03_FOUNDATION_GUIDE.md

Week 1: Foundation & Architecture - Complete Execution Guide
BAT Integration Project

Consolidates: Phase_1_Foundation_Integration_Plan.md,
Phase_1_Quick_Start_Checklist.md, BAT_MIGRATION_QUICK_START.md
Created: November 10, 2025
Last Updated: November 10, 2025
Version: 2.0
Status: Active - Week 1 Primary Guide

🎯 WEEK 1 OVERVIEW

Purpose
Build the architectural foundation that determines project success. Make three critical decisions that lock in the database structure, coding standards, and import strategy for the entire 12-week project.

Timeline
Duration: 5 days (November 11-15, 2025)
Investment: 18 hours total (enhanced from 14 hours)
Daily Commitment: 4 hours/day (Monday-Thursday), 2 hours Friday

Why This Week is Critical
- ❗ These decisions affect all future work (Weeks 2-12)
- ❗ Wrong choices = 4-6 weeks of rework later
- ❗ Right choices = smooth execution for 11 weeks
- ❗ Can't start Week 5 imports without clear target structure
- ❗ Foundation locks after Friday - changes require formal review

What Makes Week 1 Different
Monday (Complete ☑): Analysis and documentation
Tuesday (CRITICAL 🟤): Architecture decisions that determine everything
Wednesday-Thursday: Documentation based on decisions
Friday: Team validation and lock foundation

📋 DAILY BREAKDOWN

MONDAY (November 11) - Audit Day ☑ COMPLETE

Morning Session (2 hours): Item Numbering Audit
Goal: Document current numbering patterns in both systems

Richmond Analysis
```
Tasks:
□ Open sample plan sheets: G603, G914, LE93 G603B, LE94 G603A
□ Check Material Database sheets: Combined_A_to_G, RAH SKUs
□ Document current numbering patterns
  ├─ Check for 6-digit hierarchical codes
  ├─ Identify prefixes and categories
  ├─ Note any inconsistencies
  └─ Take screenshots of key areas
□ Analyze structure:
  ├─ First 2 digits = category?
  ├─ Next 2 digits = subcategory?
  └─ Last 2 digits = sequence?

Output: Richmond item numbering section
```

Holt Analysis
```
Tasks:
□ Open sample plan sheets: 1670ABCD CR, 1890ABD CR, 2321ABC CR
□ Extract 50+ item codes from plan sheets
□ Decode 9-digit pattern: [Plan 4][Pack 1][Category 2][Sequence 2]
□ Document pack types:
  ├─ 1 = Foundation
  ├─ 2 = Framing
  ├─ 3 = Exterior
  ├─ 4-9 = Other categories
□ Map elevation encoding:
  ├─ 100 = Elevation A
  ├─ 200 = Elevation B
  ├─ 300 = Elevation C
  ├─ 400 = Elevation D
□ Take detailed notes with examples

Output: Holt item numbering section
```

Deliverable: ☑ item_numbering_patterns.txt (combined report, 746 items analyzed)

Afternoon Session (2 hours): Richmond Structure Audit
Goal: Map Richmond pricing infrastructure for Week 2 updater

Pricing Sheet Analysis
```
Tasks:
```

```
□ Open Richmond BAT "PRICING TAB" sheet
□ Map all column headers
□ Identify price levels (expecting L1-L5)
□ Document IWP RS structure (RS = Random Sizes)
□ Document IWP S4S structure (S4S = Surfaced 4 Sides)
□ Analyze RL+ADDERS sheet
  ├─ RL = Random Length pricing
  ├─ Freight calculations
  ├─ Margin calculations
□ Review RL_AV sheet (historical price tracking)
□ Check Customer Price Levels sheet
□ Screenshot key formula cells
  ├─ How base costs stored
  ├─ How freight calculated
  ├─ How margins applied
□ Check for VBA macros affecting pricing
□ Document plan/elevation/option relationships

Output: richmond_structure.txt (20 pages)
```

Key Questions to Answer:
- How many price levels? (L1-L5 expected)
- Where are base costs stored?
- How is freight calculated?
- How are margins applied?
- Any VBA macros affecting pricing?
- How do elevations encode? (discovered triple-encoding)

Deliverable: ☑ richmond_structure.txt

Monday Summary: ☑ 45 pages documentation created, foundational analysis complete

---

TUESDAY (November 12) - Architecture Day 🟢 CRITICAL

Why Tuesday is the Most Important Day
This day determines project success. The three architecture decisions you make today will either:
- ☑ Enable smooth execution for 11 weeks, OR
- ❌ Cause 4-6 weeks of rework starting in Week 5

Take your time. Get it right. This is not a day to rush.

Morning Session (2 hours): Map Hierarchies

SESSION 1.1: Richmond Hierarchy Analysis (1 hour)
Goal: Understand how Richmond encodes plans, elevations, packs, and options

```
Questions to Answer:
□ How do plans encode?
  ├─ G603 (plan only?)
  ├─ G603B (plan + elevation?)
  ├─ LE93 G603B (community + plan + elevation?)
  └─ What's the pattern?

□ How do elevations work?
  ├─ Are they variants (G603B is a separate plan)?
  ├─ OR dimensions (G603 with elevation B)?
  ├─ Where does elevation appear?
  │  ├─ In sheet names?
  │  ├─ In Plan Index?
  │  ├─ In material codes?
  │  └─ In pack names?

□ Triple-encoding problem:
  Pack: |10.82BCD OPT DEN FOUNDATION
  ├─ "BCD" in pack name (encoding 1)
  ├─ "- ELVB - ELVC - ELVD" in location (encoding 2)
  ├─ "ELVB, ELVC, ELVD" in option codes (encoding 3)
  └─ Which is source of truth?

□ How do options relate to plans?
  ├─ XGREAT (extended great room) - universal?
  ├─ 2CAR5XA (2-car garage 5' ext elev A) - plan-specific?
  └─ Pattern for option codes?

□ How do packs relate to plans?
  ├─ When pack "12.x5" appears on G603 and G914...
  ├─ Are the materials identical (universal)?
  ├─ Or different per plan (plan-specific)?
  └─ Test with real data

□ Current table relationships:
  ├─ What sheets reference what?
  ├─ How are lookups structured?
  └─ Any circular references?
```

Output: richmond_hierarchy_map.txt (visual diagram with relationships)

SESSION 1.2: Holt Hierarchy Analysis (1 hour)
Goal: Understand how Holt encodes plans, elevations, communities, and packs

```
Questions to Answer:
□ How do plans encode?
  ├─ 1670 (plan only?)
  ├─ 1670ABCD (plan with all elevations?)
  ├─ 1670ABCD CR (plan + elevations + community?)
  └─ What's the standard pattern?

□ How do elevations work?
  ├─ Elevation A, B, C, D = variants?
  ├─ OR plan 1670 with dimension "elevation"?
  ├─ Encoded in 9-digit codes (100, 200, 300, 400)
  └─ Relationship to sheet structure?

□ How do communities fit?
  ├─ CR, GG, HA, HH, WR
  ├─ Do communities affect materials?
  ├─ OR just job/pricing attribute?
  ├─ Same plan in different communities = same materials?
  └─ Community-specific pricing only?

□ How does pack system work?
  ├─ Pack 10.82 (den foundation)
  ├─ Pack 12.x5 (garage 5' extension)
  ├─ Universal across plans?
  ├─ OR plan-specific?
  └─ Test with real data

□ Current table relationships:
  ├─ bidtotals_[plan]_[community]_[elevation]
  ├─ materialist_[plan]_[community]_[elevation]
  ├─ How do tables reference each other?
  └─ Lookup patterns?
```

Output: holt_hierarchy_map.txt (visual diagram with relationships)

Afternoon Session (4 hours): Critical Architecture Decisions

SESSION 2.1: DECISION 1 - Plan-Pack Relationship (1 hour)

THE QUESTION:
When pack "12.x5" (2-car garage 5' extension) appears on multiple plans like G603 and G914, are the materials identical (Universal Pack) or different per plan (Plan-Specific Pack)?

THE OPTIONS:

Option A: Universal Packs
```
Database Model:
packs table:
  - pack_id (primary key)
  - pack_name
  - description

materials table:
  - material_id
  - pack_id (foreign key to packs)
  - item_number
  - quantity

Query: "Show all plans using pack 12.x5"
SELECT DISTINCT plan_id FROM plan_packs WHERE pack_id = '12.x5'
```

Pros:
- ☑ Simpler data model
- ☑ Less data duplication
- ☑ Easier to update pack globally
- ☑ Clear when packs are truly identical
- ☑ Smaller database size

Cons:
- ❌ Can't handle plan-specific variations
- ❌ Less flexible for customization
- ❌ May not match reality (packs might differ)

Option B: Plan-Specific Packs
```
Database Model:
packs table:
  - pack_id (primary key)
  - plan_id (foreign key)
  - pack_name
  - description

materials table:
  - material_id
  - pack_id (foreign key to packs)
  - item_number
  - quantity

Query: "Show pack 12.x5 for plan G603"
SELECT * FROM packs WHERE pack_name = '12.x5' AND plan_id = 'G603'
```

```

**Pros:**
- ☑ Handles all variations
- ☑ Maximum flexibility
- ☑ Matches physical reality (each plan may differ)
- ☑ Can customize per plan

**Cons:**
- ❌ More complex data model
- ❌ More data duplication
- ❌ Harder to maintain consistency
- ❌ Larger database size

---

**Option C: Hybrid Approach** (RECOMMENDED)
```
Database Model:
packs table:
  - pack_id (primary key)
  - pack_name
  - is_universal (boolean)
  - description

plan_packs table:
  - plan_pack_id (primary key)
  - plan_id (foreign key)
  - pack_id (foreign key)
  - override_materials (boolean)

materials table:
  - material_id
  - plan_pack_id (foreign key)
  - item_number
  - quantity

Logic:
IF pack is universal:
  - All plans share same materials
ELSE:
  - Each plan can have specific materials
```

**Pros:**
- ☑ Best of both worlds
- ☑ Universal packs where appropriate
- ☑ Plan-specific packs where needed
- ☑ Flexibility for future
- ☑ Matches reality
```

Cons:
- ❌ Slightly more complex
- ❌ Requires clear universal vs specific rules

HOW TO DECIDE:

1. **Test with real data (15 min):**
```
□ Pick pack "12.x5" (2-car garage 5' extension)
□ Find it on plan G603
□ Find it on plan G914
□ Compare materials line by line
□ Are they identical? Different?
□ Document findings
```

2. **Test with another pack (15 min):**
```
□ Pick pack "10.82" (optional den foundation)
□ Find it on multiple plans
□ Compare materials
□ Pattern the same?
```

3. **Make decision (15 min):**
```
□ Based on testing, which model fits reality?
□ Consider future: Manor Homes integration?
□ Choose: A, B, or C
□ Document rationale
```

4. **Document (15 min):**
```
Create DECISION_1_Plan_Pack_Relationship.md:
1. The Question
2. The Options (A, B, C with pros/cons)
3. Testing Results (what we found)
4. Recommendation (which we chose)
5. Rationale (why we chose it)
6. Examples (show it working)
7. Edge Cases (what might break)
```

Output: DECISION_1_Plan_Pack_Relationship.md

SESSION 2.2: DECISION 2 - Plan-Elevation Model (1 hour)

THE QUESTION:
Is "G603B" one plan, or is it Plan G603 with Elevation B? How do we fix the triple-encoding problem where elevation appears in pack names, location strings, AND option codes?

THE PROBLEM:
Current state shows elevation encoded 3 times:
```
Pack name: |10.82BCD OPT DEN FOUNDATION
Location:  - ELVB - ELVC - ELVD
Options:   ELVB, ELVC, ELVD

Which is source of truth? How do we prevent inconsistency?
```

THE OPTIONS:

Option A: Elevation as Variant
```
Database Model:
plans table:
  - plan_id (primary key): "G603", "G603B", "G603C", "G914A"
  - base_plan: "G603", "G603", "G603", "G914"
  - elevation: NULL, "B", "C", "A"

materials table:
  - material_id
  - plan_id: "G603B" (complete plan code)
  - item_number
  - quantity

Query: "Show all elevations of G603"
SELECT * FROM plans WHERE base_plan = 'G603'
```

Pros:
- ☑ Matches current sheet names (LE93 G603B)
- ☑ Simple queries
- ☑ Each elevation is distinct entity

Cons:
- ❌ Doesn't solve triple-encoding
- ❌ Still duplicates elevation data
- ❌ Hard to query "show all A elevations"

Option B: Elevation as Dimension (RECOMMENDED)
```
Database Model:
plans table:
  - plan_id (primary key): "G603", "G914", "1670"
  - plan_name
  - builder_id

plan_elevations table:
  - elevation_id (primary key)
  - plan_id (foreign key)
  - elevation_code: "A", "B", "C", "D"
  - elevation_name: "Elevation A", etc.

packs table:
  - pack_id: "10.82"
  - pack_name: "OPT DEN FOUNDATION"
  (NO elevation in name!)

pack_elevations table:
  - pack_id (foreign key)
  - elevation_code: "B", "C", "D"
  (which elevations this pack applies to)

materials table:
  - material_id
  - plan_id: "G603" (NOT "G603B")
  - elevation_id (foreign key)
  - pack_id
  - item_number
  - quantity

Query: "Show all B elevations across all plans"
SELECT * FROM plan_elevations WHERE elevation_code = 'B'

Query: "Show packs for G603 Elevation B"
SELECT packs.*
FROM packs
JOIN pack_elevations ON packs.pack_id = pack_elevations.pack_id
WHERE pack_elevations.elevation_code = 'B'
```

Pros:
- ☑ SOLVES triple-encoding (single source of truth)
- ☑ Elevation stored once in dedicated table
- ☑ Can query across elevations easily
- ☑ Cleaner data model
- ☑ Easier to maintain

Cons:

- ❌ Requires more joins in queries
- ❌ Slightly more complex

HOW TO DECIDE:

1. **Analyze current usage (15 min):**
```
□ How do users refer to plans?
  - "Plan G603B" (variant)?
  - "Plan G603, elevation B" (dimension)?
□ Check customer contracts
□ Ask William/Alicia preference
```

2. **Test queries (15 min):**
```
□ Write query for Option A approach
□ Write query for Option B approach
□ Which is clearer?
□ Which performs better?
```

3. **Consider triple-encoding fix (15 min):**
```
□ Does Option A solve it? (No)
□ Does Option B solve it? (Yes - pack_elevations table)
□ Future-proof?
```

4. **Document (15 min):**
```
Create DECISION_2_Plan_Elevation_Model.md:
1. The Problem (triple-encoding)
2. The Options (A vs B with pros/cons)
3. Testing Results
4. Recommendation (likely Option B)
5. Rationale (solves triple-encoding)
6. Examples (show clean queries)
7. Migration Plan (how to import with this model)
```

Output: DECISION_2_Plan_Elevation_Model.md

SESSION 2.3: DECISION 3 - Internal Option Codes (1 hour)

THE QUESTION:

Richmond uses descriptive codes (XGREAT, 2CAR5XA). Holt uses numeric codes (167010100). What should OUR internal standard be? Do we keep one, merge both, or create something new?

THE OPTIONS:

Option A: Keep Richmond Codes
```
Standard: Descriptive alphanumeric
Examples: XGREAT, 2CAR5XA, FPSING01

Pros:
☑ Human-readable
☑ Self-documenting
☑ Easy for team to remember
☑ Current Richmond team familiar

Cons:
❌ Holt team needs translation
❌ Requires translation table Holt→Richmond
❌ May conflict with future codes
❌ Harder to enforce format rules
```

Option B: Keep Holt Codes
```
Standard: Numeric hierarchical
Examples: 167010100, 164910105, 189010400

Pattern: [Plan 4][Phase 2][Option 2][Elevation 2]

Pros:
☑ Systematic and structured
☑ Plan number embedded
☑ Phase grouping built-in
☑ Current Holt team familiar

Cons:
❌ Richmond team needs translation
❌ Requires translation table Richmond→Holt
❌ Not human-readable
❌ Long codes (9 digits)
```

Option C: Hybrid/Translation
```
```

```
Keep both systems:
- Richmond BAT uses Richmond codes
- Holt BAT uses Holt codes
- Translation table bridges them

translation_table:
  - richmond_code: "XGREAT"
  - holt_code: "167010600"
  - description: "Extended Great Room"
  - category: "Interior"

Pros:
☑ No retraining needed
☑ Both teams use familiar codes
☑ Preserves current workflows
☑ Can query by either code

Cons:
❌ Maintains two systems
❌ Translation overhead
❌ Potential for sync errors
❌ More complex
```

Option D: New Universal System (RECOMMENDED for long-term)
```
Create new standard: OPT-[Category]-[Number]

Examples:
- OPT-GAR-001 (Garage: 2-car 4' ext)
- OPT-GAR-002 (Garage: 2-car 5' ext)
- OPT-INT-001 (Interior: Extended great room)
- OPT-STR-001 (Structural: Optional den)
- OPT-EXT-001 (Exterior: Sunroom)

Categories:
- GAR = Garage options
- INT = Interior options
- STR = Structural options
- EXT = Exterior options
- ELE = Electrical options
- PLU = Plumbing options

Pros:
☑ Clean, systematic
☑ Category grouping clear
☑ Short codes (10-11 characters)
☑ Easy to extend
```

☑ Neutral (neither Richmond nor Holt)
☑ Manor Homes can adopt same system

Cons:
❌ Both teams need retraining
❌ Requires translation from both
❌ Migration effort
❌ Change management needed
```

---

**HOW TO DECIDE:**

1. **Consider merger timeline (10 min):**
```
□ March 2026 merger = 4 months away
□ Is retraining feasible?
□ Or maintain dual system temporarily?
□ Long-term: Manor Homes integration
```

2. **Evaluate translation complexity (20 min):**
```
□ How many option codes exist?
  - Richmond: ~200 codes
  - Holt: ~300 codes
□ How often added?
□ Can translation be automated?
□ Maintenance burden?
```

3. **Consider user experience (15 min):**
```
□ William preference?
□ Alicia preference?
□ Which is most intuitive?
□ Training required for each option?
```

4. **Make recommendation (15 min):**
```
Short-term (Weeks 1-12):
Recommend Option C (Hybrid/Translation)
- Both teams keep familiar codes
- Translation table bridges gap
- No retraining during migration

Long-term (Post-merger):
Recommend Option D (New Universal System)
```
```

```
- Gradual migration to OPT-[CAT]-[NUM]
- Phase in over 6-12 months
- Manor Homes adopts new standard
```

5. **Document:**
```
Create DECISION_3_Internal_Option_Codes.md:
1. The Question
2. The Options (A, B, C, D with pros/cons)
3. Analysis (translation complexity, user impact)
4. Recommendation (C short-term, D long-term)
5. Rationale (why phased approach)
6. Migration Plan (how to transition)
7. Translation Table Strategy
```

Output: DECISION_3_Internal_Option_Codes.md

Evening Session (2 hours): Database Schema Design

SESSION 3: Design Database Schema

Goal: Create schema_design_v1.sql based on the three decisions made today

Core Tables (10 tables):

```sql
-- DECISION 1 IMPACT: Plan-Pack relationship
-- Using Hybrid approach

-- Table 1: builders
CREATE TABLE builders (
    builder_id TEXT PRIMARY KEY,  -- 'RICHMOND', 'HOLT'
    builder_name TEXT NOT NULL,
    is_active INTEGER DEFAULT 1   -- SQLite boolean (0/1)
);

-- Table 2: plans (DECISION 2: Plan without elevation)
CREATE TABLE plans (
    plan_id TEXT PRIMARY KEY,      -- 'G603', '1670', '1890'
    builder_id TEXT NOT NULL,
    plan_name TEXT,
    sq_ft INTEGER,
    is_active INTEGER DEFAULT 1,
    FOREIGN KEY (builder_id) REFERENCES builders(builder_id)
);
```

```
-- Table 3: plan_elevations (DECISION 2: Elevation as dimension)
CREATE TABLE plan_elevations (
    elevation_id INTEGER PRIMARY KEY,
    plan_id TEXT NOT NULL,
    elevation_code TEXT NOT NULL,  -- 'A', 'B', 'C', 'D'
    elevation_name TEXT,            -- 'Elevation A'
    is_active INTEGER DEFAULT 1,
    FOREIGN KEY (plan_id) REFERENCES plans(plan_id),
    UNIQUE(plan_id, elevation_code)
);

-- Table 4: packs (DECISION 1 & 2: No elevation in pack, hybrid model)
CREATE TABLE packs (
    pack_id TEXT PRIMARY KEY,      -- '10.82', '12.x5'
    pack_name TEXT NOT NULL,       -- 'OPT DEN FOUNDATION'
    phase TEXT,                     -- '10', '12'
    is_universal INTEGER DEFAULT 0, -- DECISION 1: Hybrid flag
    is_active INTEGER DEFAULT 1
);

-- Table 5: pack_elevations (DECISION 2: Which elevations pack applies to)
CREATE TABLE pack_elevations (
    pack_elevation_id INTEGER PRIMARY KEY,
    pack_id TEXT NOT NULL,
    elevation_code TEXT NOT NULL,  -- 'B', 'C', 'D'
    FOREIGN KEY (pack_id) REFERENCES packs(pack_id),
    UNIQUE(pack_id, elevation_code)
);

-- Table 6: plan_packs (DECISION 1: Hybrid junction)
CREATE TABLE plan_packs (
    plan_pack_id INTEGER PRIMARY KEY,
    plan_id TEXT NOT NULL,
    pack_id TEXT NOT NULL,
    override_materials INTEGER DEFAULT 0, -- Plan-specific override?
    FOREIGN KEY (plan_id) REFERENCES plans(plan_id),
    FOREIGN KEY (pack_id) REFERENCES packs(pack_id),
    UNIQUE(plan_id, pack_id)
);

-- Table 7: items (material catalog)
CREATE TABLE items (
    item_id TEXT PRIMARY KEY,      -- SKU or item number
    item_description TEXT NOT NULL,
    category TEXT,                  -- 'Lumber', 'Trusses', etc.
    unit_of_measure TEXT,           -- 'EA', 'LF', 'SQFT'
    is_active INTEGER DEFAULT 1
);

-- Table 8: materials (DECISION 1 & 2: Links everything)
```

```
CREATE TABLE materials (
    material_id INTEGER PRIMARY KEY,
    plan_pack_id INTEGER NOT NULL, -- DECISION 1: Hybrid
    elevation_id INTEGER,           -- DECISION 2: Dimension
    item_id TEXT NOT NULL,
    quantity REAL NOT NULL,
    unit_cost REAL,
    notes TEXT,
    FOREIGN KEY (plan_pack_id) REFERENCES plan_packs(plan_pack_id),
    FOREIGN KEY (elevation_id) REFERENCES plan_elevations(elevation_id),
    FOREIGN KEY (item_id) REFERENCES items(item_id)
);

-- Table 9: pricing (price levels and history)
CREATE TABLE pricing (
    price_id INTEGER PRIMARY KEY,
    item_id TEXT NOT NULL,
    price_level TEXT NOT NULL,     -- 'L1', 'L2', 'PL01', 'PL12'
    price REAL NOT NULL,
    effective_date TEXT,            -- ISO8601: '2025-11-10'
    end_date TEXT,
    FOREIGN KEY (item_id) REFERENCES items(item_id)
);

-- Table 10: communities (Holt-specific)
CREATE TABLE communities (
    community_id TEXT PRIMARY KEY, -- 'CR', 'GG', 'WR'
    community_name TEXT NOT NULL,
    builder_id TEXT NOT NULL,
    is_active INTEGER DEFAULT 1,
    FOREIGN KEY (builder_id) REFERENCES builders(builder_id)
);

-- PRISM SQL MIGRATION NOTES:
/*
SQLite → Prism/PostgreSQL Conversions:

1. INTEGER PRIMARY KEY → SERIAL PRIMARY KEY
2. REAL → DECIMAL(10,2)
3. TEXT → VARCHAR(n) or TEXT
4. INTEGER (0/1 boolean) → BOOLEAN
5. TEXT (ISO8601 date) → TIMESTAMP

Example Prism conversion:
CREATE TABLE builders (
    builder_id VARCHAR(50) PRIMARY KEY,
    builder_name VARCHAR(200) NOT NULL,
    is_active BOOLEAN DEFAULT TRUE
);
```

```
Migration script template: See MIGRATION_SQLITE_TO_PRISM.sql
*/
```

Additional Supporting Tables (optional for v1):

```sql
-- Table 11: option_translation (DECISION 3: Hybrid approach)
CREATE TABLE option_translation (
    translation_id INTEGER PRIMARY KEY,
    richmond_code TEXT,             -- 'XGREAT', '2CAR5XA'
    holt_code TEXT,                 -- '167010600'
    universal_code TEXT,            -- 'OPT-INT-001' (future)
    description TEXT NOT NULL,
    category TEXT,                  -- 'GAR', 'INT', 'STR'
    is_active INTEGER DEFAULT 1,
    UNIQUE(richmond_code),
    UNIQUE(holt_code)
);

-- Table 12: pack_hierarchy (MindFlow structure)
CREATE TABLE pack_hierarchy (
    hierarchy_id INTEGER PRIMARY KEY,
    pack_id TEXT NOT NULL,
    parent_pack_id TEXT,
    display_order INTEGER,
    FOREIGN KEY (pack_id) REFERENCES packs(pack_id),
    FOREIGN KEY (parent_pack_id) REFERENCES packs(pack_id)
);
```

Test Queries:

```sql
-- Query 1: Show all materials for Plan G603, Elevation B
SELECT
    p.plan_id,
    pe.elevation_code,
    pk.pack_name,
    i.item_description,
    m.quantity,
    m.unit_cost
FROM materials m
JOIN plan_packs pp ON m.plan_pack_id = pp.plan_pack_id
JOIN plans p ON pp.plan_id = p.plan_id
JOIN plan_elevations pe ON m.elevation_id = pe.elevation_id
JOIN packs pk ON pp.pack_id = pk.pack_id
JOIN items i ON m.item_id = i.item_id
WHERE p.plan_id = 'G603'
  AND pe.elevation_code = 'B';
```

```
-- Query 2: Show all plans using pack 12.x5
SELECT DISTINCT p.plan_id, p.plan_name
FROM plan_packs pp
JOIN plans p ON pp.plan_id = p.plan_id
WHERE pp.pack_id = '12.x5';

-- Query 3: Translate Richmond option to Holt
SELECT holt_code, description
FROM option_translation
WHERE richmond_code = 'XGREAT';
```

Deliverables:
- schema_design_v1.sql (complete schema with Prism notes)
- Test queries demonstrating functionality
- Comments explaining each decision's impact

WEDNESDAY (November 13) - Documentation Day

Morning Session (2 hours): Draft Coding Standards Part 1

Goal: Create comprehensive coding standards document based on Tuesday's decisions

Create BAT_Coding_Standards.docx

Section 1: Introduction & Philosophy (15 min)
```
□ Purpose of coding standards
□ Why standards matter
□ How standards enable:
  ├─ Consistent data entry
  ├─ Easy querying
  ├─ System maintenance
  └─ Team collaboration
□ Learning-first approach explanation
```

Section 2: Plan Coding (20 min)
```
Based on DECISION 2:

Format: [BUILDER_PREFIX][NUMBER][VARIANT?]

Richmond Examples:
- G603 (plan without elevation)
- G914 (plan without elevation)
```

```
- LE93 (community prefix + plan)

Holt Examples:
- 1670 (plan without elevation)
- 1890 (plan without elevation)
- 2676 (plan without elevation)

Validation Rules:
- 4-6 characters
- Alphanumeric only
- No spaces
- Case-sensitive: Use uppercase

Correct: G603, G914, 1670, LE93
Incorrect: g603 (lowercase), G 603 (space), G603B (elevation in plan_id)

Rationale: Based on Decision 2 (Elevation as Dimension),
elevation is NOT part of plan_id
```

Section 3: Elevation Coding (20 min)
```
Based on DECISION 2:

Storage: Separate dimension in plan_elevations table

Format: Single letter A, B, C, or D

Examples:
- Plan G603 + Elevation B
- Plan 1670 + Elevation A

Database:
- plans.plan_id = 'G603'
- plan_elevations.elevation_code = 'B'

NOT stored as: G603B (this is old triple-encoding problem)

Validation Rules:
- Must be A, B, C, or D
- Case-sensitive: uppercase only
- One character only

Sheet Naming (for reference):
- materialist_G603_B (plan_id + elevation_code)
- bidtotals_1670_CR_A (plan + community + elevation)
```

Section 4: Pack Coding (20 min)
```
```

```
Format: |[PHASE].[VARIANT] [DESCRIPTION] - [OPTION_CODE]

Examples:
- |10 FOUNDATION
- |10.82 OPT DEN FOUNDATION
- |12.x5 OPT 2 CAR GARAGE 5' EXT FOUNDATION

Phase Numbers (typical):
- |09 = Basement/walkout
- |10 = Foundation
- |11 = Joist system
- |12 = Garage
- |13+ = Framing and beyond

Pack Types (for database):
1 = Foundation
2 = Framing
3 = Exterior
4 = Interior
5 = Garage
6 = Electrical
7 = Plumbing
8 = HVAC
9 = Special

Validation Rules:
- Starts with | (pipe character)
- Phase: 2 digits
- Variant: optional, format .XX or .xX
- Description: clear, concise
- Option code: optional, after dash

Based on DECISION 1 (Hybrid):
- Universal packs: Same materials across plans
- Plan-specific packs: Materials vary by plan
- Flag: is_universal in database
```

Section 5: Material Item Numbering (25 min)
```
Richmond System: 6-digit hierarchical
Format: [CATEGORY 2][SUBCATEGORY 2][SEQUENCE 2]

Examples:
- 404001 (Lumber)
- 404027 (Lumber)
- 404028 (Trusses)

Validation Rules:
- Exactly 6 digits
```

```
- No letters
- No special characters

Holt System: 9-digit hierarchical
Format: [PLAN 4][PACK 1][CATEGORY 2][SEQUENCE 2]

Examples:
- 167010504 (Plan 1670, Pack 1, Category 05, Sequence 04)
- 189010105 (Plan 1890, Pack 1, Category 01, Sequence 05)

Elevation Encoding (in 9-digit system):
- XXX100XX = Elevation A
- XXX200XX = Elevation B
- XXX300XX = Elevation C
- XXX400XX = Elevation D

Validation Rules:
- Exactly 9 digits
- Plan portion matches actual plan
- Pack type 1-9
- Valid elevation code (100, 200, 300, 400)

Translation:
Both systems valid. Translation table bridges them.
See option_translation table in database.
```

Deliverable: BAT_Coding_Standards.docx (Sections 1-5 complete)

Afternoon Session (2 hours): Draft Coding Standards Part 2

Section 6: Option Codes (30 min)
```
Based on DECISION 3:

SHORT-TERM STANDARD (Weeks 1-12): Hybrid
- Richmond uses: XGREAT, 2CAR5XA, FPSING01
- Holt uses: 167010100, 164910105
- Translation: option_translation table

Richmond Format:
- Descriptive alphanumeric
- Variable length (6-10 characters)
- Self-documenting

Examples:
XGREAT = Extended great room
2CAR5XA = 2-car garage, 5' extension, elevation A
```

FPSING01 = Fireplace single, option 01

Holt Format:
- Numeric hierarchical
- 9 digits: [PLAN 4][PHASE 2][OPTION 2][ELEVATION 2]

Examples:
167010600 = Plan 1670, Phase 01, Option 06, All elevations
164910105 = Plan 1649, Phase 01, Option 01, Elevation A (100=A)

Translation Example:
richmond_code: XGREAT
holt_code: 167010600
description: Extended Great Room
category: INT (Interior)

LONG-TERM STANDARD (Post-merger): Universal
Format: OPT-[CATEGORY]-[NUMBER]

Examples:
- OPT-GAR-001 = Garage option 001
- OPT-INT-001 = Interior option 001 (Extended great room)
- OPT-STR-001 = Structural option 001 (Optional den)

Categories:
- GAR = Garage options
- INT = Interior options
- STR = Structural options
- EXT = Exterior options
- ELE = Electrical options
- PLU = Plumbing options

Migration Path:
1. Weeks 1-12: Use hybrid/translation
2. Post-merger: Phase in OPT-[CAT]-[NUM]
3. Manor Homes: Adopt universal standard
```

**Section 7: Community Codes (15 min)**
```
Holt-Specific: 2-3 letter abbreviations

Active Communities:
- CR = [Full community name]
- GG = [Full community name]
- HA = [Full community name]
- HH = [Full community name]
- WR = [Full community name]

Database: communities table

- community_id (primary key)
- community_name
- builder_id = 'HOLT'

Usage:
Community is job-level attribute, not item-level
Same plan in different communities = same materials
Different pricing per community

Sheet Naming:
bidtotals_[PLAN]_[COMMUNITY]_[ELEVATION]
Example: bidtotals_1670_CR_A
```

**Section 8: Sheet/Table Naming Convention (30 min)**
```
Format: [TYPE]_[PLAN]_[COMMUNITY?]_[ELEVATION?]

Types:
- materialist = Material list for plan
- bidtotals = Bid totals for plan
- pricing = Pricing table
- schedule = Schedule/calendar
- ref = Reference data
- lookup = Lookup table

Richmond Examples:
- materialist_G603 (no elevation if base/all)
- materialist_G603_A (with elevation)
- pricing_base (shared)

Holt Examples:
- materialist_1670_CR_A (plan + community + elevation)
- bidtotals_1890_GG_B (plan + community + elevation)

Special Tables:
- PlanIndex (standard name)
- ItemPricing (standard name)
- PriceSchedule (standard name)
- ref_Communities (reference data)
- ref_PlanElevations (reference data)
- lookup_PackTypes (lookup table)

Validation Rules:
- Lowercase with underscores
- No spaces
- No special characters except underscore
- Consistent order: TYPE_PLAN_COMMUNITY_ELEVATION
```
```

Section 9: Validation Rules (15 min)
```
All Codes Must:
- Be unique within their scope
- Follow format rules exactly
- Be case-sensitive (uppercase standard)
- Contain no spaces
- Use only defined characters

Error Handling:
- Invalid codes rejected at entry
- Clear error messages
- Examples of correct format shown
- Validation before database insert

Examples:
Correct: G603, 1670, OPT-GAR-001, |10.82
Incorrect: g603, 1 670, OPT GAR 001, 10.82

Database Constraints:
- PRIMARY KEY enforces uniqueness
- CHECK constraints enforce format
- FOREIGN KEY maintains relationships
- NOT NULL prevents missing data
```

Section 10: Examples & Best Practices (20 min)
```
Good Examples:
☑ Plan: G603 (clear, follows standard)
☑ Elevation: B (single letter, uppercase)
☑ Pack: |10.82 (phase.variant format)
☑ Option: XGREAT (descriptive, memorable)
☑ Sheet: materialist_G603_B (follows convention)

Bad Examples:
❌ Plan: G603B (elevation embedded - old triple-encoding)
❌ Elevation: elev_B (extra text)
❌ Pack: 10.82 (missing pipe character)
❌ Option: xgreat (lowercase)
❌ Sheet: G603 Materials B (spaces, inconsistent)

Migration Notes:
- Old data may not follow standards
- Document exceptions
- Clean up gradually
- Validate on import

Best Practices:
1. Document as you go
```

2. Validate early and often
3. Get team input on ambiguous cases
4. Update standards when needed (formal process)
5. Train new team members using examples
```

**Deliverable:** BAT_Coding_Standards.docx (COMPLETE DRAFT)

---

## THURSDAY (November 14) - Refinement Day

### Morning Session (2 hours): Import Mapping Rules

**Goal:** Create field-by-field translation guide for import scripts

#### Create import_mapping_rules.md

```markdown
# IMPORT MAPPING RULES
## Richmond → Database

### Source: RAH_MaterialDatabase.xlsx
Sheet: Combined_A_to_G

Column Mapping:
- Column A (Plan) → plans.plan_id
- Column B (Elevation?) → plan_elevations.elevation_code
  (If embedded, extract)
- Column C (Pack) → packs.pack_id
  (Strip pipe character and variant)
- Column D (Item #) → items.item_id
- Column E (Description) → items.item_description
- Column F (Quantity) → materials.quantity
- Column G (Unit Cost) → materials.unit_cost
- Column H (Category) → items.category

Transformations:
1. Plan ID cleanup:
   - If "LE93 G603B" → plan_id = "G603", elevation = "B"
   - Extract community prefix if present

2. Pack ID cleanup:
   - If "|10.82BCD" → pack_id = "10.82"
   - Elevation codes "BCD" → pack_elevations entries

3. Elevation extraction:
   - Check plan name for elevation suffix
   - Check pack name for elevation codes
   - Create plan_elevations entries

````
Validation:
- All plans must exist in plans table
- All items must exist in items table
- Quantities must be > 0
- Costs must be numeric

Holt → Database

Source: MaterialDatabase.xlsx (Holt)
Sheets: Multiple per plan/community/elevation

Column Mapping:
- Plan (from sheet name) → plans.plan_id
- Community (from sheet name) → via plan_community junction
- Elevation (from sheet name) → plan_elevations.elevation_code
- 9-digit code (Column A) → Parse:
  - Digits 1-4 → Validate against plan_id
  - Digit 5 → packs.pack_type
  - Digits 6-7 → items.category
  - Digits 8-9 → sequence
  - Elevation encoding → plan_elevations.elevation_code
- Description → items.item_description
- Quantity → materials.quantity
- Unit Cost → materials.unit_cost

Transformations:
1. 9-digit code parsing:
   - 167010504 →
     plan: 1670, pack_type: 1, category: 05, seq: 04

2. Elevation from code:
   - XXX100XX → Elevation A
   - XXX200XX → Elevation B
   - XXX300XX → Elevation C
   - XXX400XX → Elevation D

3. Sheet name parsing:
   - "materialist_1670_CR_A" →
     plan_id: 1670, community: CR, elevation: A

Validation:
- 9-digit codes must be valid format
- Plan portion must match sheet name
- Elevation in code must match sheet elevation
- All items validated before insert
```

Deliverable: import_mapping_rules.md
````
````

Afternoon Session (2 hours): Refine Documentation

Tasks:

1. **Review all documents created (30 min):**
```
□ DECISION_1_Plan_Pack_Relationship.md
□ DECISION_2_Plan_Elevation_Model.md
□ DECISION_3_Internal_Option_Codes.md
□ schema_design_v1.sql
□ import_mapping_rules.md
□ BAT_Coding_Standards.docx (DRAFT)
```

2. **Add examples to each decision (30 min):**
```
□ Include real data examples
□ Show before/after scenarios
□ Add query examples
□ Include edge cases
```

3. **Review schema for completeness (30 min):**
```
□ All relationships defined?
□ All needed fields present?
□ Indexes identified?
□ Prism migration notes complete?
```

4. **Prepare team review materials (30 min):**
```
□ Create 1-page summary of decisions
□ Highlight key questions for team
□ Prepare examples for discussion
□ Create feedback capture form
```

Deliverable: All documents refined and ready for Friday review

---

FRIDAY (November 15) - Validation & Finalization Day

Morning Session (2 hours): Team Review

PREPARATION (before meeting):
```
```

□ Print/share all documents
□ Create agenda
□ Prepare examples
□ Set up feedback capture
```

**MEETING STRUCTURE (2 hours):**

**10 min: Overview**
```
□ Explain Week 1 purpose
□ Show what was accomplished
□ Preview the three decisions
□ Set expectations for review
```

**20 min: Hierarchy Maps**
```
With William (Richmond):
□ Review richmond_hierarchy_map.txt
□ Validate plan/elevation/pack relationships
□ Confirm option code usage
□ Identify any errors

With Alicia (Holt):
□ Review holt_hierarchy_map.txt
□ Validate community structure
□ Confirm 9-digit code understanding
□ Identify any errors
```

**40 min: Three Critical Decisions**
```
For each decision:

Decision 1: Plan-Pack Relationship (15 min)
□ Present the question
□ Show the options (Universal, Plan-Specific, Hybrid)
□ Present recommendation (Hybrid)
□ Show real data examples
□ Get feedback:
  - Do you agree?
  - Edge cases we missed?
  - Concerns?

Decision 2: Plan-Elevation Model (15 min)
□ Present triple-encoding problem
□ Show the options (Variant, Dimension)
□ Present recommendation (Dimension)
□ Show how it solves triple-encoding
```
```

```
□ Get feedback:
  - Does this match reality?
  - Any issues?

Decision 3: Internal Option Codes (10 min)
□ Present Richmond vs Holt codes
□ Show translation approach
□ Present recommendation (Hybrid short-term)
□ Get feedback:
  - Acceptable to both teams?
  - Translation concerns?
```

20 min: Database Schema Review
```
□ Show schema_design_v1.sql
□ Walk through key tables
□ Show example queries
□ Get feedback:
  - Can you query what you need?
  - Missing fields?
  - Performance concerns?
  - Prism migration questions?
```

20 min: Coding Standards Review
```
□ Show BAT_Coding_Standards.docx
□ Walk through each section
□ Show validation rules
□ Get feedback:
  - Are codes intuitive?
  - Will team adopt these?
  - Naming conflicts?
  - Training needed?
```

10 min: Capture Action Items
```
□ Document concerns raised
□ Identify changes needed
□ Assign priorities
□ Schedule follow-ups if needed
```

DELIVERABLE: team_review_feedback.txt

Afternoon Session (2 hours): Finalize & Publish

INCORPORATE FEEDBACK (45 min):
```
□ Update hierarchy maps if errors found
□ Revise decisions if concerns need addressing
□ Adjust schema based on feedback
□ Refine coding standards for clarity
□ Add team-suggested examples
```

CREATE FINAL DOCUMENTS (30 min):
```
□ DECISION_1_Plan_Pack_Relationship.md (FINAL)
□ DECISION_2_Plan_Elevation_Model.md (FINAL)
□ DECISION_3_Internal_Option_Codes.md (FINAL)
□ schema_design_FINAL.sql (with any adjustments)
□ BAT_Coding_Standards.docx (FINAL)
```

CREATE REFERENCE MATERIALS (30 min):
```
In Richmond BAT:
□ Add sheet "Coding Standards"
□ Add pack type lookup table
□ Add option category lookup table
□ Add elevation code reference
□ Add links to decision documents

In Holt BAT:
□ Add sheet "Coding Standards" (same content)
□ Add community code reference
□ Add 9-digit code explanation
□ Add links to decision documents
```

PUBLISH SUMMARY (15 min):
```
Create Phase_1_Foundation_Summary.md:

1. Executive Summary
   - What was decided
   - Why it matters
   - What it enables

2. Key Decisions
   - Decision 1: Hybrid Plan-Pack model
   - Decision 3: Elevation as Dimension
   - Decision 3: Hybrid Option Codes

3. Database Schema
```

```
   - 10 core tables
   - Key relationships
   - Query examples

4. Coding Standards
   - Plan format
   - Elevation format
   - Pack format
   - Option format
   - Validation rules

5. What's Next
   - Week 2: Pricing tools
   - Foundation is LOCKED
   - Changes require formal review
```

PREPARE FOR WEEK 2 (10 min):
```
□ Review Week 2 objectives (pricing infrastructure)
□ Ensure richmond_structure.txt is ready for updater
□ Note any open questions for resolution
□ Celebrate Week 1 completion! 🎉
```

📋 DELIVERABLES CHECKLIST

Week 1 Complete Deliverables

Audit Documents:
- [x] `item_numbering_patterns.txt` (746 items analyzed)
- [x] `richmond_structure.txt` (20 pages)
- [ ] `richmond_hierarchy_map.txt`
- [ ] `holt_hierarchy_map.txt`

Decision Documents:
- [ ] `DECISION_1_Plan_Pack_Relationship.md`
- [ ] `DECISION_2_Plan_Elevation_Model.md`
- [ ] `DECISION_3_Internal_Option_Codes.md`

Technical Specifications:
- [ ] `schema_design_v1.sql`
- [ ] `schema_design_FINAL.sql`
- [ ] `import_mapping_rules.md`

Standards Documents:
- [ ] `BAT_Coding_Standards.docx` (DRAFT)
- [ ] `BAT_Coding_Standards.docx` (FINAL)

- [ ] Reference sheets in Richmond BAT
- [ ] Reference sheets in Holt BAT

Process Documents:
- [ ] `team_review_feedback.txt`
- [ ] `Phase_1_Foundation_Summary.md`

Total: 16 documents

☑ SUCCESS CRITERIA

At end of Week 1, answer YES to all:

Strategic Questions:
- [ ] Do we know exactly how plans relate to elevations?
- [ ] Have we decided on universal vs plan-specific packs?
- [ ] Do we have clear option tracking methodology?
- [ ] Is the foundation architecture documented and approved?

Technical Questions:
- [ ] Can we write queries to retrieve any plan's materials?
- [ ] Does the schema support multiple price levels?
- [ ] Can we track pricing changes over time?
- [ ] Are all foreign key relationships defined?
- [ ] Do we have Prism SQL migration path documented?

Operational Questions:
- [ ] Will the team understand the coding system?
- [ ] Can we import Richmond data without structure changes?
- [ ] Can we import Holt data without structure changes?
- [ ] Have we identified all edge cases?

Documentation Questions:
- [ ] Can a new team member understand the design?
- [ ] Are all decisions explained with rationale?
- [ ] Do we have examples for all code formats?
- [ ] Is the foundation locked and approved?

All YES? → Week 1 SUCCESS! 🎉

⚠ COMMON PITFALLS TO AVOID

1. Rushing the Decisions
```
❌ Skip pros/cons analysis to save time
☑ Take time to consider all implications
```

☑ Test with real data before deciding
☑ Document reasoning thoroughly
```

### 2. Incomplete Documentation
```
❌ Just list facts without explaining WHY
☑ Include rationale for every decision
☑ Add examples (good and bad)
☑ Show edge cases and how to handle them
```

### 3. Ignoring Team Feedback
```
❌ Dismiss concerns as "not understanding"
☑ Listen carefully to William and Alicia
☑ Their concerns are data points
☑ Incorporate feedback before finalizing
```

### 4. Over-Complicating Schema
```
❌ Design for every possible future feature now
☑ Start simple, add complexity as needed
☑ Focus on core functionality first
☑ Note future enhancements for later phases
```

### 5. Vague Validation Rules
```
❌ "Code must be numeric"
☑ "Code must be exactly 6 digits, 0-9 only"
☑ Define error messages
☑ Show examples of valid and invalid
```

### 6. Forgetting Learning-First
```
❌ Just document WHAT the rules are
☑ Explain WHY we chose this approach
☑ Make documents teach, not just list
☑ Think: "Will this help future me?"
```

---

## 🔥 IF YOU GET STUCK

### Decision Paralysis?
```

```
1. Document the dilemma
2. List all options with pros/cons
3. Test with real data (15 min)
4. Schedule 30-min call with William/Alicia
5. Make decision and document reasoning
```

Too Much Information?
```
1. Take a break (15 min)
2. Create 1-page summary of what you learned
3. Focus on the key question
4. Simplify the options
5. Move forward
```

Conflicting Data?
```
1. Note the conflict
2. Determine which source is authoritative
3. Document the conflict in decision doc
4. Use most recent/reliable source
5. Flag for team review
```

Team Disagreement?
```
1. Document both perspectives
2. Understand root concerns
3. Look for compromise
4. Escalate to manager if needed
5. Don't stall - make a decision
```

Technical Complexity?
```
1. Simplify - MVP first
2. Note future enhancements
3. Get basic functionality working
4. Test with sample data
5. Iterate as needed
```

🎯 IMMEDIATE NEXT STEPS

Current Status: Tuesday, November 12 🏐

Today's Priority: Architecture Decisions (6 hours)

You Have:
- ☑ Monday's analysis complete
- ☑ Both BAT files accessible
- ☑ Material databases ready
- ☑ 6 hours blocked today

Today's Agenda:
1. **Morning (2 hours):** Map Richmond & Holt hierarchies
2. **Afternoon (4 hours):** Make 3 critical decisions + design schema

Tomorrow: Draft coding standards
Friday: Team validation

Remember: These decisions determine project success. Take time to get them right.

💪 MOTIVATION

Why Week 1 Matters

This week determines everything:
- ☑ Right foundation → 11 weeks of smooth execution
- ❌ Wrong foundation → 4-6 weeks of rework

You have Monday's analysis complete:
- 746 items analyzed
- Richmond structure mapped
- 45 pages documentation
- Problems identified

Today you make the decisions that:
- Solve the triple-encoding problem
- Enable clean database design
- Support both Richmond and Holt
- Scale to Manor Homes
- Make Weeks 5-8 imports possible

The foundation you build this week determines whether this project succeeds or becomes technical debt.

Let's build it right! 🚀

Document Owner: Corey Boser
Last Updated: November 10, 2025
Next Review: After Week 1 completion

Status: Active - Week 1 Primary Guide

Current Action: Tuesday Architecture Decisions →
See 02_MASTER_PLAN.md for detailed Tuesday session breakdown