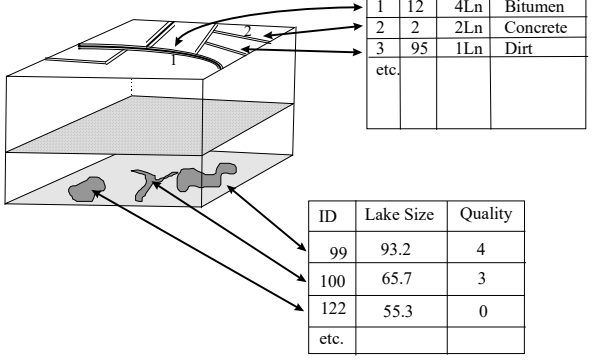
COA 690/790 GIS in Marine Science

Lecture 10
Attribute data and tables

April 11, 2019

Standard GIS Data Model

- Linked spatial and attribute data



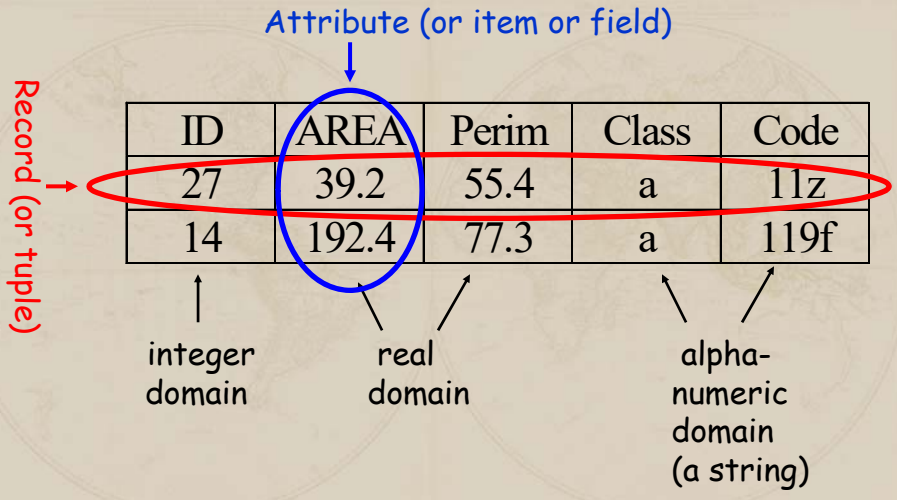
For each layer there is typically a one-to-one relationship between geographic features (point, line, or polygon) and records in a table

Name	FIPS	Pop90	Area	PopDn
Whatcom	53073	128	2170	59
Skagit	53057	80	1765	45
Clallam	53009	56	1779	32
Snohomish	53061	466	2102	222
Island	53029	60	231	261
Jefferson	53031	20	1773	11
Kitsap	53035	190	391	485
King	53033	1507	2164	696
Mason	53045	38	904	42
Gray Harbor	53027	64	1917	33
Pierce	53053	586	1651	355
Thurston	53067	161	698	231
Pacific	53049	19	945	20
Lewis	53041	59	2479	24

Database management system (DBMS)

- A specialized computer program for organizing and manipulating data.
- Stores the properties of geographic objects and the relationships among the objects.
- Efficient data storage, retrieval, indexing and reporting.
- Database: An organized collection of data
- DBMS: data independency, multiple user view, centralized control and maintenance
- Cost: training and software

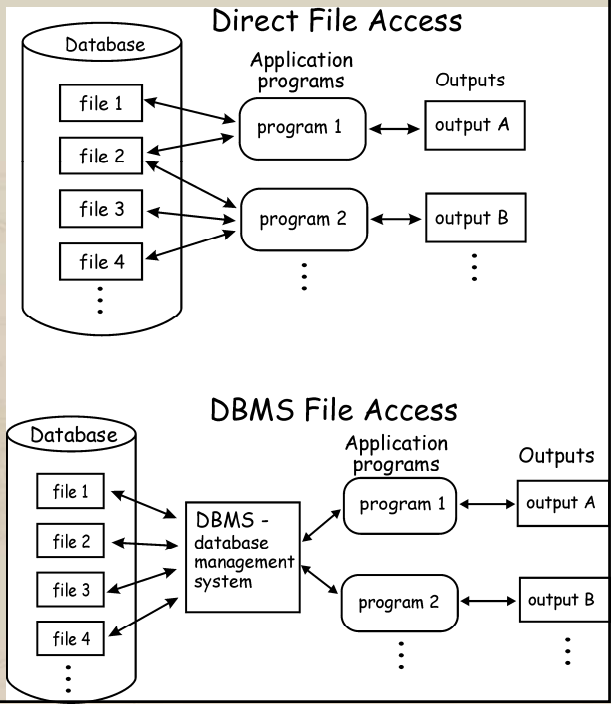
Common components of a database:



A DBMS provides

Translation (many views on the data)

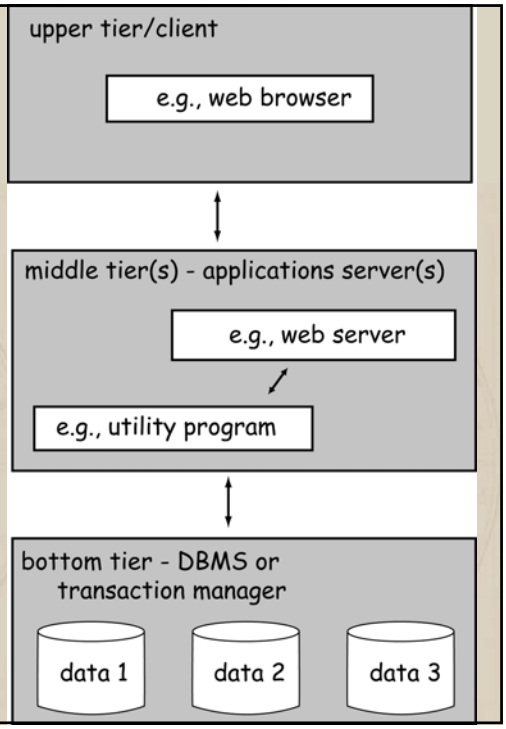
Protection (e.g., against errors due to simultaneous updates)



Multi-tiered architecture

Isolates end user (upper tier) from changes at lower tiers

Allows integration of multiple divergent data sources



Relational Database Model

Minimum row-column structure

Items/records with specified domains (possible values)

Advantages: Minimum structure, easy programming, flexible

Forests

Forest Name	Forest-ID	Location	Size
Nantahala	1	N. Carolina	184,447
Cherokee	2	N. Carolina	92,271

Trails

Trail Name	Forest-ID
Bryson's Knob	1
Slickrock Falls	2
North Fork	1
Cade's Cove	1
Cade's Cove	2
Appalachian	1
Appalachian	2

Characteristics

Trail Name	Feature	Difficulty
Bryson's Knob	Vista	E,M
Bryson's Knob	Ogrth	E,M
Slickrock Falls	Ogrth	M
Slickrock Falls	Wfall	M
North Fork	-	M
Cade's Cove	Ogrth	E
Cade's Cove	Wlife	E
Appalachian	Wfall	M,D
Appalachian	Ogrth	M,D
Appalachian	Vista	M,D
Appalachian	Wlife	M,D
Appalachian	Cmp	M,D

Recreational features

Feature	Description	Activity1	Activity2
Wfall	Waterfall	Photography	Swimming
Ogrth	Old-Growth Forest	Photography	Hiking
Vista	Scenic Overlook	Photography	Viewing
Wlife	Wildlife Viewing	Photography	Birding
Cmp	Camping	Camping	-

Disadvantages:
Relatively slow, a few restrictions on attribute content

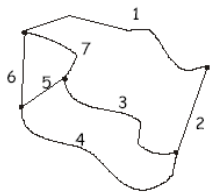
Relational Databases Are Most Common

- Flexible
- Relatively easy to create and maintain
- Computer speeds have overcome slow response in most applications
- Low training costs
- Inertia – many tools are available for RDBMS, large personnel pool

Hybrid Database

Coordinate data as a connected structure

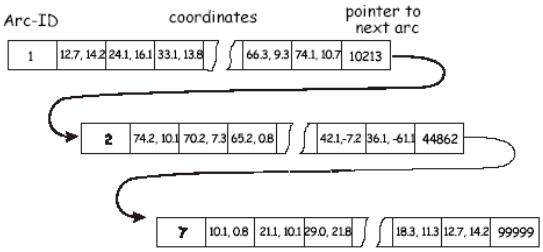
Attribute data as a relational table



Attribute Data

Arc-ID	last-arc	next-arc	type
1	6	2	A
2	2	4	C
3	2	5	F
4	2	5	D
5	6	7	A
6	5	1	F
7	3	6	B

Coordinate Data



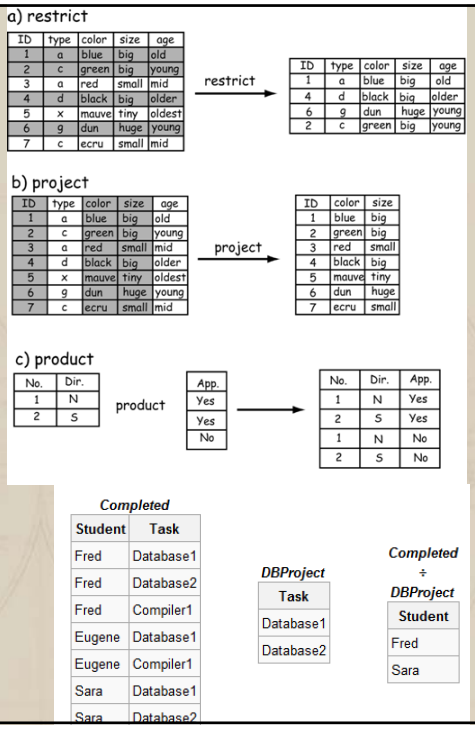
Eight Fundamental Operations

Restrict (query) – subset by rows

Project – subset by columns

Product – all possible combinations

Divide – inverse of product



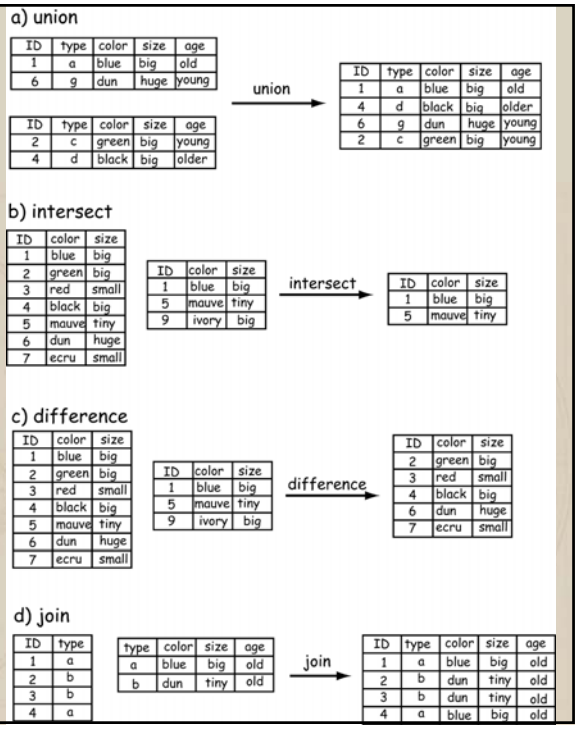
Eight Fundamental Operations

Union – combine top to bottom

Intersect – row overlap

Difference – row non-overlap

Join (relate) – combine by a key column



Main Operations with Relational Tables

Query / Restrict
Conditional selection

Calculation and Assignment

Sort
rank based on attributes

Relate/Join
Temporarily combine two tables by an index

Query / Restrict Operations with Relational Tables

Set Algebra
Uses operations less than (<), greater than (>), equal to (=), and not equal to (<>).

Boolean Algebra
uses the conditions OR, AND, and NOT to select features. Boolean expressions are evaluated by assigning an outcome, True or False, to each condition.

Query / Restrict Operations with Relational Tables

Each record is inspected and is added to the selected set if it meets one to several conditions

AND, OR and NOT may be applied alone or in combinations

AND typically decreases the number of records selected

OR typically increases the number of records selected

NOT Is the negation operation and is interpreted as meaning select those that do not meet the condition following the NOT.

Query / Restrict – simple, AND

Simple Selection:
records with Area > 20.0

ID	Area	Landuse	Municip
1	10.5	Urban	City
2	330.3	Farm	County
3	2.4	Suburban	Township
4	96.0	Suburban	County
5	22.1	Urban	City
6	30.2	Farm	Township
7	4.4	Urban	County

AND Selection:
records with (Landuse = Urban) **and** (Municip = City)

ID	Area	Landuse	Municip
1	10.5	Urban	City
2	330.3	Farm	County
3	2.4	Suburban	Township
4	96.0	Suburban	County
5	22.1	Urban	City
6	30.2	Farm	Township
7	4.4	Urban	County

Query / Restrict – OR, NOT

OR Selection:
records with Area > 20.0
or Municip = City

ID	Area	Landuse	Municip
1	10.5	Urban	City
2	330.3	Farm	County
3	2.4	Suburban	Township
4	96.0	Suburban	County
5	22.1	Urban	City
6	30.2	Farm	Township
7	4.4	Urban	County

NOT Selection:
records with
Landuse NOT Urban

ID	Area	Landuse	Municip
1	10.5	Urban	City
2	330.3	Farm	County
3	2.4	Suburban	Township
4	96.0	Suburban	County
5	22.1	Urban	City
6	30.2	Farm	Township
7	4.4	Urban	County

Operation Order is Important in Query

(D OR E) AND F may not be the same as D OR (E AND F)

NOT (A and B) may not be the same as [NOT (A) AND NOT (B)]

Typically need to clarify order with delimiters

NOT [(Landuse = Urban) AND
(Municip = County)]

ID	Area	Landuse	Municip
1	10.5	Urban	City
2	330.3	Farm	County
3	2.4	Suburban	Township
4	96.0	Suburban	County
5	22.1	Urban	City
6	30.2	Farm	Township
7	4.4	Urban	County

[NOT (Landuse = Urban)] AND
[NOT (Municip = County)]

ID	Area	Landuse	Municip
1	10.5	Urban	City
2	330.3	Farm	County
3	2.4	Suburban	Township
4	96.0	Suburban	County
5	22.1	Urban	City
6	30.2	Farm	Township
7	4.4	Urban	County

Structured Query Language (SQL)

A standard system for query syntax

Uniformly interpreted set of operations, e.g.,
CREATE
INSERT
SELECT

Anybody can (and it appear everybody has) create a database "engine" that fits under SQL – then we can switch vendors and upgrade whenever we want….in theory.

Main Operations with Relational Tables

Query / Selection
Conditional selection

Calculation and Assignment

Sort
rank based on attributes

Relate/Join
Temporarily combine two tables by an index

Calculation and Assignment

Slope = “steep”

Aspect = 45.2

Cost = [M * U + cos (distance)] / (F – P/R*T)

Main Operations with Relational Tables

Query / Selection
Conditional selection

Calculation and Assignment

Sort
rank based on attributes

Relate/Join
Temporarily combine two tables by an index

Sort – ordering by attribute values

Simple sort – ascending AREA

Name	AREA	class	Type
Emily, Lake	52,222.6	1	Limnetic zone
Emily, Lake	58,662.2	1	Limnetic zone
	60,826.6	2	Shallow lakes
	64,588.5	2	Shallow lakes
	70,590.3	2	Shallow lakes
Long Lake	88,259.5	1	Limnetic zone
	143,285.3	2	Littoral zone
Sleepy Eye Lake	170,797.1	2	Littoral zone
Mud Lake	193,318.5	2	Shallow lakes
Goldsmith Lake	201,127.1	2	Littoral zone
Emily, Lake	336,343.2	2	Littoral zone
	349,528.7	1	Limnetic zone
	384,160.1	2	Littoral zone
Emily, Lake	420,798.4	1	Limnetic zone
Savidge Lake	479,709.7	2	Littoral zone
Emily, Lake	545,381.8	1	Limnetic zone
Dog Lake	635,537.0	2	Littoral zone
Duck Lake	1,126,331.9	1	Limnetic zone
Wita Lake	1,354,583.2	2	Littoral zone
	1,418,133.3	1	Limnetic zone
Ballantyne Lake	1,428,331.5	1	Limnetic zone
Washington, Lake	1,914,835.3	1	Limnetic zone
	1,937,698.6	1	Limnetic zone
	4,040,675.7	1	Limnetic zone

Compound sort – ascending Type, then descending AREA within Type

Name	AREA	class	Type
	4,040,675.7	1	Limnetic zone
	1,937,698.6	1	Limnetic zone
Washington, Lake	1,914,835.3	1	Limnetic zone
Ballantyne Lake	1,428,331.5	1	Limnetic zone
	1,418,133.3	1	Limnetic zone
Duck Lake	1,126,331.9	1	Limnetic zone
Emily, Lake	545,381.8	1	Limnetic zone
Emily, Lake	420,798.4	1	Limnetic zone
	349,528.7	1	Limnetic zone
Long Lake	88,259.5	1	Limnetic zone
Emily, Lake	58,662.2	1	Limnetic zone
Emily, Lake	52,222.6	1	Limnetic zone
Dog Lake	635,537.0	2	Littoral zone
Wita Lake	1,354,583.2	2	Littoral zone
Savidge Lake	479,709.7	2	Littoral zone
	384,160.1	2	Littoral zone
Emily, Lake	336,343.2	2	Littoral zone
Goldsmith Lake	201,127.1	2	Littoral zone
Sleepy Eye Lake	170,797.1	2	Littoral zone
	143,285.3	2	Littoral zone
Mud Lake	193,318.5	2	Shallow lakes
	70,590.3	2	Shallow lakes
	64,588.5	2	Shallow lakes
	60,826.6	2	Shallow lakes

Main Operations with Relational Tables

Query / Selection
Conditional selection

Calculation and Assignment

Sort
rank based on attributes

Relate/Join
Temporarily combine two tables by an index

Tables in GIS

Attribute tables are often huge

We have to maintain our tables (change values, remove, add records or items)

Different people/applications are interested in different subsets of attributes (columns)

We often break our tables up into pieces (many tables), and use relational joins as needed to combine them back together

Relational Tables

Relational tables have many advantages, but

If improperly structured, table may suffer from:

Poor performance
Inconsistency
Redundancy
Difficult maintenance

This is common because most users do not understand the concepts Normal Forms in relational tables.

Tables in Non-normal Form

repeat columns, “dependent” data, empty cells by design

Land Records table, unnormalized form

parcel-ID	Alderman	Tship-ID	Tship_name	Thall-add	Own-ID	Own_name	Own_add
2303	Johnson	12	Birch	15W	122	Devlin	123_pine
618	DeSilva	14	Grant	35E	457	Suarez	453_highland
9473	Johnson	12	Birch	15W	337	Yamane	72_lotus

Own-ID	Own_name	Own_add	Own-ID	Own_name	Own_add
337	Yamane	72_lotus	890	Prestovic	12_clayton
890	Prestovic	12_clayton	231	Sherman	64_richmond
-	-	-	-	-	-

Normal Forms Are Good Because:

It reduces total data storage

Changing values in the database is easier

It “insulates” information – it is easier to retain important data

Many operations are easier to code

1st Normal Forms in Relational Tables

Tables are in first normal form when there are no repeated columns

Land Records table, unnormalized form

Parcel-ID	Alderman	Tship-ID	Tship_name	Thall_add	Own-ID	Own_name	Own_add	Own-ID	Own_name	Own_add	Own-ID	Own_name	Own_add
2303	Johnson	12	Birch	15W	122	Devlin	123_pine	337	Yamane	72_lotus	890	Prestovic	12_clayton
618	DeSilva	14	Grant	35E	457	Suarez	453_highland	890	Prestovic	12_clayton	231	Sherman	64_richmond
9473	Johnson	12	Birch	15W	337	Yamane	72_lotus	-	-	-	-	-	-

Land Records table, first normal form (1NF)

Parcel-ID	Alderman	Tship-ID	Tship_name	Thall_add	Own-ID	Own_name	Own_add
2303	Johnson	12	Birch	15W	122	Devlin	123_pine
2303	Johnson	12	Birch	15W	337	Yamane	72_lotus
2303	Johnson	12	Birch	15W	890	Prestovic	12_clayton
618	DeSilva	14	Grant	35E	457	Suarez	453_highland
618	DeSilva	14	Grant	35E	890	Prestovic	12_clayton
618	DeSilva	14	Grant	35E	231	Sherman	64_richmond
9473	Johnson	12	Birch	15W	337	Yamane	72_lotus

Advantages: easy to code queries (can look in only one column)
Disadvantages: slow searches, excess storage, cumbersome maintenance

2nd Normal Forms in Relational Tables

2NF if: *it is in 1NF and if every non-key attribute is functionally dependent on the* ***primary key***

What is a primary key?
An item or set of items that may be used to uniquely identify every row

What is functional dependency?
If you know an item (or items) for a row, then you automatically know a second set of items for the row – this means the second set of items is functionally dependent on the item (or items)

Primary keys

Item(s) that uniquely identify a row

STATE	REGION	SIZE	POPULATION
arkansas	south	small	mid
alaska	west	large	small
alabama	south	small	mid
arizona	west	large	mid
oregon	west	small	mid
wyoming	west	large	small

STATE can be a key, but not REGION, SIZE, or POPULATION

Primary keys

Item(s) that uniquely identify a row

Land Records table, first normal form (1NF)

Parcel-ID	Alderman	Tship-ID	Tship_name	Thall_add	Own-ID	Own_name	Own_add
2303	Johnson	12	Birch	15W	122	Devlin	123_pine
2303	Johnson	12	Birch	15W	337	Yamane	72_lotus
2303	Johnson	12	Birch	15W	890	Prestovic	12_clayton
618	DeSilva	14	Grant	35E	457	Suarez	453_highland
618	DeSilva	14	Grant	35E	890	Prestovic	12_clayton
618	DeSilva	14	Grant	35E	231	Sherman	64_richmond
9473	Johnson	12	Birch	15W	337	Yamane	72_lotus

Sometimes we need >1 column to form a primary key, e.g., Parcel-ID and Own-ID together may form a primary key

Functional Dependency

Knowing the value of an item (or items) means you know the values of other items in the row

e.g., if we know the person's name, then we know the address

In our example, if we know the Parcel-ID, we know the Alderman, Township name, and other Township attributes:

Parcel-ID - > Alderman Parcel-ID - > Thall_add

Parcel-ID - > Tship-ID

Parcel-ID - > Tship_name

Moving from First Normal Form (1NF to Second Normal Form (2NF), we need to:

Identify functional dependencies

Place in separate tables, one key per table

Land Records table, first normal form (1NF)

Parcel-ID	Alderman	Tship-ID	Tship_name	Thall_add	Own-ID	Own_name	Own_add
2303	Johnson	12	Birch	15W	122	Devlin	123_pine
2303	Johnson	12	Birch	15W	337	Yamane	72_lotus
2303	Johnson	12	Birch	15W	890	Prestovic	12_clayton
618	DeSilva	14	Grant	35E	457	Suarez	453_highland
618	DeSilva	14	Grant	35E	890	Prestovic	12_clayton
618	DeSilva	14	Grant	35E	231	Sherman	64_richmond
9473	Johnson	12	Birch	15W	337	Yamane	72_lotus

Given Functional Dependencies:

Parcel-ID → Alderman, Tship-ID

Tship-ID → Tship_name, Thall_add

Own-ID → Own_name, Own_add

Land Records, Second Normal Form

Land Records 1

Parcel-ID	Alderman	Tship-ID	Tship_name	Thall_add
2303	Johnson	12	Birch	15W
618	DeSilva	14	Grant	35E
9473	Johnson	12	Birch	15W

Land Records 2

Own-ID	Own_name	Own_add
122	Devlin	123_pine
337	Yamane	72_lotus
890	Prestovic	12_clayton
457	Suarez	453_highland
231	Sherman	64_richmond

Land Records 3

Parcel-ID	Own-ID
2303	122
2303	337
2303	890
618	457
618	890
618	231
9473	337

3rd Normal Forms in Relational Tables

Remove transitive functional dependencies

A transitive functional dependency is when
A -> B (if we know A, then we know B)
and
B -> C (if we know B, then we know C)
So
A -> C (if we know A, then we know C).

To be in 3NF, we must identify all transitive functional dependencies, and remove them, typically by splitting the table(s) that contain them

Land Records, Second Normal Form

Land Records 1

Parcel-ID	Alderman	Tship-ID	Tship_name	Thall_add
2303	Johnson	12	Birch	15W
618	DeSilva	14	Grant	35E
9473	Johnson	12	Birch	15W

Land Records 2

Own-ID	Own_name	Own_add
122	Devlin	123_pine
337	Yamane	72_lotus
890	Prestovic	12_clayton
457	Suarez	453_highland
231	Sherman	64_richmond

Land Records 3

Parcel-ID	Own-ID
2303	122
2303	337
2303	890
618	457
618	890
618	231
9473	337

In our example, one transitive functional dependency:

Parcel-ID -> Tship-ID, Alderman

Tship-ID -> Tship_name, Thall_add

Land Records 1

Parcel-ID	Alderman	Tship-ID	Tship_name	Thall_add
2303	Johnson	12	Birch	15W
618	DeSilva	14	Grant	35E
9473	Johnson	12	Birch	15W

Land Records, Third Normal Form

Land Records 1a

FD: Parcel-ID Alderman, Tship-ID

Parcel-ID	Alderman	Tship-ID
2303	Johnson	12
618	DeSilva	14
9473	Johnson	12

Land Records 1b

FD: Tship-ID Tship_name, Thall_add

Tship-ID	Tship_name	Thall_add
12	Birch	15W
14	Grant	35E

Land Records 2

FD: Own-ID Own_name, Own_add

Own-ID	Own_name	Own_add
122	Devlin	123_pine
337	Yamane	72_lotus
890	Prestovic	12_clayton
457	Suarez	453_highland
231	Sherman	64_richmond

Land Records 3

No Functional Dependencies

Parcel-ID	Own-ID
2303	122
2303	337
2303	890
618	457
618	890
618	231
9473	337

Bad Things in Relational Tables

Repeat (or similar) variables
e.g., parcel #, owner 1, owner2, owner3, owner 4

Multiple dependencies per record
e.g., owner name, house#, street, city, county, zipcode, state, country

Repeat records

Many blank cells

Normal Forms Summary

No repeat columns (create new records such that there are multiple records per entry)

Split the tables, so that all non-key attributes depend on a primary key.

Split tables further, if there are transitive functional dependencies. This results in tables with a single, primary key per table.

Normal Forms Are Good Because

It reduces total data storage

Changing values in the database is easier

It “insulates” information – it is easier to retain important data

Many operations are easier to code

Terminology

•Database – an organized collection of data
•DBMS – a specialized computer program
•Table – data organized in rows and columns
•Attribute – a variable or item
•Record – a collection of attributes
•Domain – the range of values an attribute may take
•Index/key – attribute(s) used to identify, organize, or order records in a database