Temporal and geospatial databases

COSC430—Advanced Databases

David Eyers

Learning objectives

- You can **design** DBs that handle revisions to data
  - Understand benefits to implementing temporal support within the database engine
  - Able to develop original and complete examples of bitemporal relations

- You can **contrast** spatial data models & geospatial DBs
  - Understand why needs go beyond normal relational operators
  - Outline geospatial capabilities of popular (O)RDBMSs

Motivating example for temporal storage

- Consider a table of employees
- The history of the changing data may be valuable
  - Storing multiple versions will affect normalisation

ENUM	ENAME	POSITION	APPT_DATE
3051	John	[illegible]	[illegible]

Temporal data in the employee table

- What is a good candidate key for this table?
  - As always: we need to think of all valid changes that might occur…

ENUM	ENAME	POSITION	APPT_DATE
3051	John	PR Assist	1.1.2007
3051	John	PR deputy	12.4.2008
3051	John	PR manager	22.3.2013

Temporal data is often important

- Many applications need to include some aspects of time in their data models
  - For example, consider an academic records database
- In many cases the implementation of temporal aspects is left to the application designers…
  - … but then we are back in the 'pre-SQL' era where the database does not understand the application's semantics

Temporal concepts

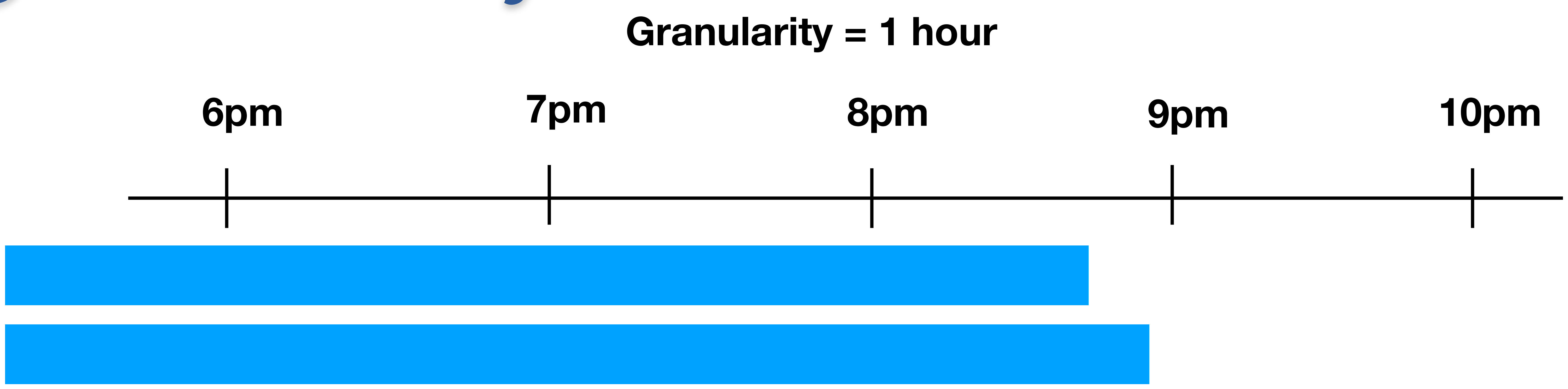
- Every object or entity has
  - a **beginning** at some point in time
  - a **lifespan**—during which its properties may change
  - an **end** at some point in time

- For example:
  - People—school years; student years; working life; …
  - A lab computer—alterations and maintenance; users; usage
  - Land management—alterations; ownership; subdivision; use

More temporal concepts

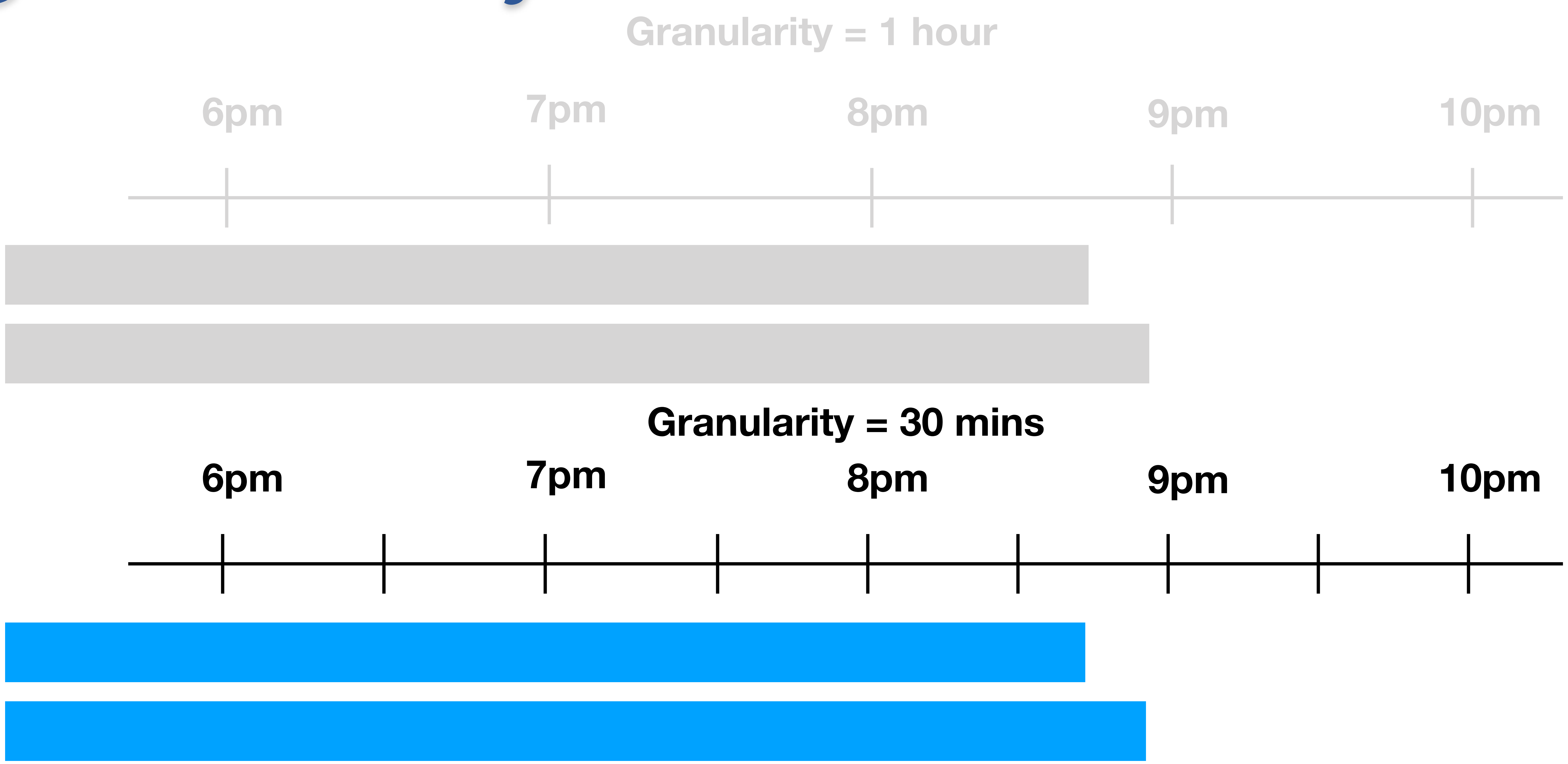
- **Time**—ordered sequence of points in some granularity
- **Granularity**—the unit of measure for a temporal datum and thus a time duration that can't be subdivided
  - e.g., birth dates are usually discussed at the granularity of days, lectures to hours, bus times to minutes, …
- **Chronon**—used (instead of point) to describe this minimal granularity for a particular application
  - An event which happens at an instant exists for one chronon

Time granularity—1 hour



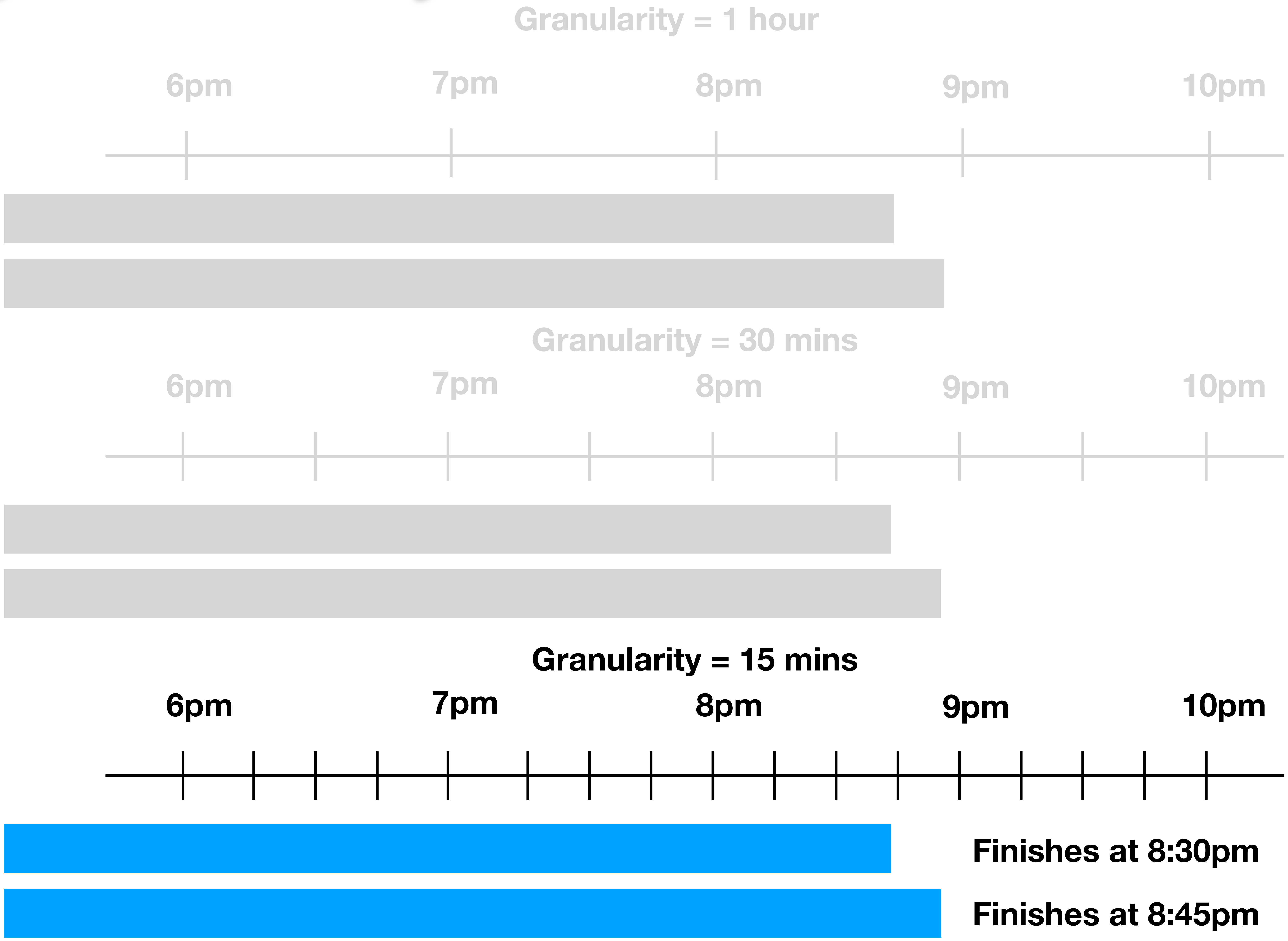
These events finish at the same time (8pm) according to the DB

Time granularity—30 minutes



These events finish at the same time (8:30pm) according to the DB

Time granularity—15 minutes



Two key types of time dimension

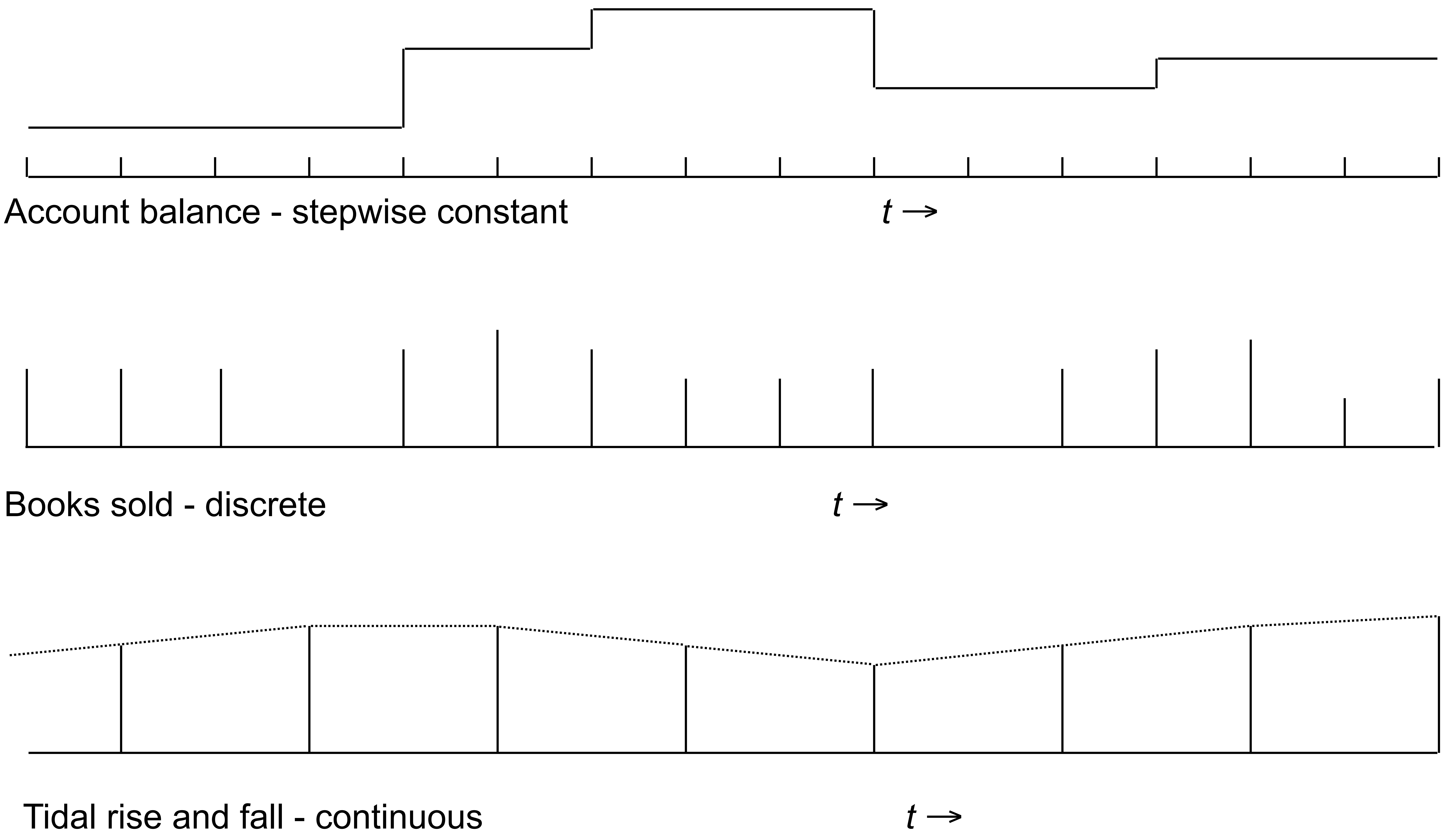
- Valid Time
  - The **real world time** during which an object exists, or when some event takes place
  - (AKA: real-world time, intrinsic time, logical time, data time)

- Transaction Time
  - The **DBMS system time** at which an event was entered into the temporal database
  - (AKA: registration time, extrinsic time, physical time)

Yet more temporal concepts

- Temporal Completeness
  - There are no gaps in a temporal database
  - e.g., database completely describes company developments
- Temporal Density
  - The state of the database at any point in time can be inferred from the available states
- Temporal Isomorphism
  - The database evolves at an order and pace corresponding to the order and pace of the events being modelled

Temporal data arise in different ways



Specific notions of "temporal" in DBs

- **Snapshot**—conventional DB
- **Valid time**—historical values of data
- **Transaction time**—can rollback errors in DB data

- **Bitemporal**—supports transaction time and valid time
  - AKA temporal or fully temporal

Allen's temporal operators / algebra

- For two time intervals X and Y, Allan describes 13 potential relationships
  - All but EQUALS have an inverse form to count, e.g., Y BEFORE X

```
X EQUALS Y     XXX------        X DURING Y     --XXX-
               YYY------                       YYYYYY
X BEFORE Y     XXX---YYY        X STARTS Y     XXX---
X MEETS Y      XXXYYY---                       YYYYY-
X OVERLAPS Y   XXX------        X FINISHES Y   --XXX-
               -YYY-----                       YYYYY-
```

Allen's temporal operators as expressions

- Given two time intervals i1 [s1,e1] and i2 [s2,e2]:

X EQUALS Y	iff s1 = s2 and e1 = e2
X BEFORE Y	iff e1 < s2
X MEETS Y	iff s2 = e1+1
X OVERLAPS Y	iff s1 < s2 and e1 >= s2 AND e1 < e2
X DURING Y	iff s1 > s2 and e1 < e2
X STARTS Y	iff s1 = s2 and e1 < e2
X FINISHES Y	iff e1 = e2 and s1 > s2

Some data model design decisions—objects

- Is **valid time** associated with tuples or attribute values?
  - Are attribute values atomic or set-valued?
  - How is valid time represented? Chronons? Sets of intervals? …
- Is **transaction time** associated with attribute values, tuples or sets of tuples?
- We'll assume both are associated with tuples using new attributes, V_start, V_end, T_start and T_end

Bitemporal e.g. using "suppliers-and-parts"

- An engineering company (an airline?) deals with 5 suppliers:
  - Smith, based in London
  - Jones, based in Paris
  - Blake, based in Paris, etc.
- The company holds stocks of parts at various locations:
  - nuts and screws in London
  - bolts in Paris
  - screws in Rome
- Each supplier may supply any or all of the parts:
  - Smith supplies nuts, bolts, screws
  - Jones supplies nuts, bolts
  - Blake supplies bolts

"Suppliers-and-parts" DB—data values

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

P

P#	PNAME	COLOUR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

A bitemporal relation: suppliers—Monday

- Three suppliers are contracted to supply goods to the company, starting on Monday
- They are loaded into the database on Monday

SNUM	SNAME	ST	CITY	V_Start	V_End	T_Start	T_End
S1	Smith	20	London	Mon 9:00	⊥	Mon 9:05	uc
S2	Jones	10	Paris	Mon 9:00	⊥	Mon 9:06	uc
S3	Blake	30	Paris	Mon 9:00	⊥	Mon 9:07	uc
					"now"	"until changed"	

A bitemporal relation: suppliers—Tuesday

- A new supplier, S4, starts as a contractor; DB updated
- Supplier S1's contract is cancelled as from today and this information is entered today

SNUM	SNAME	ST	CITY	V_Start	V_End	T_Start	T_End
S1	Smith	20	London	Mon 9:00	⊥	Mon 9:05	Tue 14:00
S2	Jones	10	Paris	Mon 9:00	⊥	Mon 9:06	uc
S3	Blake	30	Paris	Mon 9:00	⊥	Mon 9:07	uc
S4	Clark	20	London	Tue 9:00	⊥	Tue 9:05	uc
S1	Smith	20	London	Mon 9:00	Tue 12:00	Tue 14:01	uc

A bitemporal relation: suppliers—Weds.

- Supplier S3 moves to NY; this change noted today
- A new supplier, S57, Heinz, 50, Berlin starts on today and this data is entered today

SNUM	SNAME	ST	CITY	V_Start	V_End	T_Start	T_End
S1	Smith	20	London	Mon 9:00	⊥	Mon 9:05	Tue 14:00
S2	Jones	10	Paris	Mon 9:00	⊥	Mon 9:06	uc
S3	Blake	30	Paris	Mon 9:00	⊥	Mon 9:07	Wed 15:00
S4	Clark	20	London	Tue 9:00	⊥	Tue 9:05	uc
S1	Smith	20	London	Mon 9:00	Tue 12:00	Tue 14:01	uc
S3	Blake	30	Paris	Mon 9:00	Wed 14:45	Wed 15:01	uc
S3	Blake	30	New York	Wed 14:46	⊥	Wed 15:02	uc
S57	Heinz	50	Berlin	Wed 9:00	⊥	Wed 16:00	uc

A bitemporal relation: suppliers—Thursday

- Auditor finds problems:
  - Supplier S2 didn't actually start until Tuesday
  - Supplier S57 doesn't exist and never did!
  - Smith (S1) is in Paris and always has been

SNUM	SNAME	ST	CITY	V_Start	V_End	T_Start	T_End
S1	Smith	20	London	Mon 9:00	⊥	Mon 9:05	Tue 14:00
S2	Jones	10	Paris	Mon 9:00	⊥	Mon 9:06	Th 12:00
S3	Blake	30	Paris	Mon 9:00	⊥	Mon 9:07	Wed 15:00
S4	Clark	20	London	Tue 9:00	⊥	Tue 9:05	uc
S1	Smith	20	London	Mon 9:00	Tue 12:00	Tue 14:01	Th 15:00
S3	Blake	30	Paris	Mon 9:00	Wed 14:45	Wed 15:01	uc
S3	Blake	30	New York	Wed 14:46	⊥	Wed 15:02	uc
S57	Heinz	50	Berlin	Wed 9:00	⊥	Wed 16:00	Th 12:30
S2	Jones	10	Paris	Tue 9:00	⊥	Th 12:01	uc
S1	Smith	20	Paris	Mon 9:00	Tue 12:00	Th 15:01	uc

A bitemporal relation: suppliers—Friday

- New supplier, S7, starts and the DB is updated today
- Supplier S1, trading as S1a, Smith International of Rome, given new contract and DB is updated the same day

SNUM	SNAME	ST	CITY	V_Start	V_End	T_Start	T_End
S1	Smith	20	London	Mon 9:00	⊥	Mon 9:05	Tue 14:00
S2	Jones	10	Paris	Mon 9:00	⊥	Mon 9:06	Th 12:00
S3	Blake	30	Paris	Mon 9:00	⊥	Mon 9:07	Wed 15:00
S4	Clark	20	London	Tue 9:00	⊥	Tue 9:05	uc
S1	Smith	20	London	Mon 9:00	Tue 12:00	Tue 14:01	Th 15:00
S3	Blake	30	Paris	Mon 9:00	Wed 14:45	Wed 15:01	uc
S3	Blake	30	New York	Wed 14:46	⊥	Wed 15:02	uc
S57	Heinz	50	Berlin	Wed 9:00	⊥	Wed 16:00	Th 12:30
S2	Jones	10	Paris	Tue 9:00	⊥	Th 12:01	uc
S1	Smith	20	Paris	Mon 9:00	Tue 12:00	Th 15:01	uc
S7	McD	99	Dunedin	Fri 9:00	⊥	Fri 9:30	uc
S1a	Smith Int	20	Rome	Fri 9:00	⊥	Fri 15:00	uc

Rearrange rows to group supplier's history

SNUM	SNAME	ST	CITY	V_Start	V_End	T_Start	T_End
S1	Smith	20	London	Mon 9.00	⊥	Mon 9.05	Tue 14.00
S1	Smith	20	London	Mon 9.00	Tue 12.00	Tue 14.01	Th 15.00
S1	Smith	20	Paris	Mon 9.00	Tue 12.00	Th 15.01	uc
S2	Jones	10	Paris	Mon 9.00	⊥	Mon 9.06	Th 12.00
S2	Jones	10	Paris	Tue 9.00	⊥	Th 12.01	uc
S3	Blake	30	Paris	Mon 9.00	⊥	Mon 9.07	Wed 15.00
S3	Blake	30	Paris	Mon 9.00	Wed 14.45	Wed 15.01	uc
S3	Blake	30	New York	Wed 14.46	⊥	Wed 15.02	uc
S4	Clark	20	London	Tue 9.00	⊥	Tue 9.05	uc
S57	Heinz	50	Berlin	Wed 9.00	⊥	Wed 16.00	Th 12.30
S7	McD	99	Dunedin	Fri 9.00	⊥	Fri 9.30	uc
S1a	Smith Int	20	Rome	Fri 9.00	⊥	Fri 15.00	uc

Note the "no delete" approach

- No data values in a row are ever deleted
- No V_start and T_start values are ever changed
- V_end and T_end values false and "uc" may be changed, but only once
- Changes to the data are recorded by adding rows

- A properly designed bitemporal database should withstand any user's attempt to rewrite history!

List all current suppliers (asked Fri 12.00)

SELECT S.*
FROM supplier S
WHERE VALID (S)
CONTAINS “Friday 12:00”
AS OF “Friday 12:00” ;

SNUM	SNAME	ST	CITY	V_Start	V_End	T_Start	T_End
~~S1~~	~~Smith~~	~~20~~	~~London~~	~~Mon 9.00~~	~~⊥~~	~~Mon 9.05~~	~~Tue 14.00~~
~~S1~~	~~Smith~~	~~20~~	~~London~~	~~Mon 9.00~~	~~Tue 12.00~~	~~Tue 14.01~~	~~Th 15.00~~
~~S1~~	~~Smith~~	~~20~~	~~Paris~~	~~Mon 9.00~~	~~Tue 12.00~~	~~Th 15.01~~	~~uc~~
~~S2~~	~~Jones~~	~~10~~	~~Paris~~	~~Mon 9.00~~	~~⊥~~	~~Mon 9.06~~	~~Th 12.00~~
S2	Jones	10	Paris	Tue 9.00	⊥	Th 12.01	uc
~~S3~~	~~Blake~~	~~30~~	~~Paris~~	~~Mon 9.00~~	~~⊥~~	~~Mon 9.07~~	~~Wed 15.00~~
~~S3~~	~~Blake~~	~~30~~	~~Paris~~	~~Mon 9.00~~	~~Wed 14.45~~	~~Wed 15.01~~	~~uc~~
S3	Blake	30	New York	Wed 14.46	⊥	Wed 15.02	uc
S4	Clark	20	London	Tue 9.00	⊥	Tue 9.05	uc
~~S57~~	~~Heinz~~	~~50~~	~~Berlin~~	~~Wed 9.00~~	~~⊥~~	~~Wed 16.00~~	~~Th 12.30~~
S7	McD	99	Dunedin	Fri 9.00	⊥	Fri 9.30	uc
~~S1a~~	~~Smith Int~~	~~20~~	~~Rome~~	~~Fri 9.00~~	~~⊥~~	~~Fri 15.00~~	~~uc~~

List all suppliers Monday (asked Fri 16.00)

SNUM	SNAME	ST	CITY	V_Start	V_End	T_Start	T_End
~~S1~~	~~Smith~~	~~20~~	~~London~~	~~Mon 9.00~~	~~⊥~~	~~Mon 9.05~~	~~Tue 14.00~~
~~S1~~	~~Smith~~	~~20~~	~~London~~	~~Mon 9.00~~	~~Tue 12.00~~	~~Tue 14.01~~	~~Th 15.00~~
S1	Smith	20	Paris	Mon 9.00	Tue 12.00	Th 15.01	uc
~~S2~~	~~Jones~~	~~10~~	~~Paris~~	~~Mon 9.00~~	~~⊥~~	~~Mon 9.06~~	~~Th 12.00~~
~~S2~~	~~Jones~~	~~10~~	~~Paris~~	~~Tue 9.00~~	~~⊥~~	~~Th 12.01~~	~~uc~~
~~S3~~	~~Blake~~	~~30~~	~~Paris~~	~~Mon 9.00~~	~~⊥~~	~~Mon 9.07~~	~~Wed 15.00~~
S3	Blake	30	Paris	Mon 9.00	Wed 14.45	Wed 15.01	uc
~~S3~~	~~Blake~~	~~30~~	~~New York~~	~~Wed 14.46~~	~~⊥~~	~~Wed 15.02~~	~~uc~~
~~S4~~	~~Clark~~	~~20~~	~~London~~	~~Tue 9.00~~	~~⊥~~	~~Tue 9.05~~	~~uc~~
~~S57~~	~~Heinz~~	~~50~~	~~Berlin~~	~~Wed 9.00~~	~~⊥~~	~~Wed 16.00~~	~~Th 12.30~~
~~S7~~	~~McD~~	~~99~~	~~Dunedin~~	~~Fri 9.00~~	~~⊥~~	~~Fri 9.30~~	~~uc~~
~~S1a~~	~~Smith Int~~	~~20~~	~~Rome~~	~~Fri 9.00~~	~~⊥~~	~~Fri 15.00~~	~~uc~~

List all suppliers on Mon. (asked Tue 11.00)

SNUM	SNAME	ST	CITY	V_Start	V_End	T_Start	T_End
S1	Smith	20	London	Mon 9.00	⊥	Mon 9.05	Tue 14.00
~~S1~~	~~Smith~~	~~20~~	~~London~~	~~Mon 9.00~~	~~Tue 12.00~~	~~Tue 14.01~~	~~Th 15.00~~
~~S1~~	~~Smith~~	~~20~~	~~Paris~~	~~Mon 9.00~~	~~Tue 12.00~~	~~Th 15.01~~	~~uc~~
S2	Jones	10	Paris	Mon 9.00	⊥	Mon 9.06	Th 12.00
~~S2~~	~~Jones~~	~~10~~	~~Paris~~	~~Tue 9.00~~	~~⊥~~	~~Th 12.01~~	~~uc~~
S3	Blake	30	Paris	Mon 9.00	⊥	Mon 9.07	Wed 15.00
~~S3~~	~~Blake~~	~~30~~	~~Paris~~	~~Mon 9.00~~	~~Wed 14.45~~	~~Wed 15.01~~	~~uc~~
~~S3~~	~~Blake~~	~~30~~	~~New York~~	~~Wed 14.46~~	~~⊥~~	~~Wed 15.02~~	~~uc~~
~~S4~~	~~Clark~~	~~20~~	~~London~~	~~Tue 9.00~~	~~⊥~~	~~Tue 9.05~~	~~uc~~
~~S57~~	~~Heinz~~	~~50~~	~~Berlin~~	~~Wed 9.00~~	~~⊥~~	~~Wed 16.00~~	~~Th 12.30~~
~~S7~~	~~McD~~	~~99~~	~~Dunedin~~	~~Fri 9.00~~	~~⊥~~	~~Fri 9.30~~	~~uc~~
~~S1a~~	~~Smith Int~~	~~20~~	~~Rome~~	~~Fri 9.00~~	~~⊥~~	~~Fri 15.00~~	~~uc~~

Summary—temporal databases

- Most DBs store current data: but history of data can be needed
- Temporal concepts
  - granularity
  - valid time
  - transaction time
- A temporal data model
  - objects
  - operators
- Allen's temporal operators
- Bi-temporal relations:
  - track real lifetime of object (valid time) and how and when data about the object was entered into the database (transaction time)
  - Provided example relation
  - Showed queries involving both valid time and transaction time

Geographic Information Systems (GIS)

- We need to define **spatial data** before exploring GISs
  - (Spatial databases and graph databases are often related)

- Open Geospatial Consortium’s list of features:
  - **Spatial measurements**—e.g. length of line, area of polygon
  - **Spatial functions**—e.g. intersection of features, buffers
  - **Spatial predicates**—e.g. true/false queries, does x overlap y?
  - **Geometry constructions**—e.g. from list of points, create shape
  - **Observer functions**—e.g. queries’ features like center of circle

There are many ways to store spatial data

- Need to consider **topology**: *i.e.*, how space interrelates
  - Most use cases don't use complex mathematical possibilities

- Non exhaustive list of aspects to consider includes:
  - **Adjacency**—what is next to what?
  - **Proximity**—how close are different parts of the space?
  - **Connectivity**—which objects are interconnected?
  - **Enclosure**—what contains what?

Some conceptual models for spatial data

- An overview of possibilities (more detail is coming):
  - **Raster data model**—data is sampled from a grid
  - **Vector data model**—geometry of data is defined symbolically
  - **Network model**—directly represent connectivity (i.e., graphs)

- **TIN data model** for GIS—Triangulated Irregular Networks
  - Can be thought of as a **hybrid** between raster and vector
  - Acquired data is probably raster: *e.g.*, points of elevation
  - TIN—vector representation from samples: *e.g.*, contour lines

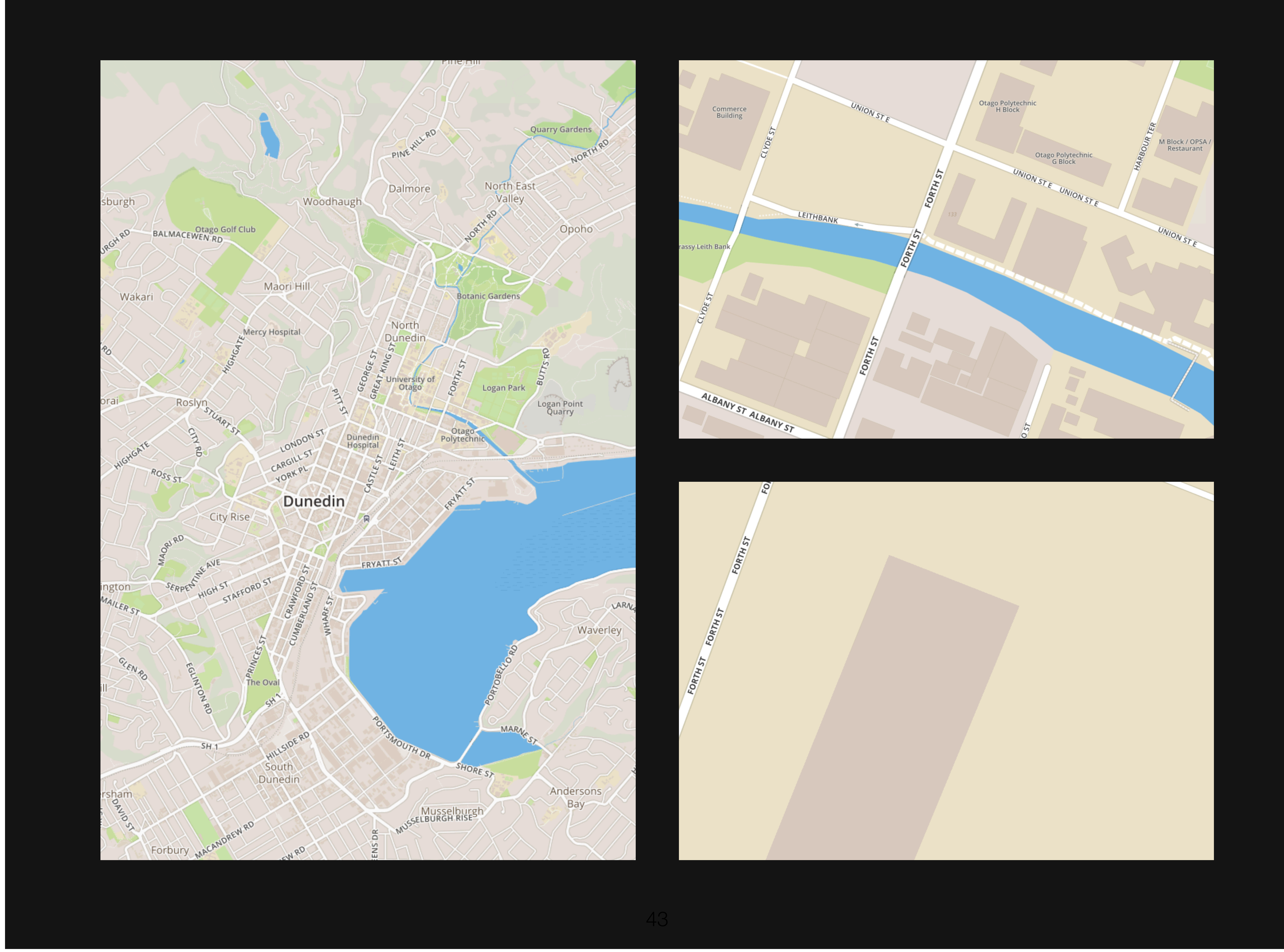
Raster data model

- Typically data is sampled on a regular grid
  - Sample values can still be either categorical or continuous
  - Modern computer displays are raster
  - Underlying data can be resampled onto raster
- What is best stored in this model?
  - Often data that is displayed as a "heat map"
  - Meteorological e.g., rainfall, sunshine, temperature distribution

Raster data example

Vector data model

- Data is represented by geometric descriptions such as:
  - Points
  - Lines
  - Polygons



- What is best stored in this model?
  - Maps of roads, land title boundaries, cable/pipe routing, etc.
  - Topology of data is captured—how items interconnect

Converting from raster to vector

- Manually performing measurements gives vector data
- High speed geospatial acquisition: gives raster data
  - e.g. LIDAR, or reference photographs

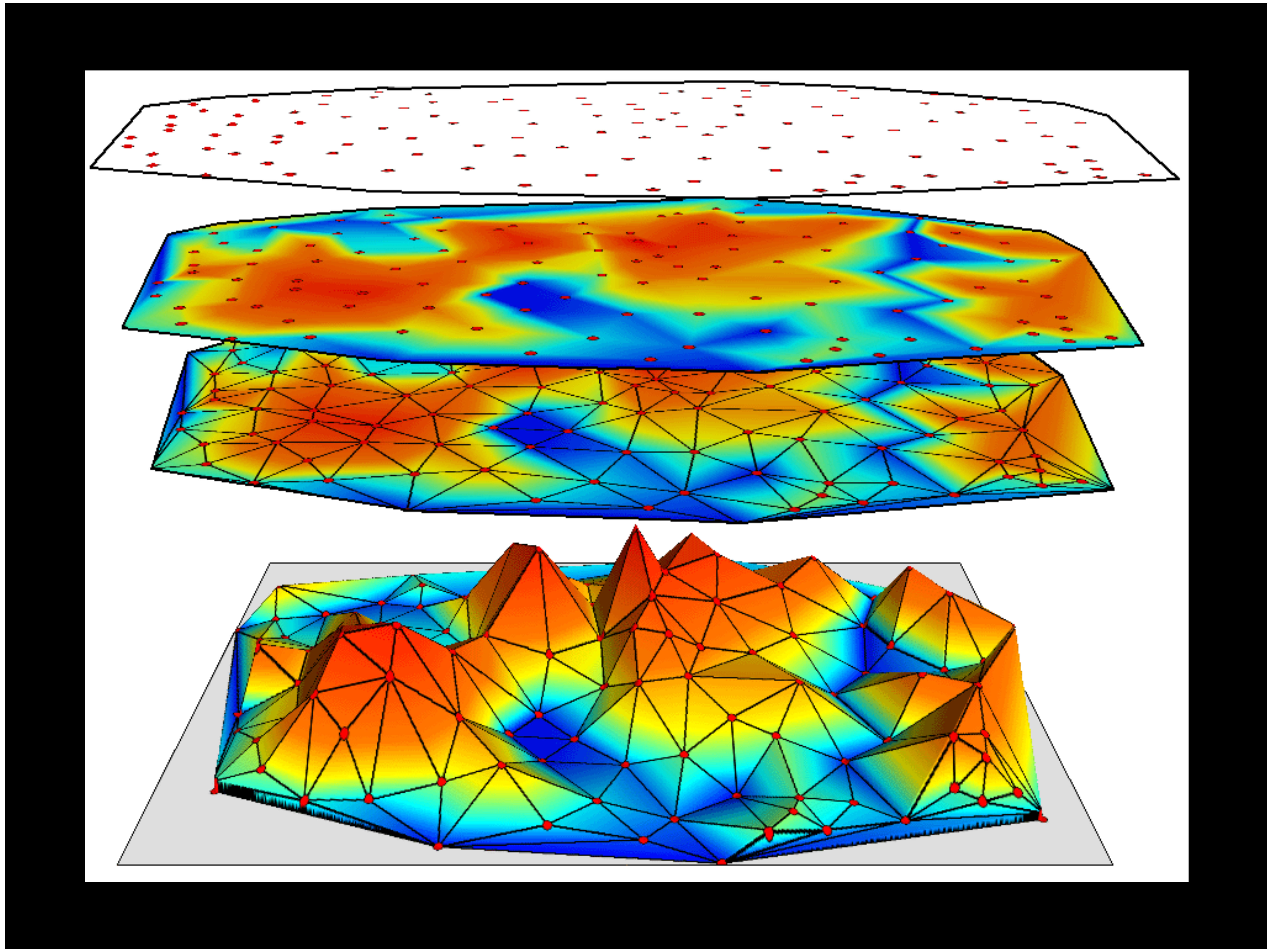
- Auto-converting raster to vector sometimes possible
  - Photogrammetry (surveying)
  - Structure from motion (computer vision)

- *E.g.*, our scalable 3D Reconstruction (S3DR) project

Classic example of a network

TIN data example

- Figure shows layers from top to bottom:
  - Discrete data points
  - Heatmap formed
  - Triangulation applied
  - Hybrid vector/raster

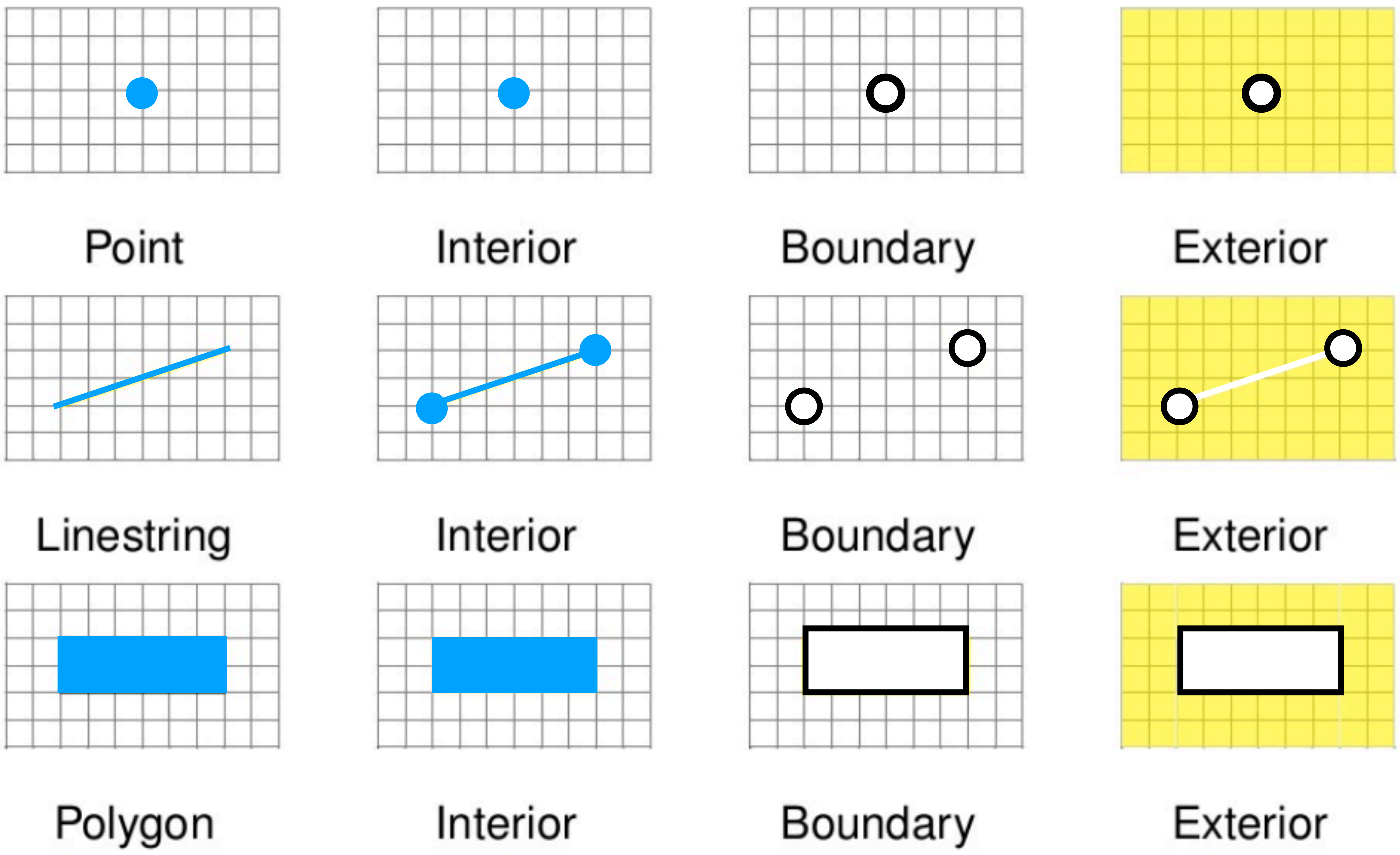


Spatial relationships

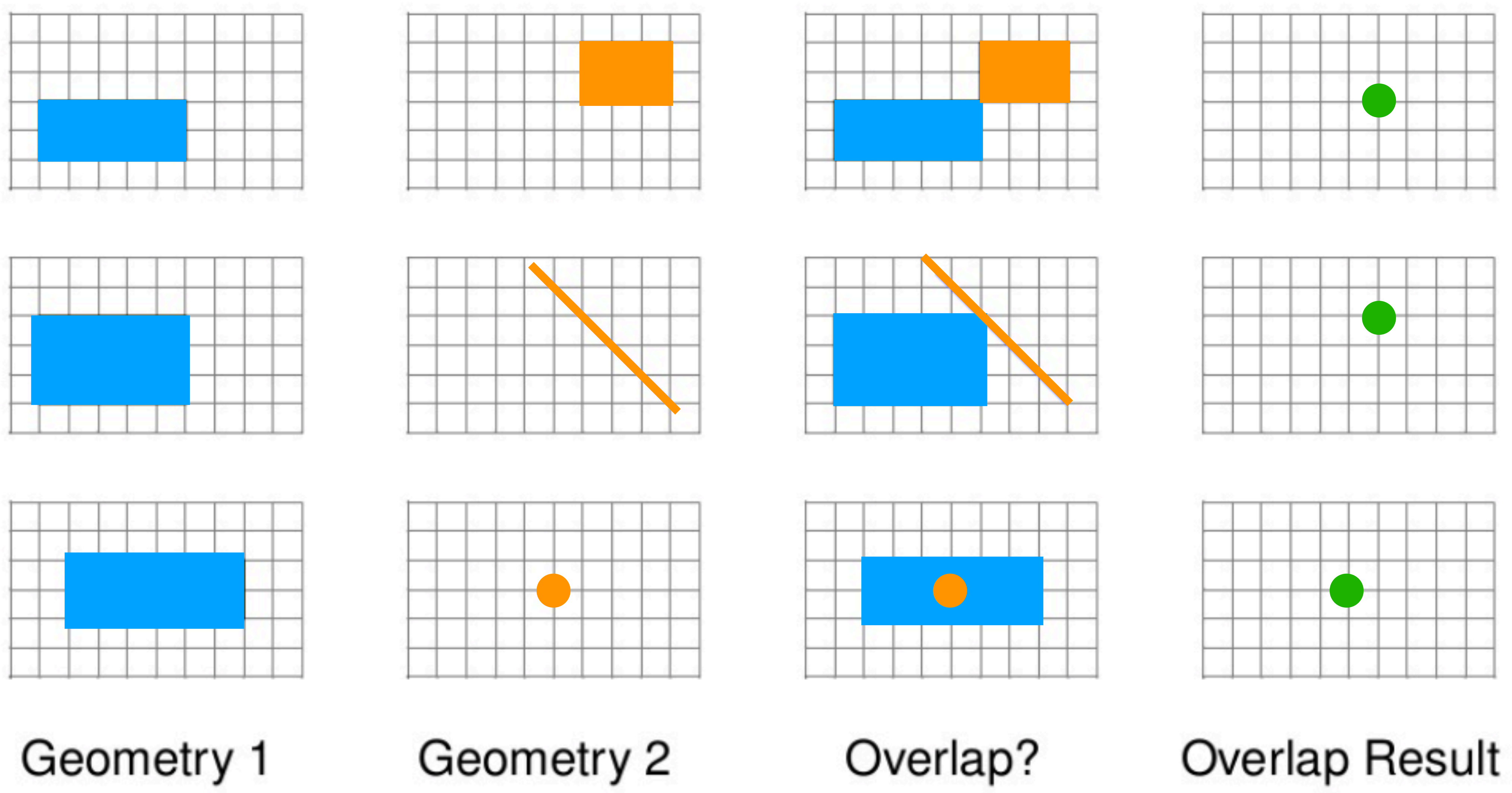
- Set-based spatial relationships
  - Need to consider relationships between lines and areas too
  - **Equal**—is the geometry spatially equal to another?
  - **Intersect/disjoint/overlap**—Do the geometries intersect?
  - **Within**—Is the geometry spatially within another?
  - **Contain**—Does one geometry completely contain another?

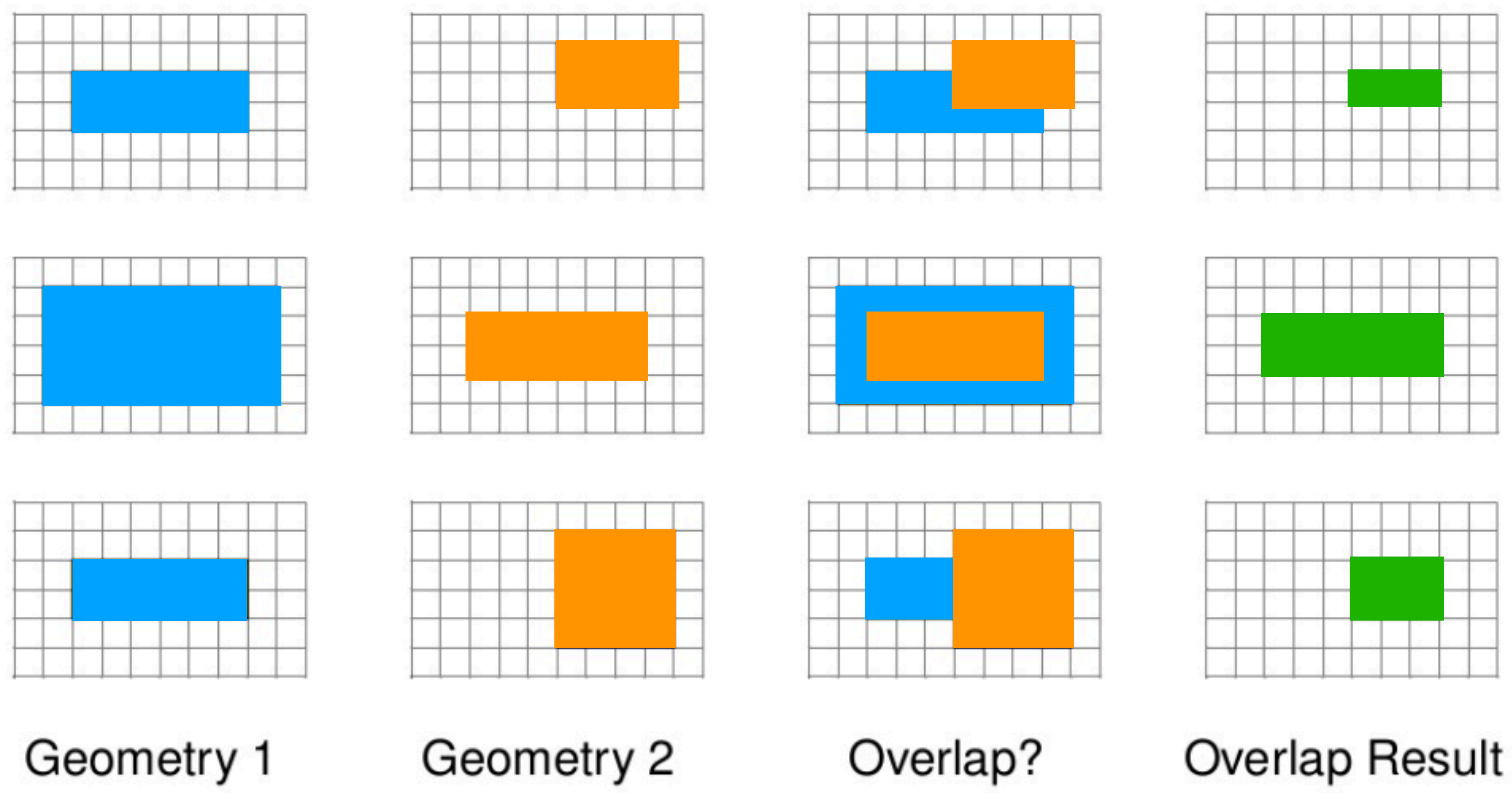
- More spatially-specific:
  - **Touch**—Do the geometries intersect at their boundaries?

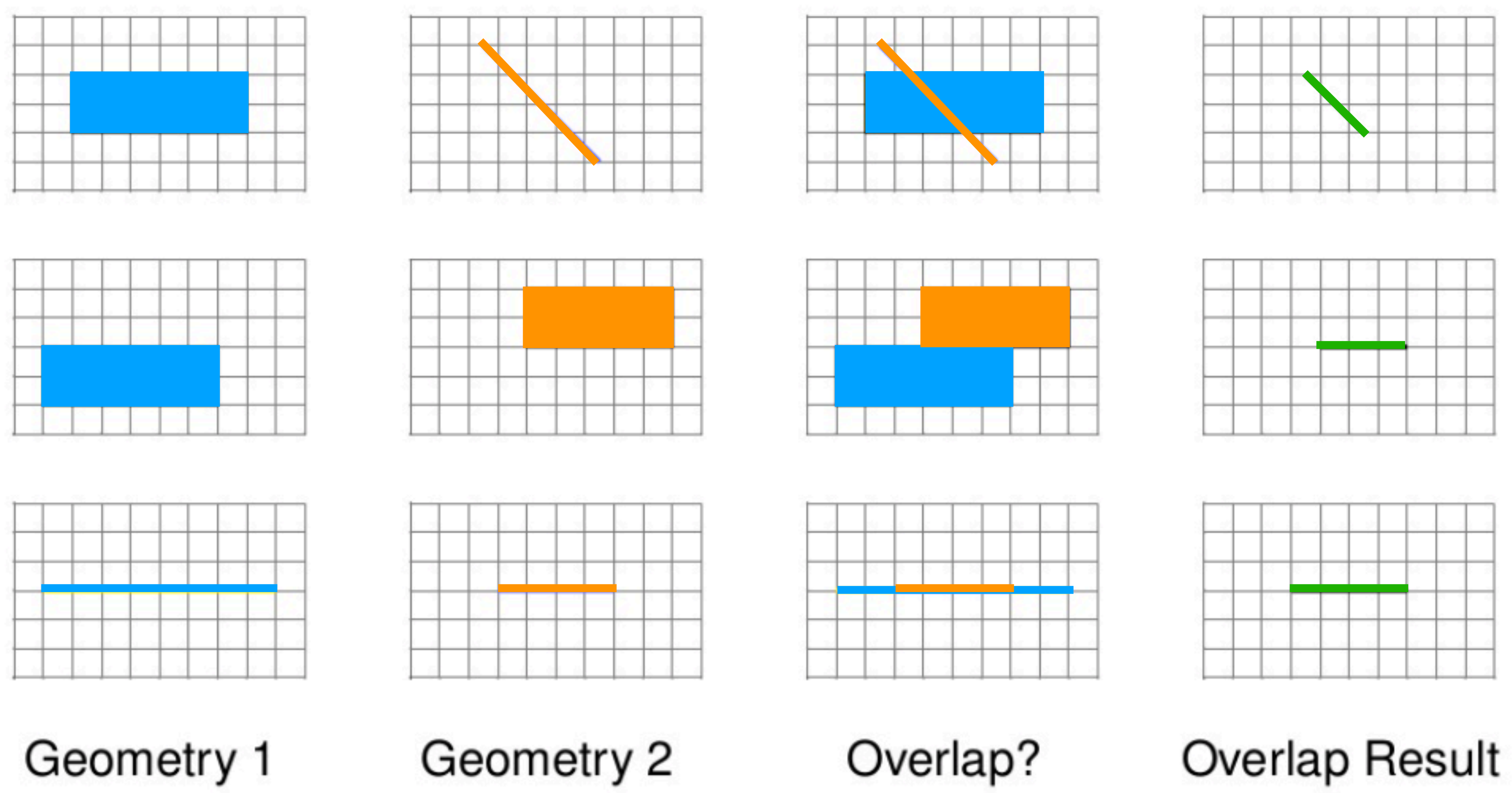
Spatial properties of geometric objects

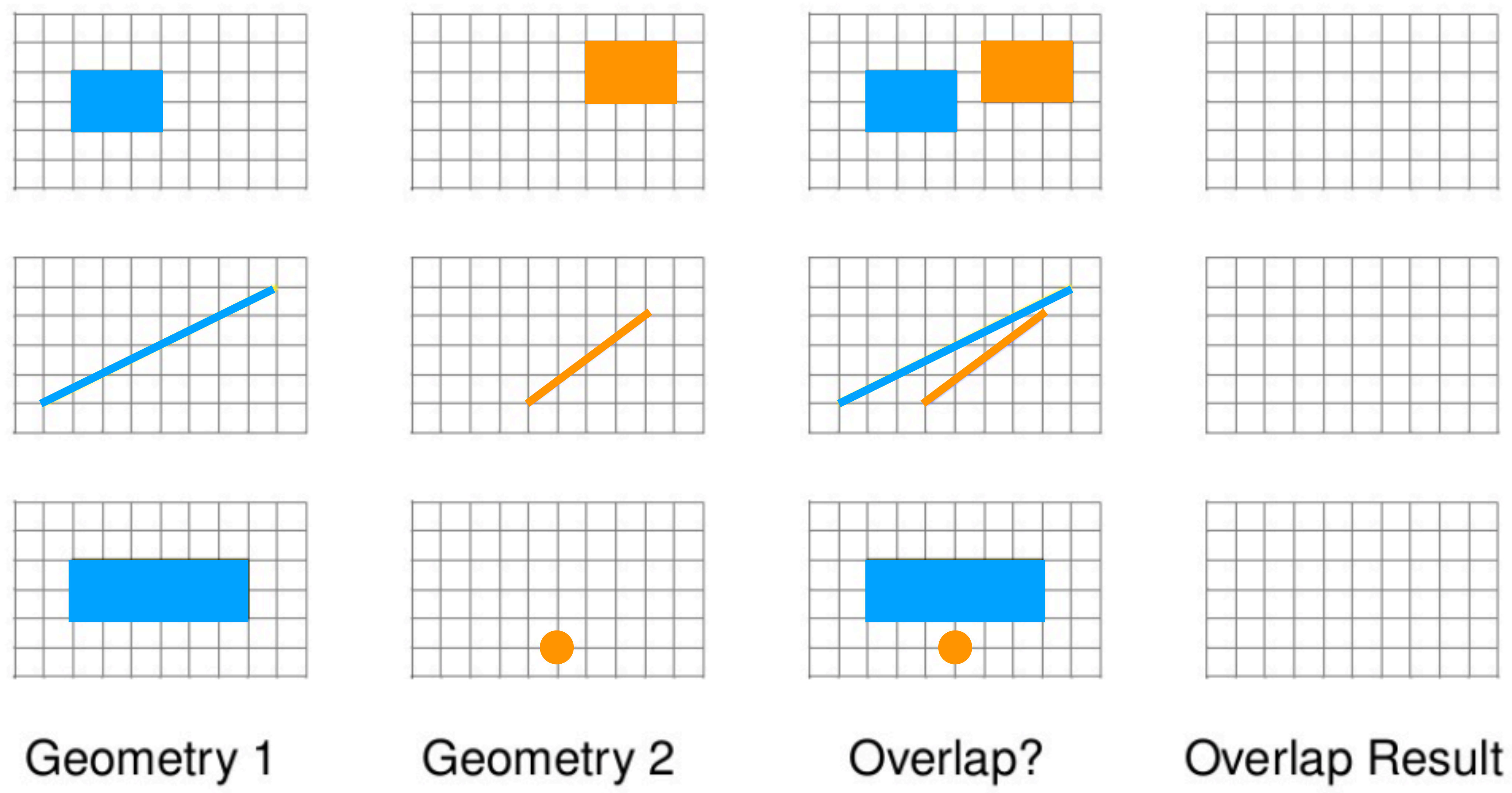


Overlap between geometric objects









Spatial analysis methods (operators)

- Usual set-based operators:
  - **Intersection**—represents point set intersection
  - **Union**—point set union
  - **Difference** (+**symmetric**)—point set difference (+either, not both)

- More spatially specific:
  - **Distance**—shortest distance between any two points in geometries
  - **Buffer**—geometry is expanded from given geometry by a distance
  - **Convex hull**—returns a geometry that tightly contains another geometry, but is convex in shape

Integrity constraints for spatial data

- Topological integrity
  - Ensure that interrelationships between features are maintained
- Semantic integrity
  - Particular use of data model may impose semantic constraints
- User-defined integrity
  - Commercial context: some business rules may apply
- Temporal considerations
  - Spatiotemporal data models may add constraints in time

History—adoption of geospatial capabilities

- Proprietary systems handled data before DBs could
- Integration into models we have discussed:
  - RDBMS systems do not provide suitable domains
  - OODBMS fit better, but were not commercially successful
  - ORDBMS incorporated objects and extended query language
- Example: Oracle Spatial and Graph in Oracle 12c
  - Also: Oracle Locator (cut down), and Oracle MapViewer

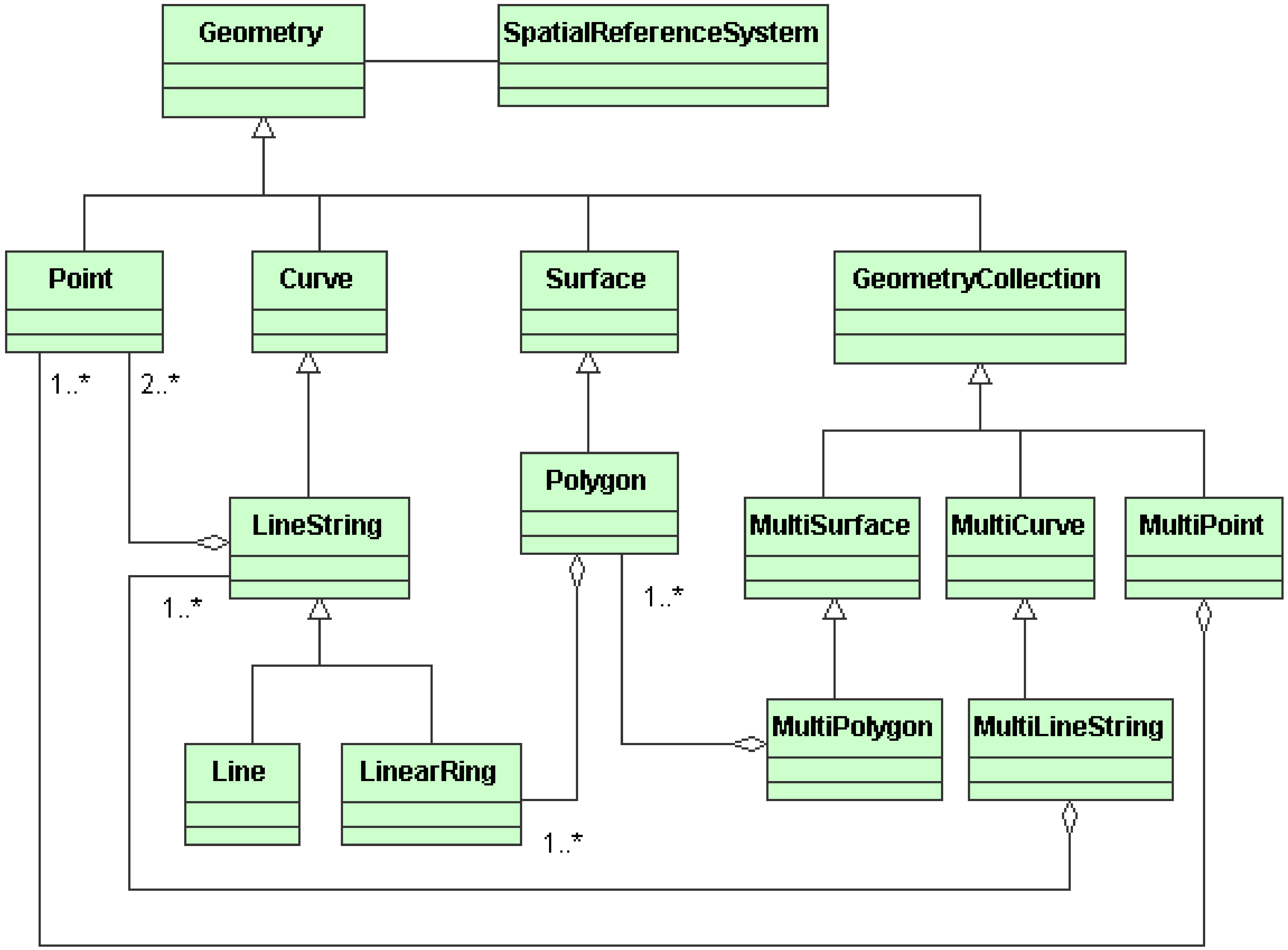
GIS data types are supported by standards

- US Federal Geographic Data Committee
- International Standards Organisation (ISO)
- Open GIS Consortium
  - Geography Markup Language (GML)
  - Provides geometry class hierarchy

- Managing geospatial data can be very challenging!
  - Apparently simple: coordinates in space
  - Agreeing on coordinates is actually far from simple—why?

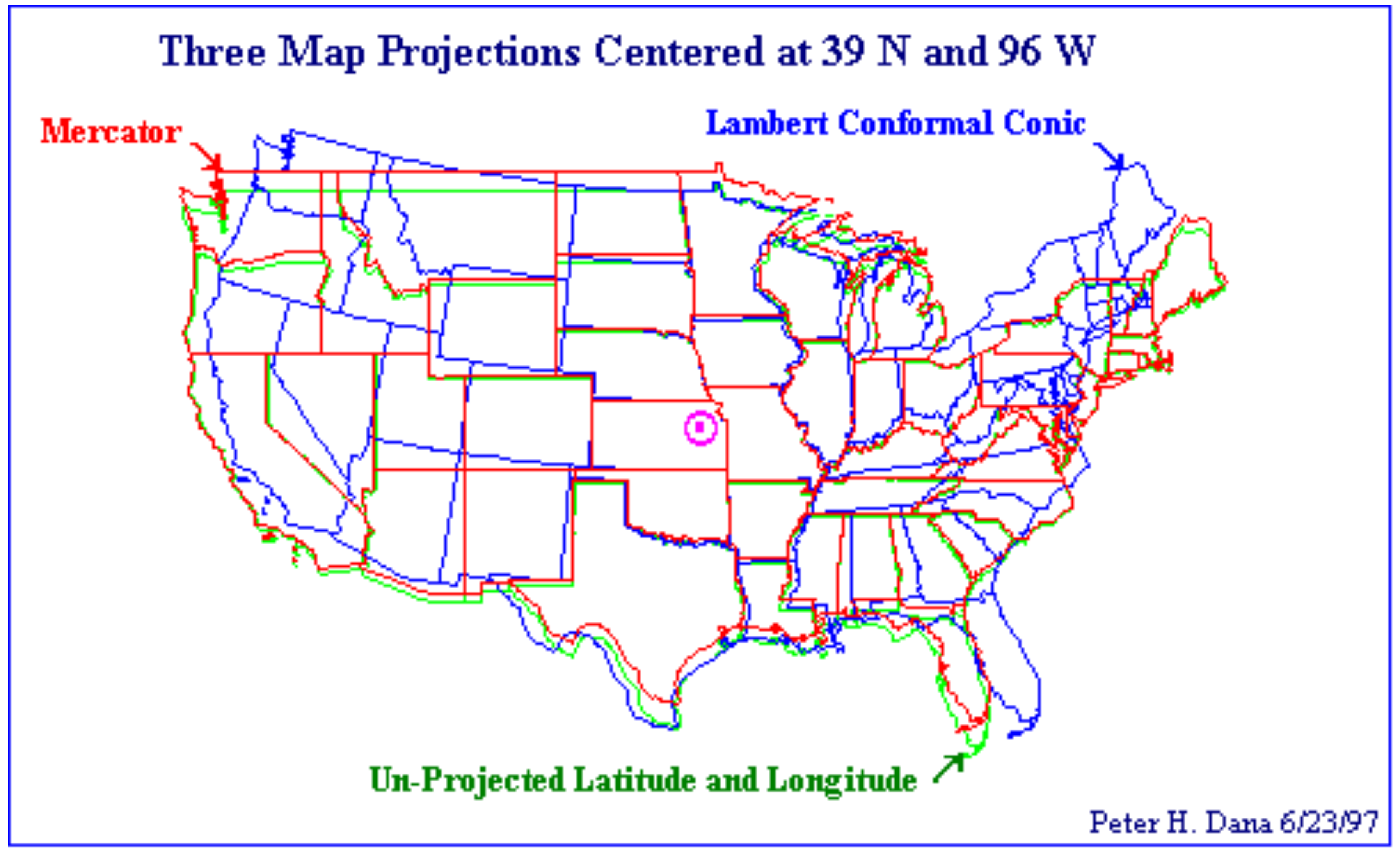
A subset of the OpenGIS GML standard

- Class diagram shown
  - Parent: Geometry
- Note that these classes handle both lines and areas
  - (also note the spatial reference system!)



Spatial reference systems

- RHS: NZ spatial reference systems
  - Top (NZTM2000) is common
  - Bottom four are deprecated
- USA in different spatial reference systems:



New Zealand Transverse Mercator 2000 (NZTM2000) →

New Zealand Transverse Mercator 2000 (NZTM2000) is the projection used for New Zealand's Topo50 1:50,000 and other small scale mapping. Spatial data users are encouraged to use NZTM2000 where a projection is required within mainland New Zealand.

NZGD2000 Meridional Circuits →

The Rules for Cadastral Survey require surveyors to carry out surveys in terms of one of 28 Transverse Mercator meridional circuits. The spatial extents of the circuits are shown in the figure below.

New Zealand Offshore Island Projections →

Transverse Mercator projections have limited longitudinal extents therefore separate projections have been defined for the Antipodes, Auckland, Bounty, Campbell Island / Motu Ihupuku, Chatham, Kermadec, Raoul and Snares Islands / Tini Heke Islands.

New Zealand Continental Shelf Lambert Conformal 2000 (NZCS2000) →

This page contains spatial data covering the New Zealand continental shelf can be represented using the NZCS2000 Lambert Conformal projection.

Ross Sea Region Projections →

This page contains tables that define the key parameters for each Ross Sea Region projection.

New Zealand Map Grid (NZMG) →

This page contains information on the NZMG, which was used for 1:50,000 topographic mapping in New Zealand until 2001, when it was replaced by the NZTM2000.

Chatham Islands Transverse Mercator 1979 (CITM1979) →

The Chatham Islands Transverse Mercator 1979 (CITM1979) is the CIGD1979 projection that has been used for 1:50,000 topographic mapping in the Chatham Islands (the latest editions were published in 1998 with limited revisions).

North and South Island Yard Grids (NIYG, SIYG) →

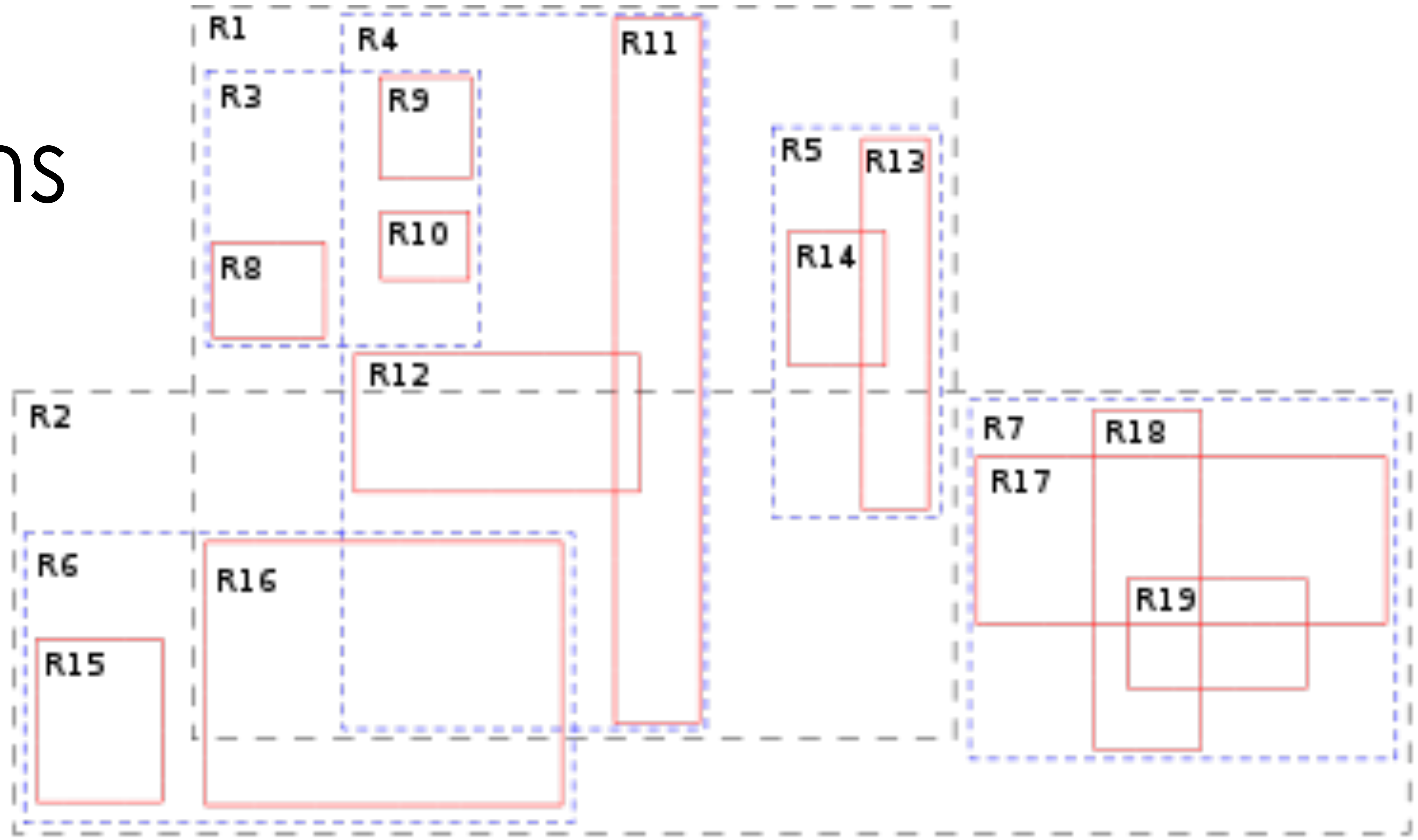
The North and South Island Yard Grids (NIYG; SIYD) are the NZGD1949 projections that were used for the 1:63,360 (inch-to-the-mile) NZMS1 topographic mapping from 1942.

Darwin Glacier Lambert Conformal 2000 Projection (DGLC2000) →

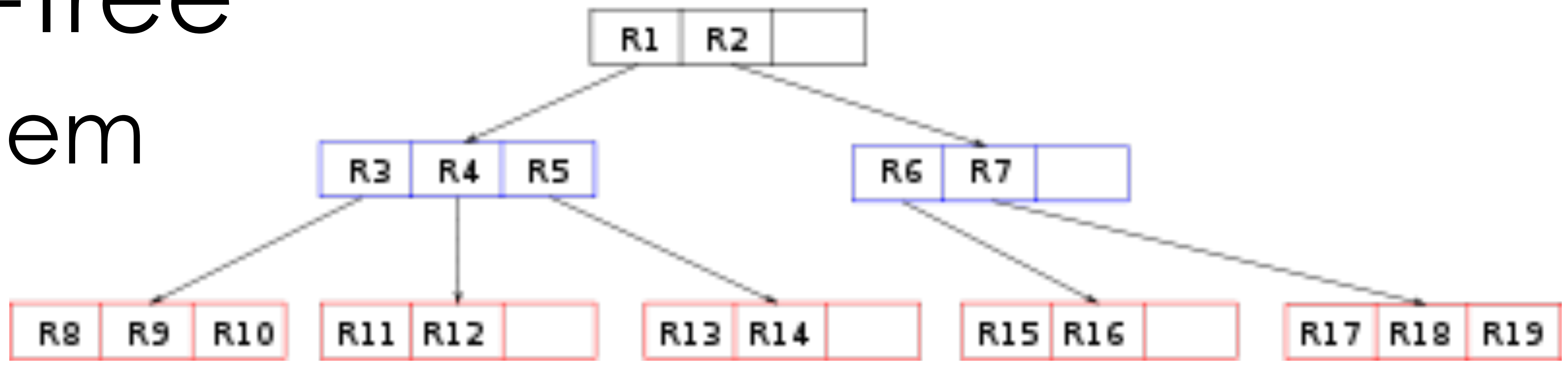
The Darwin Glacier Lambert Conformal 2000 projection (DGLC2000) was replaced by the McMurdo Sound Lambert Conformal 2000 projection (MSLC2000) on 21 March 2011.

Indexing geospatial data with R-Trees

- R-Trees aggregate regions
  - nodes define regions
  - children contained by their parent nodes
  - note siblings may overlap



- Many specific forms of R-Tree
  - … but we won’t explore them



Common applications of geospatial DBs

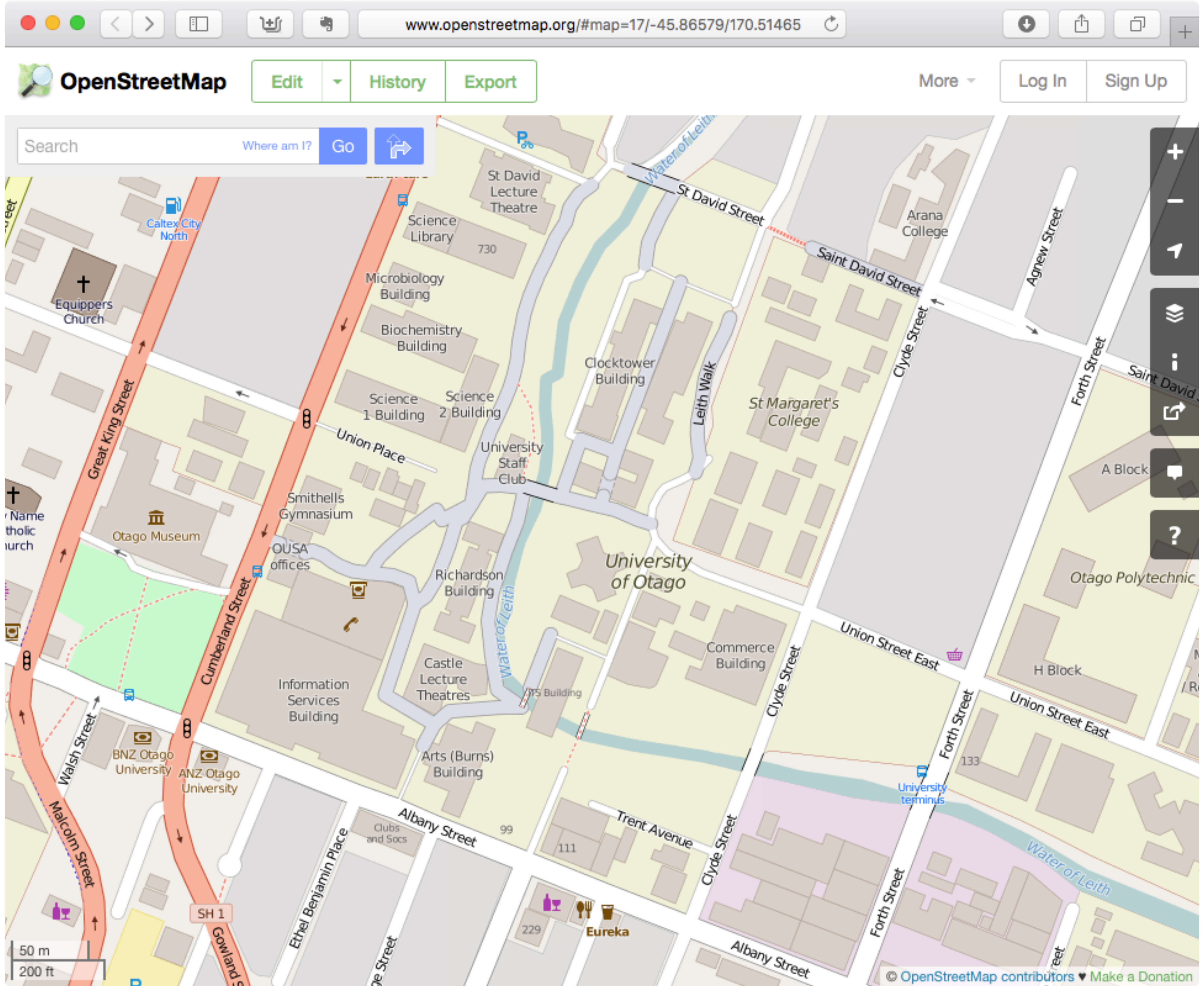
- Discussed briefly earlier. For example, map-focused:
  - OpenStreetMap
  - Google Maps
  - Navigation interfaces (now integrated into map websites)

- Or more focused on decision support systems:
  - City/region planning
  - Transportation and emergency services
  - Scientific data processing

OpenStreetMap

- OpenStreetMap takes wiki idea to mapping
  - Initialised from old maps
  - GPS traces from users
  - Manual editing

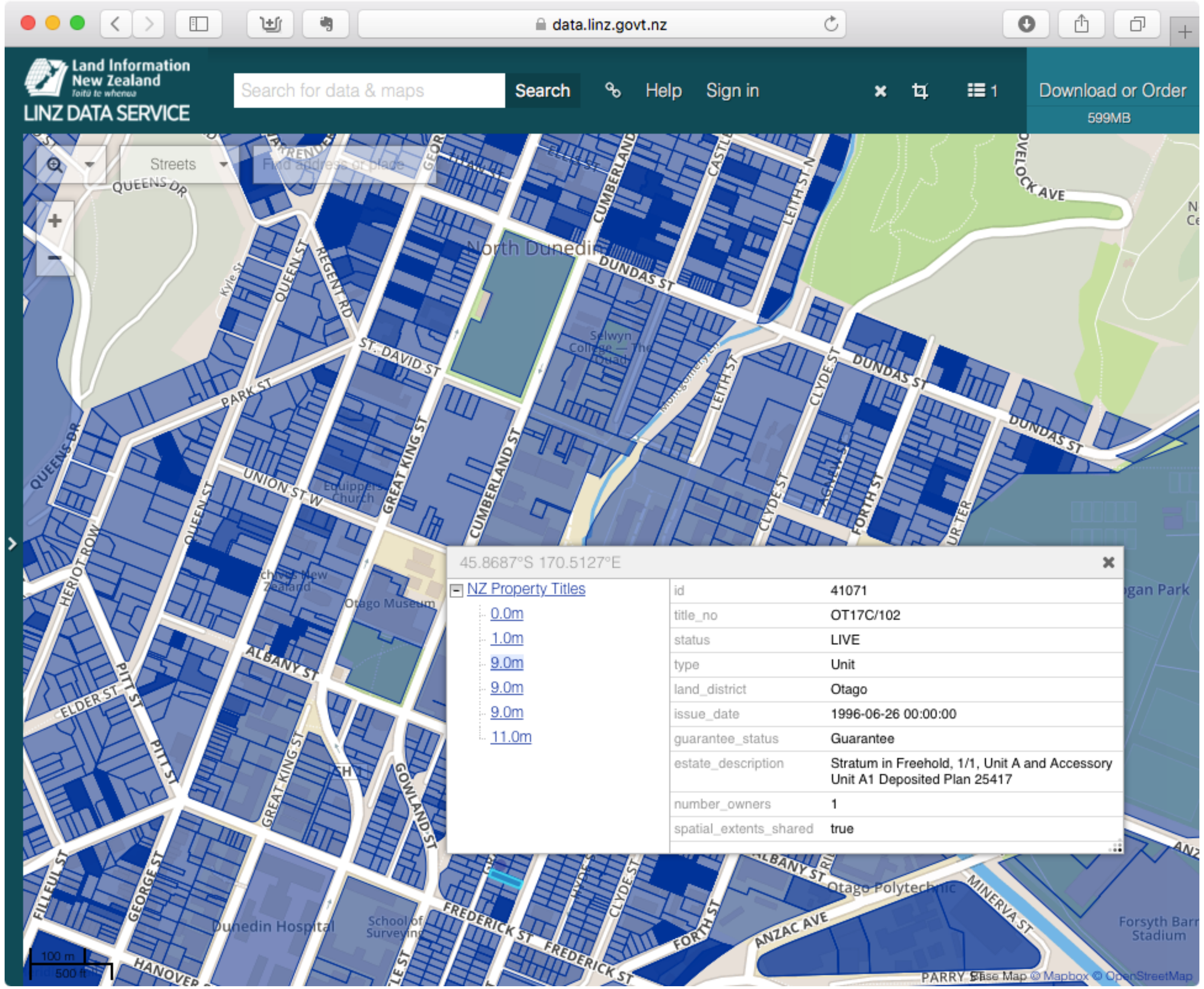
- (Someone clearly likes Dunedin power pylons)



LINZ Data Service

- Vast collection of open government data
  - Title information
  - also aerial imagery
  - maps, etc.

- LINZ data is being added into OpenStreetMap too



Software like GeoServer can host mapping

- Open source software can host maps: e.g., GeoServer
  - Certified as OGC compliant
  - Implemented in Java, available on GitHub
- Compliant with many standards
  - Web Feature Service—for vector geometries of interest
  - Web Map Service—HTTP API for retrieving geo-registered maps
  - Web Coverage Service—handles raster data

PostGIS extends PostgreSQL geospatially

- Compliant with OGC's Simple Features for SQL spec.
- Exchanges data in many different formats:
  - Import data: KML, GML, GeoJSON, GeoHash, etc.
    - KML: Keyhole Markup Language is used by Google Earth
  - Export data: GeoTIFF, NetCDF, PNG, JPEG, etc.
  - Supports ESRI shape file vector data—from ArcGIS, etc.
- Supports 3D objects—often GIS is more 2.5D

MySQL also supports geospatial extensions

- MySQL implements a subset of the OGC Geometry Types environment—*i.e.*, class diagram seen earlier
- MySQL is very widely deployed
  - Simple applications may be suited to its geospatial extensions
  - Historically significantly lagged PostGIS in speed and power
- (Oracle) MySQL geospatial competes with Oracle…

Summary—geospatial databases

- GIS systems address a variety of use cases
- There are numerous different spatial data models
  - Different models suit different applications
  - (O)RDBMSs often extended to handle spatial data
- Many geometric query operators
  - Significantly generalises temporal DB operators
- Open data and open source DBs facilitate building mapping applications more easily than ever before

Readings—for more information

- Temporal databases:
  - James F. Allen. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM* 26, 11, pp 832–843. DOI: http://dx.doi.org/10.1145/182.358434
- Geospatial databases:
  - Ralf Hartmut Güting. 1994. An introduction to spatial database systems. *The VLDB Journal*, 3(4), pp 357–399. http://dna.fernuni-hagen.de/papers/IntroSpatialDBMS.pdf
  - PostGIS documentation—adds PostgreSQL geospatial support