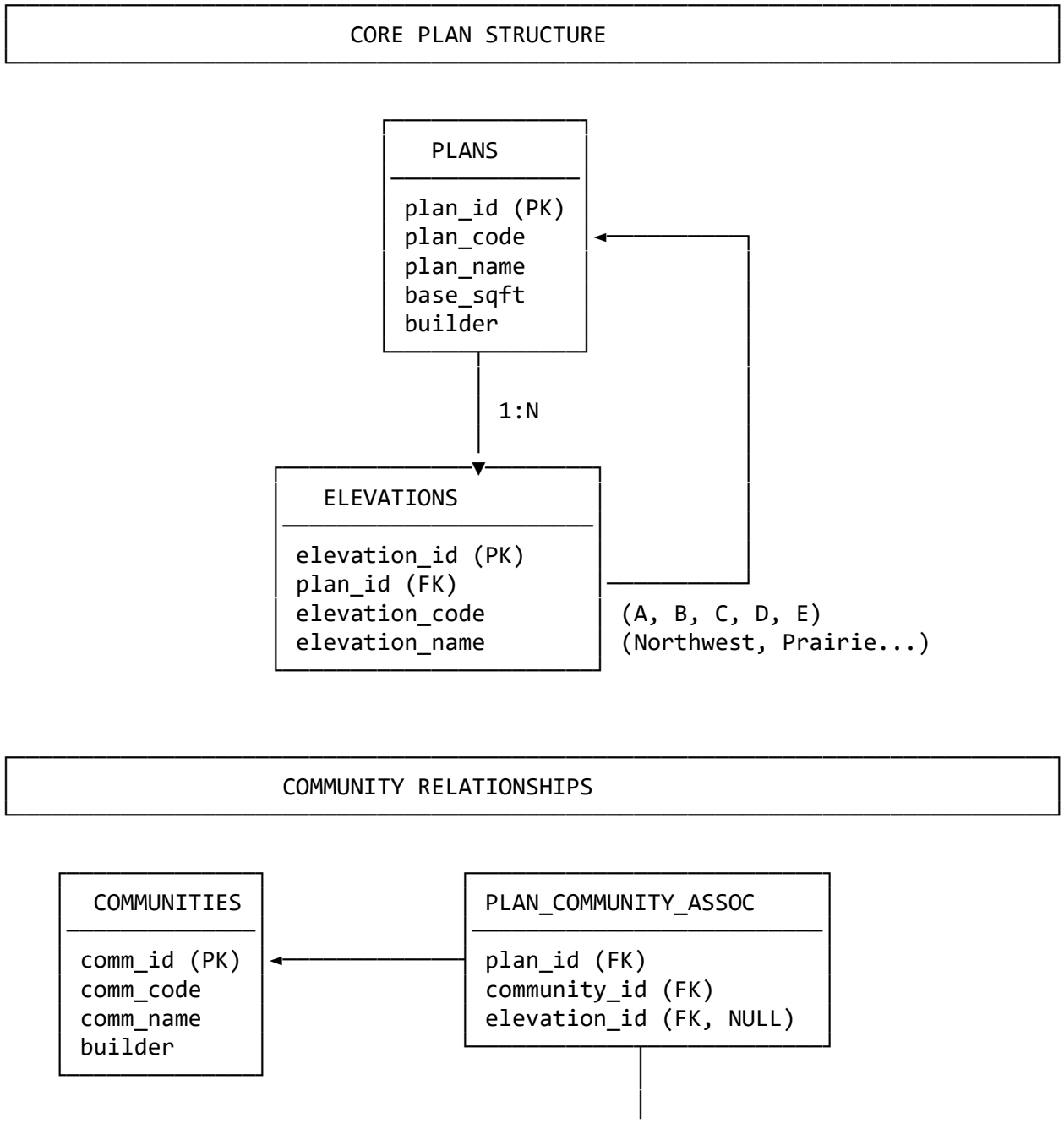
Database Schema Relationship Diagram

Project: Richmond & Holt BAT Consolidation
Schema Version: 1.0 DRAFT
Created: November 2025

Core Entity Relationships



Example:
GG = Golden Grove
CR = Coyote Ridge
HH = Harmony Heights

Links to
specific plans

PLANS

PACK STRUCTURE (Hybrid Model)

PACKS

pack_id (PK)
pack_code
pack_type_digit (1-9)
is_universal (TRUE/FALSE)

Assigned via

PACK_ASSIGNMENTS

plan_id (FK)
pack_id (FK)
is_override

Pack Type Examples:
1 = Foundation
2 = Framing
3 = Roofing
4 = Exterior Finishes
5 = Interior Finishes
9 = Options/Upgrades

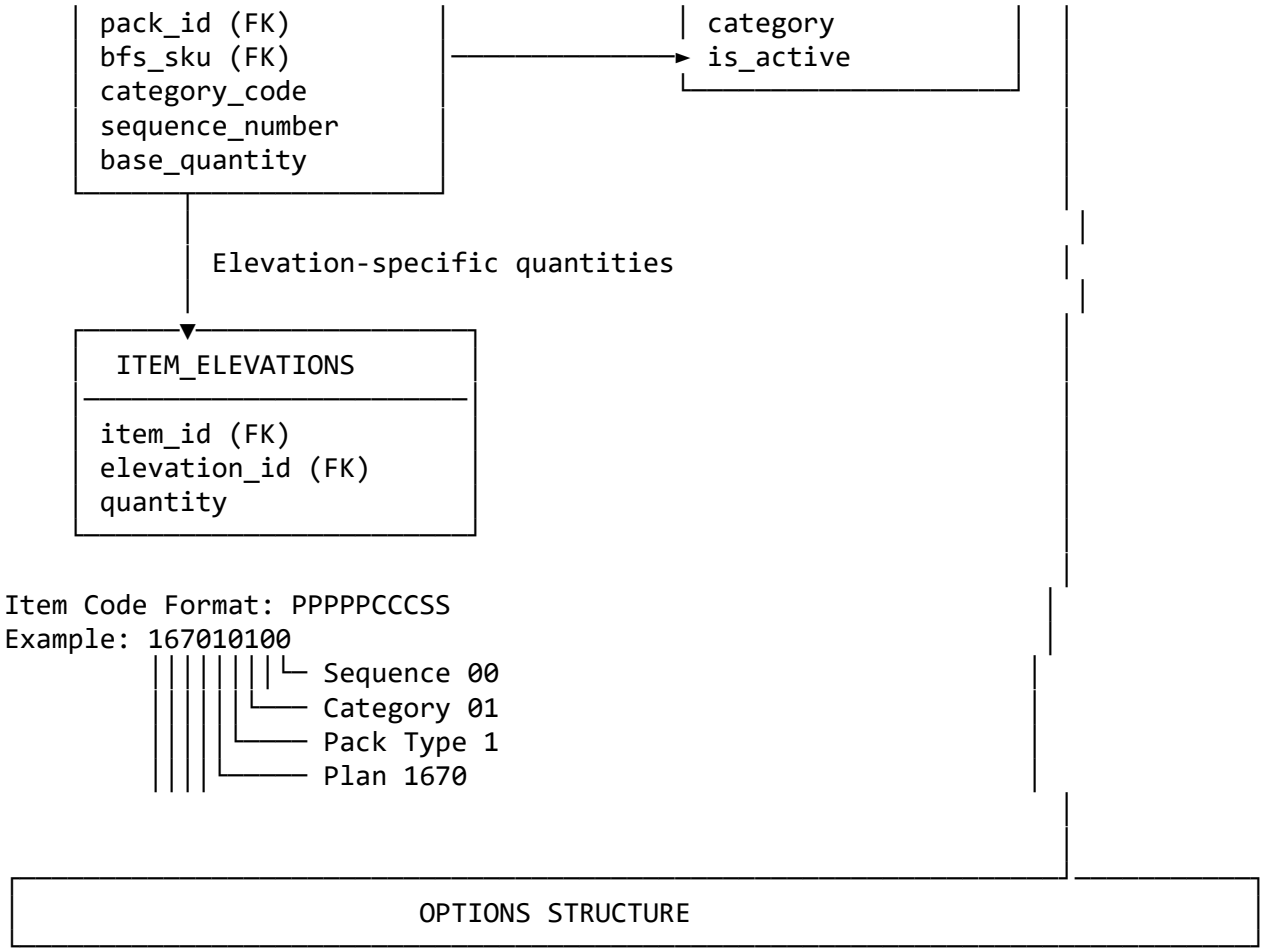
MATERIAL ITEM STRUCTURE

MATERIAL_ITEMS

item_id (PK)
item_code
plan_id (FK, NULL)

BFS_MATERIALS

bfs_sku (PK)
description
unit_of_measure

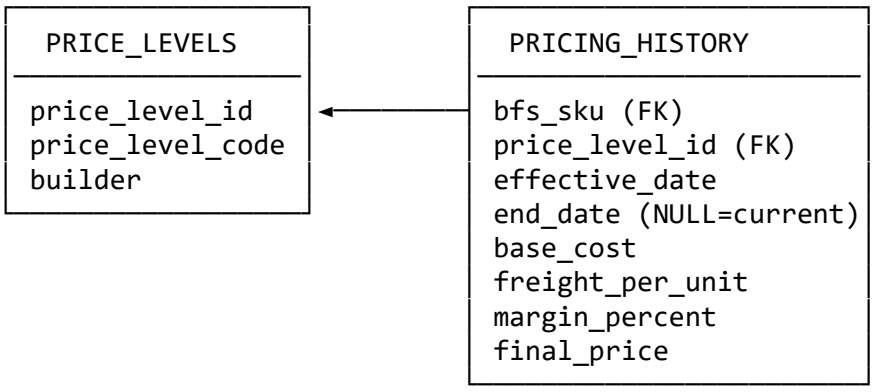


Option Code Format: OPT-[Category]-[Number]
Examples:

```
  OPT-GAR-301 = 3-Car Garage Left
  OPT-INT-101 = Gas Fireplace
  OPT-STR-401 = Bonus Room

Categories:
  GAR = Garage options
  INT = Interior options
  STR = Structural options
  EXT = Exterior options
```

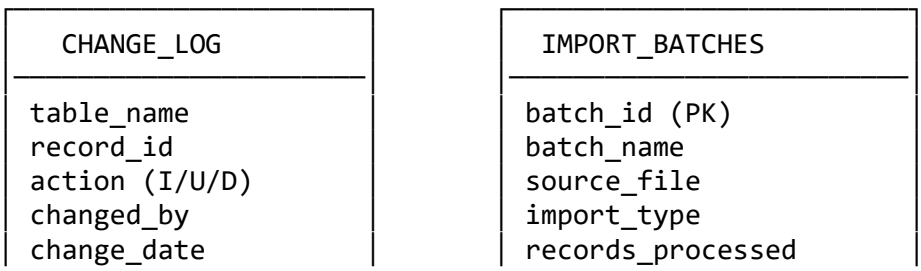
PRICING STRUCTURE




```
Price Level Examples:
  Richmond: L1, L2, L3, L4, L5
  Holt: PL01, PL02, ..., PL12

Time-Series Pricing:
  - Each price change creates new record
  - end_date = NULL for current price
  - Historical analysis enabled
```

AUDIT & TRACKING TABLES



```
| old_values (JSON)     |        | records_success           |
| new_values (JSON)     |        | records_failed            |
└───────────────────────┘        | status                    |
                                 └───────────────────────────┘
```

Purpose:
- Track all data modifications
- Enable rollback if needed
- Audit compliance
- Debug data issues

COMMON QUERY PATTERNS

1. GET ALL MATERIALS FOR A PLAN + ELEVATION:

```
SELECT mi.*, bm.description, bm.unit_of_measure
FROM material_items mi
JOIN bfs_materials bm ON mi.bfs_sku = bm.bfs_sku
LEFT JOIN item_elevations ie ON mi.item_id = ie.item_id
WHERE mi.plan_id = [plan_id]
  AND (ie.elevation_id = [elevation_id] OR ie.elevation_id IS NULL)
```

2. GET CURRENT PRICING FOR ALL ITEMS IN A PLAN:

```
SELECT mi.item_code, bm.description, ph.final_price
FROM material_items mi
JOIN bfs_materials bm ON mi.bfs_sku = bm.bfs_sku
JOIN pricing_history ph ON bm.bfs_sku = ph.bfs_sku
WHERE mi.plan_id = [plan_id]
  AND ph.price_level_id = [price_level_id]
  AND ph.end_date IS NULL
```

3. GET ALL PLANS IN A COMMUNITY:

```
SELECT p.*, c.community_name
FROM plans p
JOIN plan_community_association pca ON p.plan_id = pca.plan_id
JOIN communities c ON pca.community_id = c.community_id
WHERE c.community_code = [community_code]
  AND pca.is_active = 1
```

4. CALCULATE PLAN COST WITH OPTIONS:

```
SELECT
```

```
  mi.item_code,
  bm.description,
  CASE
    WHEN io.quantity_modifier IS NOT NULL
    THEN mi.base_quantity + io.quantity_modifier
    ELSE mi.base_quantity
  END as adjusted_quantity,
  ph.final_price,
  (adjusted_quantity * ph.final_price) as line_total
FROM material_items mi
LEFT JOIN item_options io ON mi.item_id = io.item_id
LEFT JOIN options o ON io.option_id = o.option_id
JOIN bfs_materials bm ON mi.bfs_sku = bm.bfs_sku
JOIN pricing_history ph ON bm.bfs_sku = ph.bfs_sku
WHERE mi.plan_id = [plan_id]
  AND (o.option_id IN ([selected_options]) OR o.option_id IS NULL)
```

KEY DESIGN PRINCIPLES

1. ELEVATION AS SEPARATE DIMENSION
   - ✓ Fixes triple-encoding problem
   - ✓ Matches Plan Index structure
   - ✓ Enables elevation-specific queries
   - ✓ Supports future pricing by elevation

2. HYBRID PACK MODEL
   - ✓ Universal packs reduce duplication
   - ✓ Plan-specific overrides for customization
   - ✓ Scales to 100+ plans
   - ✓ Simplifies bulk updates

3. RELATIONAL OPTIONS
   - ✓ Supports complex combinations
   - ✓ Matches contract line items
   - ✓ Enables option costing reports
   - ✓ Scales to hundreds of options

4. TIME-SERIES PRICING
   - ✓ Historical cost analysis
   - ✓ Price change tracking
   - ✓ Audit compliance
   - ✓ Trend reporting

5. EXPLICIT FOREIGN KEYS
   - ✓ Referential integrity
   - ✓ Cascade delete prevention
   - ✓ Clear relationships

✓ Query optimization

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                                                                              │
│                          MIGRATION STRATEGY                                  │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘

PHASE 1: Foundation (Week 1)
  ├─ Define schema
  ├─ Make three critical decisions
  └─ Create coding standards

PHASE 2: Core Tables (Week 2-3)
  ├─ Create PLANS, ELEVATIONS, COMMUNITIES
  ├─ Create PACKS, PACK_ASSIGNMENTS
  └─ Create BFS_MATERIALS, PRICE_LEVELS

PHASE 3: Material Import (Weeks 5-8)
  ├─ Import Richmond plans → PLANS table
  ├─ Import Holt plans → PLANS table
  ├─ Parse elevations → ELEVATIONS table
  ├─ Import materials → MATERIAL_ITEMS
  └─ Map elevation quantities → ITEM_ELEVATIONS

PHASE 4: Pricing Import (Weeks 5-8)
  ├─ Extract IWP pricing → PRICING_HISTORY
  ├─ Extract RL pricing → PRICING_HISTORY
  └─ Link price levels → PRICE_LEVELS

PHASE 5: Validation (Week 8)
  ├─ Verify all foreign keys valid
  ├─ Check data integrity
  ├─ Test query performance
  └─ Validate totals match source BATs


┌──────────────────────────────────────────────────────────────────────────────┐
│                                                                              │
│                              NEXT STEPS                                      │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘

□ Complete Phase 1 decision-making
□ Finalize schema based on team review
□ Create SQLite database from schema_design_FINAL.sql
□ Write Python import scripts
□ Begin data migration (Weeks 5-8)
□ Build web interface (Weeks 9-12)

```

Document Version: 1.0 DRAFT
Last Updated: November 2025
Status: Pending Phase 1 decisions
Next Review: After team feedback session