

Privacy, discovery, subsetting and masking: the concepts

The sensitive data and laws that shape every project, then three of the four core techniques explained in plain words before any tool appears: finding data, shrinking it, and hiding it safely.

Sensitive data, privacy laws and policies

Production is full of real people's private details, which is why teams cannot simply copy it into testing (the why-TDM chapter). Before any tool can protect data, someone has to say what "sensitive" means, which laws care about it, and how a tool can recognise it among thousands of columns.

This is the vocabulary layer of TDM. Discovery, subsetting, masking and all three tools use the words defined here: PII, PHI, PCI, policy, rule, data domain, classification.

NOT LEGAL ADVICE

The laws below are summarised in plain words so conversations at work are easier to follow. Real compliance decisions belong to the organisation's compliance officer and legal team.

2.1 Words to know first

Sensitive data

Any data that could hurt a person or a company if it leaked or was misused: identity details, health records, payment details, passwords, trade secrets. "Sensitive" is a business and legal judgement, not just a technical one.

PII (Personally Identifiable Information)

Data that identifies a person, alone or combined with other data. Examples: name, email, phone number, a government ID number, a home address.

PHI (Protected Health Information)

Health-related data that can be tied to a person: diagnoses, treatments, medical record numbers, insurance member IDs. Informatica describes its PHI pack as aimed at the healthcare and pharmaceutical industries (User Guide 10.5.10, p.79).

PCI (Payment Card Industry) data

Payment card details such as the card number and expiry date. Informatica describes its PCI pack as aimed at banking and finance (User Guide 10.5.10, p.80).

Identifier

A piece of data that points to one person. A *direct identifier* does it on its own (an SSN). An *indirect identifier* (also called a quasi-identifier) only does it when combined with others (zip code plus birth date).

Re-identification

Working out who a "nameless" record belongs to by combining it with other information.

Data classification

Sorting data into sensitivity levels (for example public, internal, confidential, restricted) so each level gets the right protection.

Data domain

A named kind of data, such as "Social Security number" or "First name", described by how the data looks and what its column is usually called. Both Informatica and Delphix use this word, with slightly different meanings.

Rule

In Informatica, the logic that changes a value: a masking rule uses a masking technique (such as substitution) with chosen settings. Rules are covered in the masking concepts chapter.

Policy

In TDM, a named bundle that says which kinds of data are sensitive and how to protect each kind. Section 2.8 shows the exact Informatica structure.

Data steward

The person who knows a data set's meaning and quality and answers questions such as "what is in this column and who may see it?".

Compliance officer

The person responsible for making sure the organisation follows laws and internal rules about data.

2.2 What is sensitive data?

Start with the ShopSphere database from the data basics chapter. Three example rows from **CUSTOMERS** :

SHOPSPHERE CUSTOMERS, FIRST THREE ROWS (ALL FAKE)

customer_id	first_name	last_name	email	phone	ssn	date_of_birth	zip_code
1001	Asha	Menon	asha. men on@e xam ple.c om	98765 43210	123 -4 5- 67 89	1990-04-12	560001
1002	Ravi	Kumar	ravi.k @exa mple. com	91234 56780	23 4- 56 -78 90	1985-11-02	600017
1003	Meera	Nair	meer a.n@ exam ple.c om	99887 76655	34 5- 67 -8 90 1	1999-07-23	682001

Ask one question of each column: *if this leaked, could someone be harmed or could ShopSphere be in trouble?*

- `customer_id` is just a number the system made up. On its own it harms nobody. But it links to orders, so it still needs careful handling (more in the masking concepts chapter).
- `first_name` and `last_name` tell you who the person is.
- `email` and `phone` let a stranger contact, spam, phish or impersonate the person.
- `ssn` (a government ID number) is the classic example: with a name and a birth date it is enough to open credit in someone's name.
- `date_of_birth` is used by banks and call centres to "prove" who you are.
- `zip_code` looks harmless. Section 2.4 shows why it is not.

WHY IT MATTERS IN TESTING

Production data is protected by firewalls, access reviews and monitoring. A test database usually has none of that. Many people can log in, copies get passed around, and some environments live with outside vendors or in the cloud. If a real SSN sits in a test table, it carries production's legal weight into a place with a fraction of production's security. That mismatch is the core of the problem.

Delphix's own introduction makes the same point from the tool side: organisations face data breaches and pressure from regulators, and encryption helps protect data on disks and in transit but is, in their words, ill-suited for protecting non-production environments used for development, testing and reporting. Their answer is to replace confidential values with fictitious but realistic ones. (Delphix "Introduction to Delphix Masking", source page [7](#)).

A QUICK TEST YOU CAN APPLY AT WORK

Look at a column and ask three questions: (1) Does it identify a person? (2) Would a regulator or customer be upset if it leaked? (3) Would a tester still be able to do their job if it held a believable fake value instead? If the answers are yes, yes, yes, the column is a masking candidate. Question 3 matters: if the column cannot be faked without breaking the test, you have found a design problem to solve, not a reason to leave real data in place.

2.3 The three families: PII, PHI and PCI

People in TDM talk in three abbreviations. They overlap, and a single column can belong to more than one family, but each one points at a different law or standard and a different industry.

2.3.1 PII: personally identifiable information

PII is data that identifies a person. The definition differs a little between countries and laws, but the everyday list is stable:

- Names, email addresses, phone numbers, postal addresses.
- Government-issued numbers: SSN (US), passport number, driver's licence number, national ID. India's Aadhaar and PAN are comparable examples.
- Date of birth, age, gender, place of birth.
- Online identifiers such as IP addresses.

ShopSphere's `CUSTOMERS` table is almost pure PII: `first_name` , `last_name` , `email` , `phone` , `ssn` , `date_of_birth` , `zip_code` and `tax_id` .

Informatica ships a ready-made **PII policy pack**. It lists 26 data domains, each paired with a default masking rule (User Guide 10.5.10, pp.78 to 79). Reading the list is a quick way to see what a vendor treats as PII:

Data domains in the Informatica PII policy pack (User Guide 10.5.10, pp.78-79), grouped by us for readability	
Group (our label)	Data domains in the pack
Names	First Name, Last Name, Full Name
Contact	Email Address, Phone, Full Address UPPER, Street Name, State, ZIP, Country
Government and travel IDs	Social Security Number, Passport, Drivers License Number, UK National Identifier, Vehicle Registration Number
Personal facts	Birth Date, Birth Place, Age, Gender
Money and work	Credit Card, Salary, Job Position, Organization Name
Other	IP Address, School Name, Grades

Notice that the pack includes things you might not call "identity", such as Salary and Grades. A vendor's list is a starting point. Your organisation decides what else counts.

2.3.2 PHI: protected health information

PHI is health data that can be linked to a person. The health data alone ("type 2 diabetes") is not what makes it PHI. The link to an identifiable individual does. A list of diagnoses with no names or IDs and no way to trace back is a different thing from a patient record.

ShopSphere is a retailer, so it has no medical records. To make the idea concrete, imagine a hypothetical ShopSphere feature, "ShopSphere Pharmacy", with a table `PRESCRIPTIONS(rx_id, customer_id, diagnosis_code, drug_name, insurer_member_id)`. Then:

EXAMPLE DATA: A HYPOTHETICAL SHOPSPHERE PRESCRIPTIONS TABLE

rx_id	customer_id	diagnosis_code	drug_name	insurer_member_id
9001	1001	E11.9	Metformin	HP-5501-882
9002	1003	J45.909	Salbutamol inhaler	HP-5501-913

Here `diagnosis_code` and `drug_name` are health facts, `insurer_member_id` is an identifier, and `customer_id` links them to Asha or Meera. Together they are PHI. The same pack idea exists in Informatica: the **PHI policy pack** lists 17 domains, among them Medical Record Number, Health Plan Beneficiary Number, Device Identifier Serial Number, Certificate License Number, Account Number, Birth Date, Email Address, IP Address, Web URL, Social Security Number and ZIP (User Guide 10.5.10, p.79).

LOOK CLOSELY AT WHAT THE PACK CONTAINS

The PHI pack is mostly *identifiers* that appear in healthcare data: names, record numbers, plan numbers, dates, ZIP. It does not try to list every diagnosis. That tells you how privacy rules think: what makes a health record dangerous is that it can be traced to a person, so removing or masking the identifiers does most of the protective work.

Delphix organises its detection logic into *profile sets*. Besides the recommended ASDD Standard set, it ships legacy sets named "Financial - Legacy" and "HIPAA - Legacy" (Delphix "Out of the box profiling settings"). Delphix's standard set also has domains with explicitly medical names, such as `MEDICAL_CODES`, `MEDICAL_DRUG` and `BLOOD_TYPE`, and a domain called `HIPAA_DATE` (Delphix "ASDD standard profile set" listing). The words "HIPAA Date" hint at a rule HIPAA has about dates, which section 2.5 explains.

2.3.3 PCI: payment card data

PCI refers to the Payment Card Industry. The *PCI DSS* (Data Security Standard) is a rulebook created by the card brands for anyone who stores, processes or sends card data. It is not a government law. It is a contractual standard: merchants who ignore it can lose the right to take card payments, and can be fined by their payment partners. In TDM talk, "PCI data" means card number, card expiry date, security code, cardholder name.

ShopSphere's shared schema has no card column, but any real online shop would have one (or would store a token from a payment provider). If it did, it would look like `4539 1596 8210 2773`. This is the number Informatica's own documentation uses as a masking example (User Guide 10.5.10, p.158).

The Informatica **PCI policy pack** is short: Account Number, Birth Date, Credit Card, Expiration Date, First Name and Last Name (User Guide 10.5.10, p.80).

DO NOT COPY THESE NUMBERS FROM THE DOCS INTO YOUR OWN TESTS

A card number from a documentation page may look fake and still be a real, working card by coincidence. When you need card numbers for testing, use numbers designed for testing (payment providers publish "test card numbers") or let a generator or masking tool make numbers that pass the format check but belong to no account.

2.4 Direct and indirect identifiers: why "just remove the names" fails

A common beginner idea is: "We will delete the name column and then the data is anonymous." It is not, and the reason is the most important privacy idea here.

2.4.1 Two kinds of identifier

Direct identifiers

Point to one person by themselves.

- Full name (mostly)
- SSN, passport number
- Email address, phone number
- Card number

Removing or replacing these is the obvious first step.

Indirect identifiers (quasi-identifiers)

Harmless alone, identifying in combination.

- Zip or postal code
- Date of birth
- Gender
- Job title, employer
- Rare product or rare diagnosis

Easy to forget, which is why they cause leaks.

Research on public records, most famously by the computer scientist Latanya Sweeney, has found that a large share of people in the United States can be singled out from just three facts: zip code, full date of birth and gender. The exact percentage depends on the data set and the year, so no figure is quoted here. Remember the shape of the result: three boring columns, one unique person.

2.4.2 A worked example with ShopSphere

Imagine a careless tester shares an "anonymised" extract with a contractor. Names, emails, phones and SSNs were deleted. What remains:

"ANONYMISED" EXTRACT: DIRECT IDENTIFIERS REMOVED, INDIRECT ONES KEPT			
customer_id	date_of_birth	zip_code	total_spent
1001	1990-04-12	560001	41,250
1002	1985-11-02	600017	9,800
1003	1999-07-23	682001	27,430

Now suppose the contractor also has an outside list: a public club membership page, a leaked mailing list, a social media profile that mentions "born 12 April 1990" and a city. Matching the outside list against the extract on birth date plus zip code finds exactly one row. The contractor now knows that Asha spent 41,250 at ShopSphere. No name was ever in the extract.

1

Take the extract.

Birth date and zip code are in it.

2

Take any outside list that has the same two facts plus a name.

The linking columns do not have to be sensitive on their own.

Expected result: Result: the outside list's "Asha Menon" matches one row of the extract.

3

Read everything else in the matched row.

Everything in the row now belongs to a named person.

WHAT THIS MEANS FOR MASKING

Protecting a data set means looking at *combinations*. It is why the PII and PHI packs both include Birth Date, ZIP and State, not only names and ID numbers. It is also why masking techniques that move a date a little or blur a number (see the masking concepts chapter) exist: they weaken the combination while keeping the data useful for testing.

FREE TEXT IS THE OTHER HIDING PLACE

Informatica's data domain page notes that a Social Security number might sit in an `SSN` column, in a column called `SOCIAL_SECURITY` in another table, and also inside a `COMMENTS` column (User Guide 10.5.10, p.73). A support agent typing "customer 123-45-6789 called about refund" into a notes field has just put an SSN where no column name will ever warn you. Delphix's Free Text Redaction algorithm exists for exactly this problem (the masking concepts chapter explains it).

2.5 Privacy laws and standards in plain words

THIS IS NOT LEGAL ADVICE

Everything in this section is a plain-language orientation , written so you can follow a meeting and ask good questions. Laws change, have exceptions, and apply differently depending on where your company and your customers are. Treat every statement here as "roughly true, check with legal". The tool documentation does not teach law, so none of this comes from the tool sources, except where marked.

Why should a tester care about laws? Because when a court or regulator looks at a data leak, they do not ask "was it production or test?". They ask "was personal data properly protected?". A copy in a test database is still personal data.

2.5.1 GDPR in plain words

The *GDPR* (General Data Protection Regulation) is a European Union law that has applied since May 2018.

- **What it protects:** "personal data", meaning any information about an identifiable person. That is broader than a name and number: online identifiers and location data can count.
- **Who it applies to:** organisations that handle the personal data of people in the EU, even if the organisation sits outside Europe. A ShopSphere selling to customers in Germany would likely be in scope wherever its servers are.
- **Ideas behind it:** there must be a lawful reason to use the data (consent is one such reason); use it only for the purpose it was collected for; collect and keep no more than needed ("data minimisation"); keep it secure; let people see and delete their data.
- **Fines at a high level:** the most serious breaches can lead to fines of up to 20 million euros or 4 percent of worldwide yearly turnover, whichever is higher. Lesser breaches have a lower ceiling.

The consequence for test data. Two GDPR ideas hit TDM directly. First, *purpose*: a customer who gave details so ShopSphere could deliver an order did not necessarily agree to have them copied into a test lab. Second, *minimisation and security*: copying a whole production database to a loosely guarded test server is hard to justify when a small, masked copy would do. The law also treats truly anonymous data (nobody can be identified any more) as outside its scope, which is one reason masking and subsetting are popular. Whether a particular masked data set counts as anonymous is a question for your compliance officer.

2.5.2 HIPAA in plain words

HIPAA is a United States law from 1996 that, through its Privacy and Security Rules, protects health information.

- **What it protects:** PHI, health information that identifies a person (section 2.3).
- **Who it applies to:** "covered entities" (health plans, healthcare providers, healthcare clearinghouses) and their "business associates", meaning vendors who handle PHI for them. A software company testing a hospital's billing system is typically a business associate.
- **Minimum necessary:** people and systems should receive only the least PHI they need for their job. Testers rarely need the real thing.
- **De-identification:** HIPAA describes two accepted ways to de-identify health data so it is no longer treated as PHI. One removes a defined list of 18 kinds of identifiers (names, detailed locations, most

date elements except the year, phone numbers, record numbers and so on). The other has a qualified expert certify that the risk of identifying anyone is very small. This may be why Delphix has a domain called HIPAA_DATE, and why dates and ZIP codes feature so heavily in PHI discussions.

- **Penalties at a high level:** civil penalties in tiers depending on how careless the organisation was, and criminal penalties for knowing misuse.

Both tools mention these regimes only briefly. Delphix's introduction lists HIPAA, PCI and GDPR as examples of mandates organisations must stay compliant with, and Informatica packages its PHI and PCI policy packs for healthcare and banking respectively (User Guide 10.5.10, pp.79 to 80). GenRocket's column-profiling page says that identifying PII early helps organisations meet requirements "such as HIPAA".

2.5.3 Other regimes, kept general

Other privacy rules you will hear about (very general; verify details with your compliance team)		
Name	Where	One-line idea
PCI DSS	Worldwide, by card-brand contract	Security standard for anyone handling payment card data. An industry standard rather than a law.
CCPA (California Consumer Privacy Act, later amended)	California, USA	Gives California residents rights over personal information held by businesses, such as knowing what is held and asking for deletion.
DPDP Act (Digital Personal Data Protection Act, 2023)	India	Consent-centred law on processing digital personal data of people in India. Rules and start dates are phased in, so check the current status before relying on any detail.

There are many more (sector rules for banks, national laws in dozens of countries, contract terms from clients). The practical lesson is the same everywhere: personal data in test environments is a liability, and the standard defence is to make sure the test copy no longer identifies anyone.

2.5.4 Four ideas that repeat across laws

Recurring ideas and what they mean for a TDM engineer		
Idea	Plain meaning	What it changes in TDM work
Consent / lawful purpose	People agree, or the law allows, use for a stated purpose.	Testing is a different purpose from running the shop, so real data in test is hard to defend.
Minimum necessary / minimisation	Use only as much data as the job needs.	This is the legal argument for subsetting (see the discovery and subsetting concepts chapter): do not copy 100 percent when 2 percent will do.
De-identification / anonymisation	Make the data no longer point to a person.	This is the legal argument for masking (see the masking concepts chapter).
Security safeguards	Protect what you keep with access control and logging.	Applies to the TDM tool itself and to any unmasked staging copy.

2.6 Data classification: how sensitive is each column?

Data classification means labelling data by how much damage a leak could do, so that you can apply protection in proportion. Organisations choose their own scheme. A typical four-level scheme looks like this:

A typical four-level classification (example; organisations' levels differ)

Level	Meaning	ShopSphere example	Typical test-data action
Public	Safe for anyone to see	PRODUCTS.product_name , PRODUCTS.price	Copy as is
Internal	Fine inside the company, not outside	ORDERS.order_date , ORDERS.total_amount	Copy as is, or lightly vary
Confidential	Harms people or the business if leaked	CUSTOMERS.email , phone , first_name , last_name , zip_code	Mask with realistic replacements
Restricted	Serious legal or financial harm	CUSTOMERS.ssn , tax_id , date_of_birth , card numbers	Mask strongly, or never move to test

Two things are worth noticing. First, classification is about *columns* (and sometimes whole tables), not rows. Second, the result of classifying is a plan: each level maps to an action. A column with no label gets no action, so unlabelled sensitive data is the dangerous kind.

2.6.1 How the tools record sensitivity

- **Informatica.** When you create a data domain you set a *sensitivity level* for all its columns, chosen from levels that the Administrator defines (User Guide 10.5.10, p.75). Separately, each column in a project has a *Sensitive* flag. The guide says the flag is "for reference only" and does not affect processing, and that assigning a data domain to a column marks it sensitive automatically (p.89). A different property, *Restricted*, does change behaviour: the integration service does not write a restricted column's value to the target (p.89).
- **Delphix.** A *domain* ties a category of sensitive data (the terminology page's example is social security numbers) to the way it should be secured. The profiler assigns the domain to a column and, with it, the domain's default masking algorithm (Delphix "Discovering your sensitive data").

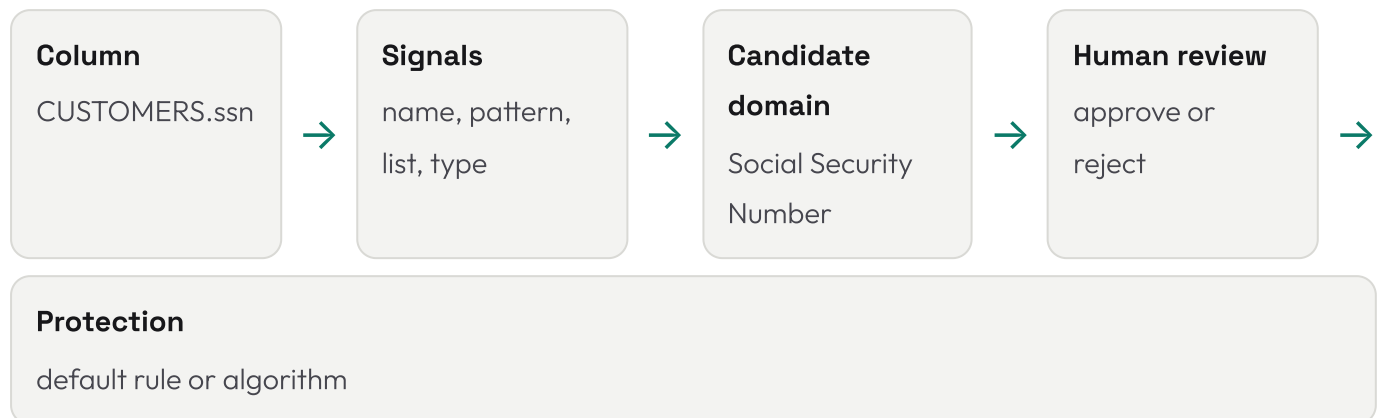
SENSITIVE FLAG VERSUS MASKING RULE

In Informatica, marking a column "sensitive" does not mask it. It documents your decision. Masking happens only when a rule is assigned to the column and a plan runs (see the Informatica data masking chapter). Beginners often tick "sensitive" and think the job is done.

Many laws treat some data as especially sensitive: ethnic origin, political opinions, religion, trade union membership, health, sexual orientation, biometrics. The Delphix ASDD Standard profile set has domains for these (for example ETHNICITY, RELIGIOUS_ORIEN, POLITICAL_ORIEN, UNION_MEMBERSHIP_NO, BIOMETRIC and MEDICAL_CODES), and the default masking method it shows for BIOMETRIC is NullValueLookup. For the most delicate data, blanking a value is often better than faking it.

2.7 How sensitive data is found

A real production database can have hundreds of tables and tens of thousands of columns. Nobody can read them all, and column names are often cryptic (C0L_17 , FLD_X2). So tools look for sensitive data automatically. At concept level they all rely on a small set of signals. The full hands-on process is the subject of the discovery and subsetting concepts chapter, the Informatica discovery and profiling chapter and the Delphix profiling chapter. Here we only need the ideas.



This chain is the same in all three tools, even though each names the boxes differently. Keep it in mind: every feature in the later tool chapters is a way to fill in or control one of these boxes.

2.7.1 Three signals

The three detection signals (general concept, with where each tool uses it)			
Signal	What it checks	Strength	Weakness
Column name (metadata)	Does the name look like <code>ssn</code> , <code>email</code> , <code>phone</code> ?	Fast, needs no data access	Misses cryptic names. Flags harmless columns with unlucky names.
Data pattern	Do the values <i>look like</i> an SSN, a phone number, an email?	Works even when the name is useless	Slower, reads real data. Nine digits may be an SSN or an order code.
Dictionary or list of values	Are the values found in a list, such as common first names or country names?	Finds things with no fixed shape, such as names	Lists are language-specific and never complete.

A fourth signal, **data type and length**, is used to rule things out: an SSN cannot live in a column that holds one-character flags.

2.7.2 What Informatica does: data patterns and metadata patterns

Informatica describes a *data domain* using regular expressions. A *regular expression* (regex) is a compact text recipe for a pattern, for example "three digits, a hyphen, two digits, a hyphen, four digits". The guide separates two kinds (User Guide 10.5.10, pp.74 to 75):

- A **data pattern** matches the values. Its example for Social Security numbers is `[0-9]{3}-[0-9]{2}-[0-9]{4}`.
- A **metadata pattern** matches column names. Its examples for Social Security numbers are `.*SSN*`, `.*SOCIAL*` and `.*SECURITY*`.

A column joins the data domain if it matches *any* expression in the domain. When you run a data domain profile you choose to search the data only, the column name only, or both, how many rows to read (default 1,000) and a minimum conformance percent, since not every row has to match the pattern (pp.95 and 100).

Data pattern	<code>[0-9]{3}-[0-9]{2}-[0-9]{4}</code>	matches	123-45-6789
Metadata pattern	<code>.*SSN*</code>	matches	SSN, CUST_SSN, SSN_NO

2.7.3 What Delphix does: profile sets, classifiers and domains

Delphix calls the detection step *profiling*. A profiling job examines metadata (column names and types) and possibly the data itself, and when it decides a column is sensitive it assigns the matching domain and that domain's default masking algorithm. It only covers the tables and files in the chosen rule set (Delphix "Discovering your sensitive data", source page 7).

The logic is packaged in **classifiers**, grouped into **profile sets**. Four classifier frameworks exist, and they map neatly to our signals:

Delphix classifier frameworks (Delphix "Discovering your sensitive data")		
Framework	What it examines	Level
PATH	The path to the data; for a database that means table and column name. Uses regex or exact match.	Column
TYPE	Data type and length. Used to reject domain matches. Types: String, Number, Date, Binary.	Column
REGEX	The data itself, using regular expressions to match or reject a domain.	Data
LIST	Whether data values appear in a list of values, to match or reject a domain.	Data

The recommended built-in profile set, the "ASDD Standard" set, includes list classifiers that can detect domains such as FIRST_NAME and LAST_NAME even when column names are meaningless, though data-level detection is limited to English values (Delphix "Out of the box profiling settings").

2.7.4 One real example: how Delphix recognises an SSN

The ASDD Standard listing (Delphix "ASDD standard profile set", source page 7) shows the SSN domain is defined by three classifiers. In our words:

1

A PATH classifier.

A regex on the column path catches names like `ssn` or `social security number` with variants such as `social_security_no`. Its match strength is listed as 0.67.

2

A REGEX classifier on the data.

Patterns for the nine-digit format. The dashed or spaced form (123-45-6789) is listed with match strength 0.7, while nine bare digits are listed with only 0.1.

Expected result: The lower number for bare digits fits the idea in the weaknesses table above: nine digits in a row are far more likely to be something other than an SSN.

3

A TYPE classifier.

The column must be a Number or String of at least 9 characters. This rejects tiny columns.

Not documented: how the strengths are combined into a final decision. Read the product page for your version before relying on the numbers. The design is what matters: several weak clues, each with a weight, are combined to decide.

2.7.5 What GenRocket does: column names only

GenRocket's "Data Column Profiling" scans column *headers*, not the data. It checks whether each column name contains a keyword from a list: 102 built-in names plus any your organisation adds (for example `user_email` matches `email`, `client_ssn` matches `ssn`). It then shows you the matches to choose from (GenRocket "PII Discovery with Data Column Profiling", source page 7). It is the "column name" signal on its own. The upside is that the tool never needs to read real values.

You will read many patterns in policies, so here is the minimum: `[0-9]` means one digit, `{n}` repeats the previous item n times, `?` makes it optional, `*` means zero or more, and a backslash makes a special character literal (User Guide 10.5.10, pp.74 to 75). A data pattern says what a value *looks like*, not what it *is*: `[0-9]{10}` matches a phone number and also a ten-digit order reference. That is the source of false alarms (false positives); a real SSN column stored without hyphens that fails the pattern is a false negative. Dialects differ: Delphix patterns start with `(?i)` for case-insensitive matching.

2.7.6 ShopSphere example: what each signal finds

EXAMPLE: HOW THE THREE SIGNALS WOULD TREAT SHOPSPHERE COLUMNS

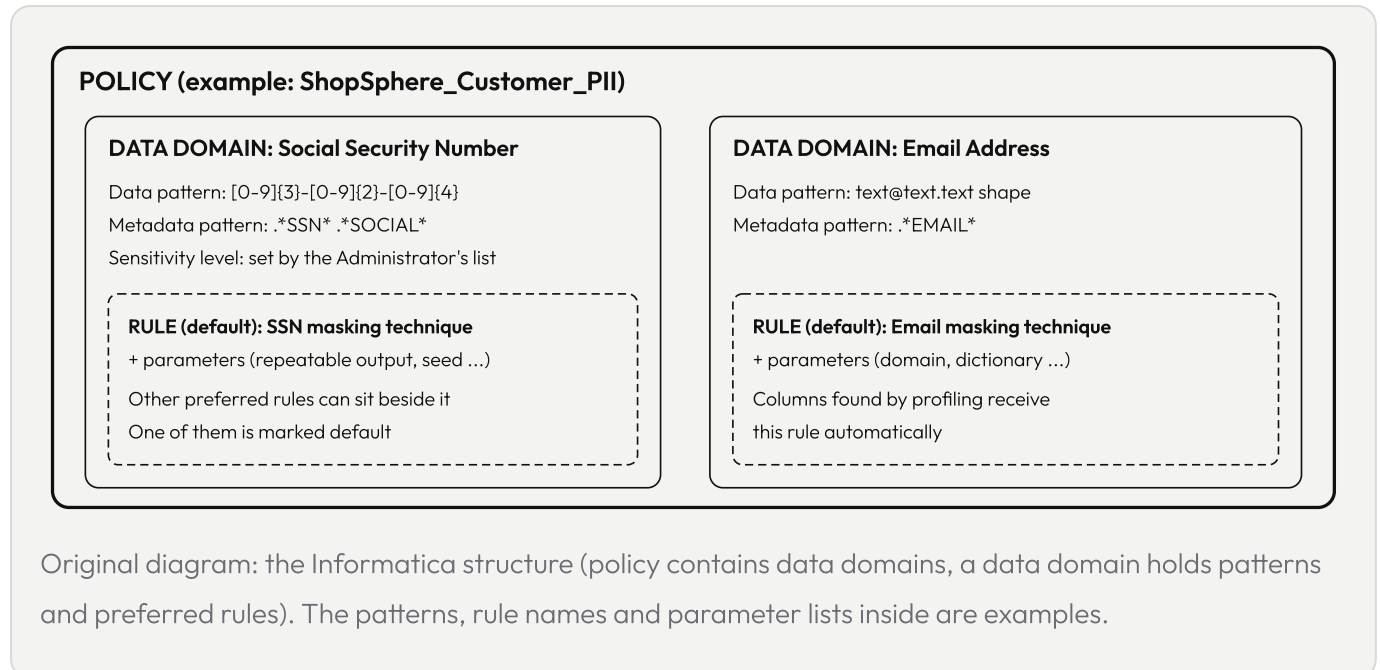
Column	Name signal	Data signal	Verdict
ssn	Strong (ssn)	Strong (999-99-9999)	Sensitive, easy
email	Strong	Strong (text@text.text)	Sensitive, easy
phone	Strong	Medium: ten digits, many things are ten digits	Sensitive
first_name	Medium (contains "name")	List of first names finds Asha, Ravi, Meera	Sensitive
tax_id	Depends on the tool: Delphix's ASDD Standard set has a Tax ID name pattern, a short keyword list may not	Country-specific formats	Found by some tools, missed by others: verify
zip_code	Strong	Six digits, ambiguous	Indirect identifier, needs a human decision
order_id	None	Pure number, could resemble anything	Possible false alarm

AUTOMATIC DOES NOT MEAN FINISHED

Detection produces *candidates*. A person must review them: remove false alarms and add the misses (a column like `tax_id` is caught by a tool whose pattern list includes it and missed by one whose list does not). Informatica builds this review into its workflow: profile results show columns and proposed domains, and a user approves the ones to use (User Guide 10.5.10, p.105). The the discovery and subsetting concepts chapter covers false positives and false negatives in detail.

2.8 Policies, rules and data domains

Now the vocabulary promised at the start. A TDM tool needs one place that records the answer to "what is sensitive here, and how do we protect each kind?". Informatica calls that place a **policy**. The structure is worth learning exactly, because it explains many menus you will see later. (User Guide 10.5.10, p.71)



2.8.1 What each piece is

Policy (Informatica)

"A data masking component that describes the methods to maintain the privacy of specific types of source data". It contains data domains. You assign a policy to a data source in a project, and one policy can serve several projects (p.71).

Data domain (Informatica)

Describes the functional meaning of a column based on its data or its name. It carries data patterns, metadata patterns, a sensitivity level, a status (enabled or disabled) and, optionally, preferred masking and generation rules (pp.73 to 75).

Rule (Informatica)

"A logic to mask sensitive data or to generate test data": a technique plus its parameters. Rules exist for string, numeric and date data (p.72).

Preferred rule and default rule

When you attach several rules to a domain they are "preferred rules", and one is the default, applied to all columns in the domain unless you pick another. The guide's example: a Last_Name domain with a shuffle rule as default and a substitution rule as another preferred rule (p.73).

Policy pack

A ready-made policy shipped with TDM. There are three: PII, PHI and PCI, visible after installation (p.77).

The order of work does not matter. The guide says you can create policies, rules and domains in any order, then connect them: create masking rules, define domains, assign rules to domains, create a policy, assign domains to it (p.72).

EXAMPLE SHOPSPHERE POLICY

ShopSphere's TDM engineer, with the compliance officer's approval, builds a policy

`ShopSphere_Customer_PII` from these domains. The domain names echo the PII pack; the rule choices are our example.

EXAMPLE SHOPSPHERE POLICY: DOMAINS, HOW FOUND, DEFAULT RULE (TECHNIQUE NAMES ARE INFORMATICA TECHNIQUES FROM THE MASKING CONCEPTS CHAPTER)

Data domain	Found by	Columns it should catch	Default rule (example)
First Name	Name pattern, list	CUSTOMERS.first_name	Substitution from a dictionary
Last Name	Name pattern, list	CUSTOMERS.last_name	Substitution from a dictionary
Email Address	Data and name pattern	CUSTOMERS.email	Email masking
Phone	Data and name pattern	CUSTOMERS.phone	Phone masking
Social Security Number	Data and name pattern	CUSTOMERS.ssn	SSN masking
Birth Date	Name pattern	CUSTOMERS.date_of_birth	Random masking with date blurring

2.8.2 The same ideas in Delphix words

Delphix splits the same job differently. Use this table to translate when you move between chapters (Delphix "Discovering your sensitive data", "Managing domains" and "Delphix masking terminology").

Vocabulary translation. Matching rows are close, not identical.

Idea	Informatica	Delphix	GenRocket
Named kind of sensitive data	Data domain	Domain (a category plus the way to secure it, with one default masking algorithm)	Data profile name (a keyword; no domain object is described on the page)
How a column is recognised	Data and metadata patterns on the domain	Classifiers, grouped into profile sets, each tied to a domain	Column-header keywords
Bundle of "what to protect and how"	Policy (with policy packs PII, PHI, PCI)	Profile set plus the domains' default algorithms	Chosen per table in a G-Subset configuration
The change applied to a value	Rule (technique plus parameters)	Algorithm instance (an algorithm framework plus configuration)	Synthetic Data Replacement
The set of tables to work on	Project and plan	Rule set (a group of tables or files)	Tables added to a configuration

THE WORD "RULE" MEANS DIFFERENT THINGS

In Informatica a *rule* changes data. In Delphix a *rule set* is only a list of tables or files that a profiling or masking job runs against (Delphix "Delphix masking terminology"). Mixing these up is the most common vocabulary mistake when someone switches tools.

Domains

Name ↑	Owner	Masking Method	Tokenization Method	Actions
ACCOUNT_NO	System	dlpx-core:CM Alpha-Numeric		...
ACCOUNT_TK	System	NullValueLookup	AccountTK	...
ADDRESS	System	AddrLookup		...
ADDRESS_LINE2	System	AddrLine2Lookup		...
AGE	System	dlpx-core:Age SL		...
BANK_ACCOUNT_NO	System	dlpx-core:CM Numeric		...
BENEFICIARY_NO	System	dlpx-core:CM Alpha-Numeric		...
BIOMETRIC	System	NullValueLookup		...
BLOOD_TYPE	System	dlpx-core:BloodType SL		...
CERTIFICATE_NO	System	dlpx-core:CM Alpha-Numeric		...

Displaying 1 to 10 of 69

Official screenshot: Settings, Domains in Delphix Continuous Compliance. Each row is a domain (ACCOUNT_NO, ADDRESS, AGE ...) with its owner and its default masking method, and the grid shows 10 of 69 domains in this build. This is the "domain equals category plus default algorithm" idea in one picture. [Source page ↗](#)

Two behaviours from the Delphix "Managing domains" page are worth knowing: each domain has a single default masking algorithm, so you need a separate domain for each distinct algorithm you want the profiler to assign, and changing a domain's default algorithm only affects future profiling runs, not assignments that already exist in rule sets.

Both tools use the domain idea for a consistency reason as well. Delphix says the domain mechanism helps ensure the same masking algorithm is applied consistently across rule sets whenever a particular type of sensitive data is discovered. Informatica says all columns in a domain are masked with the same rule, so "SSN" and "SOCIAL_SECURITY" in different tables end up masked the same way (p.73). Consistency across tables is a major theme of the masking concepts chapter.

2.9 Who owns what?

Tools do not decide what is sensitive. People do. Titles differ between companies, but these roles appear in most TDM programmes:

Typical roles around sensitive data (organisation-specific; confirm yours)		
Role	Owns	Typical question they answer
Compliance officer	Which laws and internal rules apply, what "sensitive" means for this company, sign-off on the masking approach.	"Is a masked copy that keeps birth year acceptable under our rules?"
Data steward	Meaning, quality and correct use of a data set. Knows what each column holds.	"Is <code>tax_id</code> a person's ID or a company's?"
TDM engineer	Turning decisions into a working policy, running discovery, masking and subsetting, checking results.	"Which rule fits this column, and did the job keep keys consistent?"
Database administrator	Source and target databases, access, performance.	"Can the TDM tool read this schema, and what is the load window?"
QA engineer / developer	Needs usable test data.	"After masking, do my scenarios still work?"
Security team	Access control and logging for the TDM tool and test environments.	"Who may run a masking job or see the unmasked source?"

The tools reflect the split of duties. In Informatica, you need the *View Policies* and *Manage Policies* privileges to view and work with policies (User Guide 10.5.10, p.77), and the Administrator defines the list of sensitivity levels (p.75). Delphix has roles too: an "Editing a user-defined algorithm" permission is described on its algorithms page, and a separate best-practice page covers defining masking roles.

KEEP A WRITTEN RECORD

A good habit: when the compliance officer and data steward decide how a column is handled, write the decision down next to the column name (a spreadsheet is fine) and reference it from the policy. Both tools can attach descriptions. Informatica also lets you link Informatica Business Glossary terms to policies, domains and rules, so people can look up what a term means from one central place (User Guide 10.5.10, p.82).

2.10 Walkthrough: from a legal requirement to a policy

Here is the whole chain in one story, so the pieces above fit together. This is the shape of a real engagement, simplified.

- 1 Start with the requirement, in business words.**

ShopSphere's compliance officer says: "No real customer identity may leave production, in India or Germany. Card data never leaves the payment zone." The reason to write it in plain words is that everything later is traced back to this sentence.

Expected result: A one-page statement of scope: which systems, which data subjects, which laws.
- 2 Find candidate sensitive columns.**

Run discovery (the discovery and subsetting concepts chapter, then the tool chapters). The tool proposes columns and domains from names, patterns and lists.

Expected result: A candidate list of columns with a proposed domain and a match score or conformance percent.
- 3 Review with the data steward.**

Remove false alarms, add misses, resolve doubtful ones such as `zip_code` and `tax_id`. Informatica's flow lets you approve each suggested domain (User Guide 10.5.10, p.105).

Expected result: An approved list, with a short reason for each yes and no.
- 4 Classify.**

Give each approved column a level (section 2.6) and decide the action for each level: mask, null out, or exclude.

Expected result: A table like the lab answer below, signed off by the compliance officer.
- 5 Choose a protection for each domain.**

Pick techniques that keep data usable for tests (see the masking concepts chapter). In Informatica, attach them as preferred rules and mark one default per domain; in Delphix, choose the default algorithm for each domain.

Expected result: Each domain has one default protection and a written reason.
- 6 Bundle into a policy and store it.**

In Informatica you add the domains to a named policy and assign the policy to a project's data source (p.71). You can export policies, domains and rules to XML files and import them into another TDM repository, so a tested policy can travel from a test setup to the real one (p.80). Importing fails if the object already has assignments, so clear them or rename the file first (pp.80 to 81).

Expected result: A versioned policy and a copy of its exported files.

7

Prove it works.

Run on a small sample, look at the results, and have a second person check. The masking concepts chapter lists what to check.

Expected result: A short report: columns covered, rule used, sample before and after.

8

Repeat on change.

New columns appear whenever the application changes. Delphix notes that a profiling job only covers tables and files already in the rule set, and that newly reachable objects must be added by hand (Delphix "Discovering your sensitive data"). Re-run discovery on every release.

Expected result: A standing step in the release checklist.

2.11 Where test copies leak

Protecting the test database is not enough. Real data also escapes through developer laptops ("quick samples"), bug tickets and screenshots, application logs, backups taken before masking finished, files sent to vendors, free-text notes and the TDM tool's own staging area. A good policy states where unmasked data may exist, for how long and who can reach it. GenRocket's G-Subset documentation shows one design answer: data stays behind the customer's firewall, sensitive columns are never read, and temporary dump files are removed after the run (GenRocket "G-Subset Overview").

Lab: classify the ShopSphere columns

Goal: on paper or in a spreadsheet, decide the sensitivity level and the family (PII, PHI, PCI, none) for each ShopSphere column, and name the signal you would expect a tool to use to find it.

1

List every column of the four tables from the shared schema.

Use CUSTOMERS , ORDERS , ORDER_ITEMS and PRODUCTS .

2

For each column, assign Public, Internal, Confidential or Restricted using the table in section 2.6.

Ask the three-question test from section 2.2.

3

Mark the family and the detection signal.

Is it found by name, by data pattern, by a list, or only by a human?

Expected result: You should end with a list of about nine sensitive columns and at least one that some tools would miss.

4

Flag columns that are indirect identifiers.

Which combinations could single out a customer?

► **Show a worked answer**

2.12 A judgement call: free text

Policies cover the common situations. The interesting work is in the edges.

EXAMPLE: "SUPPORT AGENTS PASTE CUSTOMER DETAILS INTO NOTES"

An `ORDER_NOTES` column (not in the shared schema, added for this example) contains lines like "Customer 123-45-6789 asked for a refund, call 9876543210". No column name flags it, and a column-level data pattern may miss it because the value is a whole sentence. Options: exclude the column from test copies, null it out, or use a free-text redaction algorithm that looks for patterns inside the text. Delphix provides a Free Text Redaction framework for this and says it needs expertise to configure (Delphix "Free Text Redaction (Algorithm frameworks)"). Always check the output with a search for digit patterns afterwards.

2.13 Common mistakes and gotchas

MISTAKES BEGINNERS MAKE

- **"We removed the names, so it is anonymous."** Birth date plus zip plus gender can identify people (section 2.4).
- **Trusting automatic discovery.** It gives candidates. Review them, and add what it misses.
- **Forgetting free-text columns.** Notes, comments and descriptions can hold names and ID numbers that no column name reveals.
- **Ticking "sensitive" and stopping.** In Informatica the flag is only a label. A masking rule must still be assigned (p.89).
- **Assuming a law's definition fits every country.** Ask your compliance officer which regimes apply to your data subjects.
- **Mixing up Delphix "rule set" with Informatica "rule".**
- **Treating PCI as a law.** It is an industry standard, enforced through payment contracts.
- **Copying documentation sample values into tests.** Samples like card numbers can be real by accident.

2.14 Recap

REMEMBER

- Sensitive data is whatever could harm a person or the company if leaked. PII identifies people, PHI ties health facts to a person, PCI covers payment card data.
- Direct identifiers point to one person. Indirect identifiers (zip, birth date, gender) identify when combined, so deleting names is not enough.
- GDPR and HIPAA, in plain words, say: have a lawful purpose, use the minimum necessary, secure it, and de-identify where you can. This is the legal case for subsetting and masking. This is not legal advice.
- Classification gives each column a sensitivity level, and each level maps to an action.
- Tools find sensitive columns by column names, data patterns and value lists, plus type checks. The output is candidates for human review.
- An Informatica policy contains data domains. A domain holds patterns, a sensitivity level and preferred rules. A rule is a technique plus parameters. Delphix uses domains, classifiers, profile sets and algorithm instances for the same ideas.
- People decide: the compliance officer sets what is sensitive, the data steward explains the data, the TDM engineer builds and runs the policy.

Discovery, profiling and subsetting: the concepts

Imagine you have to move a flat. Before you pack, you walk through every room and list what is there: which boxes are fragile, which items belong together, which papers are private. Only then do you decide what to take and how to wrap it. Test data work follows the same order. First you *look inside* the production database (discovery and profiling). Then you *take only what you need* (subsetting). After that, you protect what you took (masking, covered in the masking concepts chapter).

This section covers the two first steps as ideas, without tool buttons. They rely on tables and keys (the data basics chapter) and on what sensitive data is (the sensitive data chapter). Tool detail is in Informatica discovery and profiling, Informatica data subset and Delphix profiling.

2.15 Words to know first

Data discovery

The step of finding out what is in a data source: which tables and columns exist, which hold sensitive data, how tables relate. Informatica uses "data discovery" for the whole activity and "profile" for each job inside it.

Profile (verb: to profile)

A job that scans a data source and reports facts about it. Delphix and GenRocket also say "profiling".

Metadata

Data about data: table names, column names, data types, lengths, key definitions. The column `ssn` and its type "string of 11 characters" are metadata. The value 123-45-6789 is data.

Primary key (PK)

A column (or set of columns) whose values are unique for every row and identify it, such as `customer_id`.

Foreign key (FK)

A column whose values point to a primary key in another table, such as `ORDERS.customer_id` pointing to `CUSTOMERS.customer_id`.

Parent table and child table

The table holding the primary key is the parent; the table holding the foreign key is the child. `CUSTOMERS` is the parent of `ORDERS`, and `ORDERS` is the parent of `ORDER_ITEMS`.

Referential integrity

The promise that every foreign key value points to a row that really exists. A child row with no parent is an *orphan*, and it means the data is broken.

Data subset

A smaller copy of a database that still hangs together: fewer rows, same structure, referential integrity intact.

Driving table (driver table)

The table you start from when you pick a subset. The tool finds related rows in other tables starting from it. Informatica uses exactly this term for the starting table of an entity.

Entity (Informatica)

A driving table plus all the tables related to it. It is the unit that Informatica copies to a subset database.

2.16 Data discovery and profiling

2.16.1 What it is

Data profiling means scanning a data source with software and producing a report of facts about it. *Data discovery* is the wider goal: using those facts to decide what to mask, what to subset, and how tables connect. Informatica puts it plainly: run profiles to discover source data for data masking and data subset operations (User Guide 10.5.10, p.86).

A profile changes nothing: it measures, lists and flags. Unlike a building survey, its flags are guesses that people must confirm.

2.16.2 Why it comes first

Masking and subsetting are only as good as your knowledge of the data.

- You cannot mask a sensitive column you did not know about. Missing one column of SSNs means a leak.
- You cannot subset safely without knowing the relationships. If you cut `ORDERS` without knowing it points at `CUSTOMERS`, you will leave orphans.
- You cannot choose a masking technique without knowing the column's type, length and shape.
- Real databases are rarely documented. Old tables, renamed columns and half-forgotten fields are normal.

WHY IT MATTERS

Both Informatica and Delphix describe profiling as the step that feeds the next one.

Informatica says that before masking you can discover which columns to mask with the same rules, and before subsetting you can discover relationships between tables and the entities for the subset (User Guide 10.5.10, p.86). Delphix says that after connecting data, the next step is discovering which of it should be secured (Delphix "Discovering your sensitive data").

2.16.3 What a profile reports

Typically: each column's data type and length; value patterns and their share (99 percent of `phone` values are ten digits); ranges, minimum, maximum and the share of nulls; the most common values; sensitive-looking columns (`email` matches an Email Address domain); primary key candidates (`customer_id` is 100 percent unique); and foreign key candidates (`ORDERS.customer_id` matches `CUSTOMERS.customer_id`).

2.16.4 The four profile types in Informatica

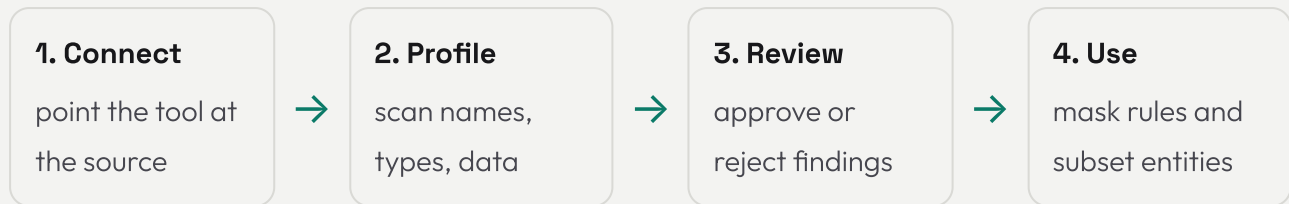
The User Guide lists four kinds of profile (pp.86 to 87). Seeing them side by side shows how one tool splits the work:

Informatica profile types (User Guide 10.5.10, pp.86-87)

Type	What it determines	Feeds
Foreign key (entity)	Relationships between tables, by evaluating table structure for potential links	Subsetting (entities, constraints)
Primary key	Potential primary keys, with the percentage of unique values in a column or combination of columns	Subsetting
Data domain	Which columns belong in a data domain, from data values or from column names	Masking
Column	Characteristics of columns: range of values, list of values, data patterns	Data generation

The guide says you can run data domain and primary key profiles at the same time, and also import profiles created in Informatica Developer. Two small but important facts: TDM adds discovered

keys and relationships as *logical* constraints in its own repository and does not change the source tables (p.86 and p.87), and profiles run on relational sources such as Oracle, Microsoft SQL Server, IBM DB2 for Linux, UNIX and Windows, MySQL, Sybase and Teradata, with data domain profiles also supported on Hive and HDFS (p.87). Check your version for the current source list. A later chapter, Informatica discovery and profiling, walks through the screens.



2.17 Metadata profiling and data profiling

A profile can look at two different things, and the difference decides how fast it is and what it can find.

Metadata profiling

Reads only *descriptions*: table names, column names, types, lengths, declared keys.

- Fast, and does not need to read any real customer values.
- Finds a column called `ssn` , or `user_email` .
- Misses cryptic names such as `FLD_X2` , and cannot tell whether `notes` contains an SSN.

Data profiling

Reads actual *values* (often a sample) and measures patterns, counts and uniqueness.

- Slower, because it reads rows.
- Finds a column of SSN-shaped values even when its name is meaningless.
- Needs access to real production data, which is itself a privacy decision.

The tools expose the same split under different names.

Metadata versus data profiling in the three tools

Tool	How it splits the two	Source
Informatica	A data domain profile can search for patterns in the data only, in the column name only, or in both. You can run the metadata pass first and then run again for the data.	User Guide 10.5.10, pp.95 and 100
Delphix	"Column-level" profiling uses regular expressions on column names (metadata). "Data-level" profiling uses regular expressions on the actual data. In the classifier model, PATH and TYPE work at column level, REGEX and LIST at data level.	Delphix "Introduction to Delphix Masking" and "Discovering your sensitive data"
GenRocket	Data Column Profiling scans column headers only, not the actual data, so GenRocket does not need to read production values.	GenRocket "PII Discovery with Data Column Profiling"

2.17.1 Sampling: a look, not a census

Scanning every row of a table with 80 million rows would be slow, so profiles often sample. *Sampling* means reading only some rows. That is efficient, but it has a catch: a value that appears in only 1 row in 10,000 may not be in the sample. If a few rows of a mostly-clean column hold real phone numbers, a small sample may miss them. Informatica reflects this with options such as "Maximum Rows to Profile" and "Minimum Conformance Percent" (p.95): the second one is the share of profiled rows that must match the pattern before the column is claimed for the domain.

2.18 Finding keys and relationships

Subsetting needs to know which tables are parents and which are children. Ideally the database declares its primary and foreign keys and the tool simply reads them. In practice many databases declare few or none (for performance, or because the application enforces the links). GenRocket's documentation says this directly: custom referential integrity is added "when referential integrity does not exist between tables in the source database" but the user still needs to subset (GenRocket "How to Add Custom Referential Integrity to Imported Table Schemas"). Discovery fills that gap by inferring the links.

2.18.1 Primary key discovery

A primary key candidate is a column, or a small set of columns, whose values are unique. Informatica's primary key profile reports the percentage of unique values and lets you limit how hard it searches (User Guide 10.5.10, pp.92 to 93).

Its options and defaults: the maximum number of columns combined into a key (3), the maximum rows to profile (1,000) and a conformance criterion, the minimum percentage of unique values allowed (99 percent) (p.93).

Why would you allow less than 100 percent? Because real data is messy. If 99.8 percent of values are unique and a few rows are duplicates, the column is probably meant to be a key and the duplicates are data-quality bugs. Informatica notes that a primary key profile is useful to find such non-unique rows (p.92). The result table shows, per candidate, the table, the column or combination of columns, the uniqueness percentage, whether it is verified, any existing constraint and a status of verified, approved or rejected (pp.103 to 104). When you approve a candidate, TDM adds it as a key to the table in its repository, not in the source database. Rejecting a candidate does not affect the subset (p.104).

UNIQUE DOES NOT MEAN A GOOD KEY

`ssn` is 100 percent unique in the table above, yet it would be a poor primary key: it is sensitive, so masking will change it, and every table that used it as a link would need identical treatment. A human picks `customer_id`, the "meaningless but stable" identifier. Uniqueness tells you what *can* be a key. A person decides what *should* be.

2.18.2 Foreign key and entity discovery

To find parent-child links, Informatica's entity profile "returns a list of possible entities by comparing column names in the source tables" (p.93). In the guide's example, `Employee`, `Employee_Detail` and `Employee_Address` all contain an `EmpID` column, so the profile suggests `Employee` as an entity with two child tables. When you later subset, a filter on `EmpID` applies to every table in that relationship.

The profile options show what matters when matching columns (p.94): whether to compare using the datatypes the tool inferred or the documented ones, whether comparison is case sensitive, whether to trim spaces first, the maximum number of foreign key results to return and the minimum confidence percent (the share of foreign key values that match). A coincidental match is the risk: two unrelated integer columns can overlap by chance, so discovery proposes and a person approves.

One practical detail: if you choose the entity profile type and the physical database has no primary keys, the profile returns nothing. Run the profile as primary key and entity together, and it can discover both the primary and foreign keys even when the database declares none (p.93).

Delphix says its masking keeps referential integrity by scanning metadata and data to identify and preserve the primary and foreign key relationships between elements, so that data is masked the same way across tables and databases (Delphix "Introduction to Delphix Masking"). How keys are defined in a Delphix rule set is covered in the Delphix rule sets and keys chapter.

2.19 Reading the results: false positives, false negatives and human review

A profile is a set of educated guesses. Every guess can be wrong in two ways:

The two kinds of mistake in discovery			
Mistake	Meaning	ShopSphere example	Cost
False positive	Flagged, but not actually what the tool thinks	<code>order_id</code> flagged as a phone number because it has ten digits	Wasted effort, or a harmful change (a key gets masked)
False negative	Real, but not flagged	A <code>contact_no</code> column of real phone numbers not matched by any pattern	Real data leaks into test

Between the two, false negatives are the more dangerous in a privacy project, because nobody notices them. That is why discovery tools build review into the workflow. In Informatica, each result row has a status, and you can verify, approve or reject it. The User Guide says the Verified column is for your own tracking and does not affect data subset operations, and that marking the data domain classification as completed does not affect any process either; it simply records that you reviewed everything (pp.104 to 105). The data viewer for a selected column shows the first 200 records, so reviewers can look at real values before approving (p.105).

2.19.1 A review routine that works

1

Sort the results by confidence.

Informatica shows % data conformance and whether the column name matched (p.105). High on both is easy to accept.

Expected result: A short list of obvious accepts and a longer list that needs thought.

2

Look at the actual values for doubtful rows.

Names alone mislead. Five sample values usually settle it.

3

Ask the data steward about anything ambiguous.

"Is `tax_id` a personal or a company number?" is not something a tool can answer.

4

Scan the unflagged columns too, and record your decisions.

Check free-text and generically named columns. A profile only covers the tables it was given, and re-profiling will not remember your reasons.

Expected result: A short list of columns added by hand, and a written list of accepted and rejected findings.

2.19.2 The fourth type: the column profile

Informatica's **column profile** measures range, list of values and patterns of columns, mainly to infer rules for generating test data; it runs on Oracle and Microsoft SQL Server databases (p.97). It returns in the test data classification and generation chapter.

Lab: profile three mystery columns by hand

Goal: practise what a profile computes. The columns come from a ShopSphere table whose column names were lost. Compute for each: the share of nulls, the number of distinct values, the pattern, and your best guess of the data domain.

FIVE ROWS FROM A TABLE WITH CRYPTIC COLUMN NAMES

row	col_a	col_b	col_c
1	9876543210	123-45-6789	1001
2	9123456780	234-56-7890	1002
3	9988776655	N/A	1003
4	(empty)	345-67-8901	1004
5	9876543210	123-45-6789	1005

1

Count nulls for each column.

Treat "(empty)" as null. Treat "N/A" as a value, because it is text.

2

Count distinct values.

Divide by the number of non-null values to get a uniqueness percentage (tools may compute this slightly differently, but the idea is the same).

3

Describe the pattern.

Use digits as 9, and spell out hyphens.

4

Guess the domain and the next action.

Which column is a primary key candidate? Which is sensitive? Which needs a human decision?

► Show a worked answer

2.20 What is data subsetting?

Start with the problem. ShopSphere's production database holds eight years of history: tens of millions of customers, hundreds of millions of orders, terabytes in total. A tester working on the checkout screen does not need all of it. Copying everything to every test environment is slow, expensive and risky. *Data subsetting* means copying only a *slice* of the database, small enough to be cheap, and consistent enough to be useful.

Informatica describes the goal as a "small, targeted, and referentially intact copy" of production data for use in a nonproduction environment such as development, test or training, and gives the example of financial data for a specific region or time period (User Guide 10.5.10, p.112). GenRocket frames it the same way: production data may contain sensitive data, "you also do not need all that data for application testing, only a subset" (GenRocket "Synthetic Data Masking (SDM) and Data Subsetting Overview").

2.20.1 Why teams subset

- **Size and storage cost:** smaller copies need less disk and money (GenRocket lists saved storage as a benefit).
- **Speed:** a small copy refreshes overnight instead of over days, and tests and reports run faster.

- **Privacy:** fewer real records copied means less exposure and less to mask (the minimum necessary idea from the sensitive data chapter).
- **Focus:** you can pick exactly what a test needs, such as one region or one scenario.

2.20.2 A subset is not just "the first 1,000 rows"

Beginners sometimes subset by taking the first N rows of each table. That breaks the data: the first 1,000 orders belong to customers that may not be in the first 1,000 customers. The result is orphans: orders with no customer, items with no order. A real subset starts from a choice (a set of customers, a time window) and then follows the relationships so that every kept row has all the parents it points to. Everything in the next sections is about doing that correctly.

2.21 Types of subset

"Type" here means *how you choose which rows to keep*. Several can be combined. The table lists the kinds the documentation supports, plus two general concepts.

Ways to choose a subset

Type	How the rows are chosen	Informatica	GenRocket
By filter condition	Keep rows that satisfy a condition on a column (zip code, status, region).	Entity filter criteria on columns of the driving table, using values or parameters; several criteria combine with AND (pp.116 to 117).	Where clause; AND or OR allowed; example <code>address.state = 'CA'</code> (Subsetting how-to).
By percentage of rows	Keep a given share of the rows, such as 25 or 50 percent.	Limit by percentage in the plan (pp.294, 304).	"% of rows", usable alone or with a Where clause.
By absolute number of rows	Keep a fixed number of rows, such as 100.	Limit by absolute value (pp.294, 304).	"# of rows".
By interval of rows	Keep a defined interval of rows.	Limit by an interval of rows (pp.294, 304).	Not described; the overview example starts at customer id 251 and keeps 100 rows.
By time window	A filter on a date column: last 90 days, a financial year.	The guide's example subset is for "a specific region or time period" (p.112); its database-function example uses a date test, <code>HIREDATE > add_months(current_date, -3)</code> (p.119).	Possible through a Where clause on a date column; the documentation examples use id and state filters.

Type	How the rows are chosen	Informatica	GenRocket
By entity or driver table	Pick rows in one table and bring along everything related.	Entity with a driving table; example: all data for EmplID 10 (p.121).	The table that holds the subset conditions drives the rest; related rows come with it.
Whole tables copied in full	Some tables (lookups, products) are copied entirely.	A group with residual tables, to copy unfiltered data (p.116).	Full-load tables (How to Select Tables for Full Migration).
Random sample	Rows picked by chance so the subset resembles the whole.	Not documented whether a percentage limit picks rows at random.	Not described as random.
Targeted sample	Rows picked on purpose to cover a scenario.	Done with filters on entity criteria.	Done with the Where clause.

A percentage limit is not automatically a random sample. "Keep 25 percent" says how many rows, not which; the documentation does not say how rows are picked. GenRocket's own example takes a contiguous block (start at customer 251, take 100). Check the tool before relying on representativeness.

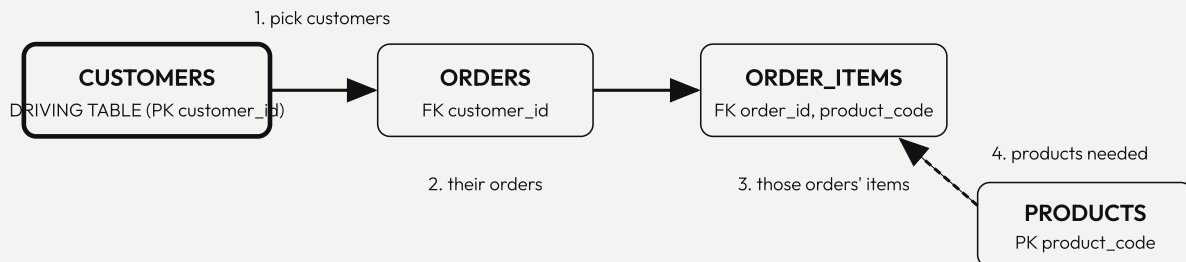
In short: a random sample resembles the whole (realistic averages) but may miss rare cases; a targeted subset covers chosen scenarios but is unrepresentative statistically. Most teams combine them. For cases production lacks entirely, generate data instead (the test data classification and generation chapter).

Informatica filters can use a *criteria parameter* instead of a fixed value, so one entity and workflow can produce different subsets by changing the parameter file (for example EMPID greater than 100, then 500) (User Guide 10.5.10, p.118).

2.22 Driver tables and child rows

Now the core mechanic. Every subset has a **starting point**: a table and a condition. The tool picks the matching rows there, then *follows the relationships* to select related rows in other tables.

Solid arrows: parent to child rows pulled down. Dashed arrow: a parent row the kept items point to.



Original diagram: a subset driven from CUSTOMERS. Child rows follow downward; parent rows that kept rows depend on (the products) are pulled in as well.

2.22.1 Informatica: the entity and its driving table

In Informatica, an **entity** "consists of a driving table and related tables". The driving table is the starting point; TDM finds the tables related to it from physical (declared) and logical (discovered or added) constraints, and adds them to the entity by default. You then define filter criteria on columns of tables in the entity (User Guide 10.5.10, p.114). The guide's example: Employee as the driving table, with primary key EmpID, three child tables holding an EmpID foreign key, and a fourth table, BankAccounts, linked through an account number from one of the children (p.115). The Entity Map shows the star on the driving table.

The Entity Map can show all tables, the parents or children of a selected table, or cyclic relationships, and you can disable a relationship; a table left unrelated is excluded (p.115). The guide advises running data discovery before creating an entity so the constraints exist (p.116).

THE GUIDE'S OWN SMALL EXAMPLE

An employee has ID 10 and you want everything about that employee. Employee is the driving table; Employee_History and Employee_Order are related to it by EmpID. Define the criteria "EmpID equals 10". After the plan runs, the target database holds one Employee row for EmpID 10, all of that employee's history rows and all of their orders (p.121).

2.22.2 GenRocket: the subset table and the hierarchy

In GenRocket's G-Subset, only *one* table in a configuration can have subsetting conditions (Where clause and/or row limit). Rows that meet the condition are included, "all records in related tables associated with those meeting the Where Clause condition will be inserted as well" (Subsetting how-to). The Traverse Full Hierarchy page shows what "related" means. Its example database has Organization (1 row), Department (4), Employee (120), Address, Attendance, Payroll and Qualification (120 each) and Project (8). The condition is on Payroll: `hours > 10`, limited to 10 rows.

- **Default behaviour:** G-Subset goes *up* from the subset table to the top-most parent to build its queries, then migrates from the top down: Organization, Department, Employee, Payroll. Only the ancestor rows related to the chosen Payroll rows move: 10 employees, the departments they belong to, and so on.
- **Traverse Full Hierarchy enabled:** it also brings the *other children* of those ancestors: Project (2 rows), and Address, Attendance and Qualification rows for the same 10 employees.

The same page notes that the option is only available when a subsetting condition has been added to a table, and that masking rules you set for those extra tables are applied to them too.

CHOOSE THE DRIVING TABLE BY THE TEST QUESTION

If the test is "does the customer page show the right orders?", drive from `CUSTOMERS`. If the test is "does the monthly report add up?", drive from `ORDERS` with a date filter and let the parents (customers) and children (items) follow. The table you drive from decides which rows are complete and which are partial.

2.23 Keeping referential integrity

Referential integrity is the rule that every foreign key value points to a row that exists. It is the one property a subset must never lose. If it breaks, the application crashes or shows empty pages, and testers blame the data. GenRocket's page defines it in the same plain terms: it "refers to the accuracy and consistency of the defined relationships" between parent and child tables (GenRocket "How to Add Custom Referential Integrity to Imported Table Schemas").

2.23.1 How a naive subset breaks it

Suppose someone takes the first 6 rows of `ORDERS` and the first 4 rows of `CUSTOMERS` "to keep it small". Order 5006 belongs to customer 1005, who is the fifth customer and was left out. Order 5006 is now an orphan.

```
SELECT o.order_id, o.customer_id
FROM   ORDERS o
LEFT JOIN CUSTOMERS c ON c.customer_id = o.customer_id
WHERE  c.customer_id IS NULL;      -- any row returned is an orphan order
```

On the naive subset this query returns one row (order 5006, customer 1005). On a correct subset it returns nothing. The same check works one level down (`ORDER_ITEMS` to `ORDERS` , and `ORDER_ITEMS` to `PRODUCTS`). Run these checks after every subset: they are cheap and catch the commonest failure.

2.23.2 Integrity options in Informatica plans

Informatica goes beyond "no orphans". It also offers several ways to decide *how much* extra data to pull to keep the subset coherent. You choose the option in the plan's advanced settings, so the same entity can yield different subsets (User Guide 10.5.10, pp.121 to 123). By default all constraint relationships in an entity are "major", which can pull in excess data; you can optimise relations so that the entity has the minimum number of major relations while still covering all tables (p.119).

Informatica data integrity options for a subset plan (User Guide 10.5.10, pp.121-123), explained in plain words

Option	What it does	Size of result	Notes from the guide
Transactional Integrity for Major Relation Type	A selected child row pulls in its parents; those parents pull in all of their children; those children may point to more parents, and so on until nothing new is added.	Can include extra records	Maintains transactional and referential integrity. Criteria apply to the entity as a whole; multiple criteria combine with AND, which can return nothing if they select unrelated tables.
Referential Integrity Only	Selects the fewest rows needed to keep every foreign key valid.	Fewer rows	May not include all the children of a parent, so transactional integrity can break. Falls back to the transactional option if the entity has more than one criterion.
Two-way	Pushes the criteria down, then back up, alternately, until every table in the entity has some data.	More than the first two	Keeps referential integrity and ensures every table meeting the criteria has data. Requires a staging connection.
One-way	Applies criteria on individual tables and pushes them downward only; tables above the top-most filtered table are included in full.	More than Two-way	Aims for the smallest subset that does not break referential integrity while including all data from tables above the filtered one.

The guide also says cyclic and self-referencing relationships (a table that points to itself, such as employees pointing to managers) need a staging connection for recursive evaluation (pp.114 and 122).

2.24 Worked example: a ShopSphere subset, before and after

Requirement from the test lead: "Give me the Bengaluru customers (zip code starting with 560), with all of their orders, items and the products those items use." The driving table is `CUSTOMERS` ; the filter is `zip_code` starting with 560. Only the columns needed are shown, and customers 1004 to 1008 are extra example rows (1001 to 1003 are the shared sample rows).

2.24.1 Before: the full source

CUSTOMERS (8 ROWS) WITH THE FILTER RESULT

customer_id	first_name	last_name	zip_code	In subset?
1001	Asha	Menon	560001	Yes
1002	Ravi	Kumar	600017	No
1003	Meera	Nair	682001	No
1004	Kiran	Rao	560034	Yes
1005	Divya	Iyer	600042	No
1006	Arjun	Das	700001	No
1007	Neha	Shah	560001	Yes
1008	Sameer	Khan	400001	No

ORDERS (12 ROWS): KEPT BECAUSE THEIR CUSTOMER WAS KEPT

order_id	customer_id	order_date	total_amount	In subset?
5001	1001	2026-01-14	1200.00	Yes
5002	1002	2026-01-20	450.00	No
5003	1001	2026-02-02	890.00	Yes
5004	1003	2026-02-11	150.00	No
5005	1004	2026-02-15	2300.00	Yes
5006	1005	2026-03-01	99.00	No
5007	1007	2026-03-05	700.00	Yes
5008	1006	2026-03-09	595.00	No
5009	1004	2026-03-12	99.00	Yes
5010	1008	2026-03-20	300.00	No
5011	1002	2026-04-02	550.00	No
5012	1001	2026-04-10	99.00	Yes

ORDER_ITEMS (15 ROWS): KEPT BECAUSE THEIR ORDER WAS KEPT

item_id	order_id	product_code	quantity	unit_price	In subset?
9001	5001	P-100	1	1200.00	Yes
9002	5002	P-200	3	150.00	No
9003	5003	P-300	2	445.00	Yes
9004	5004	P-200	1	150.00	No
9005	5005	P-100	1	1200.00	Yes
9006	5005	P-400	2	550.00	Yes
9007	5006	P-500	1	99.00	No
9008	5007	P-400	1	550.00	Yes
9009	5007	P-200	1	150.00	Yes
9010	5008	P-300	1	445.00	No
9011	5008	P-200	1	150.00	No
9012	5009	P-500	1	99.00	Yes
9013	5010	P-200	2	150.00	No
9014	5011	P-400	1	550.00	No
9015	5012	P-500	1	99.00	Yes

PRODUCTS (7 ROWS): KEPT ONLY IF A KEPT ITEM USES THEM

product_code	product_name	price	In subset?
P-100	Headphones	1200.00	Yes (items 9001, 9005)
P-200	USB cable	150.00	Yes (item 9009)
P-300	Desk lamp	445.00	Yes (item 9003)
P-400	Keyboard	550.00	Yes (items 9006, 9008)
P-500	Mouse pad	99.00	Yes (items 9012, 9015)
P-600	Webcam	2100.00	No
P-700	Monitor arm	1800.00	No

2.24.2 Step by step: how the tool arrives at the result

1

Apply the filter to the driving table.

`CUSTOMERS` where zip_code starts with 560.

Expected result: Customers 1001, 1004 and 1007 (3 rows).

2

Follow the relationship down to orders.

Keep every order whose customer_id is in the kept set.

Expected result: Orders 5001, 5003, 5012 (customer 1001), 5005, 5009 (customer 1004) and 5007 (customer 1007): 6 rows.

3

Follow the relationship down to order items.

Keep items whose order_id is in the kept orders.

Expected result: Items 9001, 9003, 9015, 9005, 9006, 9012, 9008, 9009: 8 rows.

4

Follow the items up to products.

Items point at products, and a kept item must not point at a missing product.

Expected result: P-100 to P-500 are kept (5 rows). P-600 and P-700 are not needed.

5

Check for orphans.

Run the left-join checks from section 2.23 on each parent-child pair.

Expected result: Zero orphans.

2.24.3 After: the subset

SUBSET RESULT: ROW COUNTS AND IDS, BEFORE TO AFTER

Table	Before	After	Rows kept
CUSTOMERS	8	3	1001, 1004, 1007
ORDERS	12	6	5001, 5003, 5005, 5007, 5009, 5012
ORDER_ITEMS	15	8	9001, 9003, 9005, 9006, 9008, 9009, 9012, 9015
PRODUCTS	7	5	P-100, P-200, P-300, P-400, P-500
Total	42	22	about half of the rows in this toy example

TOY NUMBERS, REAL PRINCIPLE

Half is a toy ratio with eight customers; one city of a real ShopSphere might keep 2 to 3 percent. Small shared tables such as the product catalogue are usually copied in full: Informatica uses a group with residual tables to copy unfiltered data (p.116), GenRocket uses full-load tables.

Lab: subset by a time window instead

Goal: repeat the exercise driving from `ORDERS` with the filter "order_date in March 2026", using the tables above. You will see how the choice of driving table changes the result.

1

Filter the driving table.

List the orders whose order_date falls in March 2026.

2

Pull the parents.

Which customers do those orders point to? They must be in the subset or the orders become orphans.

3**Pull the children.**

List the items belonging to the kept orders.

4**Pull the products those items use.****5****Count rows per table and compare with the zip code subset.****6****Apply transactional integrity.**

For each kept customer, also keep all of that customer's other orders (and their items and products). What is added?

► **Show a worked answer**

2.25 Reduced copies and full copies

A subset is one way to make a test copy small. There are other ways to get a test copy, and each has a place. The comparison below is a general guide, with the Delphix statements marked.

A full physical copy is the most realistic but the slowest and most exposed. A subset is small and quick to refresh but may miss rare cases. A virtual copy looks full but shares storage with its source, so it is cheap to create, but it still holds all the data and needs masking.

Delphix describes virtualization as producing "a functioning database or filesystem copy that is lightweight and ephemeral" (terminology page), and its introduction claims that ten virtual copies can fit into the space of one physical copy. The Delphix pages describe only simple row filters, not key-following subsets, so treat "virtual copy" and "subset" as two different answers to the same cost problem: Delphix makes full copies cheap; Informatica and GenRocket make copies small.

2.26 How masking and subsetting combine

You rarely do one without the other: subsetting shrinks the set of real records, and masking changes what remains. Three orders are possible.

Possible orders of subsetting and masking

Order	How it works	Good	Watch out
Subset, then mask	Select the slice from real data, then mask the slice	Less to mask, and filters see real values (for example real zip codes)	Unmasked data exists briefly in the slice or staging area
Mask, then subset	Mask everything, then select the slice from the masked copy	Nothing unmasked leaves the masking step	Masks the whole database (slow), and filters on masked columns pick different rows than expected
Both in one run	Select and mask while copying to the target	Fast, one job, one schedule	Needs a tool that supports it, and careful setup

Both Informatica and GenRocket support the third pattern. In Informatica, a plan can contain masking components (policies and rules) and subset components (entities, groups) together (User Guide 10.5.10, p.304; and p.135 for masking plans). The plan also says whether to mask data in place in the source database or in stream into a target database (p.150). In GenRocket, "Mask and subset data set together by providing both the masking and subset parameters in a G-Subset Configuration" (G-Subset Overview). Its sensitive columns are replaced with synthetic values in a temporary file before the rows are inserted, and the sensitive values are never read.

The key requirement when you combine them is **consistency**. The masked customer 1004 in **CUSTOMERS** must be the same masked customer 1004 that the kept orders point to. Informatica's docs describe "auto cascades" that mask a parent key and its child foreign keys with the same value (p.91), and Delphix says its masking keeps primary and foreign keys consistent across tables and databases (Delphix "Introduction to Delphix Masking"). The masking concepts chapter explains how.

THE ZIP CODE SUBSET FROM SECTION 2.24, WITH NAMES MASKED (MASKED VALUES ARE EXAMPLES)

customer_id	first_name (before)	last_name (before)	first_name (after)	last_name (after)	zip_code
1001	Asha	Menon	Leela	Pillai	560001
1004	Kiran	Rao	Mohan	Verma	560034
1007	Neha	Shah	Tara	Bose	560001

Notice that the key `customer_id` is unchanged, so the 6 kept orders still point at the right rows. Notice too that zip codes were left alone here. As the sensitive data chapter showed, a zip code is an indirect identifier, so whether it stays is a decision for the compliance officer.

2.26.1 GenRocket's subsetting at a glance

GenRocket's database subsetting is called G-Subset (formerly G-Migration+). It plans from the source schema, runs in the customer's environment and migrates to a destination database of the same type. Conditions are a Where clause, a percentage or a number of rows, on one table only; related rows follow, other tables can be marked for full load, and custom referential integrity can be added when the source declares none. It does not support composite keys at present (GenRocket "G-Subset Overview", source page 7).

2.27 Common mistakes and gotchas

MISTAKES BEGINNERS MAKE

- **Taking "the first N rows" of every table.** It creates orphans. Always start from a driving table and follow relationships.
- **Trusting discovery output without review.** Reject coincidental key matches, add missed sensitive columns.
- **Choosing a sensitive column as a key.** A unique SSN is a poor primary key because masking will change it.
- **Assuming a percentage limit is random.** The documentation does not say so; verify.
- **Forgetting reference tables.** A subset with an empty `PRODUCTS` table breaks every order screen. Copy lookup tables in full.
- **Filtering on a column you also mask.** Decide whether the filter sees real or masked values.

2.28 Recap

REMEMBER

- Discovery and profiling come first: they report types, patterns, sensitive-looking columns, key candidates and relationships, from metadata (names and types) and from data (values, usually sampled).
- Profiles guess. False positives waste effort or change keys; false negatives leak data. A human reviews and approves.
- A subset is a small, targeted, referentially intact copy: cheaper to store, faster to refresh and test, and less real data exposed.
- You choose rows by filter condition, percentage, row count or interval, time window, or by a driving table; tools also copy some tables in full.
- A driving table is the starting point: related rows are followed down to children and up to parents so there are no orphans. Informatica calls the group an entity; GenRocket follows the hierarchy from the subset table.
- Informatica offers transactional, referential-only, two-way and one-way integrity options that trade subset size against completeness.
- Subsetting and masking are partners. Keep keys consistent when you combine them, and check for orphans afterwards.

Data masking: the concepts

After the sensitive columns are found (the sensitive data chapter) and the data is cut down to a slice (the discovery and subsetting concepts chapter), **masking** makes the copy safe to hand to testers by replacing real sensitive values with realistic fake ones. Tool detail is in Informatica data masking and Delphix algorithms and masking jobs.

2.29 Words to know first

Masking

Replacing sensitive values with realistic but fictitious values so test data cannot identify real people. Delphix defines it as iterating through a dataset to identify all sensitive fields and replace them with desensitised values to eliminate risk in lower environments.

Technique (Informatica) / algorithm (Delphix)

The method that changes a value, such as substitution or date shift. Informatica turns a technique plus settings into a *rule*. Delphix turns an algorithm framework plus settings into an *algorithm instance*.

Deterministic

Always gives the same output for the same input (and same settings). The opposite is *non-deterministic* or random: a different output each time.

Repeatable

Informatica's word for deterministic output across runs and tables, switched on with a seed.

Seed / key

A starting value that controls the output of a deterministic technique. Same seed (Informatica) or same algorithm key (Delphix) gives the same masked values; changing it changes them.

Dictionary / lookup file

A list of replacement values, such as thousands of first names, that substitution picks from.

Collision

Two different inputs masked to the same output. Fine for names, harmful for a column that must be unique.

Irreversible

You cannot get the original value back from the masked one.

Format-preserving

The masked value keeps the original's shape: a 10-digit phone stays 10 digits, a card number keeps its checksum.

In-place and on-the-fly

In-place masking changes the data where it sits. On-the-fly masking reads unmasked data and writes masked data to a different place.

2.30 Masking and its neighbours

People use these words loosely. They mean different things, and the difference decides whether a copy is safe.

Obfuscation, masking, encryption, tokenization, anonymisation, pseudonymisation

Term	What it means	Reversible?	Keeps data usable for tests?	ShopSphere example
Obfuscation	Umbrella word: make data hard to read or understand, by any method. Masking is one kind.	Depends	Depends	Any of the rows below
Masking	Replace real values with realistic fake ones, usually permanently, for non-production use.	No (by design)	Yes	Asha Menon becomes Leela Pillai
Encryption	Scramble data with a key so only key holders can unscramble it.	Yes, with the key	Only if it preserves format; ordinary ciphertext breaks tests	123-45-6789 becomes q8Zk...=
Tokenization	Replace a value with a token, keeping a protected way to reverse it later.	Yes, by design	Partly	Send tokens to a vendor; map back afterwards

Term	What it means	Reversible?	Keeps data usable for tests?	ShopSphere example
Anonymisation	A <i>goal</i> : data that no longer identifies anyone, even in combination. Legal tests are strict.	No	Varies	Masked and generalised so no row is unique
Pseudonymisation	Replace identifiers with pseudonyms, but a way back exists elsewhere. Still personal data under GDPR.	Yes, with the extra information	Yes	customer_id replaced by a token

How the sources use the words:

- Delphix says *tokenization* is "the only type of algorithm that allows you to reverse its masking". Its example is masking data before sending it to a vendor and reversing it afterwards to act on the vendor's findings (terminology page). Its introduction labels masking as anonymisation and tokenization as pseudonymisation. It also says unlike encryption, which can be bypassed by obtaining user credentials, masking irreversibly protects data in downstream environments, and that encryption is ill-suited to protecting non-production environments (Delphix "Introduction to Delphix Masking").
- Informatica treats *encryption* as one masking technique. Standard encryption offers AES, CRC and MD5, and format preserving encryption can be decrypted with the same pass phrase and configuration (User Guide 10.5.10, p.161).
- GenRocket argues that traditional masking "can be reverse engineered" and that its Synthetic Data Replacement generates new values that cannot be traced back (GenRocket "G-Subset Overview" and "What is Synthetic Data Replacement").

A third option: replace instead of obscure. In the Q&A of its 2020 webinar, GenRocket's CTO was asked how to mask sensitive data when moving production to staging. He said the

sensitive attributes are marked in the domain, and during the move their values are never touched, so nothing is obfuscated; they are replaced with synthetic values. His argument was that this is fully secure because the real value is never handled, and that production values may not be valid anyway, so patterns give testers control. This is one vendor's approach with a trade-off: the replacements lose the realism of real data. It suits tests that depend on a value's format, such as a valid-looking SSN, and does not suit tests that depend on real distributions, such as how often a postcode occurs. The vendor's documentation calls it Synthetic Data Replacement (see the list above).

REVERSIBLE IS A RISK IN TEST DATA

If a test copy can be turned back into the real data, then stealing the copy plus the key or mapping table is a breach. For ordinary test environments, prefer irreversible masking. Use encryption or tokenization when you truly need to come back, such as a vendor round trip, and guard the key like production data.

ANONYMISATION IS A DESTINATION, MASKING IS A VEHICLE

Masking is a set of techniques. Whether the masked result counts as anonymous under a law depends on the whole data set (the combinations you saw in the sensitive data chapter) and is the compliance officer's call.

2.31 What good masked data looks like

A tool could blank every sensitive column in a second. That would be safe and useless. Masking has competing goals, and good masking meets all of them.

The goals of masking

Goal	Meaning	What goes wrong without it
Safe	Real values cannot be recovered or inferred	A leak of test data is a leak of customers
Realistic	Looks like real data: names look like names, emails like emails	Testers cannot spot UI bugs, and validation rules reject the data
Valid	Passes the application's rules: format, checksum, length, ranges	The card form refuses every masked card number
Consistent	The same real value maps to the same fake value everywhere it appears	Joins break; a customer has different names in two tables
Repeatable	A re-run gives the same results when it should	Test cases that rely on a specific customer fail after each refresh
Fit for purpose	Keeps the properties the tests depend on (a customer has several orders, a date precedes another)	The data is fake but the test no longer tests anything

Both vendors state these goals. Informatica: replace source data in sensitive columns with "realistic test data" for nonproduction environments (p.135). Delphix: the data stays "production-like in quality", masking is consistent while "maintaining referential integrity across heterogeneous data sources", and it is irreversible (Delphix "Introduction to Delphix Masking").

Some goals pull against each other. Perfect safety says "change everything completely". Perfect fidelity says "keep it as it was". Every technique in the next section is a compromise somewhere between those poles, which is why choosing is a skill.

2.32 Common masking techniques

Each technique below is shown with a before and after on ShopSphere data. Masked values are examples unless marked as taken from the documentation. Names differ between tools,

so the table maps them.

The techniques and what each tool calls them		
Technique	Informatica (chapter 8)	Delphix
Substitution	Substitution	Secure Lookup
Shuffle	Shuffle	SecureShuffle
Key masking	Key	Character Mapping, Segment Mapping, Mapping (unique outputs)
Encryption	Encryption (standard and format preserving)	Tokenization (reversible)
Nulling, redaction	Nullification	Free Text Redaction, null-value lookup
Date shifting	Random with date blurring	Date Shift, Dependent Date Shift
Numeric variance	Random with numeric blurring or range	Min Max
Format-preserving	Credit card, Phone, SSN, email, IP, URL masking; format preserving encryption	Payment Card, Character Mapping
Character replacement	Mask format, source and result string characters	Character Replacement
Algorithm chaining	Advanced masking (variable columns), mapplets	String Algorithm Chain
Partial redaction ("show last 4")	Key masking with "+" (no mask) positions	Preserve ranges in Character Mapping

2.32.1 Substitution

Replace the value with a different, believable one picked from a list. Informatica: it "replaces a column of data with similar but unrelated data from a dictionary", which can be a flat file or a database table (p.176). Delphix Secure Lookup is the same idea: it picks a replacement from a lookup file using a cryptographic hash of the original value, so the output is consistent for each input (terminology page; the framework page says the default hash method is SHA256).

SUBSTITUTION ON CUSTOMERS (EXAMPLE MASKED VALUES)

customer_id	first_name before	after	last_name before	after
1001	Asha	Leela	Menon	Pillai
1002	Ravi	Sanjay	Kumar	Gupta
1003	Meera	Anita	Nair	Joshi

Strengths: very realistic. Limits: the list can run out. Informatica's *unique substitution* makes the operation fail when the source has more distinct values than the dictionary has (p.176), and Delphix says that two inputs can collide on one output, which is fine for names and a problem for unique columns (terminology page). Informatica also lets you pick names by a lookup column such as gender or country, so female names stay female (pp.177 to 179).

2.32.2 Shuffle

Move the existing values of a column between rows. Informatica's own example moves first names between four rows: Tom, Sue, Bob, Eli become Bob, Eli, Tom, Sue (p.173). Delphix SecureShuffle guarantees that each value moves to a different row, but an input can still mask to itself when values repeat (SecureShuffle page).

SHUFFLE OF LAST_NAME

customer_id	last_name before	after
1001	Menon	Kumar
1002	Kumar	Nair
1003	Nair	Menon

The catch: the values are still real. Someone who knows Asha Menon's first name and city could still recognise "Menon" in the column, and a rare value stays rare. Delphix warns that shuffling does not redact or modify any value, so consider carefully whether it meets your security requirements. Informatica offers a *constrained* shuffle (for example, only within the same city) and a *representative* shuffle, in which all rows with the same source value get the same shuffled value (p.174).

2.32.3 Key masking

Produce a deterministic masked value from the input and a seed: same input, same seed, same output. Informatica's Key technique works for dates, numbers and strings, and you can limit each output character to letters, digits or alphanumerics with a mask format (pp.164 to 168). For numbers, the guide says to use the same seed in the primary key column and the foreign key column so referential integrity is maintained (p.167). Delphix does the same job with Character Mapping, which maps each value to another value built from the same character groups, giving no two inputs the same output, which suits primary and foreign key columns (Character Mapping page).

NUMERIC KEY MASKING WITH THE SAME SEED EVERYWHERE (EXAMPLE MASKED VALUES)

Column	Before	After
CUSTOMERS.customer_id	1001	48213
ORDERS.customer_id (for Asha's orders)	1001	48213

2.32.4 Encryption and tokenization

Informatica offers standard encryption (AES, CRC, MD5, with a key of 16 characters or fewer) and *format preserving encryption*. The latter can keep the format and length, and can be decrypted with the same pass phrase and configuration. Its example: `Abc123@xyz.com` becomes `Mpz849#dje!kuw`, or `Mpz849@dje.kuw` if the "@" and "." are excluded from encryption (pp.161 to 162). The Delphix Tokenization framework uses AES-128 and produces longer alphanumeric tokens, built for the vendor round trip described in 6.2 (Tokenization page). Because AES and format preserving encryption can be reversed, use them only for that.

2.32.5 Nulling and redaction

Remove the value, or part of it. Informatica's Nullification replaces a column with NULL, works on all data types and has no parameters (p.168). The simplest safe answer for a column

the tests do not use, such as `tax_id` , is NULL. The cost: NOT NULL columns refuse it and the application paths that read the column go untested. **Free text** needs smarter redaction: Delphix's Free Text Redaction finds sensitive words and patterns inside a text column (for example "St", "Blvd", or the shape 123-45-6789 for an SSN) and redacts the whole matched word, but it "requires some expertise to use" (Free Text Redaction page).

2.32.6 Date shifting and blurring

Move a date by an amount so that the exact date is hidden but the date still looks plausible. Delphix Date Shift masks each date to another within a configured range (for example 3 to 5 days later), based on the algorithm key; a fixed shift (same minimum and maximum) preserves uniqueness and referential integrity, unlike variable ranges. Its docs show `2021-02-03 12:30:00` becoming `2021-02-06 12:30:00` (Date Shift page). Informatica blurs a date within a range of years, months, days or hours: a source of 2 February 2006 with a two-year bound becomes a date between 2 February 2004 and 2 February 2008 (p.170).

DATE SHIFT OF DATE_OF_BIRTH, FIXED SHIFT OF +3 DAYS (EXAMPLE)		
customer_id	Before	After
1001	1990-04-12	1990-04-15
1002	1985-11-02	1985-11-05

Related dates need care. If `order_date` must stay after `date_of_birth` , shifting them independently can break the order. Delphix's *Dependent Date Shift* is a multi-column algorithm that masks two date-times while preserving the interval between them (Algorithms page).

2.32.7 Numeric variance

Move a number by a controlled amount. Informatica numeric blurring uses a fixed or percent variance. With the input 66: fixed bounds 10 and 10 give a result between 56 and 76; percent bounds 50 and 50 give 33 to 99 (p.170). A range option returns a number between a minimum and a maximum. Delphix Min Max pulls extreme values toward the middle, because a record with the highest salary may reveal a company's CEO (Min Max Number page). On ShopSphere, a 10 percent blur turns `total_amount` 1200.00 into something between 1080.00 and 1320.00, which keeps totals plausible but breaks the rule "total equals the sum of its items" unless the items are changed too, so decide whether tests rely on that.

2.32.8 Format-preserving masking

The masked value keeps the format the application checks. Informatica has built-in formats. Credit card masking returns a number with a valid checksum, leaves the six-digit bank identification number unchanged, and can keep or change the card issuer: 4539 1596 8210 2773 becomes 4539 1516 0556 7067 (p.158). Phone masking keeps the format: (408)382 0658 becomes (408)256 3106 (p.169). SSN masking generates valid numbers that avoid ones the Social Security Administration has issued (p.175). Delphix Payment Card preserves the Luhn check and the leading digits you choose: 5419033646326699 becomes 5419036803270758 when the first six digits are kept (Payment Card page).

FORMAT-PRESERVING MASKING OF SHOPSPHERE COLUMNS (EXAMPLE)

Column	Before	After
phone	9876543210	9412087365
ssn	123-45-6789	561-80-3472
email	asha.menon@example.com	leela.pillai@example.com

2.32.9 Character replacement

Replace characters according to a pattern. In Informatica's key and random techniques you set a *mask format*: A for letters, D for digits, N for alphanumerics, X for any character, + for "leave as is", and R for the remaining characters (pp.165, 170); and you can mask only chosen source characters or restrict the result to chosen characters. Delphix's Character Replacement framework swaps listed characters (for example vowels to "+") and wraps another string algorithm, and the page notes it has no UI or default instance at present, so it is configured through the API. A ShopSphere use: a tax ID shaped like AKPPM4821Q (five letters, four digits, one letter) masked with format AAAAADDDDA becomes something like TRWQD9037B, same shape, no real content.

2.32.10 Algorithm chaining

Combine several steps into one result. Informatica's *advanced masking* uses input, variable and output columns: mask FIRST_NAME and LAST_NAME separately with substitution into variable columns, then output FIRST_MASKED || ' ' || LAST_MASKED as the full name (p.158). Its email masking can build the address from the masked first and last name (p.159). Delphix's String Algorithm Chain runs two to eight string algorithms one after another, and its docs give a reason to chain: with deterministic masking, a very rare input value keeps its low

frequency in the output and could be used to infer the original, so a first algorithm can alter such rare values before the main one runs (String Algorithm Chain page).

WHY CHAIN AT ALL?

Real columns are related: the email contains the name, the full name contains the first name. If each is masked independently, the masked email will name someone different from the masked name. Chaining makes the dependent value follow the one it depends on.

2.33 Same input, same output: deterministic masking and referential integrity

This is the idea that separates amateur masking from professional masking. Imagine ShopSphere masks `CUSTOMERS.customer_id` randomly, and then masks `ORDERS.customer_id` randomly in a separate step:

WHAT HAPPENS WHEN THE SAME KEY IS MASKED INDEPENDENTLY, VERSUS CONSISTENTLY (EXAMPLE)			
Column	Original	Random each time	Deterministic (same seed)
CUSTOMERS.customer_id (Asha)	1001	48213	48213
ORDERS.customer_id (Asha's order 5001)	1001	90377	48213
Join CUSTOMERS to ORDERS	works	no match: order 5001 is an orphan	works

Masking a *primary key* or *foreign key* is only safe if every copy of the same real value turns into the same masked value. The same holds across systems: if the billing database and the orders database both hold customer 1001, they must both end up with 48213, or reconciliation tests will fail. This property is called *deterministic* (Informatica: *repeatable*) masking, and referential integrity depends on it.

2.33.1 How the tools achieve it

- **Informatica seed and repeatable output.** With repeatable output, "John Smith" is masked as "Frank Martinez" in every table, account number 1234 always becomes 6549

and 5678 always becomes 3214, across runs. A seed, a number from 1 through 999 with default 1, is the starting point; use the same seed on the same kind of column in different tables. An administrator can set a central seed that overrides all others, and a seed can come from a parameter file (pp.154 to 155).

- **Informatica cascades.** An *auto cascade* masks a parent key and the related foreign keys in child tables with the same value, applied by default where key constraints exist (p.91). A *value cascade* does the same for similar columns that are not linked by a key, such as `employee_num` in one table and `EmpID` in another, by choosing a driving column (p.90).
- **Delphix key and domain.** Most algorithms carry a key; the same algorithm with the same key gives the same output, and changing the key changes every output (the Algorithms page gives an example where "Michelle" becomes "Rachael" and after rekeying "Ben"). Because each domain has one default algorithm, every column found in that domain is masked the same way (Discovering your sensitive data).
- **Delphix across engines.** Algorithms can be synchronised between masking engines so that two engines mask the same input the same way. Algorithms that cannot guarantee the same output, such as the variable Date Shift, SecureShuffle and the random Secure Lookup, are called non-deterministic and cannot be used for syncable masking, while a fixed Date Shift is deterministic (Algorithm syncability page).

THINGS THAT QUIETLY BREAK CONSISTENCY

- **Different spellings of the same value.** Informatica's key masking is case sensitive by default, so "john brown" and "JOHN BROWN" can mask differently unless you convert the source to one format first with a preprocessing expression (p.166). Delphix Secure Lookup has a case-sensitive option and whitespace options with similar effects.
- **The value hiding inside another column.** A name in `FIRST_NAME` may also appear inside `FULL_NAME` or `DESCRIPTION`. Informatica's delimited string masking masks such substrings consistently, for example "Donne, John" and "John Donne, General Manager" (pp.166 to 167).
- **Random options.** A "random" mode is fine for a column that nothing links to. The Delphix Secure Lookup RANDOMIZE method "does not maintain referential integrity and should never be used" where it is required.
- **Different algorithm or seed for the same data in two jobs.** Keep one policy and one seed or key per kind of data (this is what a policy from the sensitive data chapter is for).

REPEATABLE OUTPUT HAS A PRICE

The same property that keeps joins working also means the mapping can be recomputed by anyone who has the seed or key and the algorithm. For a small value space (an SSN has about a billion possibilities) that is guessable. Treat seeds and keys as secrets, store them outside the test environment, and rotate them when policy says so (Delphix lets you randomise an algorithm key through its API; the Algorithms page gives the endpoint).

2.34 Irreversibility, uniqueness and collisions

Irreversible by design. A good masking technique throws the original away. Substitution picks an unrelated value; shuffle and key masking cannot be undone without the seed and source. Encryption and tokenization are the exceptions (section 2.30). **Unique by need.** Some columns must not repeat: primary keys, SSNs, emails used as login names. A technique that can produce the same output for two inputs (a collision) will break them. Choose by need:

When uniqueness matters		
Need	Technique that gives it	Source
Collisions are fine (names, cities)	Secure Lookup / Substitution	Delphix and Informatica: names naturally repeat
Unique values from a list	Informatica unique substitution (fails if the dictionary is too small); Delphix Mapping (each output to one input, but stores an encrypted mapping and tables are masked one at a time)	p.176; terminology page
Unique short numbers or codes, no collisions	Delphix Segment Mapping (no collisions if the allowed set is at least as large as the input; up to 36 characters) or Character Mapping	Terminology and Character Mapping pages
Unique SSNs	Informatica notes SSN masking cannot always return all-unique numbers because it avoids numbers already issued (p.175)	p.175

2.35 Static or dynamic, in-place or on-the-fly

Two different choices are often mixed up. **Static versus dynamic** asks *when* data is masked. **In-place versus on-the-fly** asks *where* the masked data ends up.

Static masking

Create a masked copy once. The copy is stored, and from then on anyone using it sees masked data. This is what test data management normally does.

Dynamic masking

Mask at the moment of reading, depending on who asks. The stored data stays real, and an application or proxy hides values from some users. It protects production views, not test copies. The TDM tool documentation does not cover it.

A word of caution on vocabulary: GenRocket also uses "dynamically" for masking that happens as rows are inserted into the destination, which is closer to on-the-fly than to the dynamic masking above (GenRocket "SDM and Data Subsetting Overview").

In-place and on-the-fly		
Method	What happens	Typical use
In-place	Updates the data in the source environment itself with masked values	You first make a copy (for example a virtual copy), then mask the copy in place. Never point it at production.
On-the-fly	Reads unmasked data from the source and writes masked data to a different target	Producing a masked copy directly from a source, together with subsetting

Delphix defines both in its masking job settings: in-place jobs update the source environment with the masked values; on-the-fly jobs read unmasked data from the source and write masked data to the target (create, view, edit, delete, verify jobs page). Informatica says a masking plan indicates whether to mask data in place in the source database or in stream in a target database (p.150). The masked-copy approach fits the rule from the sensitive data chapter: ask at which moment unmasked data exists outside production.

2.36 Risks and what goes wrong

Common risks in masking projects		
Risk	What happens	Defence
Re-identification	Masked names, but birth date, zip and gender still single out a person	Treat indirect identifiers too: date shift, generalise, subset
Sensitive data left in free text	An SSN inside a notes column survives column-level masking	Free text redaction, or exclude or null the column; search for patterns afterwards
Masking a primary key without cascading	Joins break, orphans appear	Same seed or key everywhere, auto and value cascades, or leave surrogate keys unmasked
Non-conformant data left unmasked	Delphix: a value that does not fit the algorithm's expected format is by default not masked and only a warning is recorded	Check job warnings; set non-conformant handling to fail or null where leaking is unacceptable (terminology page)
Masked value too long or wrong type	Job fails or truncates. Informatica's Change Metadata encryption needs a larger column precision using a formula given in the guide (p.162); it also does not mask integers over 28 characters (p.152)	Check lengths and types; use format-preserving options
Null and blank values	Treated unexpectedly	Informatica exception handling can set constant, ignore, treat as value or log an error (p.155)

Risk	What happens	Defence
Collisions on unique columns	Duplicate key errors on load	Use a unique technique (section 2.34)
Performance and memory	Large lookup files and parallel jobs need memory; Delphix suggests about 1 GB per job or stream, more for wide rows or big lookups, and about one vCPU per concurrent job	Size the engine; mask a subset (best practices page)
Realism lost	Fake data fails validation or no longer exercises the code	Format-preserving techniques; test the rules on a sample first

2.37 Choosing a technique

Start from the column, not the technique. Ask four questions: What is it (name, number, date, free text)? Must it stay unique or linked to other tables? Must it keep a format the application checks? Do tests depend on its meaning (for example an age band)? The table gives a reasoned first choice for the ShopSphere columns. (technique names are from section 2.32)

Decision table for ShopSphere columns (a starting point, to be confirmed with the compliance officer)

Column	Constraint	First choice	Why
first_name, last_name	Realistic; repeats allowed	Substitution from a dictionary	Looks real; no uniqueness needed
email	Valid format; maybe unique; should match the masked name	Email masking built from the masked names (chaining)	Keeps consistency inside the row
phone	10 digits	Format-preserving phone masking	Keeps the shape the application validates
ssn	999-99-9999 format; unique	Format-preserving SSN masking, deterministic	Valid shape; same output wherever it appears
date_of_birth	Plausible; age bands matter	Date shift or blurring by a few days	Hides the exact date, keeps the age band
zip_code	Valid code; region may matter	Substitution from valid codes, or keep the first digits	Reduces re-identification, keeps regional logic
tax_id	Shape like AAAAADDDDA	Character replacement with a mask format, or null	Same shape, no real content
customer_id (key)	Joins to ORDERS	Leave as is (a meaningless surrogate), or key masking with the same seed in every table	A made-up number identifies nobody on its own, but must stay consistent

Column	Constraint	First choice	Why
total_amount	Must match order items	Leave alone, unless amounts are sensitive	Masking it breaks arithmetic
Free-text notes	Unknown content	Free text redaction or null	Cannot be fixed by column-level rules

NOT EVERY COLUMN NEEDS MASKING

Masking has a cost: it can break realism and arithmetic. A surrogate key like `customer_id` that carries no meaning is often safe to leave, as long as nothing else in the row identifies a person. The compliance officer decides, not the tool.

2.38 Validating masked data

A masking job that finishes without errors is not proof that the data is safe or useful. Check the result against the goals from section 2.31.

A validation checklist

Check	How
Nothing sensitive is left	Run discovery again on the masked copy. Search free-text columns for SSN-like and phone-like patterns. Compare: no real value from the source should appear in the target for masked columns.
Joins still work	Run orphan queries between each parent and child table.
Consistency	The same real value gives the same masked value in every table and system that holds it.
Valid format	Pattern check on each masked column, for example SSN <code>[0-9]{3}-[0-9]{2}-[0-9]{4}</code> and card numbers pass the checksum.
Uniqueness	No duplicates in columns that must be unique.
Realism	Look at a sample by eye. Null counts and value distributions should be believable.
Repeatability	Run twice with the same settings; results match when they should.
Job logs	Read warnings. Delphix shows a yellow warning marker on a job when non-conformant data was found; Informatica logs exceptions per its exception-handling settings.

```
-- duplicates in a column that must be unique
SELECT ssn, COUNT(*) FROM CUSTOMERS GROUP BY ssn HAVING COUNT(*) > 1;

-- any order without a customer (orphan)
SELECT o.order_id FROM ORDERS o
LEFT JOIN CUSTOMERS c ON c.customer_id = o.customer_id
WHERE c.customer_id IS NULL;

-- any SSN-shaped text left in a notes column (pattern syntax varies by database)
SELECT order_id FROM ORDER_NOTES WHERE notes LIKE '%___-__-____%';
```

Tools help with previewing. Informatica's Rule Simulator shows original and masked values for a standard rule before you assign it, using default sample data, your own sample or a connection, with up to 100 rows and a default of 20 (pp.137 and 140). Delphix says its platform generates reports and email notifications to confirm that sensitive data has been masked (Delphix "Introduction to Delphix Masking"). Neither removes the need to inspect the result yourself.

Lab: design and check a mask for the ShopSphere customers

Goal: pick a technique for every sensitive column of CUSTOMERS, apply it to the three sample rows by hand, and prove the result is consistent.

1

Write the rule set.

For each column in the decision table, choose one technique and say whether it must be deterministic. Decide the seed or key story: one seed, shared by every column that joins elsewhere.

2

Apply it to rows 1001 to 1003.

Use made-up values that respect each column's format. Build the email from the masked first and last names.

Expected result: Three masked rows with valid phone, SSN and date formats and no original value surviving.

3

Check the rules.

Does any masked value equal a real one? Is any value duplicated? Does the masked email match the masked name? Would an orphan appear in ORDERS if customer_id were masked here but not there?

4

Re-run in your head with the same seed.

Would 1001 still become the same row next week? If not, which technique is random?

► Show a worked answer

2.39 Common mistakes and gotchas

MISTAKES BEGINNERS MAKE

- **Calling encryption "masking".** It can be reversed. Use it only when reversal is intended.
- **Masking keys independently.** Joins break. Use the same seed or key, or cascades.
- **Ignoring the job warnings.** In Delphix, non-conformant values are left unmasked by default and only flagged.
- **Trusting one technique for everything.** Shuffle leaves real values in place; substitution can collide; date shifting can reorder dates.
- **Forgetting free text and indirect identifiers.** They defeat column-level masking.
- **Sharing the seed or key with the test team.** It makes the mapping reproducible by anyone who has it.
- **Declaring success without validating.** Run discovery again on the masked data and check joins.

2.40 Recap

REMEMBER

- Masking replaces real values with realistic fake ones, usually irreversibly. Encryption and tokenization are reversible; anonymisation is a legal goal that masking helps reach.
- Good masked data is safe, realistic, valid, consistent, repeatable and still fit for the tests.
- Common techniques: substitution, shuffle, key masking, encryption, nulling and redaction, date shifting, numeric variance, format-preserving masking, character replacement, and chaining several of them.
- Deterministic masking (same input, same output, from a seed or key) is what keeps keys and joins working across tables and systems. Random masking is for columns nothing links to.
- Choose by column type, uniqueness, format and test needs. Not every column needs masking, and free text needs special handling.
- Masking can be in place or on the fly; static copies are the norm for test data.
- Always validate: rediscover, check joins, formats, uniqueness, repeatability and job warnings.