CJIS

State of Connecticut
Criminal Justice Information System

Connecticut Information Sharing System (CISS) Technology Workshop 1: Data Replication/ETL

August 23, 2012

Data Replication/ETL: Terms

Data Replication: Data Replication is the process of copying the data from a certain data source to another source while maintaining identical copies of the data that are synchronized.

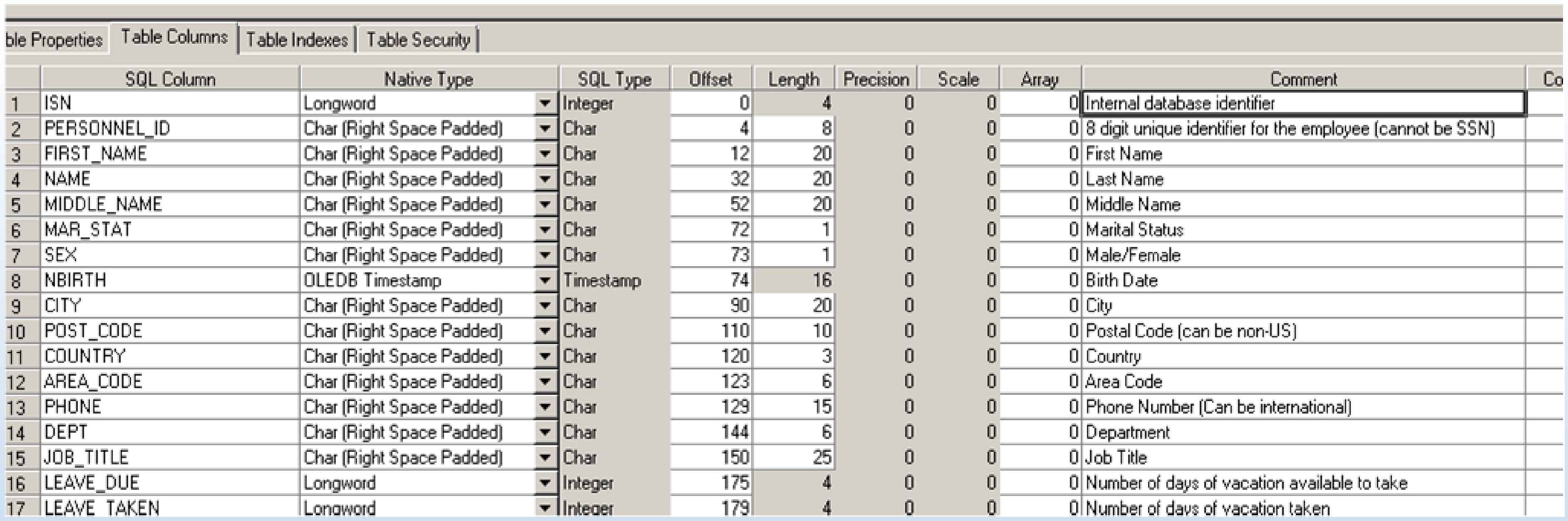
Extraction, Transformation, and Load (ETL): ETL is the process of extracting data from an environment, transforming elements, and loading the data into an environment.

Data Environments

Mainframe						Mid Tier			Desktops
				VSE		AS/400	VMS	UNIX/Linux	WINDOWS
Adabas	VSAM	IMS	DB2	VSAM	Adabas	DB2	RMS Rdb DBM Oracle	Adabas C-ISAM D-ISAM ORACLE SYBASE Microfocus Informix PostGres MySQL	Adabas SQL Server C-ISAM D-ISAM ORACLE SYBASE Microfocus Informix PostGres MySQL PowerFlex, DataFlex

Data structures have transformed since the storing of information began. In many cases, earlier file-based data still exists today. This is most common with IBM and DEC (Compaq/HP) environments. The ability to access information, from any type of data environment and to replicate it into a common structure, is vital to the success of CISS. The CISS community currently supports data structures ranging from flat and indexed files on IBM and HP mainframe/super-mini systems to databases including SQL Server, Oracle, Microsoft Access; there is even a Lotus Notes application within our world.

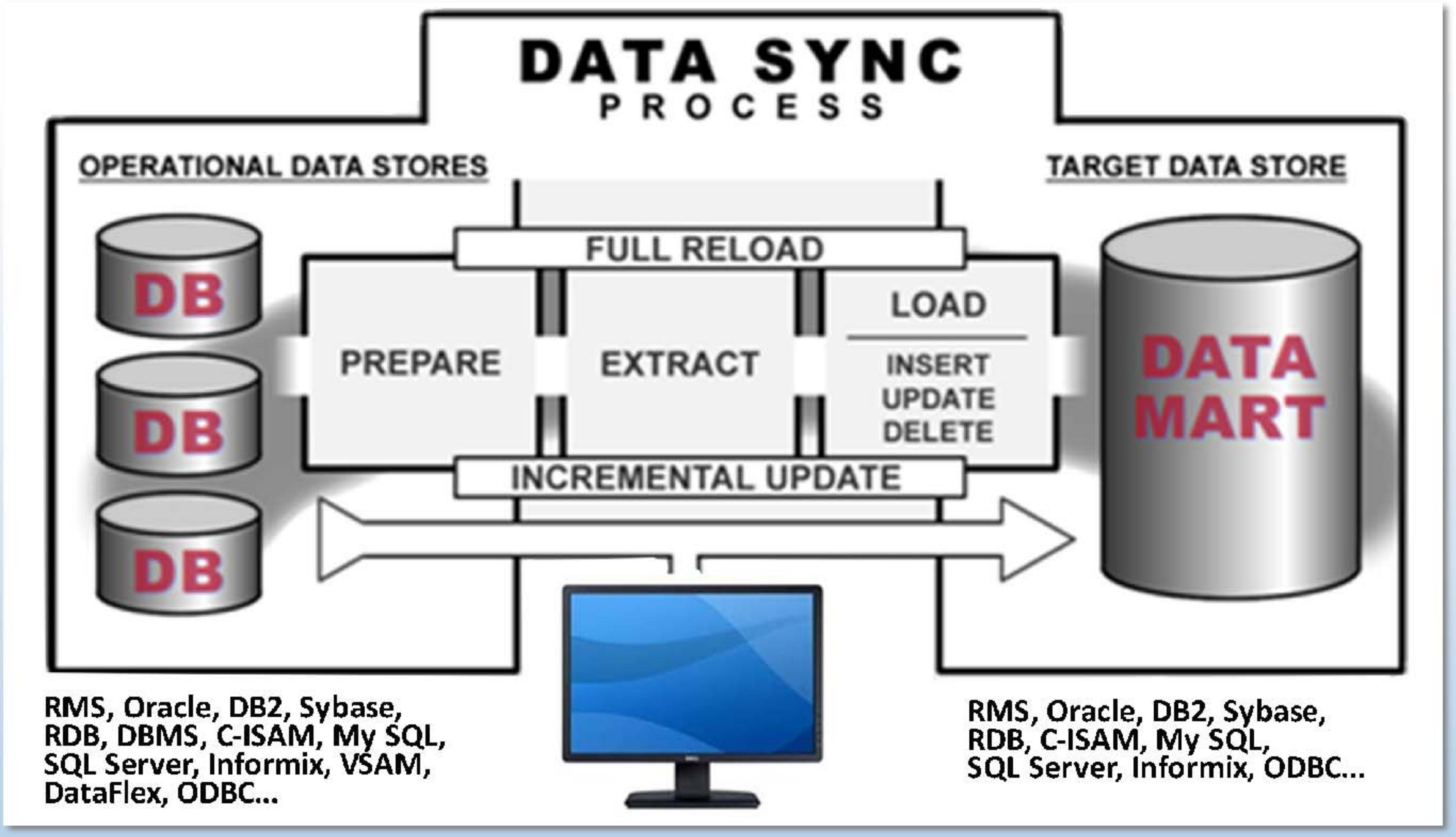
Document Your Data



ble Properties | Table Columns | Table Indexes | Table Security

	SQL Column	Native Type	SQL Type	Offset	Length	Precision	Scale	Array	Comment	Co
1	ISN	Longword	Integer	0	4	0	0	0	Internal database identifier	
2	PERSONNEL_ID	Char (Right Space Padded)	Char	4	8	0	0	0	8 digit unique identifier for the employee (cannot be SSN)	
3	FIRST_NAME	Char (Right Space Padded)	Char	12	20	0	0	0	First Name	
4	NAME	Char (Right Space Padded)	Char	32	20	0	0	0	Last Name	
5	MIDDLE_NAME	Char (Right Space Padded)	Char	52	20	0	0	0	Middle Name	
6	MAR_STAT	Char (Right Space Padded)	Char	72	1	0	0	0	Marital Status	
7	SEX	Char (Right Space Padded)	Char	73	1	0	0	0	Male/Female	
8	NBIRTH	OLEDB Timestamp	Timestamp	74	16	0	0	0	Birth Date	
9	CITY	Char (Right Space Padded)	Char	90	20	0	0	0	City	
10	POST_CODE	Char (Right Space Padded)	Char	110	10	0	0	0	Postal Code (can be non-US)	
11	COUNTRY	Char (Right Space Padded)	Char	120	3	0	0	0	Country	
12	AREA_CODE	Char (Right Space Padded)	Char	123	6	0	0	0	Area Code	
13	PHONE	Char (Right Space Padded)	Char	129	15	0	0	0	Phone Number (Can be international)	
14	DEPT	Char (Right Space Padded)	Char	144	6	0	0	0	Department	
15	JOB_TITLE	Char (Right Space Padded)	Char	150	25	0	0	0	Job Title	
16	LEAVE_DUE	Longword	Integer	175	4	0	0	0	Number of days of vacation available to take	
17	LEAVE_TAKEN	Longword	Integer	179	4	0	0	0	Number of days of vacation taken	

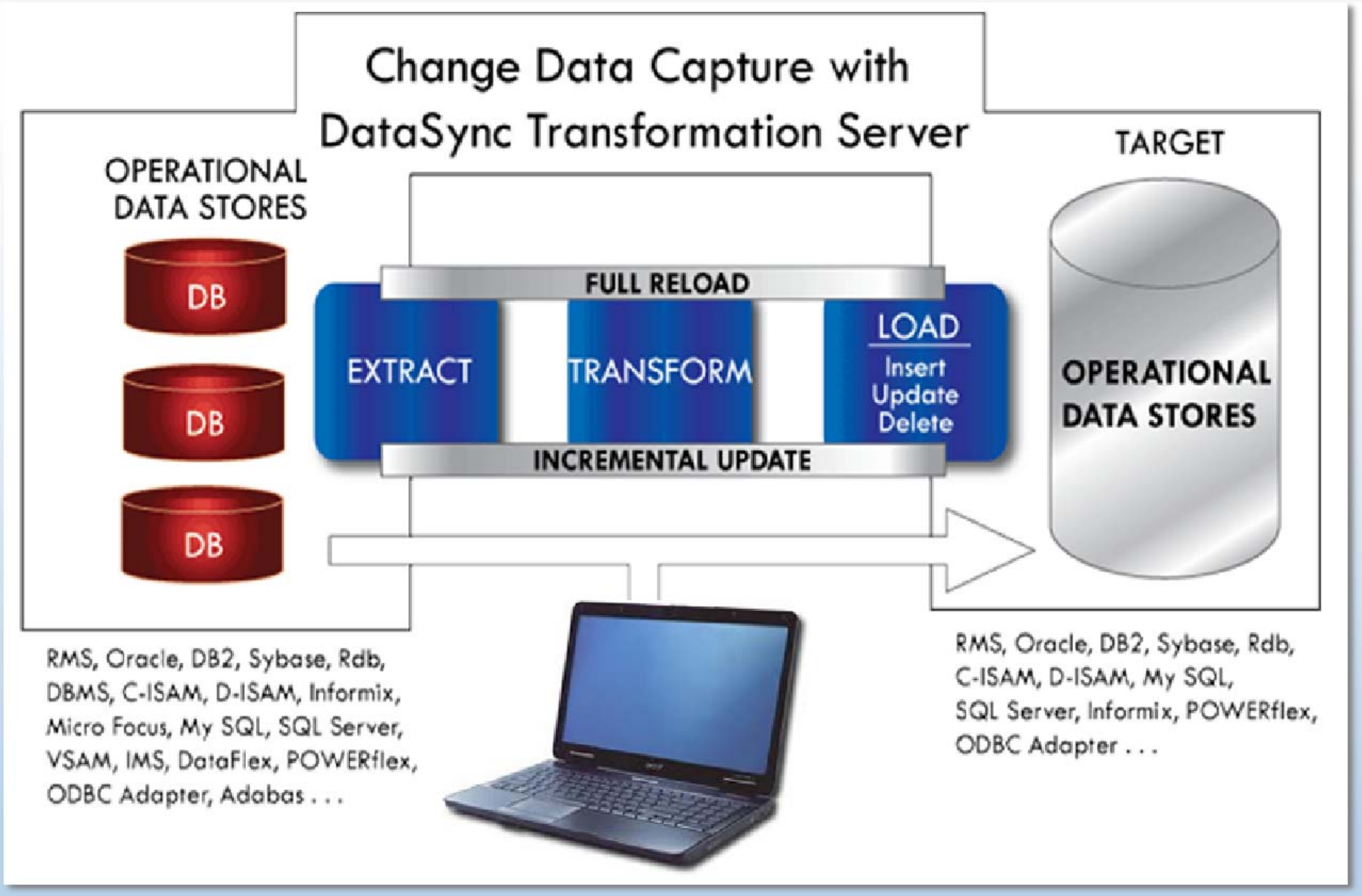
Move Data to the Right Database



RMS, Oracle, DB2, Sybase, RDB, DBMS, C-ISAM, My SQL, SQL Server, Informix, VSAM, DataFlex, ODBC...

RMS, Oracle, DB2, Sybase, RDB, C-ISAM, My SQL, SQL Server, Informix, ODBC...

Transforms Data as it Moves it


Change Data Capture with
DataSync Transformation Server
OPERATIONAL DATA STORES
DB
DB
DB
FULL RELOAD
EXTRACT
TRANSFORM
LOAD
Insert
Update
Delete
INCREMENTAL UPDATE
TARGET
OPERATIONAL DATA STORES
RMS, Oracle, DB2, Sybase, Rdb, DBMS, C-ISAM, D-ISAM, Informix, Micro Focus, My SQL, SQL Server, VSAM, IMS, DataFlex, POWERflex, ODBC Adapter, Adabas . . .
RMS, Oracle, DB2, Sybase, Rdb, C-ISAM, D-ISAM, My SQL, SQL Server, Informix, POWERflex, ODBC Adapter . . .

Replication/ETL Requirements

- Linking connected/disconnected “Clouds of data” into unified target environment(s) with or without naming transformations
- Supports direct integration of legacy data into any relational database
- Facilitates Automated Data Synchronizing
  - Highly customizable
  - Automated failure control
  - Unlimited capabilities including scripting, triggers, external applications
  - Immediate integration with Niem naming conventions via templates
- Eliminates resources (coding)
  - No legacy coding structures
  - No user-developed coding points of failure
  - Widely utilized technologies (adopted by Federal and State Agencies)
  - Facilitates RAD concepts
  - Single point of maintenance/configuration
- Automates information streams (indexes, partial, incremental, full)
- Supports most commonly used DBs and data file structures
  - Oracle, SQL Server, RMS, DB2, VSAM, ISAM, Sequential, Indexed, Adabase, Lotus Notes
- Group entities (agencies, tables, files, etc.)

Replication/ETL Requirements, cont'd.

Data Replication/ETL

Data replication/ETL is a vital component of the CISS architecture in that it enables the indexing of agency data elements from a common, secured environment. The ability to index replicated data supports the ability for Microsoft's FAST product to build and update an index using a singular data structure, SQL Server.

Relevance to CISS

One of the primary goals of CISS is to enable an individual, with proper credentials, to rapidly retrieve searched data across a spectrum of diverse information from 200+ CISS stakeholder business systems. The implementation of a well-designed and structured architecture that replicates, maps to NIEM, indexes, and presents data from such an expansive environment with <5 second response time will ensure this goal is achieved.

Replication Options

There are three options from which Agency Stakeholders can choose to support CISS searching their data environments. The three options give our CISS community flexibility to decide what method to use, each with differing levels of complexity for integration.

- **Federated Search**
- **Agency Replicates Data**
- **Crawling of their Data**

Replication – Option 1

- Data from a stakeholder’s environment is access from CISS via web service query. The agency will be required to create Web Service interfaces to be used by CISS. These services will respond to query requests from CISS, which will generate data extractions via views, stored procedures, or other methods an agency prefers to use.
- The selected data will then be returned via the original request, synchronously. Each table to be searched by CISS will require a distinct Web Service.
- **This option requires the most effort by the agency** and CISS and impacts the ability of CISS to respond to a query request rapidly. In effect, CISS does not recommend this option unless it is absolutely necessary to interface in this manner. For each search request made by a user within CISS, each of the agencies will be required to respond to a search request via web services and respond with the appropriate data. These queries are for initial search requests and detail requests.
- As an example: if there are 5,000 search requests (initial or detail) per hour, the agencies using Option 1 will be required to respond to each request – receive the request, query their data environments, build an XML message, and send the response via web service response. This scenario will significantly impact agency storage throughput, both theirs and the State of Connecticut’s network, affects Search response times and is a significant point of maintainability and failure in the Search segment of CISS.

Replication – Option 2

- Data from a stakeholder’s environment is replicated (copied) by the agency and either
  - put on a common network drive for CISS Access,
  - placed on a FTP site for CISS to retrieve, or
  - put into another data environment where CISS can replicate the data. A schedule to support collecting the replicated data will be arranged between CISS and the respective agencies.
- This option requires an effort by an agency to provide a mechanism to extract portions of their data into one of several structures (Database, index or flat file, XML, etc.) and to make the data available to CISS.
- Disadvantages to this option include delays in indexing and searching stale data, storage requirements for the agency to “hold” the replicated data and the repetitive process of building container(s) to retrieve updated data records.
- If the agency cannot identify the changed records efficiently, then the entire database (only necessary fields) will need to be replicated, repetitively, throughout the day.
- Lastly, the overhead and impact to re-index all the data from an agency will impact the agency’s and CISS’s systems and the State of Connecticut’s Network.

Replication – Option 3

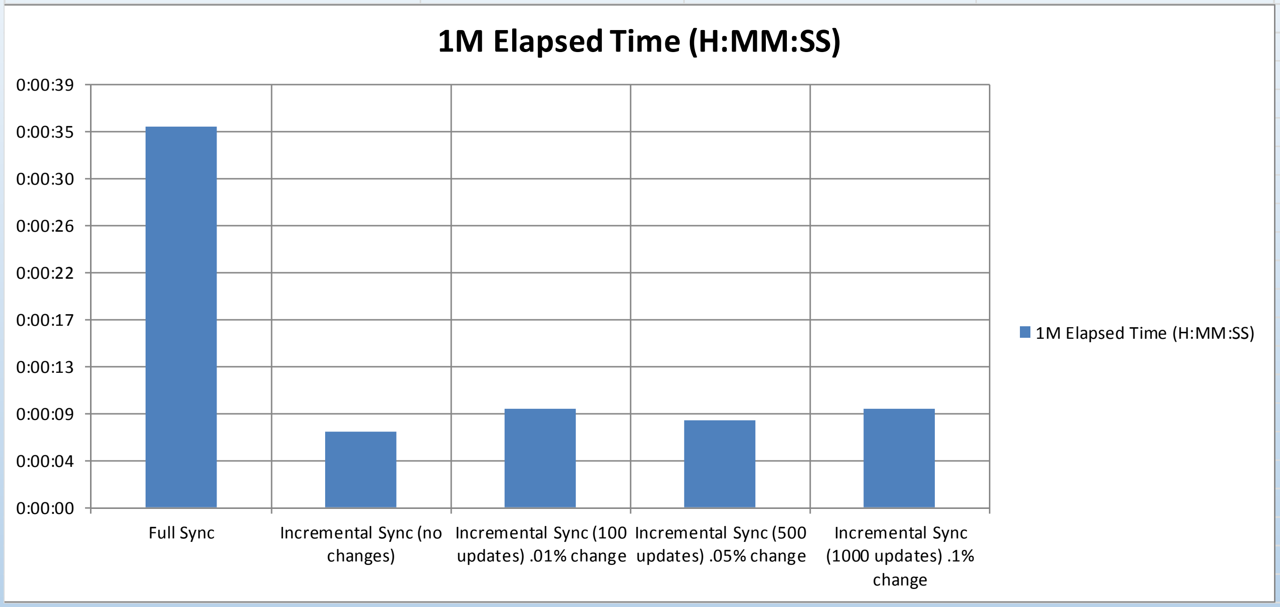
- **This option is the most efficient scenario, and lowest impact to the stakeholder, CISS and the State of Connecticut's networks.**
- It offers the ability to support agency governance of data, encryption of data, minimized network traffic, a single point of configuration, auditable, and secure.
- To support this option, an agency works with CISS to:
  - identify the pertinent data to be replicated (tables, fields, data files, etc.),
  - provides a User ID/Password with Read-Only access to their data environment (SQL, Oracle, O/S) and
  - depending on their data environment, may be required to install a listener service. The listener service applies to VAX/Alpha systems and IBM environments using VSAM/ISAM for their data file structures. Oracle and SQL Server environments require only access to their respective Database's IP Port Number (Oracle=1521, SQL Server=1433)
- This option enables CISS to scan the agency's selected data environments on a pre-determined schedule. Data environment which have high frequencies of updated data will be scanner frequently and those environments which have minor updates or are stale will be scanned infrequently. The frequency in either scenario depends on the size of the database, the nature of the data and necessity to have visibility to the data.

- Bottom Line: Replicating data has an impact on systems and networks.

- To minimize the performance impact caused by replication, it is imperative that agencies understand these three options.

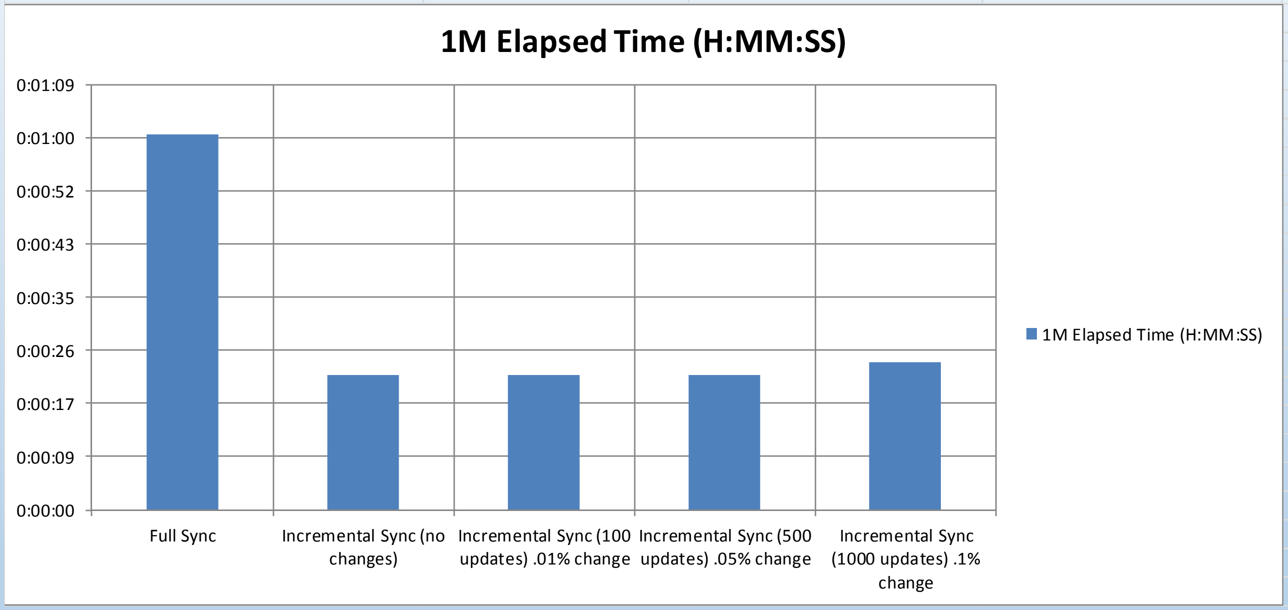
SQL Server to SQL Server (remote)	
Source PC Specificaion	Intel Core2 Extreme X9650 @ 3 Ghz, 8GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008 64bit
Destination PC Specification	Intel Xeon X3565 Extreme X9650 @ 3.2 Ghz, 12GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008R2 64bit
Source Database	SQL Server 2008R2
Target Database	SQL Server 2008R2
Table Schema:	create table sql2008R2..onemillionrowtable(id char(10), name char(40), createdate date, sold decimal(11,2), ordercount integer, notes varchar(115))
	create unique index blahone on sql2008R2..onemillionrowtable(id)
Table Width	197 bytes
Table Rows	1M
Table Columns	5

	1M Elapsed Time (H:MM:SS)	DDL for Update
Full Sync	0:00:35	
Incremental Sync (no changes)	0:00:07	
Incremental Sync (100 updates) .01% change	0:00:09	update SQL2008R2..onemillionrowtable set createdate = now() , sold = 1234.56 where id between '0000100000' and '0000100099'
Incremental Sync (500 updates) .05% change	0:00:08	update SQL2008R2..onemillionrowtable set createdate = now() , sold = 7234.56 where id between '0000100000' and '0000100499'
Incremental Sync (1000 updates) .1% change	0:00:09	update SQL2008R2..onemillionrowtable set createdate = now() , sold = 6234.56 where id between '0000100000' and '0000100999'



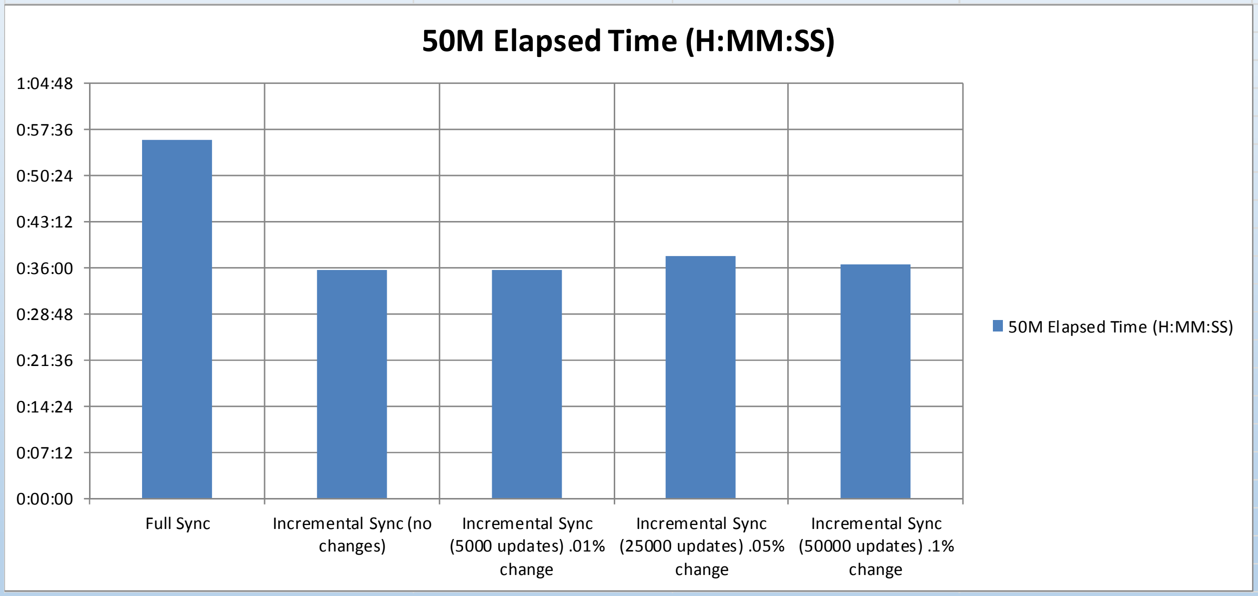
RMS to SQL Server (remote)	Itanium VMS HP rx1600 1GHz						
Source PC Specificaion	Intel Core2 Extreme X9650 @ 3 Ghz, 8GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008 64bit						
Destination PC Specification	Intel Xeon X3565 Extreme X9650 @ 3.2 Ghz, 12GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008R2 64bit						
Source Database	RMS (VMS Itanium)						
Target Database	SQL Server 2008R2						
Table Schema:	converted from RMS flat file - unique index on ID		id	Text (Right Space Padded)	Char	0	10
			name	Text (Right Space Padded)	Char	10	40
			createdate	Text Date (YYYYMMDD)	Date	50	8
Table Width	196 bytes		sold	Zoned Numeric -> Decimal	Decimal	58	6
Table Rows	1M		ordercount	Zoned Numeric -> Decimal	Decimal	64	7
Table Columns	5		notes	Text (Right Space Padded)	Char	71	125

	1M Elapsed Time (H:MM:SS)	DDL for Update
Full Sync	0:01:01	
Incremental Sync (no changes)	0:00:22	
Incremental Sync (100 updates) .01% change	0:00:22	update rms..onemillionrowtable set createdate = now() , sold = 1234.56 where id between '0000100000' and '0000100099'
Incremental Sync (500 updates) .05% change	0:00:22	update rms..onemillionrowtable set createdate = now() , sold = 7234.56 where id between '0000100000' and '0000100499'
Incremental Sync (1000 updates) .1% change	0:00:24	update rms..onemillionrowtable set createdate = now() , sold = 6234.56 where id between '0000100000' and '0000100999'



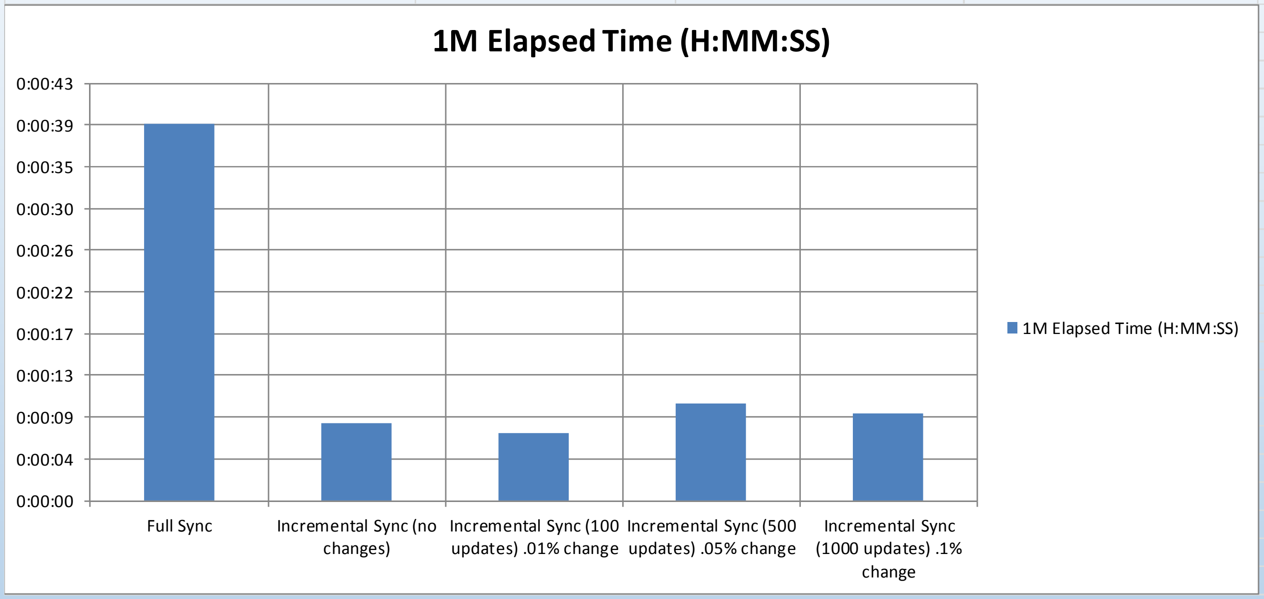
RMS to SQL Server (remote)	Itanium VMS HP rx1600 1GHz						
Source PC Specificaion	Intel Core2 Extreme X9650 @ 3 Ghz, 8GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008 64bit						
Destination PC Specification	Intel Xeon X3565 Extreme X9650 @ 3.2 Ghz, 12GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008R2 64bit						
Source Database	RMS (VMS Itanium)						
Target Database	SQL Server 2008R2						
Table Schema:	converted from RMS flat file - unique index on ID		id	Text (Right Space Padded)	Char	0	10
			name	Text (Right Space Padded)	Char	10	40
			createdate	Text Date (YYYYMMDD)	Date	50	8
Table Width	196 bytes		sold	Zoned Numeric -> Decimal	Decimal	58	6
Table Rows	50M		ordercount	Zoned Numeric -> Decimal	Decimal	64	7
Table Columns	5		notes	Text (Right Space Padded)	Char	71	125

	50M Elapsed Time (H:MM:SS)	DDL for Update
Full Sync	0:55:51	
Incremental Sync (no changes)	0:35:39	
Incremental Sync (5000 updates) .01% change	0:35:39	update rms..fiftymillionrowtable set createdate = now() , sold = 1234 where id between '0010000000' and '0010004999'
Incremental Sync (25000 updates) .05% change	0:37:40	update rms..fiftymillionrowtable set createdate = now() , sold = 7234 where id between '0010000000' and '0010024999'
Incremental Sync (50000 updates) .1% change	0:36:26	update rms..fiftymillionrowtable set createdate = now() , sold = 6234 where id between '0010000000' and '0010049999'



DB2 To SQL Server (remote)	
Source PC Specificaion	Intel Core2 Extreme X9650 @ 3 Ghz, 8GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008 64bit
Destination PC Specification	Intel Xeon X3565 Extreme X9650 @ 3.2 Ghz, 12GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008R2 64bit
Source Database	DB2 9.7.1
Target Database	SQL Server 2008R2
Table Schema:	create table db2..onemillionrowtable(id char(10), name char(40), createdate date, sold decimal(11,2), ordercount integer, notes varchar(115))
	create unique index blahone on db2..onemillionrowtable(id)
Table Width	197 bytes
Table Rows	1M
Table Columns	5

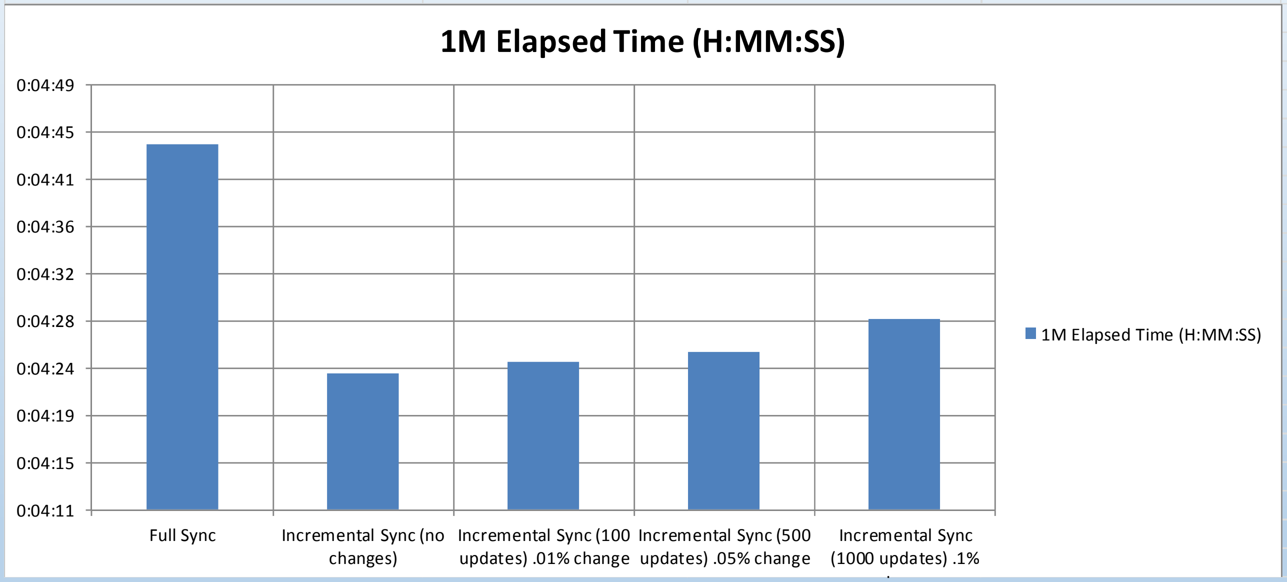
	1M Elapsed Time (H:MM:SS)	DDL for Update
Full Sync	0:00:39	
Incremental Sync (no changes)	0:00:08	
Incremental Sync (100 updates) .01% change	0:00:07	update db2..onemillionrowtable set createdate = now() , sold = 1234.56 where id between '0000100000' and '0000100099'
Incremental Sync (500 updates) .05% change	0:00:10	update db2..onemillionrowtable set createdate = now() , sold = 7234.56 where id between '0000100000' and '0000100499'
Incremental Sync (1000 updates) .1% change	0:00:09	update db2..onemillionrowtable set createdate = now() , sold = 6234.56 where id between '0000100000' and '0000100999'



VSAM to SQL Server (remote)					
Source Mainframe Specificaion	z/OS version 1 release 12 level 1009, 5 MSU, Batch				
Destination PC Specification	Intel Xeon X3565 Extreme X9650 @ 3.2 Ghz, 12GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008R2 64bit				
DataSync PC Specificaion	Intel Core2 Extreme X9650 @ 3 Ghz, 8GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008 64bit				
Source Database	VSAM				
Target Database	SQL Server 2008R2				

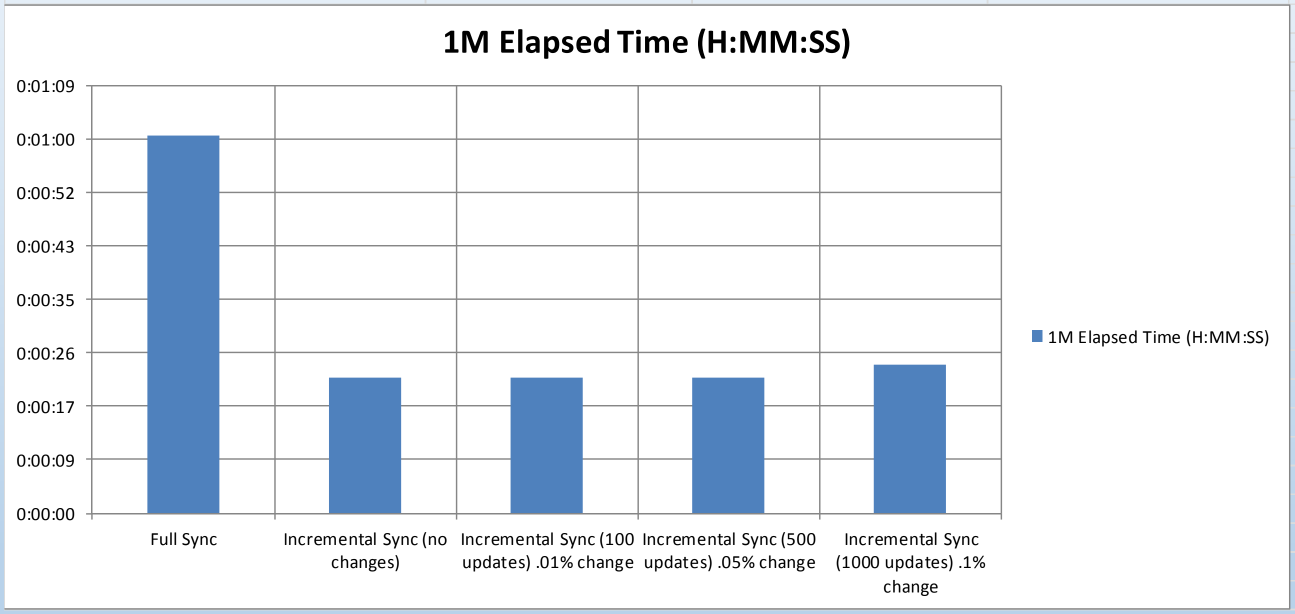
Table Schema:	Repro'd from qsam flat file - unique index on ID	id	Text (Right Space Padded)	Char	0	10
		name	Text (Right Space Padded)	Char	10	40
		createdate	Text Date (YYYYMMDD)	Date	50	8
Table Width	200 bytes	sold	Zoned Numeric -> Decimal	Decimal	58	6
Table Rows	1M	ordercount	Zoned Numeric -> Decimal	Decimal	64	7
Table Columns	5	notes	Text (Right Space Padded)	Char	71	125

	1M Elapsed Time (H:MM:SS)	DDL for Update
Full Sync	0:04:44	
Incremental Sync (no changes)	0:04:23	
Incremental Sync (100 updates) .01% change	0:04:24	update vsam..onemillionrowtable set createdate = now() , sold = 1234.56 where id between '0000100000' and '0000100099'
Incremental Sync (500 updates) .05% change	0:04:25	update vsam..onemillionrowtable set createdate = now() , sold = 7234.56 where id between '0000100000' and '0000100499'
Incremental Sync (1000 updates) .1% change	0:04:28	update vsam..onemillionrowtable set createdate = now() , sold = 6234.56 where id between '0000100000' and '0000100999'



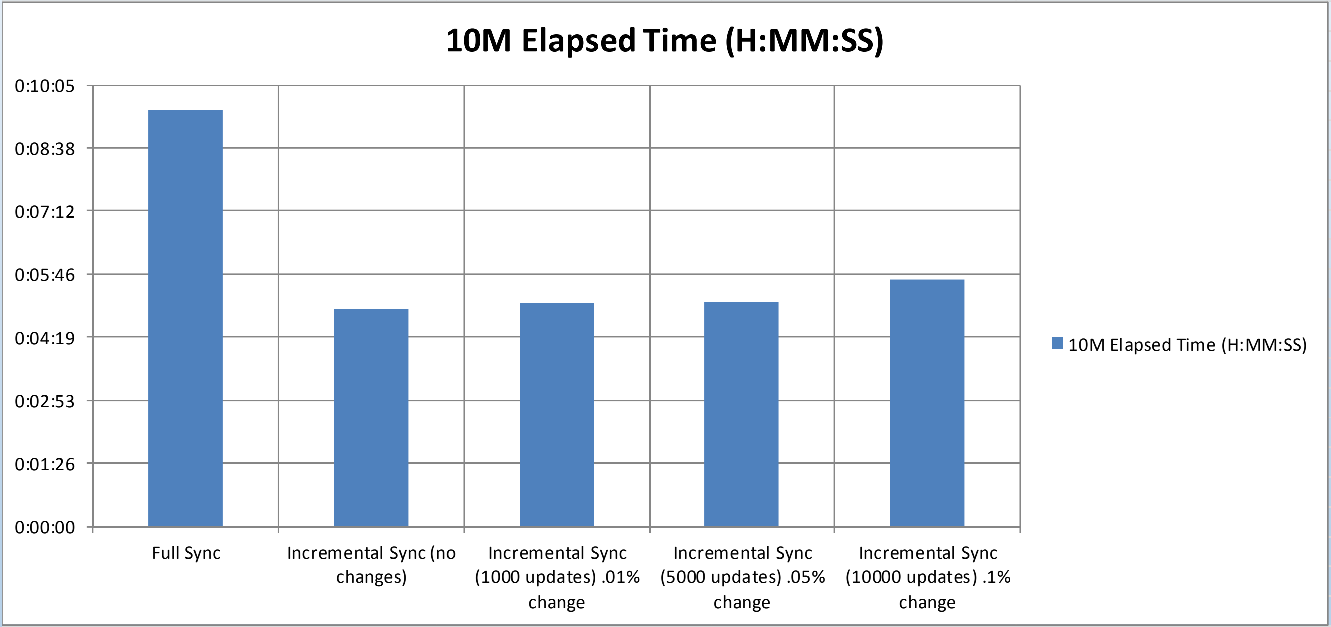
Oracle to SQL Server (remote)	
Source PC Specificaion	Intel Core2 Extreme X9650 @ 3 Ghz, 8GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008 64bit
Destination PC Specification	Intel Xeon X3565 Extreme X9650 @ 3.2 Ghz, 12GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008R2 64bit
Source Database	Oracle 11i
Target Database	SQL Server 2008R2
Table Schema:	create table orcl..onemillionrowtable(id char(10), name char(40), createdate date, sold decimal(11,2), ordercount integer, notes varchar(115))
	create unique index blahone on orcl..onemillionrowtable(id)
Table Width	197 bytes
Table Rows	1M
Table Columns	5

	1M Elapsed Time (H:MM:SS)	DDL for Update
Full Sync	0:01:01	
Incremental Sync (no changes)	0:00:22	
Incremental Sync (100 updates) .01% change	0:00:22	update orcl..onemillionrowtable set createdate = now() , sold = 1234.56 where id between '0000100000' and '0000100099'
Incremental Sync (500 updates) .05% change	0:00:22	update orcl..onemillionrowtable set createdate = now() , sold = 7234.56 where id between '0000100000' and '0000100499'
Incremental Sync (1000 updates) .1% change	0:00:24	update orcl..onemillionrowtable set createdate = now() , sold = 6234.56 where id between '0000100000' and '0000100999'



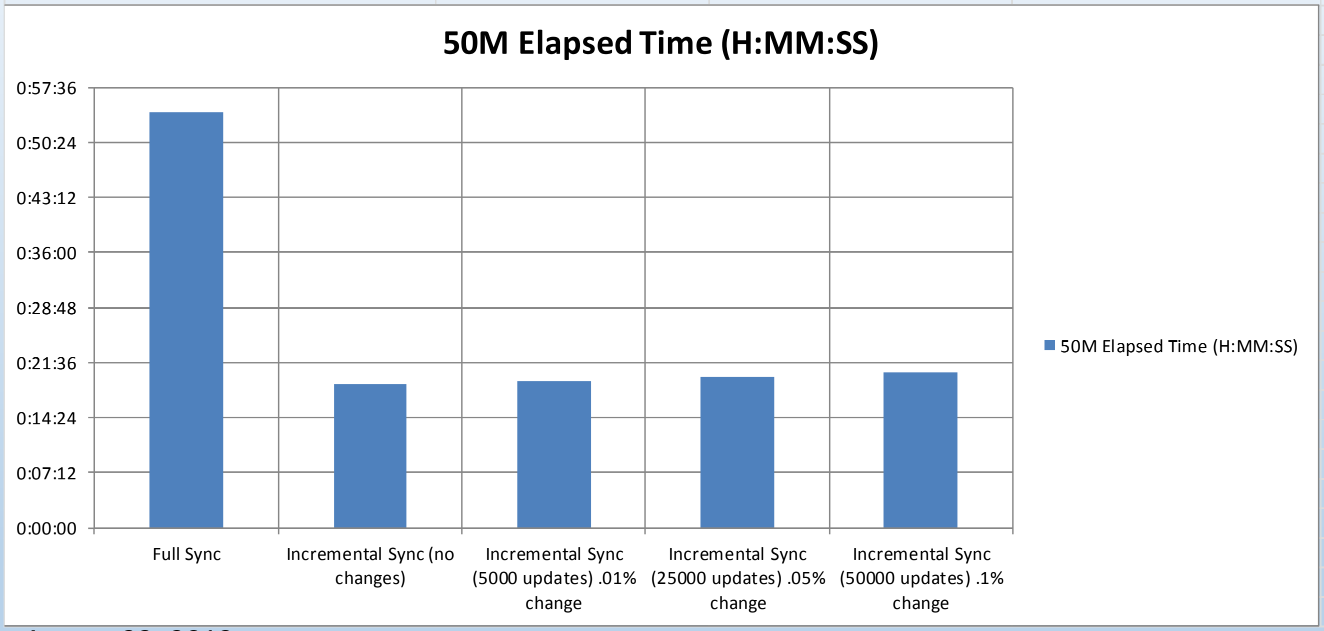
Oracle to SQL Server (remote)	
Source PC Specificaion	Intel Core2 Extreme X9650 @ 3 Ghz, 8GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008 64bit
Destination PC Specification	Intel Xeon X3565 Extreme X9650 @ 3.2 Ghz, 12GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008R2 64bit
Source Database	Oracle 11i
Target Database	SQL Server 2008R2
Table Schema:	create table orcl..tenmillionrowtable(id char(10), name char(40), createdate date, sold decimal(11,2), ordercount integer, notes varchar(115))
	create unique index blahfifty on orcl..tenmillionrowtable(id)
Table Width	197 bytes
Table Rows	10M
Table Columns	5

	10M Elapsed Time (H:MM:SS)	DDL for Update
Full Sync	0:09:31	
Incremental Sync (no changes)	0:04:57	
Incremental Sync (1000 updates) .01% change	0:05:06	update orcl..tenmillionrowtable set createdate = now() , sold = 1234.56 where id between '0000100000' and '0000100999'
Incremental Sync (5000 updates) .05% change	0:05:08	update orcl..tenmillionrowtable set createdate = now() , sold = 7234.56 where id between '0000100000' and '0000104999'
Incremental Sync (10000 updates) .1% change	0:05:39	update orcl..tenmillionrowtable set createdate = now() , sold = 6234.56 where id between '0000100000' and '0000109999'



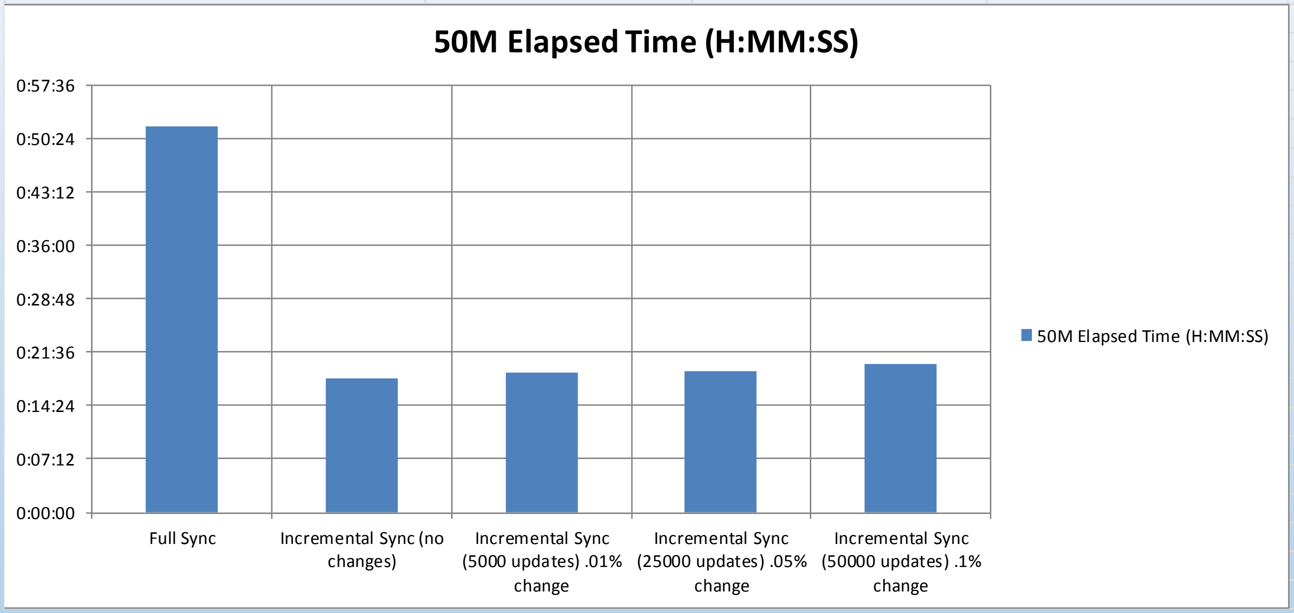
Oracle to SQL Server (local)	
Source PC Specificaion	Intel Core2 Extreme X9650 @ 3 Ghz, 8GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008 64bit
Destination PC Specification	Same as Source - Both Source & Target are on same physical PC
Source Database	Oracle 11i
Target Database	SQL Server 2008R2
Table Schema:	create table orcl..fiftymillionrowtable(id char(10), name char(40), createdate date, sold decimal(11,2), ordercount integer, notes varchar(115))
	create unique index blahfifty on orcl..fiftymillionrowtable(id)
Table Width	197 bytes
Table Rows	50M
Table Columns	5

	50M Elapsed Time (H:MM:SS)	DDL for Update
Full Sync	0:54:16	
Incremental Sync (no changes)	0:18:46	
Incremental Sync (5000 updates) .01% change	0:19:04	update orcl..fiftymillionrowtable set createdate = now() , sold = 1234.56 where id between '10000000' and '10004500'
Incremental Sync (25000 updates) .05% change	0:19:47	update orcl..fiftymillionrowtable set createdate = now() , sold = 7234.56 where id between '10000000' and '10022500'
Incremental Sync (50000 updates) .1% change	0:20:15	update orcl..fiftymillionrowtable set createdate = now() , sold = 6234.56 where id between '10000000' and '10045000'



Oracle to SQL Server (remote)	
Source PC Specificaion	Intel Core2 Extreme X9650 @ 3 Ghz, 8GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008 64bit
Destination PC Specification	Intel Xeon X3565 Extreme X9650 @ 3.2 Ghz, 12GB ram, 1TB Hard disk, 1Gbit Network, Windows 2008R2 64bit
Source Database	Oracle 11i
Target Database	SQL Server 2008R2
Table Schema:	create table orcl..fiftymillionrowtable(id char(10), name char(40), createdate date, sold decimal(11,2), ordercount integer, notes varchar(115))
	create unique index blahfifty on orcl..fiftymillionrowtable(id)
Table Width	197 bytes
Table Rows	50M
Table Columns	5

	50M Elapsed Time (H:MM:SS)	DDL for Update
Full Sync	0:51:57	
Incremental Sync (no changes)	0:18:02	
Incremental Sync (5000 updates) .01% change	0:18:46	update orcl..fiftymillionrowtable set createdate = now() , sold = 1234.56 where id between '10000000' and '10004500'
Incremental Sync (25000 updates) .05% change	0:19:05	update orcl..fiftymillionrowtable set createdate = now() , sold = 7234.56 where id between '10000000' and '10022500'
Incremental Sync (50000 updates) .1% change	0:19:59	update orcl..fiftymillionrowtable set createdate = now() , sold = 6234.56 where id between '10000000' and '10045000'



Feedback

We need your feedback —

please send us your comments, questions & suggestions.

Sean Thakkar — Sean.Thakkar@ct.gov

Mark Tezaris — Mark.Tezaris@ct.gov

Rick Ladendecker — Rick.Ladendecker@ct.gov

Nance McCauley — Nance.McCauley@ct.gov

Thank you

Appendix: Acronyms

AFIS = Automated Fingerprint Identification system
AST = Application Support System
BEST = Bureau of Enterprise Systems and Technology
BICE = Bureau of Immigration and Customs Enforcement
BOPP= Board of Pardons and Paroles
CAD = Computer Aided Dispatch
CCH= Computerized Criminal History
CIB = Centralized Infraction Bureau (Judicial)
CIB = Centralized Infractions Bureau
CIDRIS = Conn. Impaired Driver Records Information System
CISS = Conn. Information Sharing System
CIVLS = CT Integrated Vehicle & Licensing System
CJIS = Criminal Justice Information System
CJPPD = Criminal Justice Policy Development and Planning Division
CMIS = (Judicial's) Case Management Information System
COLLECT = Connecticut On-Line Law Enforcement Communications Teleprocessing network
CPCA = Conn. Police Chiefs Association
CRMVS = Criminal and Motor Vehicle System (Judicial)
CSSD =Court Support Services Division
DCJ = Division of Criminal Justice
DAS = Dept. of Administrative Services
DESPP = Department of Emergency Services & Public Protection
DEMHS = Dept of Emergency Management & Homeland Security
DMV = Dept. of Motor Vehicles
DOC = Department of Corrections
DOIT = Dept. of Information Technology
DPDS = Div. of Public Defender Services
IST = Infrastructure Support Team
JMI = Jail Management System
JUD = Judicial Branch
LEA = Law Enforcement Agency
LAW = Local Law Enforcement (e.g., DPS, CPCA)
LIMS = State Crime Laboratory Database
MNI = Master Name Index (State Police)
OBIS = Offender Based Information System (Corrections)
OBTS = Offender Based Tracking System
OVA= Office of Victim Advocacy
OVS = Office of Victim Services
RMS = Records Management System (Police Agency RMS manages & stores info on arrests, incidents)
OSET = Office of Statewide Emergency Telecommunications
POR = Protective Order Registry (Judicial)
PRAWN = Paperless Re-Arrest Warrant Network (Judicial)
PSDN = Public Safety Data Network
SCO= Superior court operations
SOR = Sex Offender Registry (Judicial)

Technology Related

COTS = Computer Off The Shelf (e.g., software)
ETL = Extraction, Transformation, and Load
IEPD = Information Exchange Package Delivery
POC = Proof of Concept
RDB = Relational Database
SDLC = Software Development Life Cycle
SOA = Service Oriented Architecture
SQL = Structured Query Language