SCI TranSSend for Oracle CODASYL DBMS

Bryan Holland / Norman Lastovica
Software Concepts International
holland@sciinc.com
lastovica@sciinc.com

What is a "CODASYL" Database?

- Initially based on understandings of computing world of 1950's and 60's…
- Relationship between record types via "Sets"
  - Physical pointers between record occurrences
- Primary API is through a procedural Data Manipulation Language (DML)

Opportunities

- Crucial business data stored in CODASYL DBMS databases on OpenVMS

- Challenging to access this information with high-performance, state-of-the-art tools
  - Ad-hoc reporting
  - Data access to downstream tools
  - Data warehousing and consolidation
  - Business intelligence
  - Auditing
  - Archiving
  - Etc.

Avoid "critical eye" on "legacy" system

Existing Techniques Insufficient

- "Scraping" - scanning vast regions of database
  - Significant IO, CPU, locking impact on production DB
  - Costly delay in access to data
  - Costly to maintain home-grown solutions
- Often searching many billions of bytes looking for a handful of changes
- Application based Data Replication models (DRS/RTR) complex and difficult-to-support technologies that impact application
- ODBC translation methods perform poorly and significantly impact database performance.
- CODASYL model is often not well suited for direct ad-hoc queries

Customer Requirements

- Rapid access for ad hoc queries & data mining
  - Separate database structure and environment
- 24x365 CODASYL DBMS OLTP databases
- Extremely large operational data stores
- Robust, high-performance data replication
  - Zero application code impact
  - Zero database structure impact
  - Minimal latency
  - Minimal or zero performance impact

A New Approach

Mirror content of CODASYL DBMS within a relational database

The challenge?

Different architectures with different design objectives

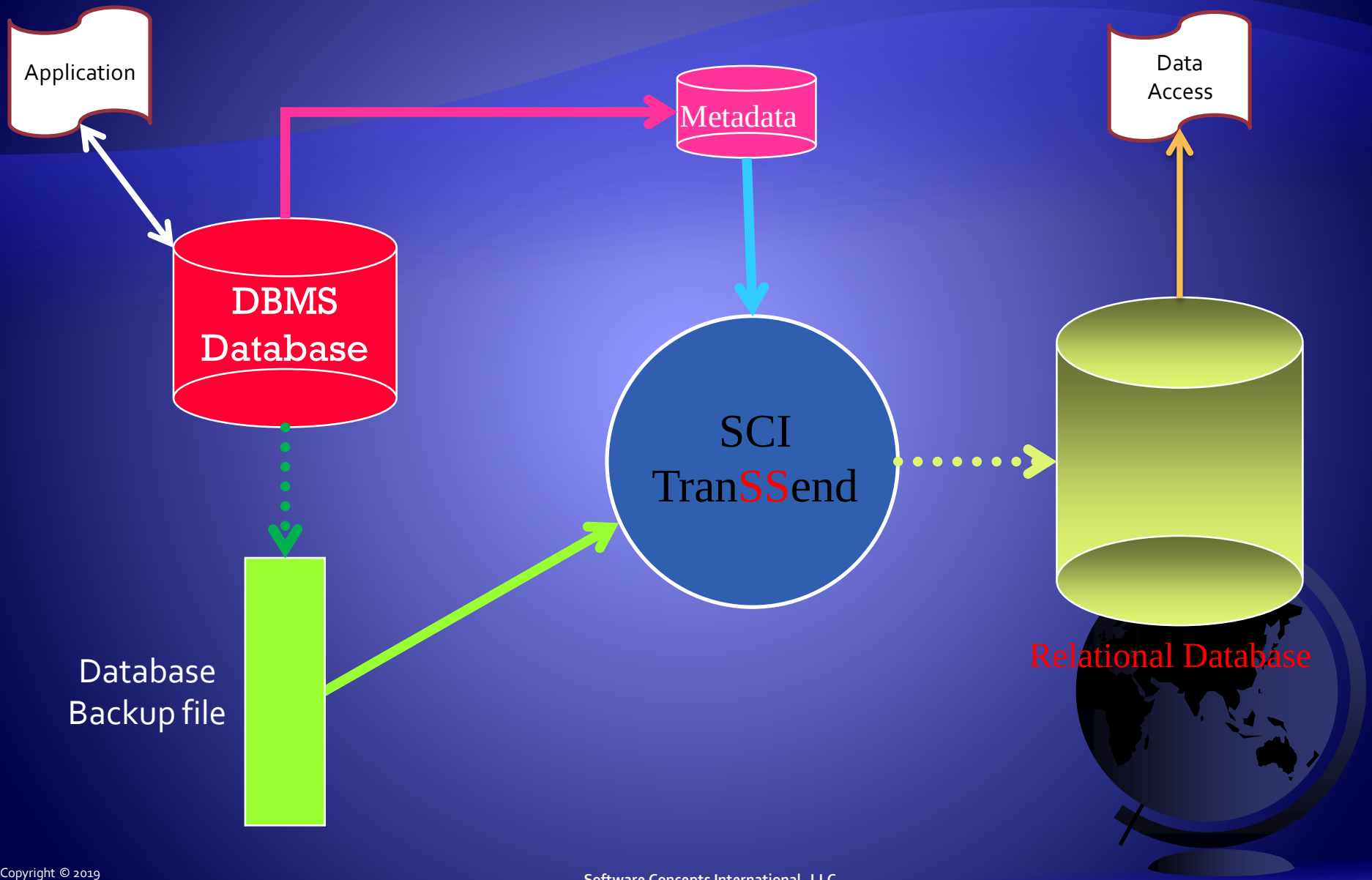
Solution

- SCI TranSSend for DBMS
  - DBMS data replication and change data capture
- Capture data with zero impact on DBMS production
  - From database backup
  - From database AIJ backup
  - From live AIJ (upcoming)
- Store in a relational database
  - E.g. Oracle Rdb, Oracle Database, MS SQL Server, SYBASE, Etc.

Replicate-All Model

- Think "DBO/RESTORE to relational database"
- Initial replication of all records/sets to relational tables from DBMS database backup
- Zero impact on production databases
  - Assumes backups are already being done

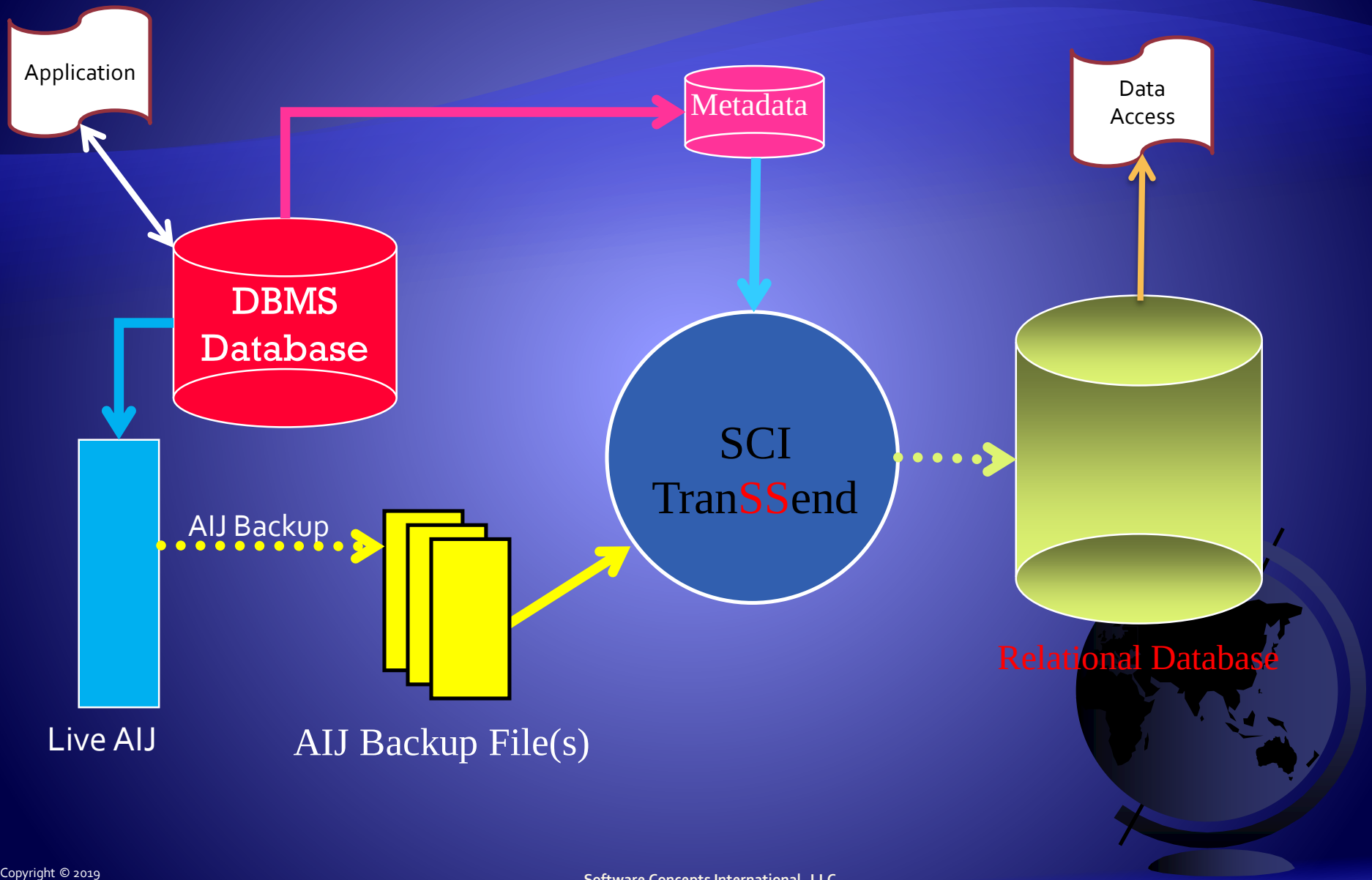
Replicate-All Model



Replicate-Changes Model

- Think "DBO/RECOVER to relational database"
- Changes (i.e. record insert, modify & erase) propagated to relational database
- Zero or minimal impact
  - AIJ backups already being done
- Lag time expressed in small number of minutes

Replicate-Changes Model



Change Replication

- Committed transactions only
  - Replicated in transaction commit sequence
  - Extracted directly from AIJ backup
- Inserts & modifies
  - 'Final' record content per transaction
- Erase
  - Data record with matching 'self' key deleted from target

Data Model

- Records → Relations
- Fields → Columns

- Every record type replicated by default
  - Discovered this is vastly preferable to "pick & choose"

- Set linkages & relationships instantiated via synthetic columns
  - Physical DBKEY of corresponding DBMS record in source database
  - SELF identifier in every relation
  - OWNER, NEXT, PRIOR for each set within a relation

Data Model

- SCI_SELF and O_*, synthetic integer columns automatically added to each relation to maintain "set" relationships from DBMS schema
- "Primary Key" and "Foreign Key"
- "NEXT" and "PRIOR" columns optional for each set

```
CREATE TABLE PART (
  SCI_SELF              NUMBER PRIMARY KEY
 ,PART_ID               VARCHAR2 (1)
 ,PART_DESC             VARCHAR2 (50)
 ,PART_STATUS           VARCHAR2 (1)
 ,O_PART_CLASS          NUMBER );
```

WorkSteam ALL_DATA Fields

- Consilium WorkStream database combines a several logical fields within a single record field
  - Prohibits easy access to data content

- SCI TranSSend supports exposing ALL_DATA field into constituent sub-fields within target relational database

- SCI offers several templates to overlay data record definitions

Triggers & Views

- Triggers on target database? Absolutely!
  - Notify down-stream systems
  - Maintain audit history
  - Maintain archival history
  - Etc.
- Views on target database? Of course!
  - Perform automatic "rollups"
  - Combine and/or hide parent-child relationships
  - Etc.
- …Along with all the rest of database features
  - Interfaces to other tools
  - Modern accessibility
    - ODBC, JDBC, SQL etc.
  - Statistical analysis
  - Very high performance data access
  - Ease-of-use
  - Data replication & protection
  - Etc.

Indexes on Target Database

- Any column combination can be indexed
  - Difficult for SCI to predict ahead of time what indexes would be required to support your business queries
  - We have some "templates"

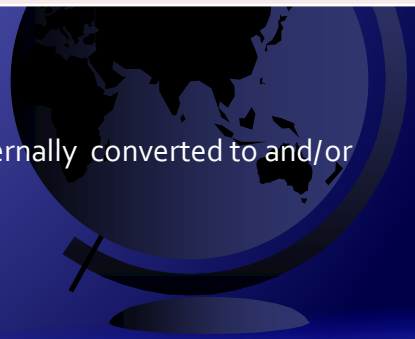
- Indexes may be freely added, removed & replaced on the target database tables
  - Zero impact on production database
  - Supporting all reporting and query requirements

Last Update TAD & TSN

- Each row in target database contains by default transaction sequence number & date/time of last update
- TSN is effectively non-ordered (e.g. higher TSN does not imply modification order vs lower TSN)
- TAD is approximate commit time
  - Daylight savings adjustments forward and backward
  - Non-synchronized across cluster nodes
  - Set to date/time of data page upon initial load

Default Data Type Mapping

DBMS type	Rdb type	Oracle type	SQL Server type
BYTE WORD LONGWORD QUADWORD	TINYINT SMALLINT INT BIGINT	NUMBER(3) NUMBER(5) NUMBER(10) NUMBER(20)	SMALLINT SMALLINT INT BIGINT
OCTAWORD PACKED DECIMAL NUMERIC	CHAR(40)/VARCHAR(40) (with optional TRIM)	NUMBER	CHAR(40)
FLOATING[1] G_FLOATING[1] D_FLOATING[1]	REAL DOUBLE PRECISION DOUBLE PRECISION	NUMBER	REAL DOUBLE PRECISION DOUBLE PRECISION
CHARACTER	CHAR or VARCHAR (with optional TRIM)	CHAR or VARCHAR2 (with optional TRIM)	CHAR or VARCHAR (with optional TRIM)
DATE	DATE VMS	TIMESTAMP[2]	DATETIME2[3]
H_FLOATING	Not presently supported	Not presently supported	Not presently supported



[1] Depending on platform and/or version and/or target database, F, G and D_FLOATING values may be internally converted to and/or through S, T, or G_FLOATING with potential slight loss of precision and/or range

[2] Default of 6 fractional digits of precision are preserved by default from VMS quadword date values

[3] Default of 7 fractional digits of precision are preserved by default from VMS quadword date values

Sizing

- For estimation purposes, target database sizing is typically in range of 2X to 6X size of DBMS database backup file

- Impacted dramatically by
  - Database platform
  - Compression features utilized
  - Physical structure
  - Data types
  - Count and shape of indexes
  - Etc.

Variety of Options

- Include/exclude record types
- Additional create table and index characteristics
- Commit frequency
- Data type mapping and/or string and timestamp trimming
- Deleted rows “preserved” in target database
- Field scaling
- Debug/trace output
- Table rename via prefix
- Multiple databases to one target
- Last update TSN / TAD
- Statistics frequency
- Field content propagation to child relation
- Next/Prior fields
- Parallel loading

Example Table From MANMAN

```
SQL> describe ED_DOCXRFREC
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 SCI_SELF                                           NUMBER
 SCI_LAST_UPDATE_TAD                                TIMESTAMP
 SCI_LAST_UPDATE_TSN                                NUMBER
 ED_DOCXRFTYP                                       VARCHAR2(3)
 ED_DOCXRFFLG                                       NUMBER
 ED_DOCXRFTPDOCNO                                   VARCHAR2(10)
 ED_DOCXRFTPDOCLN                                   NUMBER
 ED_DOCXRFMMDOCNO                                   VARCHAR2(10)
 ED_DOCXRFMMDOCLN                                   NUMBER
 N_ED_ENTDOCSET                                     NUMBER
 O_ED_ENTDOCSET                                     NUMBER
 P_ED_ENTDOCSET                                     NUMBER
 N_ED_AUDDOCSET                                     NUMBER
 O_ED_AUDDOCSET                                     NUMBER
 P_ED_AUDDOCSET                                     NUMBER
```

Example Query for WorkStream

```
SQL> SELECT     LOT.WIPLOT_LOT_NUMBER,
                ATT.WIPLTA_ATTRIBUTE_NUMBER,
                ATT.WIPLTA_ATTRIBUTE_VALUE
FROM WIPLOT_REC LOT
        JOIN WIPLTA_REC ATT ON
        ATT.O_WIPLOT_ATTRIBUTE_INDEX = LOT.SCI_SELF
WHERE LOT.WIPLOT_LOT_NUMBER = 'MQ310279.01'
ORDER BY ATT.WIPLTA_ATTRIBUTE_NUMBER;

WIPLOT_LOT_ WIPLTA_ATTRIBUTE_NUMBER WIPLTA_ATTRI
----------- ----------------------- -----------
MQ310279.01                      -3 MQ310279
MQ310279.01                      33 130103
MQ310279.01                      47 65500451
MQ310279.01                      66 00
MQ310279.01                      67 29
MQ310279.01                      68 1
MQ310279.01                      93 130103
```

Example Views for WorkStream

```
CREATE VIEW WIPLWF AS  -- Adds "WIPLOT_LOT_NUMBER" to WIPLWF_REC
SELECT  LOT.WIPLOT_LOT_NUMBER,
        LWF.WIPLWF_WAFER_ID,
        LWF.WIPLWF_PROD,
        LWF.WIPLWF_QUANTITY
FROM WIPLWF_REC LWF
JOIN WIPLOT_REC LOT ON LWF.O_WIPLOT_LOT_WAFER_INDEX = LOT.SCI_SELF;

CREATE VIEW WIPLTA AS  -- Adds "WIPLOT_LOT_NUMBER" to WIPLTA_REC
SELECT  LOT.WIPLOT_LOT_NUMBER,
        LTA.WIPLTA_FACILITY,
        LTA.WIPLTA_ATTRIBUTE_NUMBER,
        LTA.WIPLTA_ATTRIBUTE_VALUE,
        LTA.WIPLTA_ATTRIBUTE_VALUE_TYPE,
        LTA.WIPLTA_STORE_HISTORY_FLAG,
        LTA.WIPLTA_SCATTER_KEY
FROM WIPLTA_REC LTA
JOIN WIPLOT_REC LOT ON LTA.O_WIPLOT_ATTRIBUTE_INDEX = LOT.SCI_SELF;
```

Set Traversal Example

- SELECT … START WITH … CONNECT BY PRIOR used to traverse a set in "find next" order

```
SELECT      P.WIPPRD_FACILITY, P.WIPPRD_PROD,
            R.WIPPRR_ROUTE, R.WIPPRR_PREVIOUS_ROUTE
FROM WIPPRD_REC P, WIPPRR_REC R
WHERE       P.WIPPRD_PROD = '213489612' AND
            P.WIPPRD_FACILITY = 'NJL77' AND
            P.SCI_SELF = R.O_WIPPRD_ROUTE_SET
START WITH R.P_WIPPRD_ROUTE_SET = R.O_WIPPRD_ROUTE_SET
     CONNECT BY PRIOR R.SCI_SELF = R.N_WIPPRD_ROUTE_SET;
```

Performance Indicators

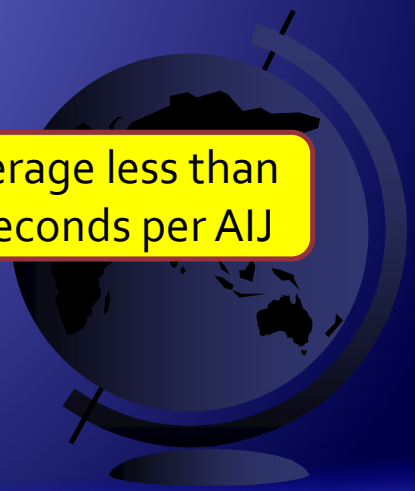
Preliminary Performance Indicators

- Populate Rdb database from full customer DBMS database backup file (initial load)
  - 621MB DBF / 2.2M rows
  - rx2620, Rdb 7.2.5.2, VMS V8.4
- Replicate entire day's worth of customer's AIJ files
  - 17 AIJ files totaling 650MB
  - 313,482 source transactions
  - 2.3M update/insert/delete actions
  - rx2620, Rdb 7.2.5.2, VMS V8.4
- Replicate AIJ every 2 minutes
  - CHARON AXP

Less than 4 minutes*

Less than 8 minutes*

Average less than 2 seconds per AIJ



*Unoptimized & untuned development code

Preliminary Performance Indicators

- Initial load of full customer DBMS database backup file
  - 4 minutes
  - 890MB DBF / 5.9M rows
- Replicate 53 days of AIJ files
  - < 3 minutes per day
  - 697 files, 5.75GB
  - 677,430 source transactions
  - 33M update/insert/delete actions
- rx2620, Oracle 10g client, VMS V8.4
- Oracle 12c server, Oracle Linux 6.5, virtual machine

Preliminary Performance Indicators

- Populate database from full customer DBMS database backup file (initial load)
  - 11GB DBF / 60M rows
  - 65 minutes
- 2x rx2620, Oracle 10g client, VMS V8.4
- Oracle 12c server, Oracle Linux 6.5, virtual machine

Preliminary Performance Indicators

- Change replication is considerably more work than initial database loading
  - Indexes must be maintained
  - Update first or insert first?
  - Delete is a multi-step process
- Different targets have different performance trade-offs, e.g.
  - Rdb requires no interprocess communication
  - Oracle allows array processing

Preliminary Performance Indicators

- One customer experience : apply 1 hour's worth of AIJ takes about 2 minutes
  - ES80 OpenVMS V8.3
  - Oracle 11.2.0.3 (unknown server configuration)

- In-house testing, 1 day's worth of customer AIJs takes about 30 minutes
  - 1,965,958 upserts & 273,595 erases
  - Rx2620 OpenVMS V8.4
  - Oracle 11.2.0.3 (On a virtual machine, OEL 6.3)

Present Targets

- Oracle Rdb on OpenVMS
  - Local or remote database
- Oracle Database
  - Local client
  - Local or remote database server ("anywhere")
- SQL Server
- Would very much like your feedback
  - JDBC? ODBC?

Environments

- SCI TranSSend need not run on same system / platform as source DBMS database
- We would very much like your feedback with your needs

Platform	Environment
Data source	• VAX / Alpha / I64 • DBMS V6 or Later
Extraction	• OpenVMS Alpha / I64 • Rdb V7 or later • Oracle client V8.0.5 or later (suggest 8.1.5 or later) • VMS 7.3-2 or later
Target	• Oracle Rdb V7 or later • Oracle Server 9.2 or later (suggest 10.2 or later) • Microsoft SQL Server 2008 R2 or later

Oracle Rdb

- Embedded & Dynamic SQL
- "Native" data types easily manipulated
  - VAX D float not supported by Rdb – converted to VAX G float internally
- REPLACE used for insert/update with Rdb 7.3 and later

Oracle Database

- OCI interface
- MERGE used for insert/update
- Array interface used for batching of inserts, "upserts" and erases where possible
  - Reduce server round trips
- Modest amount of data manipulations

Oracle Database

- Timestamp converted to time string
  - 6 fractional digits
  - 30-SEP-2013 21:04:02.567721
- NUMBER used for float & integer types
  - Conversions from VAX F, G and D to IEEE S & T
  - Then IEEE converted to NUMBER
  - (Depending on platform and version)

Microsoft SQL Server

- TDS Interface
- Integer and floating point data converted to text before being passed to server
- Timestamp converted to time string
  - 7 fractional digits

For more SCI TranSSend information:

www.sciinc.com

lastovica@sciinc.com

holland@sciinc.com