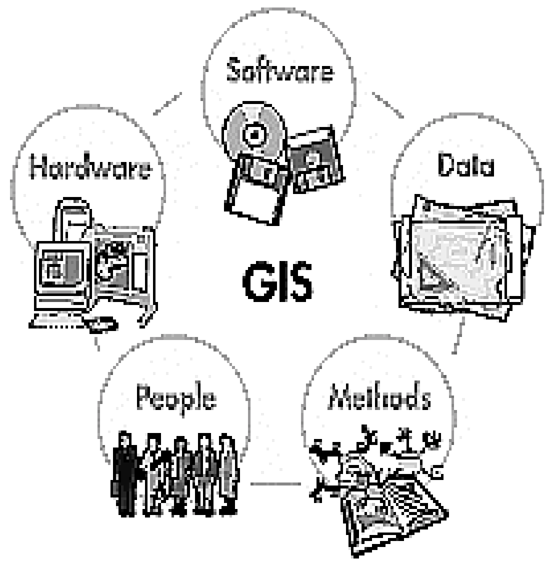
Relational Database Management Systems, Database Design, and GIS

presented by:
Tim Haithcoat
University of Missouri Columbia

With materials from:
Peter Veenstra
M.J. Harden Associates

Overview of GIS Database Design

- A geographic information system (GIS) is comprised of several elements, including
  - Hardware
  - Software
  - Users/People
  - Procedures/Methods
  - Data



- GIS Organizations…
  - Select hardware and software
  - Train their users
  - Develop procedures
  - The technology incorporated into business flow

PART 1

Relational Databases and GIS

Hybrid Model

- Comprised of two systems - one to handle the spatial elements, another to manage attribute data
- Most hybrid systems use a proprietary data model
- Separate storage systems complicate database maintenance, increase disk access and network traffic
- Requires diligence, attention to detail and special applications to maintain feature-attribute linking.
  - What happens when a user splits a line segment?
  - Where does the original attribute records go?
  - How do you maintain a historical record of line splitting?
  - How are other GIS layers affected by splitting a pipe?
- Example of a Hybrid Model? (ARC/INFO, ESRI ShapeFile)
  - Overview of GIS Database Design

Spatial Database Servers

- Continuous, non-tiled, spatial database for adding spatial data to a relational database management system (RDBMS).
- Database interface that couples spatial data to the RDBMS allowing for high-performance access to all the data in there, spatial and non-spatial.
- No more split system data management-single source editing. Requires special maintenance application to main topology, perform database edits, updates and maintenance (ArcFM)
- Utilize the inherent strengths of commercial RDMBS’s...

	Spatial Server (RDBMS)	Hybrid Model -or- Flat File
User Access	Roles, users, built-in security.	No inherent security.
Security	Stored in Proprietary Files not accessible from any other application than the RDBMS.	Disk files, easily recognizable, editable with external applications.
Data Integrity	Enforces referential integrity, data stamping, user access and rights, triggers, procedures, transactions (rollbacks, commits)	No internal enforced referencing (IDEDIT, RENODE).
Buffered Throughput	Designed for fast transfer of packets through network. Only access what you need.	Access everything within the spatial extent, accessing both spatial and attribute features each with their own data structure.
Multi-user	Multiple users can access data. Allows for row or table level locking. Optimistic and pessimistic updating. User roles determine editing rights.	Only one user can edit records. No built in locking or updating mechanisms. No built in security.
Open Data Structure	Relational database mechanism is well known. ORACLE Spatial Data Option is normalized tables, SDE uses blobs - but reveals a lot about the data structure.	ShapeFIles: One feature table, one index file and one dBase file - published - very difficult. ARC/INFO totally proprietary.
Robustness	Roll-back segments. Redo Logs files, Back and Recovery tools. Well established kernel.	Lose or corrupt the file and hope that you have some back-up.
Data Restructuring	Views can be created from tables and can be stored as objects within the database	One flat file is a flat file. Can create definitions within ArcView or reselect statements in ARC/INFO. Not predefined objects.

Part II

Relational Databases

A method for structuring data in the form of sets of records or tuples so that relations between different entities and attributes can be used for data access and transformation.

~ Burroughs, 1986

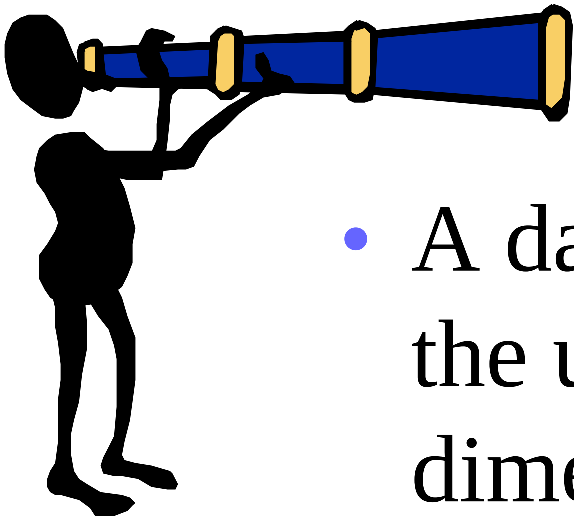
A database structure commonly used in GIS in which data is stored based on 2 dimensional tables where multiple relationships between data elements can be defined and established in an ad-hoc manner.

~ Croswell, 1991

Relational Database Management System - a database system made up of files with data elements in two-dimensional array (rows and columns). This database management system has the capability to recombine data elements to form different relations resulting in a great flexibility of data usage.

~ after Martin, 1976

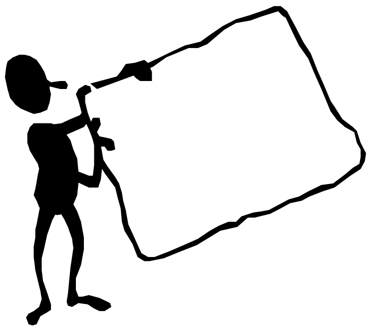
Relational Database Concepts



- A database that is perceived by the user as a collection of two-dimensional tables
- Are manipulated a set at a time, rather than a record at a time
- SQL is used to manipulate relational databases

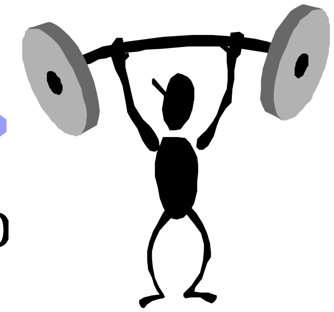
The Relational Database Concept

- Proposed by Dr. Codd in 1970
- The basis for the relational database management system (RDBMS)
- The relational model contains the following components:



  - Collection of objects or relations
  - Set of operations to act on the relations
  - Data integrity for accuracy and consistency

RDBMS: Advantages (1 of 2)



- Rigorous design methodology (normalizatio set theory)
- All other database structures can be reduced to a set of relational tables
  - Mainframe databases use Network and Hierarchical methods to store and retrieve data.
  - Access to the data is hard-coded
  - It is very difficult to extract data from this type of database without some pre-defined access path.
  - Extremely fast retrieval times for multi-user, transactional environment.
- Ease the use compared to other database systems

RDBMS: Advantages (2 of 2)

- Modifiable - new tables and rows can be added easily
- The relational join mechanism
  - Based on algebraic set theory - a set is a group of common elements where each member has some unique aspect or attribute
  - very flexible and powerful
- Fast Processing
  - Faster processors, multi-threaded operating and parallel servers
  - Indexes, fast networks and clustered disk arrays
  - 57,000 simultaneous users (Oracle/IBM)

RDBMS: Disadvantages

- Expensive solutions that require thorough planning
- Easy to create badly designed and inefficient database designs if there is not any proper data analysis prior to implementation

Architecture

DBMS

- A software package for stage, manipulate and retrieval of data from a database
- Serves many users simultaneously

Kernel

- Core software, controls query processing, access paths to data, user access management, storage management, indexing, transaction processing and read/update information

Interactive Query Tool

- Access, edit, and update of one or more linked data tables using screen based forms.

Query Language Interface

- Wrapped around the kernel, allows the ad hoc query against the database

Architecture

Utilities

- Import/export/backup tuning tools
- Parameterization/report writers

Processes (memory)

- Database Writer, Archiving, User Manager, Server Manager, Redo Log files

Database

- Physical storage of the data objects within data files
- contains the system catalogues (data dictionary)
- A collection of one or more data files stored in a structural manner
- Relationships which exist between different sets of data

Part III

Design

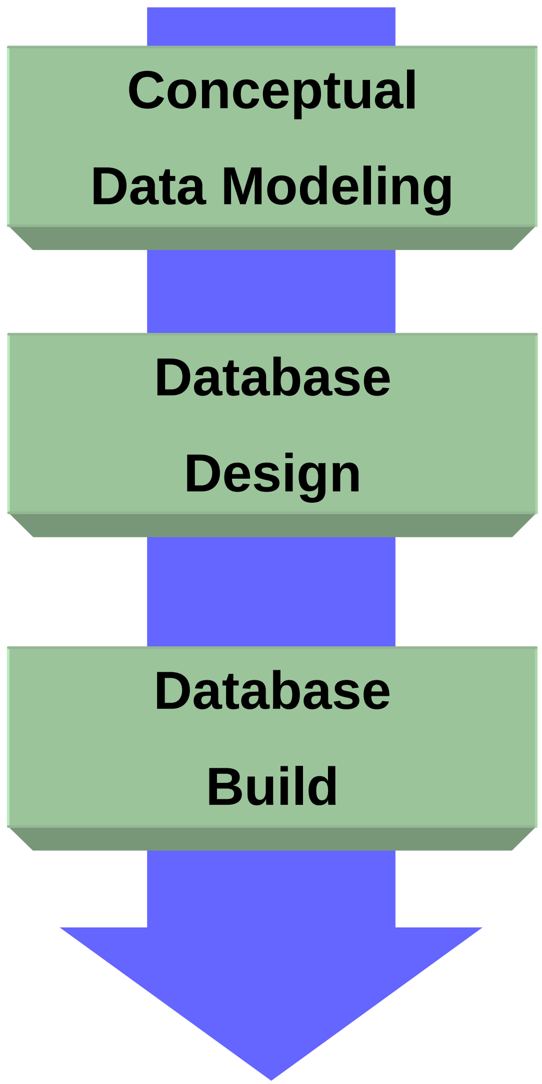
Business Information Requirements

Strategy

Analysis

Design

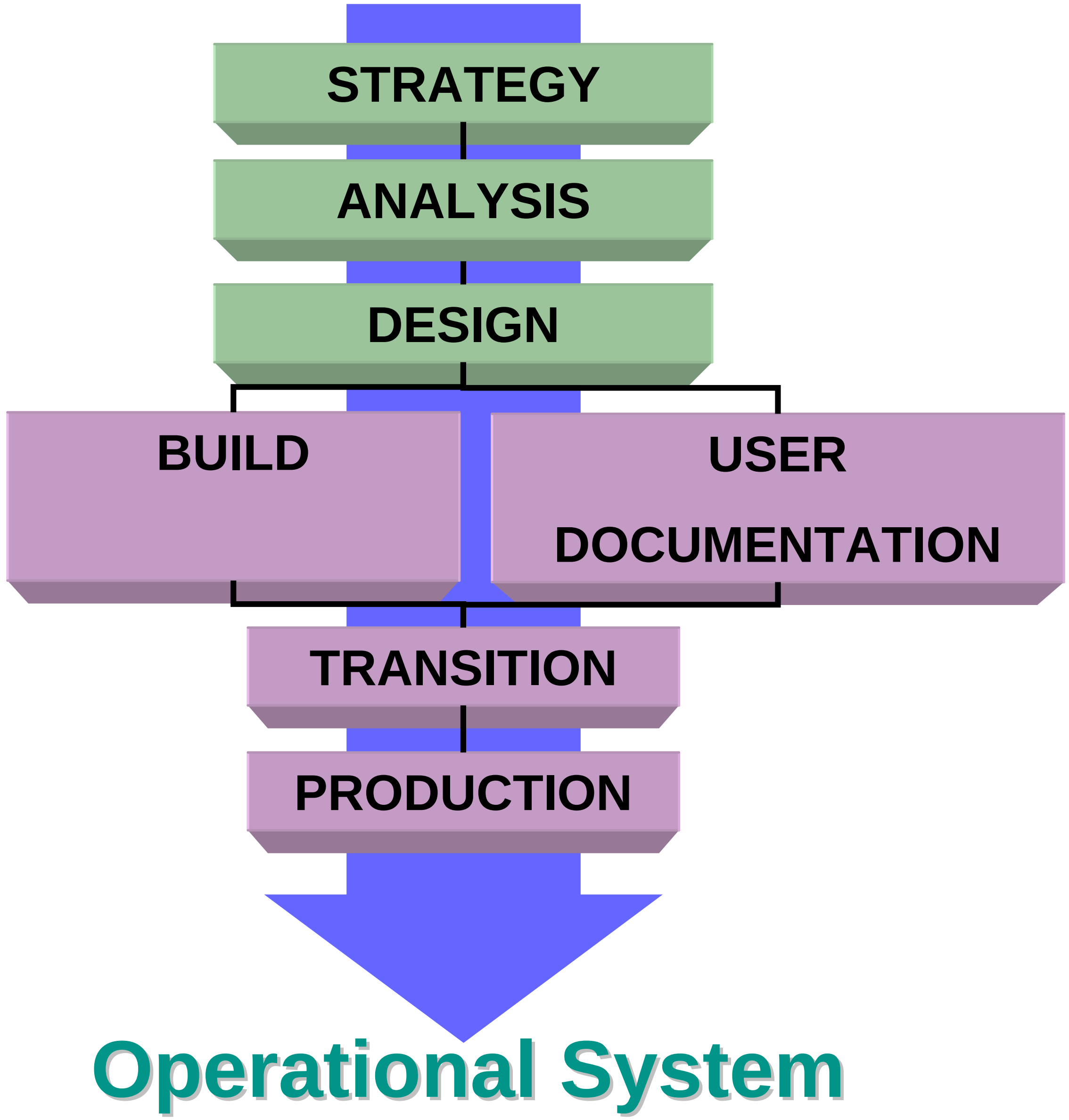
Build



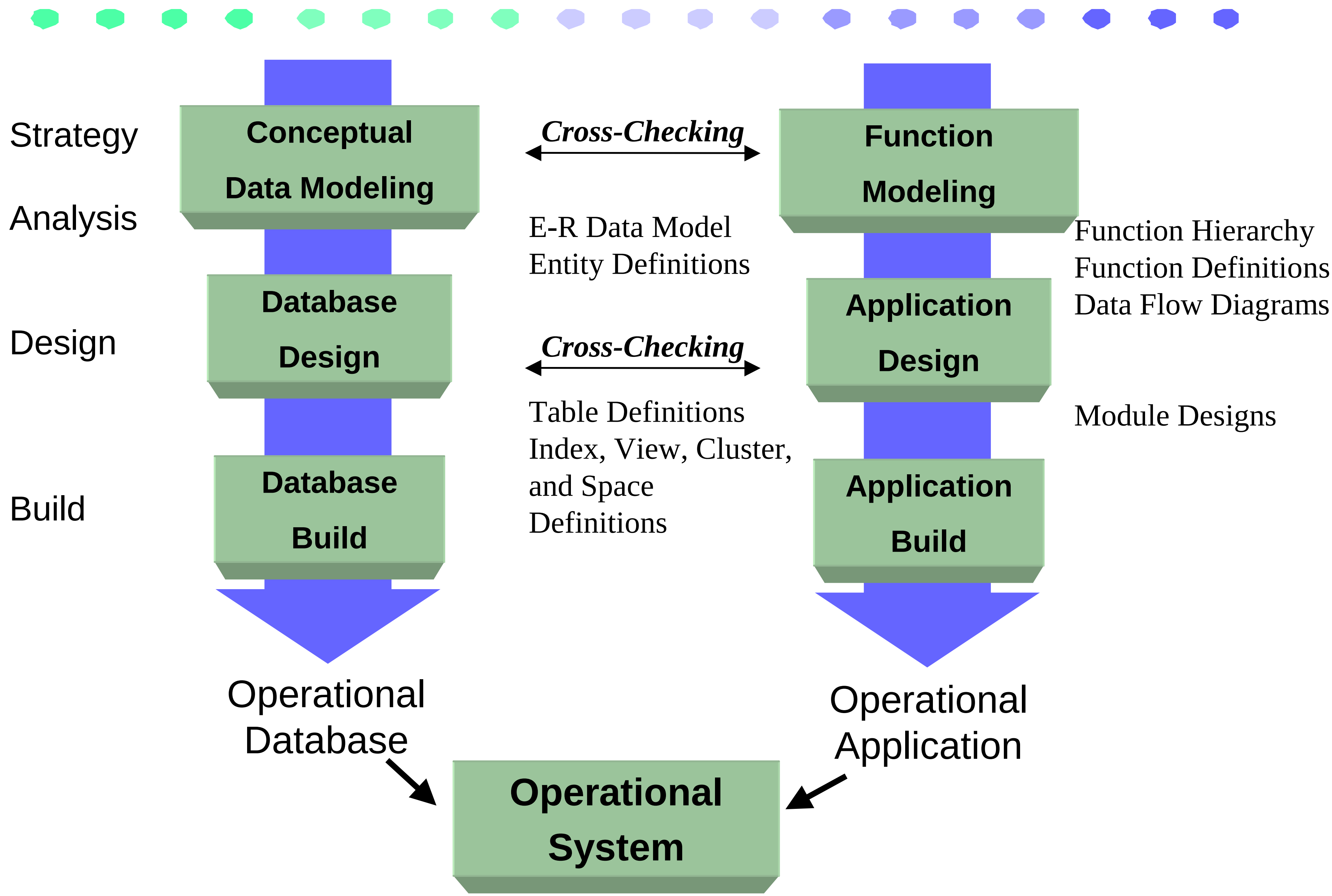
Entity-Relationship Data Model
Entity Definitions

Table Definitions
Index, View, Cluster, and
Space Definitions

Business Requirements



Operational Database



Good Database Design Prevents...

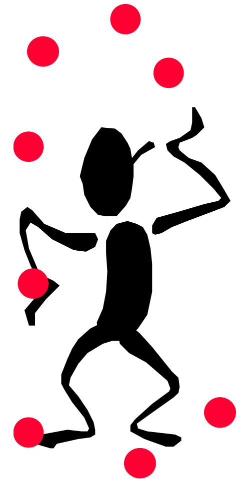
- Unnecessary or forgotten data
- Inflexibility for database re-sizing or modification
- Poor data element specification
- Poor database integration between the parts of the database
- Unsupported applications
- Major database update costs

Success of GIS Database Design

- Depends on the ability of the system to provide quality information
- Depends on the quality of usability of the data that resides on the system
- Ad-hoc approach versus systematic approach
- Begin with the “end in mind”

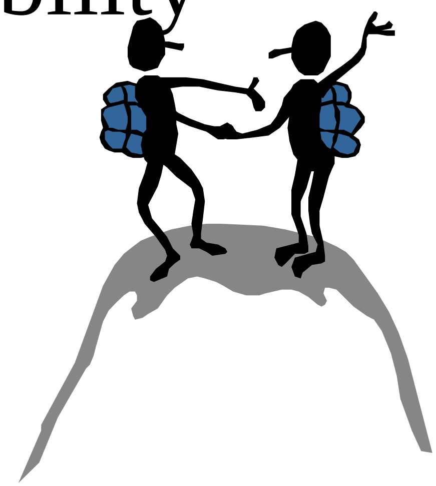
Influences on Database Design

- Applications
- Data format and size
- Data maintenance and update
- Hardware/software
- Number and sophistication of users
- Schedule and budget of the project
- Management approach



End Goal of Database Design:

- Is to maintain…
  - Data consistency/integrity
  - Reduce data redundancy
  - Increase system performance
  - Maintain maximum user flexibility
  - Create a useable system

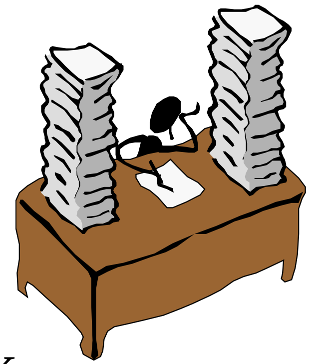


Part IV

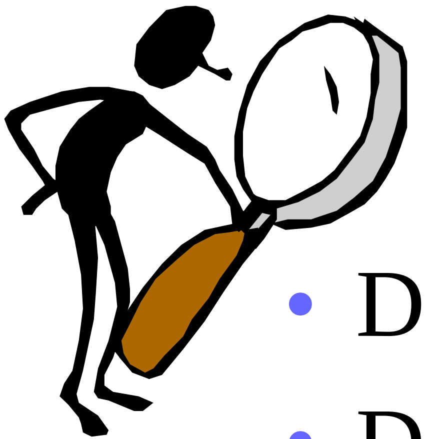
Project Life Cycle

Functional & Organizational Requirements Analysis (User Needs)

- Identify potential GIS users within the organization
- Identify initial participants in the GIS development effort
- Application identification and description
- Applications are the driving force of the GIS
  - Accomplish some task
    - *Examples:* create a map, generate a report, tack, manipulate the database, perform analysis
- Needs to be comprehensive and through in definition of applications
- Has a big impact on database design and development
- Provides initial user documentation



Principal Elements: Design Process



- Design cartographic layers
- Design business tables
  - Features attributes, legacy data, look up values…
- Implement cartographic layer tiling

Specify Cartographic Layers (1 of 2)

- Based on user needs choose the relevant cartographic layers
- Features on, under or above the earth's surface are abstracted to points, lines, or polygons
- Complex data structures are based on these data primitives
  - Networks, TINS, Regions…
- Scale determines representation of phenomena
  - A stream is a line as 1:250,000 scale
  - A stream is a polygon at 1:24,000
- Each thematic layer is stored in its own file
  - Proprietary file format

Specify Cartographic Layers (2 of 2)

- Challenges lie in co-incident line management
- Data maintenance by different departments
- Organize layers according to similar themes
  - Choose appropriate spatial feature type for representing the theme (polygon, line, grid, image)
  - Requires knowledge of the problem domain

- Develop feature symbology/annotation
- Describe features within they layers
- Relate features to previously identified applications
- Develop standards for map/tabular precision and accuracy

Cartographic Layer Partitioning

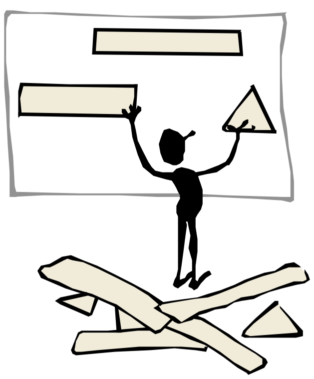
- Organize or tile data layers into meaningful sub-groups
- Increase user access times -same amount of data
- Boundaries must remain stable - difficult to change
- Choose physical units rather than political ones
- Apply abstract grids like USGS Quad Index, PLSS Schema

Metadata Documentation

- Data name
- Data create date
- Creator's name
- Data owner
- Data sensitivity
- Which groups can see the data?
- Source of data
- Construction process of the data
- Record scale constraints, perspective, magnification, filters or definitions
- Record Geodesy information (Datum, Projection)
- Record Accuracy and Errors Standards
- Federal Geographic Data Committee (FGDC) _ National Spatial Data Infrastructure (NSDI

Design Business Tables

- Conceptual and Data Modeling
- Store all the descriptive attribute (tabular) information for the project
- The manner which business data is organized is very important
- Anticipate uses as well as update procedures

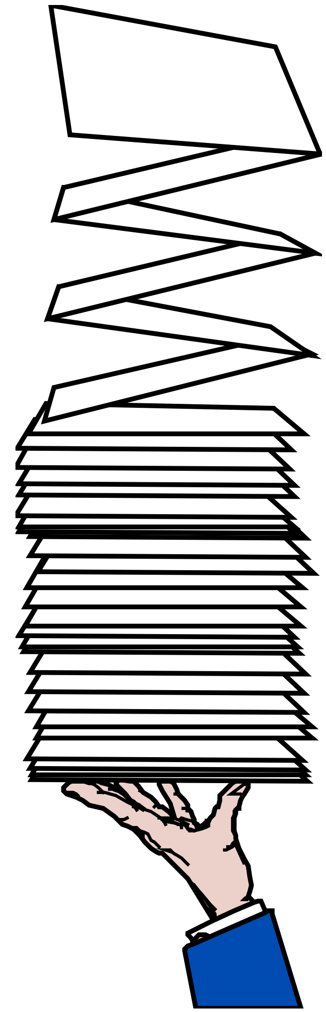


Normalize Business Tables

- Separates data into meaningful groupings making it easy to maintain, update, modify and protect
- Provides rules for organizing data into tables that relate to each other by common keys
- Requires thorough knowledge of the data in its relationships
- Normalized tables can be related to form new relationships
- Assign each feature (point, line, or polygon) a unique code
- Allows a link to the tabular business data stored in a RDBMS

Functional Analysis

- Data Flow Diagramming
- Model Applications
  - Triggers
  - Data flows
  - Results
- Model System Outputs
  - Reports
  - Calculations



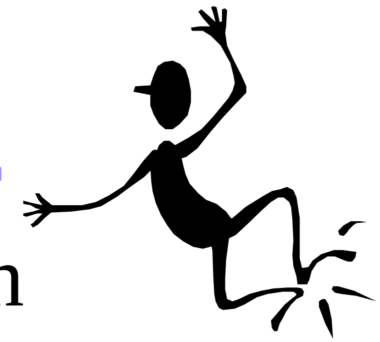
Document the Database Design

- Very important - users have confidence in the data
- Comprehensive data dictionary
  - Describe all the items, codes, constraints, value ranges and structures of each layer
  - Provides input to automatic validation and quality control operations/routines
- Diagrams the database design discussion notes about context and content of each layer
- Description of data sources for features and attributes for each layer
- Implementation, conversion, processing procedures and accuracy tolerances

Pilot Study

- Exhibit full range of complexity
- Most plans do not survive contact with the enemy
- Implementation and design plans require modification when tested
- Test physical database design performance and completeness
- Peer review applications and complete layers
- Document pilot study results - lessons learned there can be extended

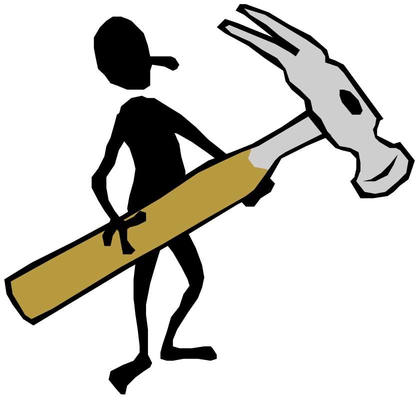
Data Conversion



- Get each layer into digital format (both graphical and tabular information)
- Apply data conversion quality control
- Objective is to catch errors and lapses in quality up-front
- Clear definition of accuracy tolerances for each database layer
- Develop metadata on the GIS database
  - Metadata is descriptive information about the data
  - What is the data source? How accurate is it?

Use and Maintain the System

- Manipulate, update and expand the database
- Administer the database
- Provide programming services
- Track new technology and take advantage of it when appropriate
- Add new users to the system
- Develop an adequate training capability



Use and Maintain the System

Two items that are never fully investigated nor outlined or defined:

Mapping Application

- Allow user to determine exactly how the final map product should be displayed (in excruciating detail!).
- Pay attention to how each theme should be displayed.
  - Does the database support this?
  - What about labeling?
  - What about symbolization?

Maintenance Application

- User signs out required features. Audit trail begins.
- User should be allowed to lock, edit, update and add features. Should lock both the spatial and attribute records associated with the feature. Should provide an audit trail.
- Should automatically update metadata information. Should be a transactional system. Should encapsulate and enforce business rules. Should validate all changes to the database.
- User signs new or updated features back into the database.

Part V

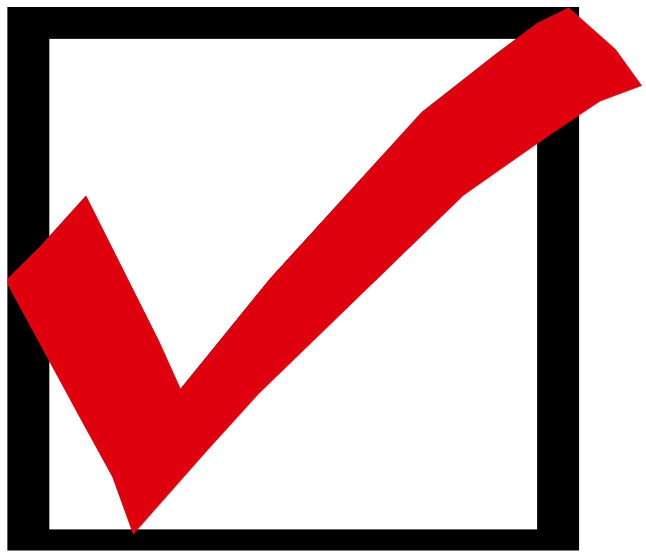
Design Process

Conceptual Modeling (Analysis)

- Top-down approach that transforms business information requirements into an operational database.
- Information requirements are tightly coupled with business function requirements
- Objective is to define and model the things of significance about which the business needs to know or hold information, and the relationships between them.
- Ignores hardware and software.
- High level look at the database.

Data Modeling (Logical)

- Objective: map the information requirements reflected in an Entity-Relationship Model into a Relational Database Design.
- Software specific.
- Hardware independent.



Physical Modeling (Build)

- Objective is to create physical relational database tables to implement the database design.
- Hardware and software dependent
- File structure and memory requirements.
- Network dependent
- The structured Query Language (SQL) is used to create and manipulate relational databases.

Part VI

Relational Database Concepts and Structure

Tables, Relationships, Set Theory

- The power of a relational database comes from its ability to relate significant data together
- Database tables are related to each through columns of data sharing identical data (called keys).
- Each table is based on mathematical set theory (each element in the set must be unique).
- Relational databases are usually manipulated a set at a time rather than a record at a time.
- The Structured Query Language (SWL) is used to manipulate relational databases.

Tables

- Describes or models phenomena that are of significance to the business
- Consist of rows of data (Tuples) that are uniquely identified from other other rows of data. Each row represents or corresponds to an instance of the phenomena being modeled.
- Made of columns or attributes that describe the phenomena being modeled.
- Are often the implementation of an entity
- Are the logical and perceived data structure, not the physical data structure, in a relational system.
- Are abstractions of reality.

Relational Database Terminology

- Each table is composed of rows and columns

S_CUSTOMER Table (Relation)

Row (Tuple) →

ID	NAME	PHONE	SALES_ REP ID
201	Unisports	55-2066101	12
202	Simms Athletics	81-20101	14
203	Delhi Sports	91-10351	14
204	Womansport	1-206-104-0103	11

↑ Column (Attribute)

- You can manipulate data in the rows by executing Structured Query Language (SQL) commands.

Relational Database Terminology

- Each row of data in a table is uniquely identified by a primary key (PK).
- You can logically relate information from multiple tables using foreign keys (FK).

ID	NAME	PHONE	SALES_ REP_ID
201	Unisports	55-2066101	12
202	Simms Atheletics	81-20101	14
203	Delhi Sports	91-10351	14
204	Womansport	1-206-104-0103	11

ID	LAST_ NAME	FIRST_ NAME
10	Havel	Marta
11	Magee	Colin
12	Giljum	Henry
14	Nguyen	Mai

Primary Key Foreign Key Primary Key

Relations Between Tables

S_EMP Table

ID	LAST_NAME	DEPT_ID
--	---------------------------	--------------
1	Velansquez	50
2	Ngau	
3	Nagayama	
4	Quick-To-See	
5	Ropeburn	
6	Urguhart	
7	Menchu	
8	Biri	
9	Catchpole	
10	Havel	32
11	Magee	33
12	Giljum	34
12	Sedeghi14 Nguyen	35
15	Dumas	41
16	Maduro	

S_DEPT Table

ID	NAME	REGION_ID
--	--------------------	--------------
30	**Finance**	**1**
31	**Sales**	**1**
32	**Sales**	**2**
43	**Operations**	
50	**Administration**	

S_REGION Table

ID	NAME
--	--------------------
1	**North American**
2	**South America**
3	**Africa/Middle East**
4	**Asia**
5	**Europe**

Table Instance Charts

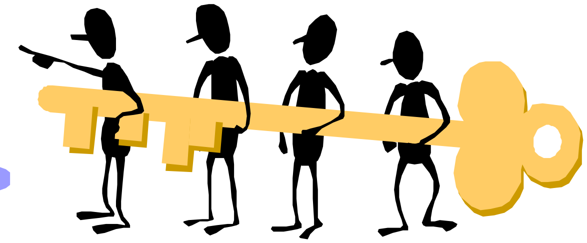
Table Name:

Column Name								
Key Type								
Nulls/ Unique								
Sample Data								

Primary Key

- A primary key (PK) column or set of columns that uniquely identifies each row in a table
- Each table must have a primary key and a primary key must be unique
- A PK consisting of multiple columns is called a Composite Primary Key
- No part of the PK can be null
- Tips for identifying PKs
  - Must be a unique value
  - Value in the PK for each tuple or row should never change
  - PK is best auto-generated - should not contain business info

Foreign Keys



- A foreign key (FK) is a column or combination of columns in one table that refers to a primary key in the same or another table
- A FK must match an existing primary key value (or else be null)
- If a FK is part of a primary key, that FK cannot be null
- In order for a relation to be established between two tables, they both must contain a common data element
  - (e.g. a field that has been defined the same in both tables)

Data Integrity

- Refers to the accuracy and consistency of the data
- Data integrity constraints should be enforced by DBMS or the application software
- The rules of the business can also determine the correct state for a database
- Such rules are called User-Defined Data Integrity Constraints

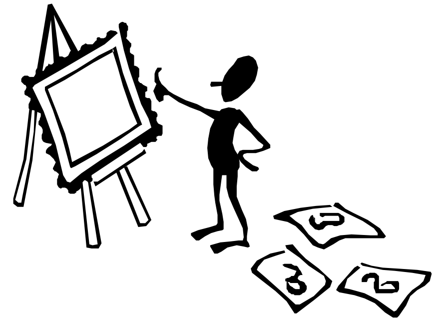
Data Integrity Constraints

- **Entity**
  - No part of the primary key can be NULL and the value must be unique
  - A NULL is the absence of a value
- **Referential**
  - A set of validation rules applied to an entity or table such as uniqueness constraints, domain validation of columns or correspondence of foreign keys to the primary key of the related table
    - Unique - each record in table must have a PK with a unique value
    - Domain - range of possible values for an individual column or attribute
    - Referential Integrity - each value for a FK within a table must correspond to the value of one record’s PK in the Foreign table or be a NULL column
    - Values in column must match the defined data type
- **User Defined**
  - Values must comply with the business rules

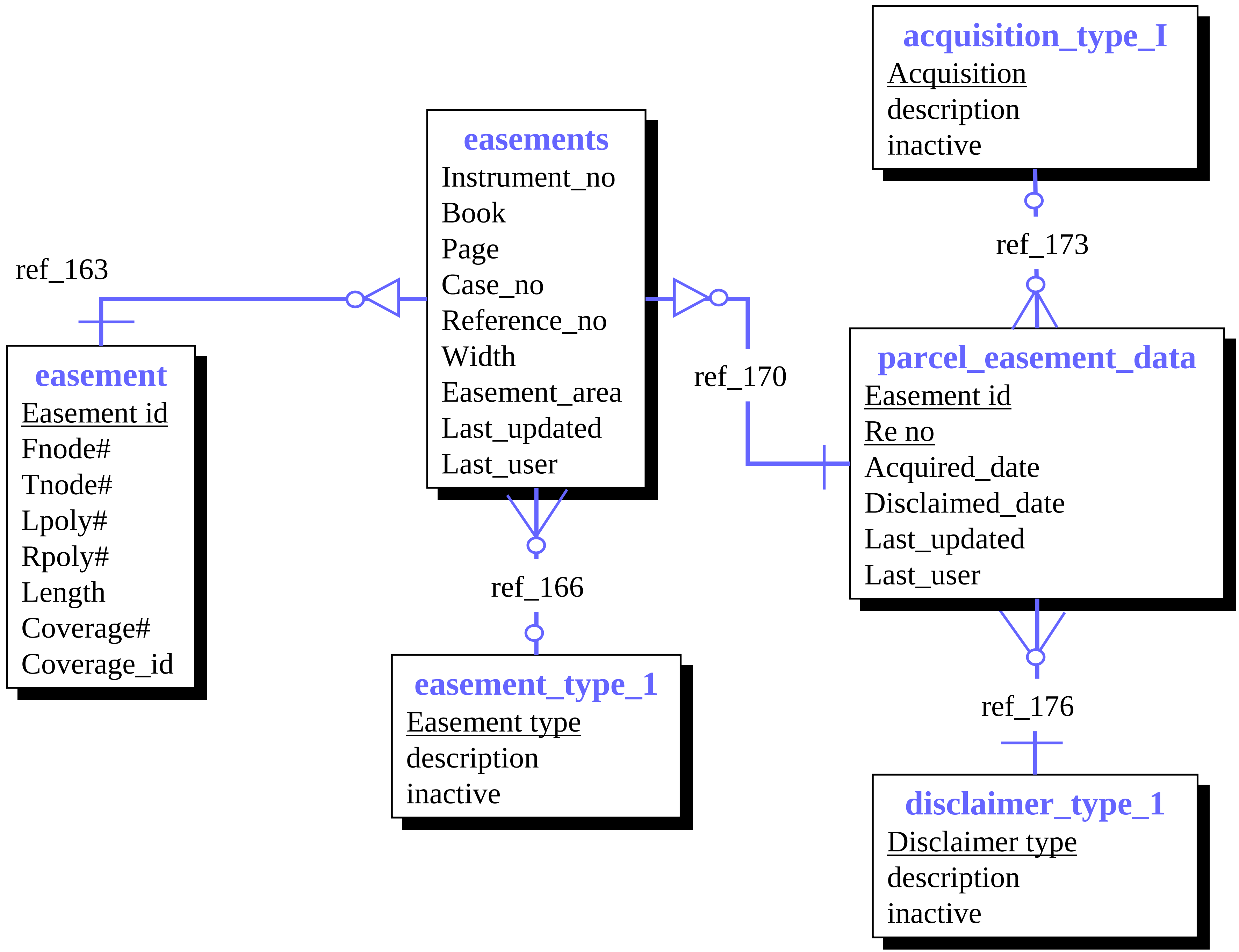
Part VII

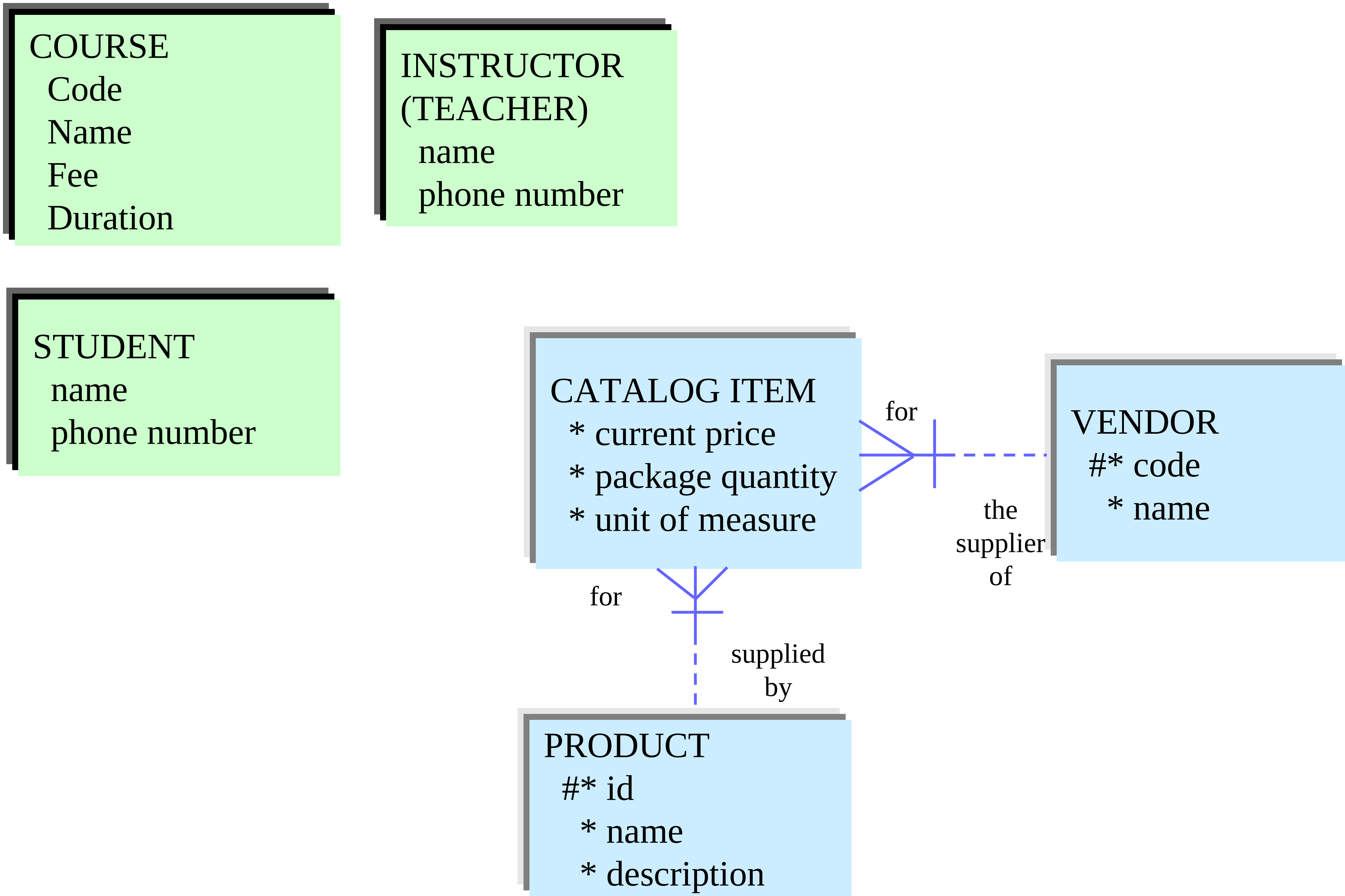
Conceptual Modeling

Conceptual Modeling



- The art of distilling a business requirements statement into a conceptual diagram
- Business requirements are determined from user needs assessments
- Is high level abstraction and occurs before database design and implementation
- Is independent of hardware or software
- **Goal:** develop an entity-relationship model representing the business requirements


acquisition_type_I
Acquisition
description
inactive
ref_173
easements
Instrument_no
Book
Page
Case_no
Reference_no
Width
Easement_area
Last_updated
Last_user
ref_163
easement
Easement id
Fnode#
Tnode#
Lpoly#
Rpoly#
Length
Coverage#
Coverage_id
ref_170
parcel_easement_data
Easement id
Re no
Acquired_date
Disclaimed_date
Last_updated
Last_user
ref_166
easement_type_1
Easement type
description
inactive
ref_176
disclaimer_type_1
Disclaimer type
description
inactive


COURSE
Code
Name
Fee
Duration
INSTRUCTOR
(TEACHER)
name
phone number
STUDENT
name
phone number
CATALOG ITEM
* current price
* package quantity
* unit of measure
for
VENDOR
#* code
* name
the
supplier
of
for
supplied
by
PRODUCT
#* id
* name
* description

Diagramming Conventions

- A line between two entities
- Lower case relationship names
- Optionality

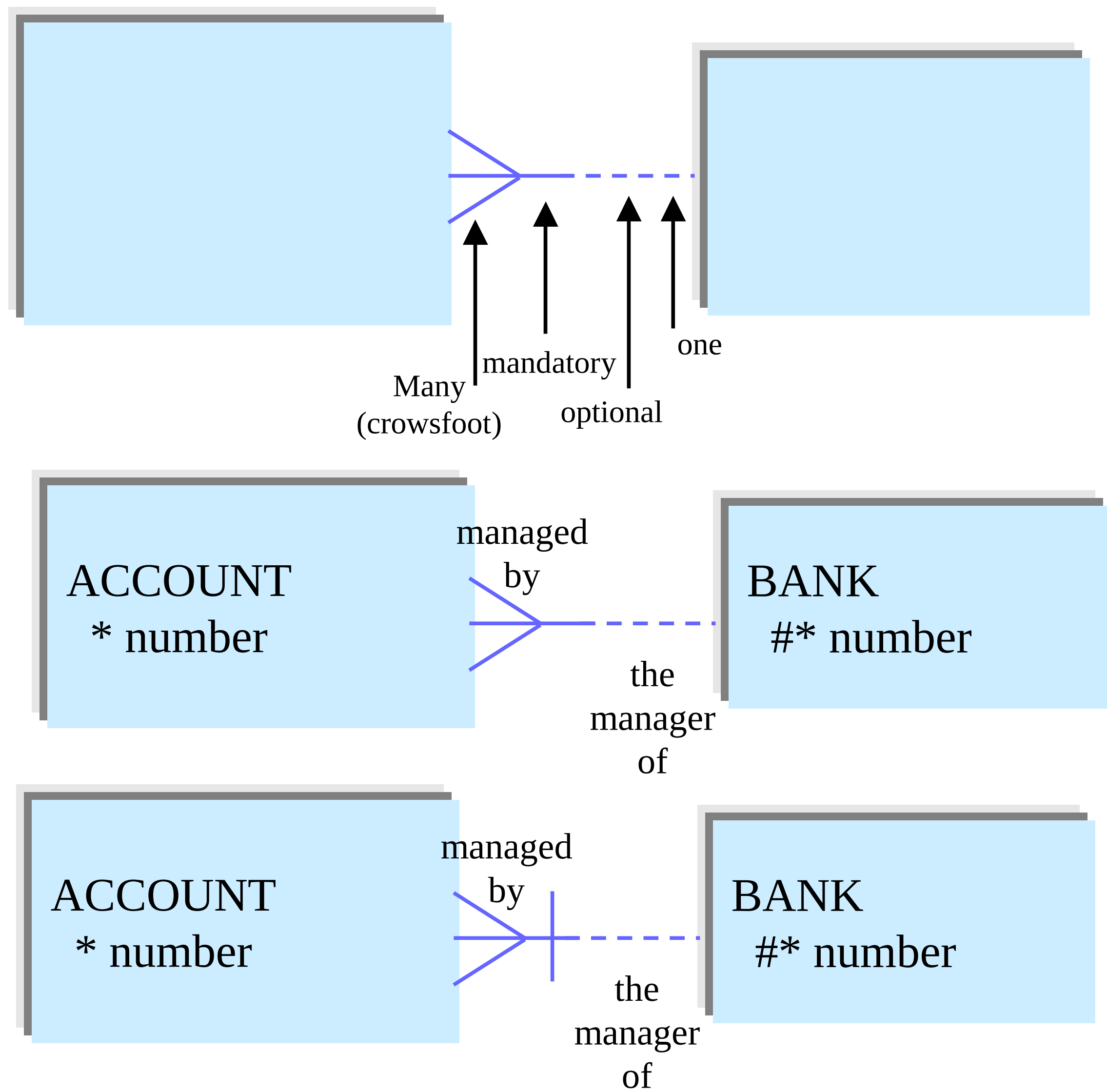
- - - - - Optional (may be)

——— Mandatory (must be)

- Degree

One or more

——— One & only one

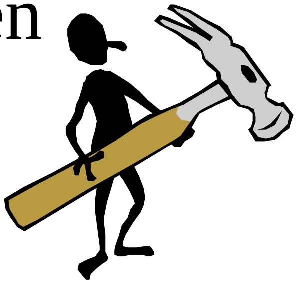


Entity-Relationship (E-R) Model

- Should accurately model the organization's information needs and support the functions of the business.
  - Entities, Relationships, Attributes
- Is an effective means for collecting and documenting an organization's information requirements
- Robust Syntax
- User Communication
- Ease of Development
- Definition of Scope
- Integration of Multiple Applications
- Can be mapped to a hierarchical, network, or relational database
- Can be used as the template for an Enterprise Object Model

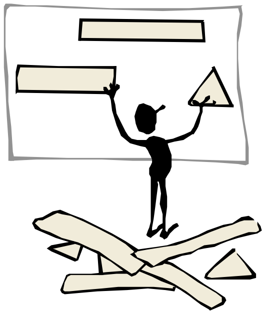
How-to Create an E-R Diagram

- Identify and model entities
- Analyze and model the relationships between the entities
- Analyze and model the attributes that describe the entities
- Identify unique identifiers for each entity
- Develop a complete entity-relationships model from the statement of information requirements
- Normalize the entities and relationships between them
- Advanced modeling



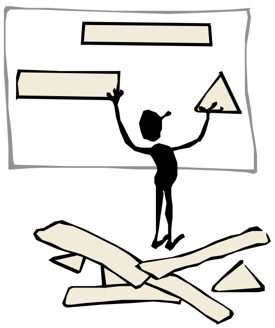
Defining Entities (1 of 2)

- A thing of significance about which information needs to be known or held.
- an object of interest to the business, a class or category of thing, a named thing
- Each entity must have multiple occurrences or instances
- Each entity instance has specific values for the entities attributes
- A each instance of must be uniquely identifiable from other instances of the same entity



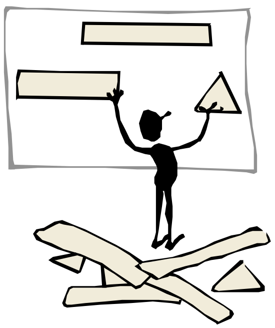
Defining Entities (2 of 2)

- An attribute or set of attributes that uniquely identify an entity is called a Unique Identifier (UID).
- Attributes describe entities and are the specific pieces of information which need to be known.
- An entity must have attributes that need to be known from the business' viewpoint or it is not an entity within the scope of the business's requirements.



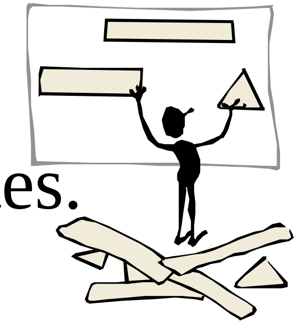
Entity Diagramming Conventions

- Soft box with any dimensions
- Singular unique entity name
- Optional synonym name in brackets
- Attribute names in lower case
- Mandatory Attributes prefaced with a *
- UID Attributes prefaced with a #



Steps to Identifying Entities

- Examine the business requirements definition or statement
- Examine the nouns? Are they items of significance?
- Name each entity.
- Is there information of interest that the business needs to hold?
- Is each instance of the entity uniquely identifiable?
- Which attribute or attributes could serve as it's UID?
- Write a description of the entity.
- Diagram each entity and a few of it's attributes.



Entity Instance Chart

Entity Name:

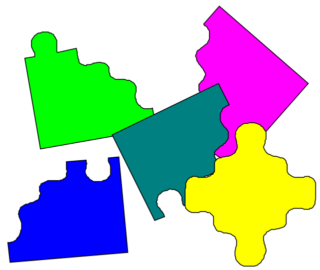
Attribute Name							
Tags							
Sample Data							

Attributes (1 of 2)

- Always clarify a data attribute with a descriptor.
- Are information about an entity that needs to be known or held.
- Describe an entity by qualifying, identifying, quantifying or expressing the state of the entity.
- Represent a description or detail, not an instance.
- Name should be clear to the user no codified for the developer.
- Name should not include the entities name.
- Attribute names should be specific.

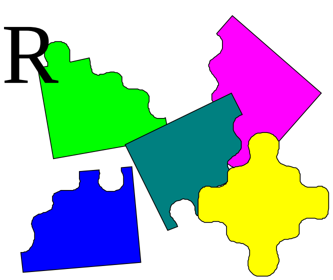
Attributes (2 of 2)

- An attribute should only be assigned to a single entity.
- Always break attributes down to their lowest meaningful components.
- The level of decomposition depends on the business requirements.
- Verify that each attribute has a single value for each entity instance.
- A multi-valued attribute or a repeating group is not a valid attribute.
- A repeated attribute indicates a missing entity.



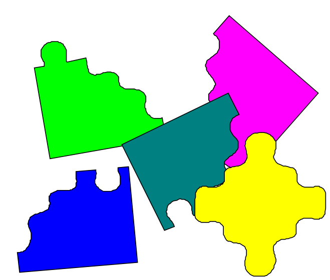
Derived Attributes

- Verify that an attribute is not derived or calculated from the existing cvalue of other attributes
- Derived attributes are redundant
- Redundant data leads to inconsistent data values
- Address the option of storing derived data in the Database Design Phase
- Do not include derived attributes in an E-R model.



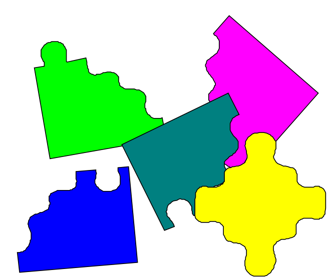
Defining Attributes

- Identify attributes by examining interview notes and by asking the user questions
- Attributes may appear in interview notes as:
  - Descriptive words or phrases
  - Nouns
  - Prepositional phrases (e.g. salary amount for employee)
  - Possessive nouns and pronouns (e.g. employee’s name)
- Questions to ask the user…
  - What info do you need to know or hold about ENTITY X?
  - What info would you like displayed or printed about ENTITY X?
- Examine documentation on existing manual procedures or automated systems to discover additional attributes or omissions.



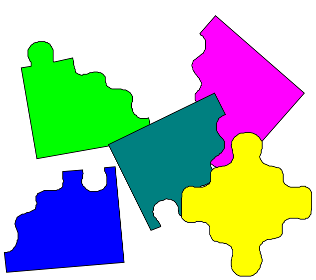
Assign Unique Identifiers (UID)

- A U ID is any combination of attributes and/or relationships that serve to uniquely identify an occurrence of an entity. Each entity occurrence must be uniquely identifiable
- All components of an entity must be mandatory (*)
- Tag each UID attribute with an (#*)



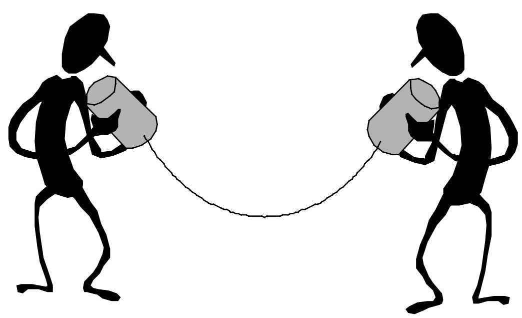
Validating Attributes

- Are all attributes decomposed?
- Are all attributes single valued?
- Is each attribute dependent on the entities entire UID?
- Is each attribute dependent on only one part of the entities UID?



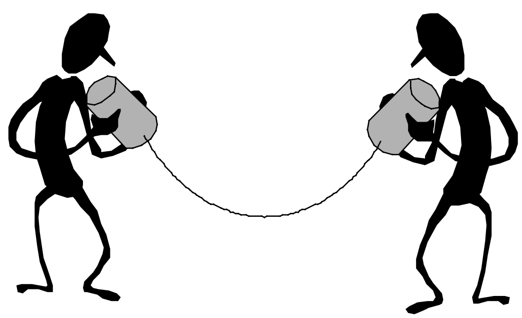
Defining Relationships

- Is a two directional significant association between two entities or between an entity and itself
- All relationships should represent the information requirements and the rules of the business.
- Can be read in one direction or the other

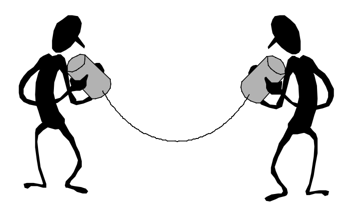


Steps to Create a Relationship

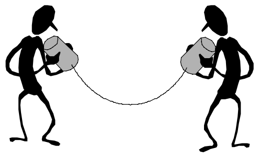
- Identify the first entity.
- Identify the optionality (must be *or* may be).
- Identify the relationship.
- Identify the cardinality (one or more *or* one and only one).
- Identify the relate entity.



Relationship Matrix Form



Types of Relationships



Many to One

(M to 1 or M:1)

- Has a degree (cardinality) of one or more in one direction & a degree of one and only one in the other direction.
- Are very common.
- M:1 relationships that are mandatory in both directions are very rare.

Many to Many

(M to M or M:M)

- Has a degree of one or more in both directions.
- Are very common.
- Are usually optional in both directions, although usually a M:M relationship is optional in one direction.

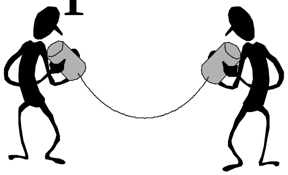
One to One

(1 to 1 or 1:1)

- Has a degree of one and only one in both directions.
- Are rare.
- 1:1 relationships that are mandatory in both directions is very rare.
- Entities which seem to have a 1:1 relationship may really be the same entity.

Steps to Analyze & Model Relationships

- Determine the existence of a relationship
- Does a significant relationship exist between ENTITY A and ENTITY B.
- Use a relationship matrix to systematically examine each pair of entities.
- Name each direction of the relationship
- Ask a relationships name - how are ENTITY A and ENTITY B related
- Log the relationship names in the relationship matrix.

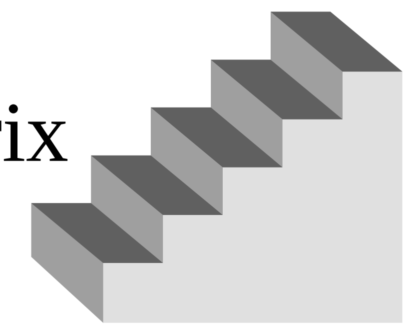


Steps to Analyze & Model Relationships

- Use a list of relationship name pairs to assist in naming relationships:
  - Based on - the basis for
  - Bought from - the supplier of
  - Description of - for
  - Operated by - the operator of
  - Represented by - the representation of
  - Responsible for - the responsibility of
- Determine the optionality of each direction of the relationship
  - Draw the relationship lines with names

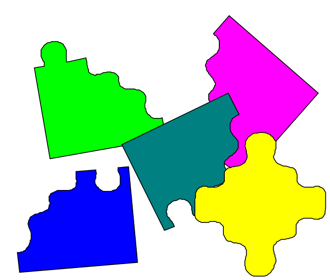
Steps to Analyze & Model Relationships

- Determine the cardinality of each direction of the relationship
  - Add the relationship degrees to the E-R diagram
- Read the relationship out loud to validate it
  - First read a relationship in one direction, and then read the relationship in the other direction
- Use a relationship matrix as an aid for the initial collection of information about the relationships between a set of entities.
- Map the contents of a relationship matrix to an E-R diagram.



Redefining Attributes

- An entity can be uniquely identified through a relationship
- Use a UID bar to indicate that a relationship is part of the entity’s UID.



Advanced Conceptual Data Modeling

- A relational database concept, but it’s principles apply to Conceptual Data Modeling.
- A normalized entity-relationship data model automatically translates into a normalized relational database design
- A step-by-step process that produces either entity or table definitions that have:
  - No repeating groups
  - The same kind of values assigned to attributes or columns
  - A distinct name
  - Distinct and uniquely identifiable rows
- Third normal for is the generally accepted gal for a database design that eliminates redundancy
- Higher normal forms a theoretical and not often used
- We go through the Normal Forms to avoid data integrity issues.

First Normal Form:

All attributes must be single valued.

- Validate that each attribute has a single value for each occurrence of the entity. No attribute should have repeating values.
- If an attribute has multiple values, create an additional entity and relate it to the original entity with a M:1 relationship.

Second Normal Form:

An attribute must be dependent on its entities entire unique identifier.

- Validate that each attribute is dependent upon it's entities entire UID. Each specific instance of UID must determine a single instance of each attribute.
- Validate that an attribute is not dependent upon only par of it's entities UID.
  - If an attribute is not dependent on its entities entire UID, it is misplaced and must be removed.

Third Normal Form:

All attributes in an entity must depend on the whole primary key, the entire primary key and nothing but the primary key (so help you Codd!)

Part VIII

Data Modeling

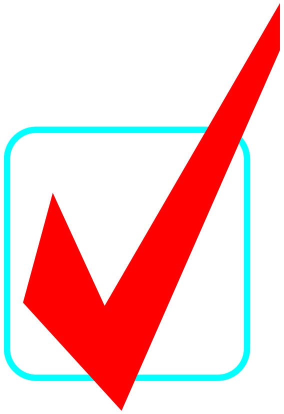
(Database Design)

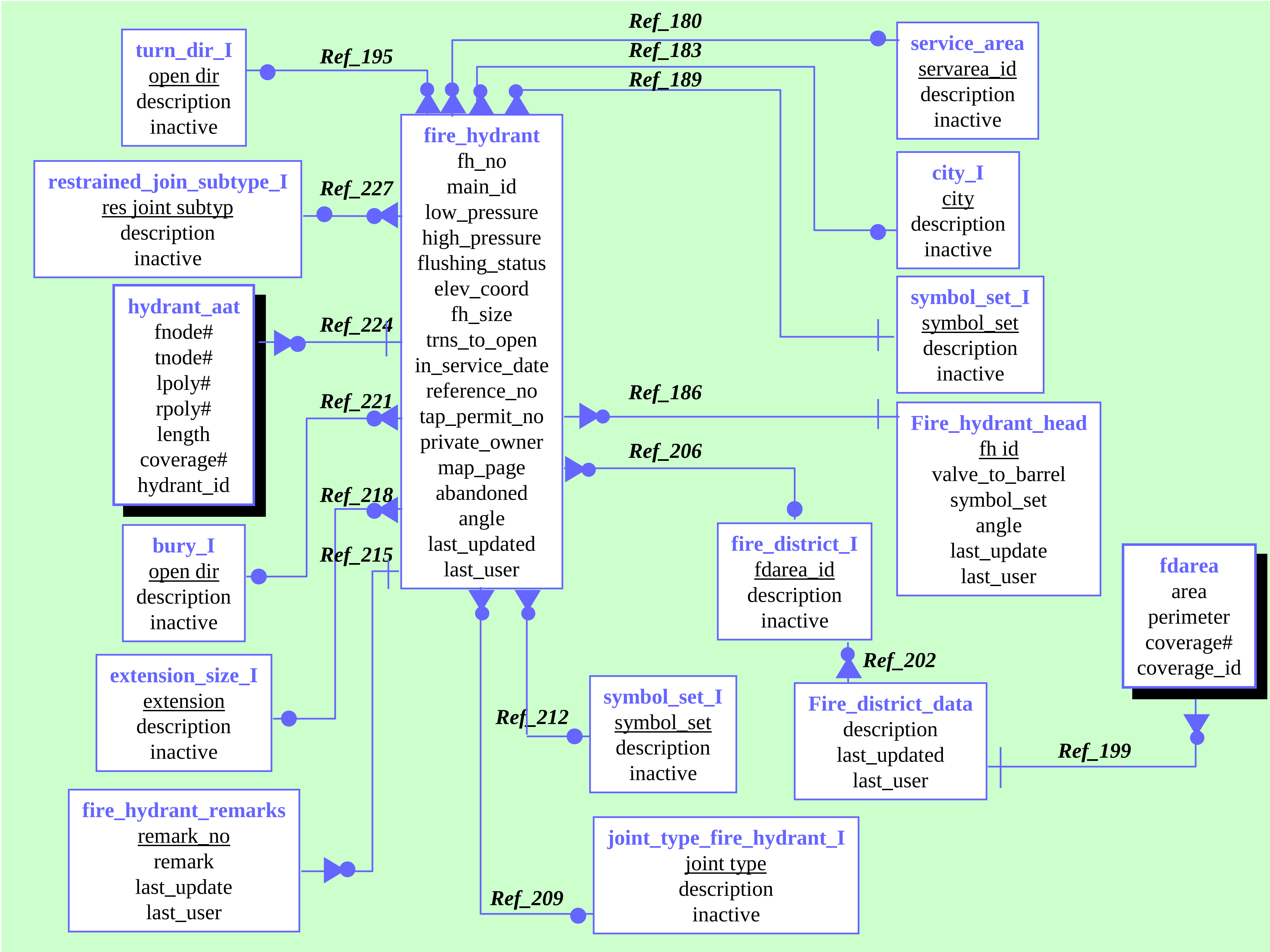
Initial Database Design

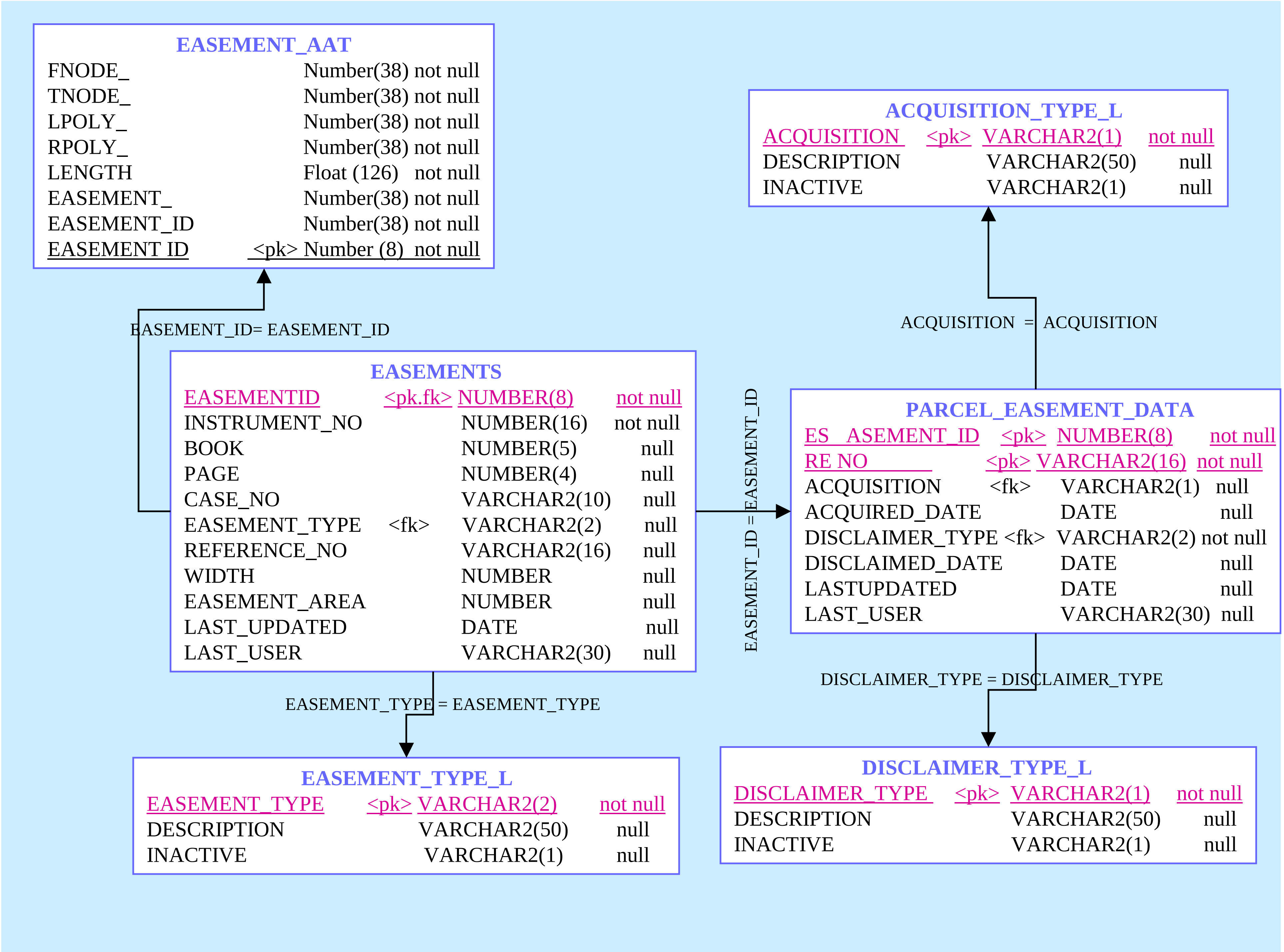
- Objective: to map the information requirements reflected in an entity relationship model into a relational database design
- Define the initial design to produce a complete database design

Database Design Process

- Document each relational table from an entity in the E-R model to a Table Instance Chart
- Map the simple entities to tables
- Map attributes to columns
- Indicate required, unique and NULL attributes
- Map unique identifiers to primary keys
- Map relationships to foreign keys
- Document sample data to each column
- Re-normalize as required




Ref_180
Ref_183
Ref_189
Ref_195
turn_dir_I
open dir
description
inactive
service_area
servarea_id
description
inactive
fire_hydrant
fh_no
main_id
low_pressure
high_pressure
flushing_status
elev_coord
fh_size
trns_to_open
in_service_date
reference_no
tap_permit_no
private_owner
map_page
abandoned
angle
last_updated
last_user
restrained_join_subtype_I
res joint subtyp
description
inactive
Ref_227
city_I
city
description
inactive
hydrant_aat
fnode#
tnode#
lpoly#
rpoly#
length
coverage#
hydrant_id
Ref_224
symbol_set_I
symbol_set
description
inactive
Ref_221
Ref_186
Fire_hydrant_head
fh id
valve_to_barrel
symbol_set
angle
last_update
last_user
Ref_206
Ref_218
fire_district_I
fdarea_id
description
inactive
Ref_215
bury_I
open dir
description
inactive
fdarea
area
perimeter
coverage#
coverage_id
Ref_202
extension_size_I
extension
description
inactive
Ref_212
symbol_set_I
symbol_set
description
inactive
Fire_district_data
description
last_updated
last_user
Ref_199
fire_hydrant_remarks
remark_no
remark
last_update
last_user
joint_type_fire_hydrant_I
joint type
description
inactive
Ref_209



EASEMENT_AAT
FNODE_ Number(38) not null
TNODE_ Number(38) not null
LPOLY_ Number(38) not null
RPOLY_ Number(38) not null
LENGTH Float (126) not null
EASEMENT_ Number(38) not null
EASEMENT_ID Number(38) not null
EASEMENT ID <pk> Number (8) not null
ACQUISITION_TYPE_L
ACQUISITION <pk> VARCHAR2(1) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
EASEMENT_ID= EASEMENT_ID
ACQUISITION = ACQUISITION
EASEMENTS
EASEMENTID <pk,fk> NUMBER(8) not null
INSTRUMENT_NO NUMBER(16) not null
BOOK NUMBER(5) null
PAGE NUMBER(4) null
CASE_NO VARCHAR2(10) null
EASEMENT_TYPE <fk> VARCHAR2(2) null
REFERENCE_NO VARCHAR2(16) null
WIDTH NUMBER null
EASEMENT_AREA NUMBER null
LAST_UPDATED DATE null
LAST_USER VARCHAR2(30) null
EASEMENT_ID = EASEMENT_ID
PARCEL_EASEMENT_DATA
ES ASEMENT_ID <pk> NUMBER(8) not null
RE NO <pk> VARCHAR2(16) not null
ACQUISITION <fk> VARCHAR2(1) null
ACQUIRED_DATE DATE null
DISCLAIMER_TYPE <fk> VARCHAR2(2) not null
DISCLAIMED_DATE DATE null
LASTUPDATED DATE null
LAST_USER VARCHAR2(30) null
DISCLAIMER_TYPE = DISCLAIMER_TYPE
EASEMENT_TYPE = EASEMENT_TYPE
EASEMENT_TYPE_L
EASEMENT_TYPE <pk> VARCHAR2(2) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
DISCLAIMER_TYPE_L
DISCLAIMER_TYPE <pk> VARCHAR2(1) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null

V_EASEMENT

PARCEL_EASEMENT_DATA.LAST_UPDATE	DATE
PARCEL_EASEMENT_DATA.LAST_USER	VARCHAR2(30)
EASEMENTS.INSTRUMENT_NO	NUMBER(16)
EASEMENTS.PAGE	NUMBER(4)
EASEMENTS.CASE_NO	VARCHAR2(10)
EASEMENTS.REFERENCE_NO	VARCHAR2(16)
EASEMETNS.WIDTH	NUMBER
EASEMENTS.EASEMENT_AREA	NUMBER
EASEMENTS.LAST_UPDATED	DATE
EASEMENTS.LAST_USER	VARCHAR2(30)
EASEMENT_TYPE_L.DESCRIPTION	VARCHAR(50)
PARCEL_EASEMENT_DATA.ACQUIRED_DATE	DATE
EASEMENTS.EASEMENT_ID	NUMBER(8)
EASEMENTS.BOOK	NUMBER(5)
EASEMENTS.EASEMENT_TYPE	VARCHAR2(2)
EASEMENTS_TYPE_L.EASEMENT_TYPE	VARCHAR2(2)

EASEMENT_TYPE_L

PARCEL_EASEMENT_DATA

EASEMENTS

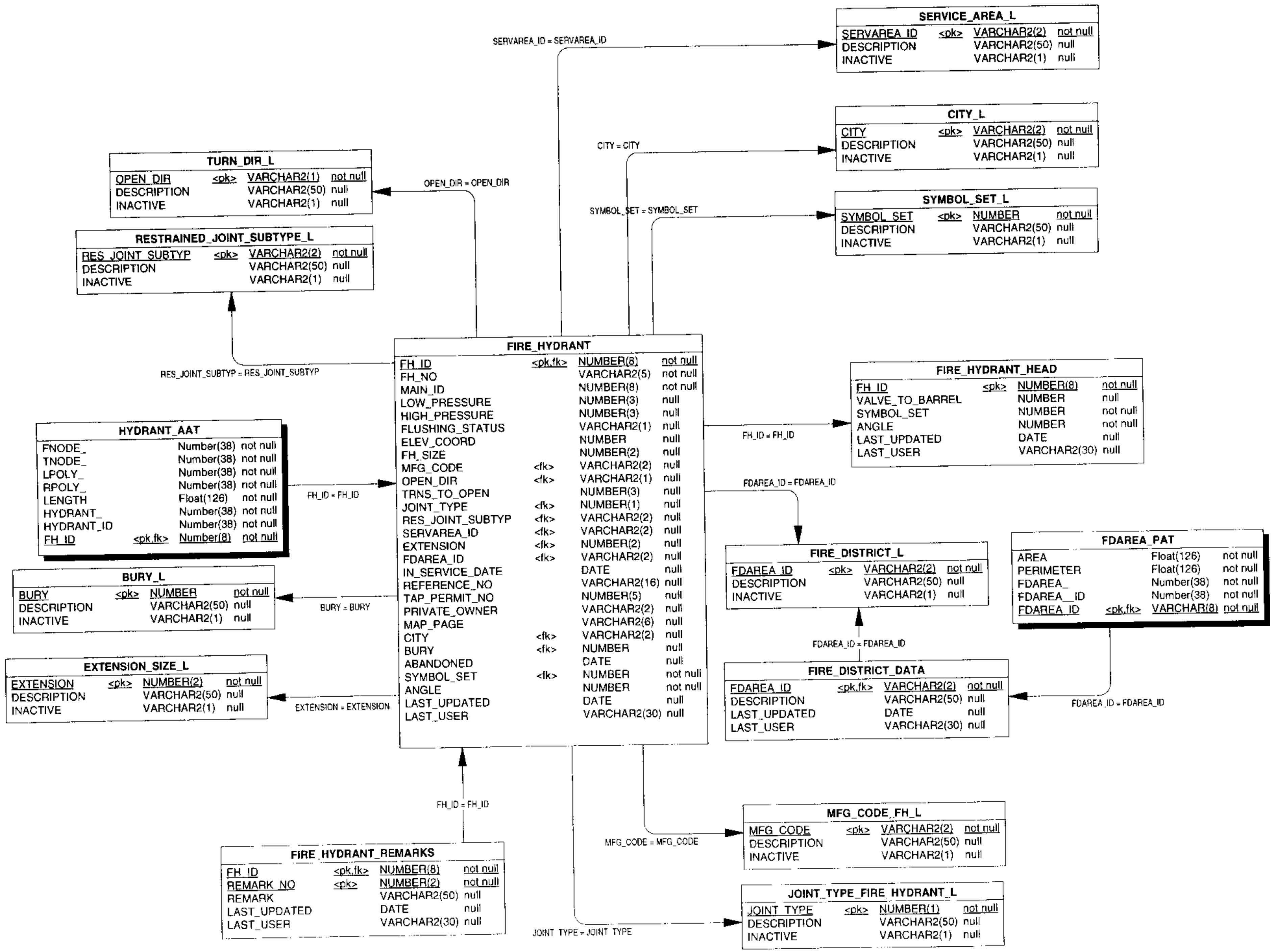
Select PARCEL_EASEMENT_DATA. LAST_UPDATED, PARCEL_EASEMENT_ DATA.LAST_USER, EASEMENTS. INSTRUMENT_NO,EASEMENTS.PAGE, EASEMENTS.CASE_NO,EASEMENTS.REF ERENCE_NO, EASEMENTS.WITH, EASEMENTS.EASEMENT_AREA, EASEMENTS.LAST_UPDATED, EASEMENTS. LAST_USER, EASEMENT_ TYPE_L.DESCRIPTION, PARCEL_ EASEMENT_DATA.ACQUIRED_DATE, EASEMENTS.EASEMENT_ID, EASEMENTS. BOOK, EASEMENTS. EASEMENT_TYPE, EASEMENT_TYPE_L. EASEMENT_TYPE from EASEMENT_ TYPE_L, PARCEL_EASEMENT_DATA, EASEMENTS where PARCEL_EASEMENT_DATA. EASEMENT_ID = EASEMENTS. EASEMENT_ID and EASEMENT_TYPE_L. EASEMENT_TYPE = EASEMENTS. EASEMENT_TYPE group by EASEMENT_ TYPE_L.EASEMENT_TYPE order by EASEMENTS.EASEMENT.ID

```
SQL>   CREATE  TABLE EMPLOEE
2     (DEPTNO    NUMBER(2)     NOT NULL PIMRARY KEY
3     DNAME      CHAR (20)     NOT NULL
4     LOC        CHAR(15)      NOT NULL) ;
```

```
SQL> CREATE TABLE EMPLOYEE
2     (EMPNO       NUMBER(5)      NOT NULL PRIMARY KEY
3     FNAME        CHAR (15)      NOT NULL
4     LNMAE        CHAR(15)       NOT NULL
5     JOB          CHAR(9)        NOT NULL
6     HIREDATE     DATE
7     SAL          NUMBER (7,2)
8     COMM         NUMBER (7,2)   REFERNCES EMPLOYEE (EMPNO)
9     MGR          CQR(4)         NOT NULL REFERNECES (DEPTNO) );
10    DEPTNO       NUMBER(2)
```

EMPLOYEE

EMP_ NUM	EMP_ NAME	DEPT_ NUM	DEPT_NAME	MGR_ NUM	MGR_ NAME	PROJECT_ NUM	PROJECT_ NAME	START_ DATE	BILLED_ HOURS
PK									
7902	SMITH	10	SALES	7988	JONES	15	FEASIBILITY	10-SEP-94	100
						35	TESTING	20-SEP-94	100
						45	HANDOVER	20-OCT-94	150
7988	JONES	20	MARKETING	7699	WALKER	15	FEASIBILITY	05-SEP-94	200
						25	ANALYSIS	15-SEP-94	250
						45	HANDOVER	20-OCT-94	200
7562	SMITH	10	SALES	7099	PHILLIPS	25	ANALYSIS	20-MAY-94	150



SERVICE_AREA_L
SERVAREA_ID <pk> VARCHAR2(2) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
CITY_L
CITY <pk> VARCHAR2(2) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
SYMBOL_SET_L
SYMBOL_SET <pk> NUMBER not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
TURN_DIR_L
OPEN_DIR <pk> VARCHAR2(1) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
RESTRAINED_JOINT_SUBTYPE_L
RES_JOINT_SUBTYP <pk> VARCHAR2(2) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
SERVAREA_ID = SERVAREA_ID
CITY = CITY
SYMBOL_SET = SYMBOL_SET
OPEN_DIR = OPEN_DIR
RES_JOINT_SUBTYP = RES_JOINT_SUBTYP
FIRE_HYDRANT
FH_ID <pk,fk> NUMBER(8) not null
FH_NO VARCHAR2(5) not null
MAIN_ID NUMBER(8) not null
LOW_PRESSURE NUMBER(3) null
HIGH_PRESSURE NUMBER(3) null
FLUSHING_STATUS VARCHAR2(1) null
ELEV_COORD NUMBER null
FH_SIZE NUMBER(2) null
MFG_CODE <fk> VARCHAR2(2) null
OPEN_DIR <fk> VARCHAR2(1) null
TRNS_TO_OPEN NUMBER(3) null
JOINT_TYPE <fk> NUMBER(1) null
RES_JOINT_SUBTYP <fk> VARCHAR2(2) null
SERVAREA_ID <fk> VARCHAR2(2) null
EXTENSION <fk> NUMBER(2) null
FDAREA_ID <fk> VARCHAR2(2) null
IN_SERVICE_DATE DATE null
REFERENCE_NO VARCHAR2(16) null
TAP_PERMIT_NO NUMBER(5) null
PRIVATE_OWNER VARCHAR2(2) null
MAP_PAGE VARCHAR2(6) null
CITY <fk> VARCHAR2(2) null
BURY <fk> NUMBER null
ABANDONED DATE null
SYMBOL_SET <fk> NUMBER not null
ANGLE NUMBER not null
LAST_UPDATED DATE null
LAST_USER VARCHAR2(30) null
FIRE_HYDRANT_HEAD
FH_ID <pk> NUMBER(8) not null
VALVE_TO_BARREL NUMBER null
SYMBOL_SET NUMBER not null
ANGLE NUMBER not null
LAST_UPDATED DATE null
LAST_USER VARCHAR2(30) null
FH_ID = FH_ID
HYDRANT_AAT
FNODE_ Number(38) not null
TNODE_ Number(38) not null
LPOLY_ Number(38) not null
RPOLY_ Number(38) not null
LENGTH Float(126) not null
HYDRANT_ Number(38) not null
HYDRANT_ID Number(38) not null
FH_ID <pk,fk> Number(8) not null
FH_ID = FH_ID
FDAREA_ID = FDAREA_ID
FIRE_DISTRICT_L
FDAREA_ID <pk> VARCHAR2(2) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
FDAREA_PAT
AREA Float(126) not null
PERIMETER Float(126) not null
FDAREA_ Number(38) not null
FDAREA__ID Number(38) not null
FDAREA_ID <pk,fk> VARCHAR(8) not null
BURY_L
BURY <pk> NUMBER not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
BURY = BURY
FDAREA_ID = FDAREA_ID
FIRE_DISTRICT_DATA
FDAREA_ID <pk,fk> VARCHAR2(2) not null
DESCRIPTION VARCHAR2(50) null
LAST_UPDATED DATE null
LAST_USER VARCHAR2(30) null
FDAREA_ID = FDAREA_ID
EXTENSION_SIZE_L
EXTENSION <pk> NUMBER(2) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
EXTENSION = EXTENSION
FH_ID = FH_ID
MFG_CODE_FH_L
MFG_CODE <pk> VARCHAR2(2) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
MFG_CODE = MFG_CODE
FIRE_HYDRANT_REMARKS
FH_ID <pk,fk> NUMBER(8) not null
REMARK_NO <pk> NUMBER(2) not null
REMARK VARCHAR2(50) null
LAST_UPDATED DATE null
LAST_USER VARCHAR2(30) null
JOINT_TYPE_FIRE_HYDRANT_L
JOINT_TYPE <pk> NUMBER(1) not null
DESCRIPTION VARCHAR2(50) null
INACTIVE VARCHAR2(1) null
JOINT TYPE = JOINT TYPE

Part VIII

Physical Design

(BUILD or IMPLEMENTATION)

- **Objective:** to create physical relational database tables to implement the database design. Structured query language (SQL) is used to create & manipulate relational databases.

Plan Physical Storage Usage

- For each table & index, estimate the amount of disk space required.
- Decide the placement of tables and indexes on logically separate tablespaces.
- Decide placement of tablespaces on physically separate disks.
- Define storage allocation procedures based upon the expected patterns of data update and growth.

Define Referential Integrity Constraints

- CASCADE DELETED, RESTRICTED UPDATES, NULLIFY
- Triggers - denotes processing carried out under certain conditions, i.e., may be actioned off before or after a row insertion

Design Indexes

- Used to speed the retrieval of data from RDBMS by reducing the amount of searching that the RDBMS must do to locate an individual record

Design Views

- Means of accessing a subset of database as if it were a table, the view may be:
  - Restricted to named columns, change column names, derive new columns, give access to a combination of related tables

Performance Issues

- Evaluate table de-normalization

Part IX

Physical and Organizational Issues in Database Design

Issues in Database Design

- What storage and media are used?
- How big is the database?
- How will the database grow over time?
- What are the required access speeds?
- Should data be partitioned by location or by layer?
- Should the data be centralized or localized - if so, on what server?
- Who is responsible for maintaining the data?
- Who performs QA/QC on updates & additions?

Part X

SQL

SQL is...

- A powerful, free form language for manipulating two dimensional tables of any size
- A command language for communication with the database server from a tool or application.
- Is divided into subsets for specific processing or interaction with the RDBMS.

Data Retrieval

- SELECT Statements
  - Used to retrieve data from the RDBMS in a ad-hoc manner.
  - The data returned is almost always presented to the user in table format (rows of data described by columns).

Basic SELECT Syntax

SELECT	Is a list of at least one column
DISTINCT	Suppresses duplicates
*	Selects all columns
Column	Selects the name column(s)
Alias	Gives the selected columns a different heading
FROM table	Specifies the table containing the columns
WHERE	Restricts the query to rows that meet a condition
Condition	Is composed of column names, expressions, constants and comparison operators
ORDERED BY	Specifies the order in which the retrieved rows are displayed
ASC	Orders rows in ascending order
DESC	Orders rows in descending order

Basic SELECT Syntax

- SELECT [DISTINCT} {*,column [alias],…}
- FROM table
- [WHERE condition(s)]
- ORDERED BY {column, expression} [ASC|DESC]];

- SELECT * FROM EASEMENTS;
- SELECT BOOK, PAGE, WIDTH * 12 AS "PROPOSED WIDTH" FROM EASEMENTS;
- SELECT BOOK || ' __ ' || PAGE FROM EASEMENTS;
- SELECT DISTINCT WIDTH FROM EASEMENTS;

- SELECT DISTINCT WIDTH FROM EAEMENTS
- ORDER BY LAST_UPDATED:

Basic SELECT Syntax (continued)

- SELECT BOOK, CASE_NO, WIDTH FROM EASEMETNS
- WERE LAST_USER = ‘VEENSTRA’
- ORDER BY LAST_UPDATED;

- Can use standards arithmetic operators (+.-,/,*)
- Can use standards comparison operations (<,>,<=,>=,=,<>)
- Can use single row functions
  - (LOWER, UPPER, INITCAP, CONCAT, SUBSTR, LENGTH, NVL) - Character Functions
  - (ROUND, TRUNC, MOD) - Number Functions
  - (MONTHS_BETWEEN, ADD-MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC) - Date Functions
  - (TO_CHAR, TO_DATE, TO_U NUMBER) - Conversion Functions
- Can use multiple row functions (GROUP BY - HAVING Clause)
  - AVG, COUNT, MIN, MAX, STDDEV, SUM, VARIANCE)

JOINS

- SELECT EASEMENTS.EASEMENT_ID, EASEMENT_TYPE_L.DESCRIPTION, EASEMENTS.LAST_UPDATE
- FROM EASEMENTS, EASEMENT_TYPE L
- WHERE EASEMENTS.EASEMENT_TYPE = EASEMENT_TYPE_L.EASEMENT_TYPE
- AND
- EASEMENT_TYPE_L.DESCRIPTION = ‘Confinement’;

Data Manipulation Language (DML)

- INSERT, UPDATE, DELETE
  - used to add data to existing tables within a database or to edit or remove existing data from within a database.

- INSERT INTO table [(column [, column…])]
- VALUES (value, [, value…]}];

- INSERT INTO table [(column [, column…])]
- Subquery;

- UPDATE table
- SET COLUMN = value[, column = value]
- [WHERE condition];

- DELETE [FROM} table
- [WHERE condition];

Data Definition Language (DDL)

- CREATE, ALTER, DROP, RENAME, TRUNCATE
  - a subset of SQL that is used to create, alter, drop or otherwise change definitions of tables, views and other database objects

- CREATE VIEW Easements
- AS SELECT…
- FROM…
- WHERE...

Data Control Language (DCL)

- GRANT, REVOKE -
  - Used by the database administrator to grant or revoke privileges to users of the RDBMS
    - Examples: connect to the database, read data, insert data, modify database objects, export or import data

Transaction Control

- COMMIT, ROLLBACK, SAVEPOINT -
  - Allows a user to cause the database to write the results of processing to the database
  - Allows the user to undo any changes made to the data within the database

Part XI

Object Oriented Databases

Objects

- Much like tables, objects abstract reality into functional or logical components - abstraction.
- Objects encapsulate certain behavior, functionality or data into discrete entities, often hiding those attributes from the outside world.
- Objects can have properties (nouns), methods (verbs) and events.
- Objects can belong to classes of objects and super-classes of objects.

Object-Relational Databases

- Object/relational databases organize information in the familiar relational tabular structures.
- Access the objects through the user of extenders, cartridges and DataBlades.
- By encapsulating methods with data structures, an ORDBMS server can execute complex analytical and data manipulation operations to search and transform multimedia and other complex objects.
  - Traditional fielded data, complex objects such as time-series and geo-spatial data and diverse binary media such as audio, video, images and applets

Object-Relational Databases (continued)

- The most important new object/relational features are user-defined types (UDTs), user-defined functions (UDFs), and the infrastructures -- indexing/access methods & optimizer enhancements -- that support them.
- The Object-Relational paradigm is quite strong.
  - Advanced Web applications are notable beneficiaries of the ORDBMS's ability to integrate management of media, traditional fielded data, and templates for dynamic page generation
  - To date, ORDBMSs have had their greatest success in managing media objects and complex data such as geospatial and financial time series data.
- Spatial Data Cartridge (Oracle), SDE - ArcFM, ARC/INFO 8.0

Part XII

GIS and RDBMS Design Experience

Experiences

- Document, document, document.
- Look at the current output, reports and existing databases.

- Work as part of a team at all stages of the projects (two heads are better than one).
- Spend the time up front on analysis.
- Continually review information with end users.
- Do not skip the conceptual data modeling.
- Be consistent, thorough and patient.
- There is no right way to do something. Create a balance between integrity and performance.