Datatype Semantics

Go up to Terms and Definitions

Discussion

Datatype semantics defines meaning through the value space, permitted representation, and permitted operations associated with a datatype.

A datatype constrains how a data value appears and how systems process that value. Datatype semantics applies to primitive datatypes such as `string`, `integer`, `decimal`, `boolean`, and `date`, and to constrained datatypes such as currency codes, jurisdiction codes, identifiers, date formats, numeric ranges, and enumerated values.

Datatype semantics differs from type semantics. A datatype constrains representation. A type identifies the semantic category of the thing represented. For example, `tradeDate`, `settlementDate`, `effectiveDate`, and `terminationDate` use the same date datatype, but each term has a different domain meaning.

Datatype semantics also differs from semantics as a whole. A datatype contributes to meaning by constraining values and operations, but a datatype does not establish the full domain meaning of a data item. Domain meaning comes from the associated term, schema, type system, conceptual model, rule, report, ontology, or other authoritative semantic source.

Datatype semantic content includes:

- Value spaces
- Lexical representations
- Permitted operations
- Format constraints
- Range constraints
- Precision and scale constraints
- Units of measure
- Enumeration values
- Pattern constraints
- Nullability rules
- Default values
- Conversion rules
- Comparison rules
- Validation rules
- Traceability to the domain meaning

A governed architecture preserves traceability from datatype semantics to the schema, type system, conceptual model, business rule, or other semantic source that establishes the intended interpretation of the data value.

Definition

meaning defined through the value space, permitted representation, and permitted operations associated with a datatype

Source

DIDO Solutions usage, informed by datatype, schema-language, database, modeling, and type-system practice.

Note

Datatype semantics constrains representation-level meaning. Type semantics constrains domain-level categorization and interpretation. A robust architecture preserves the distinction between datatype and type.

Example

An FX trade message contains the following information elements:

Information element	Datatype	Semantic type	Domain meaning
`tradeDate`	`date`	`TradeDate`	date on which the trade occurs
`settlementDate`	`date`	`SettlementDate`	date on which settlement occurs
`notionalAmount`	`decimal`	`NotionalAmount`	amount used to calculate settlement obligations
`exchangeRate`	`decimal`	`ExchangeRate`	rate used to exchange one currency for another
`counterpartyLEI`	`string`	`LEI`	legal entity identifier for a counterparty
`currencyPair`	`string`	`CurrencyPair`	ordered pair of currencies exchanged in the trade

The datatype `date` constrains the representation and valid date operations for both `tradeDate` and `settlementDate`. The datatype does not make the two information elements semantically equivalent. The associated semantic types and business rules define their different meanings.

The datatype `decimal` constrains the representation and numeric operations for both `notionalAmount` and `exchangeRate`. The semantic types prevent an implementation from treating a notional amount and an exchange rate as interchangeable decimal values.