Anonymizing Data - The Big Picture

Lauri Pietarinen
Relational Consulting

Session Code: F4
Agenda

- General Concepts
- Why extract data from production and what it takes
- Case Handelsbanken
- Creating a framework for companywide anonymizing

Key Bullet Points
- Objective 1: Learn about technical challenges of anonymising production data
- Objective 2: Learn about organisational challenges of anonymising production data
- Objective 3: Understand the parties and skills involved in extracting and anonymising production data for testing purposes
- Objective 4: Understand the role that tools play and don't play
Why Extract from production?

- Production data has “real” properties
- Production data is “integrated” and consistent over all the systems (we hope!)
- Problem solving: extract problematic case into test and fix problem
- In new systems, you own the data – in old legacy systems, the data owns you!
- Extracting means taking only a part of the data → your test database will be small

The greatest benefit of production data is that is consistent over all the systems (or at least it should be!). It is also the data on which the programs being developed must eventually run on.
Why **Not** Extract from production?

- Extracting is hard and error prone
- Need to anonymize (and that is hard!)
  - Can it be done well enough?
- Not all corner cases found in data
- Data for new systems / new functionality not found in production

There are alternatives, of course:
- Scripts
- Data generators / home built programs etc…
- Generating data via screens by hand or using robots

In practice a combination of several methods would often be used.
Basic Requirements for Anonymizing Data

1. Data cannot be traced to it’s source
2. Data has to technically work in the target environment
3. Data has to make sense to user of test data

The most important requirement is of course that the data is properly anonymized and cannot be traced to it’s source. It is not a completely black and white situation, since a knowledgeable user of the anonymized data might be able to figure out the true identity. It is always a question of “good enough”.

Test data is naturally useless, if it does not work with the intended applications and test cases. For instance keys might have been anonymized differently in different systems so that the integrity of the data is broken (e.g. customer is not found in loan-application). It might be that a names or addresses must be the same in two different systems for the application to work, but this is very application dependent. Typically this information is hard to get from the users/owners of the system before the fact. For this reason anonymizing is a highly iterative process: implement→test→fix etc…

The softer requirement is that the data is “user friendly”. For instance, names and addresses are saved in multiple locations even within the same application. Technically the application might work even though the names differ, but the user/tester will be confused. Also, it is nice that names and addresses look “real”.

Biggest Challenges

- Find out what and how to anonymize
  - From system owner
- Embedded and varied data in columns
  - Legacy systems
- Integration over varied systems
- Implement in a way that leaves the applications and users happy

Company policy and system owners should be the determining factor on how and what to anonymize. It might not always be easy to get answers.

Systems will have been built at different times using the database design principles of the moment. Each system is bound to have its own idiosyncrasies.
Ways to anonymize

• Constant text
• Constant text with random number
• Number series
  ▫ e.g. account number, customer number
• Random names attached to customer number
• Partly random
  ▫ E.g. person number (includes date of birth which must remain the same)
Case: Handelsbanken

• Founded in 1871
• 11,361 employees
• Operations in 24 countries
• 461 branches in Sweden, 149 in other Nordic countries, 15 in Netherlands, 147 in Great Britain and 19 in the rest of the world
• Four strategic platforms:
  • z/OS (DB2 V10, IMS/TM)
  • Windows
  • Unix
  • iSeries
Case: Handelsbanken

- Many systems sharing few testing environments on z/OS
- Database schemas and programs out of sync
- Testing environment simulates year 2027
  - Lots of problems with integration testing
- Test Data in bad shape
  - Ad hoc, inconsistent, broken
- It’s the integrated tests that cause the most problems!
  - And applications are more and more integrated and cannot be tested in isolation
CASE: Handelsbanken

• We want to extract intact parts of production data, **anonymise**, and insert into test-environment

• We want **constant anonymizing** (so we get the same anonymized values each time we extract, i.e. names etc.. don’t change

• We want to take larger amounts every month (Maintrack) and smaller amounts “on demand” (Fasttrack)

• We want to drive extract by customer number but also expand selection so that we get a sensible subset

• We want to control which systems are extracted
The EFF-project (Effectivare Test) started in 2010. The aim is to offer better testing environments for development and maintenance.

Currently it consists of the z/OS-platform where the largest parts of Handelsbankens systems reside, but will be further onto other platforms in coming years.

EFF is currently the largest in-house IT-infrastructure-project at SHB and employs about 10 people full time and additionally has lately involved approx 15% of the IT personnel of Handelsbanken.

The EFFx-projects were divided into two “systems” or “applications”, one for the environment (TEMI) and one for the data (TEDA).

The purpose of TEDA is to
- provide means to extract intact sets of production data, unidentify it and insert it into test
- handle data in the test environment
- extracting comes in two “flavors”:
  - MAINTRACK is for large sets of data and is executed less frequently (every 2. month)
  - FASTTRACK is for ad hoc extracts, used for example for problem solving and can be run several times a day

The TEDA-application is written in REXX, it uses about 30 DB2-tables and uses IBM Optim/TestData with Privacy Option for the “heavy lifting”

In this presentation we will concentrate on the TEDA-side and especially on unidentifying data.
Partial list of systems involved in TEDA.

Systems vary largely in number of tables, volume and complexity and age, the oldest systems being from the 80’s.

For instance SLAN (Loan-system) consists of about 450 tables and STÖF consists of only 3.
Implemented So Far - Some Statistics

- 35 (out of 100) systems
  - These are all on z/OS, IBM “Mainframe” DB2
- 851 Tables
- 2076 columns anonymized, of which
  - 663 with a constant text or number
  - 1413 using the lookup table
- 73 different number series employed
There are several distinct roles within the TEDA-project:

- The analyst meets with each system representative to discuss how their system is to be extracted and anonymized. This information is saved in a purpose built Access Database (see next slide). If the system is complex, several meetings are needed. 
- The implementator will configure the TEDA-system (updating rows in the database), create Optim objects (Table maps, Access definitions and Column maps) and modify certain parts of the application 
- TEDA developer will enable new functionality within the framework 

In practice, many systems had to be partly “reverser engineered”
For each system, the system representatives were met with to discuss the anonymizing of tables and columns. At first, the information was stored in Excel-files but it soon became apparent, that this was not an adequate way to keep track.

An Access database and application was created where data was stored in a more structured format.

In the above view we can see the system BINT, with a list of tables and further a list of columns (fält). For each column is specified
- How the column is to be anonymized
- the key type of the column
- implmentation status
- etc…
The TEDA (TEstDAta) application implements the extracting, inserting and anonymizing of production data.

It consists of 30 DB2-tables that contain
- configuration information
- Key lists, lookup tables
- Test data order information

-IBM/Optim TDM (former Relational Tools) is used for the “heavy lifting”

The purpose of the application is to “embed” all the rules governing extracting and anonymizing so that the process itself can be executed by just “pressing the proverbial button”
The input into TEDA consists of a list of systems and a list of customer numbers.

The following stages are executed:

1) Find secondary keys (e.g. accounts, other customers)
2) Populate lookup tables for unidentifying
3) Extract data (using Optim)
4) Anonymize data (using Optim)
5) Insert extracted and anonymized rows into test-database
Example: Account Number anonymizing

- We were given a set of account numbers (currently 45,000) to use as the anonymized values
  - We can’t generate random ones and can’t use values currently in production
  - 45,000 is not very much, we have run into problems
- This means that we have to save the values in a “key store” and use them when needed. For this we have a DB2-table
- When an account number needs to be anonymized, an entry is written into a “lookup-table” used by Optim/TestData
We store all 45000 given “test”-account numbers in a KEYSTORE-table.

When an account needs to be anonymized the next unused value is selected from the keystore to populate the lookup-table with the value pairs.

Since the account number is stored in several different formats in different applications, rows are created in the LOOKUP_FORMAT –table to represent each format.

The LOOKUP_FORMAT –table is used by Optim/Data Privacy to convert the values extracted from production.

The lookup-values will be saved so the same account number will always be anonymized in the same way.

If the values need to be reused, resetting the accounts is easy: set value as “unused” by updating the TS_USED column and delete all entries from LOOKUP and LOOKUP_FORMAT.
Each key type is described in the KEYTYPE-table. There is an anonymizing model associated with each key type. In the case of “keystore” an unused value is taken from the KEYSTORE-table.

For each key type there are 1..n different formats. Some formats are trivial, such as the ones with padded zeroes. Others are more complex and are implemented in REXX.

There are currently 76 key types and 171 formats in use.
Optim- Column Map definition for Anonymizing a table

```
CREATE CM D26A0100
SRC BINT.D26A0100
DEST BINT.D26A0100 |
  LOOKUP(TEDA.LOOKUP_FORMAT(KEY_ORIG,KEY_ANON)) = REFERENSNR_KUND,
  LOOKUP(TEDA.LOOKUP_FORMAT(KEY_ORIG,KEY_ANON)) = DEB_KTO,
  EXIT TEDAE001 'TXT14' = MEDDELANDETExt,
  EXIT TEDAE001 'TXT14' = FRI_TEXT,
  LOOKUP(TEDA.LOOKUP_FORMAT(KEY_ORIG,KEY_ANON)) = GOD_REFNR_01,
  EXIT TEDAE001 'TXT35' = BORTTAG_REFNR,
  LOOKUP(TEDA.LOOKUP_FORMAT(KEY_ORIG,KEY_ANON)) = REFNR_ATERSKAP,
  LOOKUP(TEDA.LOOKUP_FORMAT(KEY_ORIG,KEY_ANON)) = FULLM_GIVARE_REFNR
);
```

This is the Optim/TestData -definition for unidentifying the columns of BINT.D26A0100 (see Access Database slide on page x).

Note: the definition has been simplified for clarity!

The LOOKUP_FORMAT –table is used to anonymize the values.

An Optim Exit is used to read entries from the CONSTANT_TEXT -table
## Constant value table

<table>
<thead>
<tr>
<th>TXTID</th>
<th>TXTKONSTANT</th>
<th>TXTBESKR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXT13</td>
<td>Forts. Namn</td>
<td></td>
</tr>
<tr>
<td>TXT14</td>
<td>Fritext</td>
<td>Fritext</td>
</tr>
<tr>
<td>TXT15</td>
<td>XXXXXXXXXXXX</td>
<td>(22 X)</td>
</tr>
<tr>
<td>TXT16</td>
<td>Kundvald kontoinfo</td>
<td></td>
</tr>
<tr>
<td>TXT17</td>
<td>Kalle Kundansvarig Test</td>
<td></td>
</tr>
<tr>
<td>TXT18</td>
<td>Karin Kontaktperson Test</td>
<td></td>
</tr>
<tr>
<td>TXT19</td>
<td>&lt;Utblankat&gt;</td>
<td></td>
</tr>
<tr>
<td>TXT20</td>
<td>Testgatan</td>
<td></td>
</tr>
<tr>
<td>TXT21</td>
<td>Ville Värderingsman Test</td>
<td></td>
</tr>
<tr>
<td>TXT22</td>
<td>Testgatan</td>
<td>Telefonnummer</td>
</tr>
<tr>
<td>TXT23</td>
<td>Testgatan</td>
<td>Bank</td>
</tr>
<tr>
<td>TXT24</td>
<td>TOT8700</td>
<td></td>
</tr>
<tr>
<td>TXT25</td>
<td>YYAA01</td>
<td></td>
</tr>
<tr>
<td>TXT26</td>
<td>999999</td>
<td></td>
</tr>
<tr>
<td>TXT27</td>
<td>00399999994</td>
<td>Stopp-person custnr</td>
</tr>
</tbody>
</table>

Constant values are saved in a separate table and are referred by their TXTID.
Some Anonymizing Examples

- BOFABE3X → AAAABE3X (Swift)
- BOFABE30 → AAABBE30 (Swift)
- 195703129907 → 195703121607 (Person number)
- 02004942334 → 02565234334 (loan number)

In addition to the simple cases and formats there are numerous exceptions that we have to handle.
We will show the statistics of a sample extract for INLÅ, the main banking system.

In order for INLÅ to run its batches it needs consistent data from 22 other systems.
## Statistics of sample INLÅ extract 20.5.2013

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of customers ordered</strong></td>
<td>579</td>
</tr>
<tr>
<td>Dependent customers</td>
<td>1827</td>
</tr>
<tr>
<td>Dependent savings accounts</td>
<td>2190</td>
</tr>
<tr>
<td>Tables from INLÅ</td>
<td>71</td>
</tr>
<tr>
<td>Rows from INLÅ</td>
<td>1,3M</td>
</tr>
<tr>
<td>Number of systems</td>
<td>22</td>
</tr>
<tr>
<td>Total number of tables</td>
<td>296</td>
</tr>
<tr>
<td>Total number of rows</td>
<td>6,6M</td>
</tr>
<tr>
<td>Largest table</td>
<td>1,2M</td>
</tr>
<tr>
<td><strong>Number of columns unidentified</strong></td>
<td>637</td>
</tr>
<tr>
<td>CPU</td>
<td>1082 sec</td>
</tr>
<tr>
<td>Total runtime</td>
<td>71 min</td>
</tr>
</tbody>
</table>
Cross Platform integrity of test data is currently a serious problem. We plan to tackle it by creating a master lookup table, that will be read or copied by other platforms, so that anonymizing will be consistent over platforms.
Summary

- Strong backing by management was required
  - License to Fail!
- Tools are important but not sufficient for success
- Lot’s of application knowledge required
- Iterative development paramount
- In the end the main question is: does it work for your tests?

Projects like these are very risky because you don’t know what you will find as you progress. In the case of Handelsbanken I was very impressed by the patience and understanding of this fact by management. You could say that you need a ”License to Fail”, so that the project can learn from the experiences and on iteration do a better job.

It can’t be stressed to much that test data has to be tested! This means tight iteration cycles and fixing problems as they appear.
Lauri Pietarinen
Relational Consulting
lauri.pietarinen@relational-consulting.com

Session F4
Anonymizing Data - The Big Picture