5️⃣ ONE-LINE SUMMARY

Rule Type	INSERT	UPDATE
Business	Core + Relationships	Core + Relationships + IDs
Logical	Core + Relationships	Core + Relationships + IDs
Technical	Core + Relationships + **DB-backed Dimensions**	Core + Relationships + IDs + **DB-backed Dimensions**

If you want next, I can:

- Convert this into **Swagger validation documentation**
- Map each row to **exact constraint class names**
- Produce a **checklist for test case coverage**
- Generate a **decision table used by validation service**

Just tell me which.

include CROSS-CUTTING also in individual

Below is the **reworked, authoritative validation matrix**, where **CROSS-CUTTING validations are explicitly included inside each Rule Type × Action Type section** (as requested).
This reflects **exactly what is implemented**, not conceptual rules.

BUSINESS RULE

🔹 INSERT

Request-level

- `actionCode`
  - **Mandatory**
  - Must be `INSERT`

Identity (cross-cutting)

- Exactly **one** must be present:
  - `user.soeid`
  - **OR** `appInfo.appId`
- ❌ Both present → error
- ❌ None present → error

Entity (core)

- `entity.entityName` – **Mandatory**
- `entity.entityType` – **Mandatory**
- `entity.metadataStatusCd` – **Mandatory**
- `entity.metadataRecordType` – **Mandatory**
- `entity.entityDescription` – **Mandatory**

Entity IDs

- `entity.entityMasterId` – **Must be NULL**
- `entity.recordUid` – **Must be NULL**

Entity Name (cross-cutting)

- `entity.entityName`
  - Must be **unique across all entities in the request**

Entity Relationships

- At least one **Business Rule primary DE relationship** required
- For **each relationship**:
  - `relationshipType` – Mandatory
  - `entityType` – Mandatory
  - `metadataStatusCd` – Mandatory
  - `metadataRecordType` – Mandatory
  - `toUuid` – Mandatory
  - `toEntityMasterId` – Mandatory

Entity Dimensions

- Optional
- If present:
    - `metadataStatusCd` – Mandatory
    - Controlled parameters validated (if applicable)

🔹 UPDATE

Request-level

- `actionCode`
    - **Mandatory**
    - Must be `UPDATE`

Identity (cross-cutting)

- Exactly one of:
    - `user.soeid`
    - `appInfo.appId`

Entity (core)

- `entity.entityName` – **Mandatory**
- `entity.entityType` – **Mandatory**
- `entity.metadataStatusCd` – **Mandatory**
- `entity.metadataRecordType` – **Mandatory**
- `entity.entityDescription` – **Mandatory**

Entity IDs

- `entity.entityMasterId` – **Mandatory (>0)**
- `entity.recordUid` – **Mandatory (>0)**

Entity Name (cross-cutting)

- Unique across request

Entity Relationships

- Same as INSERT

Entity Dimensions

- Same as INSERT

LOGICAL RULE

◆ INSERT

Request-level

- `actionCode` **– Mandatory (`INSERT`)**

Identity (cross-cutting)

- Exactly one of:
  - `user.soeid`
  - `appInfo.appId`

Entity (core)

- `entity.entityName` – Mandatory
- `entity.entityType` – Mandatory
- `entity.metadataStatusCd` – Mandatory
- `entity.metadataRecordType` – Mandatory
- `entity.entityDescription` – Mandatory

Entity IDs

- `entity.entityMasterId` – Must be NULL
- `entity.recordUid` – Must be NULL

Entity Name (cross-cutting)

- Must be unique across entities

Entity Relationships

- Logical-rule relationship mandatory
- Same mandatory relationship fields as Business Rule

Entity Dimensions

- Optional
- If present:
  - `metadataStatusCd` mandatory
  - Controlled parameters validated

◆ UPDATE

Request-level

- `actionCode` – **Mandatory (`UPDATE`)**

Identity (cross-cutting)

- Exactly one of:
  - `user.soeid`
  - `appInfo.appId`

Entity (core)

- Same as INSERT

Entity IDs

- `entity.entityMasterId` – Mandatory
- `entity.recordUid` – Mandatory

Entity Name (cross-cutting)

- Unique across request

Entity Relationships

- Same as INSERT

Entity Dimensions

- Same as INSERT

TECHNICAL RULE

◆ INSERT

Request-level

- `actionCode` – **Mandatory (`INSERT`)**

Identity (cross-cutting)

- Exactly one of:
  - `user.soeid`
  - `appInfo.appId`

Entity (core)

- `entity.entityName` – Mandatory

- `entity.entityType` – Mandatory
- `entity.metadataStatusCd` – Mandatory
- `entity.metadataRecordType` – Mandatory
- `entity.entityDescription` – Mandatory

Entity IDs

- `entity.entityMasterId` – Must be NULL
- `entity.recordUid` – Must be NULL

Entity Name (cross-cutting)

- Must be unique across request

Entity Relationships

- Same base relationship validations

Entity Dimensions (STRICT – DB-BACKED)

Applies to `DcmrControlledParameter`
Each of the following **paramNames is mandatory**
and **paramValue must exist in DB reference data (loaded at startup)**:

- `ruleEngineCsiIdentifier`
- `ruleEngineZoneIdentifier`
- `dataNamespace`
- `dataQualityRuleDeploymentIdentifier`
- `dataProfileTeam`
- `dataQualityRuleSchedule`
- `expressionLanguage`
- `dataQualityRuleTimePeriod`
- `dataQualityTargetSystemCsiIdentifier`
- `eagleScope`

Validation behavior:

- Reference data loaded at **application startup**
- Missing / invalid value → validation error

`DcmrNumericParameter`

- Type resolved
- No DB constraint yet (extensible)

🔹 UPDATE

Request-level

- `actionCode` – **Mandatory (`UPDATE`)**

Identity (cross-cutting)

- Exactly one of:
  - `user.soeid`
  - `appInfo.appId`

Entity (core)

- Same as INSERT

Entity IDs

- `entity.entityMasterId` – Mandatory
- `entity.recordUid` – Mandatory

Entity Name (cross-cutting)

- Unique across request

Entity Relationships

- Same as INSERT

Entity Dimensions

- **Same DB-backed validations as INSERT**

ERROR HANDLING (APPLIES TO ALL)

- **All validations are accumulated**
- No early throws inside constraints
- Final response contains:
  - Multiple validation errors
  - Full JSON path (`datasets[0].entities[1].entityDimensions[3].paramValue`)
- Single response → user fixes everything in one go

FINAL SUMMARY TABLE

Rule Type	INSERT	UPDATE
Business	Core + Relationships + Cross-cutting	Core + Relationships + IDs + Cross-cutting
Logical	Core + Relationships + Cross-cutting	Core + Relationships + IDs + Cross-cutting
Technical	Core + Relationships + **DB-backed Dimensions** + Cross-cutting	Core + Relationships + IDs + **DB-backed Dimensions** + Cross-cutting

If you want next, I can:

- Map **each bullet → exact constraint class**
- Produce **JUnit test matrix**
- Generate **Swagger/OpenAPI validation docs**
- Create **onboarding documentation for new devs**

Say the word.