TEST DATA MASKING
FOR Db2 on z/OS

Kai Stroh, UBS Hainer

kai.stroh@ubs-hainer.com

Everybody needs to do masking, but getting it done is hard.

The challenge is to change the data in a way so that:

- You cannot derive the original data from the masked data
- The masked data looks plausible and will pass validity checks

  Credit card numbers, IBANs, SINs have validity checks

  SSNs cannot have all-zero in any group

  Street name, postal code / ZIP code and city depend on each other
- The masked data does not violate database constraints

  Unique constraints / Referential integrity constraints

MASKING IN PRACTICE

- Don't just set everything to NULL or to XXX

  Cannot be used if unique constraints or RI constraints exist

  Does not give application anything plausible to work with

- Don't just shift each digit and letter

  Usually does not violate unique or foreign key contraints, but:

  Insecure, original value can be reconstructed easily

  Can result in invalid data (E.g. credit card number with invalid check digit)

- Don’t replace data with random values

  Masking needs to be repeatable

  Random values will change every time you mask

  Random values typically violate constraints

MASKING IN PRACTICE

- Hash-based masking produces good results

  Derive your masked values from your source values

  Any conceivable input value can be processed

  Similar input values result in totally different hash values

  Can be designed to be pratically non-invertible

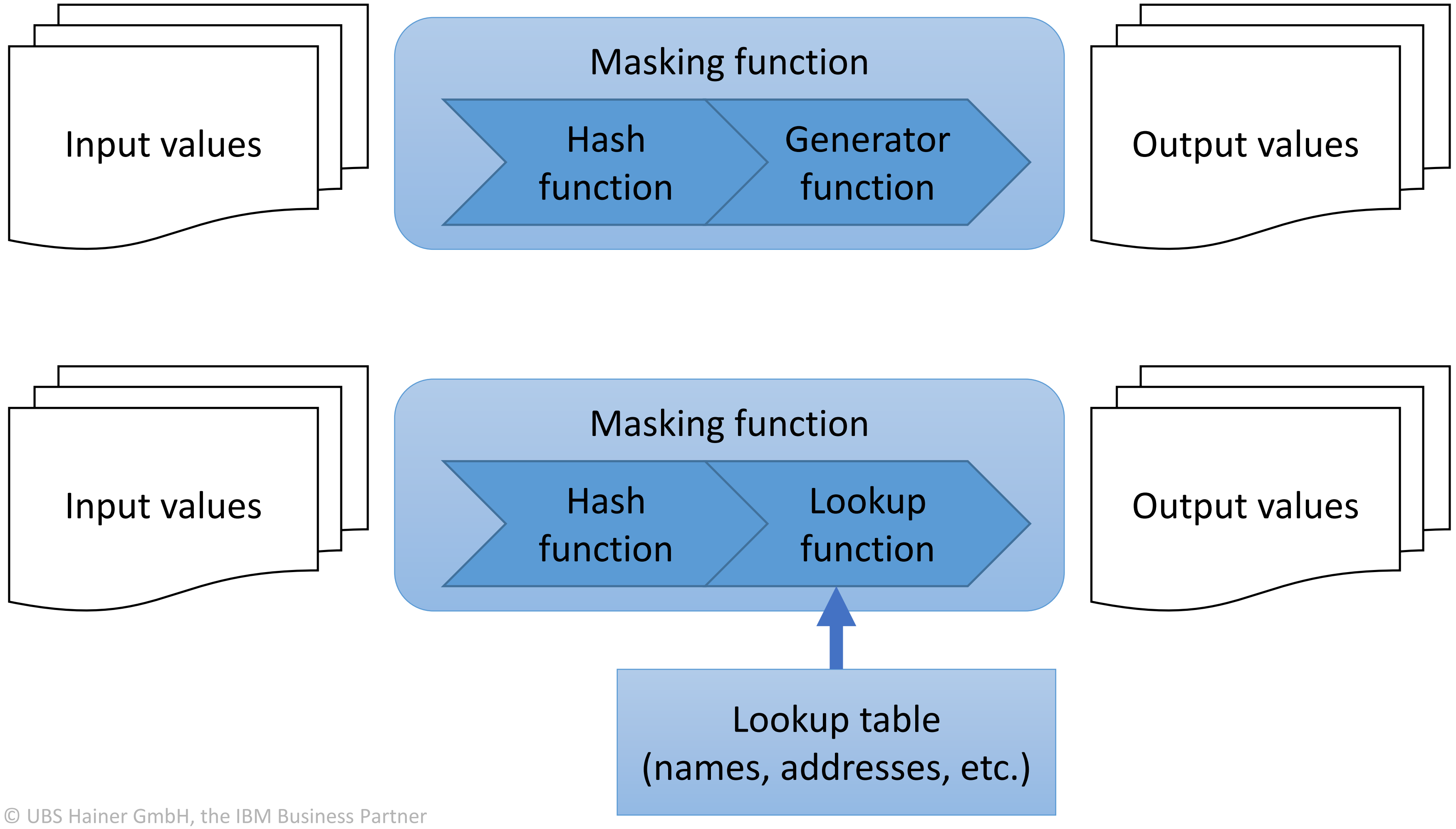
  Can use hash values as lookup table index

- Alternatively: Mapping tables

  Need to be refreshed periodically

Hash-based Masking

HASH-BASED MASKING



HASH-BASED MASKING

- Generator functions:

  When the target value can be calculated based on the hash alone

  Plain numbers

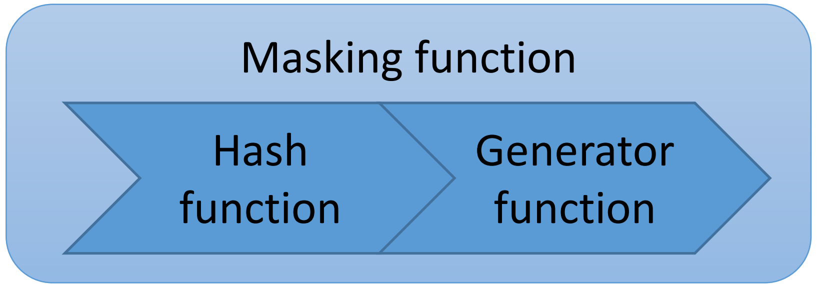
  SSNs / SINs

  License plate numbers

  Data / Time values

  Credit card numbers

  UUIDs



HASH-BASED MASKING

- Lookup functions:

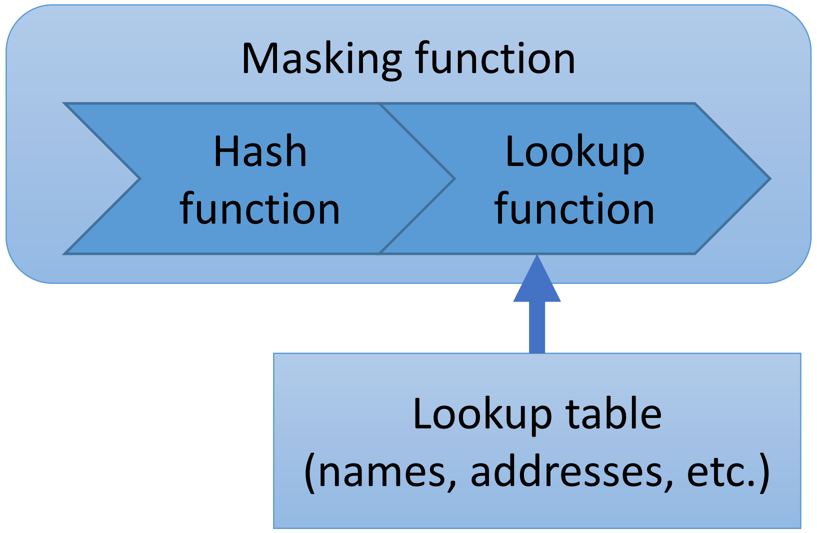
  When the target value needs to be from a list of valid values

  First name, last name

  Address

  Banking information

  Combination of the above



Masking primary and unique keys

- Hash functions are, by definition, not collision free
- This can lead to duplicate values in columns declared as unique
- BCV5 uses a fast hashing algorithm for INTEGER values that is collision free between 0 and 2,147,483,647 (= $2^{31} – 1$)

What to consider when masking data?

WHAT TO CONSIDER

ID	FIRST_NAME	LAST_NAME	DOB	SSN
23847	Annie	Miller	4/17/1977	123-45-6789
23848	George	Miller	10/21/1982	234-56-7890
23849	Melissa	Miller	6/01/1964	345-67-8901
23850	Frederick	Miller	6/28/1967	456-78-9012
23851	Karen	Miller	7/04/1983	567-89-0123
23852	Robert	Wilson	1/30/1990	678-90-1234
23853	Stephanie	Wilson	12/22/1971	789-01-2345
23854	Paul	Wilson	9/14/1985	890-12-3456
23855	Elisabeth	Wilson	2/05/1980	901-23-4567

ID	FIRST_NAME	LAST_NAME	DOB	SSN
23847	Annie	Black	4/17/1977	123-45-6789
23848	George	Black	10/21/1982	234-56-7890
23849	Melissa	Black	6/01/1964	345-67-8901
23850	Frederick	Black	6/28/1967	456-78-9012
23851	Karen	Black	7/04/1983	567-89-0123
23852	Robert	Cooper	1/30/1990	678-90-1234
23853	Stephanie	Cooper	12/22/1971	789-01-2345
23854	Paul	Cooper	9/14/1985	890-12-3456
23855	Elisabeth	Cooper	2/05/1980	901-23-4567

- Use the current field value or a different column (e.g. ID) as seed?
- Need identical masked addresses if the source address are the same?

WHAT TO CONSIDER

ID	FIRST_NAME	LAST_NAME	DOB	SSN
23847	Annie	Miller	4/17/1977	123-45-6789
23848	George	Miller	10/21/1982	234-56-7890
23849	Melissa	Miller	6/01/1964	345-67-8901
23850	Frederick	Miller	6/28/1967	456-78-9012
23851	Karen	Miller	7/04/1983	567-89-0123
23852	Robert	Wilson	1/30/1990	678-90-1234
23853	Stephanie	Wilson	12/22/1971	789-01-2345
23854	Paul	Wilson	9/14/1985	890-12-3456
23855	Elisabeth	Wilson	2/05/1980	901-23-4567

ID	FIRST_NAME	LAST_NAME	DOB	SSN
23847	Annie	Tyson	4/17/1977	123-45-6789
23848	George	Vogel	10/21/1982	234-56-7890
23849	Melissa	Gonzales	6/01/1964	345-67-8901
23850	Frederick	Adamson	6/28/1967	456-78-9012
23851	Karen	Willis	7/04/1983	567-89-0123
23852	Robert	Sterling	1/30/1990	678-90-1234
23853	Stephanie	White	12/22/1971	789-01-2345
23854	Paul	Garrison	9/14/1985	890-12-3456
23855	Elisabeth	Bergman	2/05/1980	901-23-4567

- Use the current field value or a different column (e.g. ID) as seed?
- Need identical masked addresses if the source address are the same?

WHAT TO CONSIDER: GOTCHAS

In a perfect world, everything would be at least 3NF

- There's only one table that contains names
- There's only one table that contains addresses
- There are never any typos
- Phone numbers are always stored as +12223334444

  Never as 222(333)4444 or 222-333-4444 (after 6pm call 555-6666)
- Everything is linked together through single-column primary keys
- No reduncancy

But that's not how real databases look like!

WHAT TO CONSIDER: GOTCHAS

FIRST_NAME	LAST_NAME	DOB	SSN
ANNIE	MILLER	4171977	123-45-6789
GEORGE	WILSON	10211982	234-56-7890
MELISSA	JONSON	6011964	012-34-5678
FREDERICK	BROWN	6281967	001-23-4567

FIRST_NAME	LAST_NAME	DOB	SSN
ANNIE B.	MILLER	4/17/1977	123456789
GEORGE F.	WILSON	10/21/1982	234567890
MELISSA K.	JOHNSON	6/1/1964	12345678
FREDERICK I.	BROWN	6/28/1967	1234567

FIRST_NAME	LAST_NAME	DOB	SSN
Annie	Miller	4/17/1977	123-45-6789
George	Wilson	10/21/1982	234-56-7890
Melissa	Johnson	6/1/1964	012-34-5678
Frederick	Brown	6/28/1967	001-23-4567

NAME	DOB	SSN
MILLER, ANNIE	4/17/1977	123-45-6789
WILSON, GEORGE	10/21/1982	234-56-7890
JOHNSON, MELISSA	6/1/1964	012-34-5678
BROWN, FREDERICK	6/28/1967	001-23-4567

If we want to hash our actual source names, source addresses, etc., we need to tidy it up

- Technically

Remove leading and trailing blanks

Cast numeric data to actual numeric data type if stored in text column

If processing outside of Db2: Convert to common code page

WHAT TO CONSIDER

DATA STANDARDIZATION

If we want to hash our actual source names, source addresses, etc., we need to tidy it up

- Functionally

Truncate to shortest representation that actually exists in the database

String normalization: Remove diacritics, spaces and punctuation

Can you guarantee that all of these will be identically after masking?

Does your application care?

```
St. Louis, Missouri
St. Louis (MO)
St. Louis, MO
St. Louis / MO
St Louis, MO
St Louis   MO
St Louis MO
StLouis MO
St. Louis
Saint Louis
St. Luis
```

Getting consistent masking

WHAT TO CONSIDER

DATA STANDARDIZATION | CONSISTENT DATA

- Sometimes valid values for a column depend on other columns

  Most prominent example: address

- Street, city, zip code and state are dependent on each other

ID	Street	City	State	Zip	Country
…					…
3750982	1100 Congress Ave	Austin	TX	78701	US
…	…	…	…	…	…

WHAT TO CONSIDER

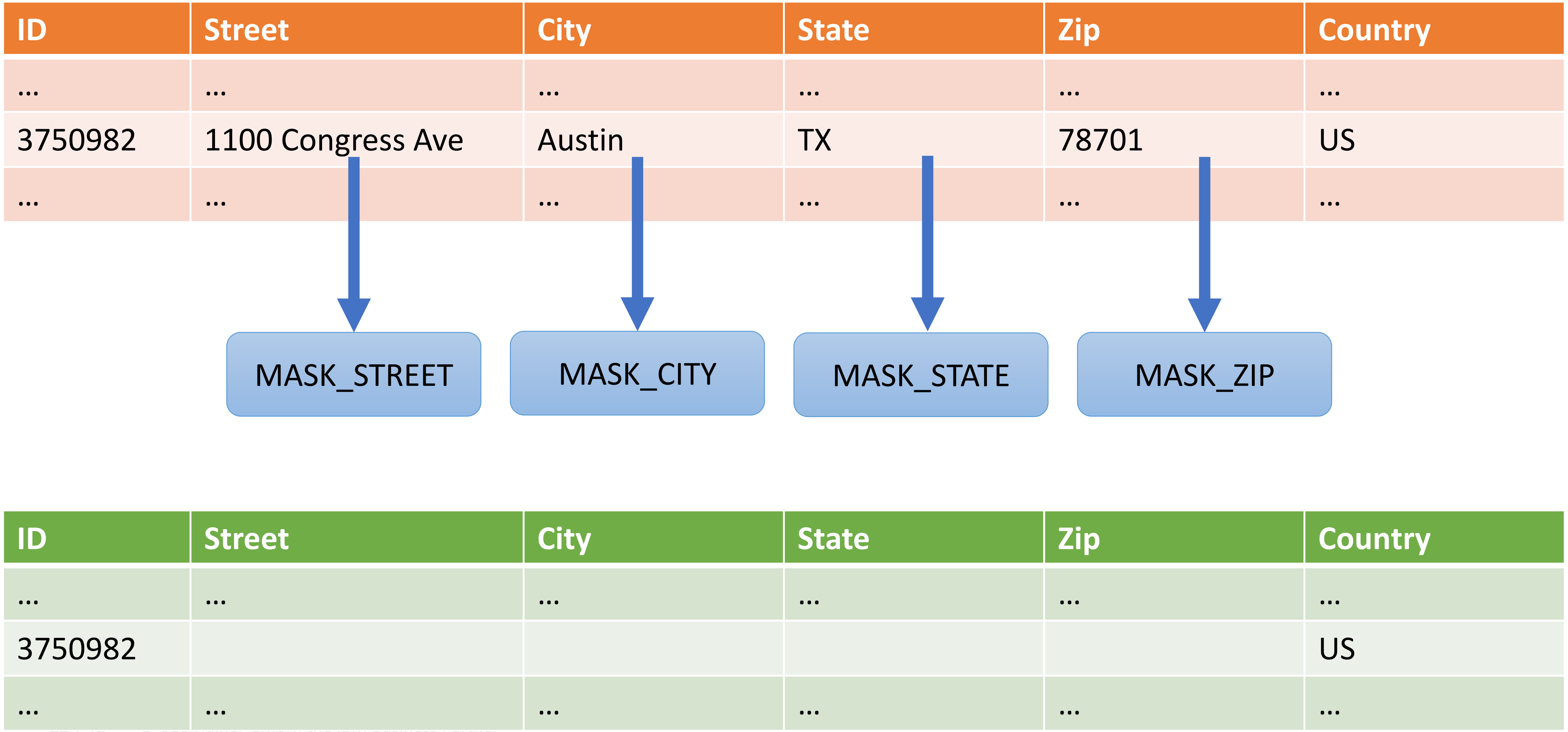
DATA STANDARDIZATION | CONSISTENT DATA

- Address masking uses lookup tables

ID	Street	City	State	Zip
1	500 N. Church St.	Palestine	TX	75081
2	215 East Lufkin Avenue	Lufkin	TX	75902
3	2840 TX-35 BUS	Rockport	TX	78382
4	100 S Center St.	Archer City	TX	76351
5	100 Trice Street	Claude	TX	79019
6	1 Courthouse Circle Dr.	Jourdanton	TX	78026
7	1 East Main Street	Bellville	TX	77418
8	300 S 1st St.	Muleshoe	TX	79347
…	…	…	…	…

WHAT TO CONSIDER

DATA STANDARDIZATION | CONSISTENT DATA



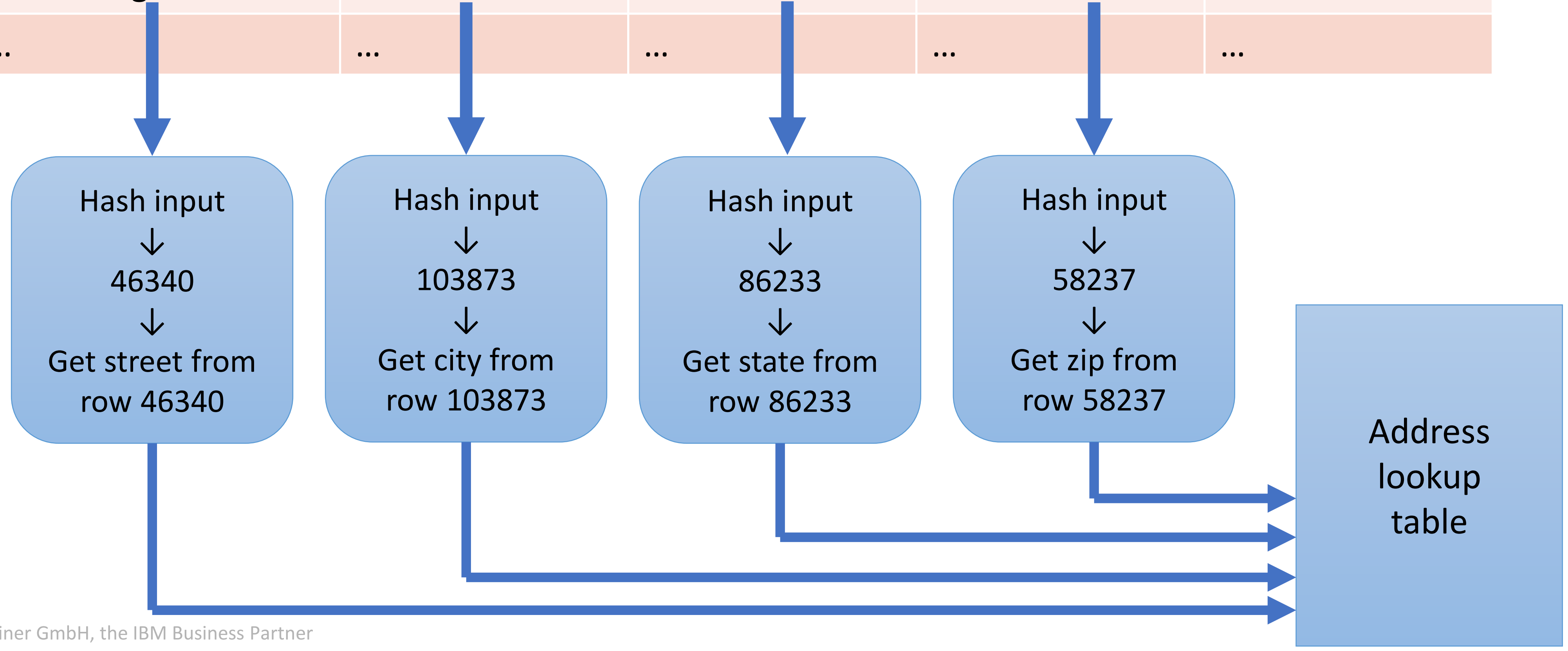
ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982	1100 Congress Ave	Austin	TX	78701	US
…	…	…	…	…	…

ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982					US
…	…	…	…	…	…

WHAT TO CONSIDER

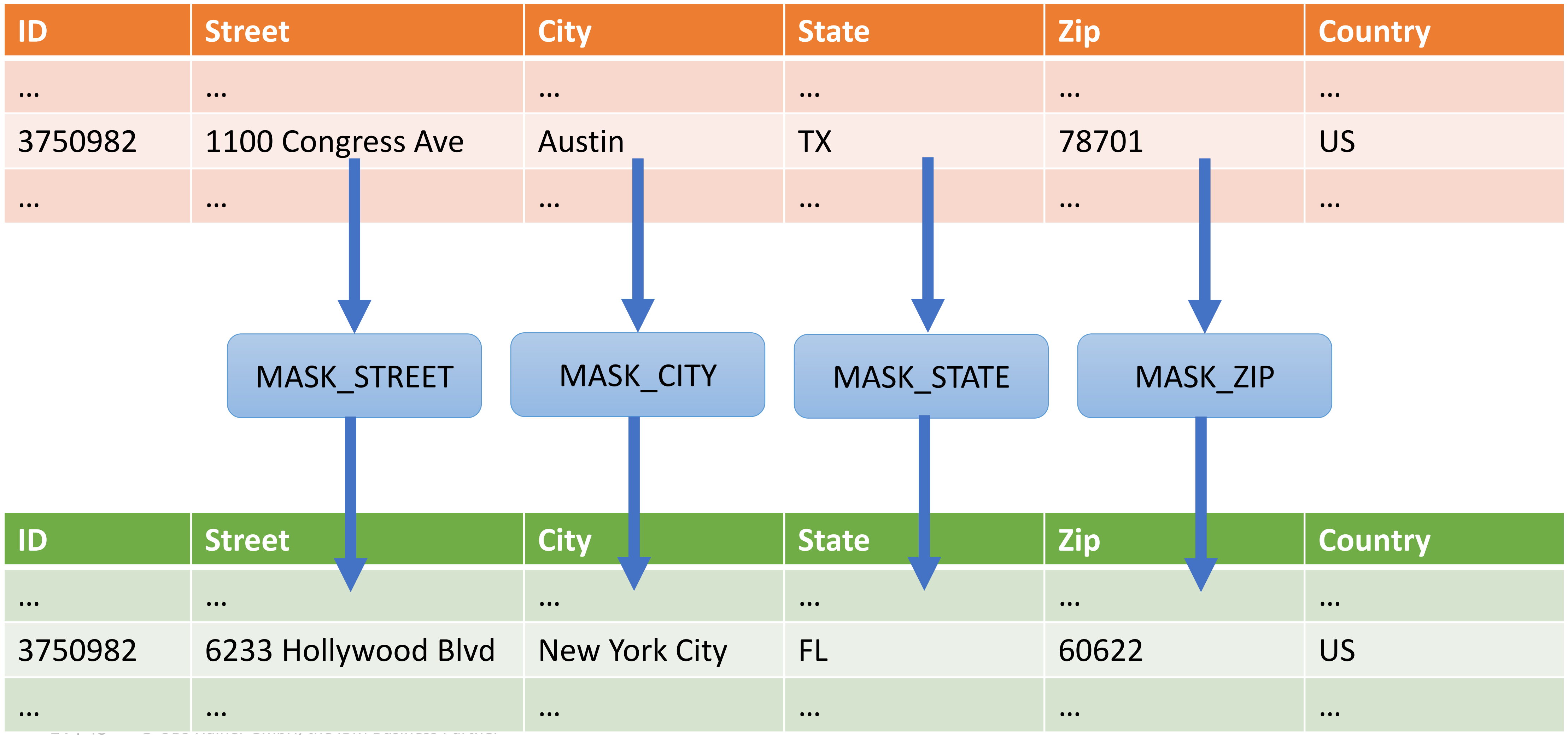
DATA STANDARDIZATION | CONSISTENT DATA

ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982	1100 Congress Ave	Austin	TX	78701	US
…	…	…	…	…	…


Hash input ↓ 46340 ↓ Get street from row 46340
Hash input ↓ 103873 ↓ Get city from row 103873
Hash input ↓ 86233 ↓ Get state from row 86233
Hash input ↓ 58237 ↓ Get zip from row 58237
Address lookup table

WHAT TO CONSIDER

DATA STANDARDIZATION | CONSISTENT DATA



ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982	1100 Congress Ave	Austin	TX	78701	US
…	…	…	…	…	…

ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982	6233 Hollywood Blvd	New York City	FL	60622	US
…	…	…	…	…	…

WHAT TO CONSIDER

DATA STANDARDIZATION | CONSISTENT DATA

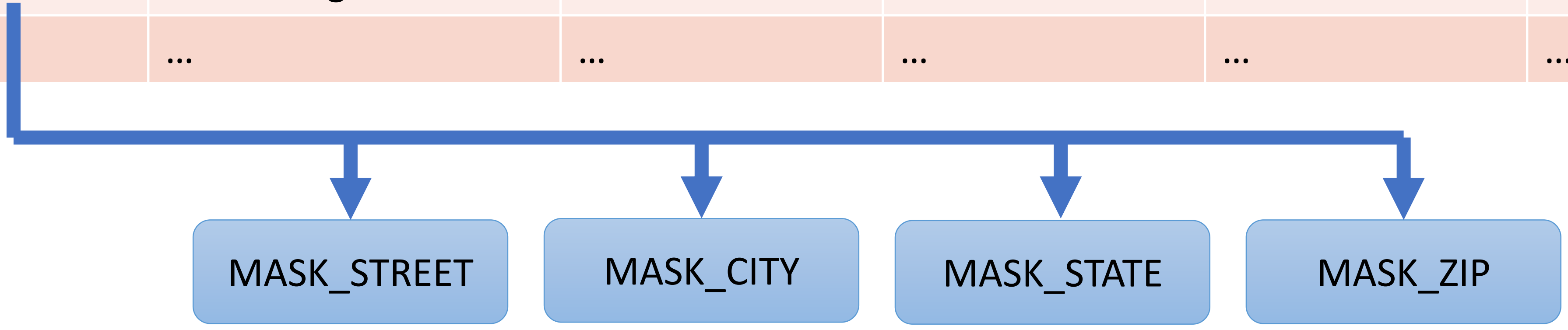
- Use the same input value when hashing different columns with related info
- Good candidate: ID column, any other primary key
- Or concatenate all related columns

Street column:	MASK_STREET	(STREET || CITY || ZIP || STATE)
City column:	MASK_CITY	(STREET || CITY || ZIP || STATE)
Zip column:	MASK_ZIP	(STREET || CITY || ZIP || STATE)
State column:	MASK_STATE	(STREET || CITY || ZIP || STATE)

WHAT TO CONSIDER

DATA STANDARDIZATION | CONSISTENT DATA

ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982	1100 Congress Ave	Austin	TX	78701	US
…	…	…	…	…	…

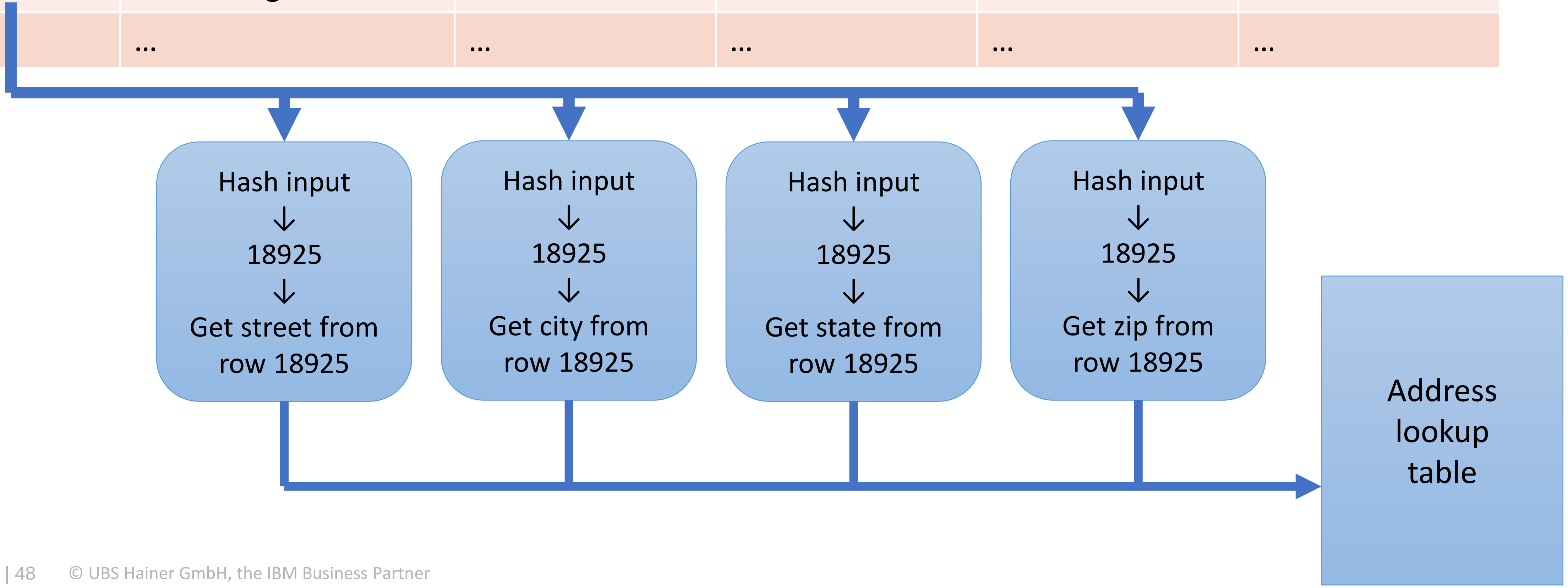


ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982					US
…	…	…	…	…	…

WHAT TO CONSIDER

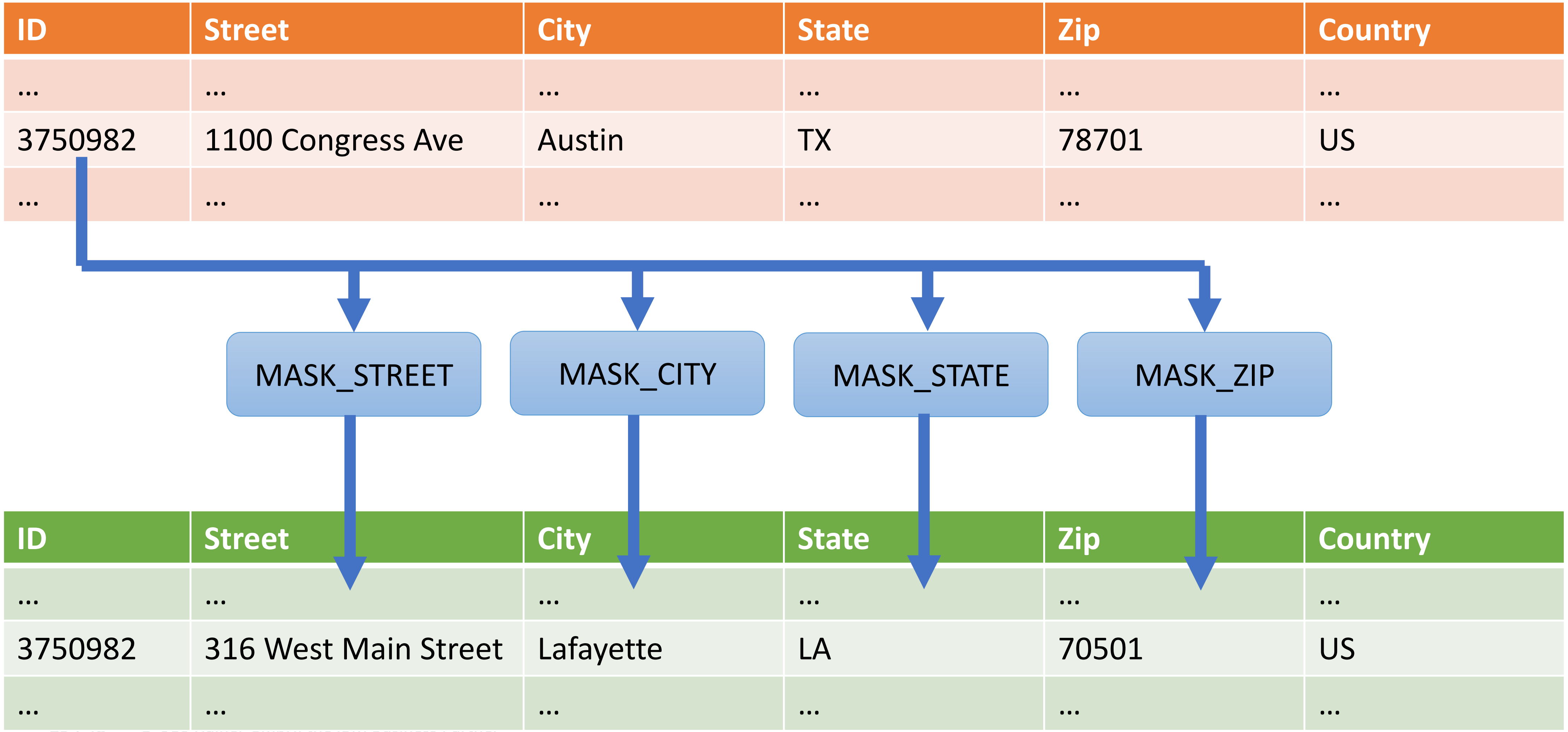
DATA STANDARDIZATION | CONSISTENT DATA

ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982	1100 Congress Ave	Austin	TX	78701	US
…	…	…	…	…	…



WHAT TO CONSIDER

DATA STANDARDIZATION | CONSISTENT DATA



ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982	1100 Congress Ave	Austin	TX	78701	US
…	…	…	…	…	…

ID	Street	City	State	Zip	Country
…	…	…	…	…	…
3750982	316 West Main Street	Lafayette	LA	70501	US
…	…	…	…	…	…

Masking names, addresses, banking info

- Hash ID
- Retrieve masked value from lookup table
- Banking information can be a bit tricky

Routing number and bank name from lookup table

Not all possible account numbers are valid – depends on the bank

Hash source value *x* to a number between 1 and $\left(10^{\lfloor \log_{10} x \rceil}\right) - 1$

(this ensures that the result has the same number of digits; left-pad with zeroes)

Applications rarely check for this

Masking email addresses

- Hash ID
- Retrieve first and last name from lookup table
- Combine first name, last name and domain name

Masking SSNs

- Hash the original SSN to a value between 0 and 999,999,999
- Check for validity
  - No group of digits can be all-zero
  - Area number cannot be 666 or 900 - 999
  - If invalid, re-hash and generate a new number
- Since June 25, 2011 the number is not tied to a location anymore

Masking credit card numbers

- Hash original credit card number to a value between 0 - 9,999,999,999,999,999

  theoretical maximum: 19 digits, in practice 15 or 16 digits

- Keep first *n* digits

  typically 6 for the Issuer Identification Number

- Calculate check digit using Luhn algorithm

Masking date values

- Either

  Decide on a minimum and maximum date

  Determine the number of days *x* between the two dates

  Hash source value to a number between 0 and *x*

  Add *x* to minimum date
- Or

  Hash source value to a number between 0 and 365

  Add x to the date 01/01/xxxx, where xxxx is the year of the source date

Can you mask any column with PII?

- Applications depend on the PII to varying degrees:
- Changing first name, last name or SSN should never affect the outcome of a test
- Changing the address can have consequences, depending on your industry:

  Testing a bookkeeping application for web shop: address does not matter

  Testing premium calculation for flood insurance: address is essential
- You may want to put constraints on the masked data, i.e. stay in zip code area

Additional considerations

- PII may hide in unstructured text fields
- Or in a JPG stored in a BLOB column
- Or in an XML column
- Or in an external file whose path is stored in the database

Implementation

- Consider implementing your algorithms as UDFs
- Functions “live” in Db2
- Fast access to Db2 tables with lookup data
- Can pull masked data from Db2 through DSNTIAUL or other tools
- No unmasked data in flat files
- May be zIIP eligible if called through DDF

Considerations for UDFs

- Write UDFs in SQL/PL
  - No external load modules
  - No WLM required, no task switching
- Performance considerations
  - Avoid calling other UDFs from within a UDF
  - Avoid recursion
  - Avoid casting
  - Prefer integer processing over string processing
  - Design your UDFs as inline functions

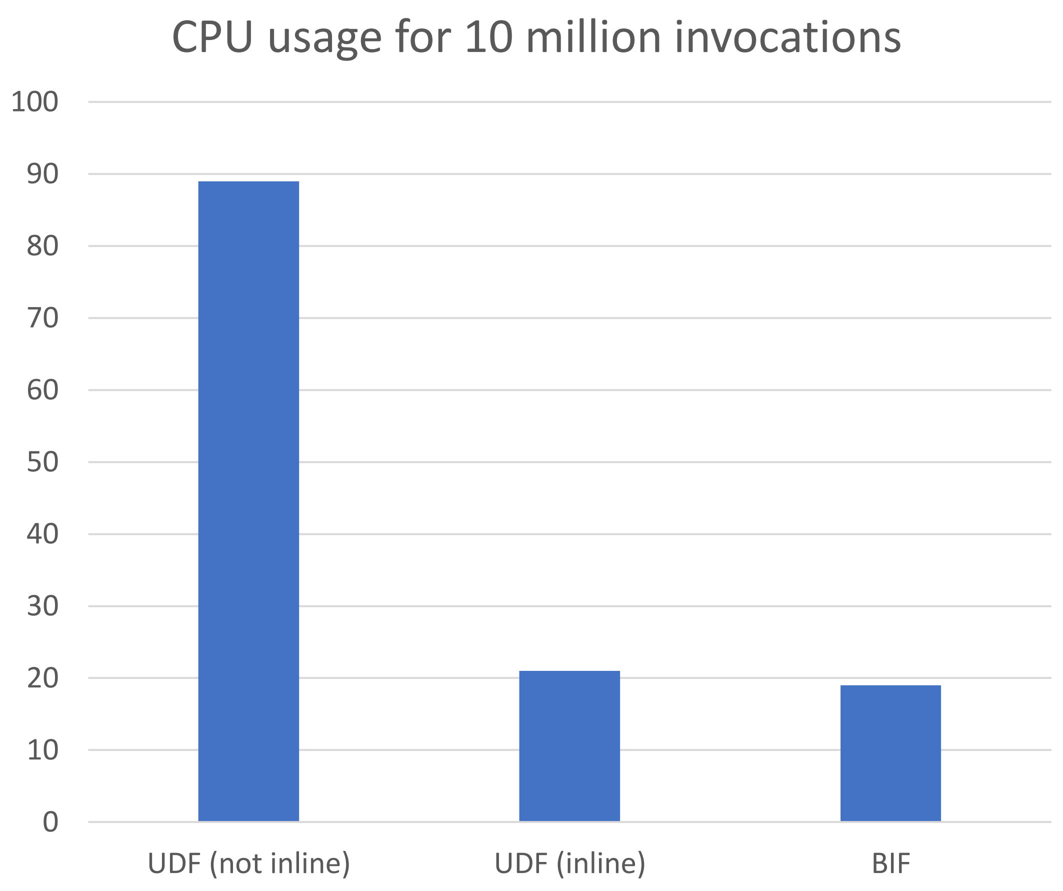
UDF performance comparison

- Inline functions:

  Function consists of a single RETURN statement

- Only use these UDF attributes

  LANGUAGE SQL, SPECIFIC, PARAMETER CCSID, NOT DETERMINISTIC, DETERMINISTIC, NO EXTERNAL ACTION, EXTERNAL ACTION, READS SQL DATA, CONTAINS SQL, CALLED ON NULL INPUT, STATIC DISPATCH



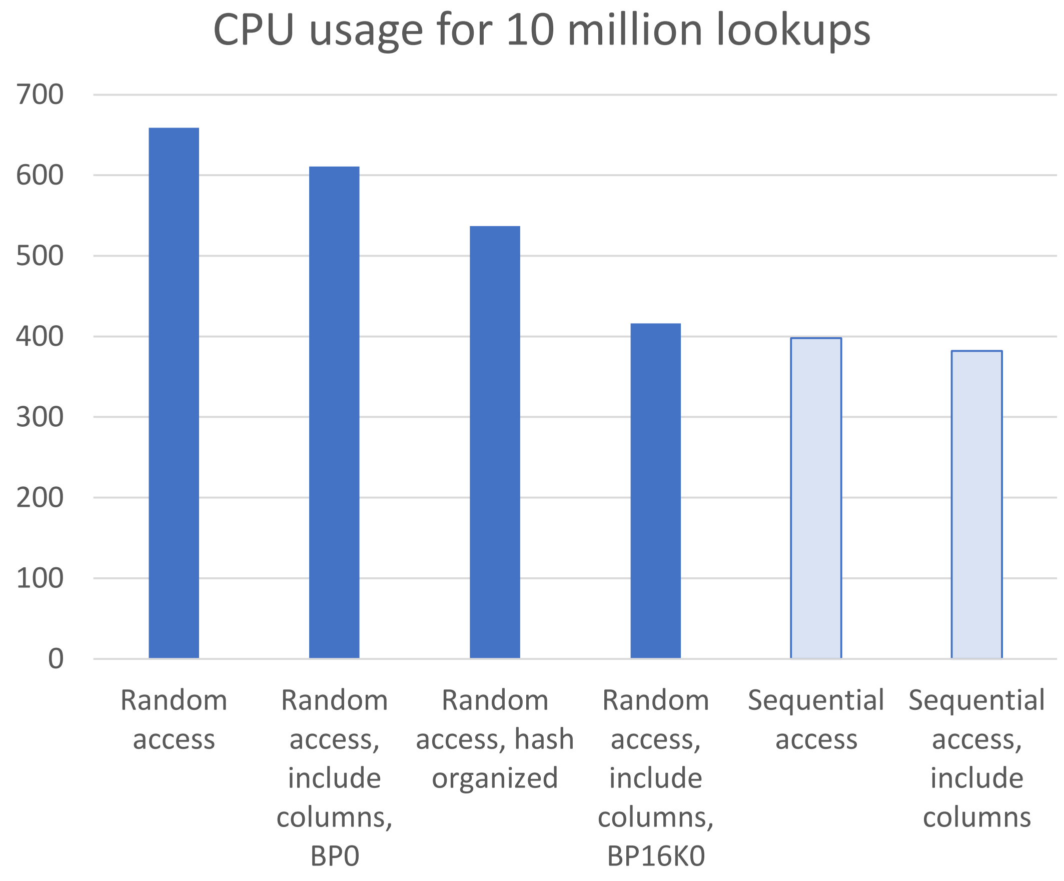
Considerations for lookup tables

- Use unique indexes with include columns

  Use large index page size to minimize number of levels

- Hash organized tablespaces OK, not great

  Also, hash organization will not be supported in future Db2 releases



Masking is not a one-time project

- It is not sufficient to mask data once and keep using the data for a decade
- Changing structures in production force a test data refresh
- Over time, test environments become less useful and "stale"
  - People's age may affect tests
  - Date and timestamp based calculations: Due dates, expiration dates, late fees
  - Masked data needs to be refreshed, too!
- Masking should be a building block in your test data management strategy

Embedding masking in a TDM concept

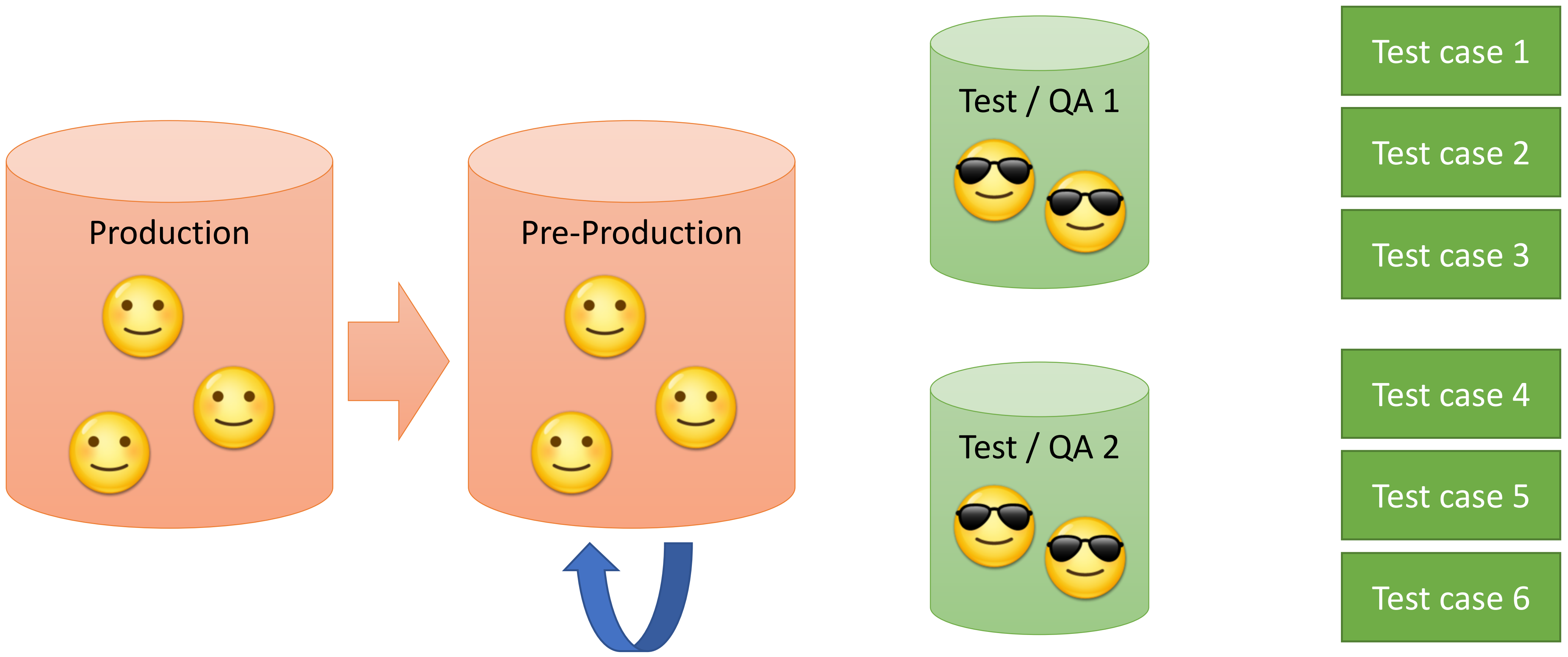
Several points in the process where you can apply masking:

- Option 1: Create a masked "base" environment

  Example: Create a clone of your productions, then mask the data in the cloned environment

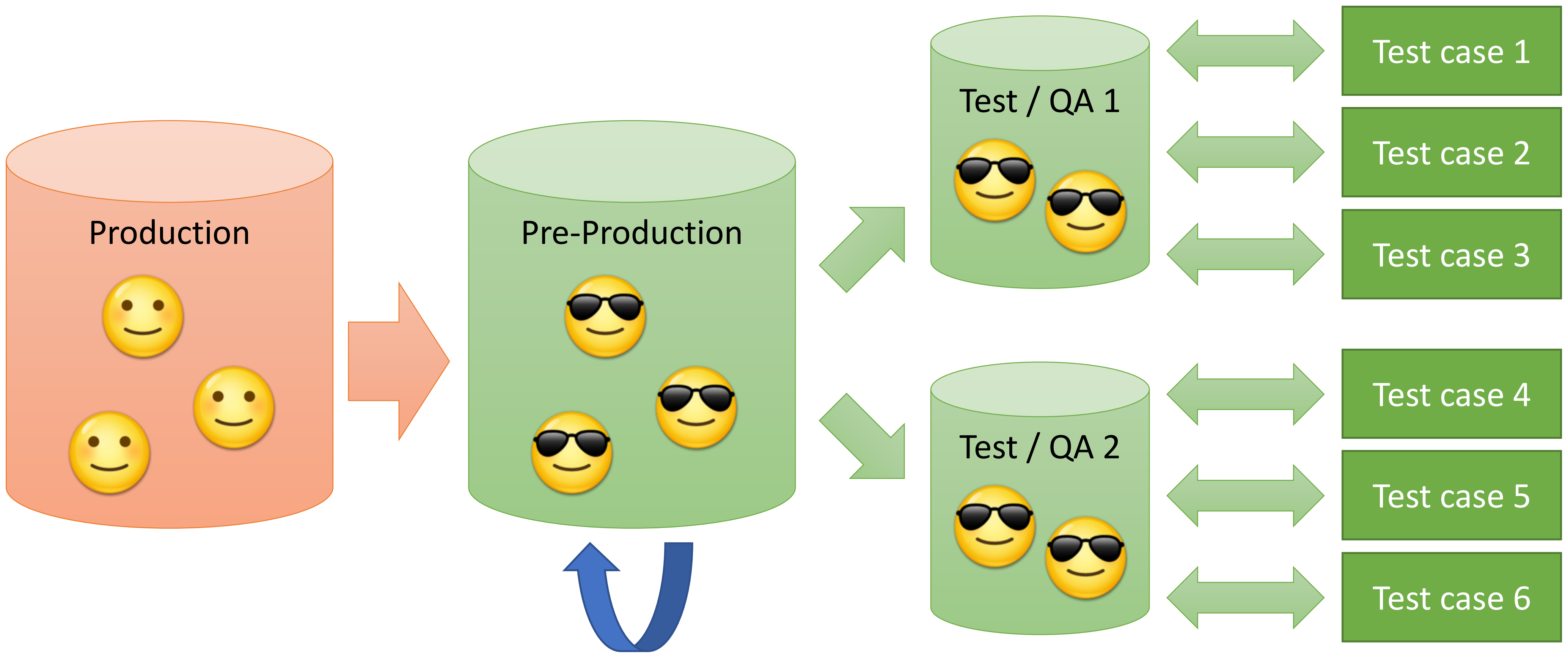
EMBEDDING MASKING:

CREATE A MASKED BASE ENVIRONMENT



EMBEDDING MASKING:

CREATE A MASKED BASE ENVIRONMENT



Several points in the process where you can apply masking:

- Option 1: Create a masked "base" environment

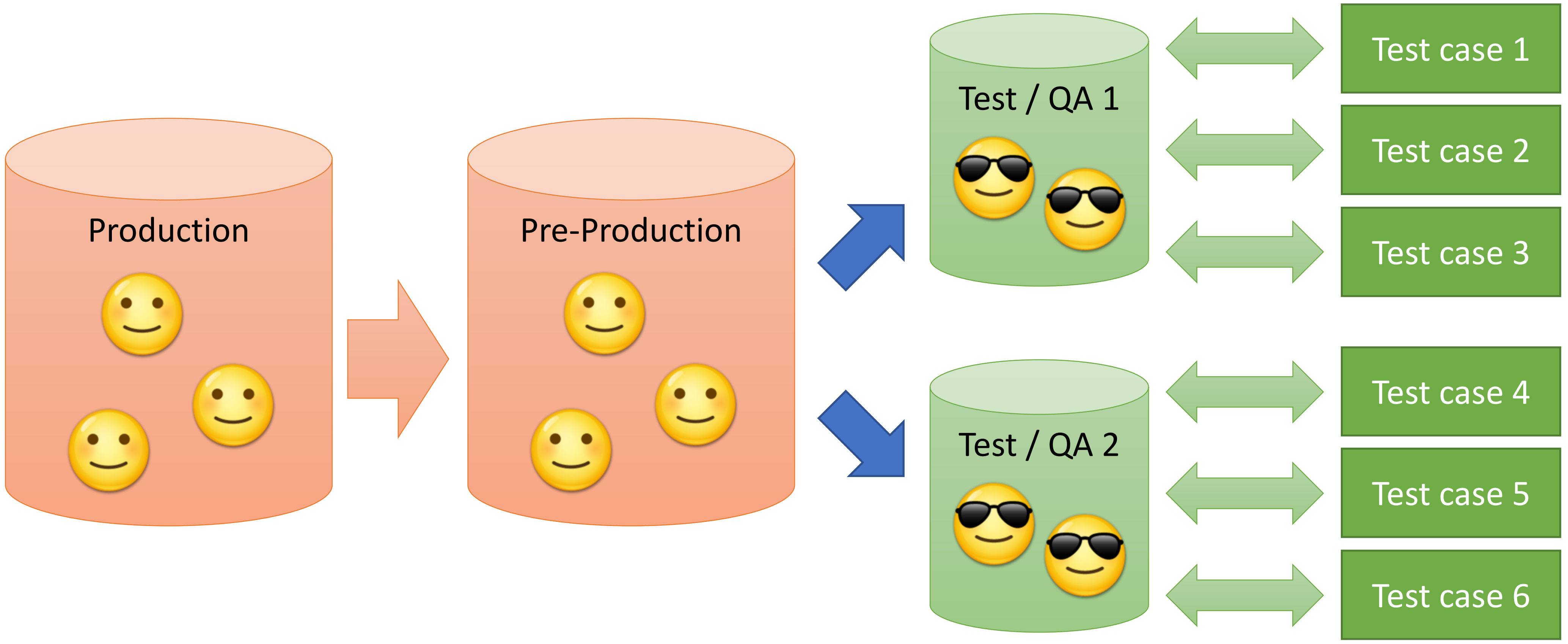
  Example: Create a clone of your productions, then mask the data in the cloned environment

- Option 2: Masked data on the fly every time it is copied

  Example: Copy directly from production to test, but mask data in-memory while it is being copied

BCV5 SAVES UP TO 90% IN 3 KEY AREAS:

CREATE A MASKED BASE ENVIRONMENT | MASK DATA ON THE FLY



Several points in the process where you can apply masking:

- Option 1: Create a masked "base" environment

  Example: Create a clone of your productions, then mask the data in the cloned environment

- Option 2: Masked data on the fly every time it is copied

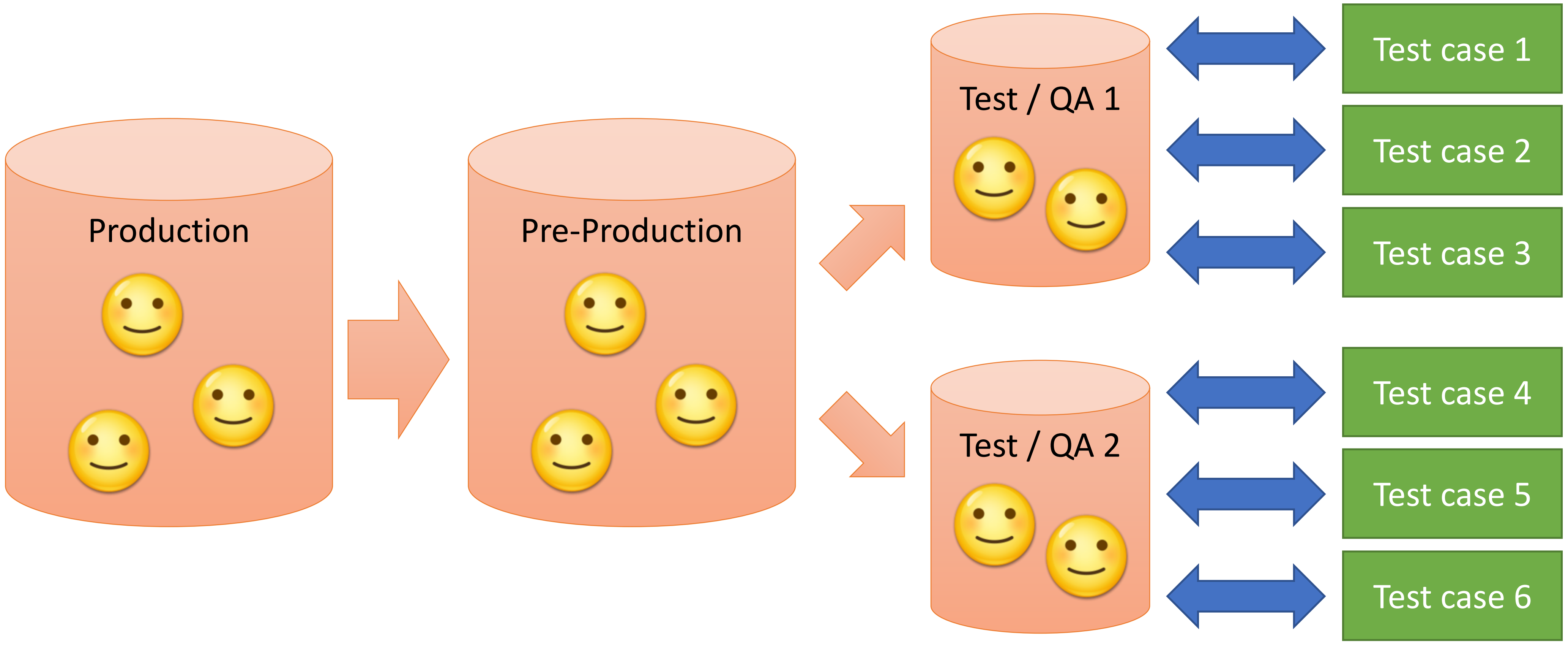
  Example: Copy directly from production to test, but mask data in-memory while it is being copied

- Option 3: Mask transparently upon data access

  Example: Use Db2 column masks to conditionally return modified values for certain users

BCV5 SAVES UP TO 90% IN 3 KEY AREAS:

CREATE A MASKED BASE ENVIRONMENT | MASK DATA ON THE FLY | MASK UPON DATA ACCESS



BCV5 Masking Tool

- Delivers over 30 powerful hashing and data masking functions
- Can be used out of the box
- No customization necessary unless you want to
- Existing hashing & masking functions can also serve as a basis for your algorithms
- Comes with pre-defined lookup tables

  Millions of names in different languages

  Millions of banks in different countries

  Millions of addresses in different countries

Thank you for your attention!

Your UBS Hainer Team

www.ubs-hainer.com
info@ubs-hainersoftware.com