**Wells Fargo, San Francisco, CA Jan 2011 – Present**

**Sr. Business/Data Analyst**

**Foreign Exchange Online System (FXOL 2.0)**

The **Foreign Exchange Online System** **FXOL 2.0** project is the combination of the original FXOL legacy project and the Wachovia Integration Gap project which resulted in one online foreign exchange application replacing the existing Wells Fargo FXOL and Wachovia’s OnlineFX applications serving the external customers of the combined organization. The FXOL 2.0 applications entailed all the enhancements with new features and FX reporting. This has capability to create new FX contracts i.e. Drawdown and Netting functionalities.

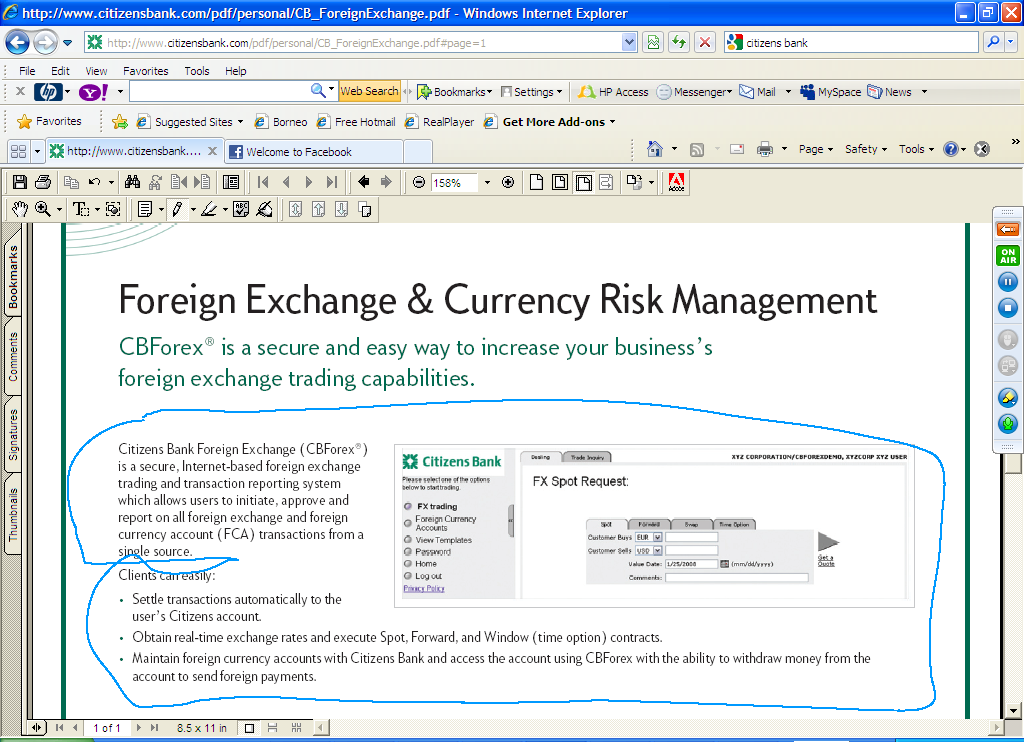
**Responsibilities:**

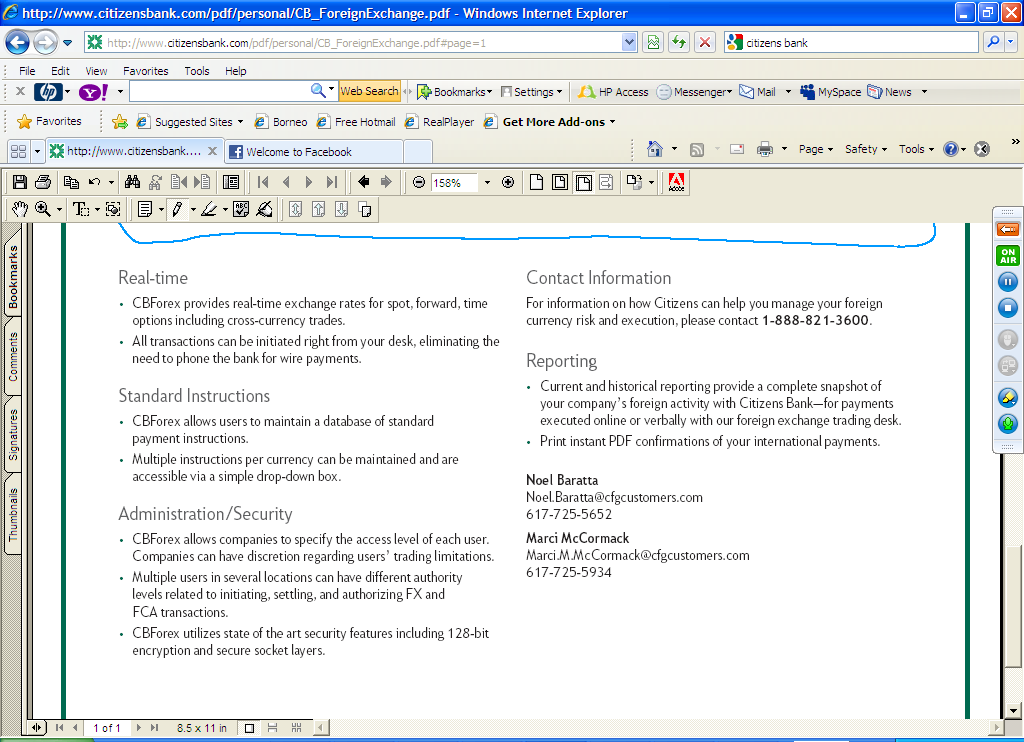
* Created SSIS packages to clean and load data to data warehouse.
* Created package to transfer data between OLTP and OLAP databases.
* Created SSIS Packages using Pivot Transformation, Fuzzy Lookup, Derived Columns, Condition Split, Term extraction, Aggregate, Execute SQL Task, Data Flow Task, and Execute Package Task etc. to generate underlying data for the reports and to export cleaned data from Excel Spreadsheets, Text file, MS Access and CSV files to data warehouse.
* Created SSIS packages for data Importing, Cleansing, and Parsing etc. Extracted, cleaned and validated
* Validated using SQL and reported bugs in Clear Quest.
* Created complex queries to automate data profiling process needed to define the structure of the pre staging and staging area.
* Extracted data from XML files.
* Migrating of 30 Data Objects from 5 different Source (Legacy) Systems into SAP ECC 6.0 complimenting the work being done by the SAP implementation team including Vendor Master, Material Master, Bank Master, Freight Conditions, Service Master, Service / Material Contracts etc
* Scheduled automated reports through Cognos Web and Cognos Script Editor/Scheduler
* Generated reports using SQL Server Reporting Services 2005/2008 from OLTP and OLAP data sources.
* Deploying and scheduling Reports using SSRS to generate all daily, weekly, monthly and quarterly Reports including current status.
* Identifying the fields in SAP and mapping the fields in source systems to equivalent SAP tables/fields
* Worked on all aspects of the data warehouse lifecycle Requirements gathering, architecture, design, data modeling, ETL, Reporting, performance tuning, testing, end user training and production support.
* Created ad-hoc reports, drill down, and drill through reports.
* Deployed reports on Microsoft office SharePoint (Moss).
* Configuration expertise in the Order To Cash (OTC) and Procure To Pay (PTP) areas
* Created user defines functions in SSRS using VB script.
* Reviewed and adjusted deliverables involved during SDLC involving database changes/development.
* Generated test data and tested database to meet the functionalities deliverables in the project documentation and specifications
* Wrote distributed queries to extract data from different servers.
* Used Execution Plan, SQL Profiler and Database Engine Tuning Advisor to optimize queries and enhance the performance of databases
* Optimized the performance of various SQL scripts, stored procedures and triggers by using embedded UDFs, CTEs and System stored procedures.
* Successfully tested all task in staging and stored all versions of data using team foundation server (TFS).

This project was in the Cash Management division. The project Foreign Exchange Trading System Application was the enhancement of the trading system that Wells Fargo Bank used for foreign currency exchange for UK, Canada, and AUS!

This project was in the cash management division. It was foreign exchange trading application. We were enhancing the system that Wells Fargo bank (in USA) was using for UK, Canada and Australia. Earlier, they were using Paragon application which was not able to satisfy the client’s (institutional investors a.k.a. wholesale customers as well as regular high net worth individuals a.k.a. retail customers) needs to trade in international markets. However, using Fidessa LatentZero (I worked on implementing it), they could trade in international markets using US accounts. Before, when they were using Paragon application, Wells Fargo banks clients could not buy UK, Canada and Australian stocks directly. They had to go through a stock trader in Wells Fargo bank and route the trades through them. Wells Fargo banks clients were stuck only trading in US stocks by themselves. With Fidessa LatentZero, a Wells Fargo bank client (like John Doe) can trade international securities sitting in USA. LatentZero was exposed to end clients like you and me.

**I worked on an application called CBForex which was used by Wells Fargo bank’s customers. These customers could be retail customers or could be wholesale customers.**

****

****

**Where was the Forex information coming from?**

**Bloomberg?**

**How did you make sure about the security of the application?**

**https; the clients that we have (companies) could give different levels of access to its employees. So, the CEO/CFO/COO level people had no trading limitations but say the HR Manager had restricted accounts which needed approval from C level executives beyond a certain amount. I was responsible for setting up this roles and limitations matrix!**

# Project Background

In 2008, Wells Fargo acquired Wachovia after a government-forced sale to avoid a failure of Wachovia. My project was to integrate database of Wachovia’s international customers with Wells Fargo’s Foreign Exchange Online System.

# Project Detail

For this integration project my team has to increase Database storage capacity, scalability and performance as number of international customers going to increase.

# Team Size

1 Sr. Business Analyst + 1 Project Manager +2 Data Analysts+ 2 Design Architects + 2 Developers + 2 QA analyst + 1 Project Sponsor =11

# Where you were working (Address)?

1 Montgomery St (between Post St & Sutter St)

San Francisco, CA 94104

Neighborhood: Financial District

(415) 396-7152

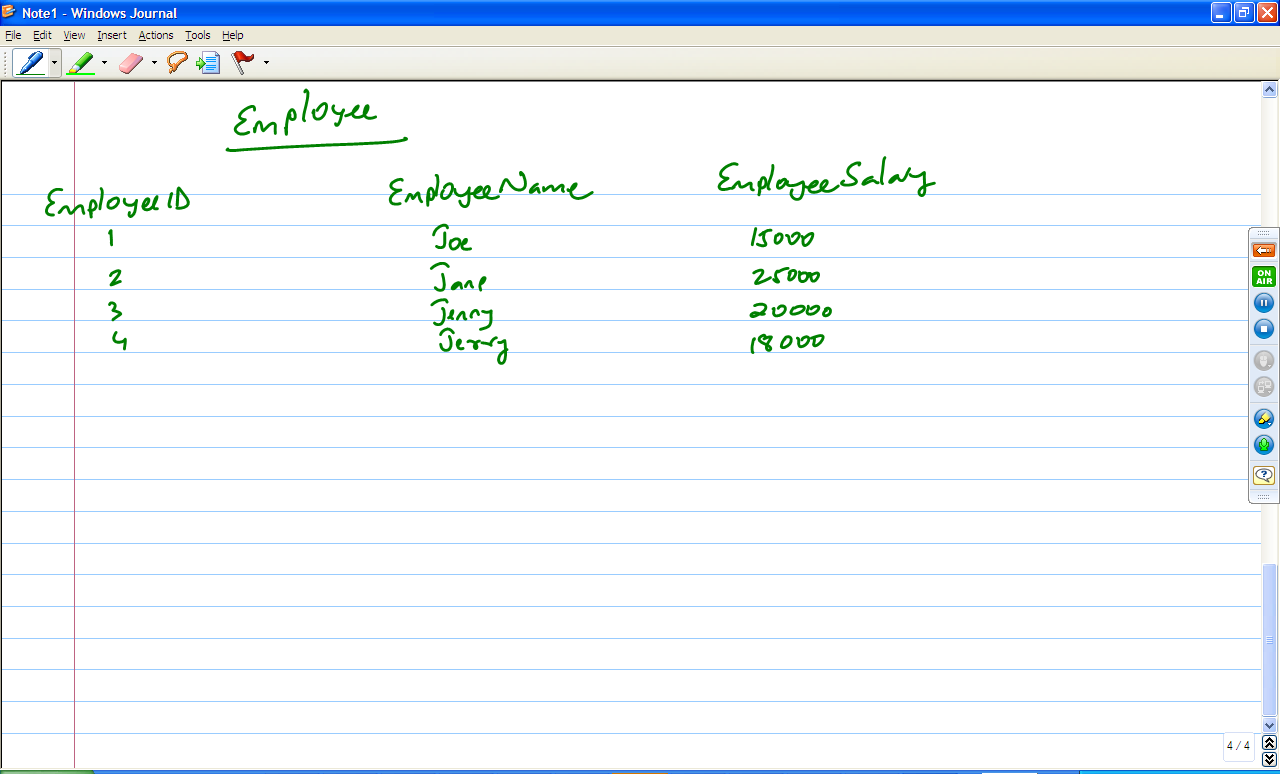
**Efficient with Triggers and store procedures**

**PL/SQL; Stored Procedures and Triggers**

PL SQL stands for Procedural Language SQL. SQL has a short coming. SQL is not a programming language. Programming languages has basic programming constructs like IF THEN ELSE, LOOP, CASE, etc. SQL does not have such constructs.

PL SQL is a full featured programming language that is built on top of SQL.

Say you had an Employees table that had the columns



Give 20% raise to those employees who make less than 18000 and 5% raise to those who make more than 18000 (SQL can’t help you with this query. You have to use PL/SQL because we need to use IF THEN ELSE statement.

I have written stored procedures and tested them. Writing stored procedures is really simple if you know PL/SQL. E.g., I wrote stored procedures in my Loan Origination System project in Wells Fargo Bank to post back the interest onto an account based on the interest rate.

To test a stored procedure, I would use the EXEC command (EXEC command is how you invoke a stored procedure).

I have also worked with triggers.

**What do you understand by a database trigger?**

A trigger is a structure that is fired when a DML (Data Manipulation Language – UPDATE, INSERT, DELETE) sql query is executed. It is automatically fired.5

Syn: create trigger <<Trigger name>> on <<table name>>

When

Begin

Sql stmts

End

As an example from my projects, every time a deposit or a withdrawal is made to a customer account, then the balance has to be updated and the history of customer balances will have to be maintained. So a trigger was created on the customer table to take care of this and every time a DML statement like update is run, this trigger would automatically be fired and the customer account history table will be updated with the old balance.

**Data Models**

There are mainly three types of data models:

**Conceptual data model**: It gives the high level overview of the relationships between the entities in the model.

* It includes important entities and the relationships between them.
* No attributes are specified
* No primary or foreign keys are specified.

**Logical data model**: It take the data model further by defining attributes for each entity and establishing relationship between in the form of primary key and foreign key.

* Includes all entities and relationships among them.
* All attributes for each entity are specified.
* The primary key for each entity is specified.
* Foreign keys are specified.
* Normalization occurs at this level.

Steps for designing logical model:

* Specify primary keys for all entities.
* Find the relationships between different entities.
* Find all attributes for each entity.
* Resolve many-to-many relationships.
* Normalization.

Difference between conceptual and logical model:

* In a logical data model, primary keys are present, whereas in a conceptual data model, no primary key is present.
* In a logical data model, all attributes are specified within an entity. No attributes are specified in a conceptual data model.
* Relationships between entities are specified using primary keys and foreign keys in a logical data model. In a conceptual data model, the relationships are simply stated, not specified, so we simply know that two entities are related, but we do not specify what attributes are used for this relationship.

**Physical data model**: A physical data model shows all the table, column name, column data type, column constraints, primary key, foreign key, relationships between the tables.

Steps for designing physical data model:

* Convert entities into tables.
* Convert relationships into foreign keys.
* Convert attributes into columns.
* Modify the physical data model based on physical constraints / requirements.

Differences between logical data model and physical data model:

* Entity names are now table names.
* Attributes are now column names.
* Data type for each column is specified. Data types can be different depending on the actual database being used.

**End to End Database Development Life Cycle Process**

**Data Analysis**

Data analysis is the process of inspecting, cleaning, transforming and modeling raw data into useful information in order to reach a certain conclusion or decision. The process of organizing the data to interpret the trend with the help of charts, graphs or textual write-ups is known as Data Analysis

**Data Mining**

Data mining is the process of analyzing data from different perspectives and summarizing it into useful information

**Data Warehousing**

Data warehouse is a database (or multiple databases) that specifically exist to allow companies to query the gazillions of data that this company has. SO, a data warehouse is also in essence, just a bunch of databases grouped together to provide information to the company about the state of our business.

**Data Mapping**

Data mapping is an exercise to figure out where does data come from, where does data go and is there any reason to transform it

**Data Profiling**

**Data profiling** is the process of examining the data available in an existing data source (e.g. a [database](http://en.wikipedia.org/wiki/Database) or a [file](http://en.wikipedia.org/wiki/Computer_file)) and collecting [statistics](http://en.wikipedia.org/wiki/Descriptive_statistics) and information about that data. The purpose of these statistics may be to:

* Find out whether existing data can easily be used for other purposes
* Improve the ability to search the data by [tagging](http://en.wikipedia.org/wiki/Tag_(metadata)) it with [keywords](http://en.wikipedia.org/wiki/Keywords), descriptions, or assigning it to a category
* Give [metrics](http://en.wikipedia.org/wiki/Software_metric) on [data quality](http://en.wikipedia.org/wiki/Data_quality), including whether the data conforms to particular standards or patterns
* Assess the risk involved in [integrating data](http://en.wikipedia.org/wiki/Data_integration) for new applications, including the challenges of [joins](http://en.wikipedia.org/wiki/Join)
* Assess whether [metadata](http://en.wikipedia.org/wiki/Metadata) accurately describes the actual values in the source database
* Understanding data challenges early in any data intensive project, so that late project surprises are avoided. Finding data problems late in the project can lead to delays and cost overruns.
* Have an enterprise view of all data, for uses such as [master data management](http://en.wikipedia.org/wiki/Master_data_management) where key data is needed, or [data governance](http://en.wikipedia.org/wiki/Data_governance) for improving data quality.

**Monitor deterioration in data quality**

**Data Services**

**Data Flow Diagram**

**Process Dependency**

**Data Dictionary**

It is a Metadata repository; it contains information about data such as meaning, relationship to other data, origin, usage and format. It provides information about the database and has the following features:

* The definitions of all schema objects in the database (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on)
* How much space has been allocated for, and is currently used by, the schema objects
* Default values for columns
* Integrity constraint information
* The names of Oracle users
* Privileges and roles each user has been granted
* Auditing information, such as who has accessed or updated various schema objects
* Other general database information

So, it could simply be a MS Word document that describes each table we have, the columns of various tables, the description of columns (why is that column there, what is the purpose of it?; sometimes columns are deprecated so this data dictionary will contain information about it;).

**How is it different from a database schema?** Data dictionary is a document that describes the various tables and columns and relationships and the reasons for columns to exist.

**Why is it required?** So, users of the database (programmers, end users, business teams) can get the most out of the database. So that they can truly understand the layout of the tables, what columns mean what?

Say there were two columns, one that said TStamp1 and TStamp2. These column names are not very descriptive or intuitive in what they mean (or what kind of data they hold). In the data dictionary, the creator of this table can say “TStamp1 is used to store the last login date for the user. TStamp2 is used to store the last logout date for the user”.

Meta data is data about the data! So, data dictionary is the way you give the world information about meta data!

**How does one create data dictionary?** One can do it manually! But, the problem with this approach, if the database schema is changing fairly often is that one can end up spending a lot of time doing this. I have used a tool called StoneField to create data dictionary (You can elaborate it once you get the basic dictionary).

**SOAP**

**Case Tools**

**Computer-aided**[**software engineering**](http://en.wikipedia.org/wiki/Software_engineering) (**CASE**) is the scientific application of a set of tools and methods to a [software](http://en.wikipedia.org/wiki/Software) system which is meant to result in high-quality, defect-free, and maintainable software products. It also refers to methods for the development of [information systems](http://en.wikipedia.org/wiki/Information_system) together with automated tools that can be used in the [software](http://en.wikipedia.org/wiki/Software_development_process)

The term "computer-aided software engineering" (CASE) can refer to the [software](http://en.wikipedia.org/wiki/Software) used for the automated development of [systems software](http://en.wikipedia.org/wiki/Systems_software), i.e., computer code. The CASE functions include analysis, design, and programming. CASE tools automate methods for designing, documenting, and producing structured computer code in the desired [programming language](http://en.wikipedia.org/wiki/Programming_language).

CASE software supports the [software process](http://en.wikipedia.org/wiki/Software_process) activities such as requirement engineering, design, program development and testing. Therefore, CASE tools include design editors, data dictionaries, compilers, debuggers, system building tools, etc.

CASE also refers to the methods dedicated to an engineering discipline for the development of information system using automated tools.

CASE is mainly used for the development of quality software which will perform effectively.

**Rest**

Representational state transfer (REST) is a style of [software architecture](http://en.wikipedia.org/wiki/Software_architecture) for [distributed](http://en.wikipedia.org/wiki/Distributed_computing) [hypermedia](http://en.wikipedia.org/wiki/Hypermedia) systems such as the [World Wide Web](http://en.wikipedia.org/wiki/World_Wide_Web). The term *representational state transfer* was introduced and defined in 2000 by [Roy Fielding](http://en.wikipedia.org/wiki/Roy_Fielding) in his doctoral dissertation. Fielding is one of the principal authors of the [Hypertext Transfer Protocol](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) (HTTP) specification versions 1.0 and 1.1.

EST-style architectures consist of [clients](http://en.wikipedia.org/wiki/Client_(computing)) and [servers](http://en.wikipedia.org/wiki/Server_(computing)). Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of representations of resources. A[re source](http://en.wikipedia.org/wiki/Resource_(Web)) can be essentially any coherent and meaningful concept that may be addressed. A [representation](http://en.wikipedia.org/wiki/Representation_(systemics)) of a resource is typically a document that captures the current or intended state of a resource.

The client begins sending requests when it is ready to make the transition to a new state. While one or more requests are outstanding, the client is considered to be in transition. The representation of each application state contains links that may be used next time the client chooses to initiate a new state transition.[[7]](http://en.wikipedia.org/wiki/Representational_state_transfer#cite_note-6)

REST was initially described in the context of [HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol), but is not limited to that protocol. RESTful architectures can be based on other [Application Layer](http://en.wikipedia.org/wiki/Application_Layer) protocols if they already provide a rich and uniform vocabulary for applications based on the transfer of meaningful representational state. RESTful applications maximize the use of the pre-existing, well-defined interface and other built-in capabilities provided by the chosen network protocol, and minimize the addition of new application-specific features on top of it.

**HTTP examples**

HTTP, for example, has a very rich vocabulary in terms of verbs (or "methods"), [URIs](http://en.wikipedia.org/wiki/Uniform_Resource_Identifier), [Internet media types](http://en.wikipedia.org/wiki/Internet_media_type), request and [response codes](http://en.wikipedia.org/wiki/List_of_HTTP_status_codes), etc. REST uses these existing features of the HTTP protocol, and thus allows existing layered proxy and gateway components to perform additional functions on the network such as HTTP caching and security enforcement.

**SOAP RPC contrast**

[SOAP](http://en.wikipedia.org/wiki/SOAP) [RPC](http://en.wikipedia.org/wiki/Remote_procedure_call) over HTTP, on the other hand, encourages each application designer to define a new and arbitrary vocabulary of nouns and verbs (for example get Users(), save Purchase Order(...)), usually overlaid onto the HTTP [POST](http://en.wikipedia.org/wiki/POST_(HTTP)) verb. This disregards many of HTTP's existing capabilities such as authentication, caching and content type negotiation, and may leave the application designer re-inventing many of these features within the new vocabulary.[[8]](http://en.wikipedia.org/wiki/Representational_state_transfer#cite_note-Scribner_and_Seely-7) Examples of doing so may include the addition of methods such as get New Users Since(Date date), save Purchase Order(string customer Logon, string password, ...).

**Embarcadero**

**Data Flux**

**DataFlux** Corporation, a wholly owned subsidiary of SAS Institute headquartered in Cary, North Carolina, is a software provider of data management technology and services

**Products:**

DataFlux software products include solutions for both IT and business users. For example, the DataFlux Data Management Platform provides organizations with the ability to plan and complete [data integration](http://en.wikipedia.org/wiki/Data_integration), [data quality](http://en.wikipedia.org/wiki/Data_quality) and [MDM](http://en.wikipedia.org/wiki/MDM) projects from a single interface, while business solutions like [data governance](http://en.wikipedia.org/wiki/Data_governance) and [data optimization](http://en.wikipedia.org/w/index.php?title=Data_optimization&action=edit&redlink=1) enable users to manage their existing data.

DataFlux works with organizations that are pursuing a better way to manage data within customer relationship management systems ([CRM](http://en.wikipedia.org/wiki/Customer_relationship_management)), enterprise resource planning systems ([ERP](http://en.wikipedia.org/wiki/Enterprise_resource_planning)) and [data warehouses](http://en.wikipedia.org/wiki/Data_warehouses).

* [DataFlux Data Management Platform](http://www.dataflux.com/Products/Data-Management-Studio.aspx)
* [DataFlux Servers](http://www.dataflux.com/Products/Servers.aspx)
* [DataFlux qMDM](http://www.dataflux.com/Products/qMDM.aspx)
* [DataFlux Accelerators](http://www.dataflux.com/Products/Accelerators.aspx)
* [DataFlux Connect](http://www.dataflux.com/Products/DataFlux-Connect.aspx)

**Data Scrubbing**

**Data scrubbing** is an error correction technique which uses a background task that periodically inspects memory for errors, and then corrects the error using [ECC memory](http://en.wikipedia.org/wiki/ECC_memory) or another copy of the data. It reduces the likelihood that single correctable errors will accumulate; thus, reducing the risk of uncorrectable errors.

**data marts**

Datamart is a subset of data warehouse which is geared towards a specific LOB (Line of business). E.g., the marketing/sales department of a company (like Cisco) would want to have only information related to products/prices/markets/demand/supply etc. So, a subset of data warehouse will be created specifically for the use of Sales/Marketing team. Similarly, the Finance team will need information on profits/losses/assets/liabilities/etc. So, a DW expert will create a subset of data warehouse called Finance Data Mart

**Data Management Methodology**