RUTGERS

Rutgers Business School
Newark and New Brunswick

33:010:458
Accounting Information Systems

Dr. Peter R. Gillett

Associate Professor
Department of Accounting, Business Ethics and Information Systems
Rutgers Business School–Newark and New Brunswick

A.I.S. Class 11: Outline

- Learning Objectives for Chapter 9
- Designing the Data Repository Structure
- Event-Oriented Modeling
- Database Normalization
- Group Work for Chapter 9
- Mid-term Examination
- Classroom Assessment

Learning Objectives for Chapter 9

- After studying this chapter you should be able to:
  * convert an extended entity relationship diagram to relational tables
  * articulate the conversion rules for mandatory relationships
  * articulate the conversion rules for optional relationships
  * explain the concept of database normalization

Learning Objectives for Chapter 9

- After studying this chapter you should be able to:
  * describe the rules for determining whether a table conforms to first, second, or third normal form
  * explain the process of implementing tables in a relational database system such as Microsoft Access
  * explain how forms in a relational database management system are used to implement information processes

Designing the Data Repository Structure for RDBMS

- There are three main issues involved in this step:
  * Identifying the required tables
  * Linking the tables
  * Specifying the attributes for the tables

Identifying Required Tables

- Every entity in the REA model will require a table with a primary key (subject to a possible decision to merge certain tables together during linking, or omit tables with only one row)
- So there should normally be a table for every:
  * resource
  * event
  * agent
- Additional tables may be required to model certain types of links between entities (see next slide)
- Finally, tables may be required for reference purposes by the information system (e.g., tables of valid State Codes, Discount Codes, etc.)

Linking Tables - Choices

- Create separate Link table
  * will always work
  * may often be unnecessary
  * sometimes the only choice (e.g., M:M)
- Post foreign keys
- Combine tables
- *All three choices would work for 1:1*

Linking Tables

- 1:1 Links
  - collapse the two entities into a single table
    or
  - post the primary key of either table as a foreign key in the other

  I will always want you to post a foreign key; if one entity is an event and the other not, post the non-event key as a foreign key on the event table.
- 1:M and M:1 Links
  - post the primary key for the '1' table as a foreign key in the 'M' table
- M:M Links
  - create a new Link Table with the primary keys from the original tables forming a composite key; add any uniquely defined attributes

Linking Tables

- Except:
  * For optional entities
    - always treat as if their cardinality were M
    - i.e., treat both 0 .. 1 and 0 .. * entities as 0 .. *
  * When modeling two events linked 1:1
    - post the key of the first event as a foreign key in the table for the second
  * When modeling two events linked1:M where the '1' event follows the 'M' event
    - treat the link as a M:M link - i.e., create an additional Link Table
  * *Why do these exceptions almost always produce the same results as the methods set out in the chapter – because the second of two related events is almost always optional in the relationship with the first, and so is in any case treated as many*

Orville Ornaments: Data Repository Structure

- JobOrder (Job#, Date, StartDate, ScheduledCompletionDate, [ProductionManger#])
- MaterialsRequisition (MaterialsRequisition#, Date, [Job#], [InventoryManger#])
- MaterialsIssue (MaterialsIssue#, Date, [MaterialsRequisition#])
- MakeOrnament (Batch#, Date, [Job#], [Craftsman#], [Ornament#], QuantityMade)

- Ornament (Ornament#, Description, StorageBin#, StandardCost, SellingPrice)
- Material (Material#, Description, StorageBin#, StandardCost)

- ProductionManager (ProductionManager#, Name, Address, DateHired, Salary, etc.)
- InventoryManager (InventoryManager#, Name, Address, DateHired, Salary, etc.)
- SuppliesClerk (SuppliesClerk#, Name, Address, DateHired, Salary, etc.)
- Craftsman (Craftsman#, Name, Address, DateHired, Salary, etc.)

Orville Ornaments: Data Repository Structure

- JobOrder-Ornament ([Job#], [Ornament#], QuantityRequired)
- MaterialsRequisition-Material ([MaterialsRequisition#], [Material#], QuantityRequired])
- MaterialsIssue-Material ([MaterialsIssue#], [Material#], QuantitytIssued)
- MaterialsIssue-SuppliesClerk ([MaterialsIssue#], [SuppliesClerk#])

- Ornament-Craftsman ([Ornament#], [Craftsman#])

Data Repository Structures

- Storing Balances
  - Technically wrong
  - Sometimes still done for convenience
  - Imposes extra processing burden to ensure correct
- Recording agents: customers/vendors
  - Can you be certain who you shipped to and who you paid?
  - Omit if you can
  - For now (and the Mid-Term!) assume not! . . .
- Inventory
  - Are items uniquely identified (like cars) or not (like books)
- Cash as a resource
  - Cash account numbers

Implementing the Design

1. Create the Access tables required by the design
2. Designate the primary keys
3. Establish relationships between tables
4. Create forms to maintain the tables for each resource and agent
5. Create (multi-table) forms for event recording processes
6. Create queries to generate desired information
7. Develop report formats for the desired reports
8. Build a custom menu system

Database Normalization

- Relational databases and other data processing systems can suffer from a number of anomalies
- Proper application of REA modeling should avoid these problems by automatically generating 3NF data
- Problems may be encountered, though, with data designed in other ways

Database Normalization

- Data anomalies
  - * Insertion anomalies
    - inability to add certain data
  - * Deletion anomalies
    - deleting data in one place causes a loss of other data that needs to be retained
  - * Update anomalies
    - changes must be made in multiple locations

Database Normalization

- Functional dependency
  - If in a table with two attributes, X and Y, there is only one possible value of Y for each possible value of X, Y is said to be functionally dependent on X
  - Thus secondary keys are functionally dependent on the primary key (although this is not the only kind of functional dependency)

Database Normalization

- Derived fields
  - * Transitive dependencies
    - Functional dependencies not originating from the primary key
  - * Calculated fields
    - Functionally dependent on values in other tables

Database Normalization

- Unnormalized tables
- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)

- Boyce/Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)
- Domain/Key Normal Form (DK/NF)

Database Normalization

- 1NF
  - No repeating groups
- 2NF
  - 1NF + Full dependence on the entire key
- 3NF
  - 2NF + No derived fields

Fabulous Furniture Forum

Fun Family Furniture

Sales Invoice

Customer:
Benjamin Sisko
Deep Space Nine
Alpha Quadrant

Customer ID: 9876

Invoice #: 12345

Date: 12/31/2350

Salesman: Jean-Luc Picard

1 x Executive Desk @ $55,000	55,000
2 x Office Chairs @ $3,500	7,000
Subtotal:	62,000
Intergalactic Shipping	33,000
Total:	$95,000

Database Normalization

- What data is on this invoice?
- (Customer#, Customer, Address, Invoice#, Date, Salesman, {Qty, Item, Price, Value}, Shipping, Total)
  * This is unnormalized data because of the recurring items
  * Value and Total should not be stored because they can be calculated from other data

Database Normalization

- (Invoice#, Customer#, Customer, Address, Date, Salesman, Shipping)
- ([Invoice#], Line#, Qty, Item, Price)
  * These tables are 1NF
  * The Invoice table is 2NF but not 3NF because Customer and Address depend on Customer# which depends on Invoice#
  * Salesman names and Item descriptions are repeated on these tables unnecessarily
  * On the Invoice-Line table Price is not determined by the key at all - we still need to know the Item number

Database Normalization

- (Invoice#, [Customer#], Date, [Salesman#], Shipping)
- (Customer#, Customer, Address, . . .)
- (Salesman#, Salesman, . . .)
- (Item#, Item, . . .)
- ([Invoice#], Line#, [Item#], Qty, Price)
  - But now the final table is not 2NF because Price depends only on [Item#] - and Line# is useless

Database Normalization

- (Invoice#, [Customer#], Date, [Salesman#], Shipping)
- (Customer#, Customer, . . .)
- (Salesman#, Salesman, . . .)
- (Item#, Item, Price, . . .)
- ([Invoice#], [Item#], Qty)
  * These tables are 3NF

Chapter 9 Appendix

- Reading the Appendix at this stage helps orient you with respect to what needs to be done
- However, we will be studying ACCESS much more thoroughly in the second half of the semester
- Consequently, I will not hold you responsible for this Appendix on the Mid-Term examination

Group Work for Chapter 9

- Problems 8, 9 & 12

Mid-Term Examination

- Main features last year
  * 250 points
  * Five main sections
    - A: 25 multiple-choice questions (similar to quizzes) worth 3 points each (75 points – 30%)
    - B: Flowcharting (21 points – 8%)
    - C: An REA modeling problem (75 points – 30%)
    - D: Some abbreviations worth 1 points each (10 points – 4%)
    - E: Short questions and problems (69 points – 28%)
  * 80 minutes total time: 8:10 – 9:30 next Wednesday night in Beck
  * Constrained by time not difficulty . . . 3 points per minute . . .
  * . . . so you will be penalized for turning in late

Classroom Assessment

- The most important thing I have learned in this class so far is:

- Of all the topics I have studied in A.I.S. so far, the least clear to me is: