Chapter 8: Attribute data and tables

Joaquin Cavieres

Geoinformatics, Bayreuth University

Introduction

Spatial data in GIS are often split into two components:

- The coordinate information that describes object geometry
- The attribute information that describes the nonspatial properties of objects

For non spatial data, they often referred as tabular data.

Spatial data in GIS are often split into two components:

- The coordinate information that describes object geometry
- The attribute information that describes the nonspatial properties of objects

For non spatial data, they often referred as tabular data.

Spatial data in GIS are often split into two components:

- The coordinate information that describes object geometry
- The attribute information that describes the nonspatial properties of objects

For non spatial data, they often referred as tabular data.

Tabular data summarize the most important nonspatial characteristics of each cartographic object (Fig. 1).

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
Grays 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

Figure 1: Data in a GIS include both spatial (left) and attribute (right) components (Bolstad (2016)).

In the previous figure the atrributes are:

- The county
- Federal Information Processing Standards (FIPS) code
- Population
- Area
- Population density

Attribute information in a GIS are typically entered, analyzed, and reported using a *database management system* (DBMS). The DBMS stores the properties of geographic objects and the relationships among the objects.

Sometimes the terms DBMS and database are used interchangeably, but they are different:

- A DBMS is a computer program that allows you to work with data
- A database is an organized collection of data, often created or manipulated with the help of a DBMS

But, can not we just use a spreadsheet?

Functions and programs can be written to address the inefficient use of space, slow processing, and difficulty editing in spreadsheets or other apparently flat file formats, thus:

While spreadsheets can be used for simple collections of data, DBMSs are better for most applications that process large amounts of data.

But, can not we just use a spreadsheet?

Functions and programs can be written to address the inefficient use of space, slow processing, and difficulty editing in spreadsheets or other apparently flat file formats, thus:

While spreadsheets can be used for simple collections of data, DBMSs are better for most applications that process large amounts of data.

But, can not we just use a spreadsheet?

Functions and programs can be written to address the inefficient use of space, slow processing, and difficulty editing in spreadsheets or other apparently flat file formats, thus:

While spreadsheets can be used for simple collections of data, DBMSs are better for most applications that process large amounts of data.

DBMSs provide other advantages such as:

- Data independence
- Multiple user views
- Centralized control and maintenance

Database components and characteristics

The basic components of a traditional database are *data items* or *attributes* (Fig. 2)

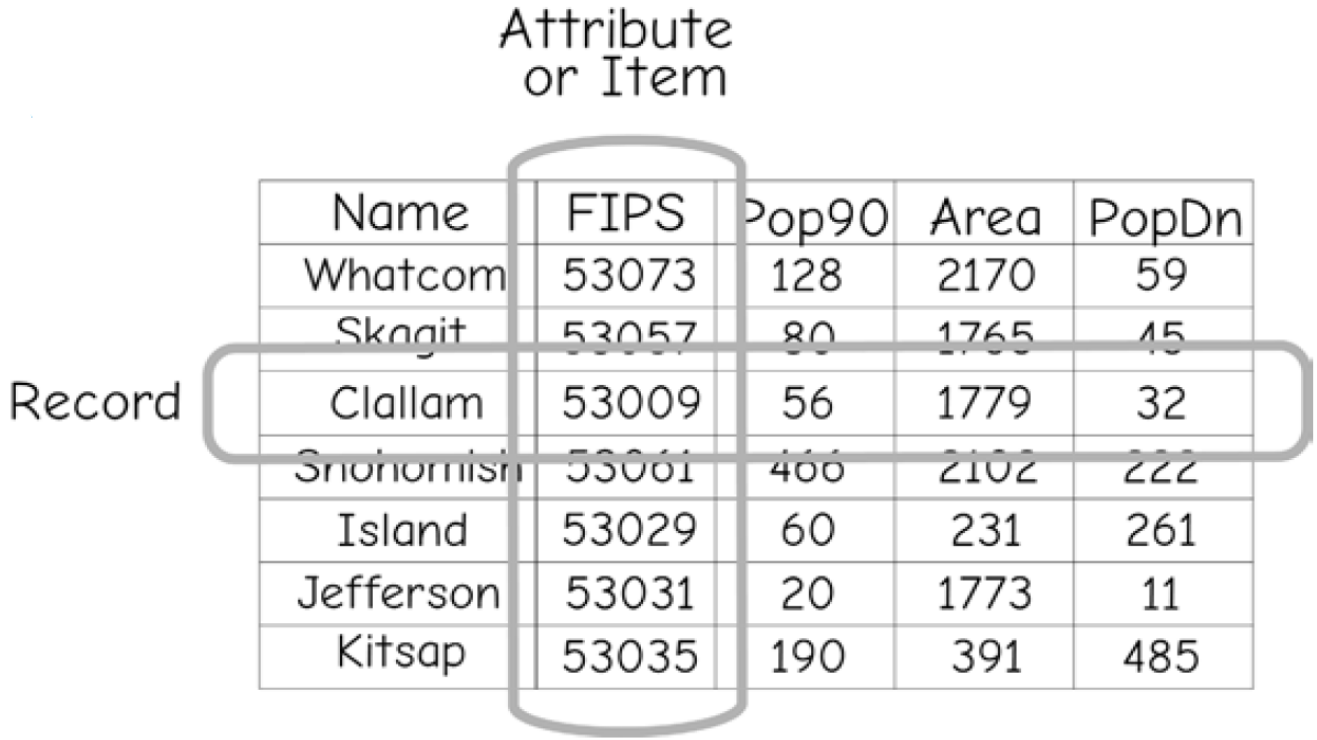

Figure 2: Components of an attribute data table (Bolstad (2016)).

Items have a *type* and a *domain* that restrict the values they may take, and they define essential characteristics of an item.

Common types include:

- Real numbers, integer numbers, hexadecimal numbers and binary numbers.

Domains define the acceptable values an item may take, for example,

- Limits (larger than 0 but smaller than 10)
- A type of color that can only take on the values red, green, blue

A collection of related data items that are treated as a unit represents an *entity*.

In a GIS, the database entities are typically roads, counties, lakes, or other types of geographic features. A specific entity, such as a specific county, is an instance of that entity

These related data items are often organized as a row or line in a table, called a *record*, and a file can contain a collection of records, and a group of files may define the database.

The concept of an entity, when referred to in a database, could be slightly different than an entity in a GIS data model, for the same:

An entity in a geographic data model is often used for the real-world item or phenomenon we are trying to represent with a cartographic object.

Computer scientists and database managers often define an entity as the principal data objects about which information will be collected.

The concept of an entity, when referred to in a database, could be slightly different than an entity in a GIS data model, for the same:

An entity in a geographic data model is often used for the real-world item or phenomenon we are trying to represent with a cartographic object.

Computer scientists and database managers often define an entity as the principal data objects about which information will be collected.

The concept of an entity, when referred to in a database, could be slightly different than an entity in a GIS data model, for the same:

An entity in a geographic data model is often used for the real-world item or phenomenon we are trying to represent with a cartographic object.

Computer scientists and database managers often define an entity as the principal data objects about which information will be collected.

A DBMS typically supports complex structures, primarily to provide data security, to maintain stability, and to allow multiple users or programs to access the same data simultaneously (Fig. 3).

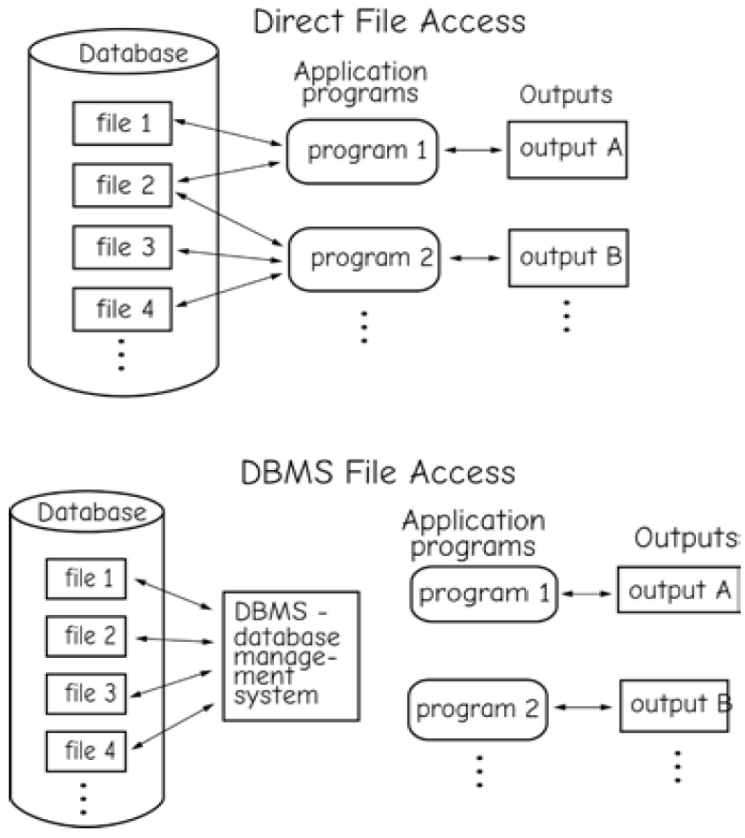

Figure 3: Direct and database management system file access (Bolstad (2016)).

The DBMS is sometimes referred to as a *database server*, and the applications programs as *clients.*

- The server provides or serves up data to the client applications.
- Clients can be built by the DBMS vendor, written by the DBMS user, or sold as add-ons by third-party software developers.

The separation of data and functions into multiple levels is often referred to as a multi-tiered architecture (Fig. 4).

Figure 4: Direct and database management system file access (Bolstad (2016)).

The uppermost tier of multitier architectures a user is typically a user interface.

This tier can be a display by a single purpose or topic specific program such as a GIS, or a Web-based interface with the primary purpose of gathering requests from the user.

Physical, logical, and conceptual structures

A database can be viewed as a conceptual, logical, and physical structures. These structures define the entities and their relationships and specify how the data files or tables are related one to another.

The conceptual structure is often represented in a *schema* and describes the database structure in standard shorthand notations (E-R diagrams.)

The connections within the computer files may by achieved in many ways. Much of this structures is designed to speed access, aid updates, and provide data integrity. This structuring is part of the *physical design* of the database.

Relational Databases

The tables in a relational database design are also called relations (Fig. 5)

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	Activity 1	Activity 2
Wfall	Waterfall	Photography	Swimming
Ogrth	Old-Growth Forest	Photography	Hiking
Vista	Scenic Overlook	Photography	Viewing
Wlife	Wildlife Viewing	Photography	Birding
Cmp	Camping	Camping	-

Figure 5: Forest data in a relational database structure (Bolstad (2016)).

Tables are related through keys, one or more columns that meet certain requirements and can be used to index the rows. Keys are often columns that uniquely identify every row in a table.

Keys are used to join data from one table to associated data in another table (Fig. 6).

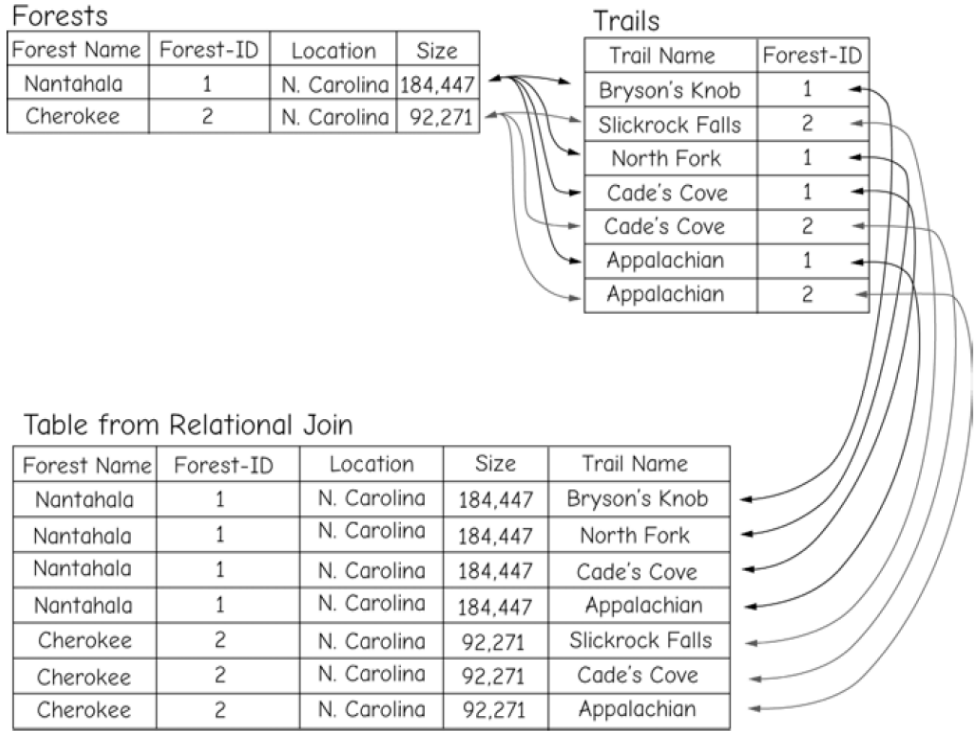

Figure 6: Forest and trails data in a relational data structure. Rows hold records associated with an entity, and columns hold items. A key, here Forest-ID, is used to join tables (Bolstad (2016)).

Keys are the key to the utility and flexibility of relational databases. They allow us to mix and match data from various tables.

Given their importance, there are some restrictions on keys, for example, null values are typically not allowed.

Primary Operators

The general definition of relational databases defines a *relational algebra* and it supports eight primary operators:

- Restrict
- Project
- Union
- Intersection
- Difference
- Product
- joint
- Divide

These operations are applied in queries to select specific records and items (Fig. 7 and 8)

a) restrict

ID	type	color	size	age
1	a	blue	big	old
2	c	green	big	young
3	a	red	small	mid
4	d	black	big	older
5	x	mauve	tiny	oldest
6	g	dun	huge	young
7	c	ecru	small	mid

restrict →

ID	type	color	size	age
1	a	blue	big	old
4	d	black	big	older
6	g	dun	huge	young
2	c	green	big	young

b) project

ID	type	color	size	age
1	a	blue	big	old
2	c	green	big	young
3	a	red	small	mid
4	d	black	big	older
5	x	mauve	tiny	oldest
6	g	dun	huge	young
7	c	ecru	small	mid

project →

ID	color	size
1	blue	big
2	green	big
3	red	small
4	black	big
5	mauve	tiny
6	dun	huge
7	ecru	small

Figure 7: Relational algebra as originally defined supported eight operators: restrict and project (Bolstad (2016)).

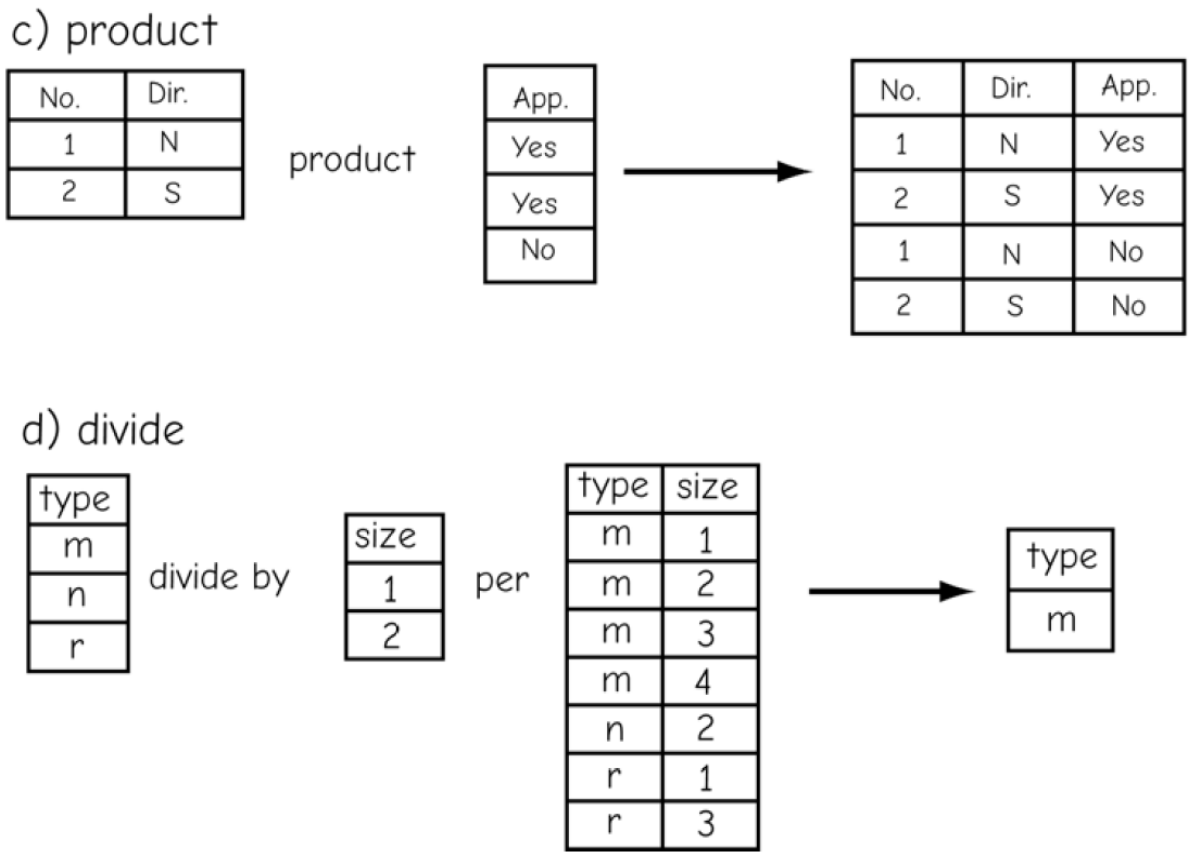

Figure 8: Relational algebra as originally defined supported eight operators: product and division (Bolstad (2016)).

How works each of one of them?

Union

- A union operation combines tables to return records found in either or both tables. the result of a union is at least the size of the largest of the two tables, and no larger than the sum of the two tables.

Intersection

- The intersection operation returns records that occur in both input tables, and omits records found in only one of the two input tables

a) union

ID	type	color	size	age
1	a	blue	big	old
6	g	dun	huge	young

ID	type	color	size	age
2	c	green	big	young
4	d	black	big	older

union

ID	type	color	size	age
1	a	blue	big	old
4	d	black	big	older
6	g	dun	huge	young
2	c	green	big	young

b) intersect

ID	color	size
1	blue	big
2	green	big
3	red	small
4	black	big
5	mauve	tiny
6	dun	huge
7	ecru	small

ID	color	size
1	blue	big
5	mauve	tiny
9	ivory	big

intersect

ID	color	size
1	blue	big
5	mauve	tiny

Figure 9: Relational algebra as originally defined supported eight operators: union and intersect (Bolstad (2016)).

How works each of one of them?

Difference

- The difference operation returns those records that are in the first, but not the second table (Fig. 10). This example shows all those records that are not in both tables.

Join

- A join operation combines two tables through values found in keys. Values in one or more keys are matched across tables, and the information is combined based on the matching.

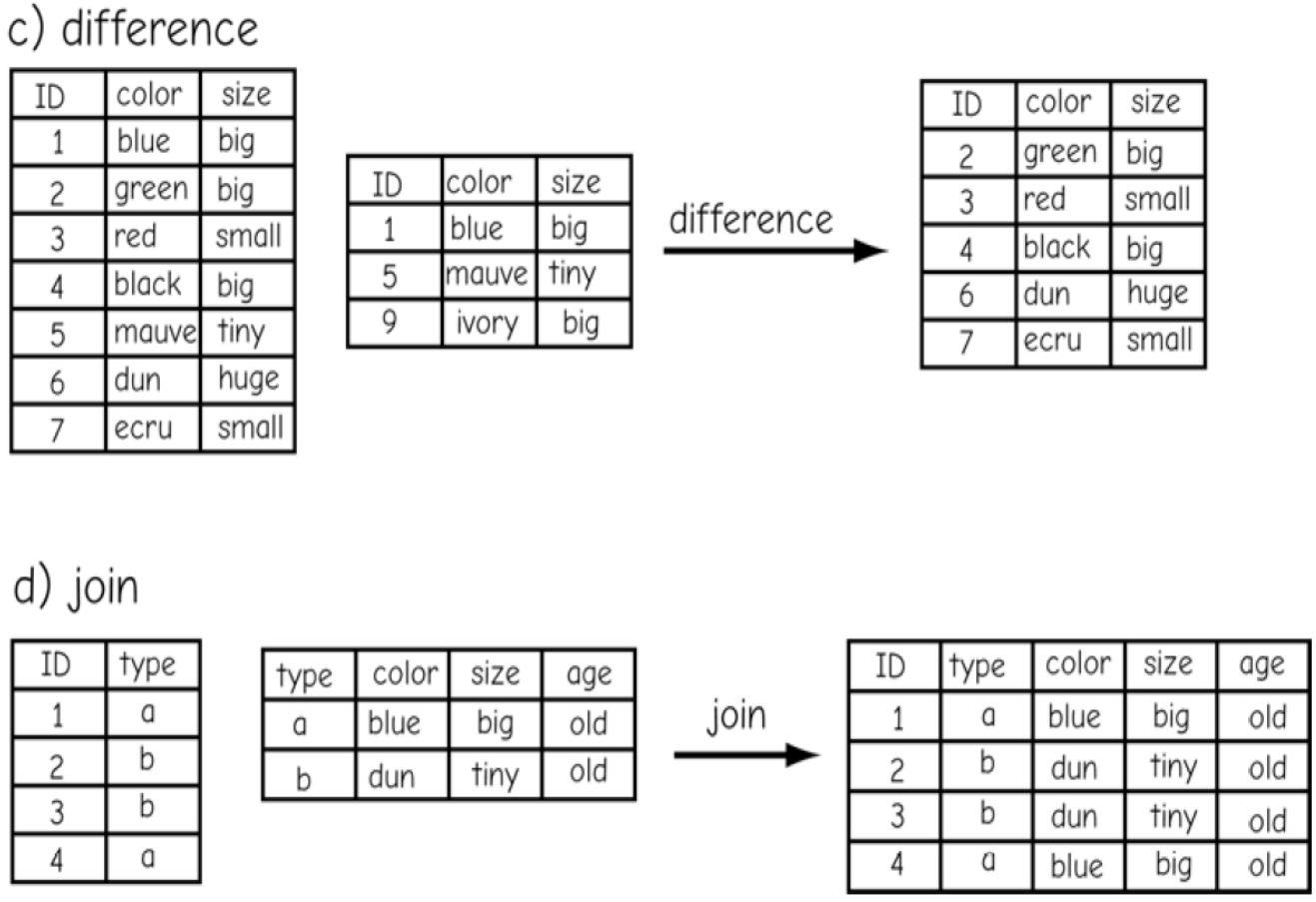

Figure 10: Relational algebra as originally defined supported eight operators: difference and join (Bolstad (2016)).

Limitations of those operators:

- The tables must have the same set of variables or items
- These tables will always have an empty intersection set

Hybrid database designs in GIS

Hybrid designs store coordinate data using specialized database structures, and attribute data in a relational database.

Thousands to millions of coordinate pairs or cells are typically required to represent the location and shape of objects in a GIS. Even with modern computers, the retrieval of coordinate data stored in a relational database design is often too slow.

Therefore, the coordinate data are frequently stored using structures designed for rapid retrieval (Fig. 11).

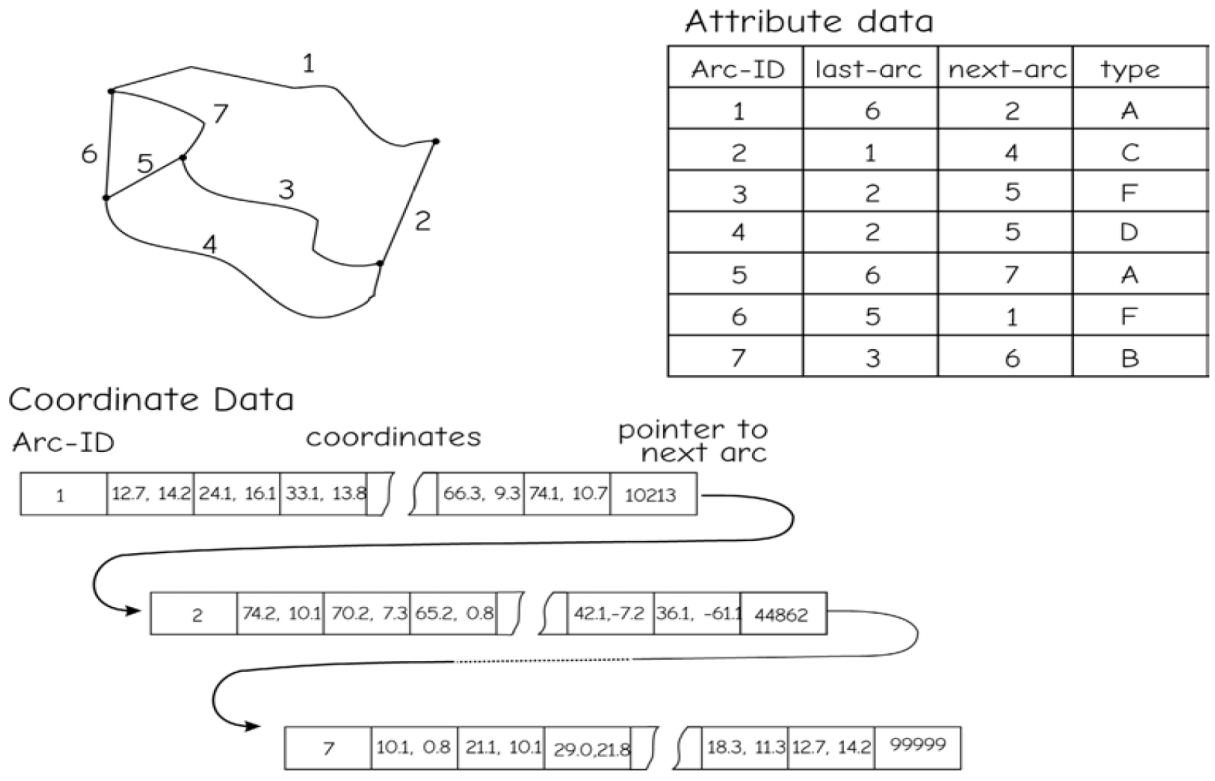

Figure 11: A small example of a hybrid database system for spatial data (Bolstad (2016)).

Hybrid data designs typically store attribute data in a DBMS. These data are linked to the geographic data through unique identifiers or labels that are an attribute in the DBMS.

Selection based on attributes

The restrict operator: table queries

A query could be viewed as the selection of a subset of records based on the values of specified attributes. Queries can be simple, using one variable, or they can be compound, using conditions on more than one variable, or using multiple conditions on one variable.

Many GIS softwares provide a query builder, a graphical user interface (GUI) that helps in applying selection operations (Fig. 12).

Figure 12: A query building GUI of the sort often provided in GIS software, here from QGIS (Bolstad (2016)).

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

Figure 13: Simple selection, applying one criterion to select records (left), and compound selection, applying multiple requirements (right) (Bolstad (2016)).

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

Figure 14: OR and NOT compound selections (Bolstad (2016)).

Complex selection:
records with [(Landuse = Urban) AND (Mill Rate = B)] OR
{NOT(Municip = City) AND (Density > 200)}

ID	Area	Landuse	Municip	Density	Mill Rate
1	10.5	Urban	City	1,112.2	A
2	330.3	Farm	County	1.9	C
3	2.4	Suburban	Township	237.5	C
4	96.0	Suburban	County	98.1	A
5	22.1	Urban	City	916.2	B
6	30.2	Farm	Township	3.7	A
7	4.4	Urban	County	153.8	D

Figure 15: An example of a complex selection, combining various selection operators (Bolstad (2016)).

Remember that in a GIS the tables are usually connected in some way to geographic features, thus selections of table elements imply the selection of associated geographic elements.

It is always a good idea to verify that the selection works as expected (Fig. 6).

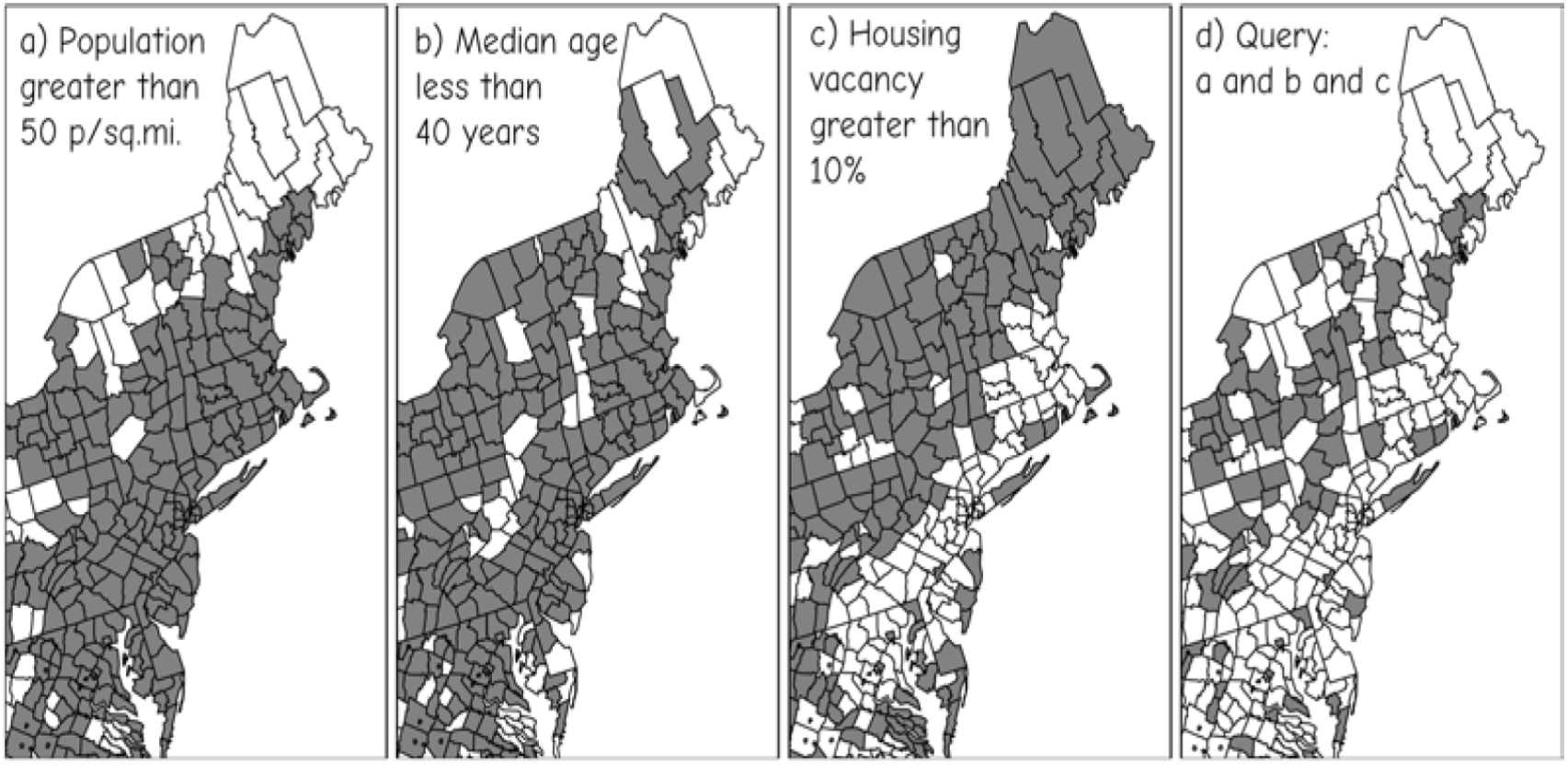

Figure 16: Component and composite selection criteria, applied to counties in the northeastern United States. A visual check of the composite against subcomponents is often helpful, especially when learning (Bolstad (2016)).

SQL provides the capability to both define and manipulate data. Data types can be defined, and tables containing variables of a given type can be specified.

Because SQL as initially defined has limitations for spatial data processing, many spatial operations are not easily represented in SQL.

SQL provides the capability to both define and manipulate data. Data types can be defined, and tables containing variables of a given type can be specified.

Because SQL as initially defined has limitations for spatial data processing, many spatial operations are not easily represented in SQL.

Joining (or relating) tables

Relational databases are so powerful in part because we can structure out data in ways that reduce duplication, are easier to maintain, and are flexible; much of this flexibility is because we can join tables.

Joins are based on joint items or join fields (Fig. 17)

The join operator lets you combine related information in various ways.

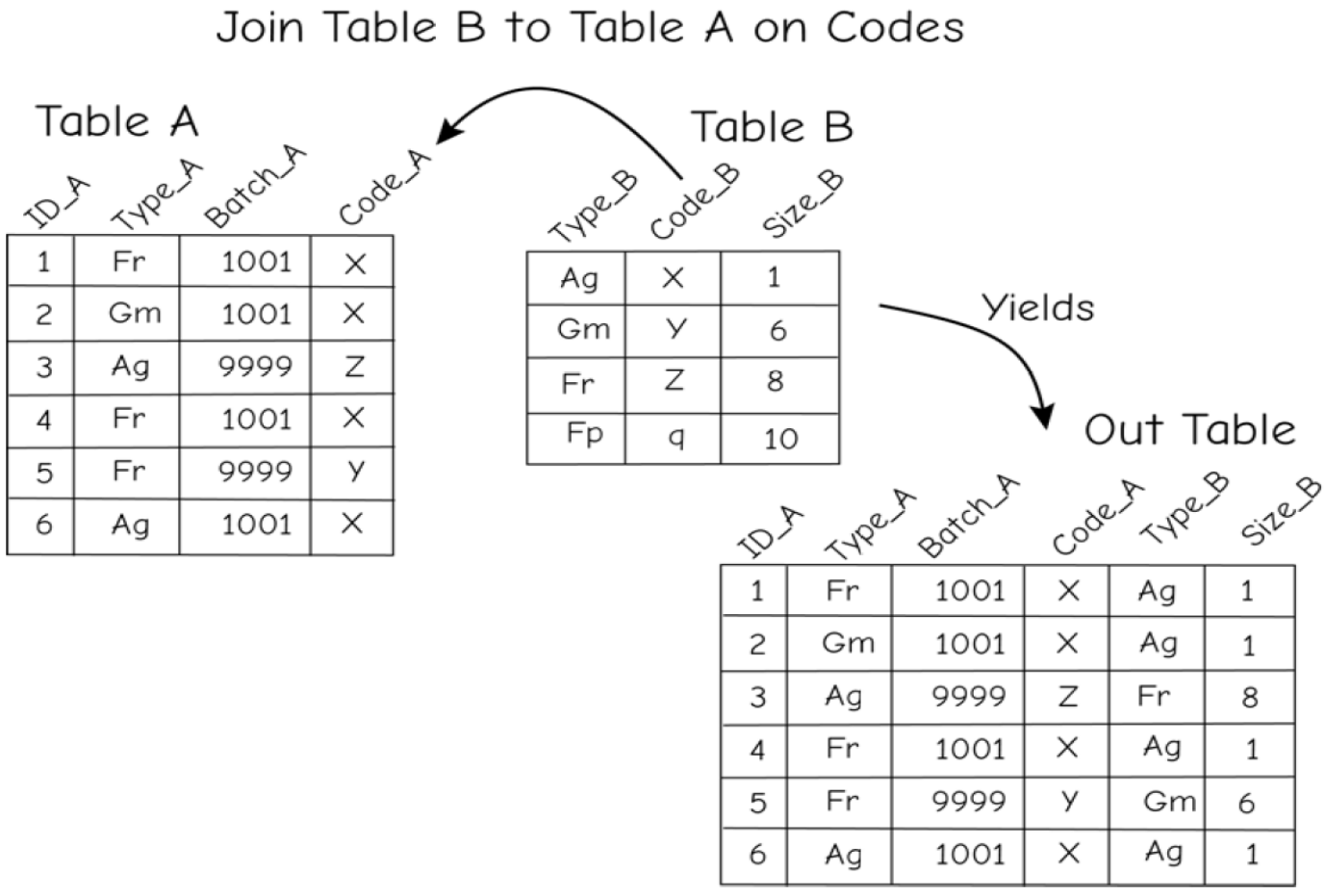

Figure 17: This figure illustrates a simple join. Table B is joined to Table A, matching the Code_B values to corresponding Code_A values, to create Out Table (Bolstad (2016)).

Primary keys and joins

We noted previously that keys are "key" to relational databases,

The most important is the *primary key*, an item (or sometimes multiple items) that uniquely identifies each row in a table. The primary key, or an item that could serve as a primary key, is usually used as a join item in the "source" table of a join operation.

There are differents types of join:

- *inner join*: An output row is created only from rows that match on the join items across tables.
- *outer join*: Saves the information for non matching rows, placing blank or null values for the items for rows that dont have a match in target table.
- *left-right join*: They depends on whether the source or target non matches are in the source or target tables
- *natural join*: Equally named columns are not copied, or cross joins, in which all rows in the source table are combined with all rows of the target table

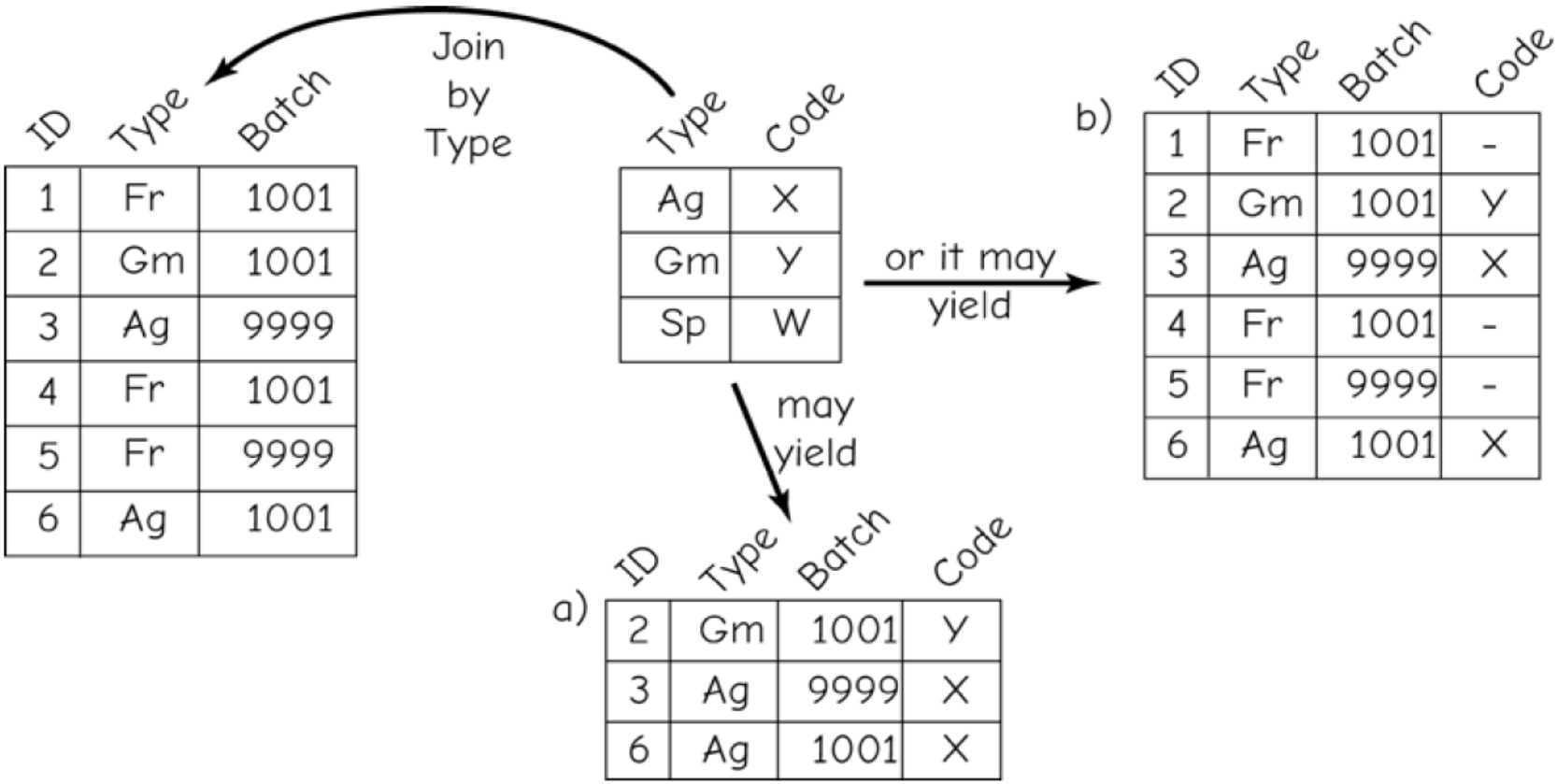

Figure 18: Examples of inner (a) and outer (b) joins. Note that an inner join only saves matching rows, while an outer join saves values for both matching and unmatching rows (Bolstad (2016)).

There are several more options to describe the "join" functionality, but since this is an introductory course, they should be described in detail in future courses.

Normal forms in relational databases

Keys and functional dependencies

The previous sections showed us the need to carefully structure our tables in a relational database, and how the keys in tables are especially important. If not we can have problems in:

- Performance
- Consistency
- Redundancy
- Maintenance.

See next Figure 19

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
-	-	-	-	-	-

Figure 19: Land records data in unnormalized form. The table is shown in two parts because it is too wide to fit across the page (Bolstad (2016)).

For the same, we can place relational databases in normal forms to avoid many of these problems. Data are structured in sequentially higher normal forms to improve correctness, consistency, simplicity, nonredundancy, and stability.

But, prior to describing normal forms we must introduce some terminology and properties of relational tables.

For the same, we can place relational databases in normal forms to avoid many of these problems. Data are structured in sequentially higher normal forms to improve correctness, consistency, simplicity, nonredundancy, and stability.

But, prior to describing normal forms we must introduce some terminology and properties of relational tables.

Relational tables use keys to index data and there are different kind of keys:

- A super key is one or more attributes that can be used to uniquely identify every record (row) for a table
- A candidate key could be a subset of attributes of a super key
- The primary key for indexing a table is chosen from the set of candidate keys

Functional dependency is another important concept. Attributes are functionally dependent if at a given point in time each value of the dependent attribute is determined by a value of another attribute (Fig. 20).

ID	Name	CNum	CType	Thread	Angle
1	Tec	3	M	12	45
2	Cap	1	E	14	20
3	Ext	2	M	12	22
4	Cap	1	M	12	18
5	Tec	3	E	14	20
6	Cap	1	E	14	22
7	Ext	2	Er	14	45

Functional Dependencies:

ID -> Name, CNum, Ctype, Thread, Angle

CNum -> Name (or Name -> CNum)

CType -> Thread

Figure 20: Example functional dependencies (Bolstad (2016)).

Relational database designs are flexible and the use of keys and functional dependencies places restrictions such as:

- There cannot be repeated records
- There must be a primary key in a table
- No member of a column that forms part of the primary key can have a null value

The first and second normal forms

Generally we create tables in normal forms by first gathering all our data in a single table.

Normal forms typically result in many compact, linked tables, so it is quite common to split tables as the database is *normalized*

Tables with repeat groupings (as in the table at the top) are called unnormalized (Fig. 21).

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_highlar
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
-	-	-	-	-	-

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

Figure 21: Relational tables in unnormalized (top) and first normal forms (bottom) (Bolstad (2016)).

Thus, a table is in first normal form when there are no repeat columns.

A table is in second normal form (2NF) if it is in first normal form and every non-key attribute is functionally dependent only on the primary key.

Remember that functional dependency means that knowing the value for one attribute of a record automatically specifies the value for the functionally dependent attribute

The non-key attributes can be directly dependent on the primary key through some functional dependency, or they can be dependent through a transitive dependency.

Thank You

References I

Bolstad, P. (2016). *GIS fundamentals: A first text on geographic information systems*. Eider (PressMinnesota).