test1

• Stage source data in QVD files and then load from the QVD as this will avoid strain on the source systems and possibly network bandwidth as well as be a lot quicker, safer and more productive • Break out different data source load process into different script sections and use an Exit Script section, which can be easily moved to test each of your load processes separately • If possible, develop with a meaningful subset of data using Where clauses and/or Exists clauses in the load process to	• Stage source data in QVD files and then load from the QVD as this will avoid strain on the source systems and possibly network bandwidth as well as be a lot quicker, safer and more productive • Break out different data source load process into different script sections and use an Exit Script section, which can be easily moved to test each of your load processes separately • If possible, develop with a meaningful subset of data using Where clauses and/or Exists clauses in the load process to	• Stage source data in QVD files and then load from the QVD as this will avoid strain on the source systems and possibly network bandwidth as well as be a lot quicker, safer and more productive • Break out different data source load process into different script sections and use an Exit Script section, which can be easily moved to test each of your load processes separately • If possible, develop with a meaningful subset of data using Where clauses and/or Exists clauses in the load process to	• Stage source data in QVD files and then load from the QVD as this will avoid strain on the source systems and possibly network bandwidth as well as be a lot quicker, safer and more productive • Break out different data source load process into different script sections and use an Exit Script section, which can be easily moved to test each of your load processes separately • If possible, develop with a meaningful subset of data using Where clauses and/or Exists clauses in the load process to

test1 (cont)

\\eudvmms-qs501\dev-\1000.Dat-a_QVD_-Layer\1.Q-VD\1.Extr-act\QV-_QVD_B-efore\	\\eudvmms-qs501\dev-\1000.Dat-a_QVD_-Layer\1.Q-VD\1.Extr-act\QV-_QVD_B-efore\

Test 2

• Stage source data in QVD files and then load from the QVD as this will avoid strain on the source systems and possibly network bandwidth as well as be a lot quicker, safer and more productive • Break out different data source load process into different script sections and use an Exit Script section, which can be easily moved to test each of your load processes separately • If possible, develop with a meaningful subset of data using Where clauses and/or Exists clauses in the load process to ensure you maintain relevant key matches • Avoid trying to create overly large applications covering multiple use cases, it is far more efficient to create several smaller applications each covering a discrete user journey • Remove synthetic keys and where possible and circular references • Remove (or comment out a better practice) all unused fields from the load • Remove or simplify time stamps (for example you don’t need 1/100th of a second so you could use the ceil function to round up to the nearest minute) or highly unique system fields • Use Limited Load in debug mode to test your logic of the script before running a full reload or use the First function to limit the load • Use Autonumber to replace text string based key

ensure you maintain relevant key matches • Avoid trying to create overly large applications covering multiple use cases, it is far more efficient to create several smaller applications each covering a discrete user journey • Remove synthetic keys and where possible and circular references • Remove (or comment out a better practice) all unused fields from the load • Remove or simplify time stamps (for example you don't need 1/100th of a second so you could use the ceil function to round up to the nearest minute) or highly unique system fields • Use Limited Load in debug mode to test	ensure you maintain relevant key matches • Avoid trying to create overly large applications covering multiple use cases, it is far more efficient to create several smaller applications each covering a discrete user journey • Remove synthetic keys and where possible and circular references • Remove (or comment out a better practice) all unused fields from the load • Remove or simplify time stamps (for example you don't need 1/100th of a second so you could use the ceil function to round up to the nearest minute) or highly unique system fields • Use Limited Load in debug mode to test	ensure you maintain relevant key matches • Avoid trying to create overly large applications covering multiple use cases, it is far more efficient to create several smaller applications each covering a discrete user journey • Remove synthetic keys and where possible and circular references • Remove (or comment out a better practice) all unused fields from the load • Remove or simplify time stamps (for example you don't need 1/100th of a second so you could use the ceil function to round up to the nearest minute) or highly unique system fields • Use Limited Load in debug mode to test	ensure you maintain relevant key matches • Avoid trying to create overly large applications covering multiple use cases, it is far more efficient to create several smaller applications each covering a discrete user journey • Remove synthetic keys and where possible and circular references • Remove (or comment out a better practice) all unused fields from the load • Remove or simplify time stamps (for example you don't need 1/100th of a second so you could use the ceil function to round up to the nearest minute) or highly unique system fields • Use Limited Load in debug mode to test

fields with more efficient integers • Remove, join or concatenate unnecessary snow flaked tables • Avoid using nested if statements – alternatives are mapping tables in the load script and pick (match functions and Set Analysis with flag fields in the User Interface • Consider the use of incremental loads for large data sets that need to be regularly updated, this will reduce the load on the source system and speed up the overall load process

your logic of the script before running a full reload or use the First function to limit the load • Use Autonumber to replace text string based key fields with more efficient integers • Remove, join or concatenate unnecessary snow flaked tables • Avoid using nested if statements – alternatives are mapping tables in the load script and pick (match functions and Set Analysis with flag fields in the User Interface • Consider the use of incremental loads for large data sets that need to be regularly updated, this will reduce the load on the source system and speed up the overall	your logic of the script before running a full reload or use the First function to limit the load • Use Autonumber to replace text string based key fields with more efficient integers • Remove, join or concatenate unnecessary snow flaked tables • Avoid using nested if statements – alternatives are mapping tables in the load script and pick (match functions and Set Analysis with flag fields in the User Interface • Consider the use of incremental loads for large data sets that need to be regularly updated, this will reduce the load on the source system and speed up the overall	your logic of the script before running a full reload or use the First function to limit the load • Use Autonumber to replace text string based key fields with more efficient integers • Remove, join or concatenate unnecessary snow flaked tables • Avoid using nested if statements – alternatives are mapping tables in the load script and pick (match functions and Set Analysis with flag fields in the User Interface • Consider the use of incremental loads for large data sets that need to be regularly updated, this will reduce the load on the source system and speed up the overall	your logic of the script before running a full reload or use the First function to limit the load • Use Autonumber to replace text string based key fields with more efficient integers • Remove, join or concatenate unnecessary snow flaked tables • Avoid using nested if statements – alternatives are mapping tables in the load script and pick (match functions and Set Analysis with flag fields in the User Interface • Consider the use of incremental loads for large data sets that need to be regularly updated, this will reduce the load on the source system and speed up the overall

load process	load process	load process	load process

By **Abbay Kutte**
cheatography.com/abbay-kutte/

Not published yet.
Last updated 20th November, 2019.