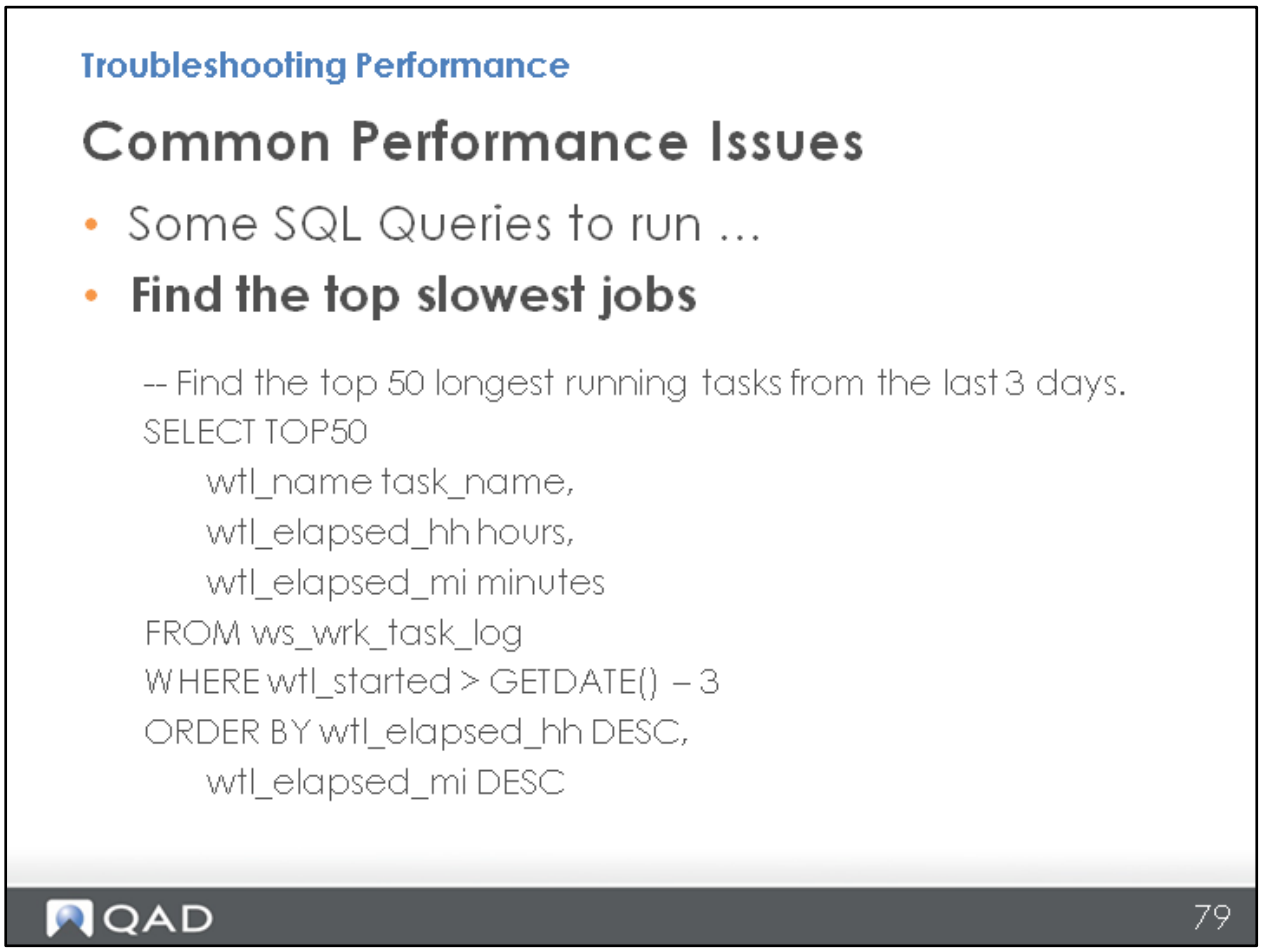
Troubleshooting Performance



Common Performance Issues

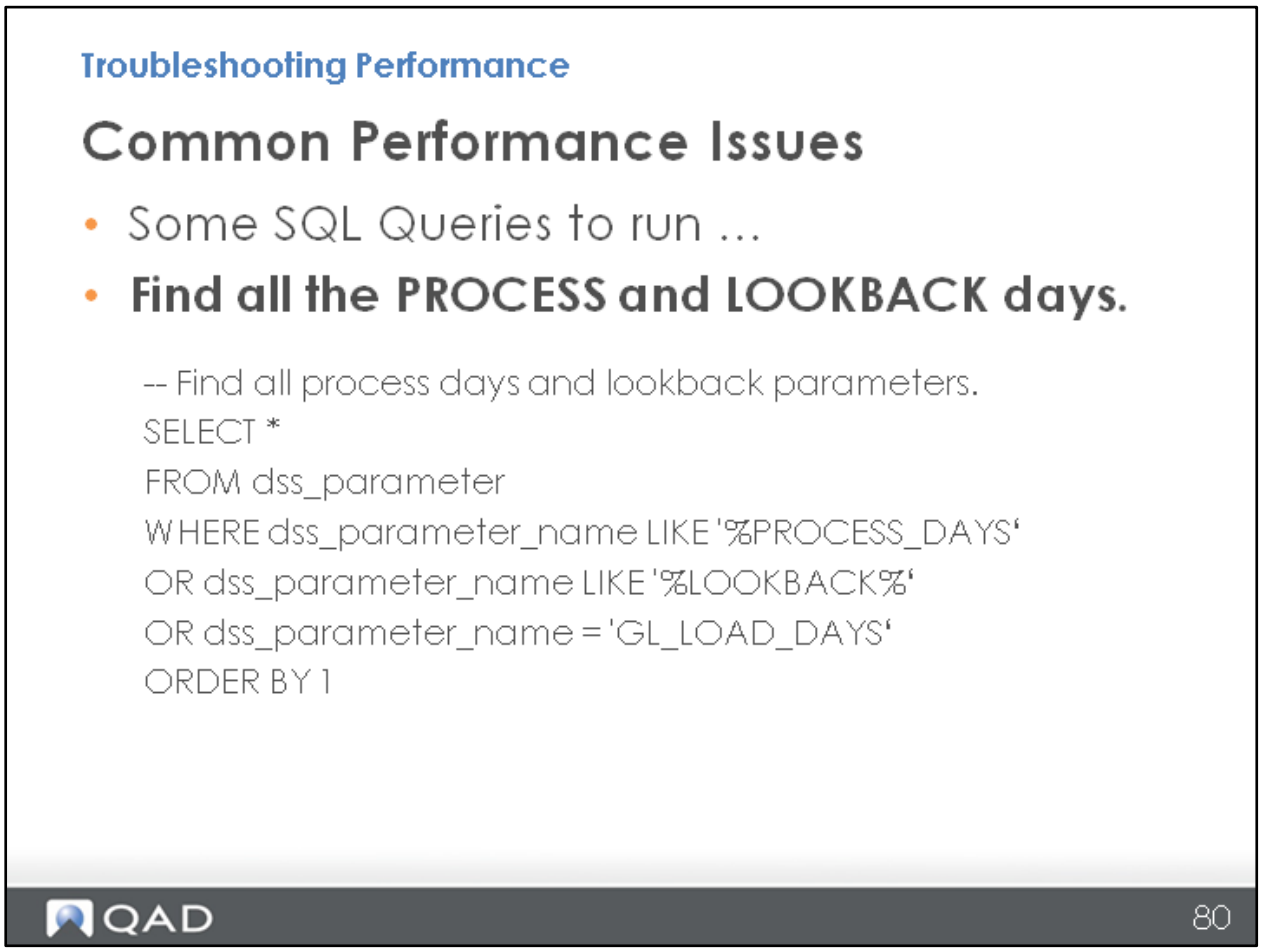


Common Performance Issues



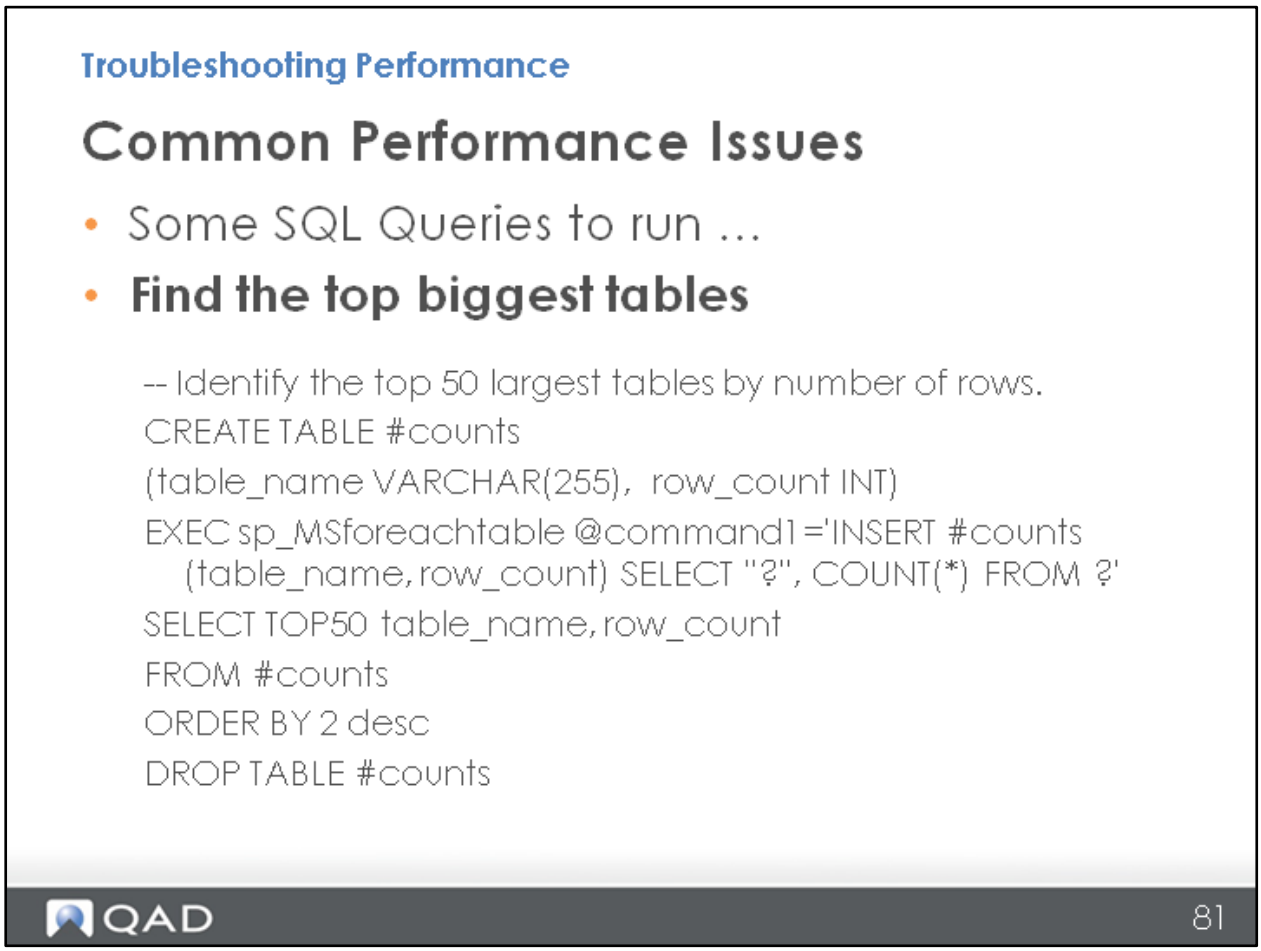
This query checks the scheduler and returns the slowest running tasks that it finds. Change the number of days from 3 to a bigger number if there have been no daily runs for a while. The results of the query provides a list of our biggest bottlenecks in run times for us to focus on.

Common Performance Issues



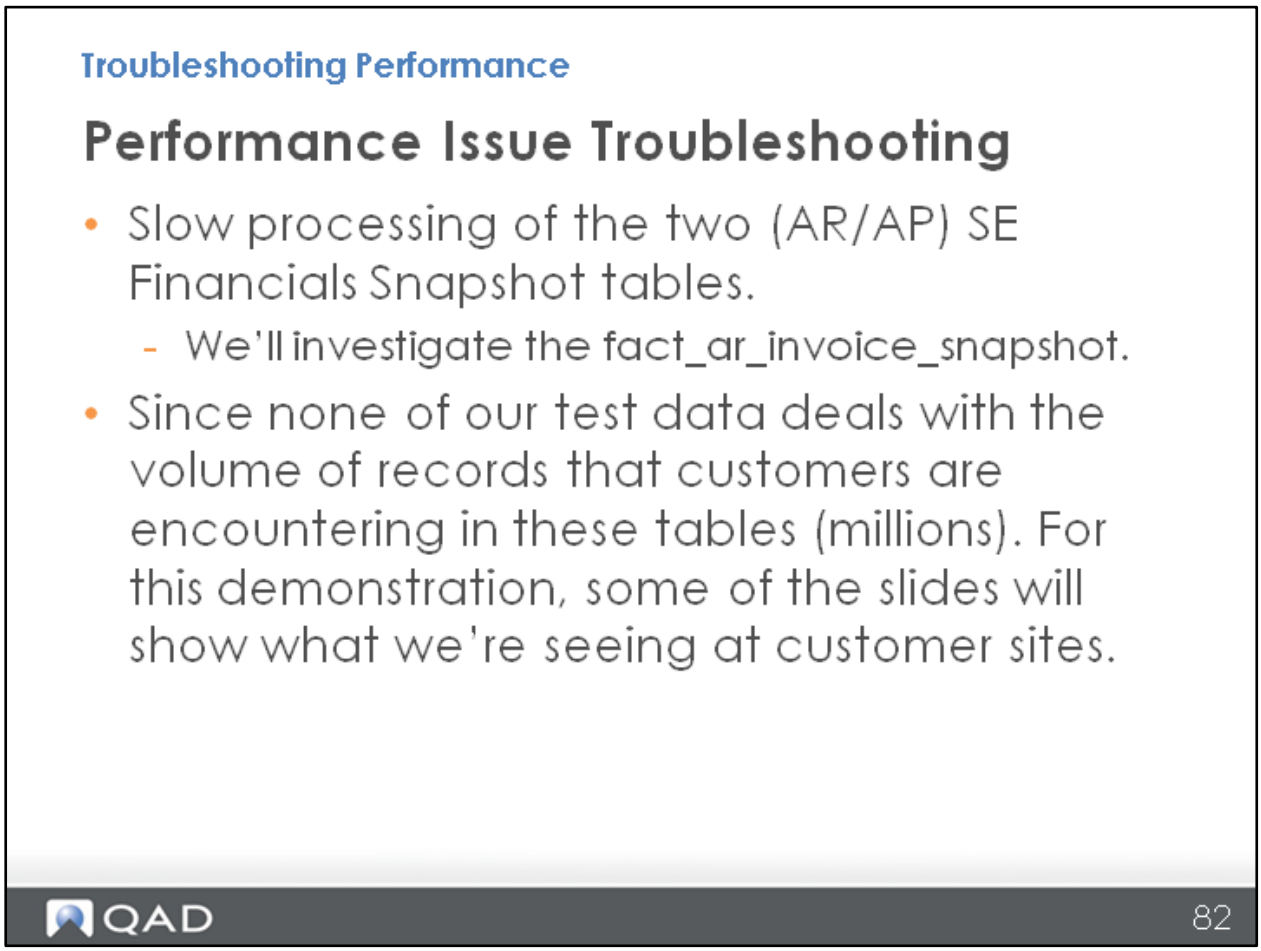
The results of this query gives us a quick view at what all the process days and lookback parameters are set to for each module that is installed. If there are any really large numbers that are not appropriate, we'd quickly see them in these results (instead of looking through the parameter list).

Common Performance Issues

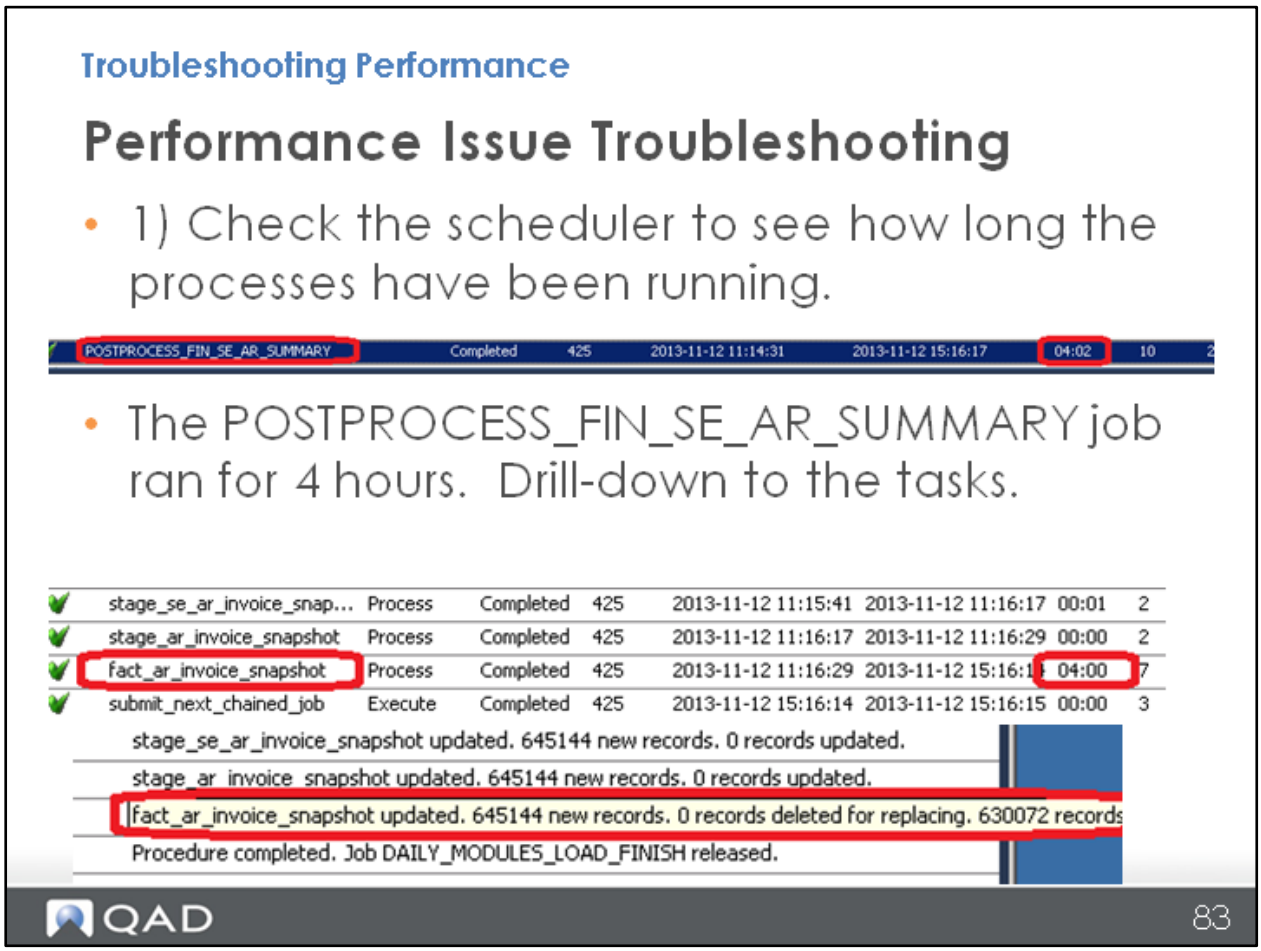


This query reveals which fact, dim and perm tables contain the most rows. If we're seeing extreme slowness on some of these tables, the reason may be that we have more years worth of data than the users need, or it may just be appropriate. Either way, if we're seeing slow jobs tied to tables with 50 million rows, we then already have an idea of why that task may be running slow.

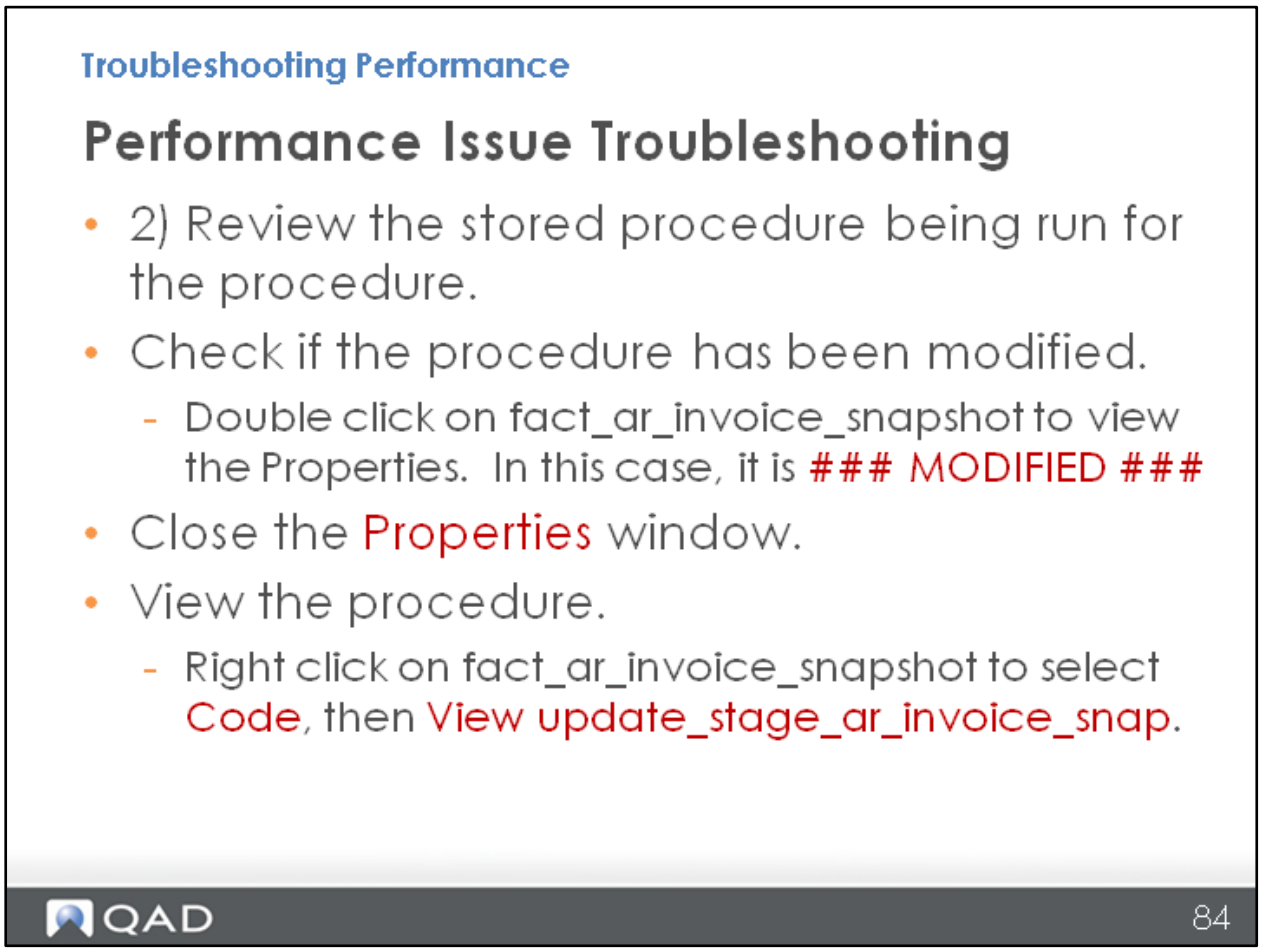
Performance Issue Troubleshooting



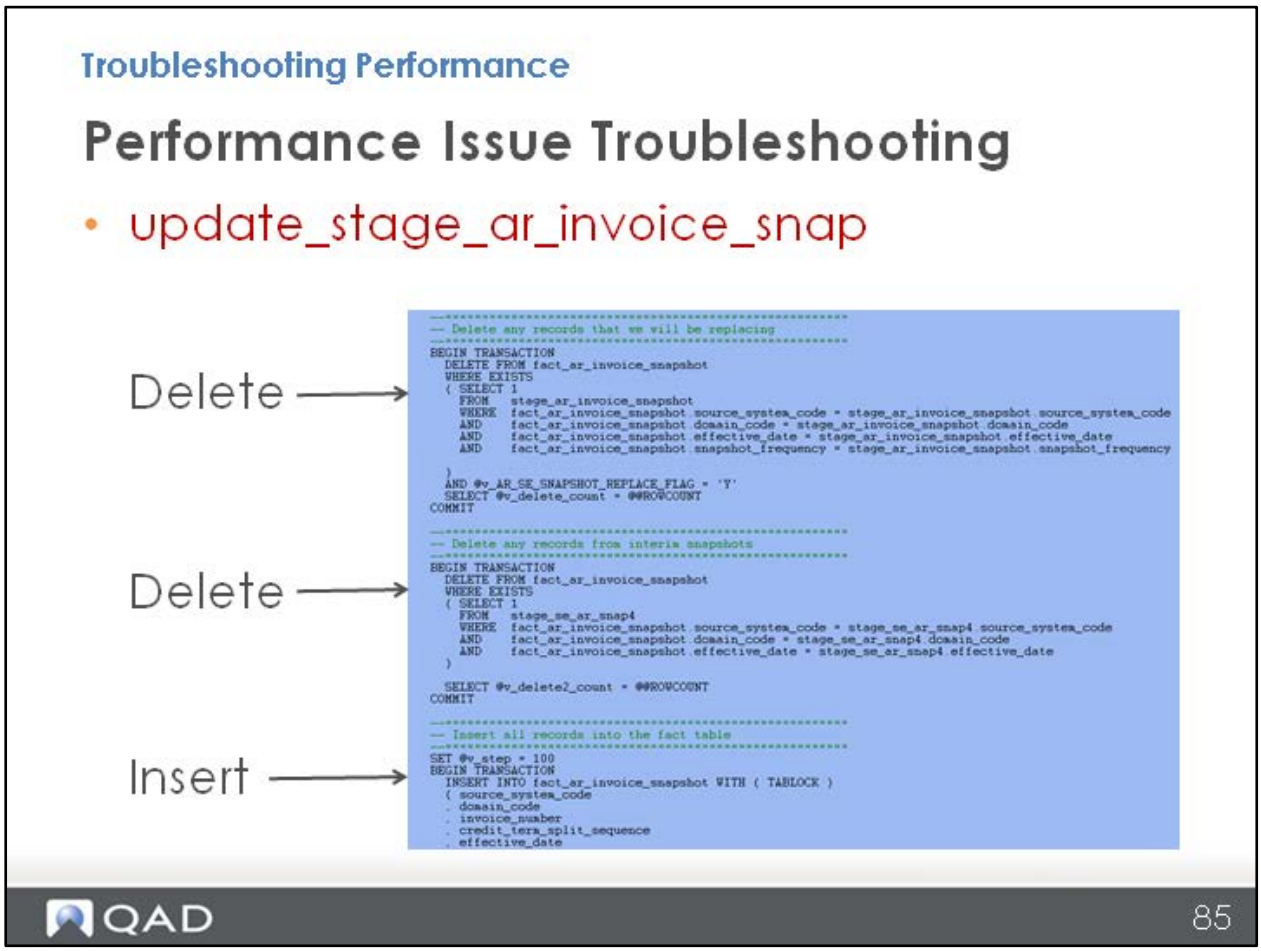
Performance Issue Troubleshooting



Performance Issue Troubleshooting

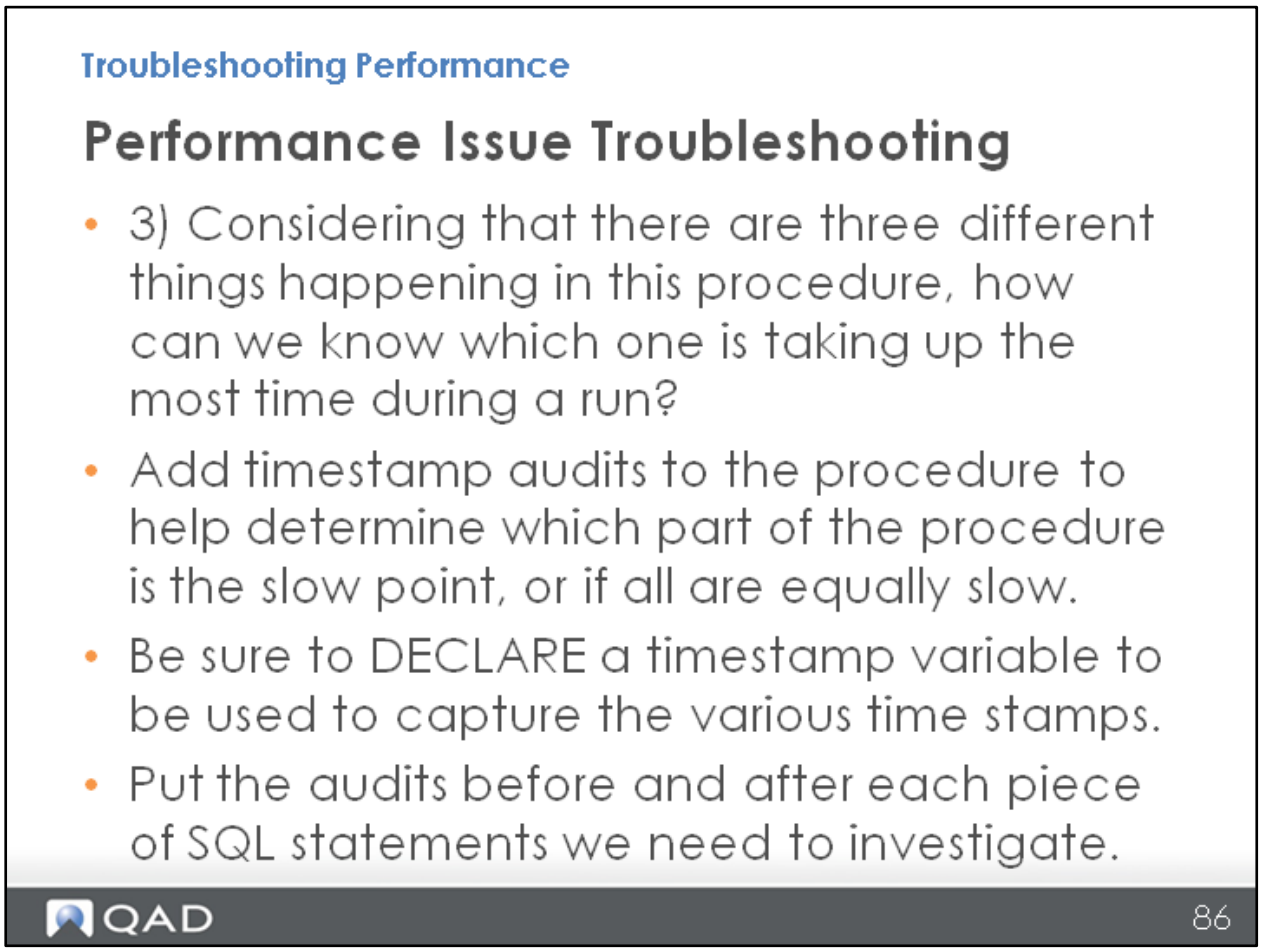


Performance Issue Troubleshooting

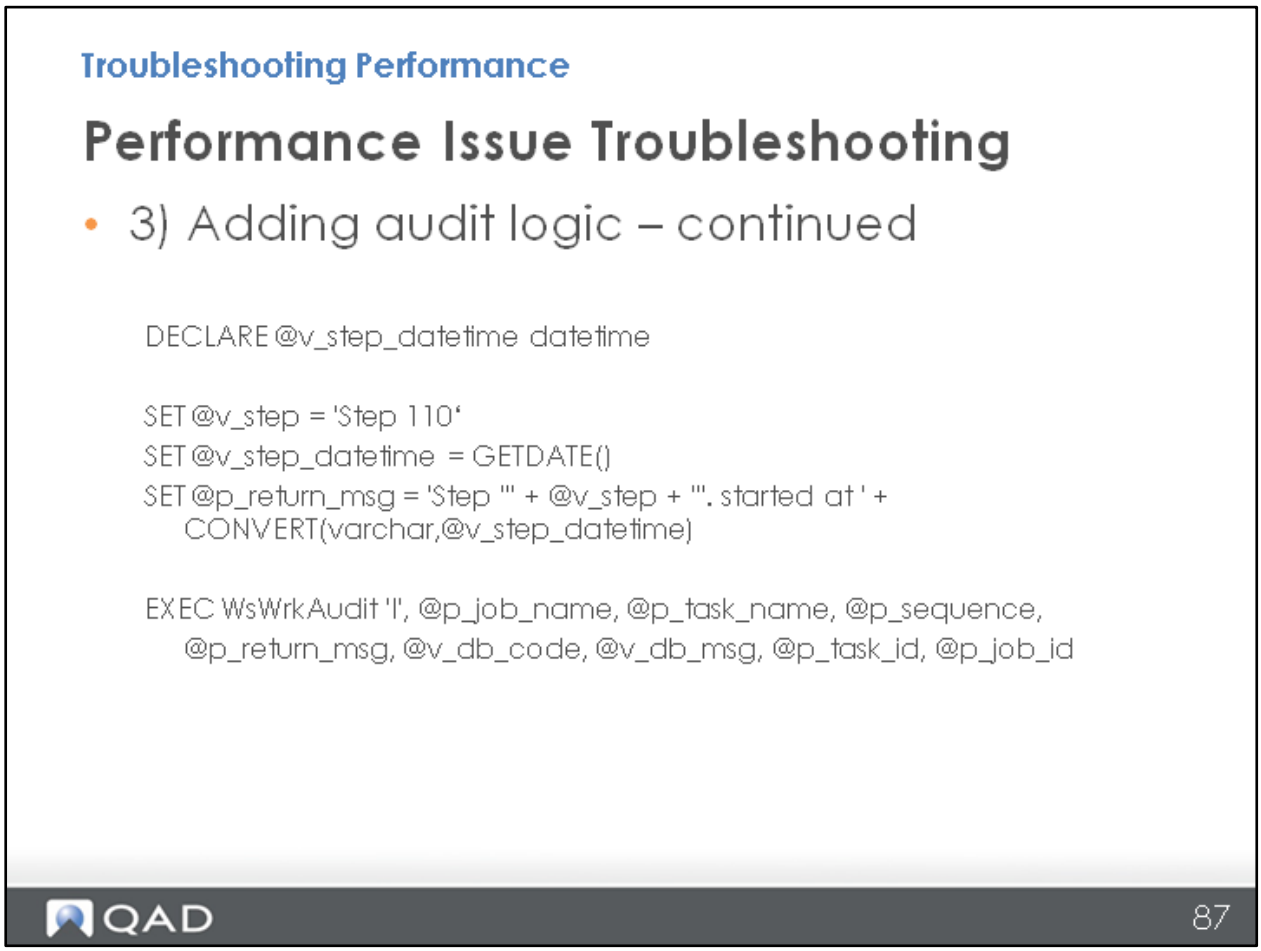


While looking at the procedure, it's clear that it does more than just insert data into the snapshot tables. First it does two deletes before inserting the data into the snapshot tables. It's basically removing data already in the fact table that exists also in two staging tables prior to inserting the data into the fact tables from one of those staging tables.

Performance Issue Troubleshooting



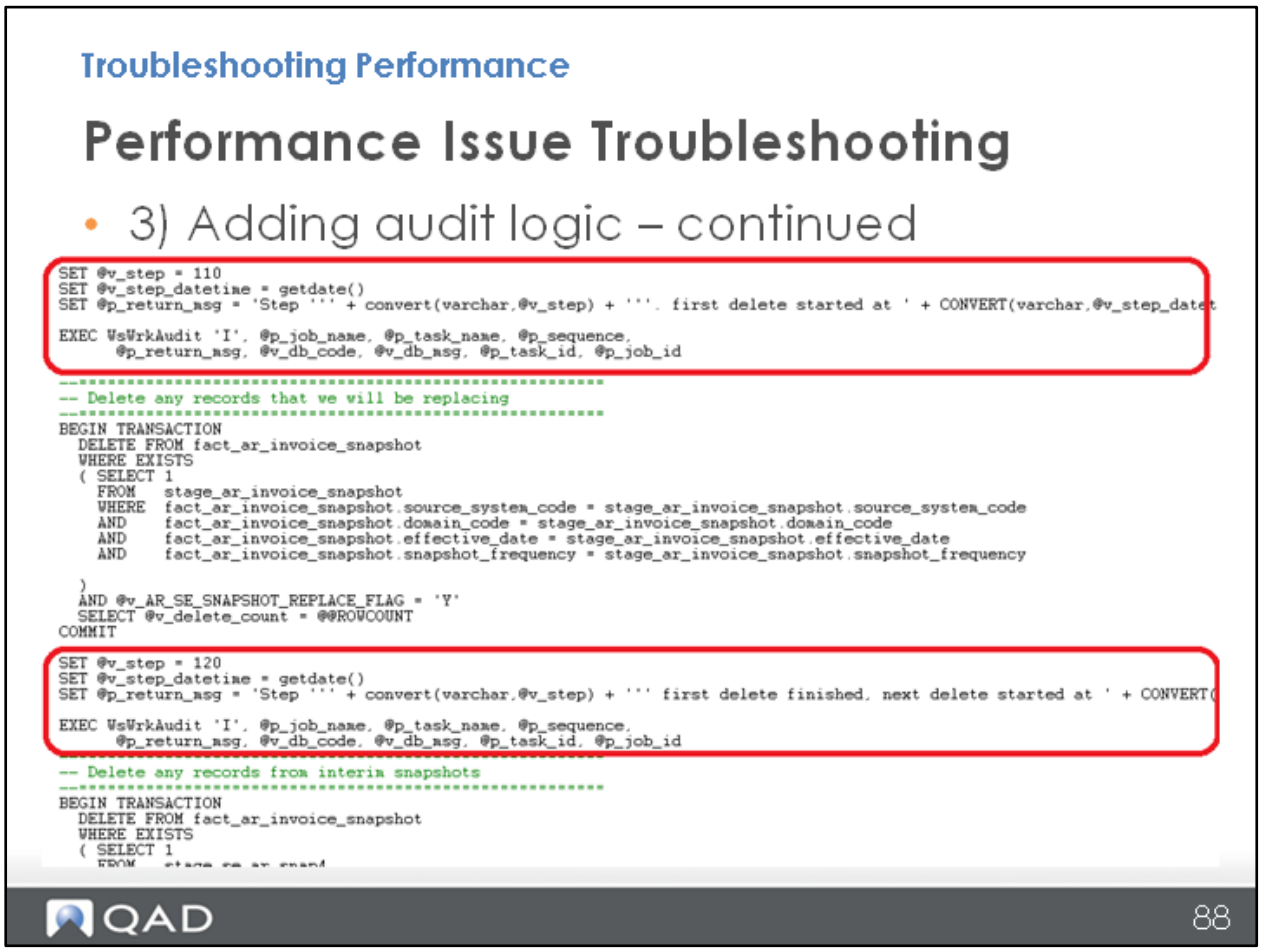
Performance Issue Troubleshooting



Declare the time variable near where all the other DECLARE columns occurred.

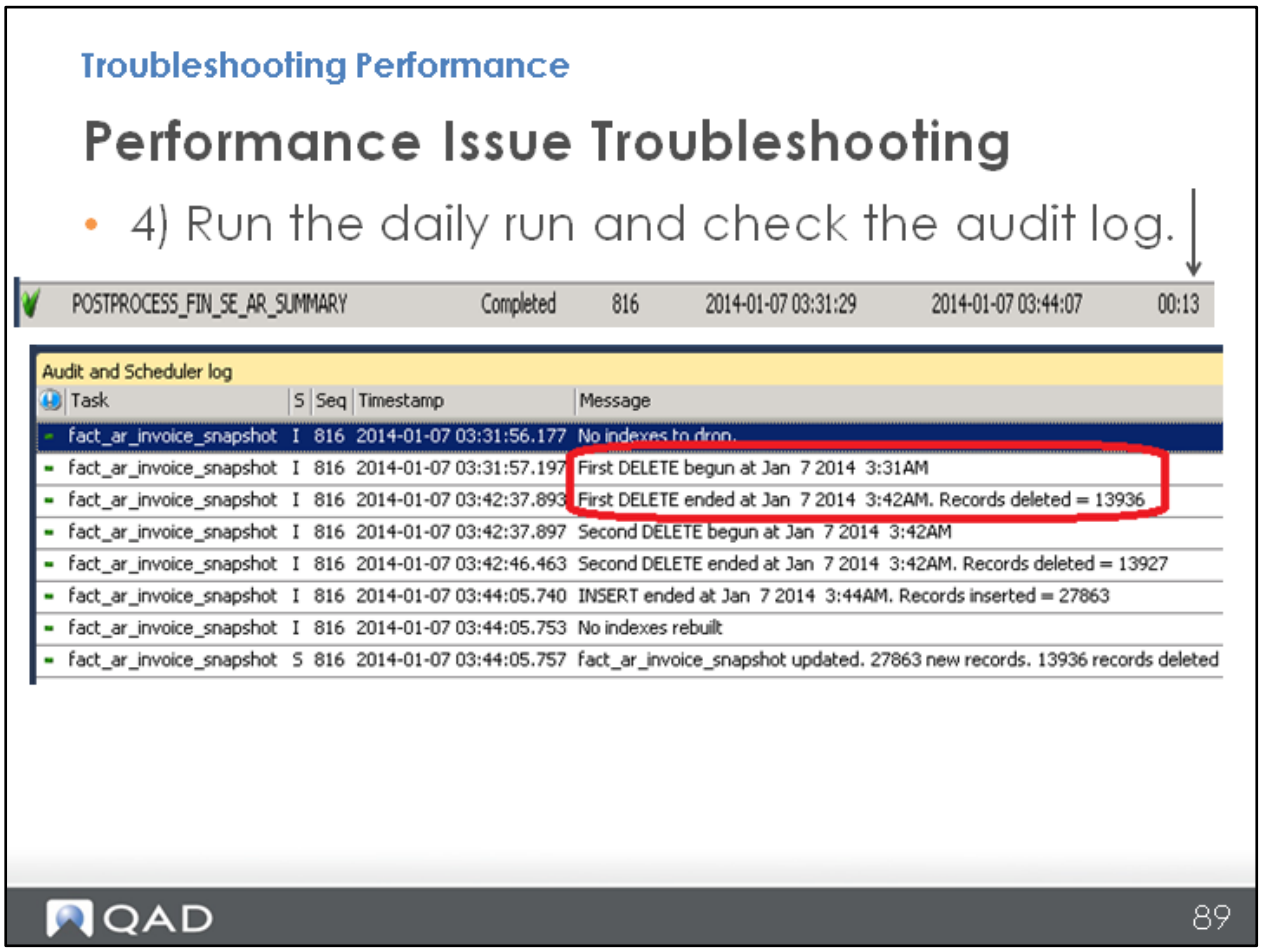
Before and after each of the three SQL deletes/inserts, add variable definitions and the EXEC WSWrkAudit to send audit messages to the Audit Log portion of the Scheduler.

Performance Issue Troubleshooting



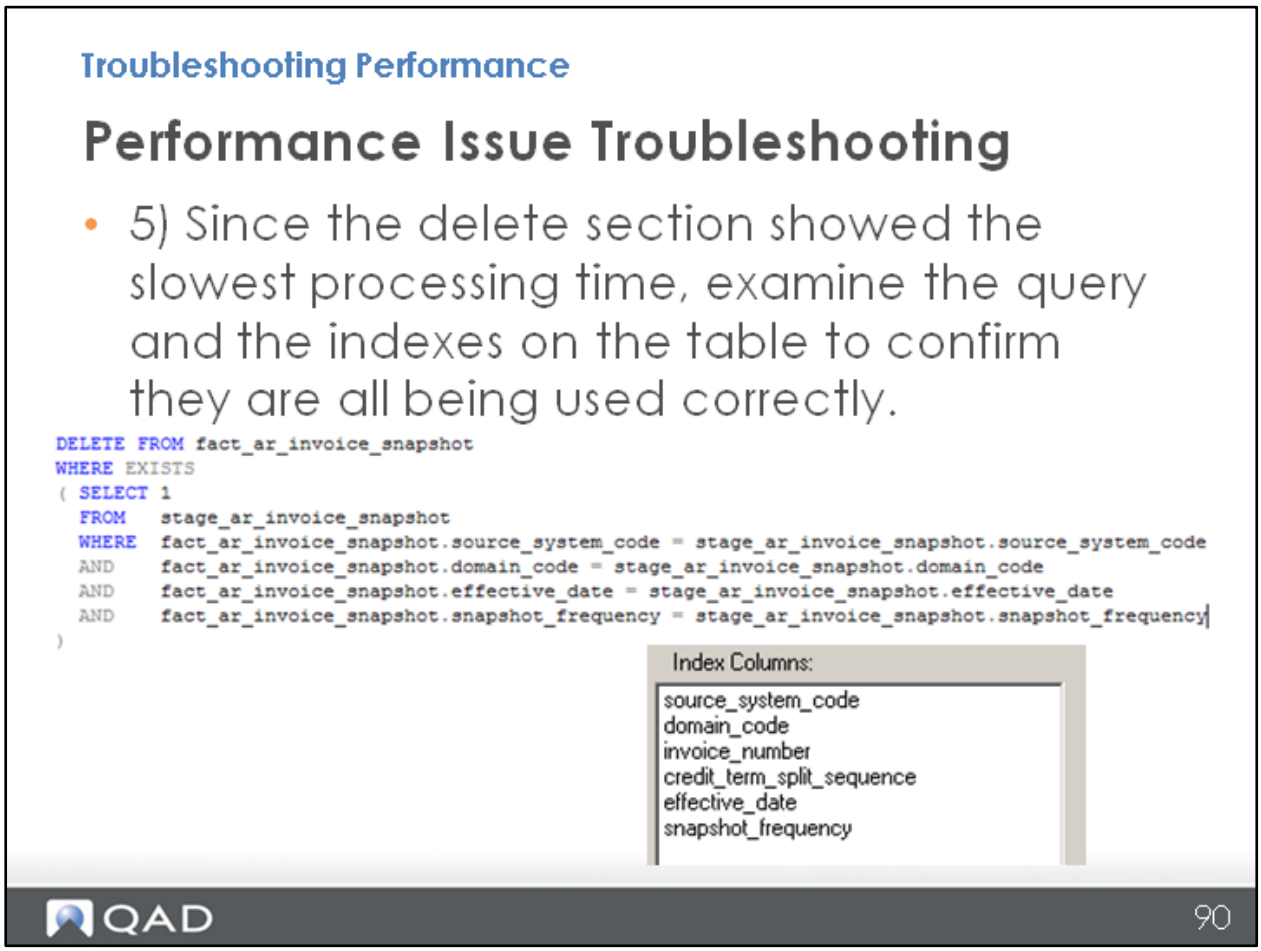
Insert the various audit information into the procedure before, between and after the three SQL parts and recompile the procedure.

Performance Issue Troubleshooting

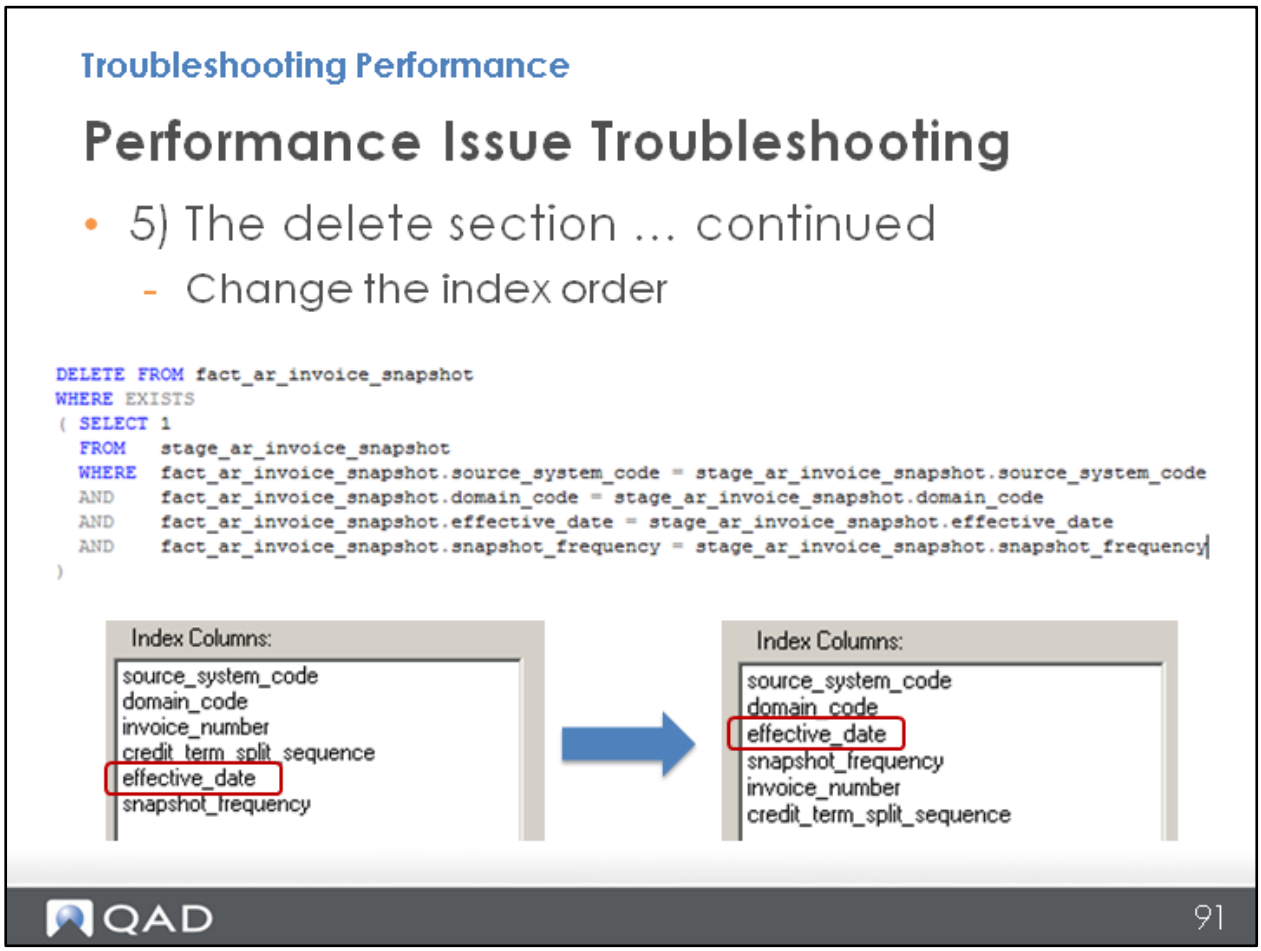


After running the job via the scheduler, you can now see which portions of the job took the longest. In this example, it's clear that the biggest time gap during the first delete which finished 11 minutes after the prior audit log message. That's where the biggest amount of processing time is taking and will be the focus of the research.

Performance Issue Troubleshooting

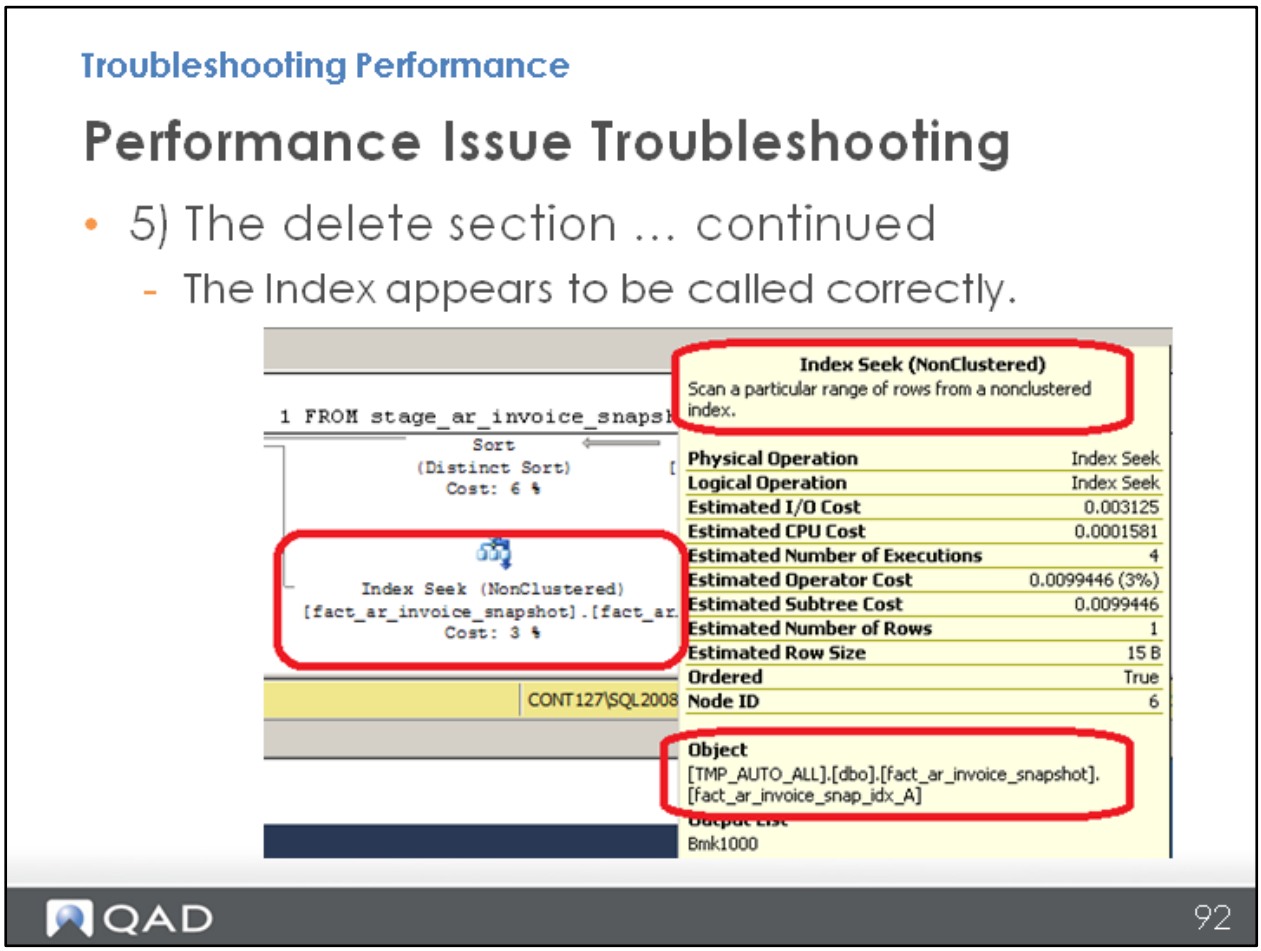


Performance Issue Troubleshooting



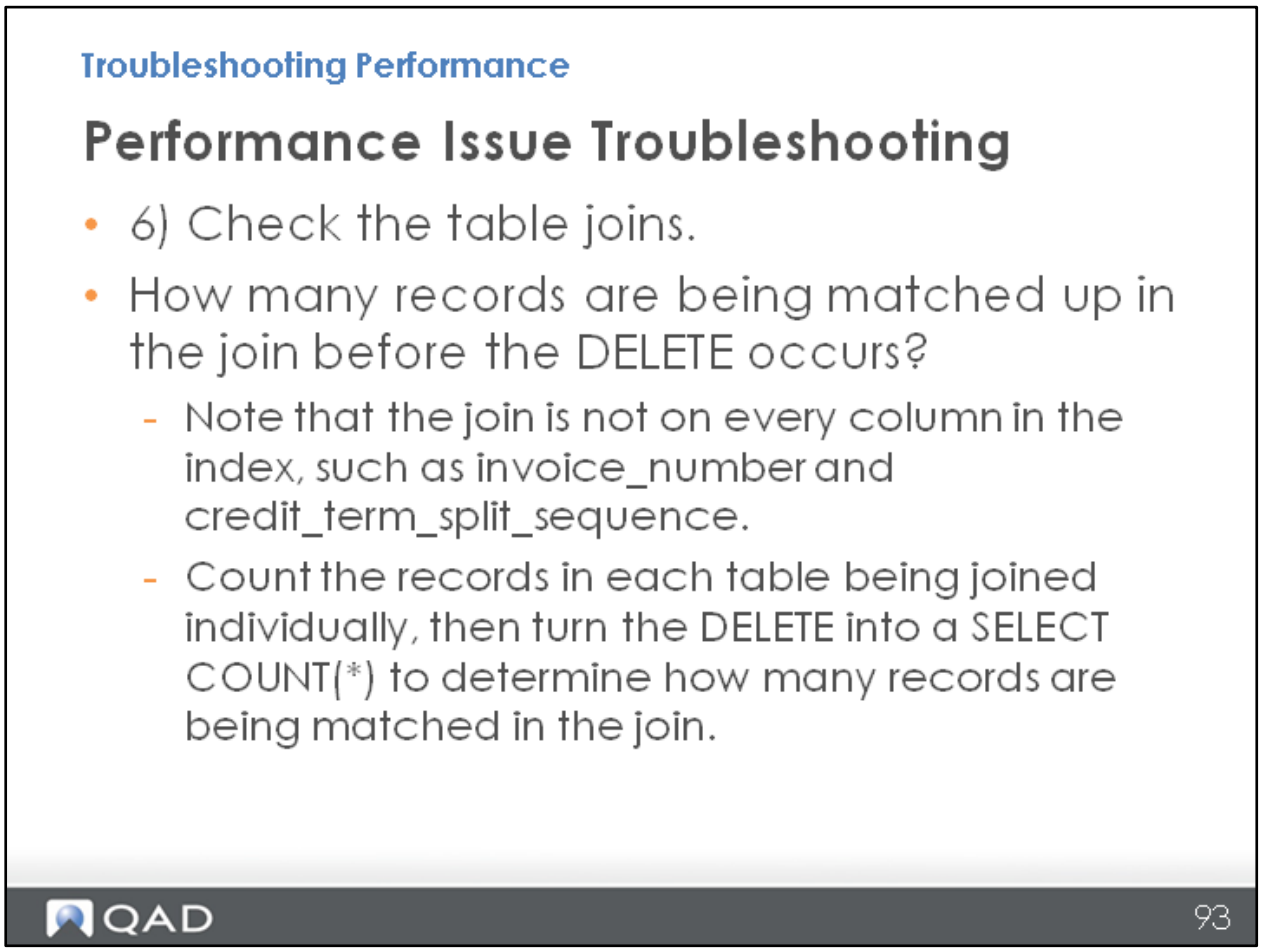
Indexes that are referenced only partially will only use the columns in the sequence they are defined in the index up to the point that there is a gap in columns referenced.

Performance Issue Troubleshooting

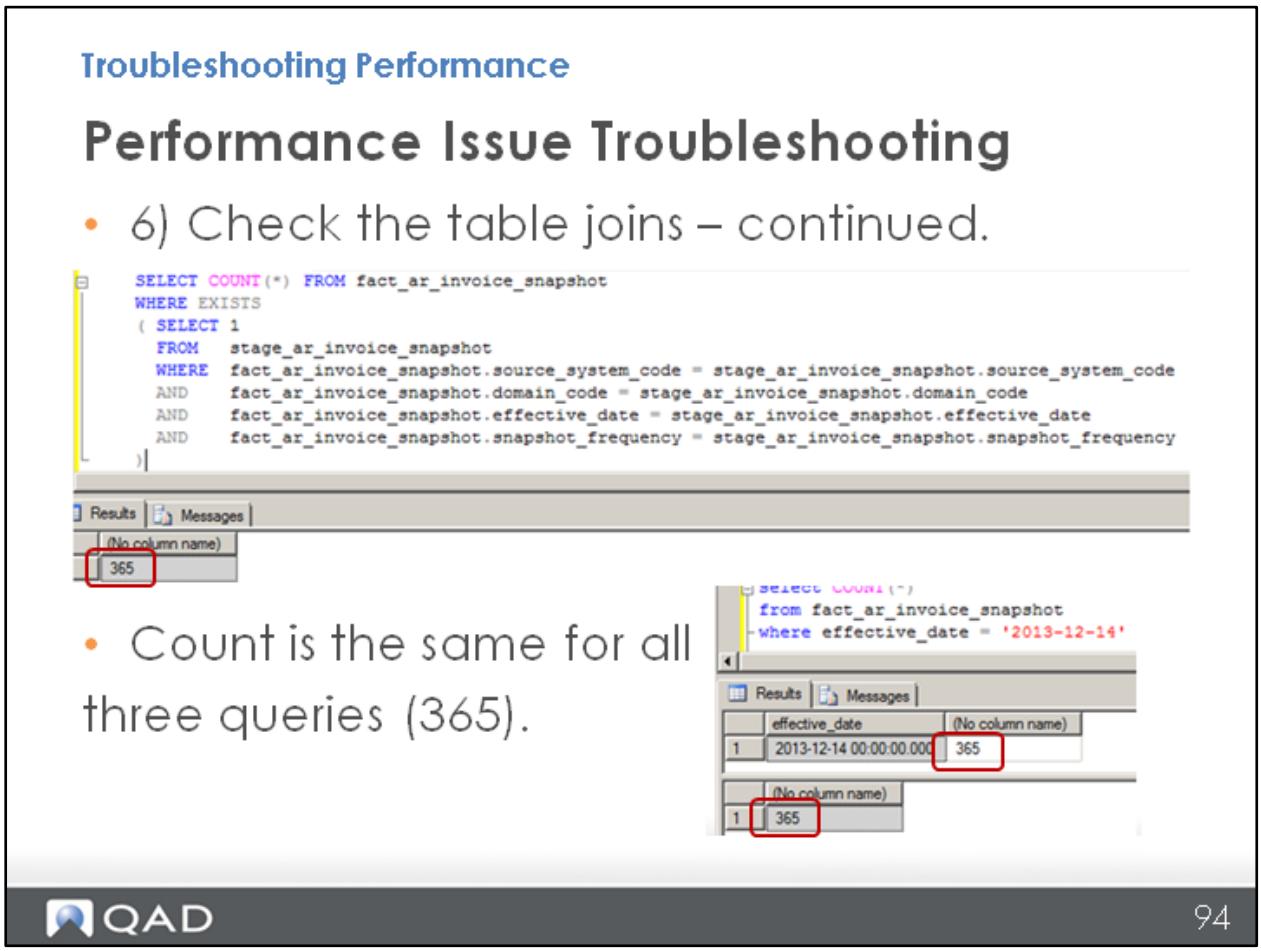


Index is being called properly. What else could be slowing things down?

Performance Issue Troubleshooting

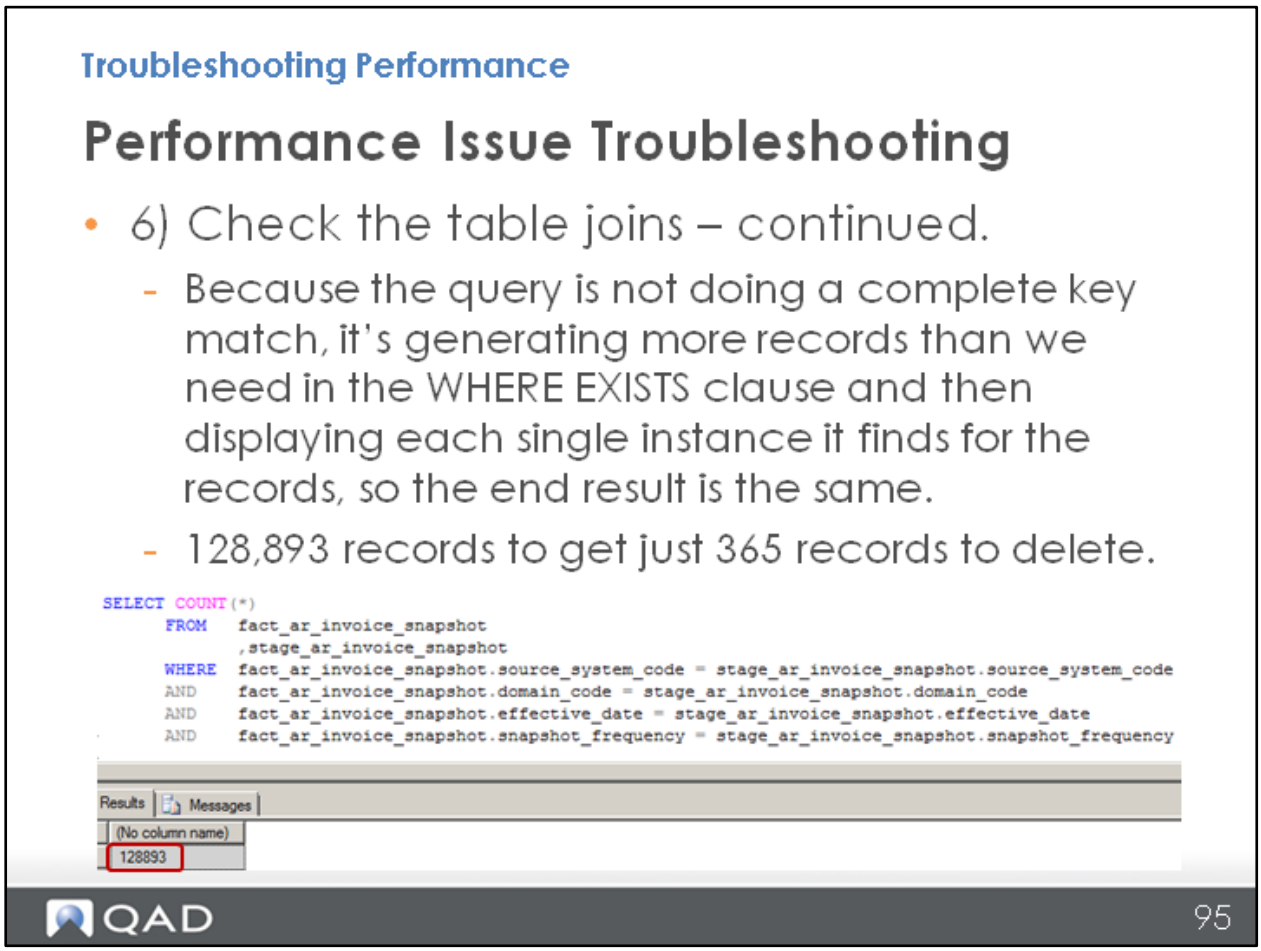


Performance Issue Troubleshooting



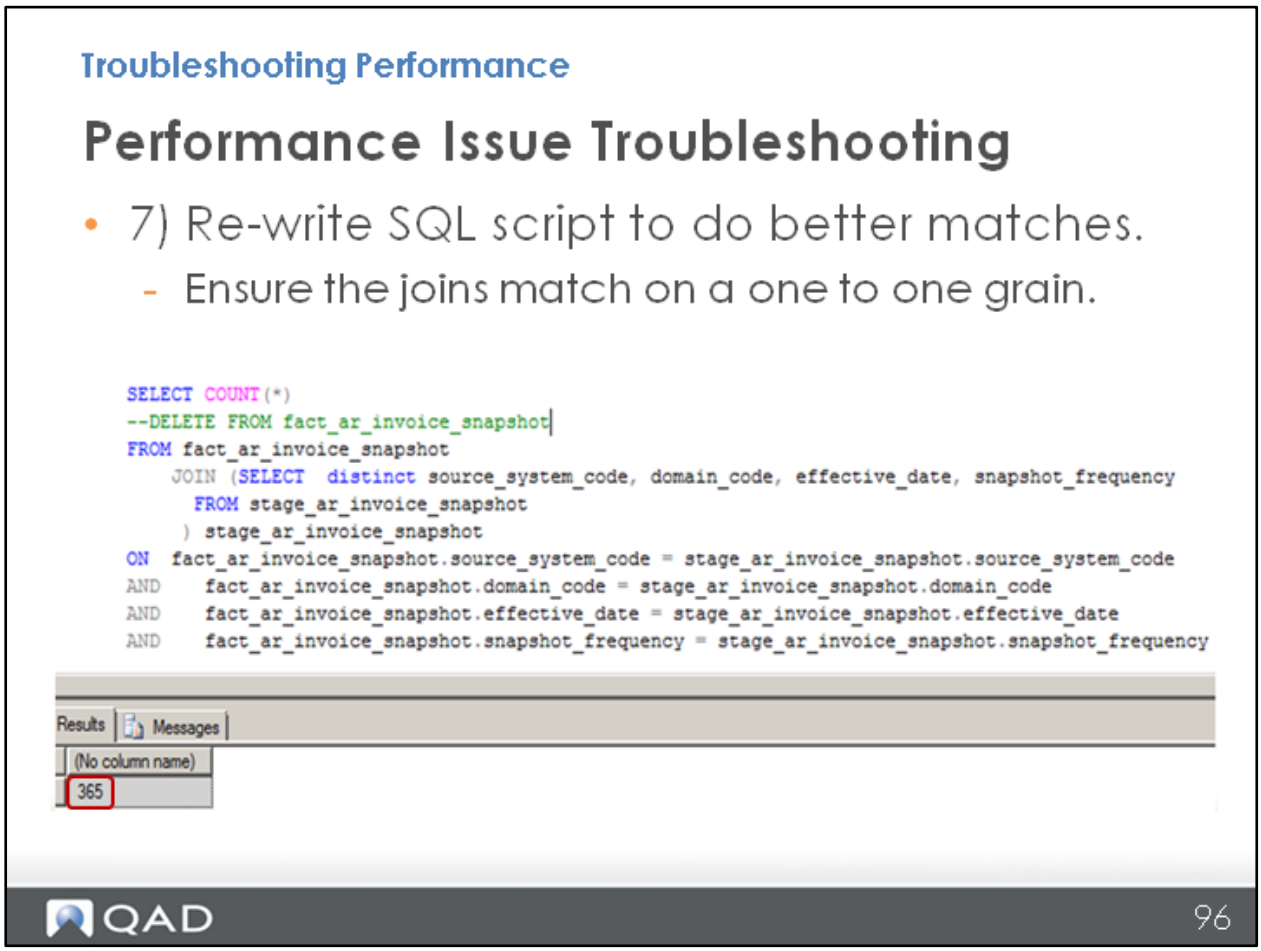
The count is the same for all three queries, but something is being overlooked.

Performance Issue Troubleshooting



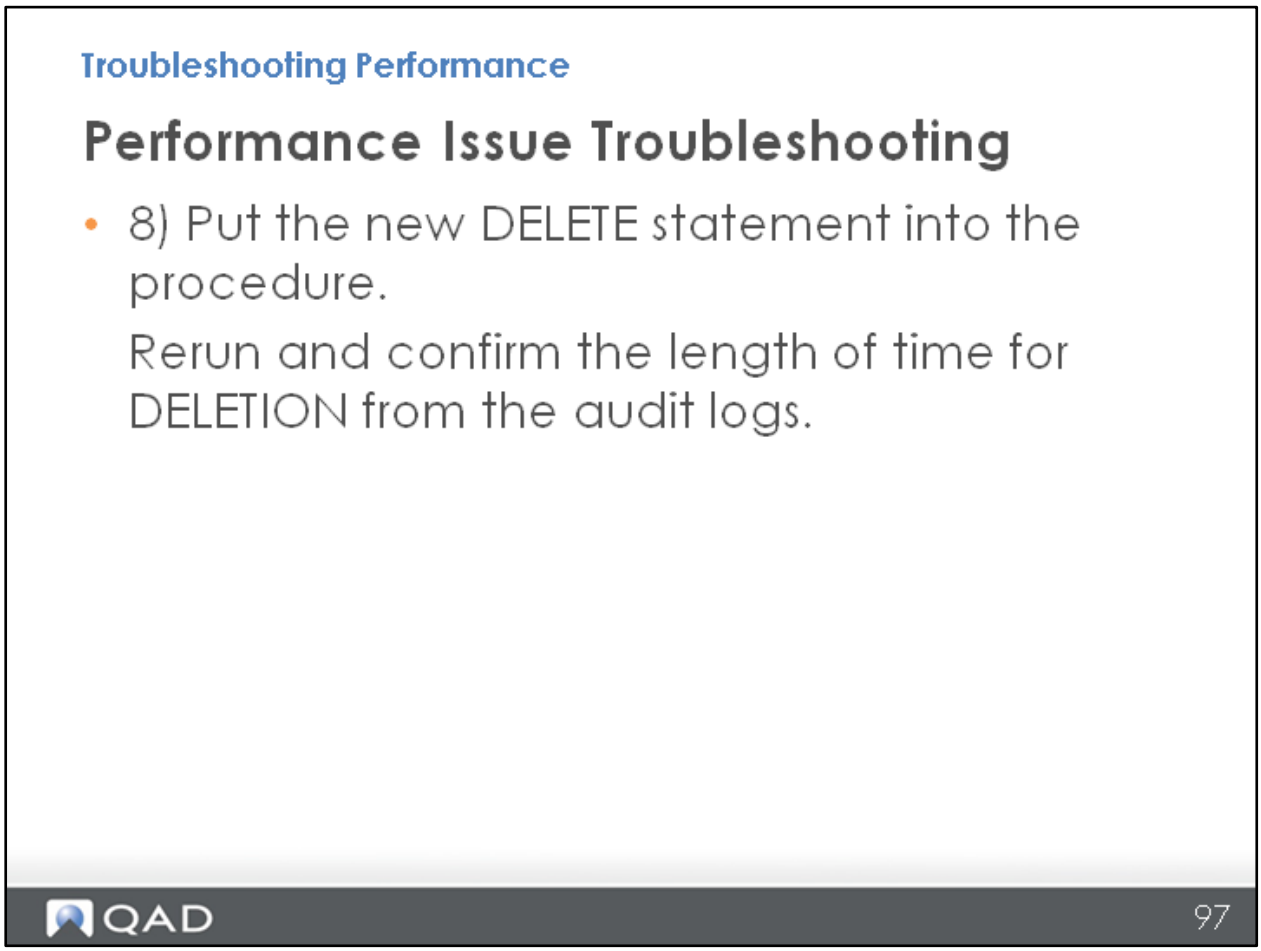
You can imagine what would happen if there are tens of thousands of records in the stage table and the fact table in this comparison. Potentially millions of records would be matched unnecessarily, taking up resources before the actual delete occurs.

Performance Issue Troubleshooting



Streamline the query to ensure the matches are unique.

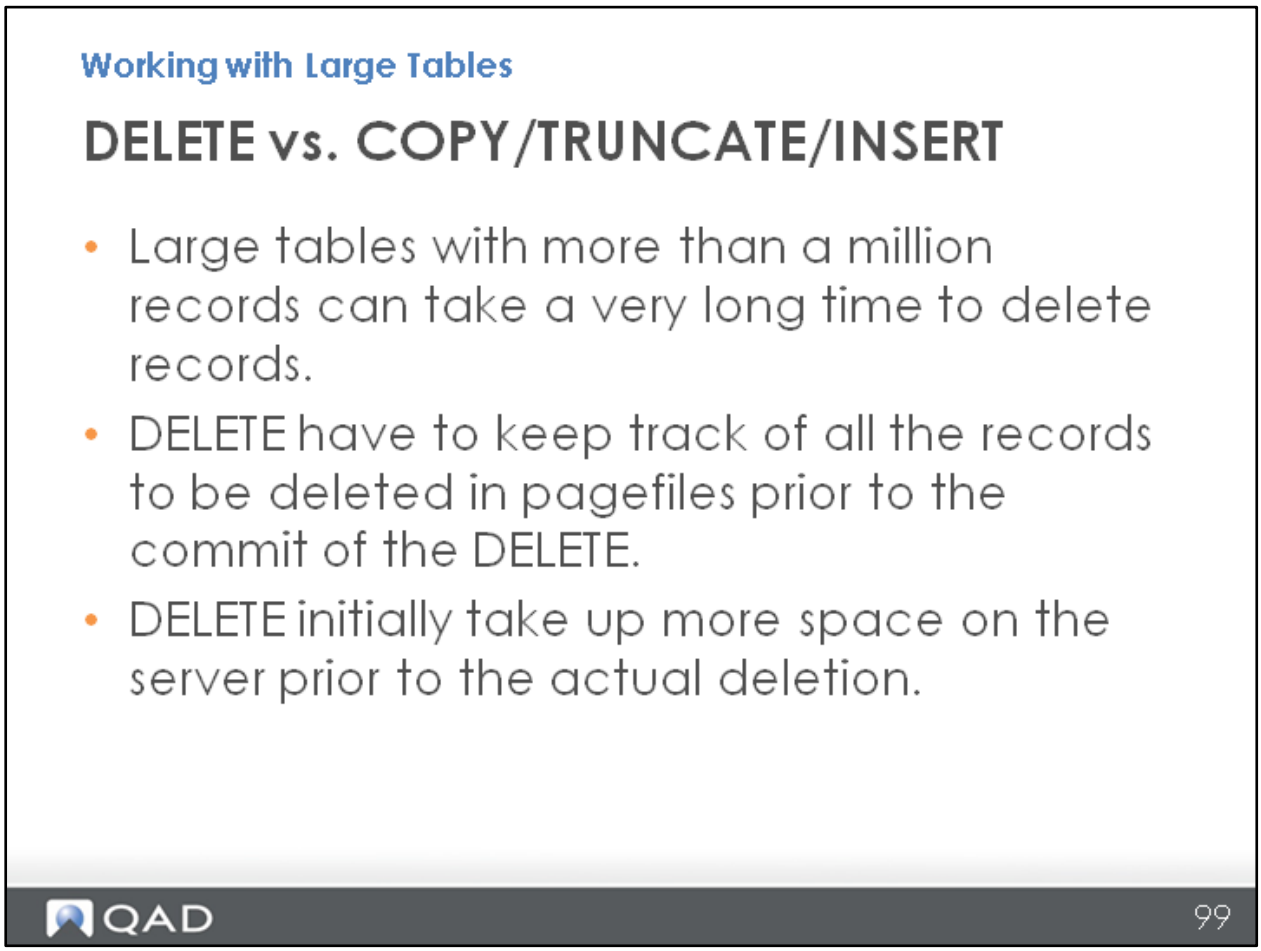
Performance Issue Troubleshooting



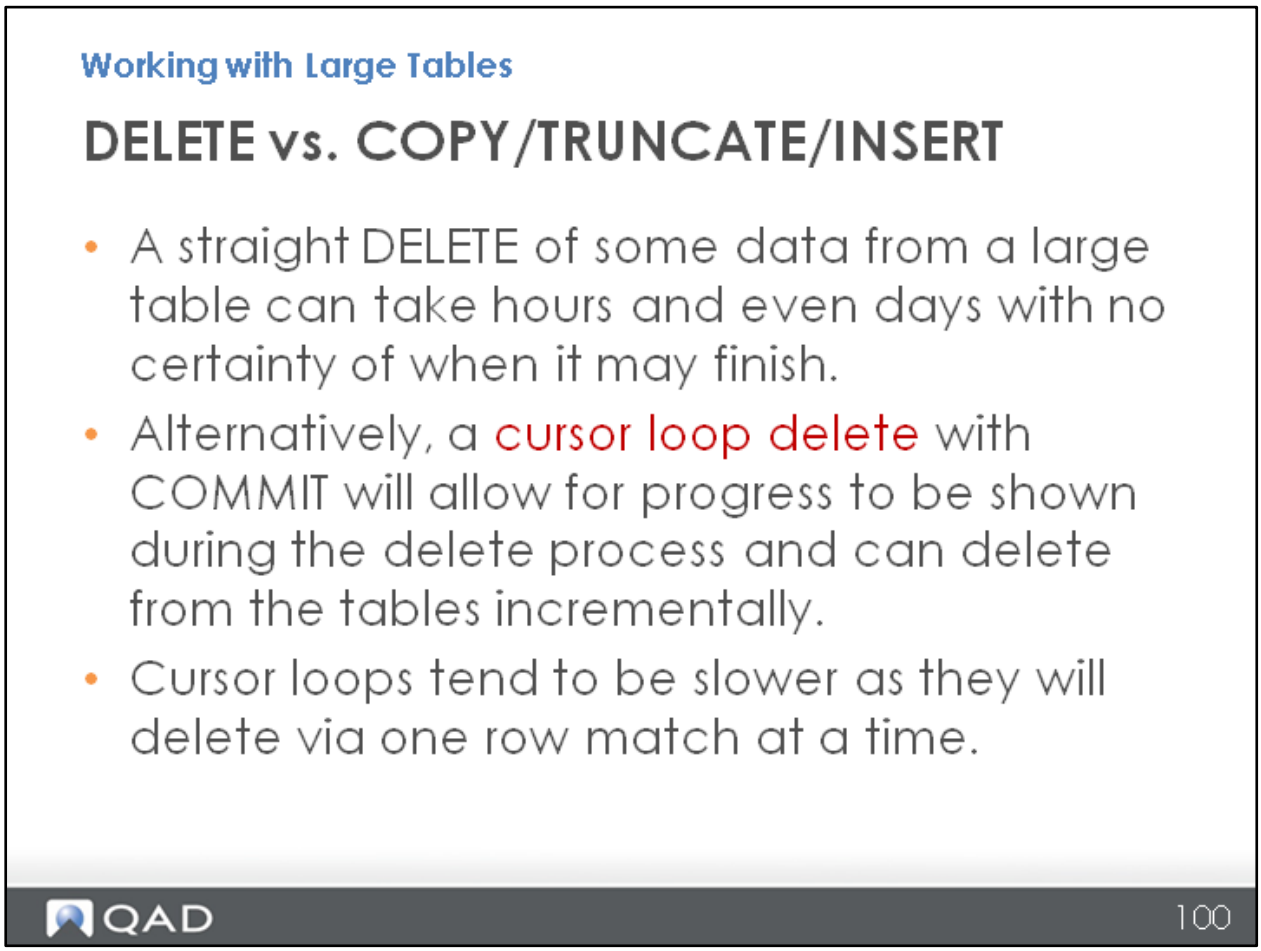
Working With Large Tables



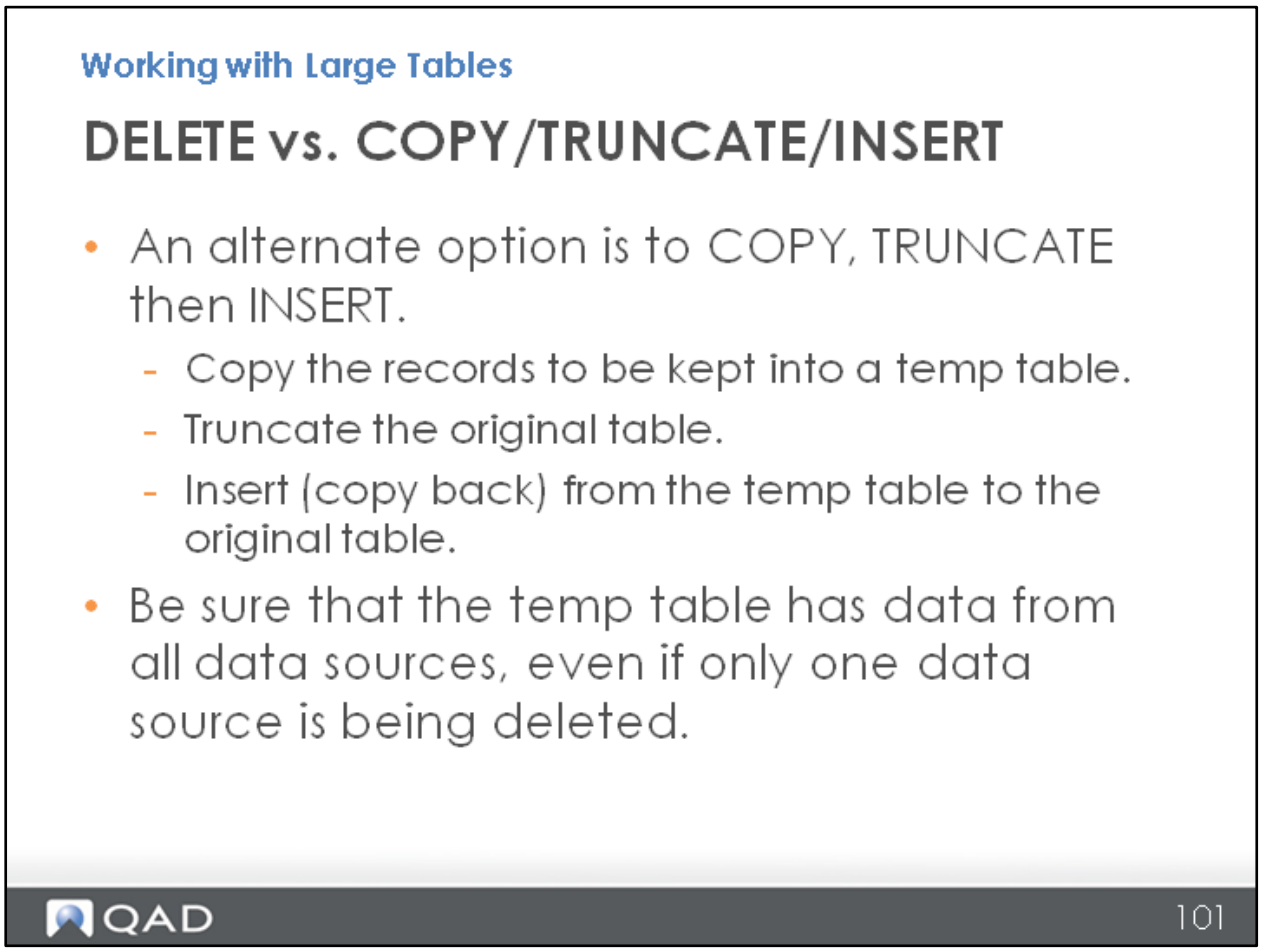
DELETE vs. COPY/TRUNCATE/INSERT



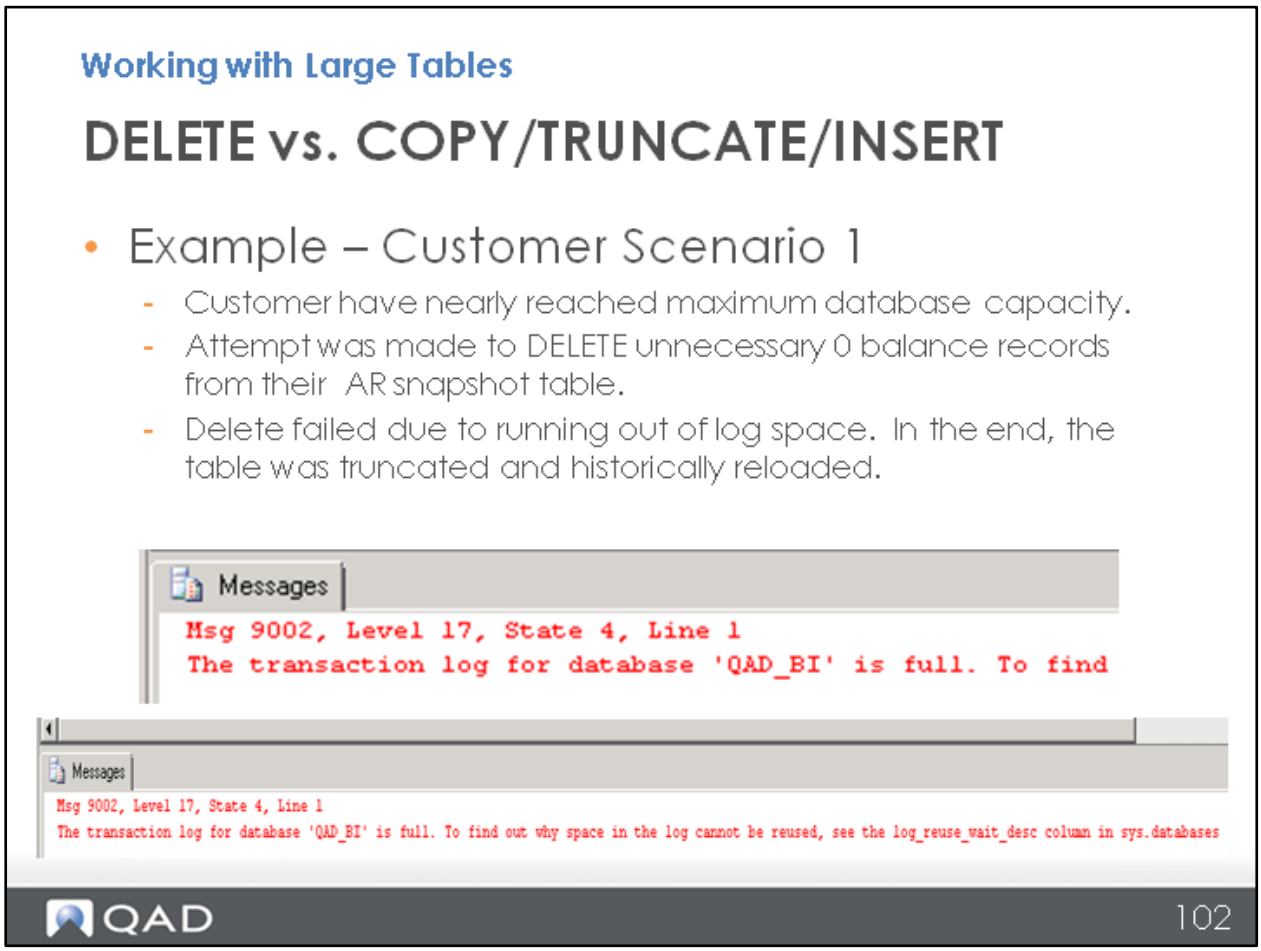
DELETE vs. COPY/TRUNCATE/INSERT



DELETE vs. COPY/TRUNCATE/INSERT

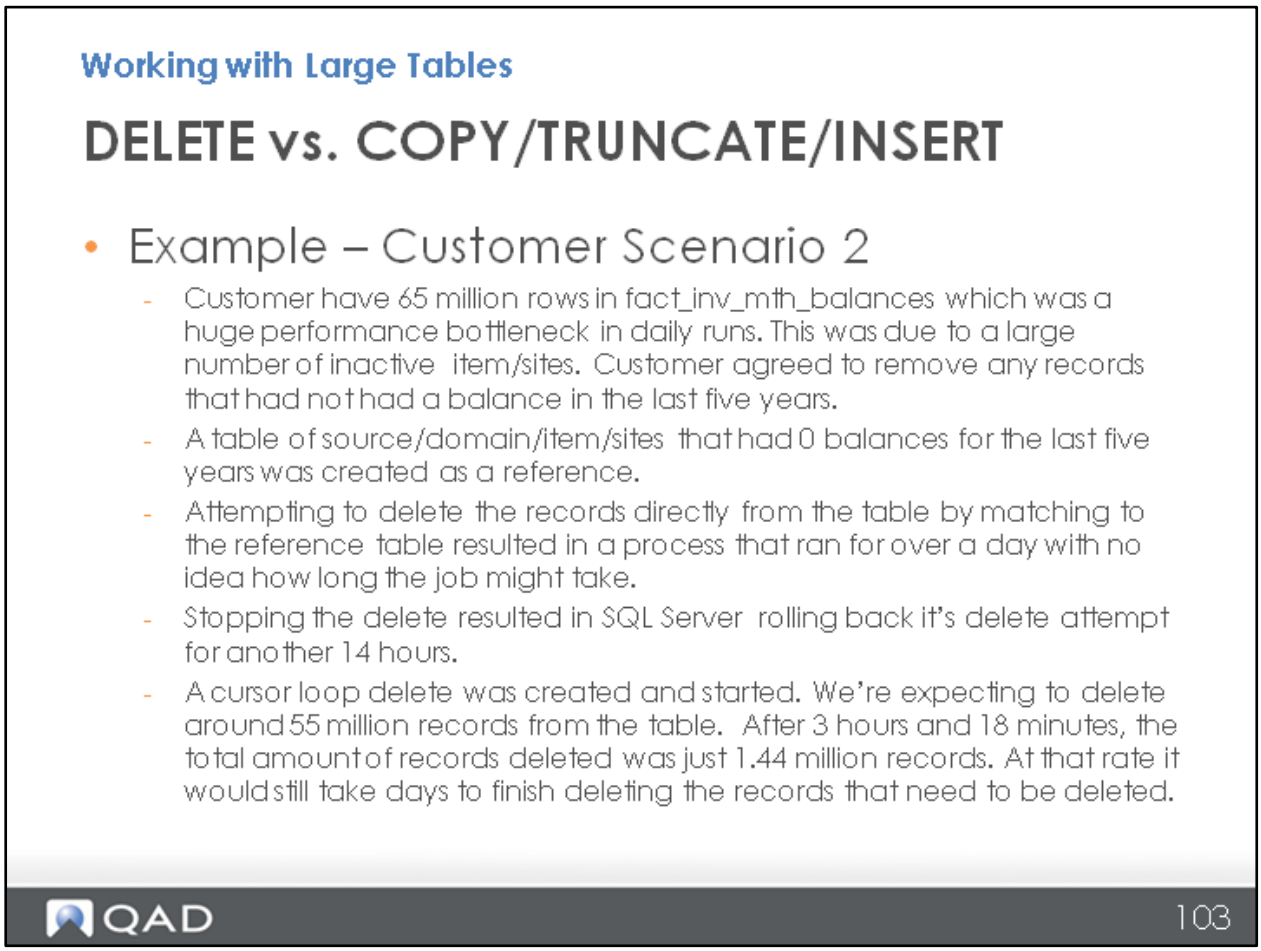


DELETE vs. COPY/TRUNCATE/INSERT



DELETE from tables take up lots of log space, because it needs to roll back in the event of failure.

DELETE vs. COPY/TRUNCATE/INSERT



DELETE vs. COPY/TRUNCATE/INSERT

Working with Large Tables

DELETE vs. COPY/TRUNCATE/INSERT

- Example – Customer Scenario 2
- As an alternative option, a series of inserts to create duplicate tables in conjunction with a TRUNCATE may be much faster.

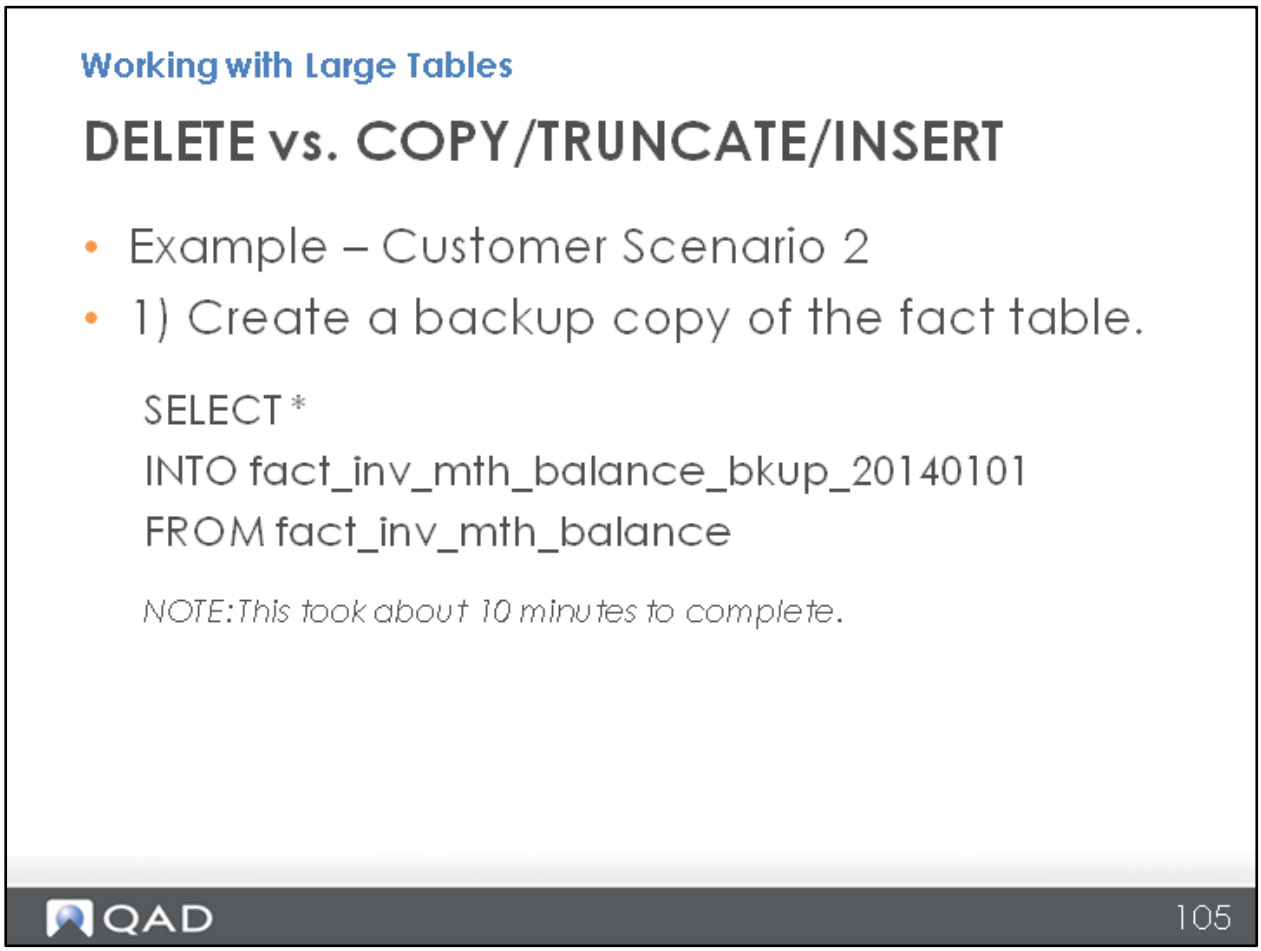
Job		Status	Seq	Start	Finish	Elapsed
Execute	customx_fact_inv_mth_balance_delete 775	Running	776	2014-01-06 13:29:00		03:18

lit and Scheduler log

Task	Status	Seq	Timestamp	Message
:customx_fact_inv_mth_balance_delete	I	776	2014-01-06 17:31:45.190	fact_inv_mth_balance record delete commited. 1224325 records delete..
:customx_fact_inv_mth_balance_delete	I	776	2014-01-06 17:40:13.843	fact_inv_mth_balance record delete commited. 1296325 records delete..
:customx_fact_inv_mth_balance_delete	I	776	2014-01-06 17:47:41.963	fact_inv_mth_balance record delete commited. 1368325 records delete..
:customx_fact_inv_mth_balance_delete	I	776	2014-01-06 17:55:11.343	fact_inv_mth_balance record delete commite . 1440325 records delete..

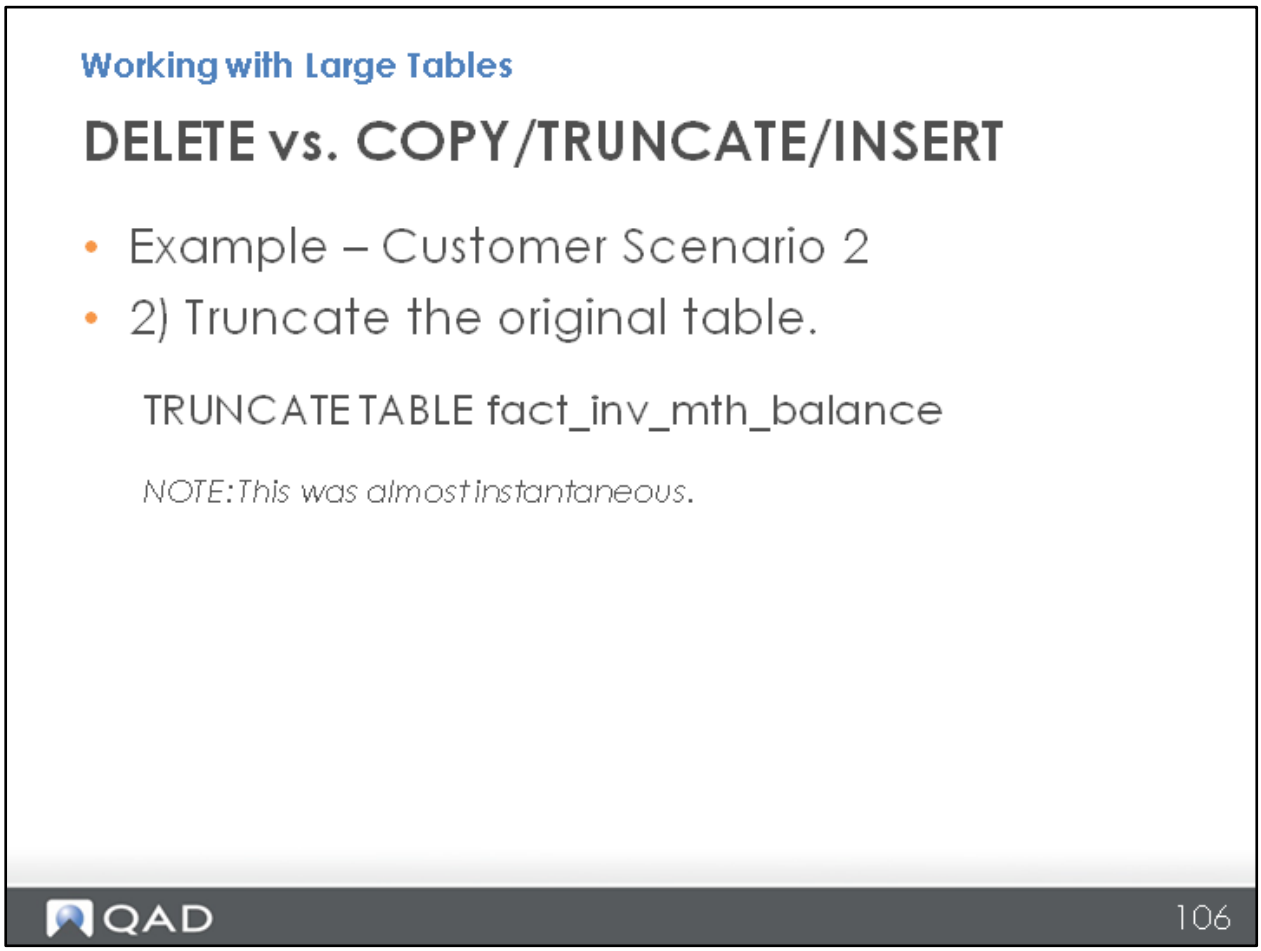
QAD 104

DELETE vs. COPY/TRUNCATE/INSERT

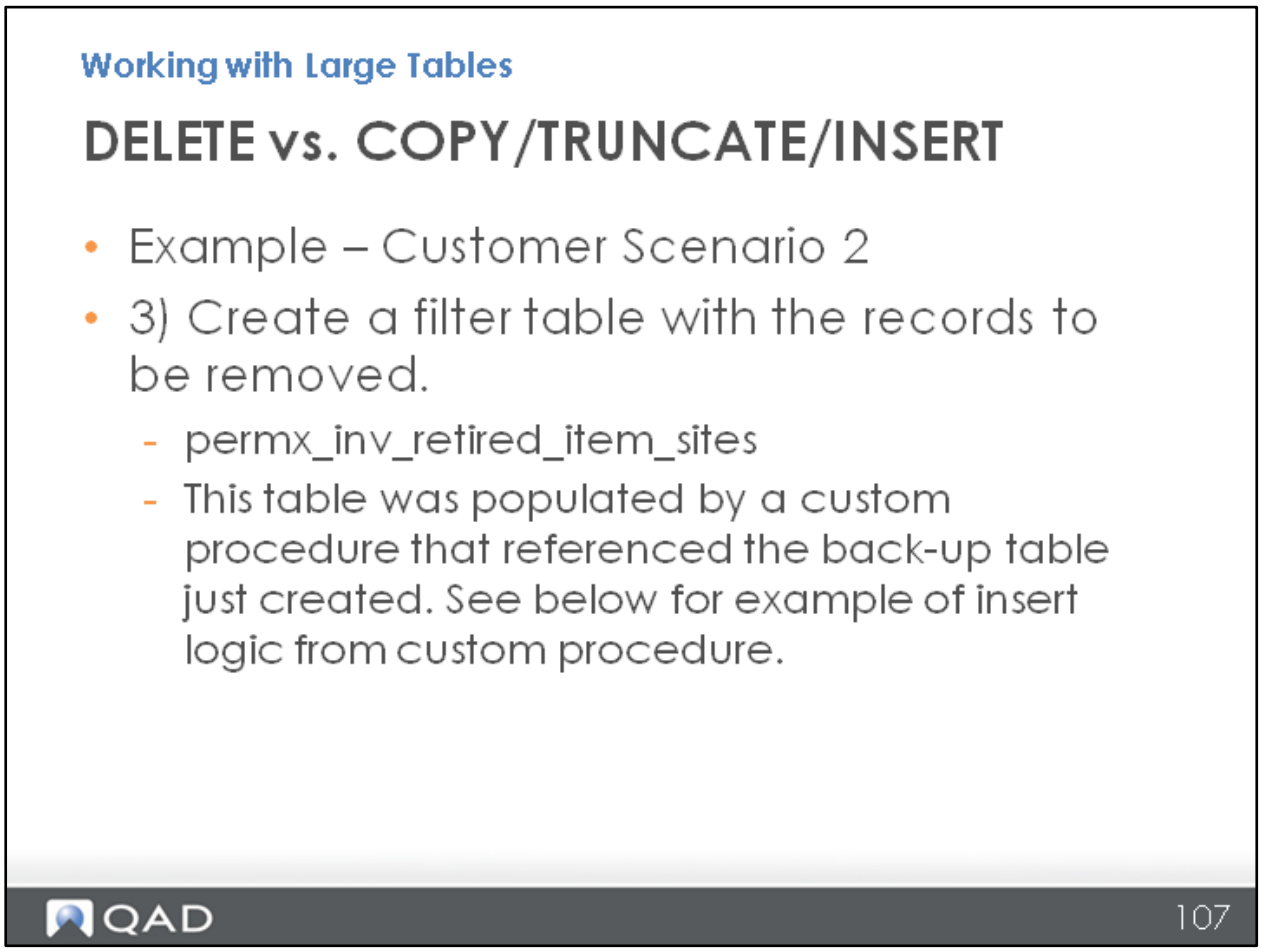


Create a backup of the original to ensure that if something goes wrong, the integrity of the data from the table to be TRUNCATED is maintained.

DELETE vs. COPY/TRUNCATE/INSERT



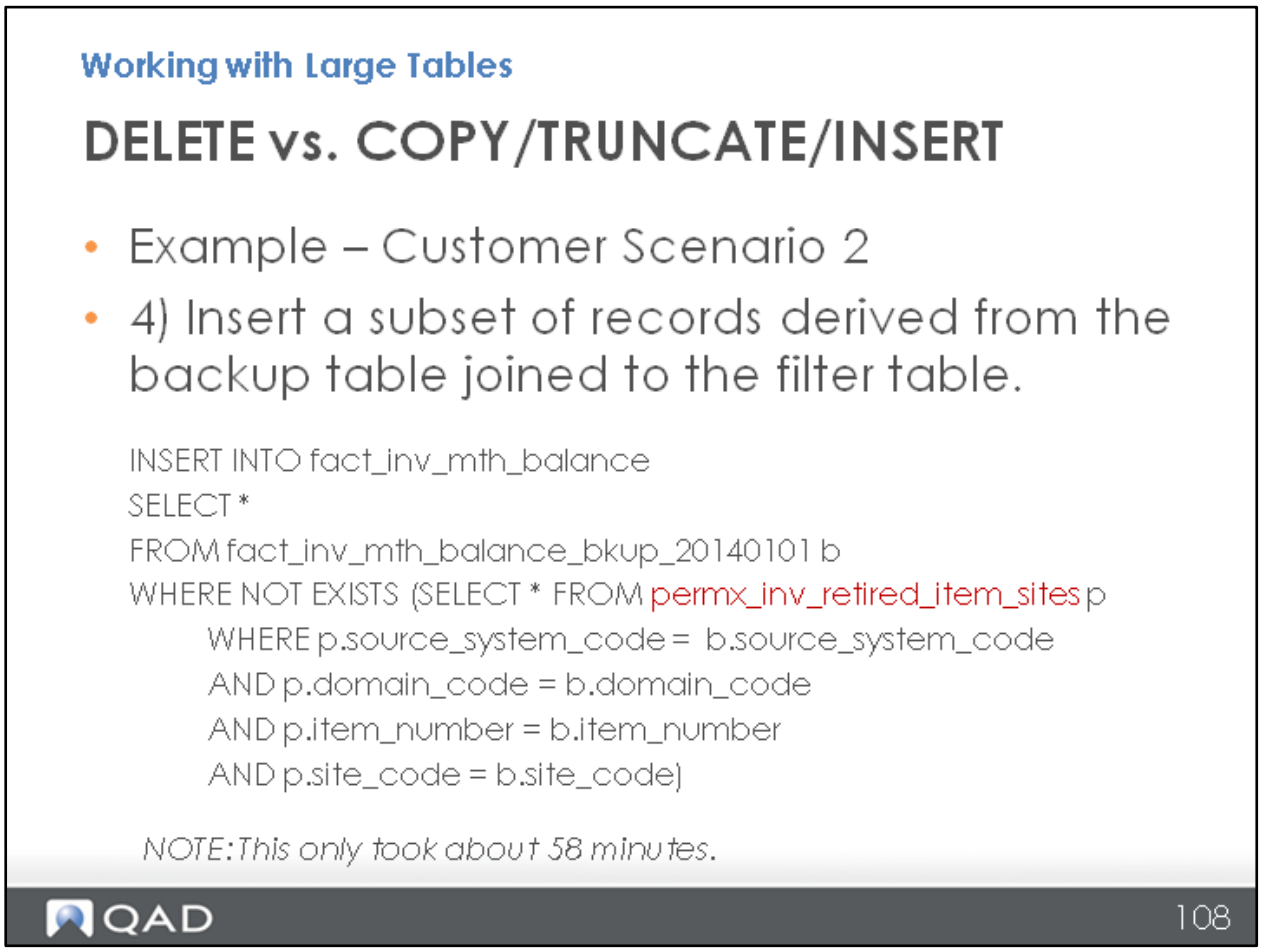
DELETE vs. COPY/TRUNCATE/INSERT



The inline view within this procedure looks for any instances where the number of records exceeds 150 which is a thumbnail calculation based on 3 consignment types * 12 months * 4 years. It also totals up the absolute value of the quantities begin, change and end for each month. If in that 4+ year span, there have been no changes in quantity to the item_number/site_code combination, the item/site combination was deemed inactive and put in this table to be used as a filter so that going forward only records not in the filter would make it to the fact table.

```
select source_system_code, domain_code, item_number, site_code
into permx_inv_retired_item_sites
from (select source_system_code,
             domain_code,
             item_number,
             site_code,
             sum(abs(quantity_changed)+abs(quantity_begin)+abs(quantity_end))
total_chg,
             count(*)
      from fact_inv_mth_balances_bkup_20140101
      group by source_system_code,
               domain_code,
               item_number,
               site_code,
               having count(*) > 150 -- 3 consignment types * 12 months * approx
4 years
     ) x
where total_chg = 0
```

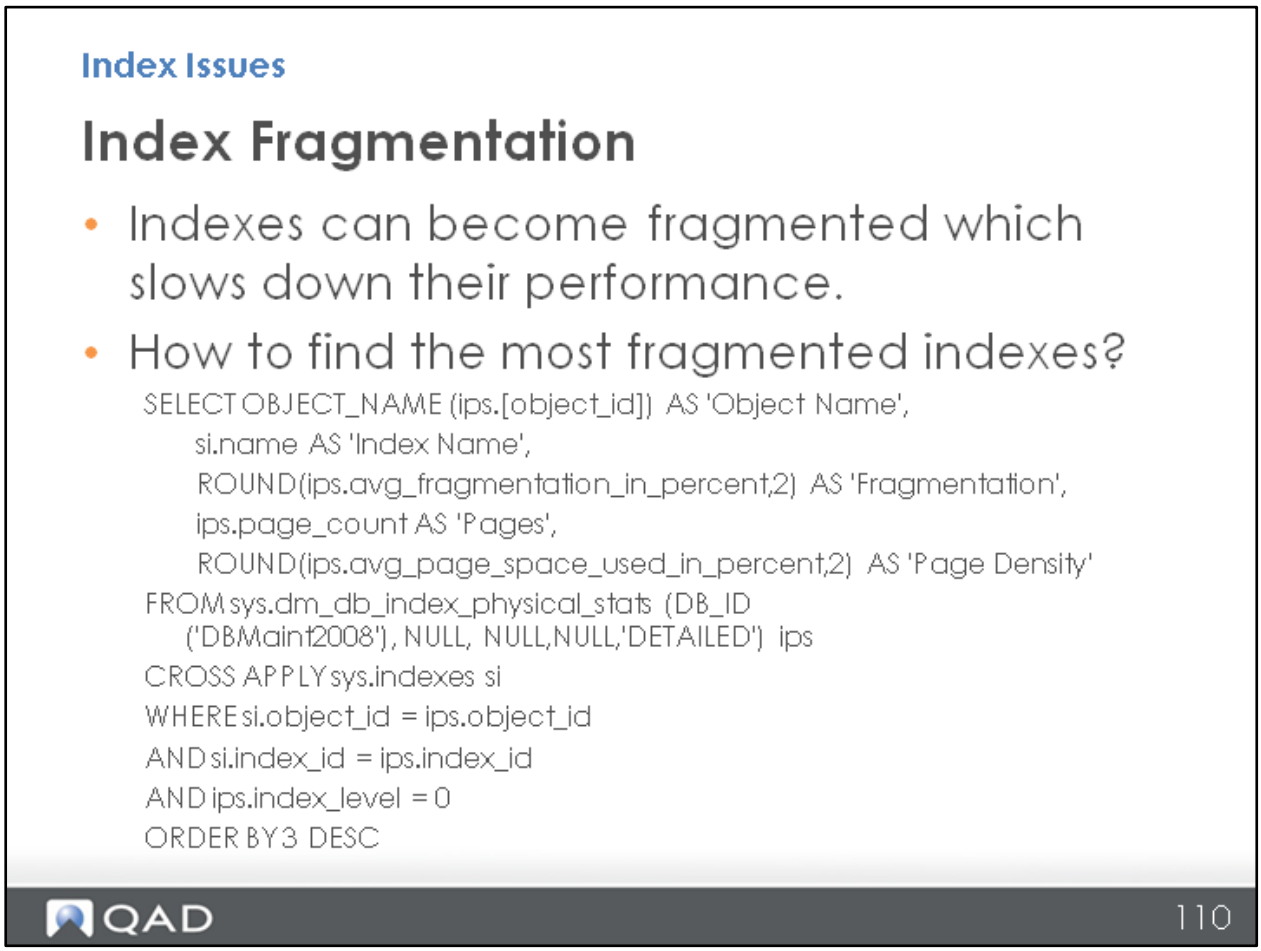
DELETE vs. COPY/TRUNCATE/INSERT



Index Issues



Index Fragmentation



Index Fragmentation

Index Issues

Index Fragmentation

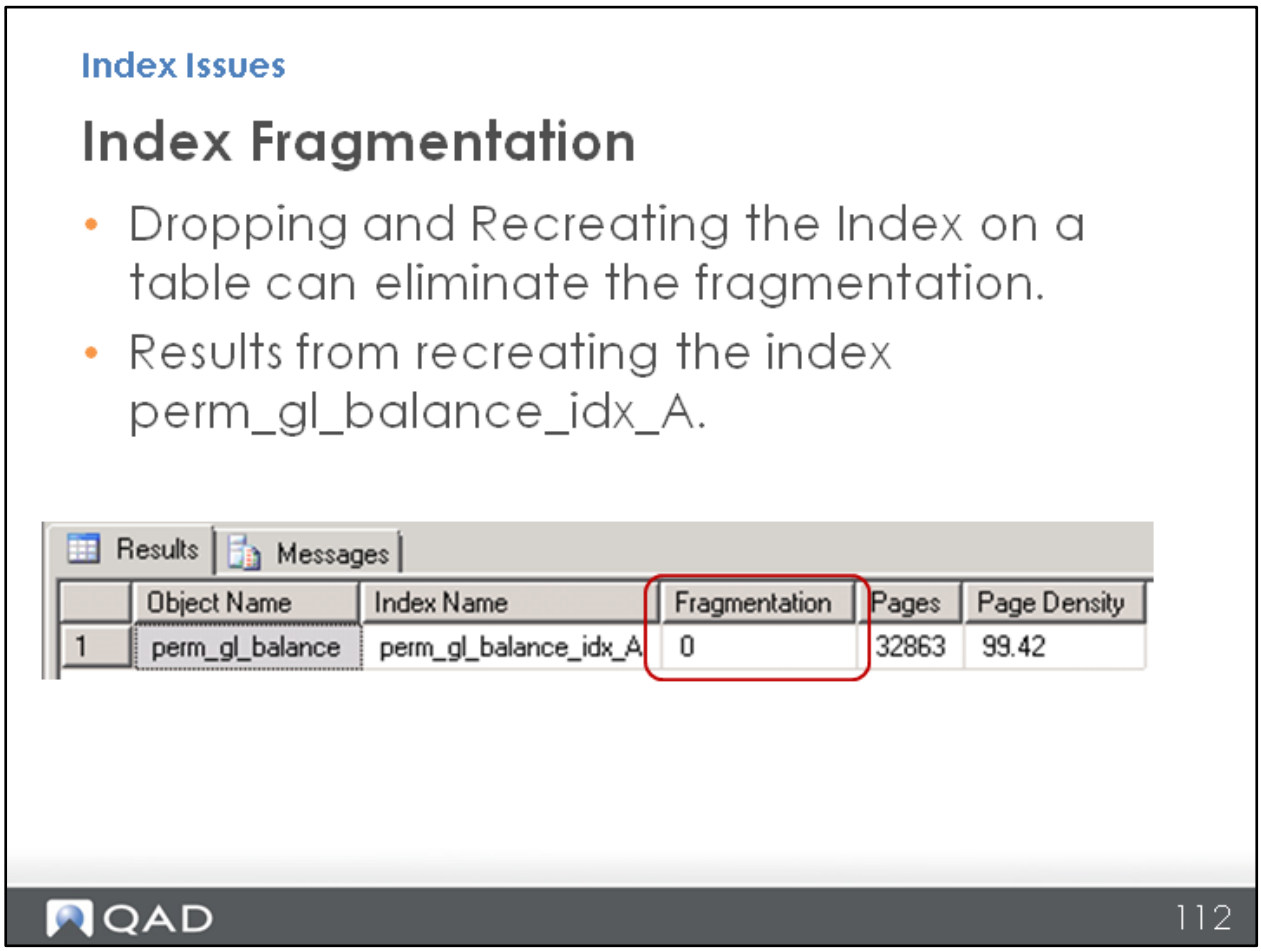
- Results from query

Results | Messages

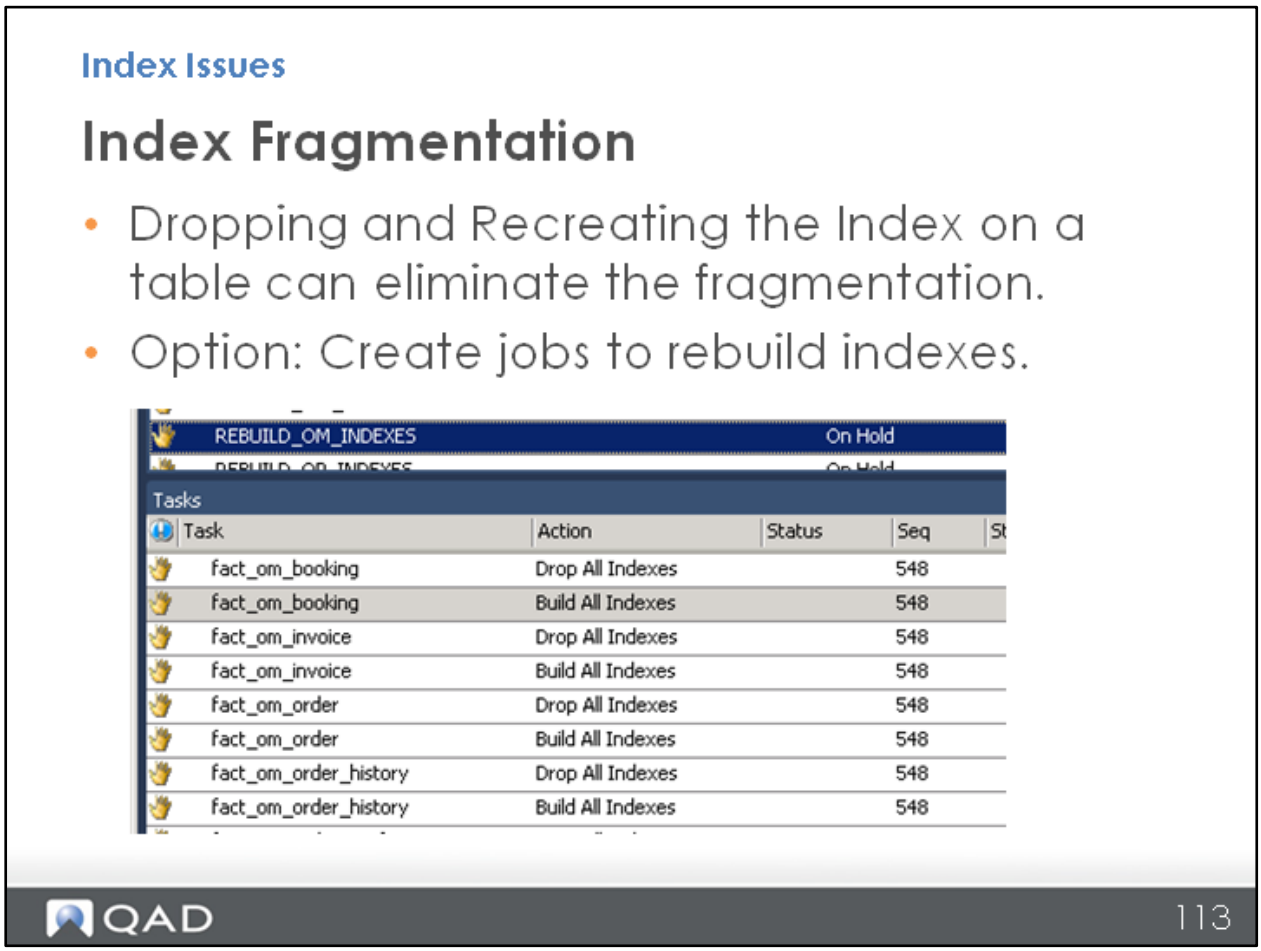
	Object Name	Index Name	Fragmentation	Pages	Page Density
1	fact_ar_invoice_snapshot	fact_ar_invoice_snap_idx_16	99.79	4729	47.45
2	perm_gl_balance	perm_gl_balance_idx_A	99.77	37147	87.95
3	fact_ar_invoice_snapshot	fact_ar_invoice_snap_idx_9	99.49	4708	47.66
4	fact_ap_voucher	fact_ap_voucher_idx_7	99.46	933	47.94
5	stage_om_ord_perf_date	NULL	99.46	20887	99.4
6	fact_om_order	fact_om_order_idx_24	99.27	4930	49.83
7	fact_om_order	fact_om_order_idx_12	99.25	4918	49.95
8	fact_po_order_history	fact_po_order_history_idx_1	99.12	2610	53.01
9	fact_ar_invoice_history	fact_ar_invoice_hist_idx_15	99.11	7970	51.27
10	fact_po_order	fact_po_order_idx_27	99.04	6054	51.12
11	fact_om_order	fact_om_order_idx_14	99.04	4885	50.29
12	stage_inv_transaction	stage_inv_transaction_idx_A	99.03	2689	58.14
13	fact_ap_voucher	fact_ap_voucher_idx_17	98.93	938	47.69
14	fact_ap_voucher	fact_ap_voucher_idx_2	98.93	938	47.69

QAD 111

Index Fragmentation

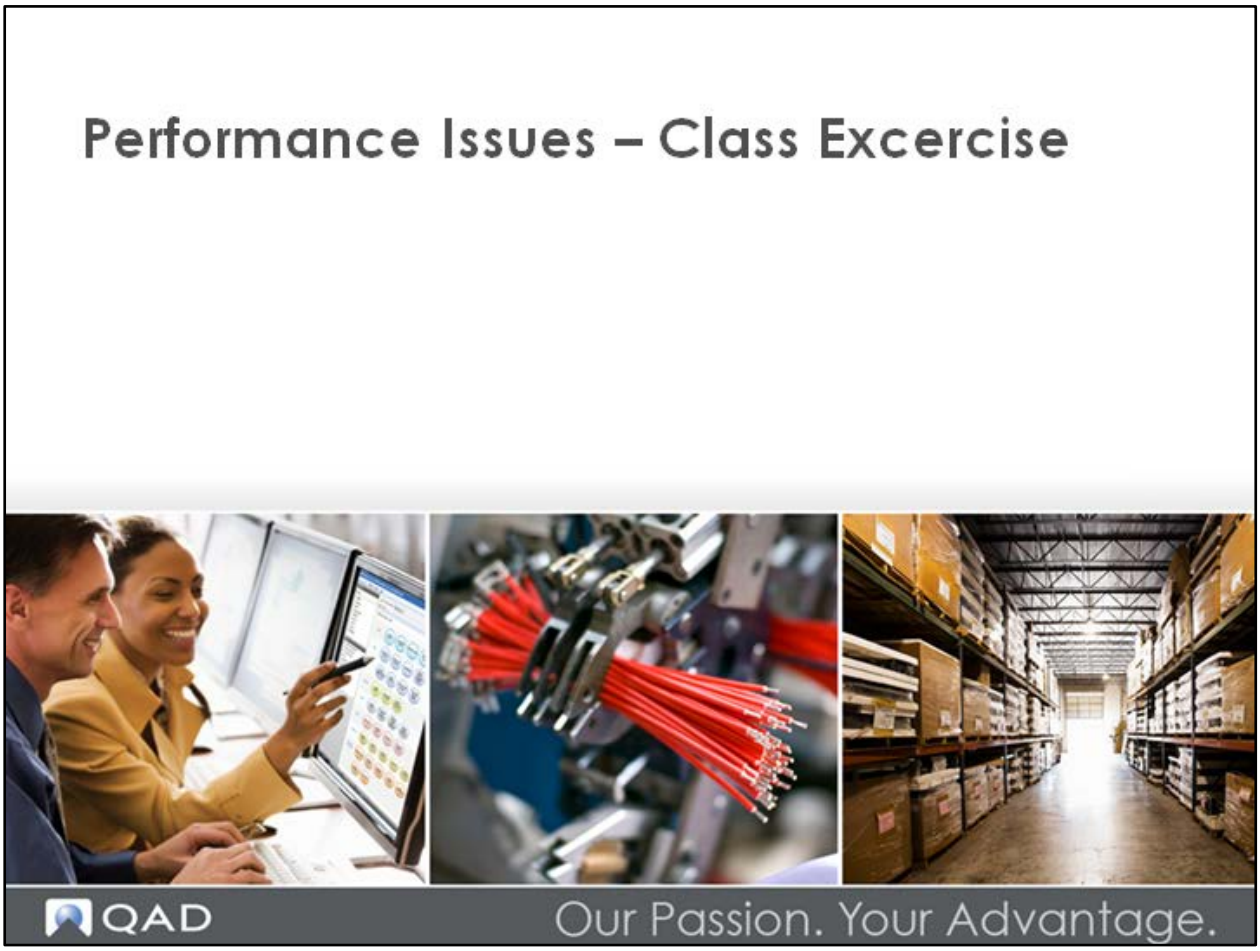


Index Fragmentation

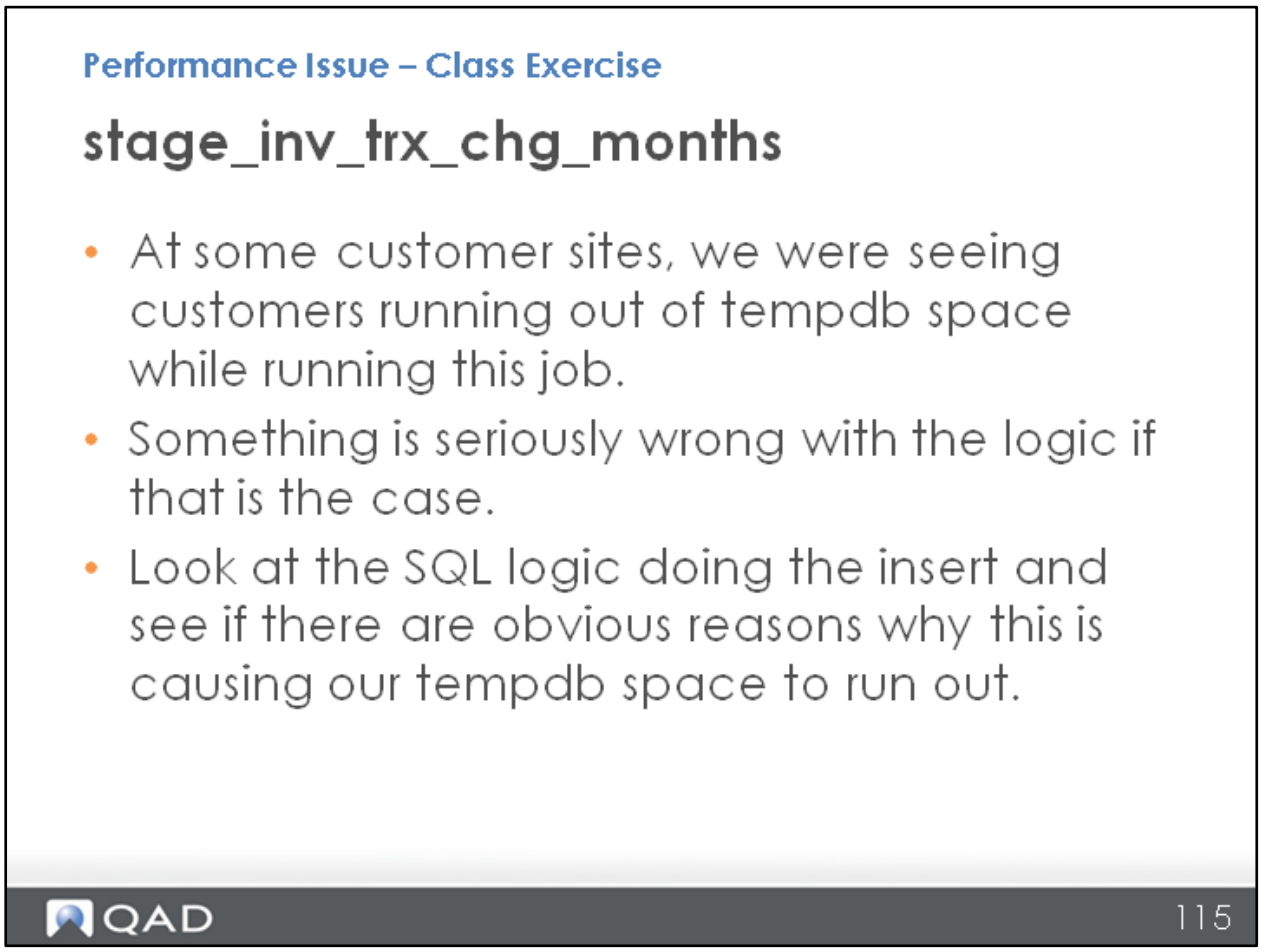


The jobs could be quickly built for existing installations.

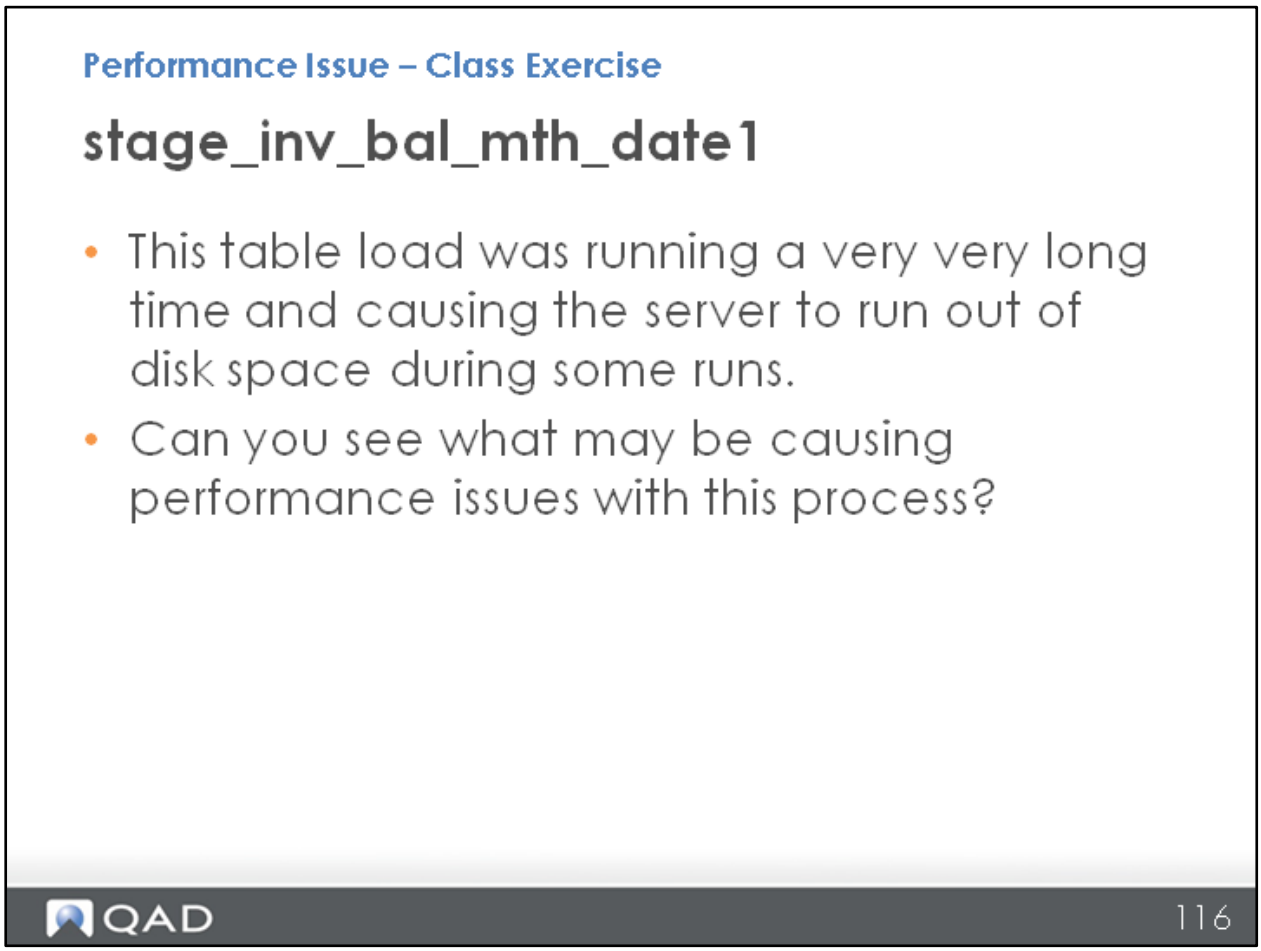
Performance Issues – Class Excercise



stage_inv_trx_chg_months



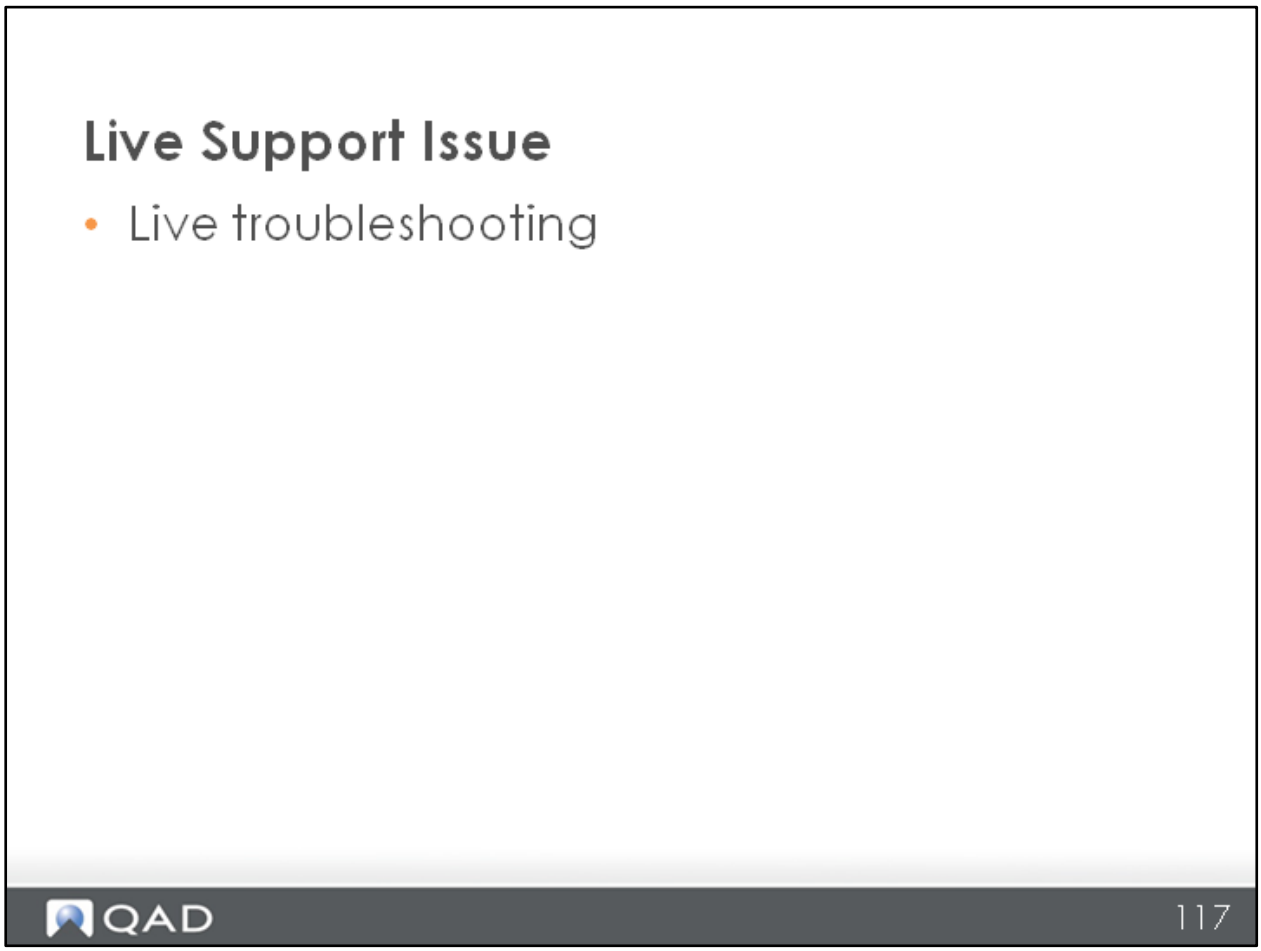
stage_inv_bal_mth_date1



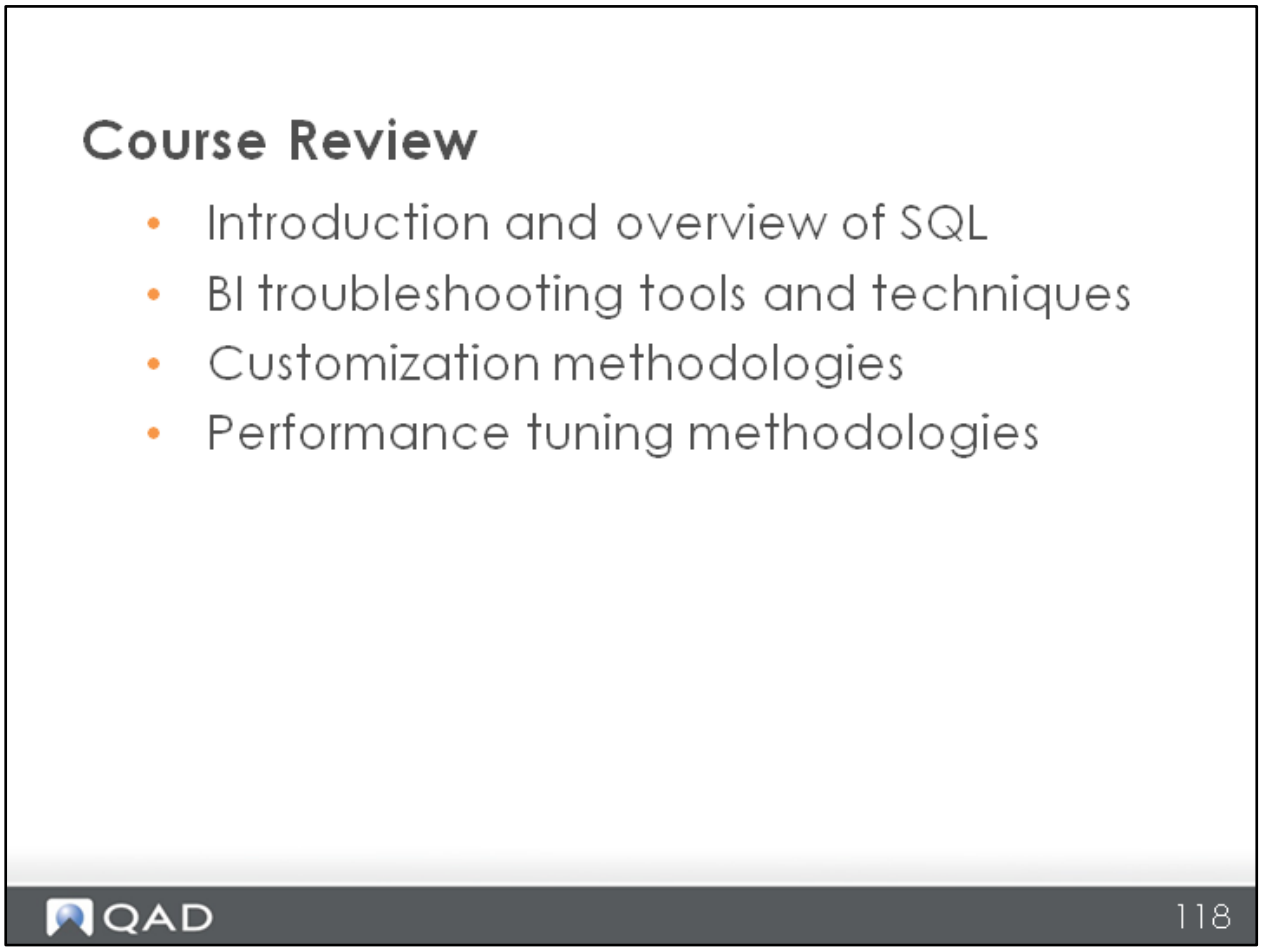
Here's the query in question –

```
SELECT
    fact_inv_transaction.source_system_code,
    fact_inv_transaction.domain_code,
    ISNULL(dim_effective_date.dim_effective_date_key,0) dim_effective_date_key,
    MAX(ded2.effective_date),
    ISNULL(dim_effective_date.effective_fin_month,0) effective_fin_month,
    ISNULL(dim_effective_date.effective_fin_day_in_month,0)
effective_fin_day_in_month,
    @v_dss_update_time
    FROM fact_inv_transaction,
         dim_effective_date,
         dim_effective_date ded2
 WHERE dim_effective_date.effective_date =
DATEADD(dd,DATEDIFF(dd,0,@v_INV_LAST_IN_MSTR_EXTRACT),0)
   AND dim_effective_date.source_system_code =
fact_inv_transaction.source_system_code
   AND dim_effective_date.domain_code = fact_inv_transaction.domain_code
   AND dim_effective_date.source_system_code = ded2.source_system_code
   AND dim_effective_date.domain_code = ded2.domain_code
   AND dim_effective_date.effective_fin_month = ded2.effective_fin_month
   GROUP BY fact_inv_transaction.source_system_code
    , fact_inv_transaction.domain_code
    , dim_effective_date.dim_effective_date_key
    , dim_effective_date.effective_fin_month
    , dim_effective_date.effective_fin_day_in_month
  UNION
     SELECT
      fact_inv_transaction.source_system_code,
      fact_inv_transaction.domain_code,
      ISNULL(MAX(ded2.dim_effective_date_key),0) dim_effective_date_key,
      MAX(ded2.effective_date),
      ISNULL(ded2.effective_fin_month,0) effective_fin_month,
      ISNULL(max(ded2.effective_fin_day_in_month),0) effective_fin_day_in_month,
      @v_dss_update_time
      FROM fact_inv_transaction,
           (SELECT source_system_code, domain_code, MAX(fiscal_period)
max_fin_month
            FROM fact_inv_mth_balance
            GROUP BY source_system_code, domain_code) max_fm,
          dim_effective_date,
          dim_effective_date ded2
   WHERE dim_effective_date.effective_date =
DATEADD(dd,DATEDIFF(dd,0,@v_INV_LAST_IN_MSTR_EXTRACT),0)
     AND dim_effective_date.source_system_code =
fact_inv_transaction.source_system_code
     AND dim_effective_date.domain_code = fact_inv_transaction.domain_code
     AND dim_effective_date.source_system_code = max_fm.source_system_code
     AND dim_effective_date.domain_code = max_fm.domain_code
     AND max_fm.source_system_code = ded2.source_system_code
     AND max_fm.domain_code = ded2.domain_code
     AND max_fm.max_fin_month <= ded2.effective_fin_month
     AND ded2.effective_fin_month < dim_effective_date.effective_fin_month
     GROUP BY fact_inv_transaction.source_system_code
      , fact_inv_transaction.domain_code
      , dim_effective_date.dim_effective_date_key
      , ded2.effective_fin_month
      , dim_effective_date.effective_fin_day_in_month
```

Live Support Issue



Course Review



Cover any areas the class still would like more explanation for.

Exam Review

