



**Agile Database Techniques: Effective Strategies for the Agile Software Developer**  
**HDT822**  
**Four Days**

---

**Prerequisites**

Students should have experience designing databases and data warehouses. Knowledge of Agile design techniques is helpful.

**Who Should Attend**

This course is targeted at database designers, data modelers, database analysts, and anyone who needs to design databases.

**Course Description**

This course provides students with the skills necessary to design databases using Agile design techniques. It is based on the Scott Ambler book [Agile Database Techniques: Effective Strategies for the Agile Software Developer](#) published on October 17, 2003 by John Wiley, ISBN: 0471202837.



## Agile Database Techniques: Effective Strategies for the Agile Software Developer HDT822

---

### Course Topics

- The Agile Data Method
- From Use Cases to Databases – Real-World UML
- Data Modeling 101
- Data Normalization
- Class Normalization
- Relational Database Technology, Like It, Or Not
- The Object-Relational Impedance Mismatch
- Legacy Databases – Everything You Need to Know But Are Afraid to Deal With
- Vive L'Evolution
- Agile Model-Driven Development (AMDD)
- Test-Driven Development (TDD)
- Database Refactoring
- Database Encapsulation Strategies
- Mapping Objects to Relational Databases
- Performance Tuning
- Tools for Evolutionary Database Development
- Implementing Concurrency Control
- Finding Objects in Relational Databases
- Implementing Referential Integrity and Shared Business
- Implementing Concurrency Control
- Finding Objects in Relational Databases
- Implementing Referential Integrity and Shared Business Logic
- Implementing Security Access Control
- Implementing Reports
- Realistic XML
- How You Can Become Agile
- Bringing Agility Into Your Organization



## **Agile Database Techniques: Effective Strategies for the Agile Software Developer HDT822**

---

- I. The Agile Data Method**
  - A. Why Working Together Is Currently Hard
  - B. Detecting That You Have a Problem
  - C. The Agile Movement
  - D. The Philosophies of Agile Data
  - E. Agile Data In a Nutshell
  - F. Agile Software Developers
  - G. Does Agile Data Solve Our Problems?
  
- II. From Use Cases to Databases – Real-World UML**
  - A. An Overview of Object-Oriented Concepts
  - B. An Introduction to the Unified Modeling Language (UML)
    - 1. Core UML Diagrams
      - i. Use Case Diagrams
      - ii. Sequence Diagrams
      - iii. Class Diagrams
    - 2. Supplementary UML Diagrams
      - i. Activity Diagrams
      - ii. Collaboration Diagrams
      - iii. Component Diagrams
      - iv. Deployment Diagrams
      - v. State Chart Diagrams
  - C. A UML Profile For Data Modeling
    - 1. Indicating the Type of Model or Storage Mechanism
    - 2. Modeling Tables, Entities, and Views
    - 3. Modeling Relationships
    - 4. Modeling Data Attributes and Columns
    - 5. Modeling Keys
    - 6. Modeling Constraints and Triggers
    - 7. Modeling Stored Procedures
    - 8. Modeling Sections Within a Database
    - 9. Modeling Everything Else



## Agile Database Techniques: Effective Strategies for the Agile Software Developer HDT822

---

### III. Data Modeling 101

- A. The Role of the Agile DBA
- B. What Is Data Modeling?
  1. How Are Data Models Used In Practice?
  2. Notation 101: How to Read Data Models
- C. How to Model Data
  1. Identify Data Entities
  2. Identify Attributes
  3. Apply Data-Naming Conventions
  4. Identifying Relationships
  5. Apply Data Model Patterns
  6. Assign Keys
- D. How to Become Better at Modeling Data

### IV. Data Normalization

- A. Why Data Normalization?
- B. The Role of the Agile DBA
- C. The Rules of Data Normalization
  1. First Normal Form (1NF)
  2. Second Normal Form (2NF)
  3. Third Normal Form (3NF)
  4. Beyond 3NF

### V. Class Normalization

- A. How Does Class Normalization Relate to Other Object Design Practices?
- B. The Role of the Agile DBA
- D. The Rules of Object Normalization
  1. First Object Normal Form (1ONF)
  2. Second Object Normal Form (2ONF)
  3. Third Object Normal Form (3ONF)
  4. Beyond 3ONF

### VI. Relational Database Technology, Like It, Or Not

- A. Relational Database Technology
- B. Coupling: Your Greatest Enemy
- C. Additional Challenges With Relational Databases
- D. Encapsulation: Your Greatest Ally
- E. Beyond Relational Databases: You Actually Have a Choice



## **Agile Database Techniques: Effective Strategies for the Agile Software Developer HDT822**

---

- VII. The Object-Relational Impedance Mismatch**
  - A. The Role of the Agile DBA
  - B. The Technological Impedance Mismatch
  - C. The Cultural Impedance Mismatch
  
- VIII. Legacy Databases – Everything You Need to Know But Are Afraid to Deal With**
  - A. The Role of the Agile DBA
  - B. Sources of Legacy Data
  - C. Understanding Common Problems with Legacy Data
    - 1. Data Quality Challenges
    - 2. Database Design Problems
    - 3. Data Architecture Problems
    - 4. Process Mistakes
  - D. Strategies for Working with Legacy Data
    - 1. Try to Avoid Working with Legacy Data
    - 2. Develop a Data Error-Handling Strategy
    - 3. Work Iteratively and Incrementally
    - 4. Prefer Read-Only Legacy Data Access
    - 5. Encapsulate Legacy Data Access
    - 6. Introduce Data Adapters for Simple Legacy Access
    - 7. Introduce a Staging Database for Complex Legacy Access
    - 8. Adopt Existing Tools
  - E. Data Integration Technologies
  
- IX. Vive L'Evolution**
  - A. The Need for Methodological Flexibility
  - B. Beware of Data-Oriented BDUF
  - C. Evolutionary Development on a Project
  - D. The “Natural Order” of Things and Evolutionary Development
  
- X. Agile Model-Driven Development (AMDD)**
  - A. The Role of the Agile DBA
  - B. What Is Agile Modeling?
  - C. When Is a Model Agile?
  - D. What Is Agile Model-Driven Development (AMDD)?
  - E. Agile Documentation



## Agile Database Techniques: Effective Strategies for the Agile Software Developer HDT822

---

### **XI. Test-Driven Development (TDD)**

- A. How Does TDD Work?
- B. The Steps of TDD
- C. TDD and Traditional Testing
- D. TDD and Documentation
- E. Test-Driven Database Development
- F. TDD and Agile Model-Driven Development (AMDD)

### **XII. Database Refactoring**

- A. Refactoring
- B. Database Refactoring
  - 1. Preserving Semantics
  - 2. What Database Refactorings Aren't
  - 3. Categories of Database Refactorings
- C. Why Database Refactoring Is Hard
- D. How to Refactor Your Database
  - 1. Step 1: Start in Your Development Sandbox
  - 2. Verify That a Database Refactoring Is Required
  - 3. Choose the Most Appropriate Database Refactoring
  - 4. Determine Data Cleansing Needs
  - 5. Write Your Unit Tests
  - 6. Deprecate the Original Schema
  - 7. Implement the Change
  - 8. Update Your Database Management Script(s)
  - 9. Run Your Regression Tests
  - 10. Document the Refactoring
  - 11. Version Control Your Work
  - 12. Step 2: Implement the Code in Your Integration Sandbox(es)
  - 13. Step 3: Install the Code in Production
- E. Common Database Refactoring Smells
- F. Adopting Database Refactoring within Your Organization
- G. Database Refactoring Best Practices
- H. Database Refactoring in the Real World



### **XIII. Database Encapsulation Strategies**

- A. Database Encapsulation Layers
- B. The Role of the Agile DBA
- C. Encapsulation-Layer Architectures
- D. Encapsulation-Layer Implementation Strategies
  - 1. Brute Force (the Encapsulation Strategy That Isn't One)
  - 2. Data Access Objects
  - 3. Persistence Frameworks

- 4. Services
- 5. When to Use Each Strategy
- 6. Transitioning Between Strategies
- E. Marshaling and Data Validation
- F. Error Handling

#### **XIV. Mapping Objects to Relational Databases**

- A. The Role of the Agile DBA
- B. Basic Mapping Concepts
  - 1. Shadow Information
  - 2. Mapping Meta Data
- C. Mapping Inheritance Structures
  - 1. Map Entire Class Hierarchy to a Table
  - 2. Map Each Concrete Class to Its Own Table
  - 3. Map Each Class to Its Own Table
  - 4. Map Classes to a Generic Structure
  - 5. Comparing the Mapping Strategies
  - 6. Mapping Multiple Inheritance
- D. Mapping Object Relationships
  - 1. Types of Relationships
  - 2. How Relationships Are Implemented Between Objects
  - 3. How Relationships Are Implemented in Relational Databases
  - 4. Relationship Mapping
    - i. Mapping One-to-One Relationships
    - ii. Mapping One-to-Many Relationships
    - iii. Mapping Many-to-Many Relationships
- E. Mapping Ordered Collections
- F. Mapping Recursive Relationships
- G. Mapping Class Scope Properties
- H. Why Data Schemas Should Not Drive Object Schemas
- I. Implementation Impact on Your Objects
- J. Implementation for the Model Driven Architecture (MDA)
- K. Patternizing the Mappings



#### **XV. Performance Tuning**

- A. An Overview of Performance Tuning
- B. The Role of the Agile DBA
- C. Step One: Identify a Performance Problem
- D. Step Two: Profile the Problem
- E. Step Three: Tune the Problem Away
  - 1. System Tuning
  - 2. Database Access Tuning

## Agile Database Techniques: Effective Strategies for the Agile Software Developer HDT822

---

- i. Choose the Right Access Strategy
- ii. Tune Your SQL
- iii. Tune Your Mappings
- 3. Database Tuning
  - i. Denormalize Your Data Schema
  - ii. Rework Database Logs
  - iii. Update Your Database Configuration
  - iv. Reorganize Data Storage
  - v. Rework Your Database Architecture/Design
- 4. Application Tuning
  - i. Share Common Logic
  - ii. Implement Lazy Reads
  - iii. Introduce Caches
  - iv. Combine Granular Functionality

### **XVI. Tools for Evolutionary Database Development**

- A. Tools
- B. Sandboxes
- C. Scripts

### **XVII. Implementing Concurrency Control**

- A. The Role of the DBA
- B. Collisions
  - 1. Types of Locking
    - i. Pessimistic Locking
    - ii. Optimistic Locking
    - iii. Overly Optimistic Locking
  - 2. Resolving Collisions
  - 3. The Basics of Transactions
    - i. ACID Properties
    - ii. Two-Phase Commits
    - iii. Nested Transactions
  - 4. Implementing Transactions
    - i. Database Transactions
    - ii. Object Transactions
    - iii. Distributed Object Transactions
    - iv. Including Nontransactional Sources in a Transaction





## Agile Database Techniques: Effective Strategies for the Agile Software Developer HDT822

---

### **XVIII. Finding Objects in Relational Databases**

- A.** The Role of the DBA
- B.** Find Strategies
  - 1. Brut Force (Embedded SQL)
  - 2. Query Objects
  - 3. Meta Data-Driven
  - 4. When to Apply Each Strategy
- C.** Implementation Techniques
  - 1. Use the Native Error-Handling Strategy
  - 2. Expect “Logic” Errors
  - 3. Always Return a Collection
  - 4. Use Proxies and Lazy Initialization for Search Lists
  - 5. Use Lazy Reads for High-Overhead Attributes
  - 6. Program for People
- D.** Representing Find Results

### **XIX. Implementing Referential Integrity and Shared Business Logic**

- A.** The Role of the DBA
- B.** How Object Technology Complicates Referential Integrity
  - 1. Multiple Entity/Relationship Representation
  - 2. Object Relationship Management
    - i. Database Cascades
    - ii. Object Relationship Cascades
    - iii. Implications of Cascades
    - iv. Cascading Strategies
- C.** Lazy Reads
- D.** Caches
- E.** Aggregation, Composition, and Association
- F.** Architectural Layering
- G.** Removal from Memory versus Persistent Deletion
- H.** Where Should You Implement Referential Integrity?
  - 1. Referential Integrity Implementation Options
  - 2. Business Logic Implementation Options
  - 3. General Implementation Strategies



## Agile Database Techniques: Effective Strategies for the Agile Software Developer HDT822

---

### **XX. Implementing Security Access Control**

- A.** The Role of the Agile DBA
- B.** Authentication
- C.** Authorization
  - 1. Issues
  - 2. Database Implementation Strategies
  - 3. Security Design Patterns
  - 4. Object-Oriented Implementation Strategies
  - 5. Implications
- D.** Effective Security Strategies

### **XXI. Implementing Reports**

- E.** The Role of the Agile DBA
- F.** Database Deployment Architecture
- G.** Reporting Within Your Application
- H.** Reporting Outside Your Application
- I.** Database Design Strategies
- J.** Implementation Strategies
- K.** Challenges That Make Reporting Hard

### **XXII. Realistic XML**

- A.** The Role of the Agile DBA
- B.** An XML Primer
  - 1. Strengths of XML
  - 2. Weaknesses of XML
- C.** Practical Applications of XML
- D.** Vocabularies
- E.** How to Model XML
- F.** XML Mapping and Data Binding
- G.** How to Persist XML in Relational Databases
- H.** How to Persist XML in XML Database
- I.** XML Development Strategies



## Agile Database Techniques: Effective Strategies for the Agile Software Developer HDT822

---

### **XXIII. How You Can Become Agile**

- A. You Do Not Have to Be Superhuman
- B. Agility Is Really Just a Method
- C. Become a Generalizing Specialist

### **XXIV. Bringing Agility Into Your Organization**

- A. Change the Way You Look at Software Development
- B. Understand the Challenges You Face
- C. Actually Try It
- D. Block Nonagile Coworkers
- E. Be Realistic
- F. Parting Thoughts