

# Data Quality in SQL

by

Joe Celko

copyright 2004

# Data Quality

- Data quality stinks.
  - A major problem: Google it!
  - Sarbanes-Oxley can put you in jail
- It got that way because us database guys let it.
  - We have the tools in SQL to prevent 95-98% of common errors
  - We need to start to do it right and not just fast

# Joe Celko - Articles

- Member of ANSI X3H2 since 1987
- SQL for Smarties - DBMS Magazine
- Celko on SQL - DBP&D
- SQL Puzzle - Boxes & Arrows
- DBMS/Report - Systems Integration
- WATCOM SQL Column - PBDJ
- Celko on Software - COMPUTING(UK)
- Celko - Intelligent Enterprise
- SELECT FROM Austin - DB/M (Netherlands)

# Joe Celko - Books

- JOE CELKO'S SQL FOR SMARTIES - 1995, 1999 Morgan-Kaufmann
- INSTANT SQL - 1995, Wrox Press
- JOE CELKO'S SQL PUZZLES & ANSWERS - 1997, Morgan-Kaufmann
- DATA & DATABASES - 1999, Morgan-Kaufmann
- TREES & HIERARCHIES IN SQL- 2004, Morgan-Kaufmann

# Encoding

- **We encode data to get it into a database**
  - Alphabets
  - Numbers
  - Symbols
- **Various way to formally manipulate the codes**
  - Math for numbers
  - String operators
- **You can put “direct” data into databases these days, but most of it is still encoded**
  - Special tools for text and documents
  - Pictures, music, etc. are hard to search

# Bad Encoding Schemes

- **Does not allow for growth in its domain**
- **Georgia Auto tags**
  - Georgia auto tag type codes started as one digit on a punch card
  - Commemorative tags got popular - every college, veterans group, popular cause wanted one
  - The codes became a mess of special multi-punches on the punch cards that had to be translated in the file system.
- **American Honda**
  - “We will never have more than 10,000 dealerships in the United States”

# Bad Encoding Schemes -2

- **Ambiguous codes**
- **ISBN**
  - International Standard Book Number
  - always 10 places and four parts (language, publisher, book number, check digit)
  - language, publisher and book number are variable length subfields
  - There have been ISBNs that can be parsed two ways
- **A “miscellaneous” code that gets used a lot is a bad sign**

# Bad Encoding Schemes -3

- **Lack of support for exceptions**
  - Unknown values
  - Missing values
  - Non-applicable value
  - Miscellaneous or unclassified
  - Overflows, underflows, division by zero, etc.
  - Errors in one field
  - Errors in more than one field (pregnant male)
  - Computable but not known
- **SPARC committee listed 14 kinds of missing data (Interim Report 75-02-08)**
- **Someone else published 22 kinds of missing data**

# Bad Encoding Schemes -4

- If you think designing encoding schemes is not important, do college math in Roman Numerals for a week
- Try living without alphabetical order for a week
  - A monk invented alphabetical order in the Middle Ages
- Queries and aggregations can be made much easier with a good encoding scheme
- Calculations are more accurate, too.

# Guidelines -1

- **#1 error: Oversized columns**
  - If you give an input clerk or user NVARCHAR(255) you will eventually get a Chinese sutra in that column
- **#2 error: No CHECK () Constraints**
  - Why did anyone think that taking time to add constraints once would be worse than:
    - Cleaning up messy data forever?
    - Praying that 1000 applications got it right
- **#3 error: No DEFAULT clauses**
  - Why would all users always pick the same default value?

## Guidelines -2

- **If a code is exposed to users :**
- **Can they validate it?**
  - Check digits (do they still teach this?)
  - Syntax (can you `grep()` it?)
- **Can they verify it?**
  - Trusted outside source
  - Method for finding the code in the source

# Guidelines -3

- **Use existing ISO or industry specific standard codes**
- **Avoid inventing your own encodings**
  - **do you want to maintain them yourself?**
- **Allow for expansion in the codes**
- **Use explicit exception codes**
- **Keep a translation of codes for the user in the database**
- **Validate and verify**

# Lack of a Data Dictionary -1

- Read ISO-11179 for data element names
  - <genus><species> pattern
  - Tells *what* something is in the data model
- NO affixes about the data type ("str-")
- NO affixes about the storage ("tbl-", "vw-")
- NO affixes about the table ("cust-", ..)
- NO affixes about the local usage ("pk-", "fk-")
- NO reserved words (date, time, group, ..)
- NO vague words (id, date, status, ..)
- NO affix strings (status\_value\_id", ..)

## Lack of a Data Dictionary -2

- The same data element has a dozen names
  - Customer\_id, Cust\_id, Customer\_nbr, ..
- You have redundancy and do not know about it!
- Instead of being in a CREATE DOMAIN statement , a data element will be defined in a dozen places and a dozen slightly different ways .

# Using an Autonumber -1

- Since this "magic, all-purpose, one-size-fits-all" pseudo-identifier exists only as a result of the physical state of a particular piece of hardware at a particular time as read by the current release of a particular database product, how do you verify that an entity has such a number in the reality you are modeling?

## Using an Autonumber -2

- Proprietary and highly non-relational attempt to return to sequential files or even OIDs.
- Either the natural key is not protected or the autonumber is a deadly redundancy
- Portability is a bitch
- If it is the only key, you can submit the same row a thousand times, a million times. Your data integrity is trashed
- Gaps in the autonumber sequence occur
  - Missing check numbers, invoice numbers , etc are a serious auditing problem

# Using an Autonumber -3

```
CREATE TABLE MotorPool
(id IDENTITY (1,1) NOT NULL PRIMARY KEY,
 ssn CHAR(9) NOT NULL REFERENCES Personnel(ssn),
 vin CHAR(17) NOT NULL REFERENCES Vehicle(vin));
```

```
CREATE TABLE Personnel
(id IDENTITY (1,1) NOT NULL PRIMARY KEY,
 ssn CHAR(9) NOT NULL UNIQUE,
..);
```

```
CREATE TABLE Vehicles
(id IDENTITY (1,1) NOT NULL PRIMARY KEY,
 vin CHAR(17) NOT NULL UNIQUE,
..);
```

- Change a row in Personnel and look at the motorpool

```
UPDATE Personnel
SET ssn = '666666666'
WHERE id = 1;
```

# Using an Autonumber -4

- Change a row in Personnel and look at the motorpool

```
UPDATE Personnel
```

```
  SET ssn = '666666666'
```

```
 WHERE id = 1;
```

Or

```
UPDATE Personnel
```

```
  SET ssn = '666666666'
```

```
 WHERE ssn = '999999999'
```

Or

```
BEGIN ATOMIC
```

```
DELETE FROM Personnel WHERE id = 1;
```

```
INSERT INTO Personnel VALUES ('666666666')
```

```
END;
```

# Using an Autonumber -3

```
CREATE TABLE MotorPool
(id IDENTITY (1,1) NOT NULL PRIMARY KEY,
 ssn CHAR(9) NOT NULL REFERENCES Personnel(ssn),
 vin CHAR(17) NOT NULL REFERENCES Vehicle(vin));
```

```
CREATE TABLE Personnel
(id IDENTITY (1,1) NOT NULL PRIMARY KEY,
 ssn CHAR(9) NOT NULL UNIQUE,
..);
```

```
CREATE TABLE Vehicles
(id IDENTITY (1,1) NOT NULL PRIMARY KEY,
 vin CHAR(17) NOT NULL UNIQUE,
..);
```

- Change a row in Personnel and look at the motorpool

```
UPDATE Personnel
SET ssn = '666666666'
WHERE id = 1;
```

# Responsibility -1

- Q: Who is responsible for the bad data?
- A: The Database guys, ***NOT*** application developers!
  
- Mop up the leak, but fix the pipe
  - Application developers mop leaks**
  - Database developers fix pipes**

# Responsibility -1

- Q: Who is responsible for the bad data?
- A: The Database guys, ***NOT*** application developers!
  
- Mop up the leak, but fix the pipe
  - Application developers mop leaks**
  - Database developers fix pipes**

## Responsibility -2

- Example: do a query that pulls the first 25 and the last 25 names in an address column
- You find a lot of name lines with extra blanks and garbage characters, so you (pick one):
  - A) Tell the app developers to catch it in the front end
  - B) Do an UPDATE to clean up the table
  - C) Write a CHECK() to catch it in the back end
  - D) All of the above
- Extra credit: you get an address validation package for the front and back ends!!
  - Lots of them for address data
  - Not so easy for other data elements

## DCL in SQL -1

- DCL= Data Control Language
- This is the third and least respected sub-language in SQL
- It is not really a security system, but more of an access control language
- Encryption is not part of the Standards

## DCL in SQL -2

- There are two kinds of people in the Schema
- Users: they work with the data, but cannot change the structure of the schema. They write DML (Data Manipulation Language).
- Admin: they can change the structure of the schema and control access to the schema objects. They write DDL (Data Declaration Language).
- In actual products, the lines might blur a bit
  - SQL Server allows users to create temporary tables on the fly in their sessions.
- In shops, the lines might blur a bit
  - One employee wears several hats

## DCL in SQL -3

- Access is a four-part relationship
- Grantor = a user or admin who controls the Privileges on a Schema Object
- Privileges = the actions that can be done on a Schema Object
- Schema Object = TABLE, VIEW, DOMAIN, COLLATION, stored procedure, trigger, etc.
- Grantee = a user who is given Privileges on a Schema Object

## DCL in SQL -4

- I am not going to give a lecture on DCL (Hooray!)
- By show of hands, how many people here can write GRANT and REVOKE statements?
- By show of hands, how many shops have a person who can write GRANT and REVOKE statements by hand, without a tool?

# Security vs.. Access Control -1

- Security stinks in Standard SQL
  - and most products
  - and most installations
- If you know the schema, make a copy on a new machine, take the existing database and copy the *file* that contains the target to the new machine.
- You can loop thru values in the columns of tables to which you have REFERENCES privileges until you get a hit
- You and conspirators can set up circular GRANTS

# Security vs. Access Control -2

- Standard SQL does not lie to users
  - If you are Perry White, then you are told Clark Kent is “A mild mannered reporter for a great Metropolitan Newspaper.”
  - If you are Lois Lane, then you are told Clark Kent is “Superman, a strange visitor from another planet.”
- Standard SQL does not encrypt data
  - A vendor option or third party product on most products now
- Standard SQL does not challenge users log-ins, change passwords, etc.
  - A vendor option or third party product on most products now

# Reasonableness -1

- It is easy to spot invalid data
  - There is no such date as '2003-02-31' and the system will catch the error for you
  - We do not have a code 1999 in the Dewey Decimal Classification
    - CHECK() constraints will catch the error**
    - If you wrote a CHECK() constraint, that is!**
- It is easy to spot absurdly impossible data
  - There are no pregnant males in your hospital
  - Easy for a human being to see, but often not put into constraints in the schema

## Reasonableness -2

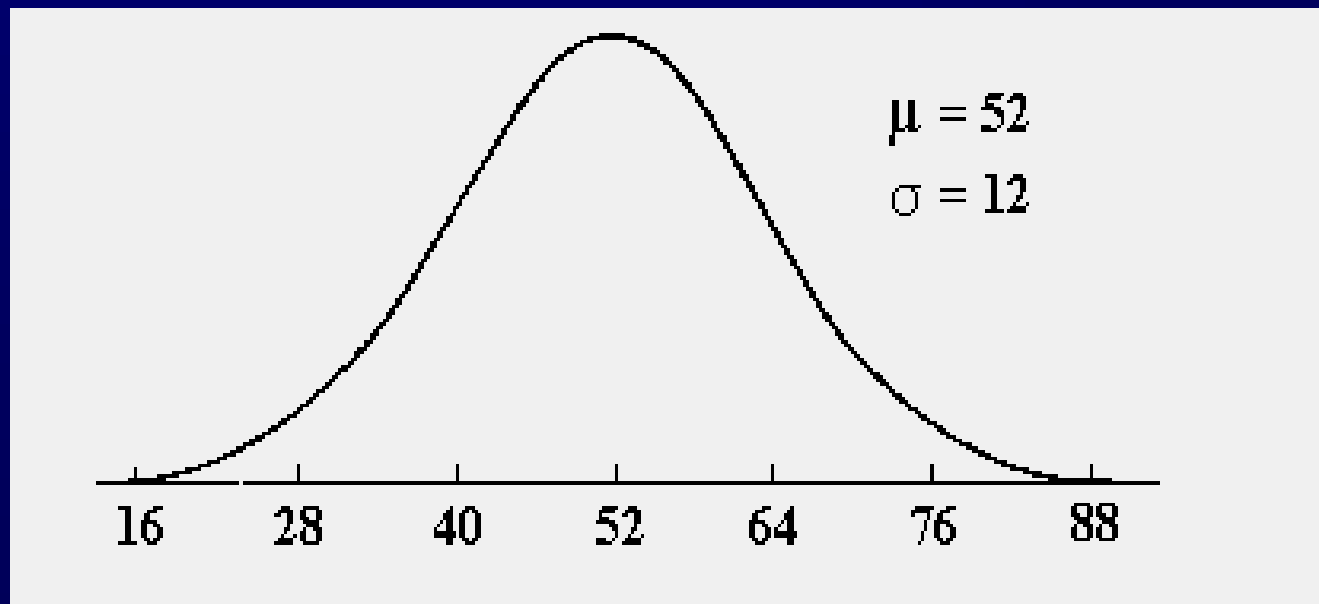
- Possible, but unlikely, data is much harder to spot
- Perhaps this customer really *does* want to order 5000 bananas
  - If the customer is Diary Queen, then it is reasonable
  - If the customer is Joe Celko, then it is unreasonable
- We need to have some ideas as to what the data should look like to know if it is reasonable or not
- The three most common statistical distributions in a database are

# Reasonableness -3

- The Normal (Gaussian) distribution looks like a bell
- The shape is defined by an average ( $\mu$ ) and a standard deviation ( $\delta$ ) on either side of the average
- The curve can be flat or tall, skewed to one side or the other
- In a normal distribution,
  - 68% of the values fall within  $\pm 1$  standard deviation
  - 95% of the values fall within  $\pm 2$  standard deviations
  - 99.7% of the values fall within  $\pm 3$  standard

# Reasonableness -3

- If you to play with an interactive program to see the shape of the normal distribution go to :
  - <http://www-stat.stanford.edu/~naras/jsm/NormalDensity/NormalDensity.html>
- if  $m = 52$  and  $d = 12$ , then the middle value would be labeled with 52, points to the right would have the values of 64 ( $52 + 12$ ), 76, and 88, and points to the left would have the values 40, 28, and 16.



# Zipfian Distribution

- A distribution of probabilities of occurrence that follows Zipf's law (George Kingsley Zipf).
  - Also known as Yule distribution.
- Originally came from the study of words
  - Common words are both short and very common
  - In the English language words like "and," "the," "to," and "of" occur often, while words like "undeniable" are rare.
  - $P(n) = 1/(a*n)$  is the frequency of occurrence of the n-th ranked item and a is close to 1
- This also tells you how to arrange data for the best search time

Look for a horse and not a zebra

# Correlations -1

- A correlation says that when X occurs, then we expect to also see Y a certain percentage of the time
- A correlation coefficient is a number between -1 and 1 which measures the degree to which two variables are *linearly* related.
  - A correlation coefficient of 1 means a perfect positive linear relationship
  - A correlation coefficient of -1 means a perfect negative linear relationship
  - A correlation coefficient of 0 means that there is no linear relationship between the variables
- In the real world, it is hard to find correlations

## Correlations -2

- There are other, non-linear, relationships
  - You can apply a formula to convert them into linear relationships to make the math easier
- Correlation is not causality
  - X might cause Y
  - Y might cause X
  - X and Y might cause each other (feedback loop)
  - X and Y might be caused by Z, and we don't know about Z
  - The whole thing might be dumb, blind luck
- Picking the right correlation statistic to use depends on the nature of the data
  - We are not going into details here.

# Scrubbing Data

- First look for data that is wrong
  - You know you need to fix this stuff
- Next look for data that is out of its expected range
  - It might still be correct!
  - Data within the expected range might be wrong
- Statistical software is easy to find
  - It is also hard to use correctly
  - Get a professional to help you
- Do not just clean up the bad data
  - Find out how it got to be bad
  - Stop the process that is creating or allowing it

# Questions & Answers

?