

## Teaching Data Management

Larry Burns  
Consultant  
PACCAR Data Services

### The Need for Teaching Data Mgt.

- The importance of applications is easily understood and recognized; the importance of data much less so.
- Data is seen from an application-centric point of view; the business value of data reuse is not always recognized.
- Data management is seen as an impediment to application development.

2

## The Need for Teaching Data Mgt.

- Data management is seen as being too tied to relational database technology (and theory).
- Data management practitioners are seen as being dogmatic, impractical, and unhelpful.

3

## Who do we need to teach?

- Application Developers
- Project Managers
- Business Users
- Management

4

## Application Developers...

- Tend to be application- or project-focused
- Need to develop apps. as quickly as possible
- Appreciate abstraction and encapsulation
- Are looking for performance, security, ease of use, and reduced maintenance
- Need to work with non-relational data
- Don't care about relational database theory

5

## Project Managers...

- Tend to be project-focused, but have a somewhat broader business view
- Need to develop apps. as quickly as possible
- Want apps. that satisfy the business users
- Want quality products at a lower cost
- Don't care about relational database theory

6

## **Business Users...**

- Are focused in a particular business subject area
- Want to see a higher ROI from projects
- Are more amenable to the idea of data reuse
- Want easy-to-use, well-performing applications with a low cost of maintenance
- Don't care about relational database theory

7

## **Management...**

- Are concerned with the performance and financial health of the company
- Want to focus on opportunities, not maintaining the status quo
- Are very amenable to the idea of data reuse (if properly presented)
- Don't care about relational database theory

8

## **What do we need to teach?**

- Application Independence (data reuse)
- Data Integrity
- Data Security
- Performance
- Ease of Use
- Low Cost of Maintenance

**9**

## **Application Independence**

- Ensure business relevance of data (for maximum usability)
- Enable data reuse (for increased ROI)
- Encompass business subject areas (to maximize reuse)

**10**

## **Application Independence**

- Avoid tight coupling of applications with data to ensure scalability, performance, ease of maintenance and enhancement.
- Facilitate object reuse.

**11**

## **Data Integrity**

- Ensure that data always has a valid business meaning and value
- Always reflect a valid state of the business
- Data should be (as much as possible) self-monitoring and self-correcting

**12**

## Data Security

- Four pillars of Information Ethics (Mason):
  - Privacy, Accuracy, Property, Accessibility
- Make sure that true and accurate data is always available to authorized persons, but **only** to authorized persons
- Address the privacy concerns of customers, partners, and regulators

13

## Performance / Ease of Use

- Ensure quick and easy access to data by approved users in a usable and business-relevant form
- Maximize the business value of applications (and data)
- Improve the cost-effectiveness of business processes
- Increase user/customer satisfaction

14

## **Low Cost of Maintenance**

- Ensure that the maintenance cost of data and applications does not exceed their business value
- Enable the fastest possible response to changes in business processes and new requirements

15

## **Database Best Practices**

- Use a normalized **logical** design (and base table structure) for application independence
- Use views to create non-normalized, application-specific, object-friendly **physical** views of data
- Abstract and encapsulate database structure and functionality as much as possible

16

## **Database Best Practices**

- Enforce integrity and security constraints at the database (not application) level
- Keep database processing on the database server as much as possible

17

## **Implementing Best Practices**

- Automate database development processes as much as possible
- Use recommendations, not requirements
- Link to support levels
- Get project and support requirements up-front
- Communicate, communicate, communicate!

18

## **What should our approach be?**

- Make using the database as quick, easy and painless as possible
- Stay business-focused
- Adopt a “can do” attitude; be as helpful as possible
- Accept defeats and failures as “lessons learned”

**19**

## **What should our approach be?**

- Communicate with people on their level, and in their terms
- Concentrate on solving **their** problems, not yours

**20**

## What forums can we use?

- Classes (e.g., DBIX)
- Blogs
- Online forums
- Meetings
- One on one

21

## Bio and Contact Information

Larry Burns has worked in IT for more than 25 years as a database administrator, application developer, consultant and teacher. He holds a B.S. in Mathematics from the University of Washington and a Masters degree in Software Engineering from Seattle University. He currently works for Paccar ITD Data Services as a database consultant on numerous application development projects, and teaches a series of data management classes for application developers. He is also an instructor and advisor in the certificate program for Data Resource Management at the University of Washington in Seattle. You can contact him at [Larry.Burns@paccar.com](mailto:Larry.Burns@paccar.com).

22

## **Supplement: Abstraction and Encapsulation**

- Abstraction
  - Identifying the “what”
- Encapsulation
  - Packaging the “how”
- Present an easy-to-use interface that enables the “what” and hides the “how”
- Exs: light switch, car dashboard

**23**

## **Abstraction and Encapsulation (cont'd)**

- Abstraction and Encapsulation in Database Design:
  - Fundamental Stored Procedures (FSPs)
  - Data Access Components / Layers
  - Views
  - Stored Procedures
  - Functions
  - Triggers

**24**

## Abstraction and Encapsulation (cont'd)

- Fundamental stored procedures (FSPs)
  - Handle transaction control
  - Perform error handling
  - Enforce security
  - Maintain supertype/subtype relationships
  - Handle concurrency control via timestamp checking
  - Provide multi-language text support
  - Automatically generated from DB schema

25

## Abstraction and Encapsulation (cont'd)

- Data Access Component (DAC)
  - Automatically handles updates using FSPs
  - Creates, populates and executes ADO.NET objects for queries, procedures, typed datasets, connections, transactions, etc.
  - Maintains database timestamps and uses them for updates
  - Works with any RDBMS

26

## **Abstraction and Encapsulation (cont'd)**

- **Joined Views**

- Create an application-specific view of data
- Developers don't have to code complex SQL joins
- Results in greatly improved performance
- Views can map directly to application objects
- Can join relational and XML data
- Views can enforce security and support encryption

**27**

## **Abstraction and Encapsulation (cont'd)**

- **Stored Procedures**

- Can encapsulate data-specific application or business processes
- Results in greatly improved performance
- Reduce network traffic
- Makes debugging, performance tuning and maintenance easier
- Can be used to enforce security

**28**

## **Abstraction and Encapsulation (cont'd)**

- **Functions**
  - Useful for datatype conversion and data reformatting
  - Can make relational data look like XML (and vice-versa)

**29**

## **Abstraction and Encapsulation (cont'd)**

- **Triggers**
  - Can encapsulate data-specific application or business processes
  - Useful for complex and cross-database RI checking and updating
  - “Instead Of” triggers can be used to map updates on views to underlying base tables

**30**