LECTURE 14

ETL Detail: Data Cleansing

Learning Goals

- What is Dirty Data?
- Types of Anomalies
- Automatic Data Cleansing

Background

- Other names: Called as data scrubbing or cleaning.
- More than data arranging.
- Big problem, big effect: Enormous problem, as most data is dirty. GIGO
- Dirty is relative.
- Paradox: Must involve domain expert, as detailed domain knowledge is required, so it becomes semi-automatic, but has to be automatic because of large data sets.
- Data duplication.

It is crucial to the process of loading the data warehouse that you not just rearrange the grouping of source data and deliver it to a destination database, but that you also ensure the quality of that data. Data cleansing is vitally important to the overall health of your warehouse project and ultimately affects the health of your company. Do not take this statement lightly.

The true scope of a data cleansing project is enormous. Much of production data is dirty, and you don't even want to consider what work cleaning it up would take. By "dirty," I mean that it does not conform to proper domain definitions or "make sense." The age-old adage "garbage in, garbage out" still applies, and you can do nothing about it short of analyzing and correcting the corporate data. What is noise in one domain may be information in another.

Usually the process of data cleansing cannot be performed without the involvement of a domain expert because the detection and correction of anomalies requires detailed domain knowledge. Data cleansing is therefore described as semi-automatic but it should be as automatic as possible because of the large amount of data that usually is be processed and the time required for an expert to cleanse it manually.

The original aim of data cleansing was to eliminate duplicates in a data collection, a problem occurring already in single database applications and gets worse when integrating data from different sources. You will have a complete lecture on it.

Lighter Side of Dirty Data

- Year of birth 1995 current year 2005
- Born in 1986 hired in 1985
- Who would take it seriously? Computers while summarizing, aggregating, populating etc.
- Small discrepancies become irrelevant for large averages, but what about sums, medians, maximum, minimum etc.?

Value validation is the process of ensuring that each value that is sent to the data warehouse is accurate. You may had that experience in which you look at the contents of one of your major flat files or database tables and intuitively pick that the data is incorrect. No way could that employee be born in 2004! You know your company doesn't hire infants. You may also discover another incorrect record. How could someone be born in 1978 but hired in 1977?

All too often, these types of data integrity problems are laughed at and then ignored. All too often people say "No one would actually take that information seriously, would they?" Well, maybe people won't, but MIS systems will. That information can be summarized, aggregated, and/or manipulated in some way, and then populated into another data element. And when that data element is moved into the DWH, analytical processing will be performed on it that can affect the way your company does business. What if data from the DWH is being analyzed to revise hiring practices? That data may make a wrong impact on the business decisions if enough of the hire and birth dates are inaccurate.

Small data discrepancies can become statistically irrelevant when large volumes of data are averaged. But averaging is not the only analytical function that is used in analytical data warehouse queries. What about sums, medians, max/min, and other aggregate and scalar functions? Even further, can you actually prove that the scope of your data problems is as small as you think it is? The answer is probably "no."

Serious Problems due to dirty data

- Decisions taken at government level using wrong data resulting in undesirable results.
- In direct mail marketing sending letters to wrong addresses loss of money and bad reputation.

Administration:

The government analyses data collected by population census to decide which regions of the country require further investments in health, education, clean drinking water, electricity etc. because of current and expected future trends. If the rate of birth in one region has increased over the last couple of years, the existing health facilities and doctors employed might not be sufficient to handle the number of current and expected patients. Thus, additional dispensaries or employment of doctors will be needed. Inaccuracies in analyzed data can lead to false conclusions and misdirected release of funds with catastrophic results for a poor country like Pakistan.

Supporting business processes: Erroneous data leads to unnecessary costs and probably bad reputation when used to support business processes. Consider a company using a list of consumer addresses and buying habits and preferences to advertise a new product by direct mailing. Invalid addresses cause the letters to be returned as undeliverable. People being duplicated in the mailing list account for multiple letters sent to the same person, leading to unnecessary expenses and frustration. Inaccurate information about consumer buying habits and preferences contaminate and falsify the target group, resulting in advertisement of products that do not correspond to consumer's needs. Companies trading such data face the possibility of an additional loss of reputation in case of erroneous data.

Classes of Anomalies

- Syntactically Dirty Data

1. Lexical Errors
2. Irregularities

- Semantically Dirty Data

1. Integrity Constraint Violation

2. Business rule contradiction
3. Duplication

- Coverage Anomalies

1. Missing Attributes
2. Missing Records

Syntactically Dirty Data

Lexical errors:

For example, assume the data to be stored in table form with each row representing a topple and each column an attribute. If we expect the table to have five columns because each topple has five attributes but some or all of the rows contain only four columns then the actual structure of the data does not conform to the specified format.

Irregularities are concerned with the non-uniform use of values, units and abbreviations.
This can happen for example if we use different currencies to specify an employee's salary. This is especially profound if the currency is not explicitly listed with each value, and is assumed to be uniform. Annual salary or 20,000 means $20,000 or Rs. 20,000. This results in values being correct representations of facts if we have the necessary knowledge about their representation needed to interpret them.

Semantically dirty data

Integrity constraint violations describe topples (or sets of topples) that do not satisfy one or more of the integrity constraints. Integrity constraints are used to describe our understanding of the mini-world by restricting the set of valid instances. Each constraint is a rule representing knowledge about the domain and the values allowed for representing certain facts.

Contradictions are values within one tuple or between topples that violate some kind of dependency between the values. An example for the first case could be a contradiction between the attribute AGE and DATE_OF_BIRTH for a topple representing persons. Contradictions are either violations of functional dependencies that can be represented as integrity constraints or duplicates with inexact values. They are therefore not regarded as separate data anomaly.

Duplicates are two or more topples representing the same entity from the mini-world. The values of these topples do not need to be completely identical. Inexact duplicates are specific cases of contradiction between two or more topples. They represent the same entity but with different values for all or some of its properties. This hardens the detection of duplicates and there mergence.

Classes of Anomalies…

Coverage or lack of it

Missing Attribute

Result of omissions while collecting the data.

A constraint violation if we have null values for attributes where NOT NULL constraint exists.

Case more complicated where no such constraint exists.

Have to decide whether the value exists in the real world and has to be deduced here or not.

Coverage Problems

Missing values

Result of omissions while collecting the data. A constraint violation if we have null values for attributes where NOT NULL constraint exists. Case more complicated where no such constraint exists. Have to decide whether the value exists in the real world and has to be deduced here or not.

Missing tuples

Result from omissions of complete entities existent in the mini-world that are not represented by tuples in the data collection. Anomalies could further be classified according to the amount of data accounting for the constraint violation. This can be single values, values within a tuple, values within one or more columns of the data collection or tuples and sets of tuples from different relations.

Why Missing Rows?

Equipment malfunction (bar code reader, keyboard etc.)
Inconsistent with other recorded data and thus deleted.
Data not entered due to misunderstanding/illegibility.
Data not considered important at the time of entry (e.g. Y2K).

There can be a number of reasons for missing rows or missing data. For example there may be fault with the hardware for recording or inputting the data, this could be a bar code reader which is faulty and missing part of the UPC (Universal Product Code) or on a low tech level, there could be problem with a keyboard (numeric part) that results in entry of different keys as same keys and only one is kept and all others are removed, which infact corresponded to unique keys. In many cases I cannot even read what I have written, I am sure I am not alone. If I can't read my own handwriting sometime, than in many instances people will be unable to read my handwriting. This means either the data will not be entered or it will be entered incorrectly, even using automatic means using OCR (Optical Character Reader) software's. Another example is the famous (or infamous) Y2K problem, when the first two most significant digits of the year where not recorded, and in many cases it was almost impossible to differentiate between 1901 and 2001!

Handling missing data

- Dropping records.
- "Manually" filling missing values.
- Using a global constant as filler.
- Using the attribute mean (or median) as filler.
- Using the most probable value as filler.

There can be a number of ways of handling missing data, the most convenient being to just drop the records with missing values for certain attributes. The problem with this approach is what do we gain by dropping records with missing values, and how to decide which records to drop i.e. with one missing value or which one or how many missing values? Obviously this is a complex problem, and writing code would not be an easy task, so one easy way out could be to manually fill missing values or use a global constant as a filler. For example if gender is not known for the customers, then filling the gender by (say) NA. For numeric attributes, filling using a global

value may make some sense, as it could be the mean or median of the column value. Or this could also be the most probable value to be used as a filler.

Key Based Classification of Problems

- Primary key problems
- Non-Primary key problems

The problems associated with the data can correspond to either the primary keys or non primary keys, as the nature of the problem and the corresponding solution varies with the type of the key involved. In the subsequent slides we will discuss each of them one by one.

Primary key problems

- Same PK but different data.
- Same entity with different keys.
- PK in one system but not in other.
- Same PK but in different formats.
- Some of the problems associated with PK are;

Records may have the same primary key but might have different data. This can occur if primary keys are reused or when two organizations or units merge.
The same entity might occur with different primary keys. This can easily arise when different segments of an entity design databases independently of one another

Data may have a primary key in one system but not in another. The classic example of this kind of error occurs when an entity is represented in more than one source database. It is quite possible that the entity is central to one of the databases and therefore has a primary key field or fields while the same entity is so peripheral to the purposes of the other database that it is not dignified by being given a primary key.

Primary Keys may be intended to be the same but might occur in different formats. Probably, the most widespread instance of this error occurs when NID are used as primary keys: are they character data or numeric; if character data, do they contain dashes?

Non primary key problems…

- Different encoding in different sources.
- Multiple ways to represent the same information.
- Sources might contain invalid data.
- Two fields with different data but same name.

Data may be encoded differently in different sources. The domain of a "gender" field in some database may be {'F', 'M'} or as {"Female", "Male"} or even as {1, 0}.

There are often multiple ways to represent the same piece of information. ?FAST?, "National University", "FAST NU" and "Nat. Univ. of Computers " can all can be found in the literature as representing the same institution.

Sources might contain invalid data. A point of sale terminal may require that the sales clerk enters a customer's telephone number. If the customer does not wish to give it, clerks may enter 999-999-9999.

Two fields may contain different data but have the same name. There are a couple of ways in which this can happen. "Total Sales" probably means fiscal year sales to one part of an enterprise and calendar year sales to another. The second instance can be much more dangerous. If an application is used by multiple divisions, it is likely that a field that is necessary for one business unit is irrelevant to another and may be left blank by the second unit or, worse, used for otherwise undocumented purposes.

Non primary key problems

- Required fields left blank.
- Data erroneous or incomplete.
- Data contains null values.

Required fields may be left blank. Clerical errors account for most of these, but mobile phones did not come into use until early 90's, so contact numbers recorded before then will not have them. Data may be erroneous or inconsistent. The telephone number may be for Lahore but the

city listed as Karachi.

The data may contain null values. Null values can occur for a wide variety of reasons, the most common of these are:

- Data that is genuinely missing or unknown,
- An attribute does not apply to an entity,
- Data that is pending, or
- Data that is only partially known.

Automatic Data Cleansing…

- Statistical
- Pattern Based
- Clustering
- Association Rules

Some of the data cleansing techniques are listed. Let's discuss each of them in detail.

Statistical: Identifying outlier fields and records using the values of mean, standard deviation, range, etc., based on Chebyshev's theorem, considering the confidence intervals for each field. Outlier values for particular fields are identified based on automatically computed statistics. For each field the average and the standard deviation are utilized and based on Chebyshev's theorem those records that have values in a given field outside a number of standard deviations from the mean are identified. The number of standard deviations to be considered is customizable. Confidence intervals are taken into consideration for each field.

Pattern-based: Identify outlier fields and records that do not conform to existing patterns in the data. Combined techniques (partitioning, classification, and clustering) are used to identify patterns that apply to most records. A pattern is defined by a group of records that have similar characteristics ("behavior") for p% of the fields in the data set, where p is a user-defined value (usually above 90).

Clustering: Identify outlier records using clustering based on Euclidian (or other) distance. Existing clustering algorithms provide little support for identifying outliers. However, in some

cases clustering the entire record space can reveal outliers that are not identified at the field level inspection. The main drawback of this method is computational time. The clustering algorithms have high computational complexity. For large record spaces and large number of records, the run time of the clustering algorithms is prohibitive.

Association rules: Association rules with high confidence and support define a different kind of pattern. As before, records that do not follow these rules are considered outliers. The power of association rules is that they can deal with data of different types. However, Boolean association rules do not provide enough quantitative and qualitative information

Data Duplication Elimination & BSN Method

Learning Goals

- Why data duplicated?
- Problems due to data duplication
- Understand Formal definition & Nomenclature
- Basic Sorted Neighborhood (BSN) Method

The *duplicate elimination* task is typically performed after most other transformation and cleaning steps have taken place, especially after having cleaned single-source errors and conflicting representations. It is performed either on two cleaned sources at a time or on a single already integrated data set. Duplicate elimination requires to first identify (i.e. match) similar records concerning the same real world entity. In the second step, similar records are merged into one record containing all relevant attributes without redundancy. Furthermore, redundant records are removed.

Why data duplicated?

A data warehouse is created from heterogeneous sources, with heterogeneous databases (different schema/representation) of the same entity.

The data coming from outside the organization owning the DWH can have even lower quality data i.e. different representation for same entity, transcription or typographical errors.

A data warehouse is created by merging large databases acquired from different sources with

heterogeneous representations of information. This raises the issue of data quality, the foremost being identification and elimination of duplicates, crucial for accurate statistical analyses. Other than using own historical/transactional data, it is not uncommon for large businesses to acquire scores of databases each month, with a total size of hundreds of millions to over a billion records that need to be added to the warehouse. The fundamental problem is that the data supplied by various sources typically include identifiers or string data that are either different among different datasets or simply erroneous due to a variety of reasons including typographical or transcription errors or purposeful fraudulent activity, such as aliases in the case of names.

Problems due to data duplication

Data duplication can result in costly errors, such as:

- False frequency distributions.
- Incorrect aggregates due to double counting.

Difficulty with catching fabricated identities by credit card companies. Without accurate identification of duplicated information frequency distributions and various other aggregates will produce false or misleading statistics leading to perhaps untrustworthy new knowledge and bad decisions. Thus this has become an increasingly important and complex problem for many organizations that are in the process of establishing a data warehouse or updating the one already in existence.

Credit card companies routinely assess the financial risk of potential new customers who may purposely hide their true identities and thus their history or manufacture new ones.

The sources of corruption of data are many. To name a few, errors due to data entry mistakes, faulty sensor readings or more malicious activities provide scores of erroneous datasets that propagate errors in each successive generation of data.

Data Duplication: Non-Unique PK

- **Duplicate Identification Numbers**
- **Multiple Customer Numbers**

Name	Phone Number	Cust. No.
M. Ismail Siddiqi	*021.666.1244*	*780701*
M. Ismail Siddiqi	*021.666.1244*	*780203*
M. Ismail Siddiqi	*021.666.1244*	*780009*

- **Multiple Employee Numbers**

Bonus Date	Name	Department	Emp. No.
Jan. 2000	*Khan Muhammad*	*213 (MKT)*	*5353536*
Dec. 2001	*Khan Muhammad*	*567 (SLS)*	*4577833*
Mar. 2002	*Khan Muhammad*	*349 (HR)*	*3457642*

Unable to determine customer relationships (CRM)

Unable to analyze employee benefits trends

What if the customers are divided among sales people to make telephone calls? Maybe the three records of the same person go to three telesales people or maybe to the same one, but arranged as per Cust.No. The point is that no telesales person would know that a single customer is going to get multiple calls from the company, resulting in wastage of time and resources of the company, and at the same time annoying the customer.

In the other case, the employee is same, who has been moving among different departments, and possibly during the process got new employee numbers. He has also been getting bonuses regularly, maybe because every time he appears to be an employee who has never got a bonus. This resulting in loss to the company and an inability of the company to do any meaningful employee benefit analysis

Data Duplication: House Holding

- **Group together all records that belong to the same household.**

.....	S. Ahad	440, Munir Road, Lahore
.....	.....	.....
.....	Shiekh Ahad	No. 440, Munir Rd, Lhr
.....	.....	.....
.....	Shiekh Ahed	House # 440, Munir Road, Lahore

Why bother?

In a joint family system, many working people live at the same house, but have different bank accounts and ATM cards. If the bank would like to introduce a new service and wants to get in touch with its customers, it would be much better if a single letter is sent to a single household, instead of sending multiple letters. The problem is difficult because multiple persons living at the same house may have written the same address differently even worse, one person with multiple accounts at the same bank may have written his/her name and address differently.

One month is a typical business cycle in certain direct marketing operations. This means that sources of data need to be identified, acquired, conditioned and then correlated or merged within a small portion of a month in order to prepare mailings and response analyses. With tens of thousands of mails to be sent, reducing the numbers of duplicate mails sent to the same household can result in a significant savings.

Data Duplication: Individualization

- **Identify multiple records in each household which represent the same individual**

.....	M. Ahad	440, Munir Road, Lahore
.....	.....	.....
.....	Maj Ahad	440, Munir Road, Lahore

Identify multiple records in each household which represent the same

Address field is standardized, is this by coincidence? This could be your luck data, but don't count on it i.e. this is very unlikely to be the default case.

Formal definition & Nomenclature

- Problem statement:

1. "Given two databases, identify the potentially matched records Efficiently and Effectively"

- Many names, such as:

1. Record linkage
2. Merge/purge
3. Entity reconciliation
4. List washing and data cleansing.
5. Current market and tools heavily centered towards customer lists.

Within the data warehousing field, data cleansing is applied especially when several databases are merged. Records referring to the same entity are represented in different formats in the different data sets or are represented erroneously. Thus, duplicate records will appear in the merged database. The issue is to identify and eliminate these duplicates.

The problem is known as the *merge/purge problem.* Instances of this problem appearing in literature are called record linkage, semantic integration, instance identification, or object identity problem.

Data cleansing is much more than simply updating a record with good data. Serious data cleansing involves decomposing and reassembling the data. One can break down the cleansing into six steps: elementizing, standardizing, verifying, matching, house holding, and documenting. Although data cleansing can take many forms, the current marketplace and the current technology for data cleansing are heavily focused on customer lists.

Need & Tool Support

1. Logical solution to dirty data is to clean it in some way.
2. Doing it manually is very slow and prone to errors.
3. Tools are required to do it “cost” effectively to achieve reasonable quality.
4. Tools are there, some for specific fields, others for specific cleaning phase.
5. Since application specific, so work very well, but need support from other tools for broad spectrum of cleaning problems.

For existing data sets, the logical solution to the dirty data problem is to attempt to cleanse the data in some way. That is, explore the data set for possible problems and endeavor to correct the errors. Of course, for any real world data set, doing this task "by hand" is completely out of the question given the amount of man hours involved. Some organizations spend millions of dollars per year to detect data errors. A manual process of data cleansing is also laborious, time consuming, and itself prone to errors. The need for useful and powerful tools that automate or greatly assist in the data cleansing process are necessary and may be the only practical and cost effective way to achieve a reasonable quality level in an existing data set.

A large variety of tools is available in the market to support data transformation and data cleaning tasks, in particular for data warehousing. Some tools concentrate on a specific domain, such as cleaning name and address data, or a specific cleaning phase, such as data analysis or duplicate elimination. Due to their restricted domain, specialized tools typically perform very well but must be complemented by other tools to address the broad spectrum of transformation and cleaning problems.

Overview of the Basic Concept

In its simplest form, there is an identifying attribute (or combination) per record for identification.

Records can be from single source or multiple sources sharing same PK or other common unique attributes.

Sorting performed on identifying attributes and neighboring records checked.

What if no common attributes or dirty data?

The degree of similarity measured numerically, different attributes may contribute differently.

In the simplest case, there is an identifying attribute or attribute combination per record that can be used for matching records, e.g., if different sources share the same primary key or if there are other common unique attributes. For example people in income tax department get customer details from the phone company and the Electricity Company and want to identify customers who have high electricity and high telephone bill but do not pay tax accordingly. Would be the common field, NID, they have changed over time, how about addresses, too many ways to write addresses so have to figure out common attributes.

Instance matching between different sources is then achieved by a standard equi-join on the identifying attribute(s), if you are very, very lucky. In the case of a single data set, matches can be determined by sorting on the identifying attribute and checking if neighboring records match. In both cases, efficient implementations can be achieved even for large data sets. Unfortunately, without common key attributes or in the presence of dirty data such straightforward approaches are often too restrictive. For example, consider a rule that states person records are likely to correspond if name and portions of the address match.

The degree of similarity between two records, often measured by a numerical value between 0 and 1, usually depends on application characteristics. For instance, different attributes in a matching rule may contribute different weight to the overall degree of similarity.

Basic Sorted Neighborhood (BSN) Method

Concatenate data into one sequential list of N records

Steps 1: Create Keys
Compute a key for each record in the list by extracting relevant fields or portions of fields

Effectiveness of the this method highly depends on a properly chosen key

Step 2: Sort Data
Sort the records in the data list using the key of step 1

Step 3: Merge
Move a fixed size window through the sequential list of records limiting the comparisons for matching records to those records in the window

If the size of the window is *w* records then every new record entering the window is compared with the previous *w-1* records.

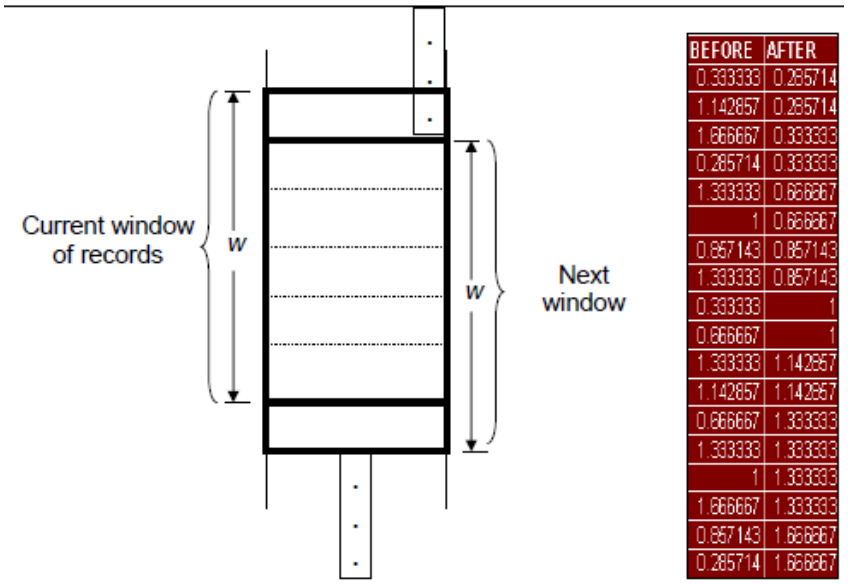
Create Keys: Compute a key for each record in the list by extracting relevant fields or portions of fields. The choice of the key depends upon an error model that may be viewed as knowledge intensive and domain specific for the effectiveness of the sorted neighborhood method. This highly depends on a properly chosen key with the assumption that common but erroneous data will have closely matching keys**.**

Sort Data: Sort the records in the data list using the key of step-1.

Merge: Move a fixed size window through the sequential list of records limiting the comparisons for matching records to those records in the window. If the size of the window is *w* records as shown in Figure 20.1, then every new record entering the window is compared with the previous *w - 1* records to find "matching" records. The first record in the window slides out of the window.

Note that in the absence of a window, the comparison would cover all the sorted records i.e. a pair-wise comparison, resulting in O (n2) comparisons.

BSN Method: Sliding Window



BEFORE	AFTER
0.333333	0.285714
1.142857	0.285714
1.666667	0.333333
0.285714	0.333333
1.333333	0.666667
1	0.666667
0.857143	0.857143
1.333333	0.857143
0.333333	1
0.666667	1
1.333333	1.142857
1.142857	1.142857
0.666667	1.333333
1.333333	1.333333
1	1.333333
1.666667	1.333333
0.857143	1.666667
0.285714	1.666667

BSN Method: Sliding Window shows the outcome of the sliding window i.e. the IDs sorted after completion of the run. The outcome will also depend on the window size, as it may so happen that when the window size is relatively small and the keys are dispersed then some (or most of the keys) may not land in their corresponding neighborhood, thus defeating the purpose of the BSN method. One way of overcoming this problem is working with different window sizes in a multipass approach.

Complexity Analysis of BSN Method

Time Complexity: O(n log n)

O (n) for Key Creation

O (n log n) for Sorting

O (w n) for matching, where $w \leq 2 \leq n$

Constants vary a lot

At least three passes required on the dataset.

For large sets disk I/O is detrimental.

Complexity or rule and window size detrimental. When this procedure is executed serially as a main memory based process the create keys phase is an O(n) operation the sorting phase is O(n

log n) and the merging phase is O(wn) where n is the number of records in the database. Thus, the total time complexity of this method is O (n log n) if w < ⌈log n⌉, O(wn) otherwise. However the constants in the equations differ greatly. It could be relatively expensive to extract relevant key values from a record during the create key phase. Sorting requires a few machine instructions to compare the keys. The merge phase requires the application of a potentially large number of rules to compare two records and thus has the potential for the largest constant factor.

Notice that w is a parameter of the window scanning procedure. The legitimate values of w may range from 2 (whereby only two consecutive elements are compared) to n (whereby each element is compared to all others). The latter case pertains to the full quadratic O (n2) time process at the maximal potential accuracy. The former case (where *w* may be viewed as a small constant relative to *n)* pertains to optimal time performance (only O (n) time) but at minimal accuracy.

Note however that for very large databases the dominant cost is likely disk I/O and hence the number of passes over the data set. In this case at least three passes would be needed one pass for conditioning the data and preparing keys at least a second pass likely more for a high speed sort, and a final pass for window processing and application of the rule program for each record entering the sliding window. Depending upon the complexity of the rule program and window size w the last pass may indeed be the dominant cost.

BSN Method: Selection of Keys

- **Selection of Keys**
  - **Effectiveness highly dependent on the key selected to sort the records middle name vs. last name,**
  - **A key is a sequence of a subset of attributes or sub-strings within the attributes chosen from the record.**
  - **The keys are used for sorting the entire dataset with the intention that matched candidates will appear close to each other.**

First	Middle	Address	NID	Key
Muhammed	Ahmad	440 Munir Road	34535322	AHM440MUN345
Muhammad	Ahmad	440 Munir Road	34535322	AHM440MUN345
Muhammed	Ahmed	440 Munir Road	34535322	AHM440MUN345
Muhammad	Ahmar	440 Munawar Road	34535334	AHM440MUN345

BSN Method: Selection of Keys

The key consists of the concatenation of several ordered fields or attributes in the data. The first three letters of a middle name are concatenated with the first three letters of the first name field followed by the address number field and all of the consonants of the road name. This is followed by the first three digits of the National ID field as shown.

These choices are made since the key designer determined that family names are not very discriminating (Khan, Malik andCheema). The first names are also not very discriminating and are typically misspelled (Muhammed, Muhammad, Mohamed) due to mistakes in vocalized sounds vowels but middle names are typically more uncommon and less prone to being misunderstood and hence less likely to be recorded incorrectly.

The keys are now used for sorting the entire dataset with the intention that all equivalent or matching data will appear close to each other in the final sorted list. Notice in table 20.1, how the first and second records are exact duplicates while the third is likely to be the same person but with a misspelled middle name i.e. Ahm**e**d instead of Ahm**a**d. We would expect that this phonetically based mistake will be caught by a reasonable equational theory. However the fourth record although having the exact same key as the prior three records appears unlikely to

be the same person.

BSN Method: Problem with keys

Since data is dirty, so keys WILL also be dirty, and matching records will not come together. Data becomes dirty due to data entry errors or use of abbreviations. Some real
examples are as follows:
Technology
Tech.
Techno. Tchnlgy

Solution is to use external standard source files to validate the data and resolve any data conflicts.

Since the data is dirty and the keys are extracted directly from the data then the keys for sorting will also be dirty. Therefore the process of sorting the records to bring matching records together will not be as effective. A substantial number of matching records may not be detected in the subsequent window scan.

Data can become dirty due to data entry errors. To speed-up data entry abbreviations are also used (all taken from the telephone directory.), such as:

Technology, Tech. Techno. Tchnlgy

The above is a typical case of a non-PK error of same entity with multiple representations. To ensure the correctness of data in the database, solution is using external source files to validate the data and resolve any data conflicts.

BSN Method: Problem with keys (e.g.)

If contents of fields are not properly ordered, similar records will NOT fall in the same window.

Example: Records 1 and 2 are similar but will occur far apart.

No	Name	Address	Gender
1	N. Jaffri, Syed	No. 420, Street 15, Chaklala 4, Rawalpindi	M
2	S. Noman	420, Scheme 4, Rwp	M
3	Saiam Noor	Flat 5, Afshan Colony, Saidpur Road, Lahore	F

Solution is to TOKENize the fields i.e. break them further. Use the tokens in different fields for sorting to fix the error.

Example: Either using the name or the address field records 1 and 2 will fall close.

No	Name	Address	Gender
1	Syed N Jaffri	420 15 4 Chaklala No Rawalpindi Street	M
2	Syed Noman	420 4 Rwp Scheme	M
3	Saiam Noor	5 Afshan Colony Flat Lahore Road Saidpur	F

We observe that characters in a string can be grouped into meaningful pieces. We can often identify important components or tokens within a Name or Address field by using a set of delimiters such as space and punctuations. Hence we can first tokenize these fields and then sort the tokens within these fields. Records are then sorted on the select key fields; note that this approach is an improvement over the standard BSN method.

BSN Method: Matching Candidates

Merging of records is a complex inferential process.

Example-1: Two persons with names spelled nearly but not identically, have the exact same address. We infer they are same person i.e. Noma Abdullah and
Noman Abdullah.

Example-2: Two persons have same National ID numbers but names and addresses are completely different. We infer same person who changed his name and moved or the records represent different persons and NID is incorrect for one of them. Use of further information such as age, gender etc. can alter the decision.

Example-3: Noma-F and Noman-M we could perhaps infer that Noma and Noman are siblings i.e. brothers and sisters. Noma-30 and Noman-5 i.e. mother and son.
The comparison of records during the merge phase to determine their equivalence is a complexinferential process that considers and requires much more information in the compared records than the keys used for sorting. For example suppose two person names are spelled nearly but not identically, and have the exact same address. We might infer they are the same person. For example Noma Abdullah and Noman Abdullah could be the same person if they have the same address.

On the other hand suppose two records have exactly the same National ID numbers but the names and addresses are completely different. We could either assume the records represent the same person who changed his name and moved or the records represent different persons and the NID number field is incorrect for one of them. Without any further information we may perhaps assume the latter. The more information there is in the records the better inferences can be made. For example, if gender and age information is available in some field of the data (Noma-F, Noman-M) we could perhaps infer that Noma and Noman are siblings.

BSN Method: Equational Theory

- To specify the inferences we need equational Theory.
- Logic is NOT based on string equivalence.
- Logic based on domain equivalence.
- Requires declarative rule language.

To specify the inferences as discussed above we need an equational theory in which the logic is based not on the string equivalence but on the domain equivalence. A natural approach to specifying an equational theory and making it practical too would be the use of a declarative rule language. Rule languages have been effectively used in a wide range of applications requiring inference over large data sets. Much research has been conducted to provide efficient means for their compilation and evaluation and this technology can be exploited here for purposes of data cleansing efficiently.

BSN Method: Rule Language

Rule Language Example

Given two records r1 and r2
IF the family_name of r1 equals the family_name of r2
AND the first names differ slightly
AND the address of r1 equals the address of r2
THEN r1 is equivalent to r2

This is rather self explanatory. The rule merely states that if for two records the last names and the addresses are same, but the first names differ slightly, then both the records are same. This makes sense, as using default values last names can be easily fixed, and sorting the address and then using default values can fix most part of the address too, as the addresses are repeating.

BSN Method: Mismatching Candidates

- Transposition of names
- How do you specify "Differ Slightly"?
- Calculate on the basis of a distance function applied to the family_name fields of two records
- Multiple options for distance functions

Notice that rules do not necessarily need to compare values from the same attribute or same domain. For instance to detect a transposition in a person's name we could write a rule that compares the first name of one record with the last name of the second record and the last name of the first record with the first name of the second record.

BSN Method: Distance Metrics

- Distance Functions

1. Phonetic distance
2. Typewriter distance
3. Edit Distance

- A widely used metric to define string similarity

1. Ed(s1,s2)= minimum # of operations (insertion, deletion, substitution) to change s1 to s2

- Example:

1. s1: Muhammad Ehasn
2. s2: Muhammed Ahasn
3. ed(s1,s2) = 2

The implementation is based upon the computation of a distance function applied to the first name fields of two records and the comparison of its results to a threshold to capture obvious typographical errors that may occur in the data. The selection of a distance function and a proper threshold is also a knowledge intensive activity that demands experimental evaluation. An improperly chosen threshold will lead to either an increase in the number of falsely matched records or to a decrease in the number of matching records that should be merged. A number of alternative distance functions for typographical mistakes are possible, including distances based upon (i) edit distance (ii) phonetic distance and (iii) typewriter distance.

Limitations of BSN Method

- BSN Method Limitations
- No single key is sufficient to catch all matching records.
- Fields that appear first in the key have higher discriminating power than those appearing after them
- If NID number is first attribute in the key 81684854432 and 18684854432 are highly likely to fall in windows that are far apart.
- Two Possible Modifications to BSN Method
- Increase *w*, the size of window
- Multiple passes over the data set

In general no single key will be sufficient to catch all matching records. The attributes or fields that appear first in the key have higher discriminating power than those appearing after them. Hence if the error in a record occurs in the particular field or portion of the field that is the most important part of the key there may be little chance a record will end up close to a matching record after sorting. For instance if an employee has two records in the database one with NID

number 81684854432 and another with NID number 18684854432 (the first two numbers were transposed) and if the NID number is used as the principal field of the key then it is very unlikely both records will fall under the same window i.e. the two records with transposed NID numbers will be far apart in the sorted list and hence they may not be merged. As it was empirically established that the number of matching records missed by one run of the sorted neighborhood method can be large unless the neighborhood grows very large.

Increasing w

Increases the computational complexity.

Quality does not increase dramatically.

Not useful unless the window spans the entire database.
W spanning the entire set of rows is infeasible under strict time and cost constraints.

What would be the computational complexity?
O(w2)

Increasing *w* clearly this increases the computational complexity and as discussed in the next section does not increase dramatically the number of similar records merged in the test cases we ran unless of course the window spans the entire database which we have presumed is infeasible under strict time and cost constraints.

Multi-pass Approach

Several independent runs of the BSN method each time with a different key and a relatively small window.

Each independent run will produce a set of pairs of records which can be merged
(takes care of transposition errors)

Apply the transitive closure property to those pairs of records. If records a and b are found to be similar and at the same time records b and c are also found to be similar the transitive closure step can mark a and c to be similar The results will be a union of all pairs discovered by all

independent runs with no duplicates plus all those pairs that can be inferred by transitivity of equality.

The alternative strategy we implemented is to execute several independent runs of the sorted neighborhood method each time using a different key and a relatively small window. We call this strategy the multipass approach. For instance in one run we use the address as the principal part of the key while in another run we use the last name of the employee as the principal part of the key. Each independent run will produce a set of pairs of records which can be merged. We then apply the transitive closure to those pairs of records.

The results will be a union of all pairs discovered by all independent runs with no duplicates plus all those pairs that can be inferred by transitivity of equality. The reason this approach works for the test cases explored here has much to do with the nature of the errors in the data. Transposing the first two digits of the NID number leads to nonmergeable records as we noted. However in such records the variability or error appearing in another field of the records may indeed not be so large. Therefore although the NID numbers in two records are grossly in error the name fields may not be. Hence first sorting on the name fields as the