# Asset Management Database Schemas

## Overview

This document describes the database schemas for the Asset Management system, including tables, relationships, indexes, and constraints designed to support comprehensive asset financial tracking, lifecycle management, and AI-powered cost optimization.

## Schema Architecture

### 🏗️ Design Principles

* **Performance Optimized**: Strategic indexes on all query-critical fields
* **Data Integrity**: Enum constraints and unique constraints prevent invalid data
* **Audit Trail**: User tracking for all modifications with timestamps
* **Scalability**: Efficient data types and normalized structure
* **Flexibility**: JSON metadata fields for extensible data storage
* **AI Ready**: Schema designed to support machine learning algorithms

## Tables

### 💰 asset\_cost\_management

**Purpose**: Tracks all financial costs associated with assets including purchases, maintenance, licenses, and operational expenses.

**Columns**:

id SERIAL PRIMARY KEY  
cost\_type enum\_asset\_cost\_management\_cost\_type NOT NULL  
amount DECIMAL(15,2) NOT NULL  
currency VARCHAR(3) DEFAULT 'USD'  
billing\_cycle enum\_asset\_cost\_management\_billing\_cycle DEFAULT 'one\_time'  
start\_date TIMESTAMPTZ  
end\_date TIMESTAMPTZ  
vendor VARCHAR(255)  
contract\_number VARCHAR(255)  
purchase\_order VARCHAR(255)  
invoice\_number VARCHAR(255)  
cost\_center VARCHAR(255)  
budget\_code VARCHAR(255)  
notes TEXT  
attachments JSONB DEFAULT '[]'  
metadata JSONB DEFAULT '{}'  
created\_by INTEGER REFERENCES users(id)  
last\_modified\_by INTEGER REFERENCES users(id)  
created\_at TIMESTAMPTZ DEFAULT NOW() NOT NULL  
updated\_at TIMESTAMPTZ DEFAULT NOW() NOT NULL  
asset\_uuid UUID

**Indexes**: - idx\_asset\_cost\_management\_asset\_uuid - Asset lookups - idx\_asset\_cost\_management\_cost\_type - Cost type filtering - idx\_asset\_cost\_management\_vendor - Vendor analysis - idx\_asset\_cost\_management\_cost\_center - Cost center reporting - idx\_asset\_cost\_management\_created\_at - Time-based queries - idx\_asset\_cost\_management\_amount - Cost range queries

**Enums**: - cost\_type: purchase, lease, maintenance, support, license, subscription, upgrade, repair, insurance, other - billing\_cycle: one\_time, monthly, quarterly, semi\_annual, annual, biennial

### 🔄 asset\_lifecycle

**Purpose**: Manages asset lifecycle information including purchase dates, warranties, EOL dates, and replacement planning.

**Columns**:

id SERIAL PRIMARY KEY  
purchase\_date DATE  
warranty\_end\_date DATE  
manufacturer\_eol\_date DATE  
internal\_eol\_date DATE  
replacement\_cycle\_months INTEGER  
estimated\_replacement\_cost DECIMAL(15,2)  
replacement\_budget\_year INTEGER  
replacement\_budget\_quarter INTEGER  
replacement\_notes TEXT  
created\_at TIMESTAMPTZ DEFAULT NOW() NOT NULL  
updated\_at TIMESTAMPTZ DEFAULT NOW() NOT NULL  
asset\_uuid UUID

**Indexes**: - idx\_asset\_lifecycle\_asset\_uuid - Asset lookups - idx\_asset\_lifecycle\_warranty\_end\_date - Warranty expiration tracking - idx\_asset\_lifecycle\_internal\_eol\_date - EOL planning - idx\_asset\_lifecycle\_replacement\_budget\_year - Budget planning - idx\_asset\_lifecycle\_purchase\_date - Age calculations

**Constraints**: - asset\_lifecycle\_asset\_uuid\_unique - One lifecycle record per asset

### 💡 asset\_operational\_costs

**Purpose**: Tracks monthly operational costs for assets including power, space, network, storage, and labor costs.

**Columns**:

id SERIAL PRIMARY KEY  
year\_month DATE NOT NULL  
power\_cost DECIMAL(15,2)  
space\_cost DECIMAL(15,2)  
network\_cost DECIMAL(15,2)  
storage\_cost DECIMAL(15,2)  
labor\_cost DECIMAL(15,2)  
other\_costs DECIMAL(15,2)  
notes TEXT  
created\_at TIMESTAMPTZ DEFAULT NOW() NOT NULL  
updated\_at TIMESTAMPTZ DEFAULT NOW() NOT NULL  
asset\_uuid UUID

**Indexes**: - idx\_asset\_operational\_costs\_asset\_uuid - Asset lookups - idx\_asset\_operational\_costs\_year\_month - Time-based queries - idx\_asset\_operational\_costs\_asset\_year\_month - Composite queries

**Constraints**: - asset\_operational\_costs\_asset\_year\_month\_unique - Prevents duplicate monthly records

### 🎯 asset\_risk\_mapping

**Purpose**: Maps assets to risk models and cost centers with confidence scoring for AI optimization.

**Columns**:

id SERIAL PRIMARY KEY  
asset\_uuid UUID  
existing\_asset\_id INTEGER  
risk\_model\_id INTEGER  
cost\_center\_id INTEGER  
mapping\_confidence DECIMAL(3,2) DEFAULT 0.85  
mapping\_method enum\_asset\_risk\_mapping\_method DEFAULT 'automatic'  
mapping\_criteria JSONB  
verified\_by INTEGER REFERENCES users(id)  
verified\_at TIMESTAMPTZ  
created\_at TIMESTAMPTZ DEFAULT NOW()  
updated\_at TIMESTAMPTZ DEFAULT NOW()

**Indexes**: - idx\_asset\_risk\_mapping\_asset\_uuid - Asset lookups - idx\_asset\_risk\_mapping\_existing\_asset - Legacy asset mapping - idx\_asset\_risk\_mapping\_risk\_model\_id - Risk model queries - idx\_asset\_risk\_mapping\_cost\_center\_id - Cost center analysis - idx\_asset\_risk\_mapping\_mapping\_method - Method filtering - idx\_asset\_risk\_mapping\_mapping\_confidence - Confidence-based queries - idx\_asset\_risk\_mapping\_verified\_by - Verification tracking

**Constraints**: - asset\_risk\_mapping\_asset\_uuid\_existing\_\_key - Unique asset-to-existing mapping

**Enums**: - mapping\_method: automatic, manual, hybrid

### 📁 asset\_groups

**Purpose**: Organizes assets into logical groups for management and reporting purposes.

**Columns**:

id SERIAL PRIMARY KEY  
name VARCHAR(255) NOT NULL  
description TEXT  
group\_type VARCHAR(100)  
parent\_group\_id INTEGER REFERENCES asset\_groups(id)  
metadata JSONB DEFAULT '{}'  
created\_by INTEGER REFERENCES users(id)  
created\_at TIMESTAMPTZ DEFAULT NOW() NOT NULL  
updated\_at TIMESTAMPTZ DEFAULT NOW() NOT NULL

**Indexes**: - idx\_asset\_groups\_name - Name-based lookups - idx\_asset\_groups\_group\_type - Type filtering - idx\_asset\_groups\_parent\_group\_id - Hierarchical queries

### 👥 asset\_group\_members

**Purpose**: Defines membership relationships between assets and groups.

**Columns**:

id SERIAL PRIMARY KEY  
group\_id INTEGER REFERENCES asset\_groups(id) ON DELETE CASCADE  
asset\_uuid UUID NOT NULL  
added\_by INTEGER REFERENCES users(id)  
added\_at TIMESTAMPTZ DEFAULT NOW() NOT NULL

**Indexes**: - idx\_asset\_group\_members\_group\_id - Group membership queries - idx\_asset\_group\_members\_asset\_uuid - Asset group lookups

**Constraints**: - asset\_group\_members\_group\_asset\_unique - Prevents duplicate memberships

## Relationships

### 🔗 Foreign Key Relationships

asset\_cost\_management.created\_by → users.id  
asset\_cost\_management.last\_modified\_by → users.id  
asset\_risk\_mapping.verified\_by → users.id  
asset\_groups.created\_by → users.id  
asset\_groups.parent\_group\_id → asset\_groups.id  
asset\_group\_members.group\_id → asset\_groups.id  
asset\_group\_members.added\_by → users.id

### 🔄 Asset UUID References

All tables reference assets via asset\_uuid field, providing loose coupling with the main assets table while maintaining referential integrity through application logic.

## Performance Optimization

### 📈 Index Strategy

1. **Primary Indexes**: All tables have optimized primary key indexes
2. **Foreign Key Indexes**: All foreign keys have corresponding indexes
3. **Query Indexes**: Strategic indexes on frequently queried columns
4. **Composite Indexes**: Multi-column indexes for complex queries
5. **Unique Indexes**: Enforce business rules and prevent duplicates

### 🚀 Query Performance Features

* **Selective Indexing**: Only critical query paths are indexed
* **Composite Indexes**: Support complex filtering scenarios
* **Partial Indexes**: Future optimization opportunity for filtered queries
* **JSON Indexing**: JSONB fields support efficient JSON operations

## Data Integrity

### ✅ Constraints

1. **Primary Keys**: All tables have auto-incrementing primary keys
2. **Foreign Keys**: Referential integrity with users and self-references
3. **Unique Constraints**: Prevent logical duplicates
4. **Check Constraints**: Data validation at database level
5. **Not Null**: Critical fields cannot be null

### 🔒 Enum Constraints

* **Cost Types**: Standardized cost categorization
* **Billing Cycles**: Consistent billing period definitions
* **Mapping Methods**: Controlled risk mapping methodology

## AI & Analytics Support

### 🤖 AI-Optimized Design

1. **Numerical Precision**: DECIMAL(15,2) for accurate financial calculations
2. **Time Series Data**: Optimized for temporal analysis
3. **Confidence Scoring**: Built-in confidence metrics for ML algorithms
4. **Metadata Storage**: Flexible JSON fields for ML feature storage
5. **Audit Trail**: Complete change tracking for model training

### 📊 Analytics Features

* **Cost Aggregation**: Efficient sum/average calculations
* **Time-based Analysis**: Optimized for trend analysis
* **Categorical Analysis**: Enum-based grouping and filtering
* **Confidence Metrics**: Built-in scoring for AI recommendations

## Migration & Deployment

### 🚀 Database Setup

# Generate migration SQL  
npm run generate:asset-management-migration  
  
# Test schema structure  
npm run test:asset-management-schemas  
  
# Apply migration (example)  
psql -d your\_database -f migration\_asset\_management.sql

### 🔧 Maintenance

1. **Regular VACUUM**: Maintain table performance
2. **Index Monitoring**: Monitor index usage and performance
3. **Statistics Updates**: Keep query planner statistics current
4. **Constraint Validation**: Periodic constraint validation

## Security Considerations

### 🔐 Access Control

* **Row-Level Security**: Can be implemented for multi-tenant scenarios
* **Column Permissions**: Sensitive financial data protection
* **Audit Logging**: Complete change tracking with user attribution
* **Data Encryption**: Sensitive fields can be encrypted at application level

### 🛡️ Data Protection

* **Input Validation**: Enum constraints prevent invalid data
* **SQL Injection**: Parameterized queries required
* **Data Masking**: Financial data can be masked in non-production environments

## Best Practices

### 📋 Development Guidelines

1. **Always use transactions** for multi-table operations
2. **Validate enum values** at application level
3. **Use prepared statements** for all queries
4. **Monitor query performance** regularly
5. **Keep statistics updated** for optimal query plans

### 🎯 Operational Guidelines

1. **Regular backups** of financial data
2. **Monitor disk usage** for JSONB fields
3. **Archive old data** based on retention policies
4. **Monitor index bloat** and rebuild as needed
5. **Validate data integrity** periodically

## Testing

### 🧪 Schema Validation

# Test schema exports and structure  
npm run test:asset-management-schemas

### 📊 Performance Testing

* **Load Testing**: Validate performance under load
* **Query Analysis**: EXPLAIN ANALYZE for critical queries
* **Index Usage**: Monitor index hit ratios
* **Constraint Performance**: Validate constraint checking performance

This schema design provides a robust foundation for enterprise asset management with built-in support for AI-powered cost optimization and comprehensive financial tracking.