D^4 Domain-Driven Database Design

Curriculum Development Discussion Outline

Presenter: Peter Heller, Database Systems Instructor, Queens College CUNY
Purpose: Exploratory discussion to assess alignment and interest in curriculum development funding

---

1. Executive Overview (2-3 minutes)

The Problem

- Industry gap: 70%+ of database projects fail or significantly exceed budgets
- Modern developers lack systematic training in database design fundamentals and in-depth analysis skills
- Industry increasingly requires developer/analyst hybrid roles—not separated developers and business analysts
- Disconnect between application domain models (DDD) and physical database implementation
- Students graduate without practical experience in full-lifecycle database design

The Solution: D^4 (Domain-Driven Database Design)

- Physical Data Model-first methodology for operational databases
- Integrates Business Glossary Definitions (BGD) into Fully Qualified Domain Names (FQDN)
- Semantic naming conventions enable each table to produce unique golden records, minimizing MDM overhead
- Governance-as-code approach using standardized enterprise object naming
- Practical, teachable framework refined since 2003 (NYC DCAS EC3 energy billing system)

The Opportunity

Scaffolded multi-course curriculum where students learn methodology while delivering solutions architect-level project work

---

2. Curriculum Scaffold Approach

Core Philosophy: Solutions Architect Training Through Practical Projects

Students learn D^4 methodology by working through progressively complex database projects—the same deliverables a solutions architect would produce in industry. Each course builds on the previous, creating a coherent multi-semester pathway.

Course Sequence Structure

Course 1: Database Systems Foundations (CSCI 331 equivalent)

Prerequisites: Data Structures, Programming proficiency

Learning Objectives:

- Master Physical Data Model-first design approach
- Implement FQDN taxonomy with BGD integration
- Develop semantic naming conventions for enterprise standardization
- Design schemas where each table produces unique golden records
- Understand governance-as-code principles

Practical Project Deliverables:

- Physical database design for operational system (e.g., order management, student registration)
- ANSI SQL DDL scripts with semantic naming enforcement
- Documentation showing BGD → FQDN mapping
- Normalization analysis demonstrating SRP at table level

Technology Stack:

- SQL Server 2025 (ANSI SQL compliant)
- Docker for consistent dev environments
- GitHub for version control and collaboration
- Azure Data Studio or SQL Server Management Studio

Student Role: Database Designer/Developer

Course 2: Advanced Database Design & Analysis (New or CSCI 381 Enhanced)

Prerequisites: Course 1 (Database Systems Foundations)

Learning Objectives:

- Apply D^4 methodology to complex business domains
- Integrate analysis skills with database design (unified developer/analyst role)
- Work with LLM APIs for schema generation and validation
- Optimize for both transactional and analytical workloads
- Create data models that support enterprise MDM strategies

Practical Project Deliverables:

- Enterprise-scale database design (e.g., financial system, healthcare application)
- Requirements analysis documentation showing business rules → FQDN translation
- AI-assisted schema validation using LLM API calls (via Ollama for cost containment)
- Data lineage documentation demonstrating golden record production
- Migration strategy from legacy schema to D^4 compliant design

Technology Stack:

- SQL Server 2025 with advanced features
- Ollama for local LLM deployment (student cost containment)

- Docker Compose for multi-container environments
- GitHub with CI/CD pipelines
- Erwin Data Modeler or similar for visual documentation

Student Role: Database Solutions Architect/Analyst

Course 3: Production Database Engineering (New or CSCI 531 Enhanced)

Prerequisites: Course 2 (Advanced Database Design & Analysis)

Learning Objectives:

- Deploy D^4 designs to production-grade environments
- Implement monitoring and observability for database systems
- Apply governance-as-code throughout full lifecycle
- Integrate databases with microservices architectures
- Handle real-world data migration and refactoring scenarios

Practical Project Deliverables:

- Dockerized database deployment with orchestration
- Automated schema validation and testing pipeline
- Performance optimization demonstrating FQDN benefits
- API integration layer following domain-driven principles
- Production readiness assessment and documentation

Technology Stack:

- Docker Swarm or Kubernetes for orchestration
- GitHub Actions for CI/CD automation
- Monitoring tools (Prometheus, Grafana)
- API frameworks integrated with database layer
- Infrastructure-as-code (Terraform or similar)

Student Role: Production Database Engineer/DevOps

Scaffolding Principles Across Courses

1. **Progressive Complexity:** Course 1 = single domain, Course 2 = enterprise integration, Course 3 = production deployment
2. **Consistent Methodology:** D^4 principles reinforced in each course with increasing sophistication
3. **Real Deliverables:** Students build portfolio-quality work that demonstrates solutions architect capabilities
4. **Cost-Conscious Technology:** Ollama for AI features eliminates per-student API costs
5. **Industry-Ready Tools:** Docker, GitHub, ANSI SQL ensure transferable skills

---

Key Differentiator: FQDN Integration

What Makes This Unique:

- Physical Model-first approach naturally integrates Business Glossary Definitions into table/column names
- FQDN taxonomy creates self-documenting schemas
- Each table designed to produce golden records, reducing enterprise MDM complexity
- Governance enforced at design time through semantic naming, not post-deployment reviews

Example:

```
Bad: customer_orders, order_items
Good: Customer.Order, Customer.Order.Item

The FQDN immediately shows:
- Domain context (Customer)
- Entity hierarchy (Order contains Items)
- Business glossary mapping (Customer, Order, Item are enterprise terms)
- Golden record capability (Customer.Order is authoritative for order data)
```

---

3. Core Technical Stack (Standardized Across Courses)

Rationale: Industry-Standard, Cost-Effective, Transferable Skills

Database Platform:

- SQL Server 2025 (ANSI SQL compliant, widely used in enterprise)
- Justification: Students learn transferable SQL skills, strong tooling ecosystem

Development Environment:

- Docker for containerization (consistent environments across student machines)
- GitHub for version control, collaboration, and portfolio building
- Azure Data Studio or SSMS (free, professional-grade tools)

AI Integration:

- Ollama for local LLM deployment (eliminates per-student API costs)
- Schema validation, documentation generation, requirements analysis support
- Students learn AI-assisted development without financial barriers

Infrastructure-as-Code:

- Docker Compose for multi-container orchestration
- GitHub Actions for CI/CD pipelines
- Keeps focus on database design while teaching modern DevOps practices

Justification for Choices:

- All tools are free or open-source (no student financial burden)

- SQL Server 2025 available through Azure for Students program
- Docker/GitHub are industry expectations for any developer role
- Ollama enables AI features without recurring costs

---

4. Impact Metrics (Focused & Measurable)

Student Learning Outcomes

1. **Design Competency:** Can students create Physical Data Models that integrate BGD into FQDN?
    - Assessment: Capstone project evaluated by industry advisory board
2. **Developer/Analyst Integration:** Do students demonstrate unified analysis + implementation skills?
    - Assessment: Project documentation showing requirements → schema translation
3. **Governance Understanding:** Can students explain and enforce semantic naming conventions?
    - Assessment: Code reviews, peer evaluations of schema quality

Employment Outcomes

1. **Job Placement:** Percentage of students securing database-related roles within 6 months
2. **Industry Feedback:** Employer surveys on graduate preparedness
3. **Portfolio Quality:** Student GitHub repositories showcasing D^4 projects

Curriculum Effectiveness

1. **Course Completion:** Retention rates across three-course sequence
2. **CUNY Adoption:** Number of other CUNY campuses piloting D^4 curriculum
3. **Open Resource Usage:** Downloads/forks of curriculum materials on GitHub

Long-Term Impact

1. **Technical Debt Reduction:** Track alumni reporting schema quality improvements in their organizations
2. **Methodology Adoption:** Companies implementing D^4 principles after hiring graduates
3. **Community Growth:** Contributions to D^4 methodology from students and alumni

Key Principle: Metrics focus on student preparedness for developer/analyst hybrid roles, not research citations or tool downloads

5. Peter's Unique Qualifications

Teaching Excellence

- **10 years teaching** CSCI 331 (Database Systems) and CSCI 381 (Erwin Data Modeling, Power BI) at Queens College
- Developed and refined D^4 methodology starting in **2003** with NYC DCAS EC3 energy billing system
- FQDN taxonomy originated from that 2003 project—20+ years of real-world validation
- Created comprehensive educational materials integrating modern tools (Python, Docker, Pydantic)
- Student-centered pedagogy emphasizing solutions architect-level deliverables

Methodology Origins & Evolution

- **2003:** Designed NYC DCAS EC3 energy billing system using Fully Qualified Domain Name taxonomy
- **Foundation:** FQDN approach became the basis for D⁴ Domain-Driven Database Design
- **Refinement:** 20+ years of iteration across enterprise projects and classroom teaching
- **Validation:** Methodology proven in both production systems and educational settings

Industry Experience

- Real-world database architecture for municipal government systems
- Enterprise-scale projects requiring MDM integration and golden record production
- Solutions architect perspective: requirements analysis through production deployment
- Bridge between academic rigor and industry pragmatism

Technical Infrastructure

- Production-grade development environment (multi-node Docker Swarm cluster)
- Expertise across SQL Server, PostgreSQL, DuckDB
- Modern development stack supporting AI-integrated workflows
- Can provide students with cloud-like environment using local infrastructure

---

6. Strategic Partnerships & Industry Connections

New York CEO Jobs Council

- **Critical Connection:** Peter was in the third cohort
- Direct access to NYC tech employers seeking database-skilled graduates
- Industry advisory board potential from council network
- Employment pipeline for students completing D⁴ curriculum
- Real-world validation through employer feedback

Potential Collaborators

- **CUNY System:** Multi-campus pilot across undergraduate CS programs
- **Microsoft:** SQL Server 2025 education initiatives, Azure for Students integration
- **NYC Tech Employers:** Internship placements, capstone project sponsors
- **Database Vendors:** Erwin, Snowflake, Databricks for tooling partnerships
- **Professional Organizations:** ACM, IEEE database education committees

Industry Advisory Board (To Be Formed)

- NYC CEO Jobs Council members
- Database architects from financial services, healthcare, government
- Hiring managers seeking developer/analyst hybrid talent
- Alumni working in database-related roles

Value Proposition to Partners:

- Access to pipeline of students with rare combination: design + analysis + governance skills

- Opportunity to shape curriculum based on real industry needs
- Early engagement with graduates trained in modern database practices

---

7. Discussion Questions for Funder

Understanding Their Priorities

1. Does this scaffolded curriculum approach align with your educational funding priorities?
2. What evidence/metrics would be most compelling for your review process?
3. Are there specific gaps in database education you've identified that this addresses?
4. Would multi-institutional collaboration (across CUNY) strengthen the proposal?
5. Is the developer/analyst hybrid role something you see as increasingly important in industry?

Addressing Potential Concerns

- **"How is this different from existing database courses?"**
    - Physical Model-first with FQDN/BGD integration
    - Developer + analyst skills taught together, not separately
    - Solutions architect-level project deliverables, not theoretical exercises
- **"Why now? What's changed?"**
    - Industry needs developer/analyst hybrids, not separated roles
    - AI integration (via Ollama) makes advanced tooling accessible to undergraduates
    - Modern DevOps practices (Docker, GitHub) now expected for all developers
- **"What about NoSQL/NewSQL?"**
    - D^4 principles (FQDN, semantic naming, golden records) apply across database paradigms
    - Starting with relational provides strongest foundation
- **"How do you ensure adoption beyond Queens College?"**
    - Open curriculum materials via GitHub
    - Instructor guides and workshop for other CUNY faculty
    - NYC CEO Jobs Council connections provide employer demand signal

Understanding Methodology Status

- D^4 methodology refined over 20+ years (2003 origins)
- This is curriculum development, not research into untested methodology
- Focus is on teaching proven practices, not validating new theory
- Open to discussing intellectual property framework if relevant to funder

---

8. Supporting Materials Available

Ready to Share:

- D^4 methodology overview document

- Sample curriculum materials from CSCI 331 and CSCI 381
- Student project examples demonstrating D⁴ application
- FQDN taxonomy examples from 2003 NYC DCAS EC3 project
- NYC CEO Jobs Council third cohort credentials

Can Be Developed Upon Interest:

- Detailed course syllabi for three-course sequence
- Assessment rubrics for solutions architect-level deliverables
- Docker/GitHub starter kits for student onboarding
- Ollama integration guides for cost-effective AI features
- Industry advisory board composition and commitment letters

9. Contact Information

Peter Heller
Database Systems Instructor, Queens College CUNY
Email: peter.heller@qc.cuny.edu
GitHub: github.com/QCadjunct (educational), github.com/ph3ll3r (personal)

Summary: Why This Curriculum Matters

The Gap:

- Employers need developer/analysts who can design, implement, and maintain databases
- Current education separates these skills or treats database design as theoretical exercise
- Students graduate without portfolio-quality database work

The Solution:

- Three-course scaffold teaching D⁴ methodology through practical projects
- Students produce solutions architect-level deliverables
- FQDN/BGD integration creates self-documenting, governance-compliant schemas
- Cost-effective AI integration via Ollama
- Industry connections via NYC CEO Jobs Council ensure employment outcomes

The Track Record:

- 20+ years of methodology refinement (2003 NYC DCAS EC3 origins)
- 10 years classroom validation at Queens College
- Proven in both production systems and educational settings
- Ready to scale across CUNY system with funder support