

PROMPT FOR CYRIL'S AI – 3P3 BOM PROTOTYPE IMPLEMENTATION

Safe Transition from Known System to 3P3 Paradigm

Context: Cyril Amegah is developing THE BRIDGE for KOOL TOOL. Luca Meggiolaro has developed the 3P3 paradigm over 30 years and now a SIMPLE PROTOTYPE is needed to demonstrate functionality with only 3 universal tables.



CURRENT SITUATION – THE PROBLEM

Critical Divergence:

- **Cyril is creating:** Separate tables (see The Bridge POC 02 – BOM, Team Materials, separate Logs)
- **Luca requires:** ONLY 3 universal tables containing EVERYTHING – (see 3p3 BOM – 3 universal tables)
- **Risk:** Parallel development incompatible with the 3P3 paradigm

The Legitimate Fear:

Cyril is right to be cautious. Transitioning from a system with 150+ specialized tables to only 3 universal tables seems impossible. But there is a trick that makes everything possible: **JSONData fields + ProcessType differentiation.**

“3P3 BOM – 3 UNIVERSAL TABLES IMPLEMENTATION GUIDE”

Essential Document Modifications:

1. PROCESS LIST INTEGRATION INTO CMP

The Process List (80+ processes from PROCESS LIST 20250819.pdf) is NOT a separate table but records in the CMP table:

```
-- EXAMPLE: Process List as TEMPLATE in CMP
INSERT INTO CMP (CmpID, CmpType, CmpSubType, CmpName, IsTemplate, JSONStructure)
VALUES
-- TASK Processes
('PRX_TSK_PHONE', 'PRX', 'TSK_PHONE_CALL', 'Phone Call Template', TRUE,
'{"standard_time": 15, "fields": ["number", "outcome", "notes"], "trigger": "client_request"}'),

('PRX_TSK_EMAIL', 'PRX', 'TSK_EMAIL', 'Email Template', TRUE,
'{"standard_time": 10, "fields": ["to", "subject", "body"], "trigger": "automatic"}'),
```

```
-- PROJECT Processes
('PRX_PRJ_CREATE', 'PRX', 'PRJ_CREATE', 'Project Creation Template', TRUE,
 '{"standard_time": 60, "fields": [{"client", "budget", "deadline"}], "trigger": "commercial"}'),

-- All 80+ processes from the Process List go here as PRX templates
;
```

2. 3P3 ENTITY ALGORITHM – THE HEART OF THE SYSTEM

3P3 EXISTENTIAL ALGORITHM:

"One Click = Three Simultaneous Births"

1. BIRTH IN CMP (or selection if template exists)
 - If new component/process → INSERT into CMP
 - If using existing template → SELECT from CMP
2. BIRTH IN ETY (always new instance)
 - Create specific instance from template
 - Assign unique DNA (e.g.: TSK_2025_0089)
 - Status = TO_DO
 - JSONData contains instance-specific data
3. BIRTH IN LOG (always new record)
 - Record the creation action
 - Track who, when, what
 - Link to ETY via TargetID

3. MINIMUM VIABLE PROTOTYPE

OBJECTIVE: Demonstrate that we can create a complete BOM with only 3 tables.

TEST CASE: 4-Panel Cover BOM

Click "New BOM" →

1. CMP: Select template PRX_BOM_STANDARD
2. ETY: Create instance BOM_2025_0089 with components in JSONData
3. LOG: Record CREATE_BOM by Marco at 14:30

Result: Fully functioning BOM without separate tables!



SPECIFIC REQUESTS FOR THE AI

1. Analyze the divergence

Compare:

- Cyril's "The Bridge POC 02" schema (separate tables)
- Luca's "3p3 BOM – 3 universal tables" schema (everything in 3 tables)

Identify EXACTLY where they diverge and how to reconcile.

2. Create Migration Script

// FileMaker Script to convert from separate tables to 3 universal ones

// 1. Migrate all Process templates → CMP with CmpType='PRX'

```
// 2. Migrate all Materials → CMP with CmpType='MAT'  
// 3. Migrate all active instances → ETY with JSONData  
// 4. Migrate all history → LOG
```

3. Proof of Concept BOM

Implement EXACTLY this case:

- User creates BOM for “4-Panel Cover”
 - System generates 3 records (one per table)
 - Show that everything works without separate tables
-

⚠ CRITICAL POINTS TO RESOLVE

1. JSONData vs Fixed Fields

```
-- INSTEAD OF TSK_PHONE table with 20 fields  
-- WE USE ETY with JSONData containing those 20 fields
```

```
ETY.JSONData = '{  
  "phone_number": "+39 333 1234567",  
  "contact_name": "Mario Rossi",  
  "call_duration": 15,  
  "call_outcome": "Interested",  
  // ... all phone call-specific fields  
'
```

2. Performance Concerns

- Are JSON queries slower? → Use JSON indexes (FileMaker 19+)
- Too many rows in one table? → Partition by year
- Complex backups? → Simpler with 3 tables!

3. User Interface

How to display different fields for different processes?

```
// Layout adapts to ProcessSubType  
If [ETY::ProcessSubType = "PHONE_CALL"]  
  Show Fields: Phone, Duration, Outcome  
Else If [ETY::ProcessSubType = "EMAIL"]  
  Show Fields: To, Subject, Body  
End If
```



VISUAL COMPARISON

Cyril's System (Current):

BOM Table	Team Materials Table	Logs Table
└─ BOM_ID	└─ Material_ID	└─ Log_ID
└─ Product_Code	└─ Material_Name	└─ Action
└─ Quantity	└─ Quantity	└─ User
└─ ...	└─ ...	└─ ...

+ 147 other tables for each process...

3P3 System (Target):

CMP (Templates)	ETY (Instances)	LOG (Actions)
└─ All Materials	└─ All BOMs	└─ All History
└─ All Processes	└─ All Tasks	└─ All Changes
└─ All Products	└─ All Projects	└─ All Events
└─ EVERYTHING!	└─ EVERYTHING!	└─ EVERYTHING!

ONLY 3 TOTAL TABLES!



REQUIRED DELIVERABLES

1. **Revised Document:** "3P3 BOM – IMPLEMENTATION GUIDE"
2. **BOM Creation Script:** Generates 3 records simultaneously
3. **Complete Test Case:** 4-Panel Cover from start to finish
4. **Migration Path:** How to convert existing system



SUPPORT FOR CYRIL

Message for Cyril:

"You're not losing any of your work. We're just reorganizing it more elegantly. All your fields, all your logic, everything remains – only in 3 containers instead of 150. It's like going from 150 drawers to 3 well-organized wardrobes with internal dividers (JSONData)."

Immediate Benefits:

- Database backup: 3 tables instead of 150
- New processes: add records, not tables
- Unified reports: query on one table, not complex JOINS
- Maintenance: 95% less code



NEXT STEPS

1. **STOP** creating new separate tables
2. **IMPLEMENT** 3-table prototype for ONE process (e.g.: TSK_PHONE)
3. **VALIDATE** that it works identically to current system
4. **SCALE** gradually to other processes
5. **CELEBRATE** the achieved simplification!

REMEMBER: The goal is not to revolutionize everything immediately, but to demonstrate that the 3P3 paradigm works with a simple and clear prototype.