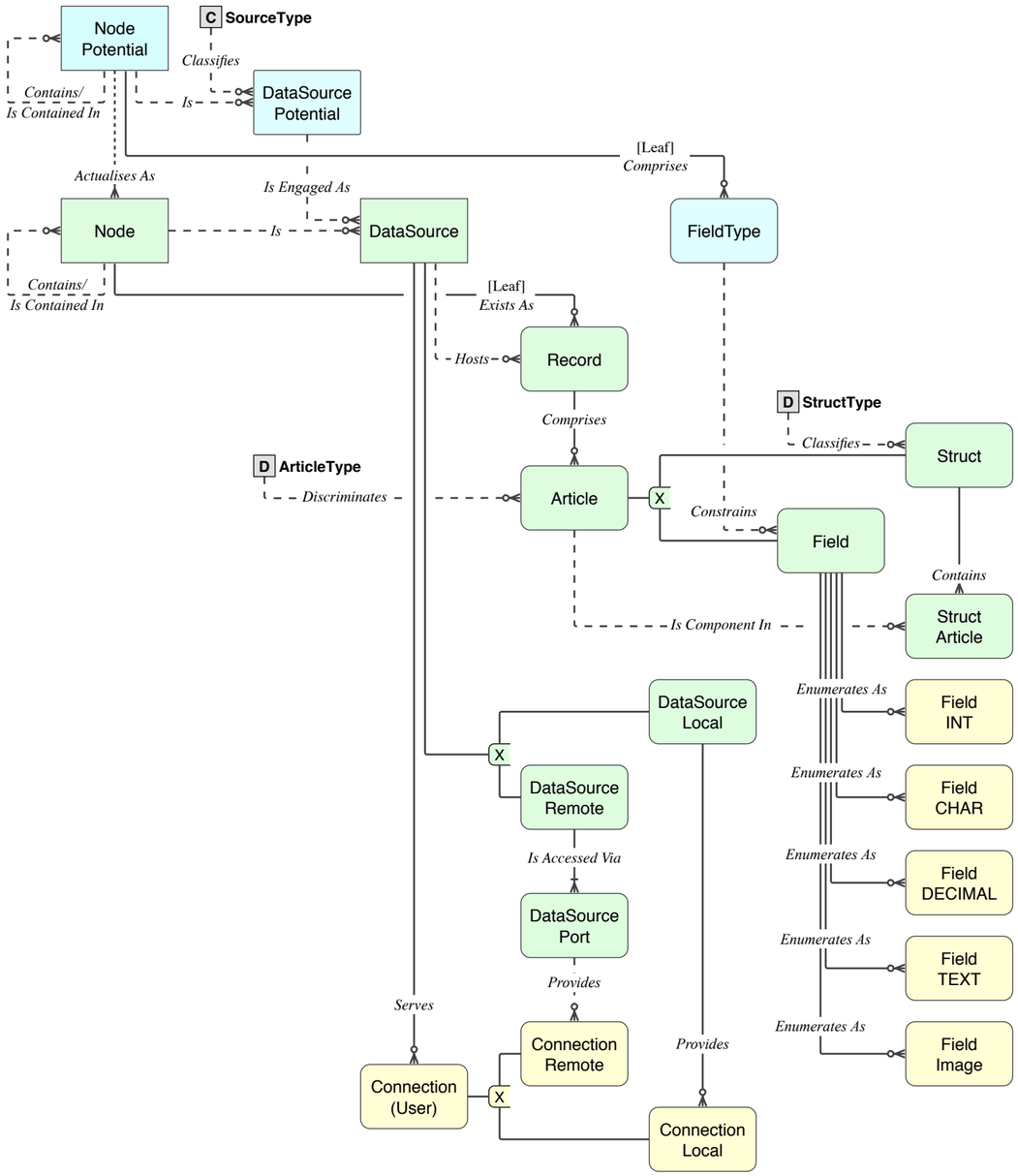
Table Relation



Entity Type

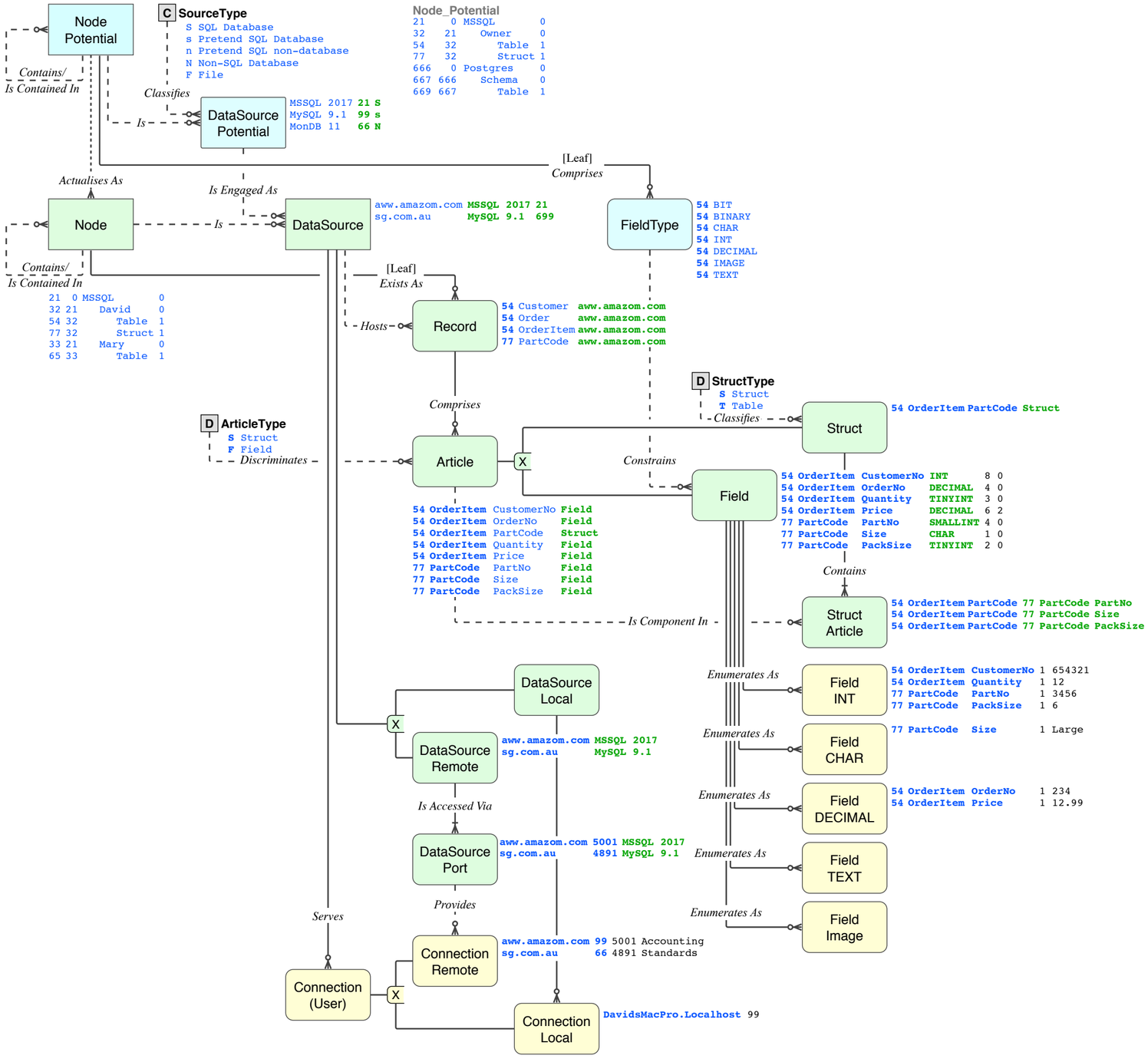
- Reference/Major (Definition Potential)
- Reference/Simple
- Identifying (Definition Actual)
- Transaction (Accessed Data)

IDEF1X Notation

Relational Subtype

Relational Hierarchy

Table Relation + Data



CatalogPotential
Defines that which is physically possible in each DataSource.
- It is set up by a developer.
- The **DataSource_Potentials** that are available in the market.
- **Node_Potential** defines the hierarchy of definition of objects, that which is physically possible in each DataSource_Potential
  - the Root level is the DataSource
  - the Leaf level is that which can contain data:
    a Record[Type], the atomic unit of storage in the DataSource, that contains FieldTypes
  - it is representative (eg. a table is defined but cannot be named)
- In order to prevent circular references (a Node containing itself), we need
  1 a recursive Function that generates *Path*, and
  2 a Constraint that checks Node NOT IN *Path*
- **FieldType** defines the DataTypes, the addressable unit of storage possible in the DataSource

CatalogueActual
- It is constrained to CataloguePotential
  - For SQL DataSources, it can be loaded automatically from the SQL Catalogues
  - For others, it is set up by UserAdmin
- The **DataSources** that are contracted; engaged, that contain data
- **Node** defines the hierarchy of objects, that which exists in each DataSource
  - the Root level is the DataSource
  - the Leaf level is that which can contain data:
    a Record, the atomic unit of storage in the DataSource, that contains Fields
  - it is actual, each table and struct is named
- **Record** (Row; Struct) is the unit addressable by a query against the DataSource
- A **Struct** requires the same defn as a Record (list of Fields), and a Struct may contain Struct, This is resolved by the abstraction **Article**, it is either a Struct xor a Field
- In order to prevent circular references (a Struct containing itself), we need
  1 a recursive Function that generates *Path*, and
  2 a Constraint that checks Struct NOT IN *Path*
- **Field** (Column) is the unit of address, a FieldType, with a narrower definition, particular to the Record
- An **array** is identified by:
  - `NumElement` declares the number of elements, which `ElementNo` must not exceed
  - The Constraints in the child tables are the same as they would be for a Subtype cluster (except that it is multi-valued)
- For **scalars**, instead of Subtype cluster on Field (which would be the proper method), use multi-valued Field cluster with `NumElement = 1`.
- Thus the Subtype cluster is avoided.
- While not restricting ourselves to the *Relational Model*, because we have to cater for non-relational and even non-database data sources, we still need to be vigilant, not silly. That means even though a data source may allow circular references, or an array of Structs, or duplicate names (CataloguePossible), we do not allow that in reality (CatalogueActual). The alternative is insanity, the domain of academics.

Current Data
The **data values** that are currently accessed by the user in the webpage, via a Connection,as a result of queries against the DataSource. It is defined by (constrained to) the CatalogueActual.
- The assumption is, the webpage is used by a single user (shown)
- Otherwise the Field cluster needs to be Constrained by Connection (not shown).

Entity Type
- Reference/Major (Definition Potential)
- Reference/Simple
- Identifying (Definition Actual)
- Transaction (Accessed Data)

IDEF1X Notation

Relational Subtype

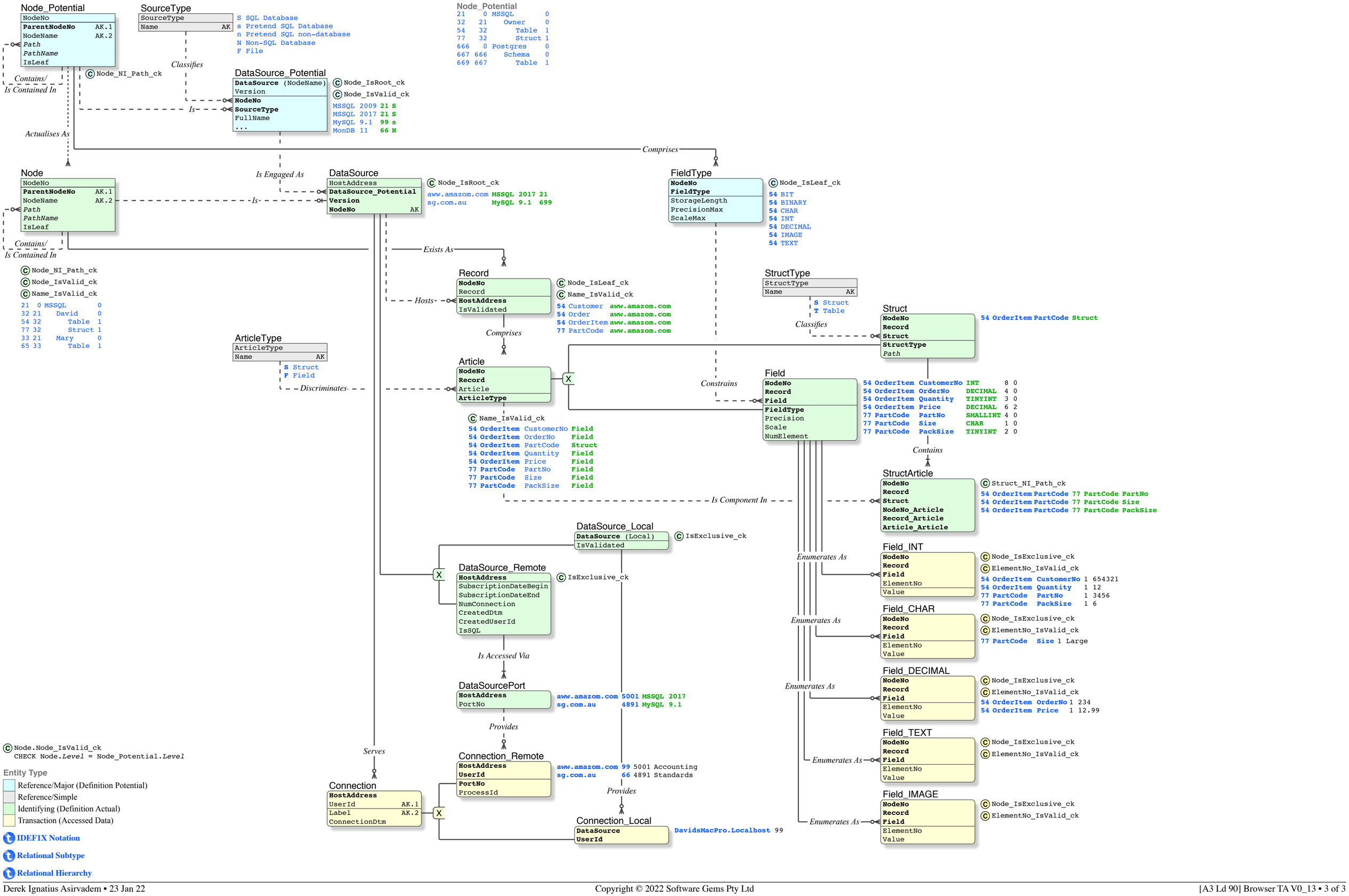
Relational Hierarchy

Table Attribute + Data

SOFTWARE GEMS

Node_Potential

21	0	MSSQL		0
32	21		Owner	0
54	32		Table	1
77	32		Struct	1
666	0	Postgres		0
667	666		Schema	0
669	667		Table	1



Node.Node_IsValid_ck
CHECK Node.*Level* = Node_Potential.*Level*

Entity Type

- Reference/Major (Definition Potential)
- Reference/Simple
- Identifying (Definition Actual)
- Transaction (Accessed Data)

IDEF1X Notation

Relational Subtype

Relational Hierarchy