DB2 Database Design: From Logical to Physical

Mullins Consulting, Inc.
http://www.CraigSMullins.com
Author

This presentation was prepared by:

Craig S. Mullins
President & Principal Consultant

Mullins Consulting, Inc.
15 Coventry Ct.
Sugar Land, TX 77479
Tel: 281-494-6153
E-mail: craig@craigsnullins.com

This document is protected under the copyright laws of the United States and other countries as an unpublished work. This document contains information that is proprietary and confidential to Mullins Consulting, Inc., which shall not be disclosed outside or duplicated, used, or disclosed in whole or in part for any purpose other than as approved by Mullins Consulting, Inc. Any use or disclosure in whole or in part of this information without the express written permission of Mullins Consulting, Inc. is prohibited. © 2011 Craig S. Mullins and Mullins Consulting, Inc. (Unpublished). All rights reserved.
Why Model Data?

Capital - Chart of Accounts

Data?

Human Resources - org chart

Facilities - blueprints

Materials - Bill of Materials
Data Modeling Objectives

- Document and communicate the business information requirements
- Enable databases with:
  - Minimum redundancy
  - Maximum integrity
  - Sharability
  - Stability
  - Flexibility
  - Consistency
  - Timely access
  - Usability
- Increase value of data resource
Data Thinking

- Don’t think “how”; think “what”
- Don’t think physical; think conceptual
- Don’t think process; think structure
- Don’t think navigation; think relationship
Data Modeling Concepts

Concepts to be covered will include:
- Entities
- Entity Types
- Entity Occurrences
- Relationships
- Attributes
- Attribute Roles
- Keys
- Diagramming Techniques
Entities

• Something that exists and is capable of being described.
• A person, place, thing, concept or event about which an organization maintains facts.

Student  Instructor
Order      Course
Employee  Item
**Entity Examples**

- **Person:** Roles people play  
  - Employee  
  - Supplier  
  - Customer  
  - Agent

- **Place:** Areas or geographic locations  
  - Office  
  - City  
  - Warehouse  
  - Region

- **Thing:** Physical object type  
  - Item  
  - Product  
  - Material  
  - Part  
  - Building  
  - Equipment

- **Event:** Things that "happen"  
  - Sale  
  - Transfer  
  - Project  
  - Reservation  
  - Order  
  - Shipment  
  - Agreement  
  - Flight

- **Concept:** Intangible ideas  
  - Warranty  
  - Account  
  - Route

---

**Diagram:**

- Entity
  - Person
  - Place
  - Thing
  - Event
  - Concept
Entity Occurrence

Entity Type

Customer: Moore
City: Pittsburgh
State: PA
Phone: 555
Credit: AMEX:
AMEX:
VISA:
Contact:

Customer: Jackson
City: An
State: MI
Phone: 555
Credit: AMEX:
AMEX:
VISA: xxxxxxxxxxxxxxxxxxx
Contact: Mike

Customer: Mullins
City: Houston
State: TX
Phone: 555-1234
Credit: Fair
AMEX: xxxxxxxxxxxxxxxxxxxx
VISA: xxxxxxxxxxxxxxxxxxxx
Contact: Mike

Entity Occurrences
Entity Naming Guidelines

• Noun, or Adjective-Noun format
  – Contract
  – Lease Agreement

• Minimize use of adjectives
  – Contractor vs Contract Employee
  – Training vs Employee Course

• Do not pluralize; use singular instead: Employee vs Employees
  – The entity type is a model or pattern rather than the set of all employees

• Use business terms and be consistent
  – Vendor or Supplier?
  – User or Client?

• Remove process specific artifacts from the entity name
  – State not Residence State
  – Agent not Selling Agent
• How the different entities are associated with each other
Diagramming Conventions

<table>
<thead>
<tr>
<th>One-to-One</th>
<th>One-to-Many</th>
<th>Many-to-Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE → TEMP_EMP</td>
<td>CUSTOMER → ACCOUNT</td>
<td>STUDENT → COURSE</td>
</tr>
<tr>
<td>EMPLOYEE → TEMP_EMP</td>
<td>CUSTOMER → ACCOUNT</td>
<td>STUDENT → COURSE</td>
</tr>
<tr>
<td>EMPLOYEE → TEMP_EMP</td>
<td>CUSTOMER → ACCOUNT</td>
<td>STUDENT → COURSE</td>
</tr>
<tr>
<td>EMPLOYEE 1-1 TEMP_EMP</td>
<td>CUSTOMER 1-M ACCOUNT</td>
<td>STUDENT M-M COURSE</td>
</tr>
<tr>
<td>EMPLOYEE → TEMP_EMP</td>
<td>CUSTOMER → ACCOUNT</td>
<td>STUDENT → COURSE</td>
</tr>
</tbody>
</table>
Data Model Guidelines

⇒ The relationship is read *clockwise* over the line

Reads left-to-right

```
Order ⟷ Is placed by ⟷ Customer
```

Read bottom-to-top

```
Order ⟷ Places ⟷ Customer
```

Read top-to-bottom

```
Order ⟷ Is placed by ⟷ Customer
```

Read right-to-left

```
Order ⟷ Places ⟷ Customer
```
Types of Entities

- Examining relationships can help to determine the type of entity:

  - **Fundamental Entities** - represent fundamental, or basic, business objects

  - **Characteristic Entities** - contain multiple attributes or facts describing a basic entity

  - **Associative Entities** - describe a relationship between two other entities
Creating an Associative Entity
Attributes

Customer: Smith
Customer #: 0001
City: New York
State: New York
Phone: 555-1938
Credit: Good

ENTITY = Nouns
ATTR = Adjectives

Customer
  - Cust Number
  - Cust Name
  - Street Address
  - City
  - State
  - Zip Code
  - Phone Num
  - Credit Limit Amt
Attributes

- An attribute is a fact about an entity. It is a data element that is an inherent property of an entity.

- An attribute will fulfill one \((\text{and only one})\) of these objectives:
  - Describe
  - Identify
  - Relate
# Attribute Naming Guidelines

<table>
<thead>
<tr>
<th>CLASS</th>
<th>ABBREVIATION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>ADDR</td>
<td>Address or location</td>
</tr>
<tr>
<td>AMOUNT</td>
<td>AMT</td>
<td>Monetary amount</td>
</tr>
<tr>
<td>CODE</td>
<td>CODE</td>
<td>Classifications, codes, types</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>Calendar dates</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>DESC</td>
<td>Descriptive text</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>ID</td>
<td>Alphanumeric unique identifier</td>
</tr>
<tr>
<td>IMAGE</td>
<td>IMG</td>
<td>Encoded digital image</td>
</tr>
<tr>
<td>NAME</td>
<td>NAME</td>
<td>Alphabetic name, identification</td>
</tr>
<tr>
<td>NUMBER</td>
<td>NUM</td>
<td>Numeric count</td>
</tr>
<tr>
<td>PERCENT</td>
<td>PCT</td>
<td>Relationships as %</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>QTY</td>
<td>Counts and units</td>
</tr>
<tr>
<td>TEXT</td>
<td>TXT</td>
<td>Free form document text</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
<td>Time; temporal data</td>
</tr>
</tbody>
</table>
Keys

- A key is an attribute, or group of attributes, that are used to identify.

- There are three types of keys:
  - Candidate Keys
  - Primary Keys
  - Foreign Keys
Primary Keys

- The attribute or attributes that *uniquely identify* an entity type
- A primary key value uniquely identifies a single occurrence of an entity

<table>
<thead>
<tr>
<th>Order</th>
<th>Order #: 0001</th>
<th>Line #: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProdID: 1234</td>
<td>Qty: 5</td>
<td></td>
</tr>
</tbody>
</table>
Foreign Keys

*(Relationship Attributes)*

- An attribute (or attributes) in one entity that relates an occurrence of that entity to an occurrence of another entity
- The relating attribute(s) contains the same value(s) as the primary key of the related entity occurrence
Normalization

The process of identifying the one best place a fact belongs.

- **First Normal Form**
  A row is in first normal form if and only if all underlying domains contain atomic values only.

- **Second Normal Form**
  A row is in second normal form if and only if it is in first normal form and every non-key attribute is fully dependent on the key.

- **Third Normal Form**
  A row is in third normal form if and only if it is in second normal form and every non-key attribute is non-transitively dependent on the primary key.
Derived and Base Data

Course
- Course Num
- Course Offering Date
- Course Name
- Course Fee

Enrollment
- Student ID
- Course Num
- Course Offering Date
- Course Completion Date
- Course Charge Back Amt(DT)

Student
- Student ID
- Student Name
- Student Date of Birth
- Student Age(D)
Data Model of Modeling Terms

Relationship
  \|-- Occurrence
      \|-- Entity
          \|-- Domain
              \|-- Attribute
                  \|-- Attribute Value
                  \|-- Primary Key
# Terminology Summary

<table>
<thead>
<tr>
<th>Common Term</th>
<th>Graphic Term</th>
<th>DP Term</th>
<th>Design Term</th>
<th>Relational Term</th>
<th>Object-Oriented Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Cabinet</td>
<td>Table</td>
<td>File</td>
<td>Entity</td>
<td>Relation, Table</td>
<td>Type, ADT, Class</td>
</tr>
<tr>
<td>File Folder or Record</td>
<td>Row</td>
<td>Record</td>
<td>Occurrence</td>
<td>Tuple, Row</td>
<td>Instance, Object</td>
</tr>
<tr>
<td>Fact</td>
<td>Column</td>
<td>Field</td>
<td>Attribute</td>
<td>Column (Domain)</td>
<td>Property</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data Item Data Element</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td>Identifier</td>
<td>Record Key</td>
<td>Primary Key</td>
<td>Primary Key</td>
<td>Object Identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Transforming Logical to Physical

- Translation of Logical Model to Physical Database
  - Create DDL
  - Entities to Tables, Attributes to Columns, Relationships and Keys to DB2 RI and Indexes, etc.
  - …but differences CAN and WILL occur
- Create Storage Structures for Database
  - Files for data and indexes
  - Partitioning
  - Clustering
  - Optimization
  - Etc.
DB2 Object Overview

STOGROUP

DATABASE

VOLC01

TABLESPACE
TABLEX
TABLEY

TABLESPACE
TABLEA
TABLEB
TABLEC
TABLEB
TABLEA

INDEX SPACE
INDEX on TABLEY

VOLC02

TABLESPACE
TABLEZ
(Partition 1)
(Partition 2)
(Partition 3)

INDEX SPACE
INDEX on TABLEZ
(Partition 1)
(Partition 2)
(Partition 3)

INDEX SPACE
INDEX on TABLEY

Background DB2 Files

LINEAR VSAM

DB2 CATALOG

MVS ICF CATALOG
DB2 Object Hierarchy
DB2 Data Types

- CHAR
- VARCHAR
- CLOB
- DBCLOB
- GRAPHIC
- VARGRAPHIC
- BLOB
- DATE
- TIME
- TIMESTAMP
- TIMESTAMP w/TIMEZONE
- XML
- BIGINT
- INTEGER
- SMALLINT
- DECIMAL
  - NUMERIC
- FLOAT
  - REAL
  - DOUBLE
- DECFLOAT
- BINARY
- VARBINARY
### Character and Numeric Data Types

<table>
<thead>
<tr>
<th>DATA TYPE</th>
<th>DESCRIPTION</th>
<th>LENGTH</th>
<th>STORAGE REQUIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(X)</td>
<td>Fixed Length String</td>
<td>Max 254</td>
<td>Up to 254 Bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA TYPE</th>
<th>DESCRIPTION</th>
<th>VALUE RANGE</th>
<th>STORAGE REQUIRED</th>
<th>COBOL PICTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>Whole Numbers</td>
<td>± 32,767</td>
<td>Half Word</td>
<td>S9(4) COMP</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Whole Numbers</td>
<td>± 2 Billion</td>
<td>Full Word</td>
<td>S9(9) COMP</td>
</tr>
<tr>
<td>DECIMAL (X,Y)</td>
<td>X-Total Digits Y-# of Decimal Digits</td>
<td>Up to 31 Significant Digits</td>
<td>Packed Format</td>
<td>S9(x-y) v9 (x) COMP-3</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Scientific Notation</td>
<td>± 7.2E +75 ± 5.4E -79</td>
<td>Double Word</td>
<td>COMP-1 or COMP-2</td>
</tr>
</tbody>
</table>

Columns that contain numbers should be numeric
No display numeric in DB2
Decimal: Precision and Scale

• The precision of a decimal number is the total number of digits in the number (do not count the decimal point).
  – For example, the number 983.201 has a precision of 6.

• The scale of a decimal number is equal to the number of digits to the right of the decimal point.
  – In the previous example, the scale is 3.
Using INTEGER and DECIMAL

<table>
<thead>
<tr>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>DEC(4,2)</td>
<td>???</td>
</tr>
<tr>
<td>6</td>
<td>18.00</td>
<td>???</td>
</tr>
</tbody>
</table>

SET COL3 = (COL1/COL2)

<table>
<thead>
<tr>
<th>COL3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>0</td>
</tr>
<tr>
<td>DEC(5,2)</td>
<td>000.33</td>
</tr>
</tbody>
</table>

Decimal with zero precision will force an explicit number of digits
## VARCHAR and LONG VARCHAR

<table>
<thead>
<tr>
<th>DATA TYPE</th>
<th>DESCRIPTION</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR(X)</td>
<td>Variable Length String</td>
<td>Max 4046</td>
</tr>
<tr>
<td>LONG VARCHAR (4K Page)</td>
<td>Variable Length String</td>
<td>Max 4046 Calculated by DB2</td>
</tr>
<tr>
<td>LONG VARCHAR (32K Page)</td>
<td>Variable Length String</td>
<td>Max 32704 Calculated by DB2</td>
</tr>
</tbody>
</table>

Actual data lengths should vary widely.

Consider compression.
Storing Large Objects:
BLOBs, CLOBs and DBCLOBs
### DATE, TIME and TIMESTAMP

<table>
<thead>
<tr>
<th>DATA TYPE</th>
<th>INTERNAL FORMAT</th>
<th>STORAGE REQUIRED</th>
<th>COBOL PIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>YYYYMMDD</td>
<td>4 Bytes Packed</td>
<td>X (10)</td>
</tr>
<tr>
<td>TIME</td>
<td>HHMMSS</td>
<td>3 Bytes Packed</td>
<td>X(08)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>YYYYMMDD HHMMSS</td>
<td>10 Bytes Packed</td>
<td>X (26)</td>
</tr>
<tr>
<td></td>
<td>NNNNNNN*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* N = Microsecond

Should be used for any columns containing chronological data.
Nulls

INVEN_LOC_TAB

<table>
<thead>
<tr>
<th>WAREHSE_NO</th>
<th>BIN_NO</th>
<th>PROD_NO</th>
<th>PROD_QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>A100</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>A150</td>
<td>2000</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>B167</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>A100</td>
<td>775</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>D400</td>
<td>1585</td>
</tr>
</tbody>
</table>

**NULL:**
Has no value
Is not = anything
Is not < anything
Is not > anything
Is not = NULL
Is an UNKNOWN value

<table>
<thead>
<tr>
<th>ATTRIBUTE QUALIFIER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT NULL</td>
<td>When no value is provided, DB2 automatically assigns nulls to the Column</td>
</tr>
<tr>
<td>NOT NULL</td>
<td>The column must always contain a value, whether a default or explicit value provided</td>
</tr>
</tbody>
</table>
# DEFAULT

<table>
<thead>
<tr>
<th>ATTRIBUTE QUALIFIER</th>
<th>DESCRIPTION</th>
<th>DEFAULT USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>(WITH) DEFAULT</td>
<td>When no value is provided, DB2 automatically assigns an appropriate default</td>
<td>Numeric Data: ZEROS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Character Data: SPACES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Variable Data: ZERO LENGTH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chronological Data: CURRENT DATE CURRENT TIME CURRENT TIMESTAMP</td>
</tr>
<tr>
<td>(WITH) DEFAULT value</td>
<td></td>
<td>Constant USER CURRENT SQLID NULL</td>
</tr>
</tbody>
</table>
Column Ordering

- Sequence columns based on logging
  - Infrequently updated non-variable columns first
  - Static (infrequently updated) variable columns
  - Frequently updated columns last
  - Frequently modified together, place next to each other

<table>
<thead>
<tr>
<th>CUST ID</th>
<th>FIRST NAME</th>
<th>LAST NAME</th>
<th>ADDRESS</th>
<th>ACCT BAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static, infrequently updated</td>
<td>Frequently updated at the same time (marriage) …but infrequently updated.</td>
<td></td>
<td></td>
<td>Frequently updated</td>
</tr>
</tbody>
</table>
Reordered Row Format

• RRF automatically puts variable columns at the end of the row
  – Does not impact the DCLGEN or how you interact with the table in your programs; just changes physical storage on disk.
  – So RRF stores fixed-length columns first and the variable columns at the end.
    • Pointers within the row will point to the beginning of the variable columns.
  – The previous format is now called Basic Row Format, or BRF.
  – Over time, your table spaces will migrate to RRF…
Freespace Alternatives

- Freespace is considered only at LOAD or REORG time

- The freespace options:
  PCTFREE
  FREEPAGE

- For Example:

  PCTFREE 33
  FREEPAGE 2

  ![Diagram showing Freespace Alternatives](image-url)
# Row Size

(to determine page size, free space)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Length (in bytes)</th>
<th>Variable (add 2)</th>
<th>Null (add 1)</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Data Length:**

**Row Overhead:** +8

**Physical Row Length:**

<table>
<thead>
<tr>
<th>Data Lengths:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>4</td>
</tr>
<tr>
<td>DECIMAL PACKED FORMAT</td>
<td>DATE</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>2</td>
</tr>
<tr>
<td>FLOAT (REAL)</td>
<td>4</td>
</tr>
<tr>
<td>FLOAT (DBL PRECISION)</td>
<td>8</td>
</tr>
<tr>
<td>DATE</td>
<td>4</td>
</tr>
<tr>
<td>TIME</td>
<td>3</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>10</td>
</tr>
</tbody>
</table>
Constraints

- Referential
  - PRIMARY KEY
  - FOREIGN KEY
- UNIQUE
- CHECK
  - (CHECK SALARY > 50000.00)
- Triggers
# Categories of Data Integrity

## PRIMARY KEY

<table>
<thead>
<tr>
<th>CUST_NO SMALLINT</th>
<th>CUST_NAME CHAR(30)</th>
<th>CUST_ADDR CHAR(25)</th>
<th>CUST_CITY CHAR(20)</th>
<th>CUST_STATE CHAR(2)</th>
<th>CUST_ZIP INT</th>
<th>CUST_AREA_CODE SMALLINT</th>
<th>CUST_PHONE INT</th>
<th>CREDIT_CAT CHAR(3)</th>
<th>SALES REP INT</th>
<th>COLLECT_AGY_NO SMALLINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACME INC</td>
<td>222 ELM ST</td>
<td>LISLE</td>
<td>IL</td>
<td>60532</td>
<td>708</td>
<td>5551212</td>
<td>AAA</td>
<td>123456</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PXI CORP</td>
<td>5 PXI PLAZA</td>
<td>DALLAS</td>
<td>TX</td>
<td>76543</td>
<td>407</td>
<td>8326745</td>
<td>AAA</td>
<td>273818</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>SYNC CORP</td>
<td>2254 HAMILTON DR</td>
<td>FT.WASHINGTON</td>
<td>PA</td>
<td>19003</td>
<td>215</td>
<td>8983736</td>
<td>C</td>
<td>563490</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>FR STANLEY INC</td>
<td>987 BEAVER DR</td>
<td>ANAPOLIS</td>
<td>MD</td>
<td>30589</td>
<td>301</td>
<td>2734147</td>
<td>AA</td>
<td>123456</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>BUYLO INC</td>
<td>235 IRON ST</td>
<td>BLOOMSBURG</td>
<td>PA</td>
<td>17815</td>
<td>717</td>
<td>9895280</td>
<td>BB</td>
<td>903757</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>WHOS AIRCRAFT</td>
<td>8837 DESERT RD</td>
<td>TUCSON</td>
<td>AZ</td>
<td>80345</td>
<td>602</td>
<td>6743333</td>
<td>A</td>
<td>583490</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>BUSINESS INFO CO</td>
<td>555 WATERSEDGE</td>
<td>LOMBARD</td>
<td>IL</td>
<td>60406</td>
<td>708</td>
<td>4836285</td>
<td>AAA</td>
<td>273818</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>CPU INC</td>
<td>1123 PEACH ST</td>
<td>COLUMBUS</td>
<td>OH</td>
<td>50387</td>
<td>216</td>
<td>4823778</td>
<td>C</td>
<td>999475</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>XYZ CORP</td>
<td>4590 WAYNE RD</td>
<td>LIVONIA</td>
<td>MI</td>
<td>46827</td>
<td>313</td>
<td>3435555</td>
<td>B</td>
<td>999475</td>
<td></td>
</tr>
</tbody>
</table>

## FOREIGN KEY

<table>
<thead>
<tr>
<th>SALES_ORDR_NO INT</th>
<th>ORDR_DATE DATE</th>
<th>CUST_NO SMALLINT</th>
<th>SALES_HIST Cust_NO SMALLINT</th>
<th>ORDR_AMT DEC(9,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1991-09-15</td>
<td>2</td>
<td>2</td>
<td>1923.45</td>
</tr>
<tr>
<td>3</td>
<td>1991-09-23</td>
<td>1</td>
<td>1</td>
<td>2407.53</td>
</tr>
<tr>
<td>4</td>
<td>1991-09-23</td>
<td>10</td>
<td>10</td>
<td>57613.89</td>
</tr>
<tr>
<td>5</td>
<td>1991-09-29</td>
<td>3</td>
<td>5</td>
<td>67000.00</td>
</tr>
<tr>
<td>6</td>
<td>1991-10-01</td>
<td>2</td>
<td>2</td>
<td>42345.88</td>
</tr>
<tr>
<td>7</td>
<td>1991-10-02</td>
<td>2</td>
<td>2</td>
<td>122345.61</td>
</tr>
<tr>
<td>8</td>
<td>1991-10-02</td>
<td>7</td>
<td>7</td>
<td>23007.34</td>
</tr>
<tr>
<td>9</td>
<td>1991-10-05</td>
<td>1</td>
<td>1</td>
<td>9823.55</td>
</tr>
<tr>
<td>10</td>
<td>1991-10-07</td>
<td>5</td>
<td>5</td>
<td>223019.27</td>
</tr>
<tr>
<td>11</td>
<td>1991-10-11</td>
<td></td>
<td>7</td>
<td>78780.99</td>
</tr>
</tbody>
</table>

## ASSOCIATION INTEGRITY

Each city/state pair has a valid zip code.

## DOMAIN INTEGRITY

Each value of CREDIT_CAT is valid.

## REFERENTIAL INTEGRITY

Each value of CUST_NO exists as a value of CUST_NO in CUST_TAB.
RI: System or User-Managed?

Referential constraints & Triggers

- Standard declarative implementation.
- Less coding required.
- Easier to modify later. (DDL and CHECK)
- More efficient.
- Ad hoc and planned updates.

Program logic

- Requires program code to be written.
- Hard to modify later.
- Sometimes there is the possibility for better insert performance.
- Works only for planned updates.
Normalized vs. Denormalized

**Normalized Tables:**
- More Tables
- Fewer Columns per Table
- Fewer Rows per Table
- Less Redundancy
- More Joins
- Update Efficient

**Denormalized Tables:**
- Fewer Tables
- More Columns per Table
- More Rows per Table
- Greater Redundancy
- Fewer Joins
- Read Efficient
Denormalization

- **Prejoined Tables** - when the cost of joining is prohibitive
- **Report Tables** - for specialized critical reports (e.g. CEO)
- **Mirror Tables** - when two types of environments require concurrent access to the same data *(OLTP vs DSS)*
- **Split Tables** - when distinct groups/apps use different parts of the same table
  - Splitting *columns* across two tables for long variable character columns.
- **Combined Tables** - to eliminate one-to-one relationships
- **Redundant Data** - to reduce the number of joins for a single column (e.g. definitional, CA to California)
- **Repeating Groups** - to reduce overall I/O (& possibly DASD)
- **Derivable Data** - to eliminate calculations & aggregations
- **Speed Tables** - to support hierarchies
- **Page Size** - to reduce page size
Table to Tablespace Relationship

### Journal Table

<table>
<thead>
<tr>
<th>JOURNAL #</th>
<th>JOURNAL DESC.</th>
<th>JOURNAL DEPT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PAYABLES</td>
<td>11210</td>
</tr>
<tr>
<td>2</td>
<td>PURCHASES</td>
<td>11211</td>
</tr>
<tr>
<td>3</td>
<td>CASH RECEIPTS</td>
<td>11211</td>
</tr>
<tr>
<td>4</td>
<td>SALES</td>
<td>11212</td>
</tr>
</tbody>
</table>

### Tablespace

- **DATA PAGE 1**: 1 PAYABL
  - **DATA PAGE 1**: ES12110...
  - **DATA PAGE 2**: 212...
- **DATA PAGE 7**: 3CASH R
  - **DATA PAGE 7**: ECEIPTS11211...
- **DATA PAGE 6**: 2PURCHA
  - **DATA PAGE 6**: SES11211...
Segmented Tablespaces:

- Are divided into equal sized page groups called 'segments'
- Segments are:
  - An ordered set of contiguously stored pages
  - From 4 to 64 pages long (in multiples of 4)
  - Allocated by DB2 as needed
- SEGMENTS contain only the rows of the table to which the segment is assigned:

![Diagram of Segmented Tablespace]
Partitioned Tablespace

Partitioned Tablespace

Partition 1
(Rows with values up to 333)

Partition 2
(Rows with values 334-666)

Partition 3
(Rows with values > 666)

Partitioned Indexspace

Partition 1
(Index entries for values up to 333)

Partition 2
(Index entries for values 334-666)

Partition 3
(Index entries for values > 666)

Global Index

Entries for all Partitions
Universal Table Spaces

• The future (and present) of DB2 table spaces
• Combine many of the best features of partitioned and segmented
  – Only one table per table space
• Partition by Growth
  – DB2 creates a new partition, as needed, as more data gets added to the table space
    • MAXPARTITIONS controls how many partitions can be created
• Range-Partitioned
  – More like a traditional partitioned table space
  – Control over partitions and which ranges of data go into which partition
Related tables should always be placed in the same database.
Tablespace to STOGROUP Relationship

User-Defined

Partition - 1
Partition - 2

STOGROUPs

Simple or Segmented
Tablespace

3390 Storage Group

3380 Storage Group

VSAM LDS

User responsible for allocation and deletion of data sets

Data sets allocated and deleted by DB2
DB2 Indexes

Index Advantages

Optimize data access:

- DB2 decides whether or not to use an index at BIND time
- DB2 maintains all indexes
- Tablespace scans can be avoided through index usage
- Recommended on foreign key columns to speed referential integrity access
- Indexes can minimize sorting
- There can be multiple indexes per table to suit the way data is processed

Guarantee uniqueness:

- Only means of ensuring uniqueness of column values
- Required on primary key column as part of referential integrity implementation

Implement clustering:

- DB2 will attempt to maintain rows in the sequence of the column values in the index
Indexing Strategy

• A proper indexing strategy can be the #1 factor to ensure optimal performance
  – First take care of unique & PK constraints
  – Then for foreign keys *(usually)*
  – Heavily used queries - predicates
  – Overloading of columns for IXO
  – Index to avoid sorting
    • ORDER BY, GROUP BY, DISTINCT
  – Choose first column wisely (high cardinality)
  – Choose clustering index wisely
  – Indexing variable columns?
  – Consider the cost of indexes
Index Organization

Indexspace

Level 1
Root Page

Level 2
Nonleaf Page
Nonleaf Page

Level 3
Nonleaf Page
Nonleaf Page
Nonleaf Page

Level 4
Leaf Page
Leaf Page
Leaf Page
Leaf Page

... to table data...
### What is a View?

#### Course Table = TCRSE

<table>
<thead>
<tr>
<th>CRSE_NUM</th>
<th>CRSE_CLASS</th>
<th>CRSE_INST</th>
<th>CRSE_NAME</th>
<th>CRSE_DATE</th>
<th>CRSE_TIME</th>
<th>CRSE_DAY</th>
<th>CRSE_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>DB2004</td>
<td>200</td>
<td>SYSTEMS ADMIN</td>
<td>1990-11-24</td>
<td>10.00.00</td>
<td>TU</td>
<td>A</td>
</tr>
<tr>
<td>27</td>
<td>DB2001</td>
<td>--</td>
<td>INTRO TO DB2</td>
<td>1990-11-30</td>
<td>09.00.00</td>
<td>MO</td>
<td>A</td>
</tr>
<tr>
<td>28</td>
<td>DB2002</td>
<td>300</td>
<td>SQL PROGRAMMING</td>
<td>1990-11-30</td>
<td>09.00.00</td>
<td>MO</td>
<td>B</td>
</tr>
<tr>
<td>29</td>
<td>PHY001</td>
<td>500</td>
<td>RELATIVITY</td>
<td>1990-10-02</td>
<td>09.00.00</td>
<td>WD</td>
<td>A</td>
</tr>
<tr>
<td>30</td>
<td>DB2003</td>
<td>--</td>
<td>DATABASE DESIGN</td>
<td>1990-12-07</td>
<td>09.00.00</td>
<td>MO</td>
<td>B</td>
</tr>
<tr>
<td>31</td>
<td>DB2001</td>
<td>800</td>
<td>INTRO TO DB2</td>
<td>1990-12-07</td>
<td>09.00.00</td>
<td>MO</td>
<td>A</td>
</tr>
<tr>
<td>32</td>
<td>LIT001</td>
<td>600</td>
<td>ENTREPRENEUR</td>
<td>1990-12-09</td>
<td>10.00.00</td>
<td>WD</td>
<td>A</td>
</tr>
<tr>
<td>33</td>
<td>DB2001</td>
<td>100</td>
<td>INTRO TO DB2</td>
<td>1990-12-09</td>
<td>09.00.00</td>
<td>WD</td>
<td>B</td>
</tr>
</tbody>
</table>

#### Course View = VCRSE

<table>
<thead>
<tr>
<th>CRSE_NUM</th>
<th>CRSE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>SYSTEMS ADMIN</td>
</tr>
<tr>
<td>27</td>
<td>INTRO TO DB2</td>
</tr>
<tr>
<td>28</td>
<td>SQL PROGRAMMING</td>
</tr>
<tr>
<td>29</td>
<td>RELATIVITY</td>
</tr>
<tr>
<td>30</td>
<td>DATABASE DESIGN</td>
</tr>
<tr>
<td>31</td>
<td>INTRO TO DB2</td>
</tr>
<tr>
<td>32</td>
<td>ENTREPRENEUR</td>
</tr>
</tbody>
</table>

#### A logical view of that table

### Course list ‘View’
View Usage Rules

• Valid reasons to implement views:
  – Security - row and column level
  – Access - efficient access paths
  – Data Derivation - put the calculations in the view
  – Mask Complexity - hide complex SQL from users
  – Rename a Table
  – Column Renaming - table with better column names (easier to use than AS)

• Synchronize all views with base tables...

DO NOT USE ONE VIEW PER BASE TABLE!
Database Design for DB2
Craig S. Mullins
Mullins Consulting, Inc.
http://www.craigmullins.com

and that’s just about all I can cover in an hour . . .