Carlos Sathler | cssathler@gmail.com
INFO-I 535 – MGMT ACCESS USE BIG DATA (Fall 2016)

Data Processing Pipelines in Science

1. The lecture discussed the data lifecycle. The data lifecycle can be seen as analogous to the human lifecycle or lifecycle of a house pet. It is uniquely oriented around the object and views the world from the perspective of the species or the individual. In a data lifecycle view of the world, data is not merely something that is input to computing; it has independent existence and continuity. Describe the similarities and differences between data lifecycle and a data pipelines.

In the video for this class Dr. Plale states: "the pipeline is what facilitates the sequence of actions that are applied to the set of data objects… it's software… that supports the steps in the is data life cycle".

So, on the one hand, the data pipeline and the data lifecycle are different in the sense that the latter describes data as an entity as it is transformed through time while the former describes a set of operations that serve as a conduit through which these transformations are effected. On the other hand, the data pipeline and the data lifecycle are similar in the sense that there is a parallel, in the form of a somewhat direct mapping between the data lifecycle *stages* and the data pipeline *operations*.

In the table below we try to capture this mapping using information drawn from the class video lectures:

Data Life Cycle Phase (per class video lecture)	**Data Pipeline Operation (per class video lecture)**
Plan Collect Assure	Data capture
Describe Preserve	Data curation : schemas, ontologies, provenance
Discover Integrate Analyze	Data analysis : workflow, algorithms, databases, data visualization
Report, publication produced	Data + document publication : active docs, data-doc integration Access : documents + data archives

Carlos Sathler | cssathler@gmail.com
INFO-I 535 – MGMT ACCESS USE BIG DATA (Fall 2016)

2. From your own experience in school, business, or research, give an example of a data pipeline. If you know of none from your experience, create a scenario where one would be needed. Your example should have enough detail to expose the different steps needed in the pipeline.

I've worked implementing ERP Human Resource Management (HCM) systems for the past 15+ years, particularly PeopleSoft. Many data pipelines are needed for the delivery of a complete ERP HCM solutions. These solutions require regular data exchanges between the HCM ERP application and service providers of different kinds, such as financial institutions and benefit companies. For example, when an employee is paid via a direct deposit to her bank account that is because there is a pipeline between the ERP system and the bank. This pipeline runs once every pay period, so that employees are paid on time (weekly, semi-monthly, etc). Another example: when an employee receives a participant card on a benefits plan that is because there is a pipeline between the ERP systems and the company's benefits provider. That pipeline runs once a year, after benefits open enrollment which typically occurs beginning of Q4 for most companies.

One pipeline that offers a good example of data collection at a place other than where it is processed is one associated with payment of hourly employees. These employees typically clock their time on a time and attendance application. Often their time data is then sent as a flat file to a system that could be a mainframe, and a job running sometimes on a Unix server will pick up that file, invoke a program that will read the file and update the ERP HCM system with the employees' time data, so a paycheck can be generated for the employee in the right amount. The Unix job controls the process will not only update the system, but also generate summary reports and create log files with warnings and error messages.

These pipelines are called "interfaces" in the ERP world, and they include all the steps mentioned by Dr. Plale in her lecture: data collection, curation, data analysis, documentation and access. Clearly there is data collection in the example above when employees clock their time; there is data curation because when the system is designed, all record structures, along with field semantics are defined and well documented. When a time data file is created, we know its provenance, and the file is time stamped and placed on a proper location in the file system. When the files are read, time data is "analyzed" automatically and processed so it is grouped, totalized, interpreted (e.g. regular time or overtime?) until it is used to update the ERP HCM system database. Reports and log files are generated (more documentation), properly distributed (sometimes an email is sent to users in the Human Resource department) and in the end designated users will have access to all the information they should be able to access, based on the user organization business rules.

I also thought of a more interesting, perhaps less traditional case of a pipeline, because it only run about 5 times, unlike the pipelines described above, which were recurring, sometimes weekly run pipelines.

In 2001 I wrote a large application in Perl to convert a mainframe batch system to a Unix batch system. By "mainframe batch system" I mean a collection of JCL scripts [1,2], the programs they invoke (e.g. Cobol, reporting applications), the mainframe utilities they launch (e.g. archiving, file transfer) and the application data files they manipulate and archive. The Perl application to convert this batch system was developed in the context of a PeopleSoft Human Resource [3] upgrade. I was a technical lead consultant. My client was migrating their PeopleSoft database from DB2 (residing in Mainframe) to Oracle (to reside in Unix), as part of a PeopleSoft version upgrade.

Carlos Sathler | cssathler@gmail.com
INFO-I 535 – MGMT ACCESS USE BIG DATA (Fall 2016)

The application I wrote ran in Unix. In order for an automated solution to be conceived and developed, all steps in the data pipeline were first performed manually or in a semi-automated fashion. That allowed for the design of a fully automated solution. The table below lists the high level operations the automated perl application performed, and associates each of them with a data pipeline operation, as described in the class video lecture. This pipeline was executed several times, not only for testing purposes, but also to support the PeopleSoft Human Resource project test cycles, which included unit testing, system testing and user acceptance testing. Of course, the pipeline was run one last time when the PeopleSoft Human Resource project went live.

Perl application operation	**Source (input)**	**Source region**	**Target (output)**	**Target region**	**Pipeline operation**
Read all PeopleSoft application JCL scripts	JCS scripts	Mainframe	N/A	Unix	Data capture
Parse JCL script to classify its contents: what does each code snippet do? What dataset(s) does it read and what dataset(s) does it create/modify? Where is the data stored?	JCL scripts	Mainframe	MS Word Visio Diagrams Design Document	Windows NT	Data curation This step was automated only to the extent that each JCL script type was properly placed per pre-conceived design into a pre-defined location in the Unix file system. The location could be thought of as a tag to classify the object.
Convert JCL scripts into fully functional ksh scripts	JCL scripts	Mainframe	Fully functional PeopleSoft application shell scripts	Unix	Data analysis: each code snipped is converted into a ksh script implementing equivalent functionality
Update all program/tool reference calls, and all dataset/file references	JCL scripts	Mainframe	Shell scripts updated with proper references to programs and files in the Unix file system	Unix	Data analysis: All program calls (e.g. Cobol, SQR [4]) to objects residing in the mainframe were mapped to calls to objects residing in Unix
Transfer mainframe PeopleSoft application datasets from mainframe to Unix file system; convert EBCDIC files to ASCII	Mainframe datasets	Mainframe	Unix file system	Unix	Data analysis: all data objects referenced in the converted Unix ksh scripts had to be transferred from the mainframe into the

Perl application operation	Source (input)	Source region	Target (output)	Target region	Pipeline operation
files.					Unix file system, at the proper location per reference created in the previous step
Permissions granting to application objects Report on application run results	Application run log	Unix	Application execution report	Unix	Data + document publication: All PeopleSoft mainframe application objects (scripts and data) moved and properly from the mainframe and properly placed in the Unix file system, per documented design. Access: Access granted to PeopleSoft application objects (programs and data files) in Unix per business requirements; access granted to application design documents, test document, project plan, etc. per business requirements. All PeopleSoft Human Resource application stakeholders have access to application objects of all kinds per the client business requirements.

[1] https://en.wikipedia.org/wiki/Job_Control_Language

[2] http://www.tutorialspoint.com/jcl/index.htm

[3] http://www.oracle.com/us/products/applications/peoplesoft-enterprise/human-capital-management/overview/index.html

[4] https://docs.oracle.com/cd/E58500_01/pt854pbh1/eng/pt/tsql/concept_TheSQRLanguage-c07b18.html#topofpage