NEXUS-1 - DATA BACKBONE

APPENDIX C.0

Schema-wide Conventions and Master Map for the NEXUS-1 industrial digital twin database

SQL SERVER / EF CORE FRIENDLY DDL BASELINE

One schema, seventeen sectors, one source of truth

Foundation section for Appendix C - SQL Schema Definitions.

GRIGORIOS AGATHANGELIDIS

DRAFT BUILD - APPENDIX C.0

Appendix C.0
Schema-wide Conventions and Master Map

This appendix gathers the foundation rules for the full SQL schema of the NEXUS-1 data backbone. It is written for SQL Server and for clean reverse-engineering into Entity Framework Core.

The goal is not to minimise the table count. The goal is to make every table explicit, every key named, every cross-sector dependency visible, and every lookup enforceable by a real foreign key.

> *One sector, one SQL schema.*
> *One meaning per key.*
> *One source of truth.*
> *Foreign keys are the passports between sectors.*

The schema is demonstrator-grade and educational. It supports the NEXUS-1 companion console and the books in the series. It is not a licensed plant information model, not a safety-class database, and not a claim about any real facility.

C.0.1 SQL Server baseline

The DDL in this appendix assumes Microsoft SQL Server and uses SQL Server-native conventions.

CONCERN	CONVENTION
Schema namespaces	One SQL schema per NEXUS-1 sector
Normal entity key	`INT IDENTITY(1,1)`
High-volume fact key	`BIGINT IDENTITY(1,1)`
Timestamps	`DATETIME2(7)` in UTC
Optimistic concurrency	`ROWVERSION` on mutable operational tables
Boolean values	`BIT NOT NULL DEFAULT 0/1`
Text values	`NVARCHAR`, never `VARCHAR`
Decimal engineering values	`DECIMAL(p,s)` where precision matters
Fast telemetry values	`FLOAT` only where physical approximation is acceptable
Delete behaviour	`NO ACTION` unless cascade is explicitly justified
Soft delete	`IsDeleted BIT NOT NULL DEFAULT 0` on mutable business tables
Audit stamping	`CreatedAtUtc`, `CreatedBy`, `ModifiedAtUtc`, `ModifiedBy`
Immutable event tables	Append-only, usually no soft delete
Lookup tables	One typed table per lookup category
Index names	`IX_<Schema>_<Table>_<ColumnList>`
PK names	`PK_<Schema>_<Table>`
FK names	`FK_<Schema>_<Table>_<PrincipalTable>`
Unique constraints	`UQ_<Schema>_<Table>_<ColumnList>`
Check constraints	`CK_<Schema>_<Table>_<RuleName>`

C.0.2 Sector schemas

The database is divided into seventeen schemas. Each schema owns its own vocabulary. The schema name is therefore not decoration; it is part of the meaning.

```
CREATE SCHEMA CorePlatform;
CREATE SCHEMA Security;
CREATE SCHEMA Organization;
CREATE SCHEMA ReactorFleet;
CREATE SCHEMA DigitalTwin;
CREATE SCHEMA Instrumentation;
CREATE SCHEMA AlarmManagement;
CREATE SCHEMA EventManagement;
CREATE SCHEMA Maintenance;
CREATE SCHEMA RootCause;
CREATE SCHEMA ReinforcementLearning;
CREATE SCHEMA Robotics;
CREATE SCHEMA RadiationMonitoring;
CREATE SCHEMA EmergencyPreparedness;
CREATE SCHEMA Compliance;
CREATE SCHEMA Reporting;
CREATE SCHEMA Audit;
```

> A `Unit` in `ReactorFleet` means a physical reactor unit. A unit of measure belongs in `CorePlatform`. An organisational unit belongs in `Organization`. The dot before the table name is part of the model.

C.0.3 Sector order

The appendix is built in dependency order, not alphabetically.

APPENDIX SECTION	SCHEMA	ROLE
C.1	`CorePlatform`	Shared platform reference data: units, settings, language, versioning
C.2	`Security`	Identity, roles, permissions, policies
C.3	`Organization`	Sites, plants, departments, teams, people
C.4	`ReactorFleet`	Fleet, units, reactors, core, rods, steam generators, plant equipment
C.5	`Instrumentation`	Tags, signals, sensors, historian readings, signal quality
C.6	`DigitalTwin`	Twin models, variables, bindings, snapshots, divergences
C.7	`AlarmManagement`	Alarm definitions, alarm events, floods, acknowledgements
C.8	`EventManagement`	Incidents, operational events, near misses
C.9	`Maintenance`	Assets, inspections, degradation, work orders
C.10	`RootCause`	Causal graph, hypotheses, evidence, rejected candidates
C.11	`ReinforcementLearning`	Q-table, states, actions, training runs, episode logs
C.12	`Robotics`	Robots, vehicles, missions, capabilities, readiness
C.13	`RadiationMonitoring`	Radiation zones, monitors, dose readings, dosimeters
C.14	`EmergencyPreparedness`	Emergency plans, drills, exercises, evacuation routes
C.15	`Compliance`	Regulations, controls, audits, findings, evidence
C.16	`Reporting`	Report definitions, dashboards, exports, generated snapshots
C.17	`Audit`	Change log, history, lineage, append-only audit records

C.0.4 Standard mutable table columns

Most mutable business tables use the following lifecycle columns.

```
CreatedAtUtc     DATETIME2(7) NOT NULL
    CONSTRAINT DF_<Table>_CreatedAtUtc DEFAULT SYSUTCDATETIME(),
CreatedBy        NVARCHAR(100) NOT NULL,
ModifiedAtUtc    DATETIME2(7) NOT NULL
    CONSTRAINT DF_<Table>_ModifiedAtUtc DEFAULT SYSUTCDATETIME(),
ModifiedBy       NVARCHAR(100) NOT NULL,
IsDeleted        BIT NOT NULL
    CONSTRAINT DF_<Table>_IsDeleted DEFAULT 0,
RowVersion       ROWVERSION NOT NULL
```

These columns are not repeated in every explanatory paragraph, but they are included in the DDL where appropriate.

`IsDeleted` is not used for every table. It belongs on mutable operational records where the row should disappear from normal application queries without being physically removed. It does not belong on immutable facts such as telemetry samples, alarm events, audit entries, or historical snapshots.

C.0.5 Standard lookup table shape

Lookup tables are typed. There is no generic `Lookup` table. Each category receives its own table so that foreign keys remain enforceable.

```
CREATE TABLE <Schema>.<LookupName> (
    <LookupName>Id INT IDENTITY(1,1) NOT NULL,
    Code           NVARCHAR(50) NOT NULL,
    Name           NVARCHAR(150) NOT NULL,
    Description    NVARCHAR(500) NULL,
    DisplayOrder   INT NOT NULL
        CONSTRAINT DF_<LookupName>_DisplayOrder DEFAULT 0,
    IsActive       BIT NOT NULL
        CONSTRAINT DF_<LookupName>_IsActive DEFAULT 1,
    CreatedAtUtc   DATETIME2(7) NOT NULL
        CONSTRAINT DF_<LookupName>_CreatedAtUtc DEFAULT SYSUTCDATETIME(),
    ModifiedAtUtc  DATETIME2(7) NOT NULL
        CONSTRAINT DF_<LookupName>_ModifiedAtUtc DEFAULT SYSUTCDATETIME(),
    RowVersion     ROWVERSION NOT NULL,

    CONSTRAINT PK_<Schema>_<LookupName>
        PRIMARY KEY (<LookupName>Id),

    CONSTRAINT UQ_<Schema>_<LookupName>_Code
        UNIQUE (Code)
);
```

Examples

```
ReactorFleet.UnitStatus
ReactorFleet.ReactorType
Instrumentation.SignalType
Instrumentation.SignalQuality
AlarmManagement.AlarmSeverity
AlarmManagement.AlarmState
RootCause.CausalEdgeType
RootCause.HypothesisStatus
ReinforcementLearning.TrainingRunStatus
Robotics.AssetStatus
RadiationMonitoring.DoseUnitType
Compliance.FindingSeverity
```

A two-value fact that will never be joined to, such as true/false, remains a `BIT`. A value that is displayed, ordered, translated, joined to, seeded, or audited becomes a lookup table.

C.0.6 Primary key conventions

Most entity tables use a surrogate integer key. High-volume append-only fact tables use `BIGINT IDENTITY`. Join tables use a composite primary key when the pair is the identity of the row.

```
-- Normal entity key
UnitId INT IDENTITY(1,1) NOT NULL

-- High-volume append-only fact key
MeasurementId BIGINT IDENTITY(1,1) NOT NULL
AlarmEventId  BIGINT IDENTITY(1,1) NOT NULL
AuditEntryId  BIGINT IDENTITY(1,1) NOT NULL

-- Composite key for a pure join table
CONSTRAINT PK_Security_RolePermission
    PRIMARY KEY (RoleId, PermissionId)
```

Join tables receive their own identity key only when the relationship carries a lifecycle, status, history, comments, or audit trail of its own.

C.0.7 Foreign key conventions

Foreign keys are always explicit and named.

```
CONSTRAINT FK_Instrumentation_Signal_EngineeringUnit
    FOREIGN KEY (EngineeringUnitId)
    REFERENCES CorePlatform.EngineeringUnit(EngineeringUnitId)
```

Cross-schema foreign keys are not avoided. They are the controlled way one sector references another. The dependency must be visible in the DDL, visible in the ER diagram, and listed in the foreign-key mapping table for the sector.

The default delete behaviour is `NO ACTION`. Cascade delete is reserved for dependent details whose existence has no meaning without the parent.

```
TrainingRun  -> Episode
ReportExport -> ReportExportFile
AuditBatch   -> AuditEntry
```

Business records, measurements, alarms, incidents, and evidence are not cascade-deleted by default.

C.0.8 Index conventions

Every foreign key receives an index unless it is already covered by a primary key or unique constraint. Common lookup tables receive a unique index or unique constraint on `Code`.

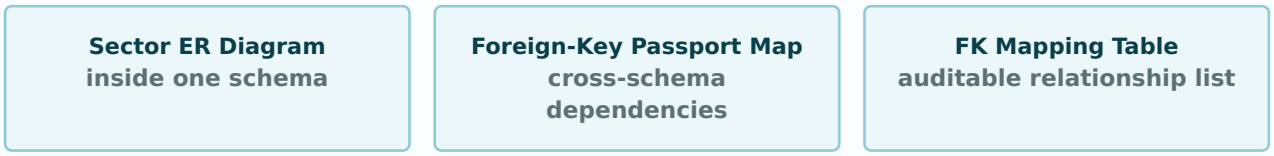
```
-- Every foreign key receives an index unless already covered.
CREATE INDEX IX_Instrumentation_Signal_UnitId
ON Instrumentation.Signal(UnitId);

-- High-volume fact table query pattern.
CREATE INDEX IX_Instrumentation_Measurement_SignalId_TimestampUtc
ON Instrumentation.Measurement(SignalId, TimestampUtc DESC);
```

Telemetry and audit tables may use columnstore indexes later in the performance chapter, but the base schema remains readable and normal first.

C.0.9 ER diagram convention

Each sector includes two diagrams. The first is the sector ER diagram. It shows the principal tables inside the schema and the most important relationships. The second is the foreign-key passport map. It shows cross-schema dependencies.



Every sector section follows the same evidence pattern: DDL, diagram, and mapping table.

```
solid arrow     same-schema relationship
dashed arrow    cross-schema foreign key
PK              primary key
FK              foreign key
LK              lookup table
FACT            append-only high-volume fact table
HIST            history or snapshot table
```

C.0.10 Master dependency map

At platform level, the strongest dependencies are shown below. The map is not a call graph. It is a referential-integrity map: the direction points from the table holding the foreign key to the table it depends on.

SECTOR	PRIMARY REFERENTIAL DEPENDENCIES
Security	CorePlatform
Organization	CorePlatform
ReactorFleet	Organization, CorePlatform
Instrumentation	ReactorFleet, CorePlatform
DigitalTwin	ReactorFleet, Instrumentation, CorePlatform
AlarmManagement	ReactorFleet, Instrumentation, Security
EventManagement	ReactorFleet, AlarmManagement, Security
Maintenance	ReactorFleet, Organization, Security
RootCause	ReactorFleet, Instrumentation, AlarmManagement, EventManagement
ReinforcementLearning	ReactorFleet, DigitalTwin
Robotics	ReactorFleet, RadiationMonitoring
RadiationMonitoring	ReactorFleet, Organization, CorePlatform
EmergencyPreparedness	ReactorFleet, Organization, RadiationMonitoring
Compliance	Organization, ReactorFleet, EventManagement, Audit
Reporting	All read-side sectors
Audit	All write-side sectors

C.0.11 Standing boundary

This schema is a software model for the NEXUS-1 companion system. It is designed to support a browser-based demonstrator, explanatory books, SQL Server implementation, EF Core reverse engineering, and a serious discussion of digital-twin data modelling.

- It is not a licensed nuclear plant schema.
- It is not safety-class software.
- It is not connected to any live plant.
- It does not claim to describe any real facility.

The engineering value is the structure: explicit ownership, typed lookups, visible foreign keys, auditable history, reproducible seeds, and one database backbone for the plant, the alarm flood, the root-cause engine, the Q-table, and the digital twin.