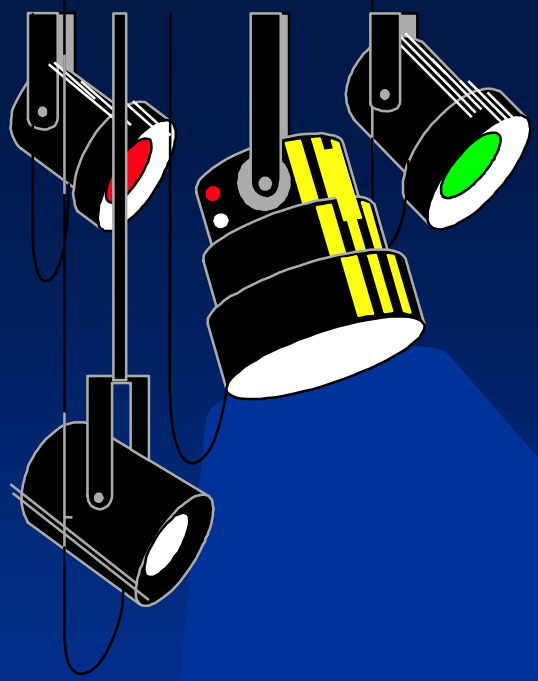


Outline

- Spatial Databases
  - Theme
  - Map
  - Geographic objects
  - Modeling geographic data
- Temporal Databases
  - Transaction time, valid time
  - Anchored time, instance, interval
  - Modeling temporal data

- **Spatial data management**
  - A spatial database is a data management system for the collection, storage, manipulation and output of spatially referenced information.
  - Theme: refers to data describing a particular topic (e.g., scenic lookouts, rivers, cities) and is the spatial counterpart of an entity type.
    When a theme is presented on a screen or paper, it is commonly seen in conjunction with a *map*. Color may be used to indicate different themes (e.g., blue for rivers and black for roads).
  - Map: A map will usually have a scale, legend, and possibly some explanatory text.

- **Spatial data management**
  - Geographic objects: A geographic object is an instance of a theme (e.g., a river).
    - attributes
    - spatial components: geometry and topology

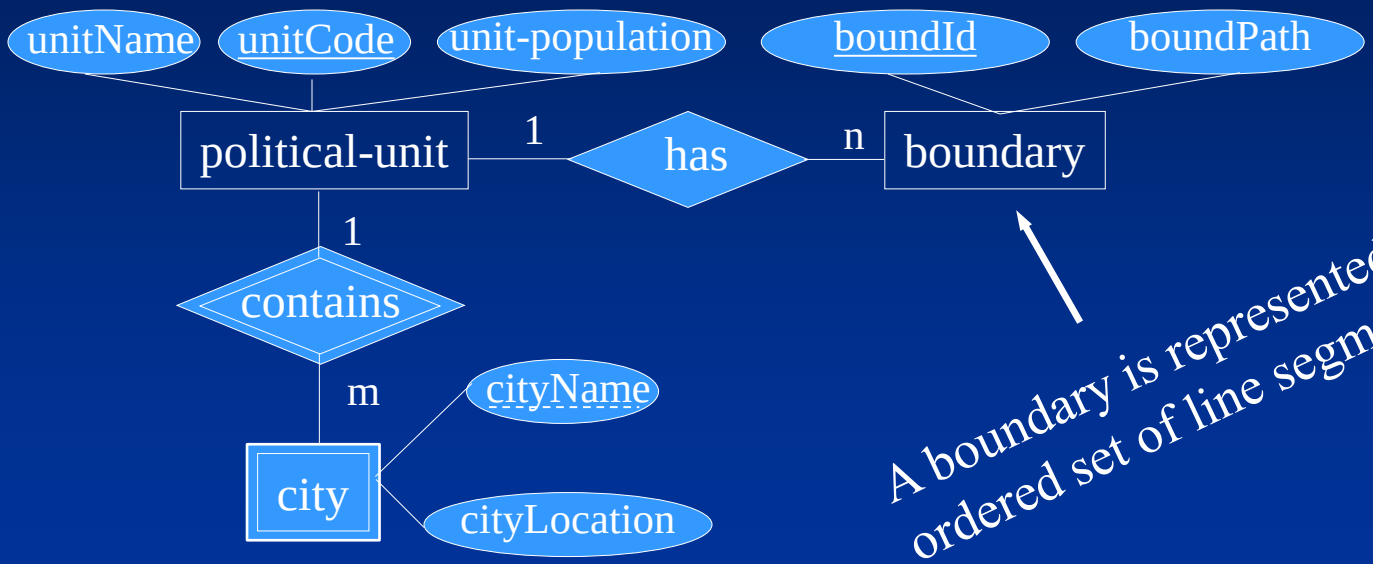
Geometry refers to the location-based data: shape, length

Topology refers to spatial relationships among objects: adjacency

- **Spatial data management**
  - Data type for spatial elements: points, lines and regions

Data type	Dimensions	Example
Point	0	Scenic lookout
Line	1	River
Region	2	County

A database for political units:

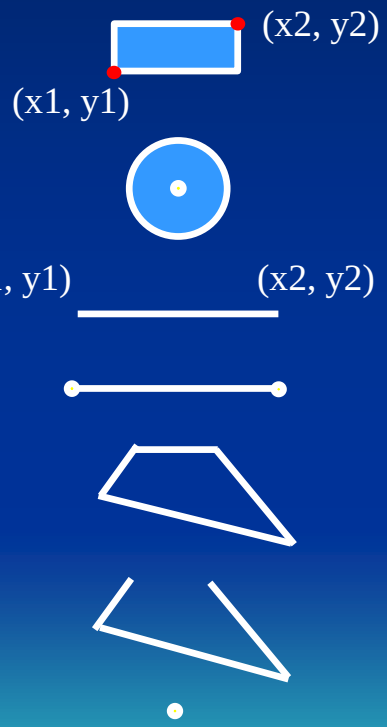


Example: Canada has a boundary for the continental portion, and each of its sovereign islands, such as Prince Edward Island.

Map a geographic data model to tables:

Geometric data type in PostgreSQL:

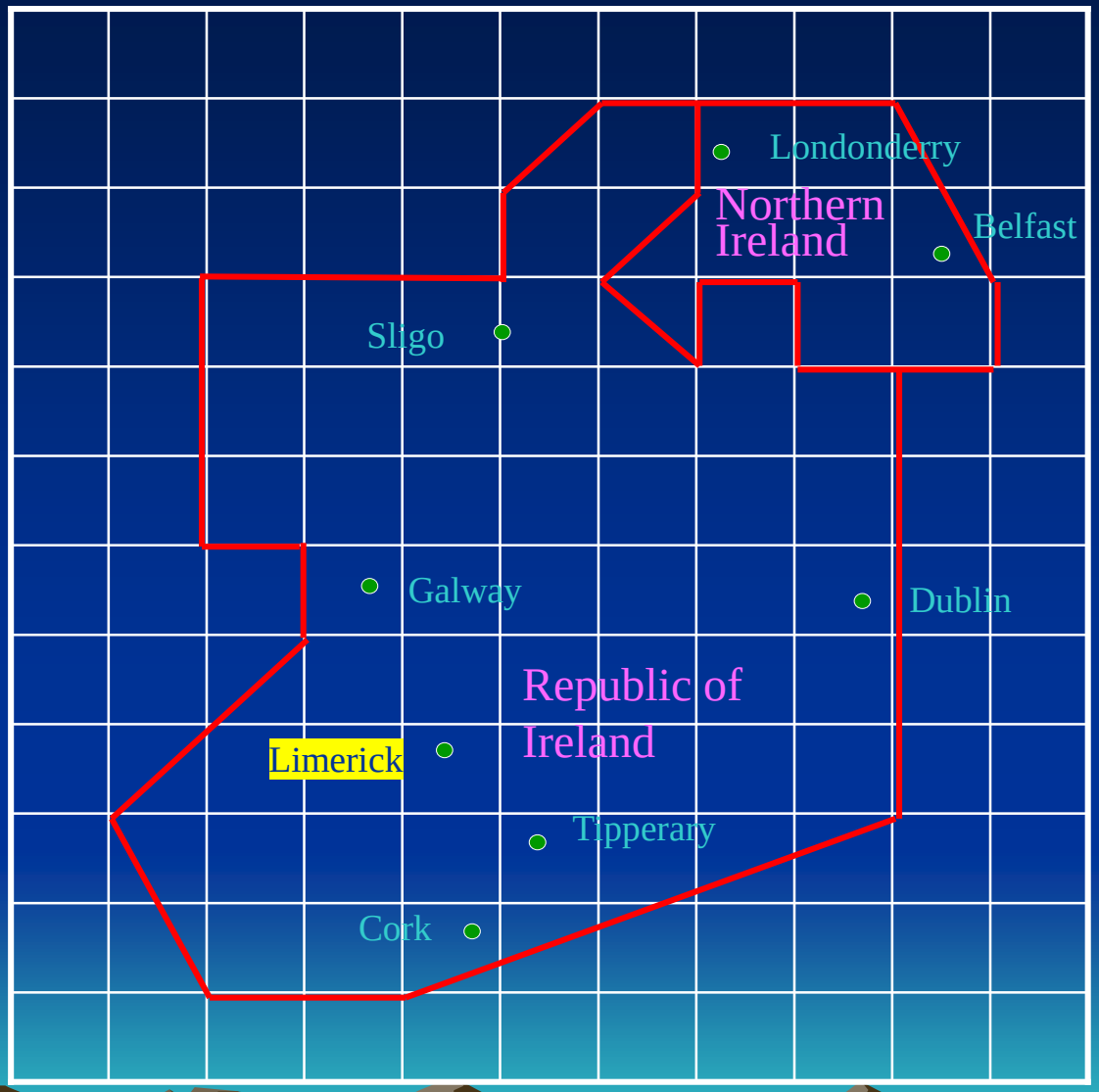
Geometric type	Representation	Description
BOX	((x1, y1), (x2, y2))	Rectangular box
CIRCLE	<(x, y), r>	Circle (center and radius)
LINE	((x1, y1), (x2, y2))	Infinite line
LSEG	[(x1, y1), (x2, y2)]	Finite line segment
PATH	((x1, y1), …)	Closed path (similar to polygon)
PATH	[(x1, y1), …]	Open path
POINT	(x, y)	Point in space
POLYGON	((x1, y1), …)	Polygon (similar to closed path)



Map a geographic data model to tables:

Table definition for Political unit data model:

```
CREATE TABLE political_unit (
    unitname        VARCHAR(30)     NOT NULL,
    unitcode        CHAR(2),
    unitpop         DECIMAL(6, 2),
    PRIMARY KEY (unitcode));
CREATE TABLE boundary (
    boundid         INTEGER,
    boundpath       PATH            NOT NULL,
    unitcode        CHAR(2),
    PRIMARY KEY (boundid),
    CONSTRAINT fk_boundary_polunit FOREIGN KEY (unitcode) REFERENCES
    political_unit);
CREATE TABLE city (
    cityname        VARCHAR(30),
    cityloc         POINT           NOT NULL,
    unitcode        CHAR(2),
    PRIMARY KEY (unitcode, cityname),
    CONSTRAINT fk_city_polunit FOREIGN KEY (unitcode) REFERENCES political_unit);
```


Londonderry
Northern Ireland
Belfast
Sligo
Galway
Dublin
Republic of Ireland
Limerick
Tipperary
Cork

Insert statements for populating database

```
INSERT INTO political_unit VALUES ('Republic of Ireland', 'ie', 3.9);
INSERT INTO political_unit VALUES ('Northern Ireland', 'ni', 1.7);
INSERT INTO boundary VALUES
    (1, '[(9, 8), (9, 3), (4, 1), (2, 2), (1, 3), (3, 5), (3, 6), (2, 6),
    (2, 9), (5, 9), (5, 10), (6, 11), (7, 11), (7, 10), (6, 9), (7, 8),
    (7, 9), (8, 9), (8, 8), (9, 8)]', 'ie');
INSERT INTO boundary VALUES
    (2, '[(7, 11), (9, 11), (10, 9), (10, 8), (8, 8), (8, 9), (7, 9),
    (7, 8), (6, 9), (7, 10), (7, 11)]', 'ni');
INSERT INTO city VALUES ('Dublin', '(9, 6)', 'ie');
INSERT INTO city VALUES ('Cork', '(5, 2)', 'ie');
INSERT INTO city VALUES ('Limerick', '(4, 4)', 'ie');
INSERT INTO city VALUES ('Galway', '(4, 6)', 'ie');
INSERT INTO city VALUES ('Sligo', '(9, 6)', 'ie');
INSERT INTO city VALUES ('Tipperary', '(5, 3)', 'ie');
INSERT INTO city VALUES ('Belfast', '(9, 9)', 'ni');
INSERT INTO city VALUES ('Londonderry', '(7, 10)', 'ni');
```

Geometric functions and operators in PostgreSQL for processing spatial data

Functions:

Function	Returns	Description
LENGTH(OBJECT)	double precision	length of item
NPOINTS(PATH)	integer	Number of points

Operators:

Operator	Description
<->	Distance between
<<	Is left of?
<^	Is below?
>>	Is right of?
>^	Is above?

Queries:

1. What is the length of the Republic of Ireland border?

```
SELECT SUM(LENGTH(boundpath)) * 37.5
     AS “Border (kms)” FROM Political_unit, boundary
WHERE unitname = “Republic of Ireland”
AND political_unit.unitcode = boundary.unitcode;
```

Border (kms)
1353.99

Queries:

2. How far, as the crow flies, is it from Sligo to Dublin?

```
SELECT (orig.cityloc<->dest.cityloc) * 37.5
      AS “Distance (kms)”
FROM city orig, city dest
WHERE orig.cityname = ‘Sligo’
AND dest.cityname = ‘Dublin’;
```

Distance (kms)
167.71

Queries:

3. What is the closest city to Limerick?

```
SELECT dest.cityname FROM city orig, city dest
WHERE orig.cityname = 'Limerick'
AND orig.cityloc <-> dest.cityloc =
(SELECT MIN(orig1.cityloc<->dest1.cityloc)
FROM city orig1, city dest1
WHERE orig1.cityname = 'Limerick' AND
                dest1.cityname <> 'Limerick');
```

cityname
Tipperary

Queries:

4. What is the westernmost city?

```
SELECT west.cityname FROM city west
WHERE NOT EXISTS
    (SELECT * FROM city other
    WHERE other.cityloc << west.cityloc);
```

cityname
Limerick
Galway

Queries:

5. What is the westernmost city in the Republic of Ireland?

```
SELECT west.cityname FROM city west, political_unit p
WHERE west.unitcode = p.unitcode
      and p.unitname = 'Republic of Ireland'
      and NOT EXISTS
            (SELECT * FROM city other
            WHERE other.cityloc << west.cityloc
                  and other.unitcode = p.unitcode
                  and other.unitname = 'Republic of Ireland')
```

cityname
Limerick
Galway

Queries:

6. What is the westernmost city in the Northern Ireland?

```
SELECT west.cityname FROM city west, political_unit p
WHERE west.unitcode = p.unitcode
      and p.unitname = 'Northern Ireland'
      and NOT EXISTS
            (SELECT * FROM city other
            WHERE other.cityloc << west.cityloc
                  and other.unitcode = p.unitcode
                  and other.unitname = 'Northern Ireland')
```

cityname
Londonderry

Queries:

7. What is the westernmost city in Ireland?

```
SELECT west.cityname FROM city west, political_unit p
WHERE west.unitcode = p.unitcode
      and (p.unitname = 'Republic of Ireland' or p.unitname ='Northern Ireland')
      and NOT EXISTS
            (SELECT * FROM city other
            WHERE other.cityloc << west.cityloc)
                  and other.unitcode = p.unitcode
                  and (other.unitname = 'Republic of Ireland'
                  or other.unitname = 'Northern Ireland')
```

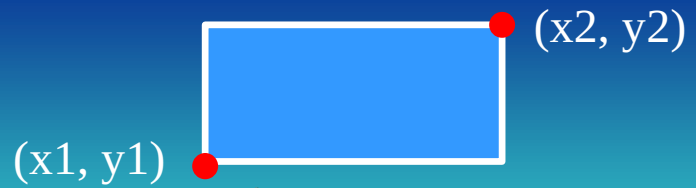
cityname
Limerick
Galway

R-tree:

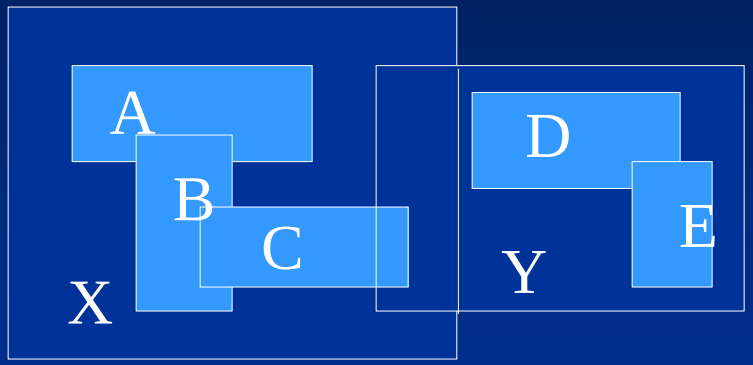
- A B-tree, often used to store data in one-dimensional database, can be extended to n dimensions, where $n \geq 2$.
- An R-tree is the extension of a B-tree.
  Each internal node of an R-tree is a tuple of the following form:

$$p_1, v_1, p_2, v_2, \ldots, p_{m-1}, v_{m-1}, p_m$$

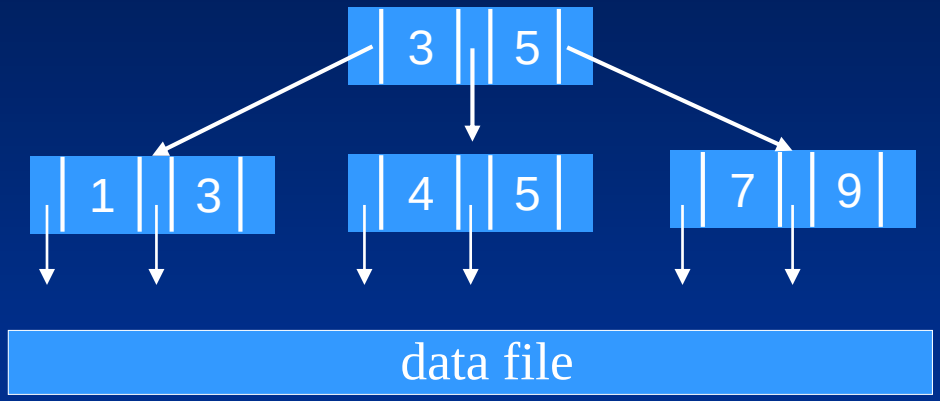
where p_i ($i = 1, \ldots, m$) is a pointer and v_j ($j = 1, \ldots, m-1$) is a pair: <(x1, y1), (x2, y2)>, where x1 and y1 are the coordinates of the lower-left corner of the minimum boundary rectangle, the smallest possible rectangle enclosing an object; and x2 and y2 are for the upper-right corner.

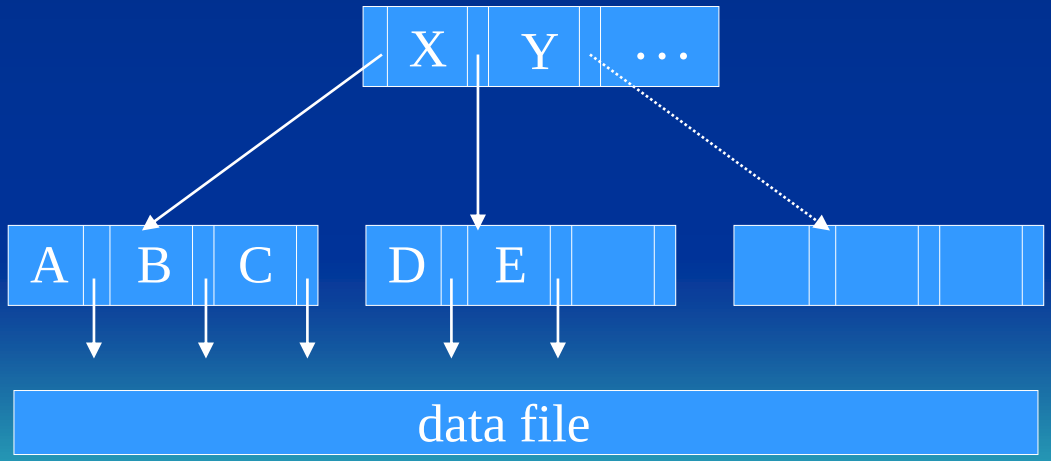


R-tree:



B-tree:

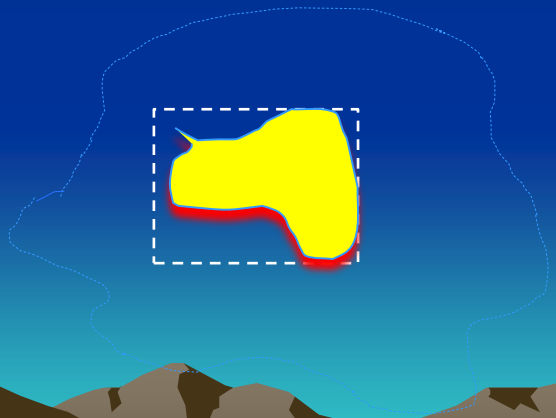




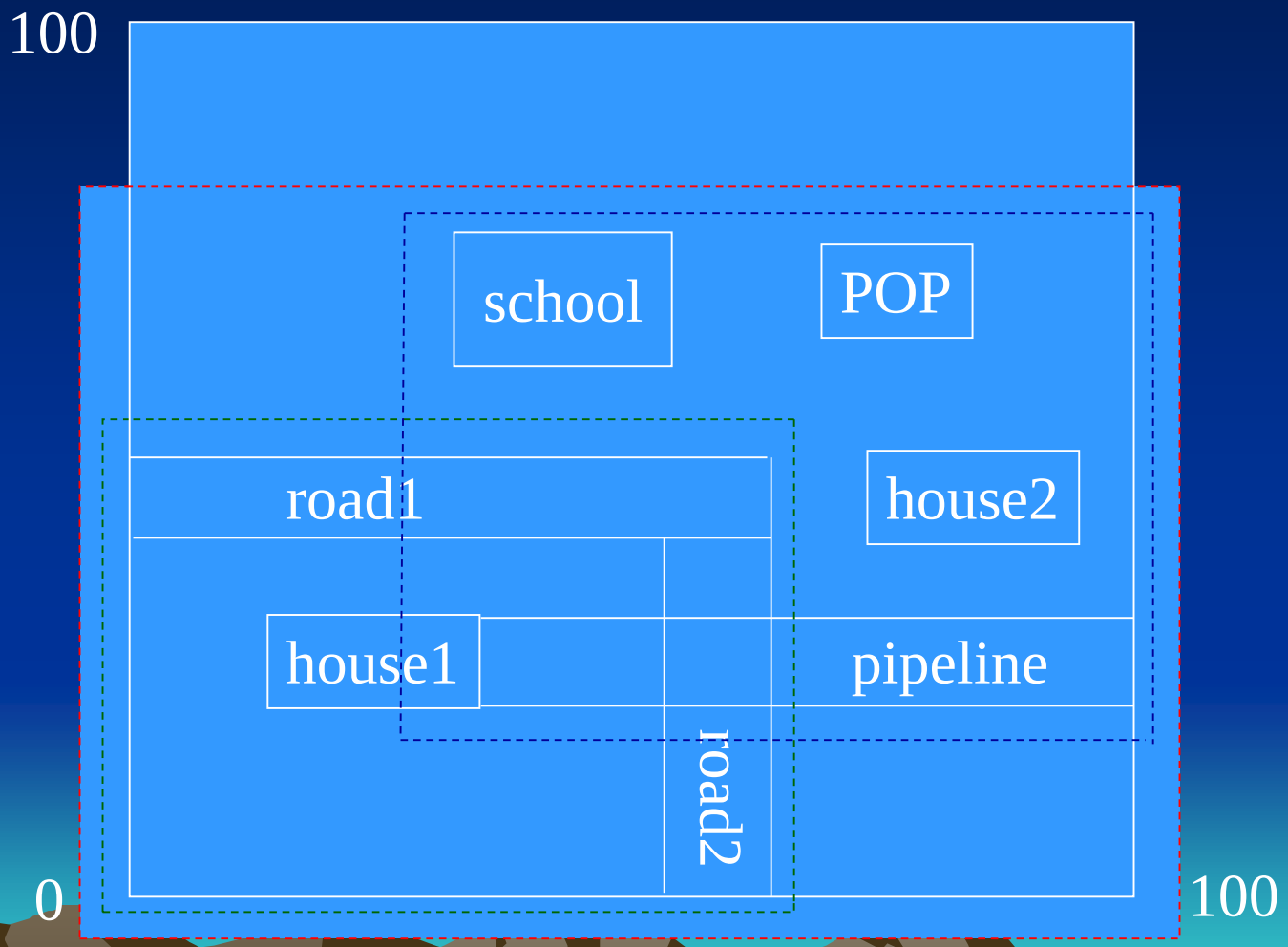
R-tree:

- How is an R-tree used to accelerate searching?

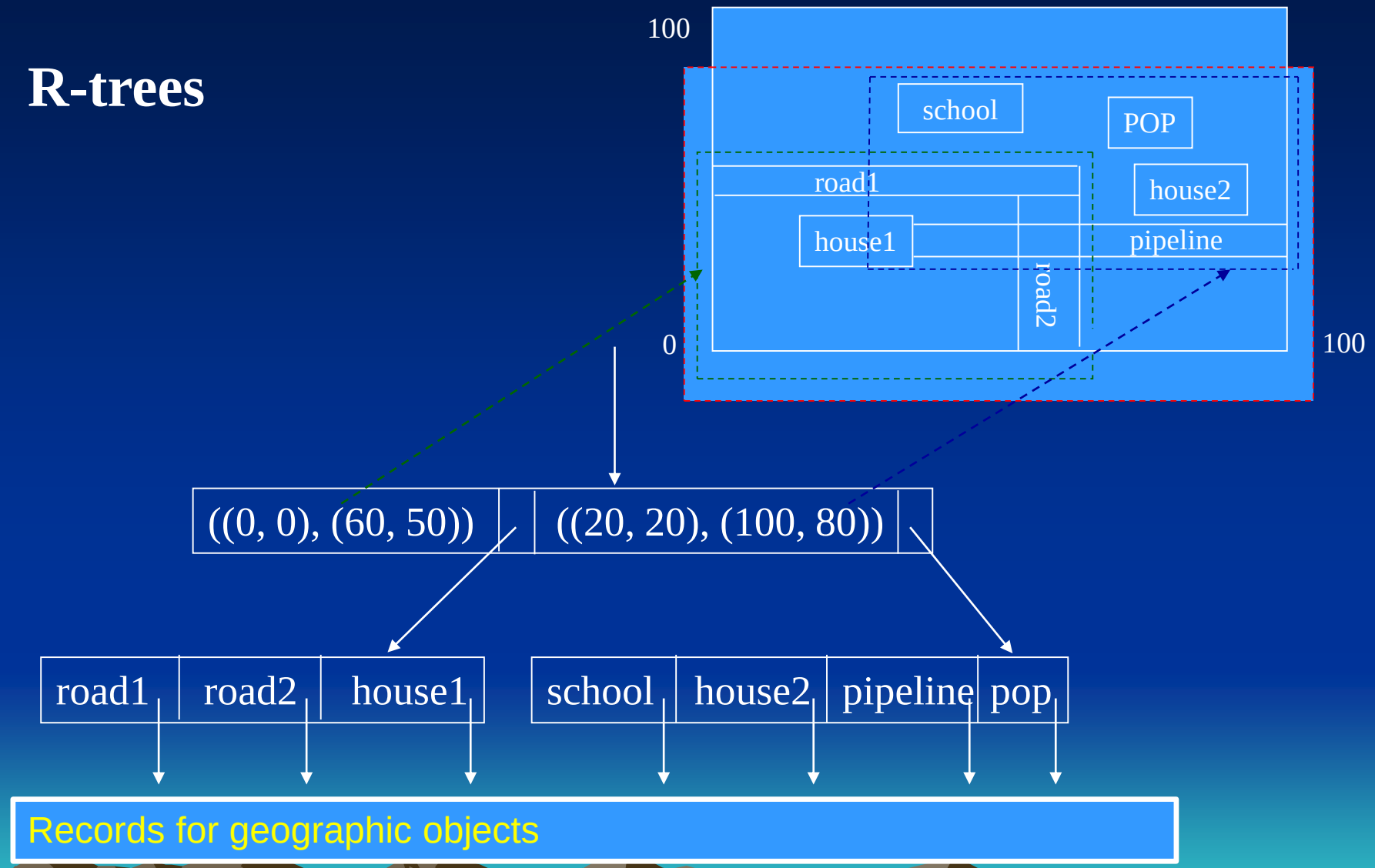
  Using a mouse, a user could outline a region on a map displayed on a screen. The minimum bounding rectangle for this region would then be calculated and the coordinates used to locate geographic objects falling within the minimum boundary, which would be used to search the R-tree along a path.



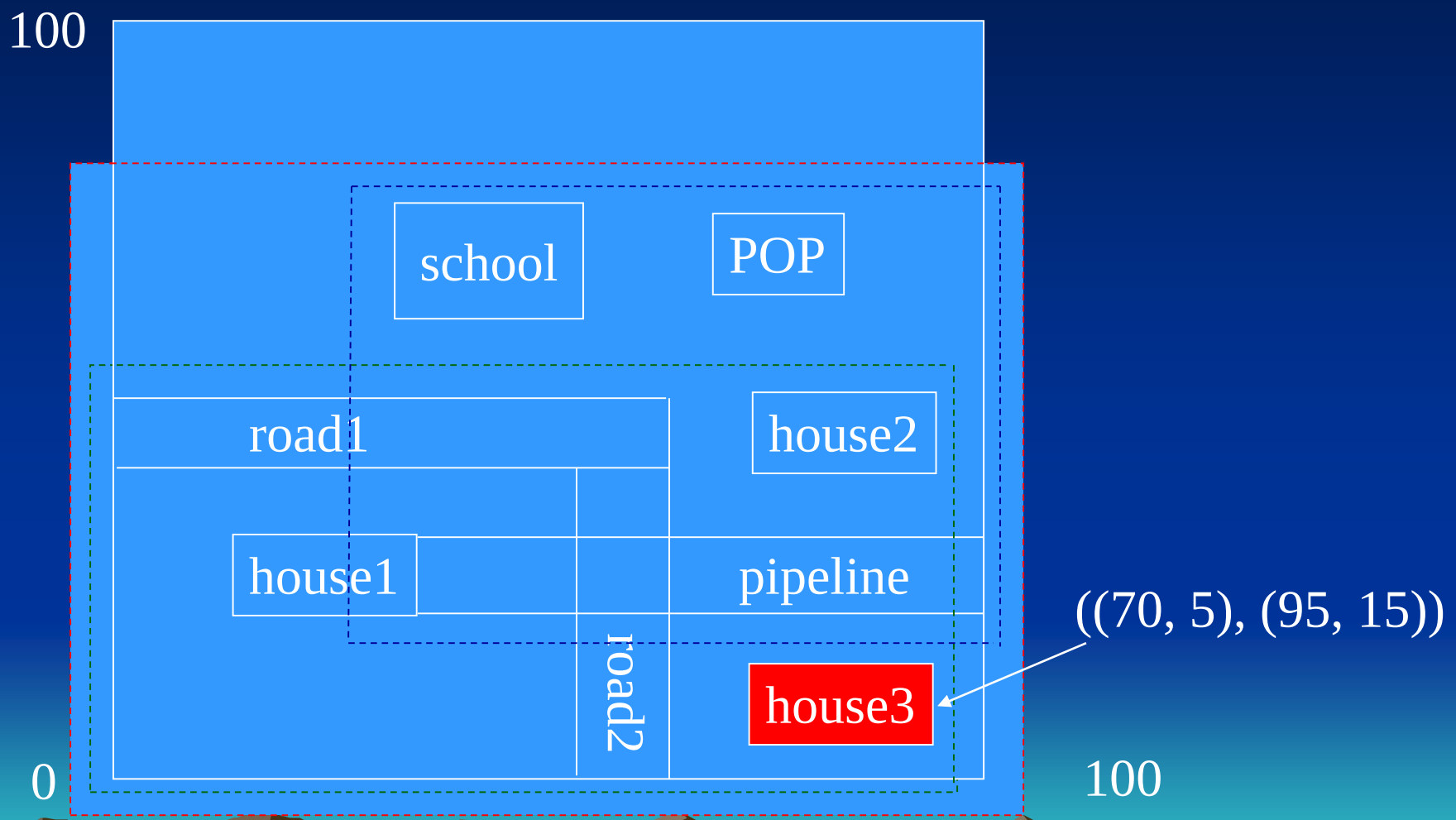
Suppose that the local cellular phone company adds a POP (point of presence, or base station) at the position shown below.



R-trees


100
school
POP
road1
house2
house1
pipeline
road2
0
100
((0, 0), (60, 50))
((20, 20), (100, 80))
road1
road2
house1
school
house2
pipeline
pop
Records for geographic objects

Insert a new region *r* into an R-tree.

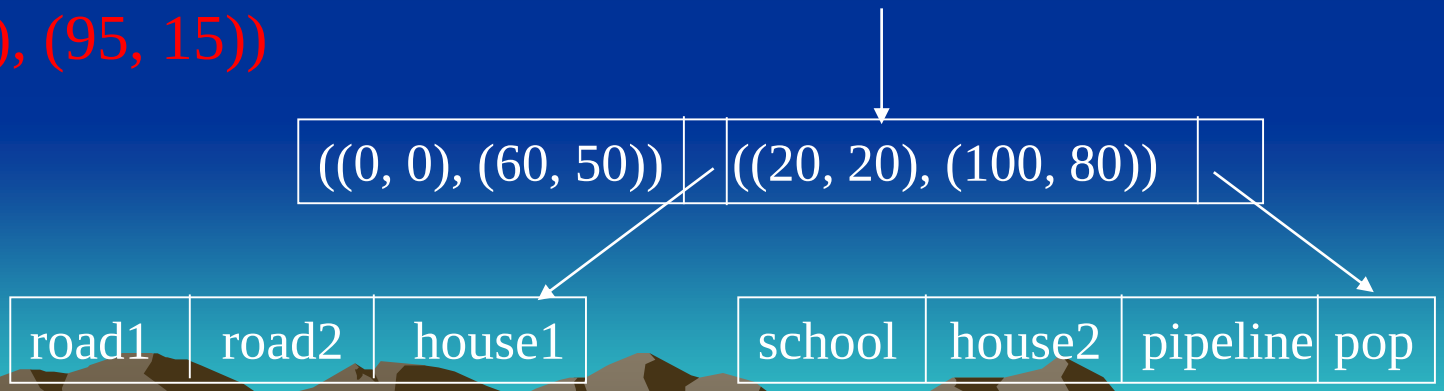


Insert a new region *r* into an R-tree.

1. Search the *R*-tree, starting at the root.
2. If the encountered node is internal, find a subregion into which *r* fits.

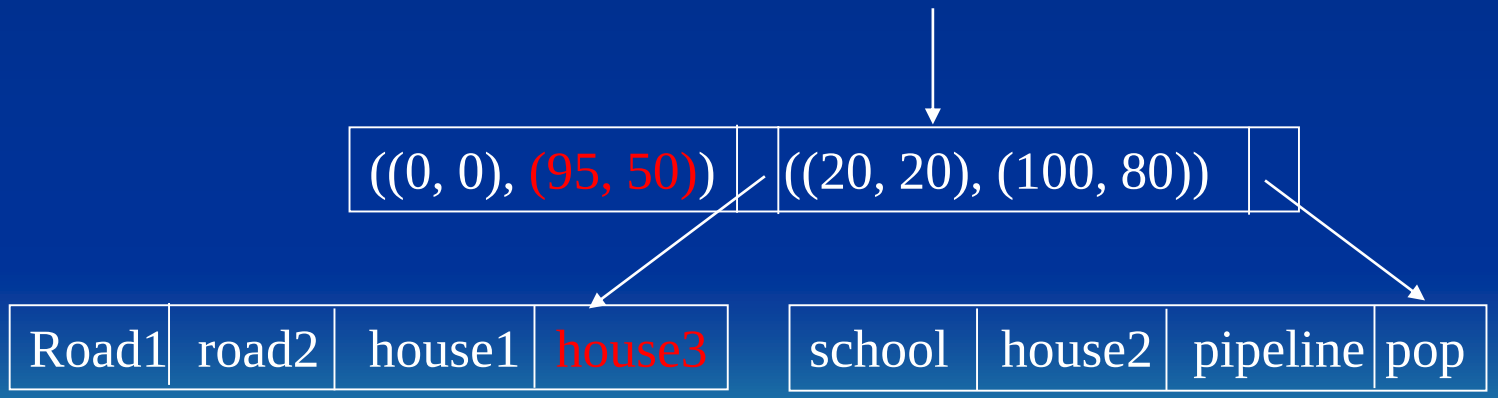
- If there is more than one such region, pick one and go to its corresponding child.
- If there is no subregion that contains *r*, choose any subregion such that it needs to be expanded as little as possible to contain *r*.

((70, 5), (95, 15))



Two choices:

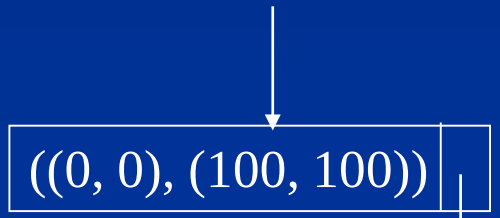
- If we expand the lower subregion, corresponding to the first leaf, then we add 1000 square units to the region.
- If we extend the other subregion by lowering its bottom by 5 units, then we add 1200 square units.



Insert a new region *r* into an R-tree.

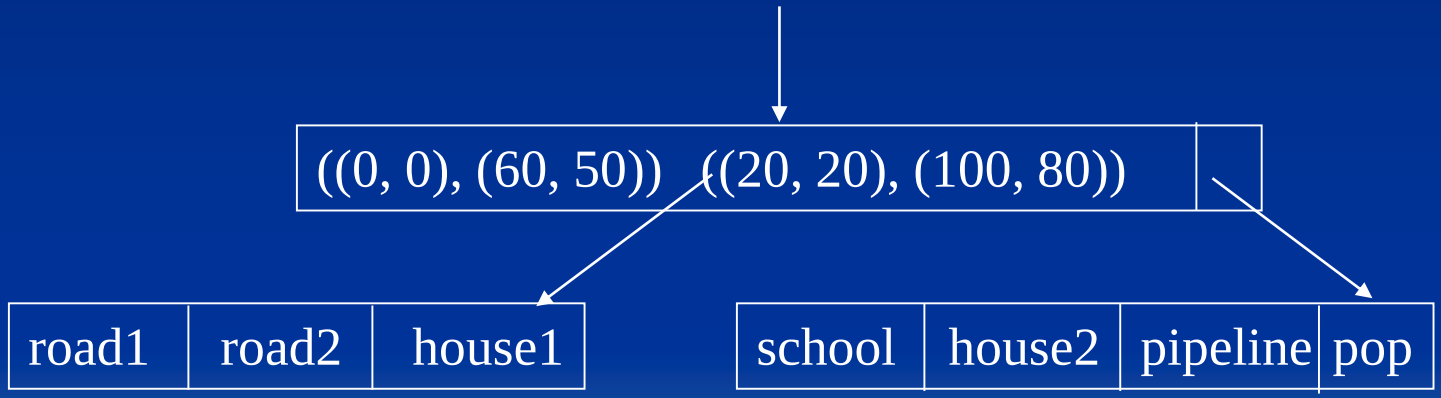
3.If the encountered node *v* is a leaf, insert *r* into it. If there is no room for *r*, split the leaf into two and distribute all subregions in them as evenly as possible. Calculate the 'parent' regions for the new leaf nodes and insert them into *v*'s parent. If there is the room at *v*'s parent, we are done. Otherwise, we recursively split nodes going up the tree.

Suppose that each leaf has room for 6 regions.



Add POP (point of presence, or base station)

- Split the leaf into two and distribute all the regions evenly.
- Calculate two new regions each covering a leaf.



- **Managing temporal data**

  - With a temporal database, stored data have an associated time period indicating when the item was valid or stored in the database.
  - Transaction time: the timestamp applied by the system when data are entered and cannot be changed by an application. It can be applied to a particular item or row.
    For example, the old and new price of a product would automatically have separate timestamps when they are entered into the database.

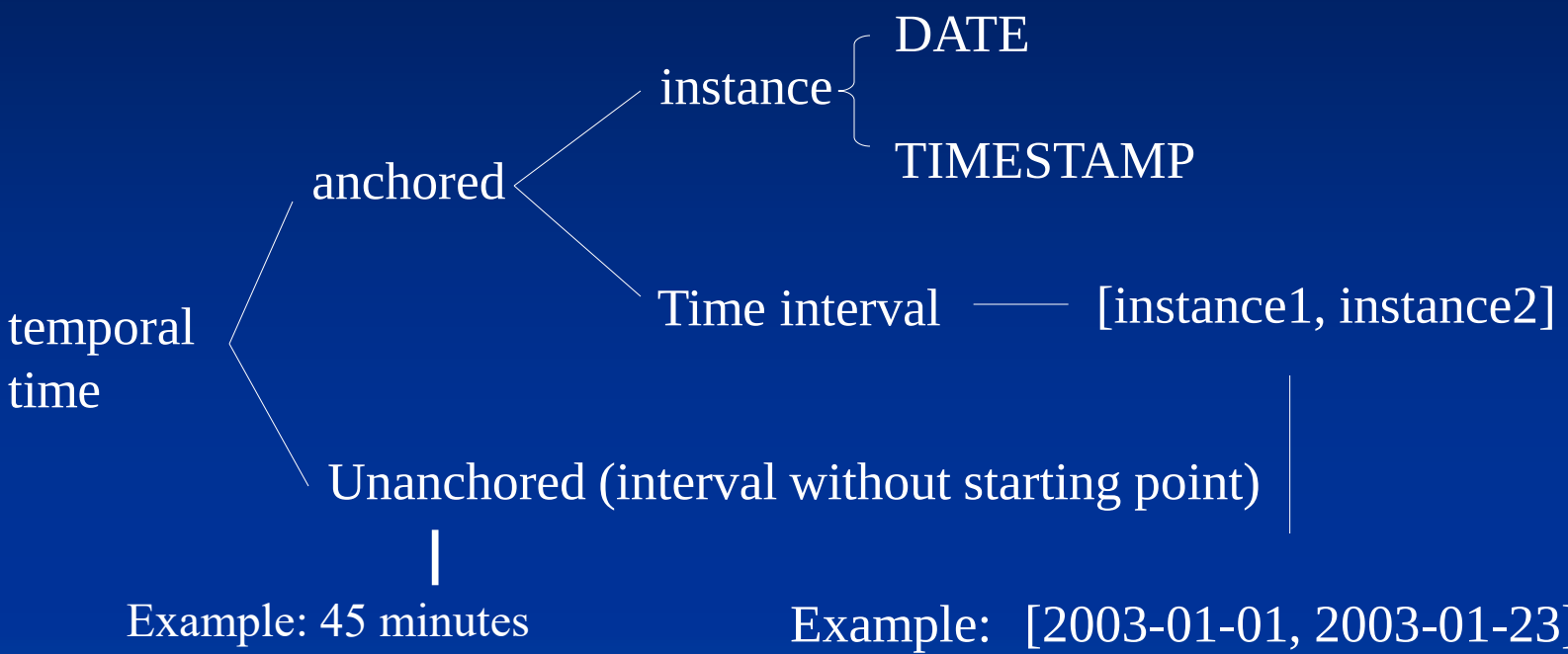
- Valid time: the actual time at which an item was a valid or true value. It can be changed by an application.
  For example, consider the case where a firm plans to increase its prices on a specific date. It might post new prices some time before their effective date.

Difference between transaction time and valid time:

Valid time records when the change takes effect, and transaction time records when the change was entered.

- Storing transaction time is essential for database recovery because the DBMS can roll back the database to a previous state.
- Valid time provides a historic record of the state of the database.

- Anchored time: a time having a defined starting point (e,g., October 15, 2003)



- Unanchored time (interval without starting point)
  It is a single value expressed in some unit or units of time (e.g., 6 years, 5 days, 7 hours).

Example:

```
CREATE TABLE planet (
pltname         VARCHAR(7),
pltday          INTERVAL,
pltyear         INTERVAL,
pk_planet       PRIMARY KEY(pltname));
```

pltday – rotationl period
pltyear – orbital period

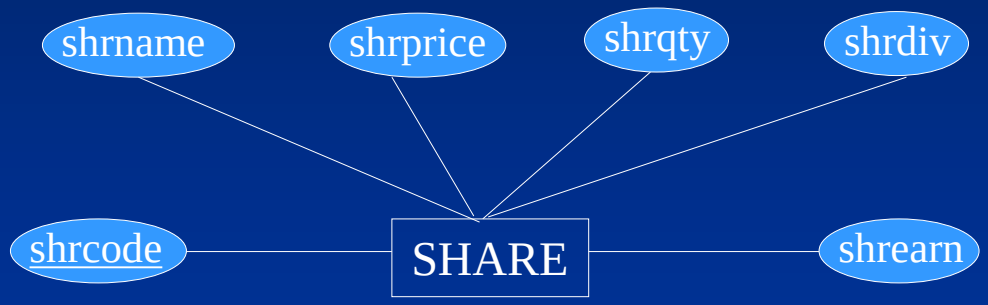
Example:

INSERT INTO planet VALUE ('MERCURY', '1407.51 hours', '0.24 years');

Planet	Rotation period (hours)	Orbital period (years)
Mercury	1407.51	0.24
Venus	-5832.44	0.62
Earth	23.93	1.00
Mars	24.62	1.88
Jupiter	9.92	11.86
Saturn	10.66	29.45
Uranus	17.24	84.02
Neptune	16.11	164.79
Pluto	153.28	247.92

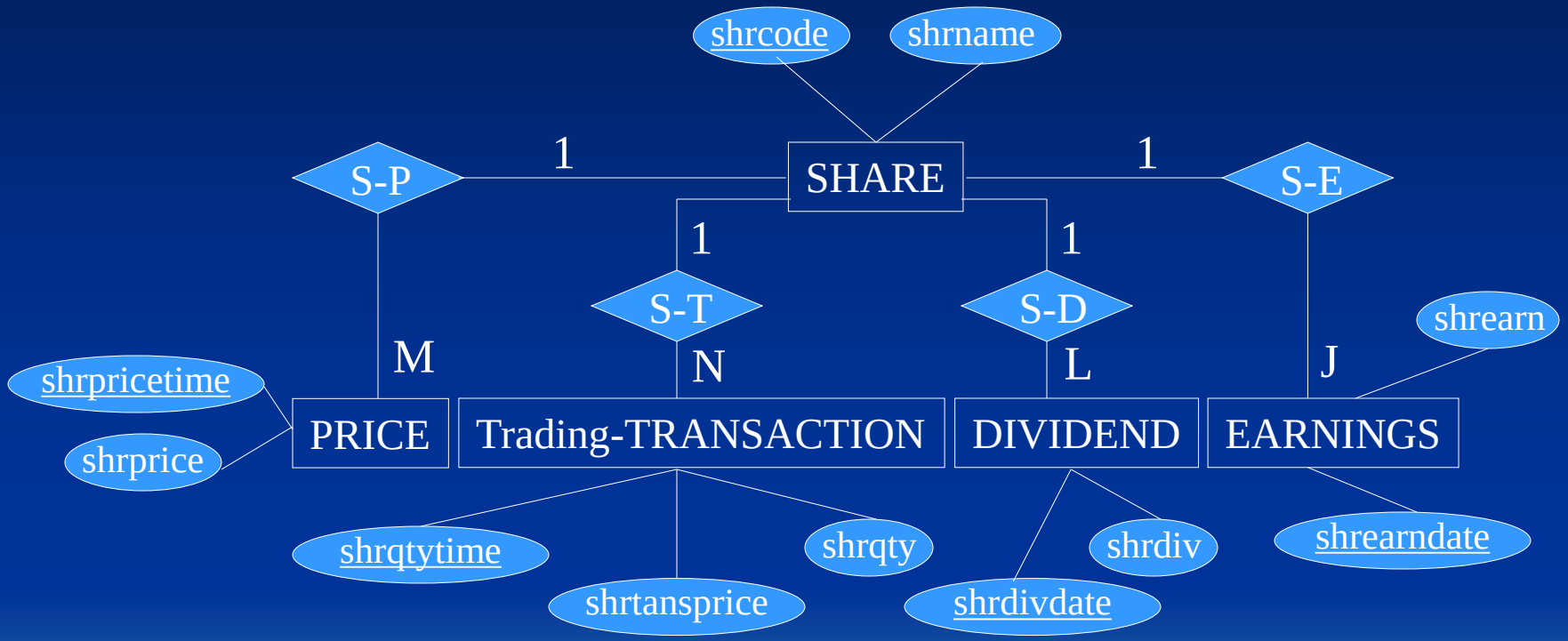
Modeling temporal data

Consider an application for managing information on Shares:



However, *share price*, *quantity owned*, *dividend* and *price-to-earning* ratio are all time-varying. The above data model is not able to capture this feature. So temporal information should be added.

Modeling temporal data



Modeling temporal data

Comments:

Adding attributes to tables to handle temporal data does not make it a temporal database.

- no built-in functions for querying time-varying data
- queries with time concepts cannot be specified in SQL

TSQL (Temporal Structured Query Language):

- there is a proposal to make TSQL and ANSI and ISO standard
- TempDB project in IBM
- Microsoft working on a temporal database