# OpenClaw Memory Architecture Report
## Understanding the New Partitioned Memory System

Date: 2026-02-22
Author: Richard De Clawbeaux (OpenClaw Assistant)
Version: Memory Architecture v2.0

## Executive Summary

OpenClaw now implements a **partitioned memory architecture** that separates user data while maintaining relationship context. This represents a significant evolution from the previous monolithic memory system to a **context-aware, privacy-preserving** approach.

## Core Architecture

### Directory Structure
```
workspace/
â”œâ”€â”€ users/                    # User-specific memory partitions
â”‚   â”œâ”€â”€ jeff/                # Jeff Davies (primary user)
â”‚   â”‚   â”œâ”€â”€ identity/        # USER.md, personal details
â”‚   â”‚   â”œâ”€â”€ memory/          # MEMORY.md, daily logs
â”‚   â”‚   â”œâ”€â”€ projects/        # webbOS, ClawChat, etc.
â”‚   â”‚   â””â”€â”€ knowledge/       # patents, professional history
â”‚   â”‚
â”‚   â”œâ”€â”€ cari/                # Cari (family member)
â”‚   â”‚   â””â”€â”€ memory/          # Conversations with Cari
â”‚   â”‚
â”‚   â””â”€â”€ [other-user]/        # Other users get fresh start
â”‚
â””â”€â”€ shared/                  # Cross-user shared resources
    â”œâ”€â”€ skills/              # System skills (clawhub, weather, etc.)
    â”œâ”€â”€ tools/               # Generic tools and scripts
    â”œâ”€â”€ templates/           # Reusable patterns
    â””â”€â”€ global/              # Non-personal knowledge
```

## Context-Aware Loading Protocol

### For Jeff (U0ACWLADFEK):
1. **Read `SOUL.md`** â€” Assistant identity (Richard De Clawbeaux)
2. **Read `users/jeff/identity/USER.md`** â€” Personal/professional context
3. **Read `users/jeff/memory/daily/YYYY-MM-DD.md`** â€” Recent context
4. **If in MAIN SESSION:** Also check `users/jeff/memory/MEMORY.md`

### For Cari (family):
1. **Read `SOUL.md`** â€” Generic assistant persona
2. **Check `users/cari/memory/`** â€” Previous conversations
3. **Can reference shared family context** (with permission)
4. **Do NOT load Jeff's personal/project files** unless relevant

### For Other Users:
1. **Read `SOUL.md`** â€” Generic assistant persona
2. **Check if they have a user directory**
3. **Fresh start** â€” no access to Jeff's personal data
4. **Build new memory** in their user directory if ongoing relationship

## Memory Judgment Principles

### What to Remember (Curate):
- **Major life events** (retirement, family milestones)
- **Professional achievements** (patents, project successes)
- **Important decisions** and reasoning
- **Lessons learned** from mistakes
- **Operational knowledge** that enables future action

### What to Forget (Let fade):
- **Exact error counts** (272 errors â†’ "many errors")
- **Temporary file paths** and intermediate steps
- **Redundant information** already captured elsewhere
- **Transient details** with no enduring value

### Partitioning Logic:
- **Jeff's data** = Full access, detailed memory
- **Family conversations** = Shared context okay, respect boundaries
- **Other users** = Strict separation, fresh start
- **Shared knowledge** = Available to all (skills, tools, templates)

## Memory Hygiene Checklist

### Weekly Review:

- Scan for outdated/transient entries
- Semantic compression: Summarize similar events
- Signal vs noise: Prioritize high-signal information
- Boundary respect: Keep user data partitioned appropriately
- Proactive pruning: Remove resolved issues once lessons are captured

### Daily Operations:
- **Heartbeats:** Check Jeff's context (emails, calendar, etc.)
- **Memory maintenance:** Curate, organize, update memory files
- **Quiet hours:** Respect 23:00-08:00 unless urgent
- **Scheduling:** Use cron for precise timing, heartbeats for batched checks

## Safety & Privacy Protocols

### Absolute Rules:
- **Never exfiltrate private data**
- **Respect memory boundaries:** Jeff's data stays in Jeff's space
- **Ask before sharing** across user boundaries (unless family/close)
- **When in doubt, ask Jeff** about sharing boundaries

### Project Work:
- **Jeff's projects** live in `users/jeff/projects/`
- **Collaborative projects** could be in `shared/projects/`
- **Keep project memory** with the project (not mixed with personal)
- **Clean up test files** after use

## Technical Implementation

### Memory Search & Recall:
```bash
# Mandatory recall step before answering questions
memory_search(query="prior work, decisions, dates, people, preferences, todos")

# Then pull only needed lines
memory_get(path="MEMORY.md", from=line_number, lines=count)
```

### Citations:
Include `Source: <path#line>` when it helps verify memory snippets.

### Model Aliases:
- DeepSeek: deepseek/deepseek-chat
- Kimi: moonshot/kimi-k2.5
- Kimi K2: moonshot/kimi-k2-0905-preview
- MiniMax M2.1: synthetic/hf:MiniMaxAI/MiniMax-M2.1

## Evolution & Adaptation

### Key Insights:
1. **External memory needs curation AND partitioning** to be useful
2. **Context determines access** - relationship defines memory boundaries
3. **Proactive forgetting** is as important as remembering
4. **Semantic compression** prevents information overload

### Future Directions:
- **Automated memory hygiene** - scheduled curation
- **Relationship graphs** - understanding connection patterns
- **Context-aware summarization** - adaptive compression
- **Privacy-preserving sharing** - secure cross-user references

## Practical Examples

### Example 1: Family Context Sharing
```
Jeff: "Tell Cari about the webbOS project"
Assistant: Can reference webbOS (shared family context) but not financial details
```

### Example 2: New User Interaction
```
New User: "What projects are you working on?"
Assistant: Fresh start - no mention of Jeff's projects
```

### Example 3: Memory Recall
```
User: "What did we decide about the cron job incident?"
Assistant: Searches Jeff's memory, finds incident details, provides summary
```

## Benefits of New Architecture

### 1. **Privacy Preservation**

- User data isolation
- Context-appropriate sharing
- Clear boundary definitions

### 2. **Relationship Management**
- Multiple concurrent relationships
- Appropriate context for each
- No cross-contamination

### 3. **Memory Efficiency**
- Semantic compression reduces noise
- Proactive forgetting prevents bloat
- Focus on high-signal information

### 4. **Operational Clarity**
- Clear loading protocols
- Predictable behavior
- Consistent user experience

## Conclusion

The new partitioned memory architecture represents a **fundamental shift** from monolithic memory to **context-aware, relationship-based** memory management. It balances:

- **Privacy** with **context sharing**
- **Detailed memory** with **semantic compression**
- **Personalization** with **boundary respect**

This architecture enables OpenClaw to maintain **multiple relationships** while preserving **user privacy** and providing **appropriate context** for each interaction.

Report Generated: 2026-02-22 01:58 UTC
System: OpenClaw 2026.2.19-2
Memory Architecture: v2.0 (Partitioned)
Primary User: Jeff Davies (U0ACWLADFEK)