**Summary of Behavior**

| **Feature** | **Primary** | **Secondary (Always On)** | **Replication Subscriber** |
| --- | --- | --- | --- |
| **Dynamic Data Masking** | Masking rules enforced by schema | Enforced same as primary | Not replicated — must reapply rules |
| **Static Data Masking** | Masked data stored in table | Masked data replicated/restored | Masked data copied, no rules |
| **Tokenization** | Only tokens stored; vault external | Tokens replicated; vault lookup still needed | Tokens replicated, vault must be reachable |

**Behavior by Masking Type at the secondary**

**1. Dynamic Data Masking (DDM)**

* **Original data is still stored** in the table, even on the secondary.
* The masking is **applied at query time** based on the user’s role/permissions.
* ✅ If you are a user with **UNMASK permission** or sysadmin role, you will see the **original values** at the secondary.
* ❌ If you don’t have those permissions, you’ll always see masked values (e.g., xxxx@domain.com).

**👉 So at migration time, you just need an account with UNMASK permissions on the secondary.**

**2. Static Data Masking (SDM)**

* This is **permanent replacement** of data before moving into a non-prod or target DB.
* The **masked value is the only value stored** — the original data is gone.
* ✅ If secondary was built from SDM output, you **cannot recover the original**.
* ❌ To migrate original data, you must go back to the **primary (unmasked source)** and copy from there.

**👉 Static masking is irreversible — no way to “view originals” at secondary.**

**3. Tokenization**

* Only the **tokenized value** is stored in the database (TKN-90812-XYZ).
* ✅ To view the original data at the secondary, you must integrate with the **tokenization vault/service**.
* The vault securely maps back TKN-90812-XYZ → 1234567890.
* ❌ Without vault access, neither primary nor secondary can reveal the original.

**👉 During migration, you either (a) migrate tokens + vault together, or (b) re-tokenize in the new system.**

**Where Static Data Masking (SDM) Fits in**

1. **Production Database (Real PHI/PII)**
   * Contains original sensitive data.
   * Developers/testers should **never connect directly** here.
2. **Restore Copy for Dev/Test**
   * You first take a **backup of production**.
   * Restore that backup to a **separate environment** (Dev/UAT/QA).
   * At this point, it’s still holding the **real data** (so access must be restricted).
3. **Apply Static Data Masking (SDM)**
   * Immediately after restore, run your **masking scripts/tools**.
   * Replace PHI/PII with fake but realistic values.
   * Once applied, the original values are gone in that copy.
4. **Provide Access to Developers/QA**
   * Developers now connect to the **masked copy**.
   * They see only the fake data (no risk of PHI/PII exposure).

**Situations Where You Might Use SDM in Production**

**1. Data Archiving or Decommissioning**

* When retiring a system or moving data to long-term archive.
* PHI/PII is masked so that even if the archive is breached, no sensitive info is exposed.
* Example: Old patient records older than 10 years, no longer needed for care but kept for compliance.

**2. Compliance-Driven Data Retention Rules**

* Some regulations (GDPR, HIPAA, CCPA) require **“forgetting” or anonymizing** certain data after a retention period.
* Instead of deleting records (which might break referential integrity), you apply SDM to anonymize them.
* Example: Masking names and SSNs after 7 years, while keeping transaction totals for auditing.

**Practical Options for Migration**

* **For DDM**:
  + Use an account with UNMASK permission at the secondary to export originals.
* **For SDM**:
  + You must go back to the unmasked production source — migration from SDM’d data is **not possible**.
* **For Tokenization**:
  + Include the **vault** in your migration plan, or work with de-tokenization API to restore originals when needed.

**Tokenization vs. Other Approaches *In-Flight***

| **Method** | **What Happens In-Flight** | **Risks/Notes** |
| --- | --- | --- |
| **Static Masking** | Masked values flow through test/dev environments. | Safe for non-prod, but useless for prod where real IDs are needed. |
| **Dynamic Masking** | Depending on role, data in-flight is masked or clear. | Risk if roles misconfigured; original data still flows internally. |
| **Encryption** | Encrypted ciphertext flows in-flight, decrypted at endpoint. | Strong, but if keys are compromised, values can be recovered. |
| **Tokenization** | Tokens flow in-flight, with no link to original without vault. | Strongest for transactional IDs; vault security is critical. |