# FREE MICROSOFT ACCESS MIGRATION STRATEGY REPORT

How to conform, consolidate, migrate, and modernize seven Microsoft Access database applications into a single, robust solution.

Intro | Standardized Technology | Application & Database Discovery | Database Modernization | Data Confirmation | Application Modernization

As of 2014, the Safety and Environmental Division (SED) of the California Public Utilities Commission (CPUC) was using multiple Microsoft Access (MS Access) database applications to track mission critical Utility Safety and Reliability (S&R) data for the entities that the CPUC oversees.

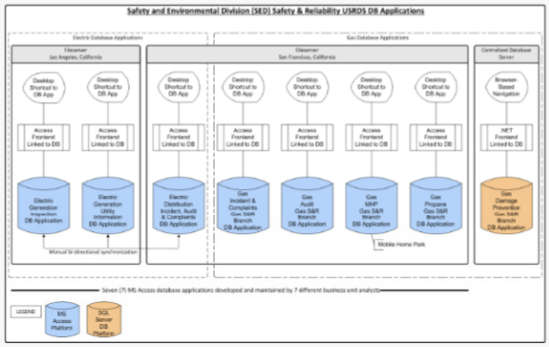
Initially, in the early 1990s, the Electric and Gas branches were one integral organization, and a single database application was used for tracking utility incidents, annual audits of utilities, and consumer-reported complaints. When the Electric and Gas organizations were separated into two branches, the database application for incidents, audits, and complaints was separated into two applications: one for the Electric branch, and one for the Gas branch. The split databases were split again and again, and then each individual application was customized further to meet the needs of a particular business unit in a particular geographic location. The resulting seven MS Access database applications each were developed by different business analysts, and subsequently, different code bases and individually unique interfaces exist.

Given the importance of the Utility Safety & Reliability Database System (USRDS) data, the SED management has requested that the seven MS Access database applications be conformed, consolidated, migrated and modernized back into a single solution.

The following diagrams document the transformation from the current state to the final application, data and technical solution architectures.

## Original Solution State – Logical

Symptoms: Poor application performance, high IT costs of supporting seven different application code bases, data inconsistent across database applications, data corruption, lack of data integrity, lack of ability to support public access, inability to backup data, application crashes, lack of regulatory compliance, lack of user confidence, and a lack of standardized user interfaces and support for single-sign-on.




## Future Solution State – Conceptual

The diagrams for the future state have been moved to the bottom of the document to aid in the readability and continuity of the case study. Use the below link or page down.

## Two Migration Approaches

The smaller the increment, the smaller the risk and the smaller the gain. We will be taking the iterative approach.

| | Cold Turkey | Iterative |
|---|---|---|
| Risk | Huge | Controllable |
| Failure | Entire project fails | Only one step fails |
| Benefits | Immediate, probably short-lived | Incremental, over time |
| Outlook | Unpredictable until deadline | Conservatively optimistic |

# FREE MICROSOFT ACCESS MIGRATION STRATEGY REPORT – STEP 1

How to conform, consolidate, migrate, and modernize seven Microsoft Access database applications into a single, robust solution.

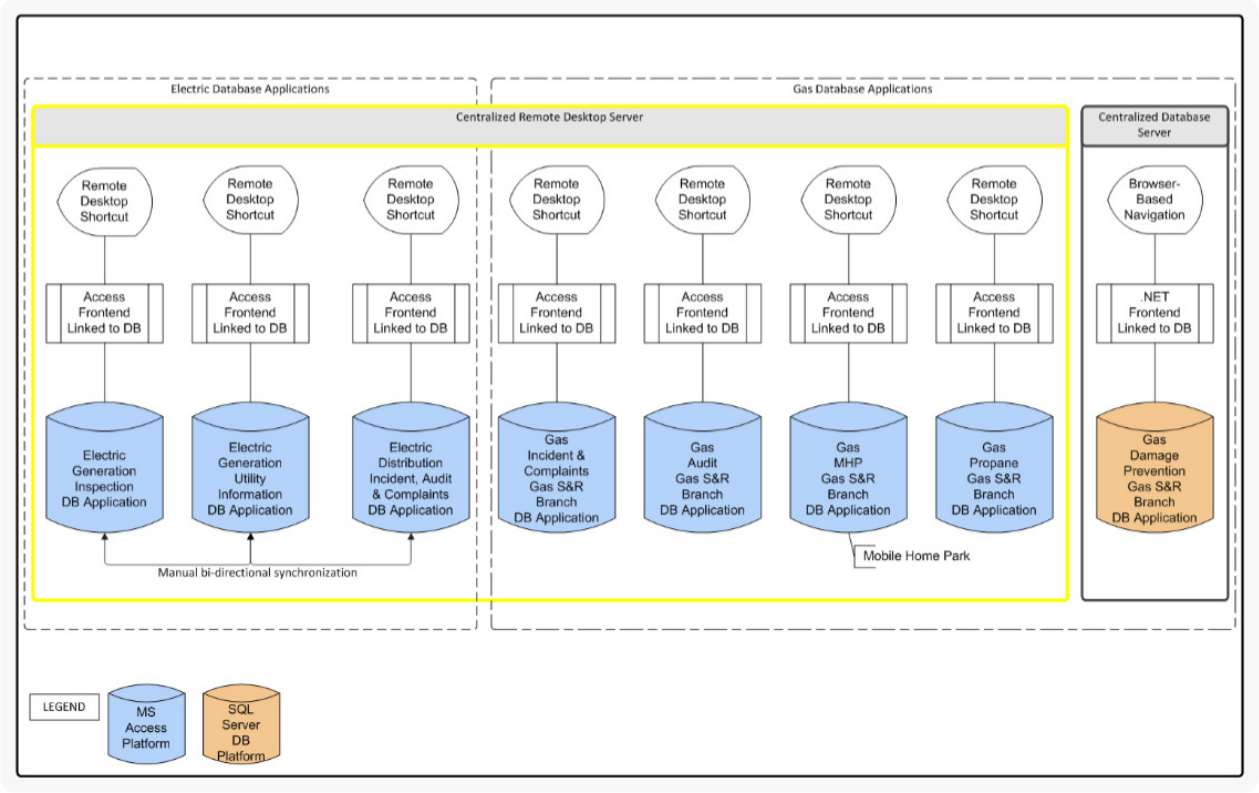
Intro | Standardized Technology | Application & Database Discovery | Database Modernization | Data Confirmation | Application Modernization

Goal: Shift from business silos (IT resources split between San Francisco and Los Angles) to a shared services model decreasing the number of platforms supported and IT costs.

Tasks: Upgrade all MS Access applications and databases to the latest version of MS Access. Move front-end applications and back-end database to centralized server with remote desktop.

Results: The coordination of the application and database moves within the organization is a risk. The centralized server removes network bottlenecks, and the organization gets a quick application performance boost across all applications, as well as more reliable backups and some added security features.

## Database Applications Moved to a Centralized Server



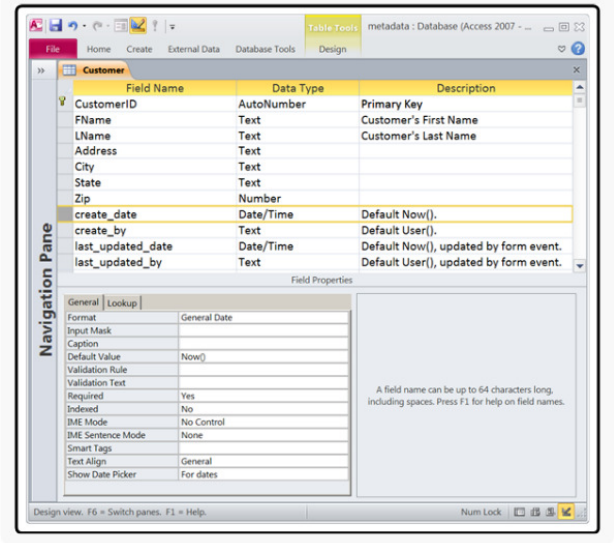
Goal: Sunset redundant application and data architectures early in the process.

Strategy: Deploy database and application auditing mechanisms to gather statistics, which will be used to find unwanted objects to remove from redundant database applications.

Database Audit: Insert four metadata columns in every table within every MS Access database and modify the applications to support the new columns.

## Application Audit

Within every form and every report, add a new function that is activated on, on open, after update (form only), and on close. The function will INSERT the object name, the system's date and time, the current user, and the event action into the new usys_application_audit table.



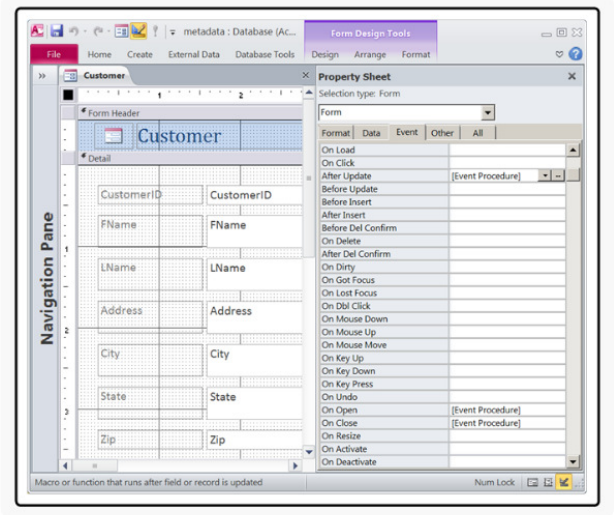

Example Data produced within usys_application_audit table:

frmPassword, 201509140700, John Doe, Opened.
frmCustomer, 201509140819, John Doe, Opened.
frmCustomer, 201509140822, John Doe, Updated.
frmCustomer, 201509140821, John Doe, Closed.
frmPassword, 201509140730, John Doe, Closed.
…

Deploy the modified MS Access applications and gather application and database usage statistics. Find candidate objects for sunsetting and retire those tables, forms and reports that are not supporting a business function. Retiring may involve just renaming the objects to OLD_(object name). The goal of this stage is to separate the wheat from the chaff, so to speak, before we mill/process the most valuable part: the customer's data.

## Application Transformation Management (Iterative)

In order to have the capability deploy transitional application architectures, it's necessary to add the feature into the application layer to allow the administrator to control which objects (forms and reports) a user within a group may execute. This capability is necessary to incrementally sunset portions of the application whenever you sunset a feature or move a feature one application to another during application consolidation.




The programming logic will require the development of a user-role-application object-rights matrix, and the supporting programming logic to be inserted into every database application.

Some of the objects that will be developed during this step are: a password form, usys_user_role_object_rights table, unified menu and navigation forms that are user-right aware, and usys_app_config_version_local and usys_app_config_server tables to allow administrators to sunset entire applications when a new release is available.

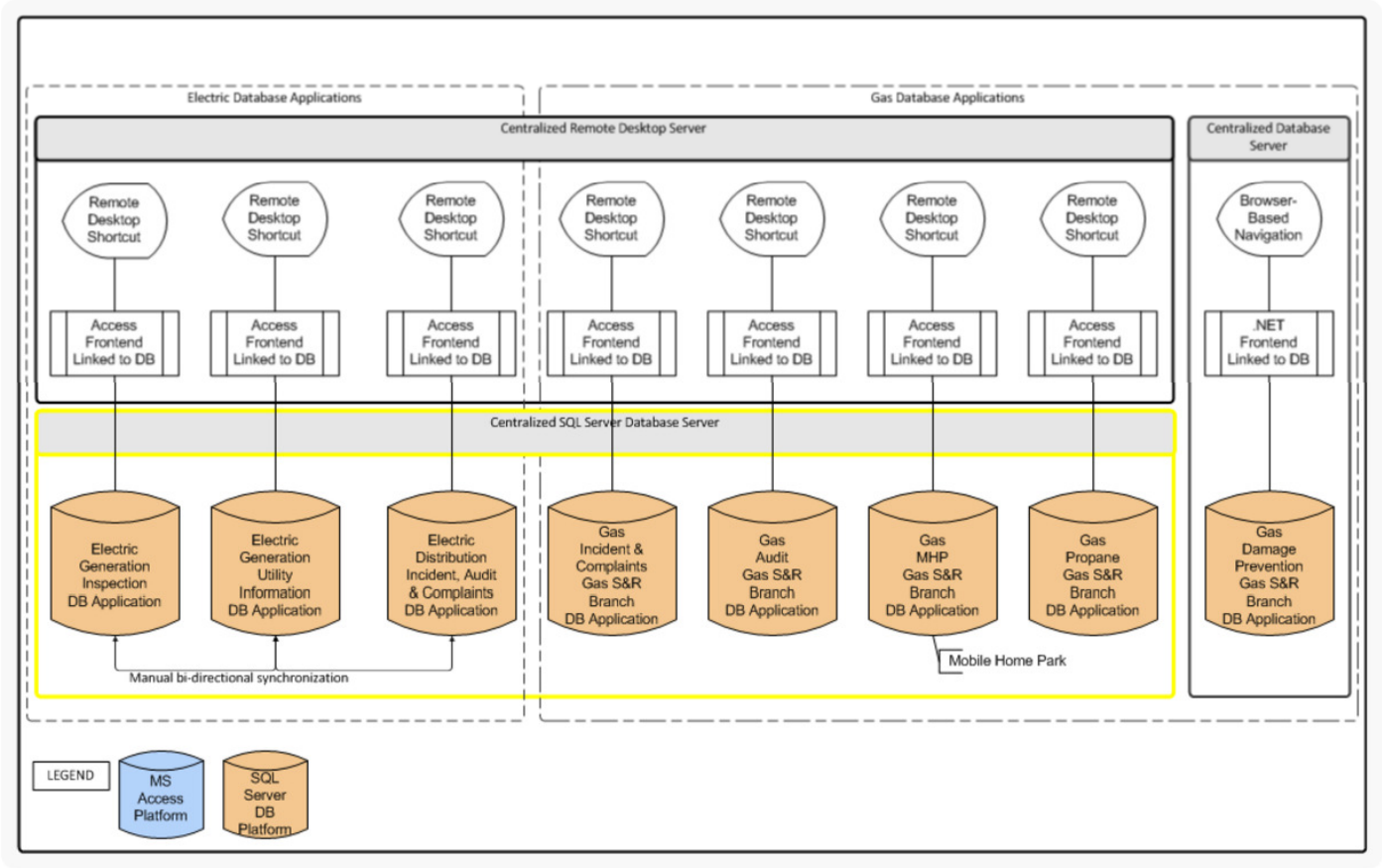
With the obvious nonessential objects removed from the database applications, stage three's goal is to prepare the applications for data consolidation.

Due to the processor-intensive nature, the need for a stable data source, and the level of complexity involved within Step 4, Step 3 is a simple, straight-forward migration of the seven MS Access databases to the client's chosen database management platform, which is SQL Server.

This is a straight port – table by table – into SQL Server. No columns names nor nonessential data type conversions or additional data entity constraints will be added.

There are many free tools that can aid in this one-to-one database table-by-table migration, but for our needs, the SQL Server Migration Assistant for Access (AccessToSQL) is the tool of choice.

The yellow rectangles below show the database tables after being migrated to a centralized SQL Server database server.

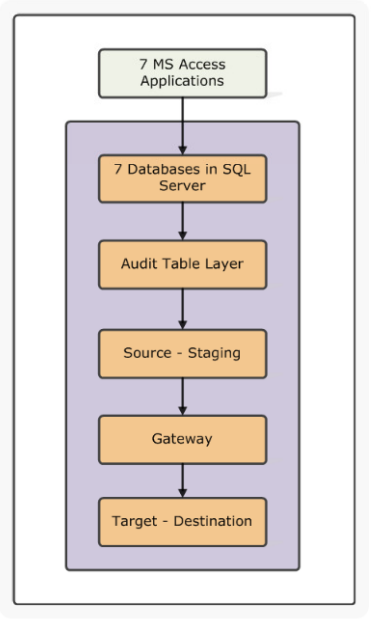


Just as we had to deploy application transformation management in Step 3, here we'll implement and maintain data transformation management.

Now that the applications have been stabilized on a unified technical platform, their extraneous objects have been removed, and application transformation management has been installed, we're ready to perform the bulk of the work to conform the seven different yet similar database schemas into a single, fully-normalized design along with a single MS Access application.

Steps 1-3 are all infrastructure orientated. During Step 4, new database application enhancements, required by the business, may be developed.

## Conceptual Solution Architecture




Data is edited through the seven MS Access applications which are linked seven database schemas on the SQL Server. Each applications' creates, reads, updates, and delete (CRUD) operations inserts a new record into the corresponding audit table, which employs an effective dating algorithm. The staging environment extracts only the current records from the audit table layer and inserts them into the staging environment. The gateway performs extract, transform, and load (ETL) operations against the staging tables and inserts and updates the target destination tables. The frequency of gateway/ETL operations determines the latency between the source and target tables and may be adjusted.

### Detailed Description of each object in the step 4 conceptual solution architecture

**Seven MS Access Applications**

The client's original MS Access application's forms and reports, with the metadata columns are added in Step 2.

**Seven Database Applications on SQL Server**

The client's original database tables migrated to SQL Server with the supporting metadata columns are added in Step 2.

These database tables will be consolidated over time and conformed by the ETL process within the gateway, until all data resides only in the target tables, and all applications have been consolidated into a single, master MS Access application linked to the target tables.

The audit table layer requires it's source to support database triggers and other high-performance Relational Database Management System (RDBMS) features not found within the MS Access technical platform, which is why the tables are moved to SQL Server.

**Audit Table Layer**

The audit layer is comprised of two types of objects: 1) tables to hold the contents of the CRUD operations, and 2) a set of read-only views taht display only the current values for every record in the system.

Database Table Name / Audit Table / View

```
GasDB1Customer
GasDB1Customer_audit
GasDB1Customer_view
```

Every table in the SQL Server database instance has a sister audit table that is filled by a single insert for every CRUD operation. For example, if GasDB1.Customer table has 500, records and 250 records receive some sort of CRUD operation, then the GasDB1Customer_audit table will contain 750 records. Every audit table uses an effective dating algorithm to keep track of the history of a record. This is performed by adding three new columns to every audit table: 1) effective_start_datetime, 2) effective_end_datetime, and 3) current_YN. In this example, only 500 of the 750 records in the audit table will have a current_YN='Y', while 250 records will have the current_YN='N'.

When a record is first created at Time1, its effective_start_date = sysdatetime, while its effective_end_datetime will equal something like 1/1/2050 12:00. One second later, another CRUD operation occurs against the same record located in the seven databases in the SQL Server layer, and a new record is inserted into the corresponding audit table at Time2 with the default datetime, but then the previous record's effective_end_datetime is updated from 1/1/2050 12:00 to sysdatetime.

Time1: INSERT GasDB1.Customer.Name = ‘Fred’.

Audit table: INSERT GasDB1.Customer_audit.Name = ‘Fred’,
effective_start_datetime=’201509210929’,
effective_end_datetime= ‘205001011200’,
current_YN = ‘Y’.

Then, one second later…

Time2: UPDATE GasDB1.Customer.Name = ‘Fredrick’,

Audit table: UPDATE GasDB1.Customer_audit WHERE Name = ‘Fred’
SET effective_end_datetime= ‘205001011230’
current_YN = ‘N’.

Audit table: INSERT GasDB1.Customer_audit.Name = ‘Fredrick’,
effective_start_datetime=’201509210929’,
effective_end_datetime= ‘205001011200’,
current_YN = ‘Y’.

The above describes how records flow into the _audit tables to form a history for every record change. The _audit tables allow us to answer the question: What was the state of every single record in the system given a particular point in time, say, 11:59am yesterday. Data consistency across all database tables becomes very important when pulling the data into the ETL process.

The _views, are stored queries written against the _audit tables that also leverage the effective dating algorithm to display only the most current instance of a given record. Simply put, WHERE (table_name)_audit.current_YN=’Y’.

One benefit to this design is that the operation of refreshing stage from the audit layer, as opposed to directly querying the seven live SQL Server databases, this design insures production performance will not impacted negatively by buffering all the records into the audit layer. Without the audit layer, a large data pull directly against the seven live SQL Server database would adversely effect the production application’s performance by potentially blocking or locking a user’s access to the database tables during the data pulling operation.

Please note that no applications are allowed to access the audit table layer directly.

**Source – Staging**

As often as is necessary, the staging environment is refreshed with data from the audit layer.

Stage consists of a single snapshot of all seven database tables at a single point in time. No database applications have access to the staging environment; its sole purpose is to be the source for the forthcoming ETL operation.

**Gateway / ETL**

The gateway is where ETL (extract from source, transform, and load into the target tables) occurs. Application and data transformation management is tightly coupled within the ETL logic. As an application change or database change occurs to consolidate or conform, the ETL is adjusted, and the applications are modified for the next release. Application transformation management is controlled by updates to the usys_user_role_object_rights table.

A source-to-target data mapping document is developed to aid in the communication and vetting process between IT and business subject matter experts (SMEs).

For the CPUC project, we’ll implement the ETL described within the source to target data mapping document in Microsoft SQL Server Integration Services (SSIS).

**Target – Destination**

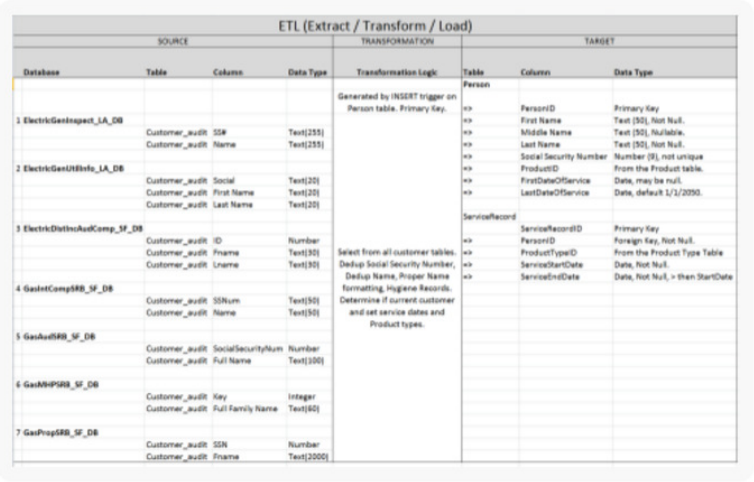
The target is a fully normalized database schema in which all seven database architectures have been consolidated and conformed into a single source. Over time, the source will contain fewer and fewer tables and the target will contain more and more tables, until all the tables and the applications are consolidated into a single application connected to only tables in the target schema, and the ETL jobs will be turned off. This is an ideal state.

The overall conceptual solution architecture will continue to be leveraged after the seven MS Access database applications have been consolidated, and other legacy database applications that are good candidates for nurturing to a common technology stack will be managed through this established transformational process.

## Example of some of the physical solution artifacts within Step 4

### Source-to-target data mapping spreadsheet

A source-to-target data mapping spreadsheet is developed to aid in the gathering of business knowledge from each department's subject matter experts who currently support the seven different legacy MS Access applications and a senior principal business analysis from the technical team. A completed mapping document becomes the specification for ETL jobs. A sample ETL job is shown below.



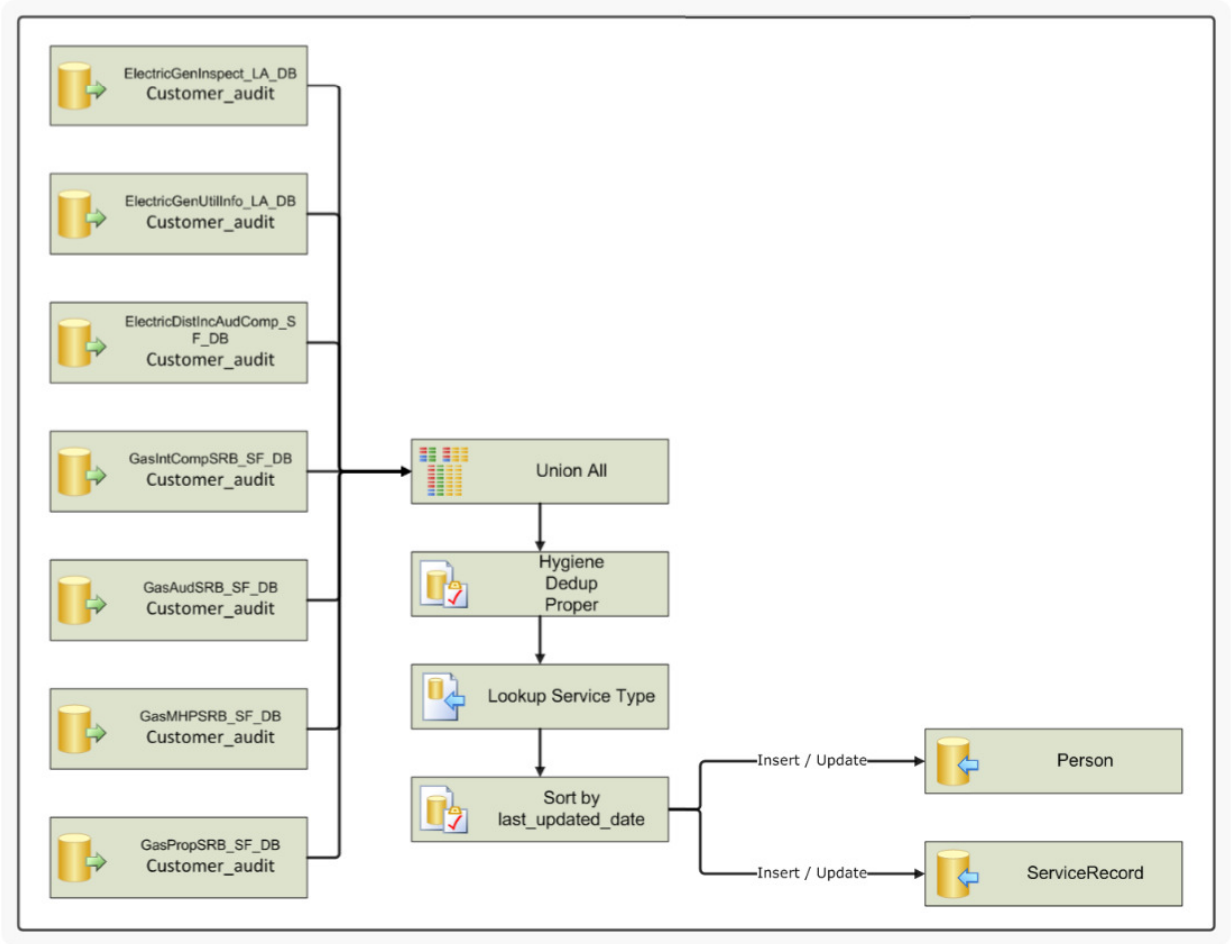
ETL (Extract / Transform / Load)

| SOURCE | | | | TRANSFORMATION | TARGET | | |
|---|---|---|---|---|---|---|---|
| Database | Table | Column | Data Type | Transformation Logic | Table | Column | Data Type |
| | | | | Generated by INSERT trigger on Person table. Primary Key. | Person | | |
| 1 ElectricGenInspect_LA_DB | | | | | => | PersonID | Primary Key |
| | Customer_audit | SS# | Text(255) | | => | First Name | Text (50), Not Null. |
| | Customer_audit | Name | Text(255) | | => | Middle Name | Text (50), Nullable. |
| | | | | | => | Last Name | Text (50), Not Null. |
| 2 ElectricGenUtilInfo_LA_DB | | | | | => | Social Security Number | Number (9), not unique |
| | Customer_audit | Social | Text(20) | | => | ProductID | from the Product table. |
| | Customer_audit | First Name | Text(20) | | => | FirstDateOfService | Date, may be null. |
| | Customer_audit | Last Name | Text(20) | | => | LastDateOfService | Date, default 1/1/2050. |
| | | | | | ServiceRecord | | |
| 3 ElectricDistIncAudComp_SF_DB | | | | | | ServiceRecordID | Primary Key |
| | Customer_audit | ID | Number | | => | PersonID | Foreign Key, Not Null. |
| | Customer_audit | Fname | Text(30) | Select from all customer tables. Dedup Social Security Number, Dedup Name, Proper Name formatting, Hygiene Records. Determine if current customer and set service dates and Product types. | => | ProductTypeID | from the Product Type Table |
| | Customer_audit | Lname | Text(30) | | => | ServiceStartDate | Date, Not Null. |
| 4 GasIntCompSRB_SF_DB | | | | | => | ServiceEndDate | Date, Not Null, > then StartDate |
| | Customer_audit | SSNum | Text(50) | | | | |
| | Customer_audit | Name | Text(50) | | | | |
| 5 GasAudSRB_SF_DB | | | | | | | |
| | Customer_audit | SocialSecurityNum | Number | | | | |
| | Customer_audit | Full Name | Text(100) | | | | |
| 6 GasMHPSRB_SF_DB | | | | | | | |
| | Customer_audit | Key | Integer | | | | |
| | Customer_audit | Full Family Name | Text(80) | | | | |
| 7 GasPropSRB_SF_DB | | | | | | | |
| | Customer_audit | SSN | Number | | | | |
| | Customer_audit | Fname | Text(2000) | | | | |

### ETL Job – SQL Server Integration Services – (SSIS)

An extract, transform, and load (ETL) job is a physical implementation of the above source-to-target data mapping spreadsheet. This is just one of many ETL jobs required to conform the seven MS Access databases into a single, normalized schema.

During the initial iterations through step 4, data that is unconformable systematically (out of bounds) requires data cleanup application screens to be developed to aid the department's subject matter exports in the processing of cleansing and conforming the data.



We will modernize the application by iteratively migrating portions of the application to the target technology, for example, C, C++, Java, C#, Objective-C, PHP, Python, Ruby, or Oracle's Hyperion.

Very few companies want to eliminate all of their MS Access applications. The best results are achieved by focusing on migrating applications that are limiting business efficiency and agility.

How do you decide which business functions are limiting business efficiency and therefore are strong candidates for migration? Typically, it's the core business functions that make the organization unique and competitive and warrant investment. Non-core business functions are ancillary to the organization and are better candidates for replacing with COTS applications or a large ERP system when the time is right.

The seven database applications at the CPUC all support core-business functions, which monitor the compliance of those facilities producing and distributing gas and electricity in California. The purpose of the Data Entity/Business Function matrix and the System/Matrix is to understand the dependencies between the database tables, application objects and business functions, with the goal of pulling out particular services and migrating them to a new technical platform, without disturbing the remaining functions.

## 1. Data Entity / Business Function Matrix

The purpose of the Data Entity/Business Function matrix is to depict the relationship between the data entities and the business functions.

The mapping enables the following: assignment of ownership of entities, understanding of the data and information exchange, support of the gap analysis and a determination of where any entities are missing, definition of the system of origin, system of record, and system of reference for data entities, and the development of data governance programs across the organization (establish data steward, develop data standards pertinent to the business function, etc.).

## 2. System / Data Matrix

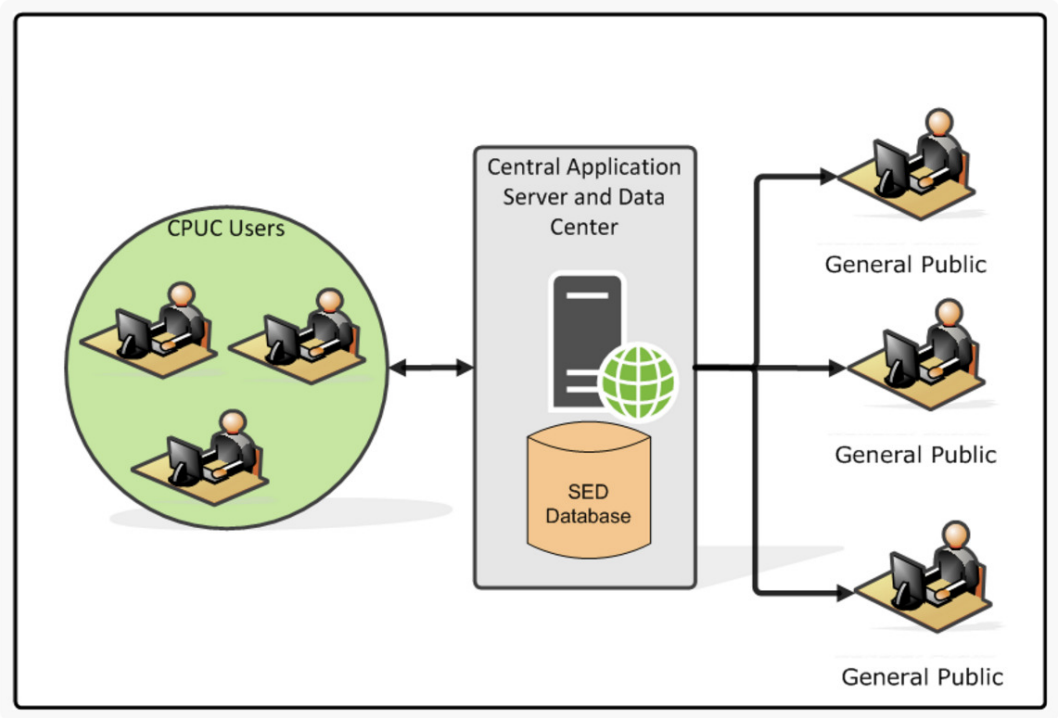
The purpose of the System / Data Matrix is to depict the relationship between systems (i.e., application components) and the data entities that are accessed and updated by them.

Systems will create, read, update, and delete (CRUD) specific data entities that are associated with them. For example, the ElectricGenInspect_LA_Frontend will create, read, update and delete customer entity information.

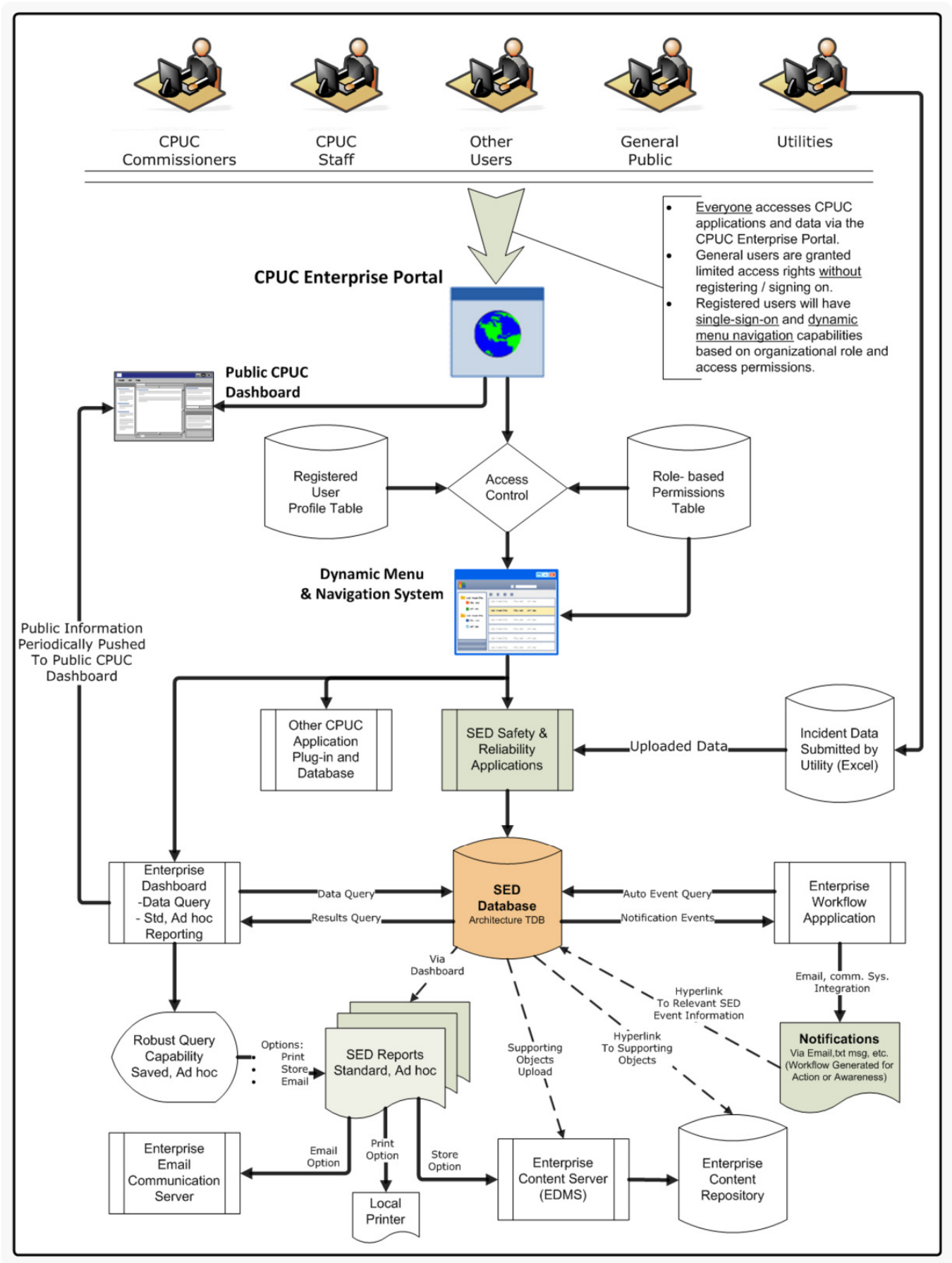
The two matrices are used to help guide which application components to group together when modernizing their business functions on the organization's new standardized application platform.

## Final Solution State – Conceptual

The major benefits are: increased application performance, IT efficiency, a single version of the truth, and application stability.



## Final Solution State – Logical



The Help4Access Application Development Methodology (adopted from The Open Group).


INTRO STEP 1 STEP 2 STEP 3 STEP 4 STEP 5