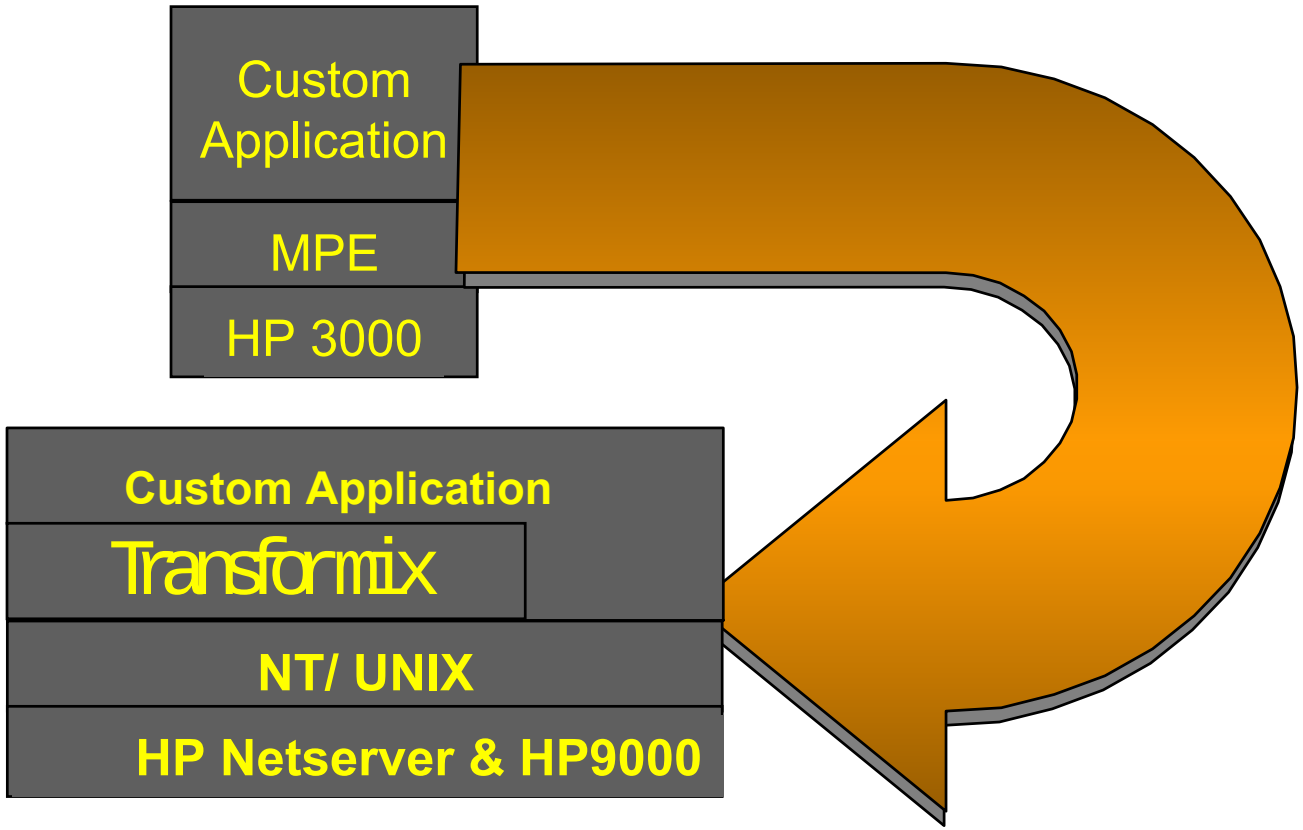
Investment Protection for MPE/iX Applications Through Application Renovation

http://www.xformix.com/

The Process

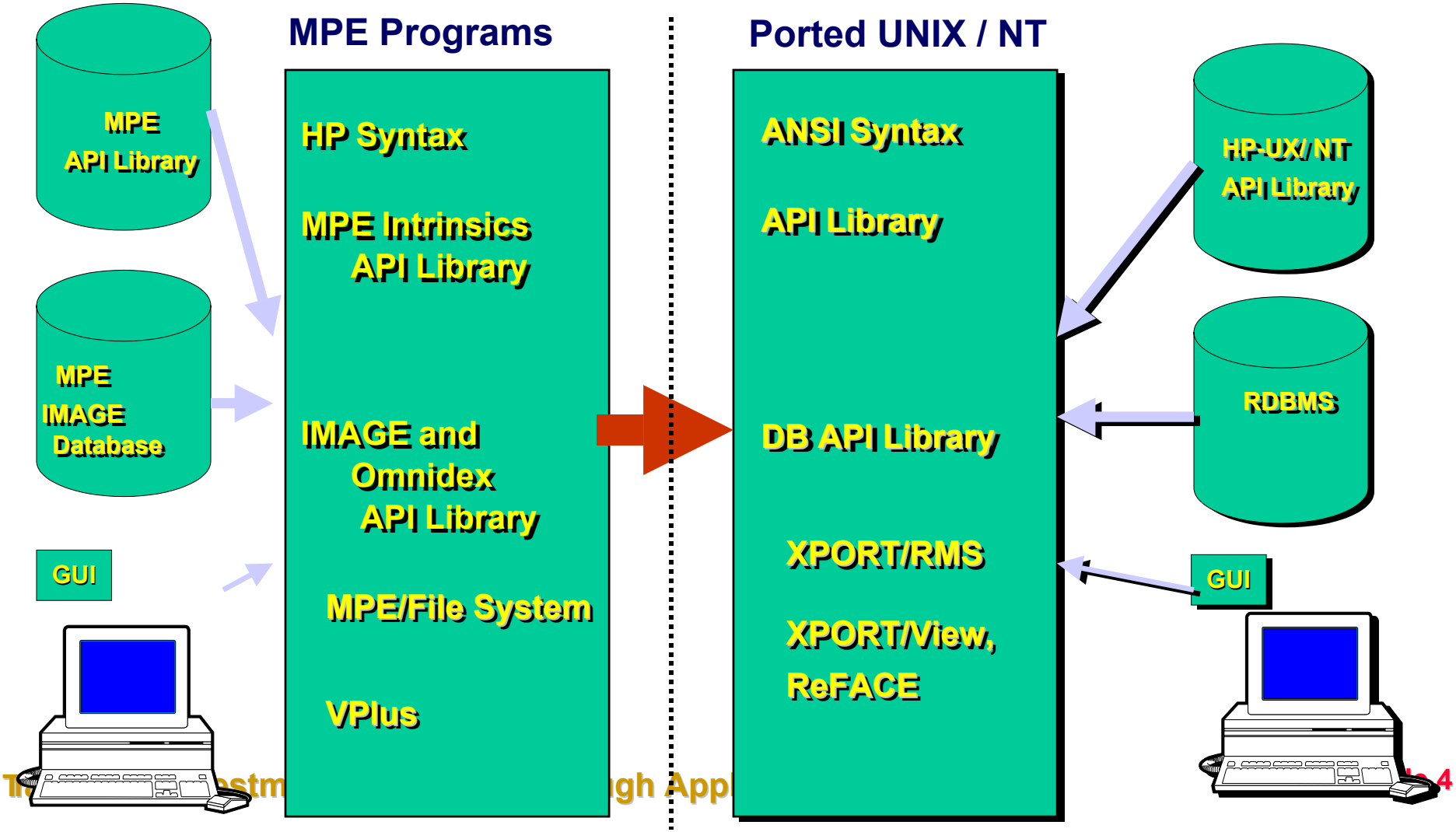
Successful Migration Achieved through Sector 7/Transformix's Five Step Process



Part I – Topic Introduction

Overall Goal

Target: Port MPE Application



HP 3000 Databases

- TurboIMAGE database management system as a data repository or Data Base Management System (DBMS).
- Other Databases
  - ALLBASE are also employed, but
  - Oracle was available for a while
  - RELATE
- IMAGE continues to be the standard.
- XFORM Toolkit a comprehensive library of tools that are used to port and migrate MPE-based applications to UNIX and NT.

Complete TurboIMAGE Porting must address

- Application Program IMAGE Usage
  - IMAGE Calls
  - Third-party indexing calls
- Data migration
- Third-party DB Utilities Suprtool
- IMAGE Shadowing and/or Mirroring (Netbase)
- Database restructuring tools (Adager, DBGeneral)
- Database Backup
- On-Line Backup

Complete TurboIMAGE Porting – Program Behavior

- Application programs depend on IMAGE and third-party indexing behavior and features.
- Dependencies
  - Return codes
  - Result sets
  - Sorts
  - Keys
  - “Hidden dependencies”
- Logic changes are required if features are missing

Part II – Application Program IMAGE Usage

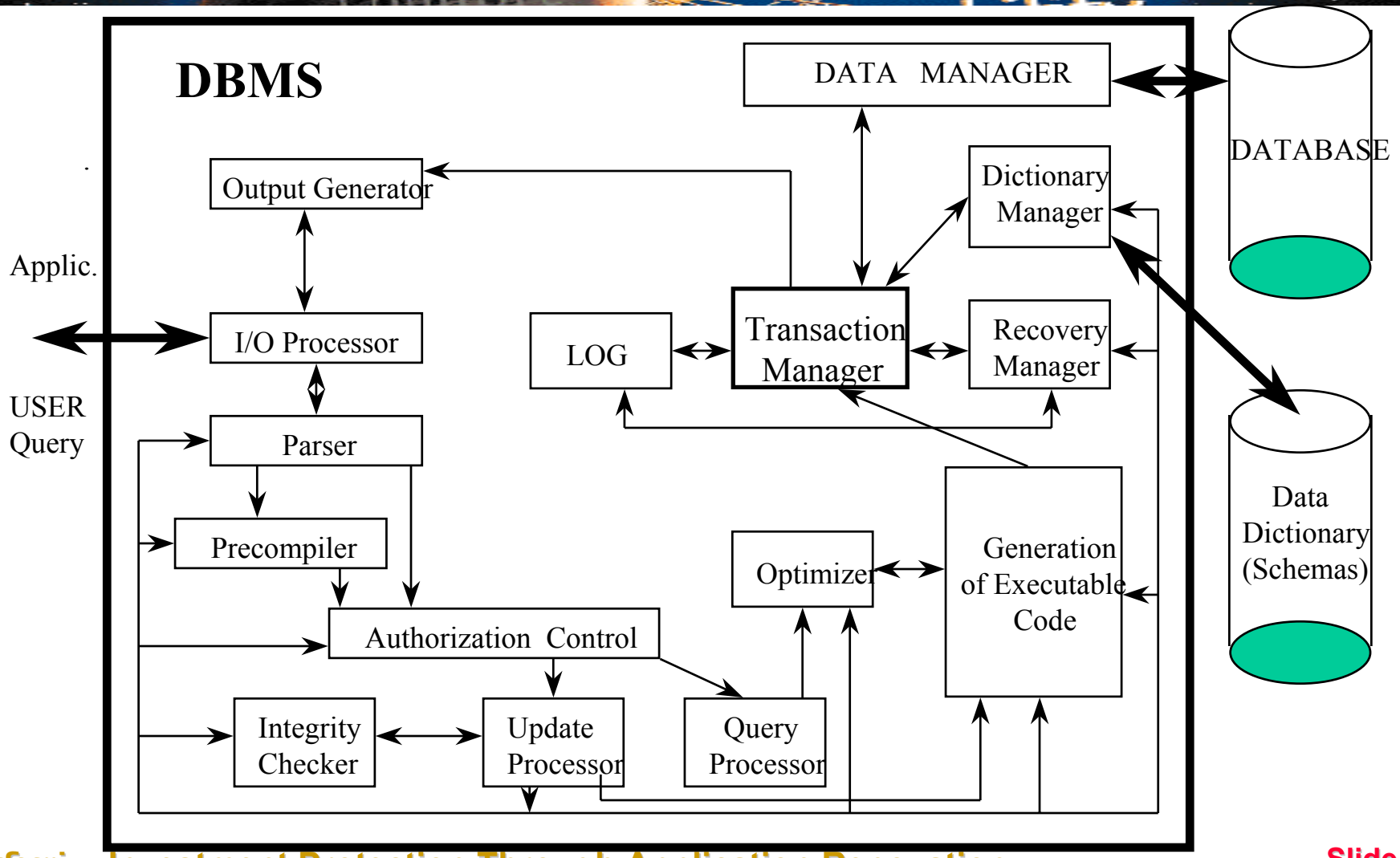
Technical differences between IMAGE and RDBMS's

- IMAGE is a network, or a limited-hierarchical, database management system.
  - Information is stored on two levels and is accessed by traversing a path from the highest level, or hierarchy, of the network to the desired lower level.
  - The two levels of hierarchy in IMAGE are the master and detail level.
  - At the master level, master datasets keep information about a uniquely identifiable entry.
  - At the detail level, detail datasets keep detailed information related to the unique entries in the top level.
  - Detail datasets are also used to relate the entries in the master level to each other.

Technical differences between IMAGE and RDBMS's

- In the relational model data is organized into tables.
- A table is a two-dimensional structure of columns and rows.
- A row is similar to an entry in an IMAGE dataset and a column is similar to a data item or field.
- Access to this data is facilitated through Structured Query Language (SQL).
- Examples of relational database management systems include Oracle, Informix, DB2, and SQL Server.

RDBMS Structure



RDBMS Database IO

- Carrying out database I/O to these relational engines from a procedural language like COBOL can be accomplished through:
  - Pre-copilers
  - Call Level Interface

RDBMS Database IO

- Precompilers.
  - These work with a standard 3GL language compiler to develop code with embedded SQL statements.
  - A precompiler is then used as a first pass on a source program to identify and translate all embedded SQL statements and generate a modified source program that the 3GL compiler can understand.
  - The 3GL compiler is then used to compile and link the program to its finished executable form, which can then access the database.
  - To aid in the development of portable database applications, some precompilers include options to detect and mark source code statements that do not conform to ANSI-standard SQL syntax.

RDBMS Database IO

- Call Interfaces.
  - These also let the programmer use a 3GL language and embedded SQL statements to carry out database I/O.
  - However, call interfaces also provide low-level, proprietary procedure and function calls that can give the programmer more power and flexibility than precompilers.
  - A program using the call interface can control the actual steps of SQL statement processing (parse, bind, execute, rebind, reexecute, and so on) to achieve optimum application performance.

Third-party Indexing

- Omnidex and Superdex
- Provide additional features
  - Partial keys search and sort
  - Soundex
  - Multi-key fields
- Functions performed outside of application program logic

Third-party Indexing Porting Issues

- Functionality hidden in TPI calls
- Porting approach must provide additional TPI functions
- It is best if additional functions do not require changes in program logic.

Third-Party Indexing Porting Problem Example

- Data exist in IMAGE database as a Detail set
- COBOL application expects data to be sent back with sort on first two characters of field in detail set
- Data is retrieved based on those two characters
- What happens to program results if that sort and does not happen automatically and the retrieval on the two characters is not available?
- How difficult is it to program the missing pieces if the tools don't provide the needed functions?

The ideal call interface

- Quick and efficient migration
- Allows the existing COBOL program to make IMAGE and third-party indexing calls as it has always done and to translate these into the call interface supported by the target RDBMS.
- The call interface of the target RDBMS gives the most control and highest performance available and is the preferred mechanism when creating a call-compatible library of IMAGE intrinsics.
- Third Party Indexing features are available and therefore program logic does not change.

COBOL Programs Access RDBMS Using IMAGE and Omnidex Calls

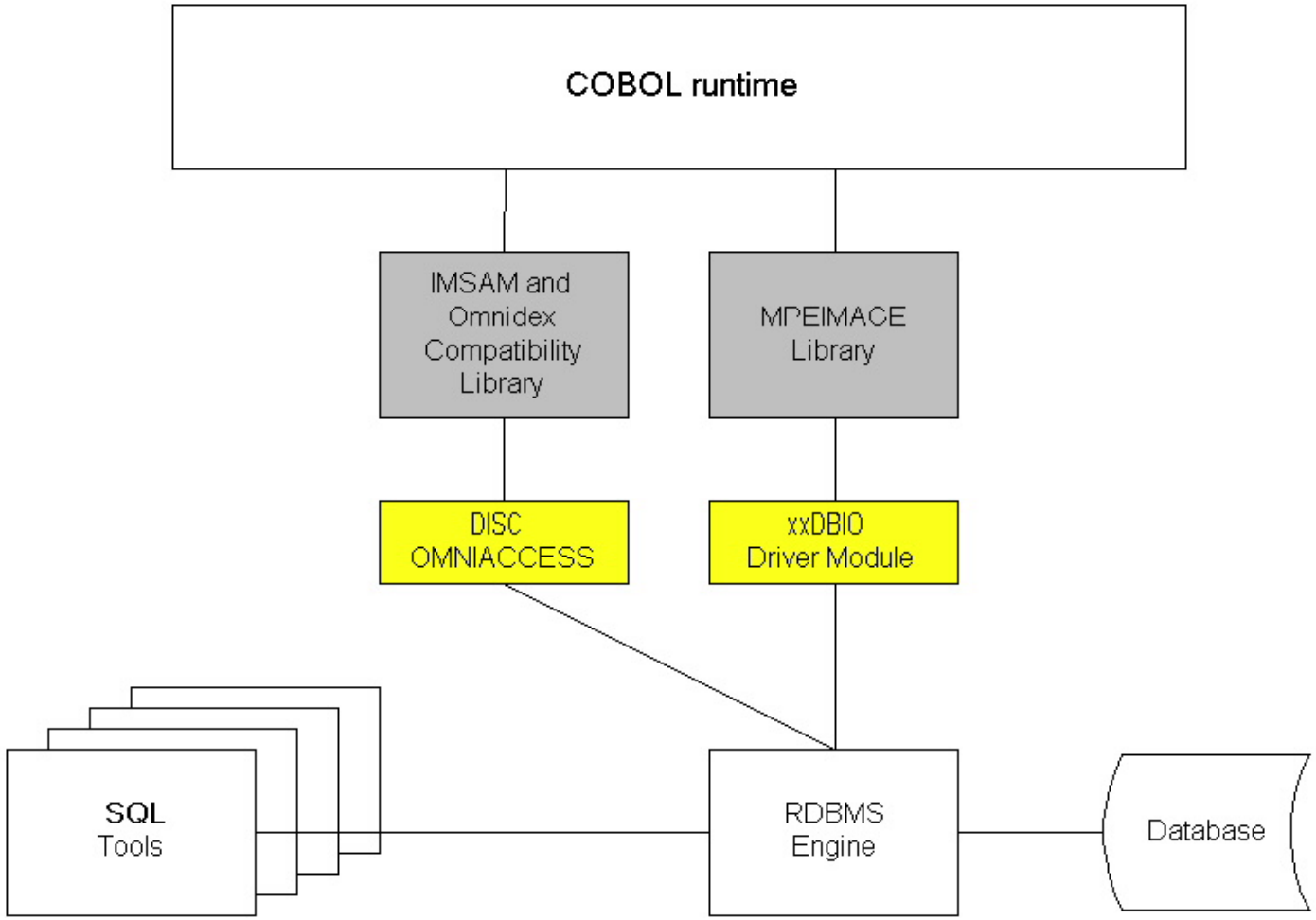


IMAGE Call Library

- This is done by creating a library of C routines that includes procedures that emulate the functionality of each IMAGE and third-party indexing intrinsic and that translate the IMAGE request into a series of native RDBMS database calls.
- This library can be either an archive or shared library.
- The library is then linked into the target COBOL run-time module to support the INTRINSIC calls and provide a call-compatible interface.
- Properly designing a call-compatible interface library can be challenging.

Backward Chained or Serial Reads

- Challenges include support of a backward chained or serial read when the RDBMS does not support FETCH previous.
- It is also necessary to adapt and enhance the ROWID address to support directed reads.
- The benefit of an automated migration that uses an IMAGE call-compatible interface is that either no changes need to be made to program sources or, if any are required, they are few.

IMAGE Hierarchical Model to Relational Model Database Migration

- IMAGE Hierarchical Model to Relational Model Database Migration
- Relational and UNIX/NT
  - When moving to UNIX or NT many customers assume that that the database model will be relational, accessed through a Structured Query Language (SQL). Relational databases offer much greater flexibility than the network model used by IMAGE, but speed of data retrieval is characteristically slower.

Data Access Paths

- IMAGE is designed to provide quick access to data by certain paths, while relational systems are designed to provide access to data by any path.
- In practice, many other factors such as buffering and optimization come into play, making the comparison less distinct.

RDBMS's Used In Migrations

- XFORM accomodates both however, we will soon offer other RDBMS's including Microsoft SQL Server, IBM DB2 and PostgreSQL.
- Any brand of database should be usable for a migration, but some offer definite advantages over others.
- Some offer precompilers which allow SQL statements to be embedded directly in user programs, while others provide low-level routines which are called programmatically.
- Add-on features such as fourth generation languages, report writers, and form generators are often offered.
- should be examined carefully before the DBMS choice is made.

Part III – IMAGE Database and Data Migration

The Multi-Part Migration Process

- First, the IMAGE schemas are analyzed programmatically to extract dataset names, data item names, compound items, data types, key items, search items, sort items, and paths.
- The analysis program produces SQL CREATE table scripts for each IMAGE master and detail set (automatic master sets as a physical table are not required, although their indexing function will be preserved).
- These SQL scripts can then be executed to produce empty tables ready to be populated with data.

The Multi-Part Migration Process

- The xschema processor makes various changes that are needed in order to map IMAGE data to the target RDBMS. For example, in Oracle, names can contain only alphanumeric characters from the database character set and the characters '_', '$', and '#'. The dash character is therefore mapped to the underscore character.
- Compound items are a feature of IMAGE that do not have an equivalent in the target RDBMS. Multiple unique column names are created to support these items. Naming of schema objects has some restrictions in the target RDBMS.

The Multi-Part Migration Process

- Migration of IMAGE data involves exporting master and detail sets to sequential files using third-party utilities or custom export programs.
- These files are then transferred to the target UNIX machine, and
- an automated tool uses the native RDBMS loader utility to populate the empty relational tables with this data.

Schema conversion.

- IMAGE schemas must be taken and translated into equivalent SQL statements for relational database creation.
  - We chose to do this programmatically, although a one-time conversion might be done by hand.
  - In the relational database world, datasets are referred to as tables, data items are called columns, and records are called rows.
  - The basic idea is to convert all dataset definitions into SQL “CREATE TABLE” statements which define the type of every column in the table.

IMAGE dataset definition			Informix table definition	
NAME:	TEST-1, MANUAL;		CREATE TABLE TEST_1 (	
ENTRY:	T1 (1),	[X2]	T1	CHAR(2) NOT NULL,
	T2,	[J]	T2	SMALLINT,
	T3,	[J2]	T3	INTEGER,
	XX;	[4X2]	XX01	CHAR(2),
			XX02	CHAR(2),
			XX03	CHAR(2),
			XX04	CHAR(2))
CAPACITY: 100;			EXTENT SIZE 16 NEXT SIZE 16;	

Data types

- Most IMAGE data types map quite well into other databases, but a few require extra effort.
- The J4 data type, for example, is a 64-bit integer value which is rarely seen outside of COBOL.
- We were forced to map all J4 items to floating point types and convert the values at run-time.
- Data types such as the ‘P’ variety are assigned to SMALLINT or INTEGER columns, depending on the number of digits required, and converted at run-time.

Data types

- The concept of compound items does not exist on most RDBMS systems, so types such as 4J2 must either be broken down into multiple INTEGER columns or grouped together as a single, large CHAR field. Of course, this can lead to other problems because not all databases systems allow binary values to be stored within character fields.
- Even the names of items must be converted, as special characters such as the dash (-) are not available under SQL.

Data types

- Our conversion program changes all such characters to underscores (_). Fast access to data is required for all search items, so we use the "CREATE INDEX" command to generate indices on all search and sort items. This allows the RDBMS to access records quickly be key value. Search items are declared as "NOT NULL", informing the DBMS that the column may never be omitted.

RDBMS “capacity”

- The IMAGE “capacity” roughly equivalent to extent sizing under Informix, a concept which closely resembles MPE/XL file extents.
- The biggest difference is that Informix disk space is assigned in large chunks which are typically shared by the entire database.
- Space for a single table is allocated one extent at a time, but there is no absolute extent limit.

RDBMS “capacity”

- To assign chunks of space one can either allocate a disk partition, known as a “raw” device, or create a standard UNIX file, referred to as, of course, a “cooked” file.
- Performance is typically better with raw devices, as the UNIX file management routines are bypassed.

File Transfers

- Data migration is another issue that is addressed by the Transformix solution. In the current state, we unload the HP3000 data into flat files using a utility called DB2DISK, the files are transferred the files to UNIX or NT.
- Next another program on the target computer called DISK2DB loads the data.
- These programs are quite simple; one uses DBGET to read data into flat files, the other uses DBPUT to load the files into the migrated database.
- If necessary, data conversion is done at this point.

Part IV – The Run-Time Environment

Run-Time

- All IMAGE routines such as DBOPEN, DGET, and DBPUT must either be replaced or rewritten under UNIX.
- It is possible to convert each application program to use SQL statements rather than IMAGE calls, but we chose to leave the application software as-is and write our own versions of the IMAGE library routines.
- In developing the XFORM database emulation libraries, each IMAGE routine was studied and an clone version was written which generates the equivalent SQL statement(s).
- For optimum performance, this was written at the lowest possible level. As far as the application can tell, it is still running under IMAGE. Any other methodology would probably end up being more of a rewrite than a conversion.

DBGET

- Emulation of DBGET requires an appropriate SELECT statement and one or more FETCH statements to actually retrieve the data. To explain the terminology, a SELECT statement requests a set of records. For example, 'SELECT T1, T2, T3 FROM TEST_1 WHERE T1 = "A"'. The requested set of records is associated with an identifier called a cursor. Records may then be retrieved from the active cursor using the FETCH statement.

Serial and chained reads

- Generate a select and multiple fetches, while all other modes of reading generate a single select/fetch combination.
- Some RDBMS platforms allow bi-directional reading of the selected records, while others permit only serial fetches.
- If the bi-directional feature is not available it will be necessary to create multiple SELECT statements whenever combinations of forward and backward reads (chained or serial) are needed.
- This can be quite complex because there is no simple way to start reading in the middle of a detail chain. Possible solutions to this problem involve maintaining the entire IMAGE chain structure, adding an additional sort field to every table, or disallowing this type of operation.

Manual master, automatic master, detail

- IMAGE classifies datasets into types: manual master, automatic master, and detail.
- Relational databases have only one type of table, so relationships between the tables must be maintained programmatically.
- Automatic master datasets could be eliminated because they are no longer needed as chain heads to access detail information.

Manual master, automatic master, detail

- However, we chose to maintain them to more fully emulate the capabilities of IMAGE.
- For example, a serial read of an automatic master dataset may be used to list the contents of several linked detail datasets.
- The master/detail relationships must also be supported programmatically, or strange problems such as detail entries with no corresponding master entries may be experienced.

Manual master, automatic master, detail

- To this end, our emulation routines perform many functions such as checking for master entries before putting detail entries, to ensure that database integrity is preserved. Naturally, our routines cannot do all this without knowing the original IMAGE structure of the database. To accomplish this, we maintain the relational equivalent of the IMAGE root file. Using this data we are able to fully emulate nearly every feature available under IMAGE.

IMAGE SORTS

- Applied as data is inserted, relational database systems sort data only on retrieval.
- Thus, all SELECT statements generated for sorted datasets must include an “ORDER BY” clause. Directed access may be accomplished using a “rowid”, which is the relational equivalent of an IMAGE direct address.
- Unfortunately, not all RDBMS platforms use a 32-bit rowid.
- Some are quite a bit larger. Some are not consistent throughout the life of a row. Under Oracle, these constraints forced us to generate our own unique address and use it as an additional search item in each table.

Part V – Normalization

Some advantages of the Relational Model:

- - no artificial constructs are introduced by the modelling process
  - since the structure is entirely value-based, no restriction is placed on the processing which may take place
  - attributes are retrieved purely on the basis of rules and conditions; they require no non-logical access paths
  - the relational model is a logical data model independent of storage considerations
  - allow powerful data manipulation and query languages to operate on them

Some disadvantages of the Relational Model:

- - since manipulation of the entire data structure is based on value matching this represents an expensive process in access terms and makes the relational model unsuitable for high-volume processing.
  - few vendors have implemented the relational model in the way intended

Database Normalization

- Our approach does not result in any normalization of the data.
- Relational database systems are based largely on mathematical relational theory.
- Designing a relational database involves the process of normalization.
- The theory of normalization is a key element of relational database design. In simple terms, normalization is the process of decomposing and arranging the attributes in a schema, which results in a set of tables with very simple structures.
- The purpose of normalization is to make tables as simple as possible and to reduce the redundancy of information within a database.
- When done properly, the normalization process will result in well-designed tables, with reduced storage of redundant data and the increased ability to effectively enforce integrity constraints.

The Normalization Process

- If attempted on an existing IMAGE database, obviously would involve major reengineering of the application and would thus be impractical.
- The lack of normalization can be perceived as a drawback, but it is perfectly acceptable to have migrated IMAGE data into a non-normalized relational form for an intermediate period of time.
- Once the migration is complete, there are no restrictions on normalizing at a managed pace appropriate for the application and future development.

Advantages/Disadvantages of Retaining the IMAGE API

- Performance because of the hierarchical access structure
- The inherent network structure and key/search item construct of IMAGE, by its very nature, produces some data redundancy.
- One of the advantages of migrating data into a RDBMS immediately even while retaining the IMAGE API is that it allows the user to take advantage of new functionality sooner.

Advantages/Disadvantages of Retaining the IMAGE API

- The flexibility of SQL allows for the ad hoc creation of indices to improve performance, as well as data structures, such as views and synonyms, to mask or massage the data structure without actually changing the table structure itself.
- The application developer is able to write new applications with the productivity tool sets that are available in the target environment and can then use the power of distributed transaction processing, client-server configurations, and full application portability.

Part VI - Miscellaneous
IMAGE Porting Issues

Miscellaneous IMAGE Porting Issues

- Shadowing and Mirroring
- Suprtool
  - May be available
- Database restructuring tools (Adager, DBGeneral)
  - No longer needed
- Database Backup and On-Line Backup
  - Other tools are available

Conclusion

- Porting TurboIMAGE databases to RDBMS includes:
  - Application Program IMAGE Usage
    - The basic product
    - Third-party indexing
    - Data migration
    - Suprtool
    - Database restructuring tools (Adager, DBGeneral)
    - Database Backup
    - On-Line Backup
- A complete solution addresses all of these aspects

Conclusion

- Program logic changes increase porting complexity
- Call level interfaces can reduce or eliminate program logic changes
- Call level interfaces can provide basic IMAGE functionality along with third party indexing functionality in such a way that logic changes are not required.