3Cloud

DATA MODELING **STANDARDS**

3CLOUD TECHNICAL GUIDE

DATA MODELING
STANDARDS & BEST PRACTICES

Contents

DATA MODELING
STANDARDS & BEST PRACTICES

Purpose

This guide captures foundational principles and proven best practices that drive effective data modeling on modern data platforms. It's designed to help teams build models that are consistent, high-performing and ready for enterprise scale. While these standards reflect our recommended approach, we recognize that flexibility is sometimes required to meet unique project demands.

Dimensional Modeling Standards

Fact Table Design

Granularity

When building fact tables, keep things focused and streamlined. The level of detail should match what's needed for reporting – no more, no less. And each fact table should represent a single business process to avoid confusion and ensure clean, accurate analysis.

QUICK TIPS

- Keep granularity aligned with reporting needs.
- Use one fact table per business process – avoid mixing unrelated event types.

Fact Table Columns

Fact tables should focus on capturing meaningful, numeric data that supports business analysis. They should primarily include additive measures – values you can sum or average – and use integer surrogate keys (like date keys) to link to dimension tables. These keys and measures should always be set as NOT NULL to ensure data integrity.

It's fine to include transaction-specific details (degenerate columns), especially in transactional or accumulating snapshot tables.[1] Descriptive attributes, however, should generally live in dimension tables. If a description only exists at the fact level, like a transaction note, it can be included sparingly. Keep fact tables narrow and efficient.

For frequently used metrics, pre-calculate and store aggregates like totals or averages to improve performance.

QUICK TIPS

- Focus on numeric, additive measures.
- Use integer surrogate keys and define them as NOT NULL.
- Include degenerate columns when needed for transactions.
- Avoid descriptive attributes unless they're only available at the fact level.
- Pre-calculate common aggregates to speed up queries.

1 Another Look at Degenerate Dimensions by Bob Becker, Kimball Group

DATA MODELING
STANDARDS & BEST PRACTICES

Partitioning
For large fact tables, consider partitioning[2] to enhance performance and maintainability. This can help improve query performance, speed up data loading and make ongoing maintenance easier. You can partition based on time intervals, key ranges or any other criteria that align with how the data is distributed and how it's typically accessed.

Fact Table General Design
Fact tables usually aren't very wide because they tend to hold a massive number of rows – keeping them narrow helps with performance and manageability.

Fact Table Types
There are three main types of fact tables, each designed for a different kind of business scenario:

Transaction tables are the most common. They capture individual business events – like a sale or a shipment – and are great for detailed, event-level analysis.

Accumulating snapshot tables are ideal for tracking progress through a process with defined steps, such as an order fulfillment workflow.

Periodic snapshot tables capture data at regular intervals (like daily or monthly), making them useful for trend analysis and point-in-time reporting.

Dimension Table Design
Dimension Table Columns
Dimension tables are where you store the descriptive details that help explain the facts in your fact tables. These attributes provide the context needed for meaningful analysis.

QUICK TIPS
- Include rich, descriptive fields that help users understand the data.
- Use surrogate keys to support historical tracking and make integration easier across systems.

2 How to Decide if You Should Use Table Partitioning by Kendra Little, Brent Ozar Unlimited

DATA MODELING
STANDARDS & BEST PRACTICES

Dimension Table Structure
To keep things efficient and user-friendly, dimension tables should be designed for performance and clarity.

QUICK TIPS
- Flatten the structure – denormalize where possible to reduce joins and speed up queries.
- To represent hierarchies, use a single table with parent-child or self-referencing keys.
- Use the right slowly changing dimension (SCD) type[3] – Type 1 to overwrite, Type 2 to add a new row or Type 3 to add a new column – based on how you need to track changes over time.

Many-to-Many Relationships

Bridge Table Rules
Bridge tables are used to manage many-to-many relationships[4,5] between tables. Getting the granularity right is key – not just for performance, but to make sure your results are accurate.

QUICK TIPS
- Omplement bridge tables exclusively for authentic many-to-many relationships.
- Create a separate bridge table for each of these relationships.

Bridge Table Columns
Bridge tables should include the necessary keys and any relevant details about the relationship.

QUICK TIPS
- Add foreign keys that point to the primary keys of the related tables.
- Include any extra columns needed to describe or measure the relationship.
- Use surrogate keys to support historical tracking and simplify integration.

Bridge Table General Design
Always document the purpose of each bridge table – what relationship it represents and which dimension tables it connects.

3 Explained: Slowly Changing Dimensions (SCD) by John Tringham, Medium

4 Building Bridges by Warren Thornthwaite, Kimball Group

5 How to handle Bridge table in Star Schema, Stack Overflow

DATA MODELING
STANDARDS & BEST PRACTICES

Data Types

General Guidelines

Choosing the right data types is key to performance, storage efficiency, and long-term flexibility.

- Select data type that align with the data's characteristics and intended use.
- Be mindful of size and precision – smaller types (like SMALLINT) can save space and improve speed.
- Avoid complex types that make querying harder.
- Use consistent data types across similar columns (e.g., always use BIT for IsActive flags).
- Regularly review data types to ensure they still meet business needs.
- Avoid tightening data type constraints without validating current and future requirements

Specific Standards

- Use VARCHAR instead of CHAR for text – VARCHAR only uses the space it needs.
- Use NVARCHAR or NCHAR only when multilingual support is required.
- For most technical fields like EmailAddress or UserName, stick with VARCHAR.
- Use BIT for binary or flag-type values (e.g., IsActive). Avoid using numeric types for these.

Key Columns

SURROGATE KEYS

- Use auto-incrementing integers (INT or BIGINT) as unique identifiers.
- Choose BIGINT if the table could exceed 1 billion rows.
- Avoid using GUIDs unless global uniqueness is required.

BUSINESS KEYS

These are unique identifiers from the source system (e.g., CustomerNumber, ProjectNumber) used in ETL/ELT to detect new or updated records.

GENERAL RULE

Avoid string-based key columns; prefer integers to ensure consistency and performance.

DATA MODELING
STANDARDS & BEST PRACTICES

Constraints
Applying the right constraints helps ensure your data stays clean, consistent and reliable.

- Set default values and constraints to handle missing or invalid data.
- Use **primary key** constraints to enforce uniqueness within fact and dimension tables.
- Use **foreign key** constraints to maintain relationships between tables.
- Apply **NOT NULL** constraints to columns that should always have a value.
- Use **unique** constraints to prevent duplicate entries where they aren't allowed.

Naming Conventions

General Naming Best Practices
Clear, consistent naming makes your data model easier to understand, maintain, and scale. Here are some simple do's and don'ts to follow:

DO THE FOLLOWING
- Use descriptive, meaningful names for tables, columns, keys and indexes.
- Stick to a consistent naming style – CamelCase is a common choice.
- Apply the same naming rules across all database objects.

DON'T DO THE FOLLOWING
- Use plurals (e.g., use Customer, not Customers) – unless it makes more sense contextually (like Series
- Use past tense words unless business rules require them.
- Use abbreviations or contractions – full words are clearer and more universal.
- Use special characters (like underscores or hyphens) CamelCase handles word separation cleanly
- Use keywords used by your database system (e.g., avoid naming a table ***State*** in SQL Server).

DATA MODELING
STANDARDS & BEST PRACTICES

Tables

Table Names

In dimensional modeling, there are three main types of tables, each with a clear naming convention to keep things consistent and easy to understand:

Fact Tables: Start with Fact followed by a singular noun that describes the subject.
Example: FactInvoice for invoice data.

Dimension Tables: Start with Dim followed by the subject of the dimension.
Example: DimEmployee for employee-related attributes.

Bridge Tables: Start with Bridge, followed by the names of the two tables it connects.
Example: BridgeOrderVendor for a many-to-many relationship between orders and vendors.

If the bridge connects a fact and a dimension table, list the fact table name first. For bridges between two dimension tables, order doesn't matter – but it's best to avoid this setup when possible.

Column Names

As a general rule, use a consistent pattern: the table name followed by Key for all primary, foreign and surrogate keys. For example, in a table called *FactOrder*, the surrogate key should be *OrderKey*. If it references *DimCustomer*, the foreign key should be *CustomerKey*.

FACT TABLES

- Start with the surrogate or primary key.
- Follow with foreign keys, ideally sorted alphabetically unless there's a business reason not to.
- Add a few descriptive columns if needed – keep them minimal and sorted alphabetically.
- Then include your numeric measures or metrics (e.g., *INT, DECIMAL*).
- End with auditing columns like *CreatedDate* or *ModifiedBy*.

DIMENSION TABLES

- Begin with the surrogate or primary key.
- Avoid foreign keys if possible to maintain a star schema. If needed, place them next and sort alphabetically.
- Add descriptive columns next, sorted alphabetically.
- Place Boolean or flag columns (like *IsActive*) toward the end – they're useful but not very readable.
- Finish with auditing columns.
- Don't store measures in dimension tables.

BRIDGE TABLES

- Start with the surrogate key, which also acts as the primary key.
- Follow with foreign keys.
- If needed, include descriptive columns that explain the relationship.
- Measures are rare here, but if used, place them after descriptors.
- Wrap up with auditing columns.

DATA MODELING
STANDARDS & BEST PRACTICES

Constraint Names

Clear, consistent constraint names make data models easier to read, understand, and maintain. Here are some common constraint types and suggested naming patterns to keep things clean:

PRIMARY KEY CONSTRAINTS

For primary keys, it's helpful to start the name with *PK*, followed by the table name. Using underscores keeps things readable – especially when names get long.

Example: A primary key on the *FactOrder* table would be named *PK_FactOrder*.

FOREIGN KEY CONSTRAINTS

For foreign keys, use *FK* as the prefix, followed by the primary key table name and then the foreign key table name – separated by underscores for clarity.

Example: A foreign key from *FactOrder* to *DimVendor* would be named *FK_FactOrder_DimVendor*.

DEFAULT CONSTRAINTS

Default constraints start with *DF*, followed by the table name and the column name – separated with underscores. This helps clearly identify where the default value applies.

Example: For an *IsActive* column in the *DimOrderDetail* table, you'd use *DF_DimOrderDetail_IsActive*.

CHECK CONSTRAINTS

Check constraints use the *CK* prefix, followed by the table name and column name, with underscores separating each part. This helps clarify what's being validated.

Example: A check on the *TransactionAmount* column in *FactAccounting* would be named *CK_FactAccounting_TransactionAmount*.

UNIQUE CONSTRAINTS

Use *UQ* to start, then add the table name and the column name that must remain unique – again, separated with underscores for readability.

Example: To enforce uniqueness on the *CustomerNumber* column in *DimCustomer*, name the constraint *UQ_DimCustomer_CustomerNumber*.

AI adoption helps eCommerce teams save an average of

6.4 HOURS PER WEEK*

In November-December 2024, AI influenced the purchase of

$299B IN HOLIDAY SHOPPING SPEND**

* eCommerce Trends (2024) Salesforce

** Holiday Shoppers Spend a Record $1.2T Online Salesforce

DATA MODELING
STANDARDS & BEST PRACTICES

Medallion Architecture[6]

Medallion Architecture doesn't replace traditional dimensional modeling. The schemas and tables within each layer can vary in structure and normalization – depending on how often the data is updated and how it's being used downstream. Different use cases call for different approaches.

Bronze Layer

Often called the Raw layer, this layer captures data exactly as it comes from the source – unprocessed, unmodified and in its original form.

- No modeling standards are applied here. The structure reflects the source system.
- Each record is tagged with metadata like ingestion time and source system details.
- Think of it as a snapshot of your source data, ready for downstream processing.

ARCHIVE STORAGE TIP:
Keep a copy of the raw data for auditing or future reference. This usually involves two steps:

- Landing zone: where incoming data temporarily lives before processing.
- Archive zone: where files are stored after successful ingestion.

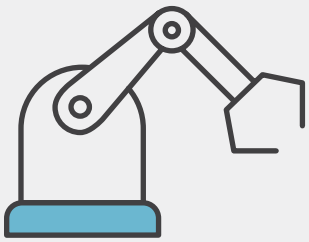
Silver Layer

The Silver layer is where things start to take shape. It's the transformative stage of the Medallion Architecture – where raw data becomes structured, cleaner and more usable.

- Data from the Bronze layer is sanitized.
- Column names are standardized for consistency across datasets.
- This is where business logic is applied turning raw inputs into meaningful outputs.

There's no strict modeling requirement here. The Silver layer offers flexibility to choose the structure that best fits your needs.

MODELING TIP:
While third normal form (3NF) is sometimes used, it's not always ideal – especially if the raw data is flat. Over-normalizing here can make querying more complex and performance-heavy.



6 What is the Medallion Lakehouse Architecture? Microsoft.com

DATA MODELING
STANDARDS & BEST PRACTICES

Gold Layer
The Gold layer follows the same modeling standards as a traditional dimensional model. For guidance, refer to the Dimensional Modeling Standards section of this guide.

Data Modeling in a Medallion Architecture
Not all data pipelines require every layer of the Medallion architecture. Bronze, Silver and Gold are a helpful way to frame the flow of data – based on the classic concepts of Stage, ODS and Data Warehouse.

The best design depends on your specific needs, the structure and variety of your data sources and the intended use cases. The Medallion approach is used here because it's widely understood and easy to translate across different architectures.

Below is an outline of common architecture types and when to use each.

Medallion Architecture Use Case Scenarios

Bronze > Gold (skipping Silver)
Not every data pipeline needs all three layers – Bronze, Silver and Gold. In some cases, the Silver layer (typically used for cleaning and standardizing data) can be skipped altogether. This approach works best when the data is already in good shape or when speed and simplicity are top priorities.

WHY SKIP THE SILVER LAYER?
Simplified Data Pipeline: Fewer stages mean less complexity. Eliminating Silver makes the pipeline easier to build, manage, and deploy quickly.

Cost Efficiency: Less storage, fewer transformations and reduced compute usage = lower costs.

Faster Access to Data: Users can tap into Bronze directly for certain use cases – especially when speed matters more than structure.

More Flexibility: When the data quality is already high or only minimal transformation is needed, skipping Silver offers more room to customize processing steps.

WHEN SKIPPING SILVER MAKES SENSE
Real-Time Data Processing
- Bronze holds raw IoT or streaming data
- Skipping Silver reduces latency
- Gold handles real-time transformation and analytics

High-Quality Source Data
- Data arrives in Bronze already cleaned and structured
- Silver layer is redundant
- Gold uses it directly for analytics/reporting

DATA MODELING
STANDARDS & BEST PRACTICES

Data Exploration & Prototyping
- Common in data science and ML scenarios
- Bronze provides raw data for exploration
- Gold layer houses refined datasets for model testing

Minimal Transformation Requirements
- Requirements don't call for cleaning or standardization
- Bronze = raw, untouched data
- Gold applies light transformation for reporting/insights

Historical Data Archiving
- Used when your want to preserve large volumes of data
- Bronze acts as long-term raw storage
- Gold does minimal refinement for historical analytics

Bronze > Silver > Gold

When end-to-end data refinement is needed, following the full Bronze–Silver–Gold structure offers clarity, control and consistency. Each layer plays a distinct role in transforming raw source data into analytics-ready insights.

WHEN TO USE A FULL MEDALLION ARCHITECTURE

In many cases, data is landed directly into the Bronze layer. From there, a full Medallion Architecture – Bronze, Silver and Gold – offers major advantages for data quality, structure and usability across teams.

WHY USE A FULL MEDALLION STRUCTURE?

Business Intelligence & Reporting: When clients need self-service dashboards or production-ready reports with trusted, curated data

Machine Learning & Advanced Analytics: When building predictive models using a mix of structured and unstructured data from multiple sources.

Data Warehousing: When there's a need to integrate various data sources into a unified, analytics-ready layer.

WHAT EACH LAYER DOES

Bronze Layer
- Raw, unprocessed data
- Landed directly from source systems
- Stored in original format for traceability

Silver Layer
- Cleaned, structured data
- Errors fixed, duplicates removed
- Standardized for cross-source consistency

Gold Layer
- Refined and aggregated
- Optimized for reporting, dashboards, and analytics
- Prepared for insights and modeling

DATA MODELING
STANDARDS & BEST PRACTICES

Raw > Bronze > Silver > Gold

Introducing a Raw layer before Bronze adds extra flexibility and control – especially when data lineage, auditing or reprocessing are priorities.

USE CASE: Enhanced Data Lineage & Traceability

= RAW ======
- Captures data exactly as it arrives—no transformations
- Supports multiple formats (CSV, JSON, Parquet, XML)
- Acts as a backup and historical archive for reprocessing

= BRONZE ======
- Light processing; primarily metadata tagging
- Often stored in Delta format for ACID compliance
- Captures ongoing changes from source systems
- Forms the base for further refinement

= SILVER ======
- Cleaned and transformed
- Improves quality and ensures consistency across sources

= GOLD ======
- Refined and analytics-ready
- Optimized for business intelligence, reporting, and modeling

USE CASE: Improved Data Quality & Governance

Incremental Refinement: Each layer improves quality step-by-step
Compliance-Friendly: Maintains clear lineage for audits and regulatory needs

REAL STORY. REAL RESULTS.

A not-for-profit healthcare network partnered with 3Cloud to modernize its data infrastructure, support ambitious growth goals and enable self-service analytics. **THE RESULT?** A secure Azure-based data platform that automated reporting with Power BI, enhanced compliance and established a governed, enterprise-wide data marketplace

READ MORE ›

DATA MODELING
STANDARDS & BEST PRACTICES

USE CASE: Flexibility & Scalability
- Each layer scales independently
- Easily adapts to new data sources and business requirements

USE CASE: Optimized Performance
Faster Queries: Gold is optimized for analytics
Resource Allocation: Raw data stays separate from refined layers for better performance and cost control

5+ Layers (Beyond Raw, Bronze, Silver and Gold)
While the traditional Medallion Architecture includes up to four layers, some scenarios benefit from going beyond – adding more stages for greater control, flexibility and precision..

USE CASE: Enhanced Data Quality & Validation
Extra layers can be used for targeted tasks like validation, enrichment, normalization or anonymization – ensuring data meets specific standards before progressing.

USE CASE: Data Governance & Compliance
- More layers = more transparency.
- Each step adds traceability, access control and auditability – key for regulated industries or sensitive data environments.

USE CASE: Scalability & Performance Optimization
Distributing workloads across more stages helps balance compute resources and enables parallel processing, improving efficiency and scale.

USE CASE: Flexibility & Adaptability
A modular, multi-layer design makes it easier to plug in new data sources or adapt to evolving business needs. Layers can be customized for specific workflows or team responsibilities.

Principles for Everyday Lakehouse

Store Raw Data Immutably
Keeping raw data immutable means it can't be changed after it's stored – which is key for trust, traceability and compliance – and helps build a foundation that's both trustworthy and future-proof.

- **Audit-Ready:** Immutable data ensures a reliable record that can't be tampered with.
- **Versioning:** Many systems support version history, so you can go back to earlier states if needed.
- **Compliance-Friendly:** Required for meeting many regulatory and security standards.
- **Error Recovery:** If something goes wrong downstream, the original data is always preserved.

DATA MODELING
STANDARDS & BEST PRACTICES

Metadata Tagging
Metadata tagging helps users quickly locate, understand, and trust the data they're working with

- **Improves discoverability** by adding context to datasets
- **Supports governance** by tracking lineage and enforcing compliance
- **Boosts data quality** by helping identify issues related to accuracy, completeness and consistency
- **Provides transparency** into where data came from and how it's been processed

Encrypt or Mask Sensitive Data
Protecting sensitive data is critical for compliance and risk management.

- **Encryption** secures data both at rest and in transit
- **Masking** limits access to sensitive information based on user roles
- **Compliance-ready** for regulations like HIPAA, GDPR and CCPA
- **Minimizes risk** of data breaches and associated financial or reputational harm

Enforce a Schema
Schema enforcement ensures data consistency and reliability from the moment it lands.

- **Validates** that incoming data matches expected types and structure
- **Prevents errors** from malformed or inconsistent data
- **Ensures** all data follows the same format for easier processing and integration

De-Duplication
Removing duplicate records improves accuracy, saves space and speeds up performance.

- **Increases data reliability** for analytics and reporting
- **Reduces storage costs** by eliminating redundancy
- **Improves efficiency** by minimizing the volume of data to process

Standardized Format
Using consistent data formats across platforms helps ensure reliability and compatibility.

- **Eases data sharing** across tools and systems
- **Maintains data quality** by enforcing structure
- **Reduces processing errors** and irregularities

DATA MODELING
STANDARDS & BEST PRACTICES

Data Quality Checks

Ongoing validation ensures that data remains trustworthy and ready for use.

- **Detects and corrects errors** before they impact downstream systems
- **Improves consistency** across datasets from multiple sources
- **Reduces reprocessing costs** by catching quality issues early

Mapping & Modeling Data

Mapping and modeling bring structure, logic and efficiency to your data workflows.

- **Mapping transforms raw data** into usable formats and captures lineage
- **Modeling defines how data is organized** and stored
- **Improves query performance** through optimized design
- **Encapsulates business logic**, enabling deeper analysis and insight generation

Data Modeling References

Microsoft Training Module: Introduction to Foundations in Data Modeling:
learn.microsoft.com/en-us/training/modules/modern-analytics-data-modeling

Dimensional Modeling Techniques:
kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques