TEST DATA MANAGEMENT

Test data management (TDM), a sister function of Test Environment Management, is the construction of test data sets that reliably represents an organization's actual data so that IT teams (developers and testers) can effectively exercise software testing. Tests include Unit Testing, System Testing, Integration Testing, User Acceptance Testing, Performance Testing, and Security Testing.

Test Data Management Functions

Test Data Management is a broad subject and covers many facets often including:

Test Data Requirements Gathering. Gathering the needs of software testing i.e. identifying the data quality needed to build your test scenarios.

Data Profiling / Data Discovery. Understanding your Production Data. This is critical if you wish to understand what valid test data looks like, identify where personally identifiable information hides, and understand data security risk and data breach avoidance.

Data Extraction (often from Production data). Note: One benefit of using production data, as a base, is it will help ensure realistic test data.

Data Transformation

+ Data Sub-setting. That is condense or prepare test data so that it is both smaller & easier to handle.

+ Data Masking / Data Privatization. Protect data & protect customer PII through obfuscating sensitive data e.g. sensitive customer data. Note: Masking will usually use a copy of the original sensitive data i.e. production data. And by data masking, for example, data encryption, you are implementing data privacy to reduce the opportunity for a data breach.

Test Data Provisioning i.e. the deployment of test data.

Data Cloning aka Data Virtualization (the concept of snapshotting & deploying "tiny" replica databases). Note: Data cloning is rapidly displacing the need for test data subsetting. Cloning can also be used beyond software testing and against real data.
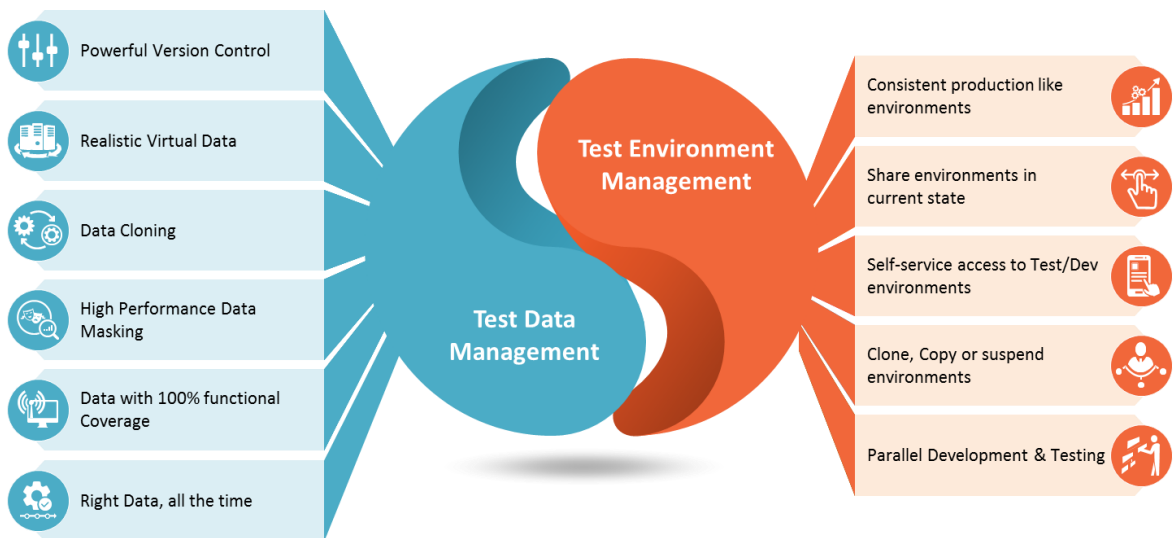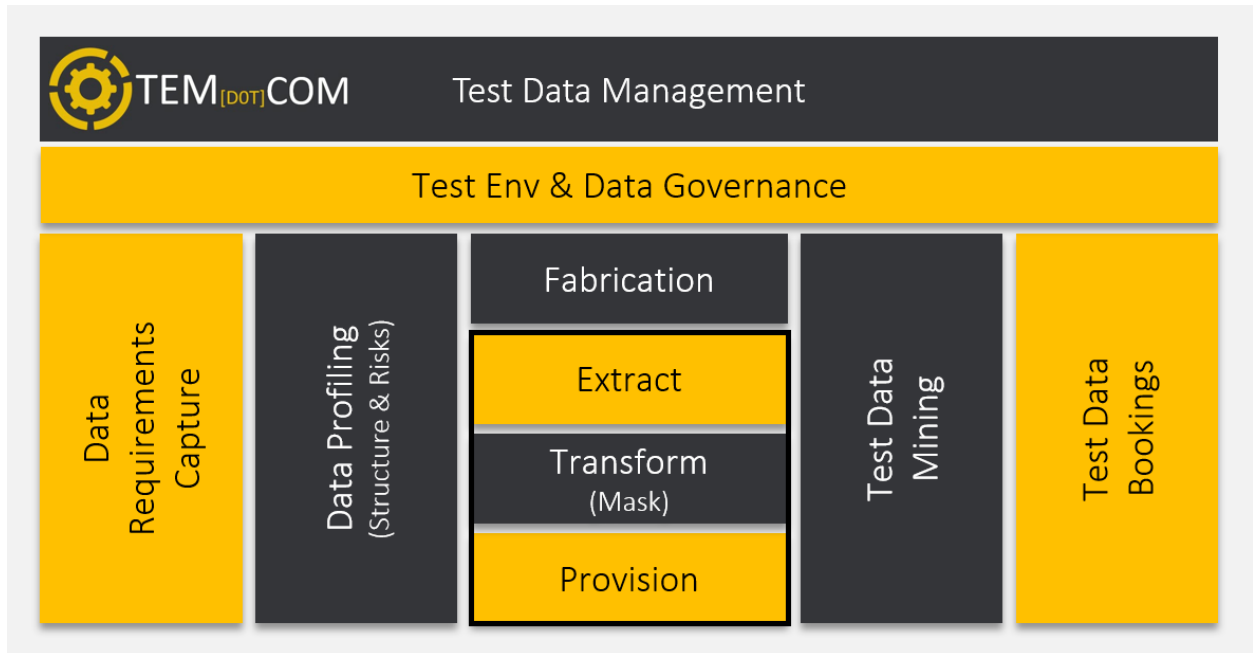
Data Fabrication / Data Synthetics. A method of test data preparation. Create test data using a data generation tool to create synthetic data (fake test data), from scratch.
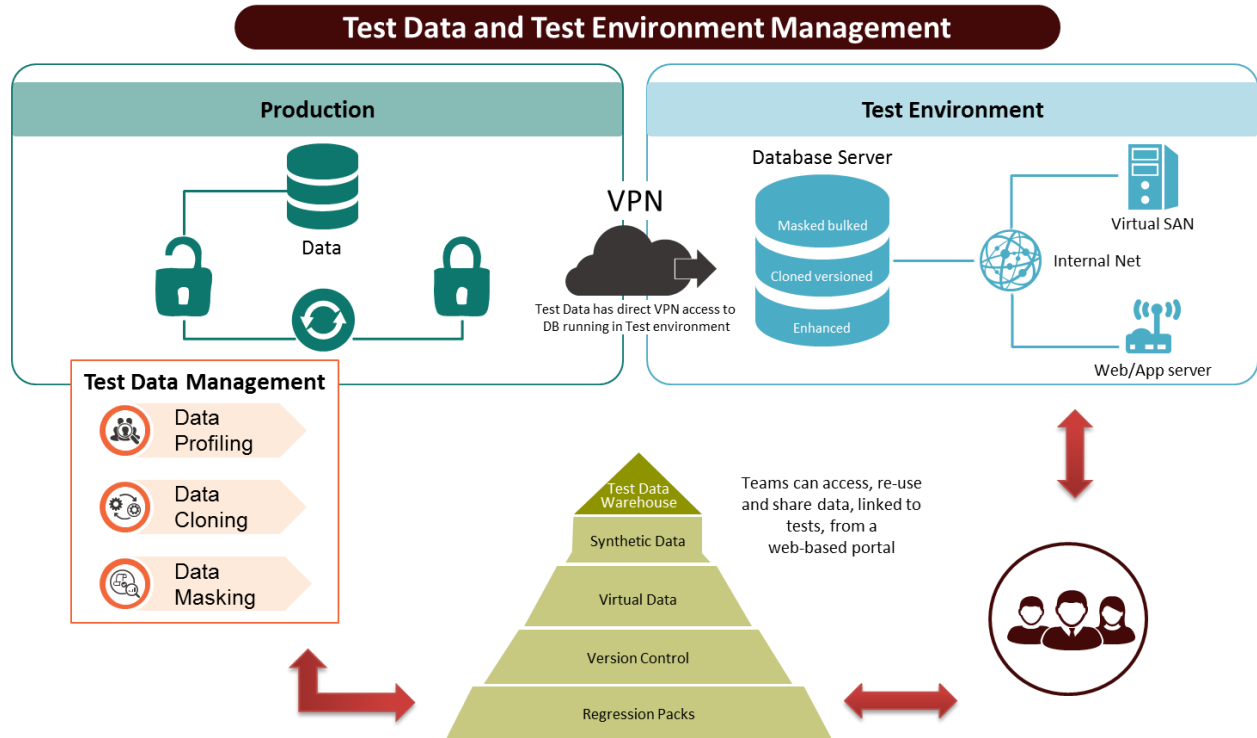
DataOps (any data operation that will help orchestrate data activities e.g. export, import, snapshot).

Test Data Mining. Methods to support ease of data access & helping software testing teams find sample data for testing. Note: Data mining can also be used to reduce the opportunity for data breaches by limiting what can be mind.

Test Data Bookings. Another method to simplify data access is Test Data Booking Management. Test Data Reservation methods help testing teams reserve test data to avoid test data contention.

Test Data Reporting e.g. Compliance, Size, Usage, Performance, etc.

Test Data and Test Environment Management

Production

Data

Test Data Management
- Data Profiling
- Data Cloning
- Data Masking

VPN

Test Data has direct VPN access to DB running in Test environment

Test Environment

Database Server
- Masked bulked
- Cloned versioned
- Enhanced

Virtual SAN

Internal Net

Web/App server

Test Data Warehouse
Synthetic Data
Virtual Data
Version Control
Regression Packs

Teams can access, re-use and share data, linked to tests, from a web-based portal

TEST ENVIRONMENT MANAGEMENT

Test Environment Management (TEM) helps companies to speed up their software releases with close daily collaboration between all team members, to capture environment demands, to establish a simple and transparent environment utilization, to organize effective cooperation through planning and scheduling resources and control their lifecycles. TEM helps to streamline delivery by providing a validated, stable and available environment to execute any test scenarios or replicate bugs.

Why is Test Environment Management important?

Teams typically need a number of test environments for each production application, which depends on various circumstances. However, every business application may demand from as little as one to as many as, hundreds of test environments. IT architecture tendencies include distributed systems, various integrations, development, and testing across shared environments. All of these factors add additional complexity to managing projects, releases, and even achieving business goals.

With the rapid increase of 'agile' development environments, the necessity for regular and even more frequent software releases has also escalated, sometimes to the level of daily and hourly, or on-demand.