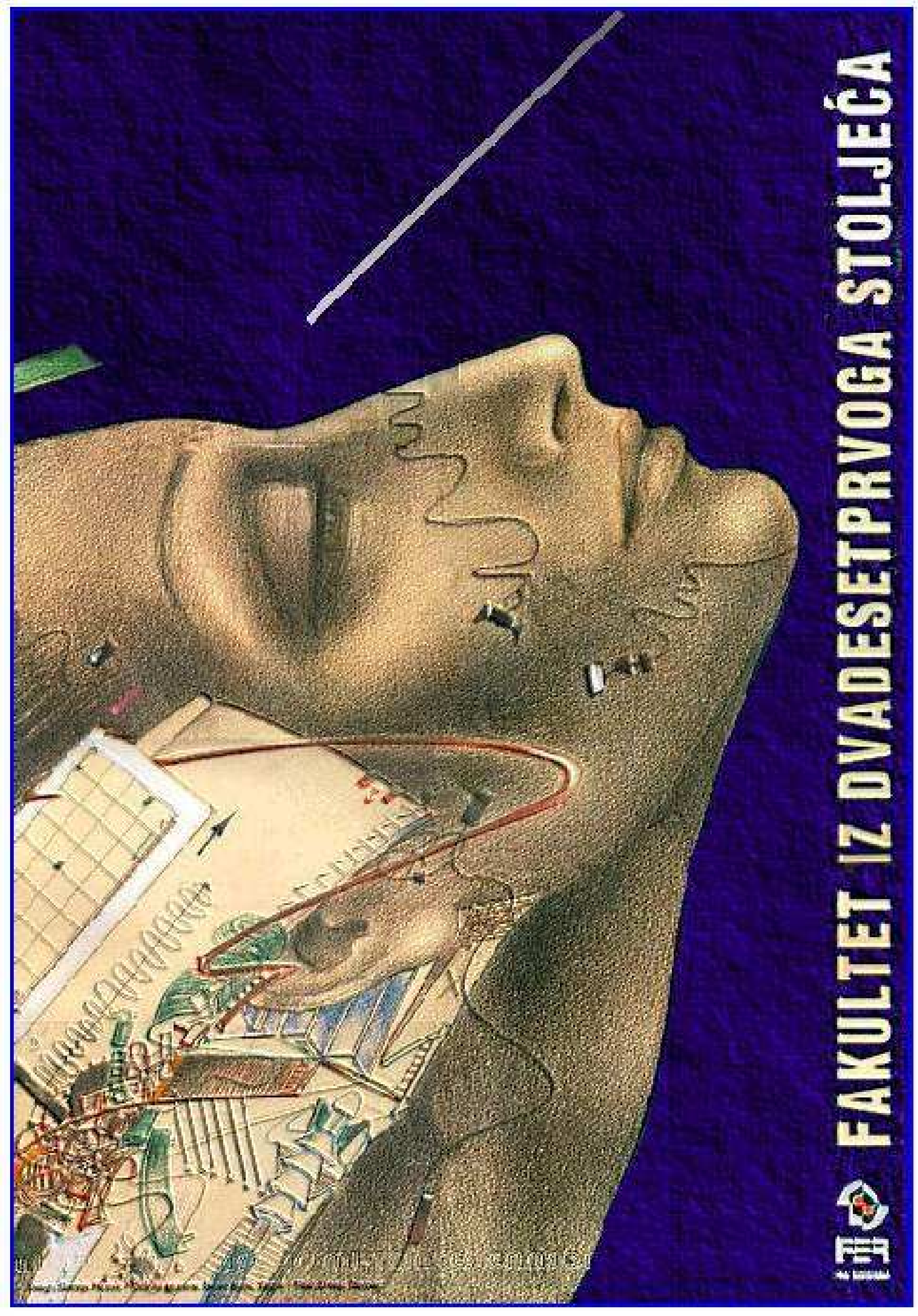
Napredni modeli i baze podataka

Predavanja

9.

Spatio-temporal Databases

Studeni 2008.

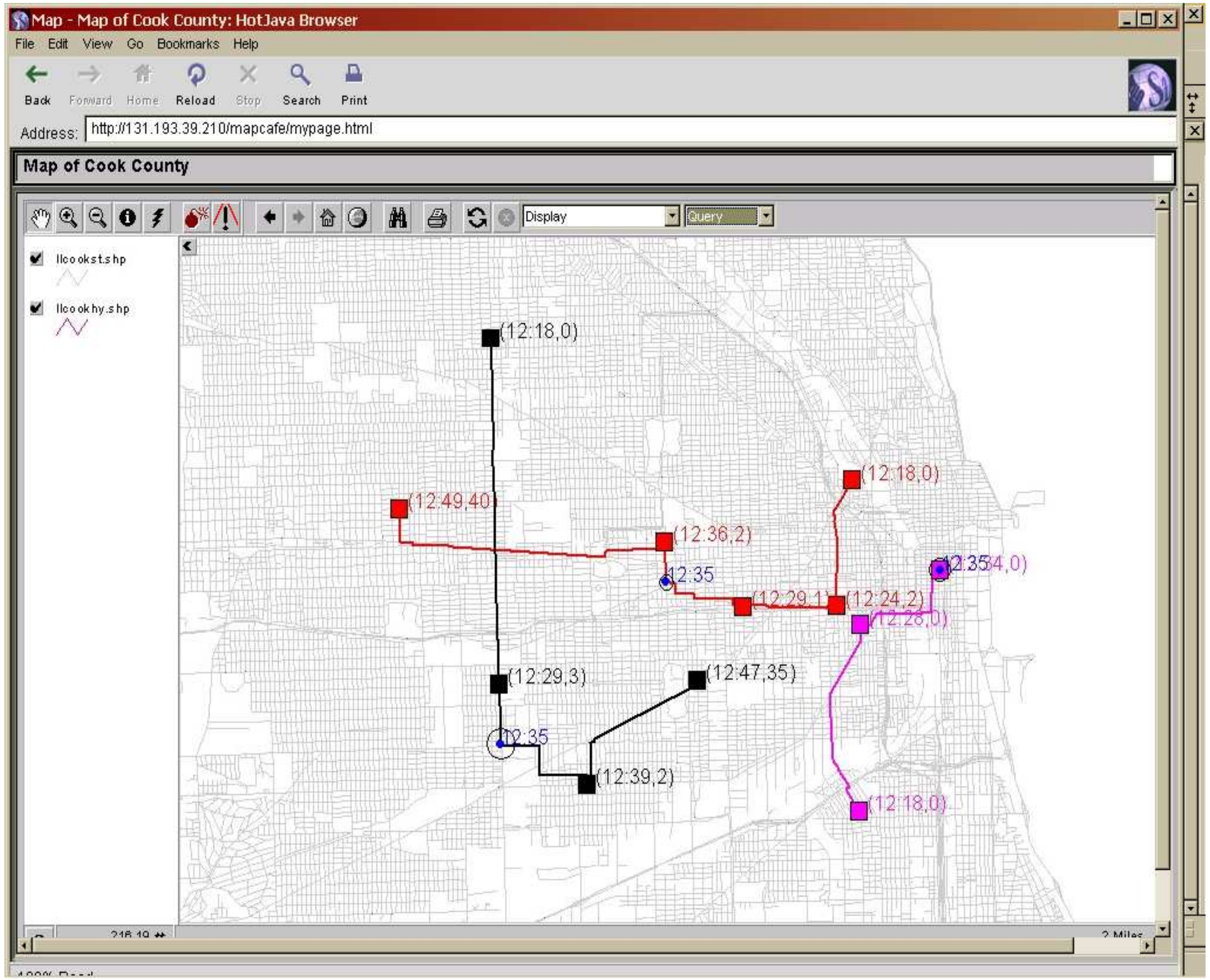


Pregled

- Motivation Examples
- Definitions
- Managing Time in Standard DB
- Time Domain and Time Dimensions
- Taxonomy of Temporal Databases
- Temporal Structured Query Language - TSQL2
- Moving Object Databases
- Examples

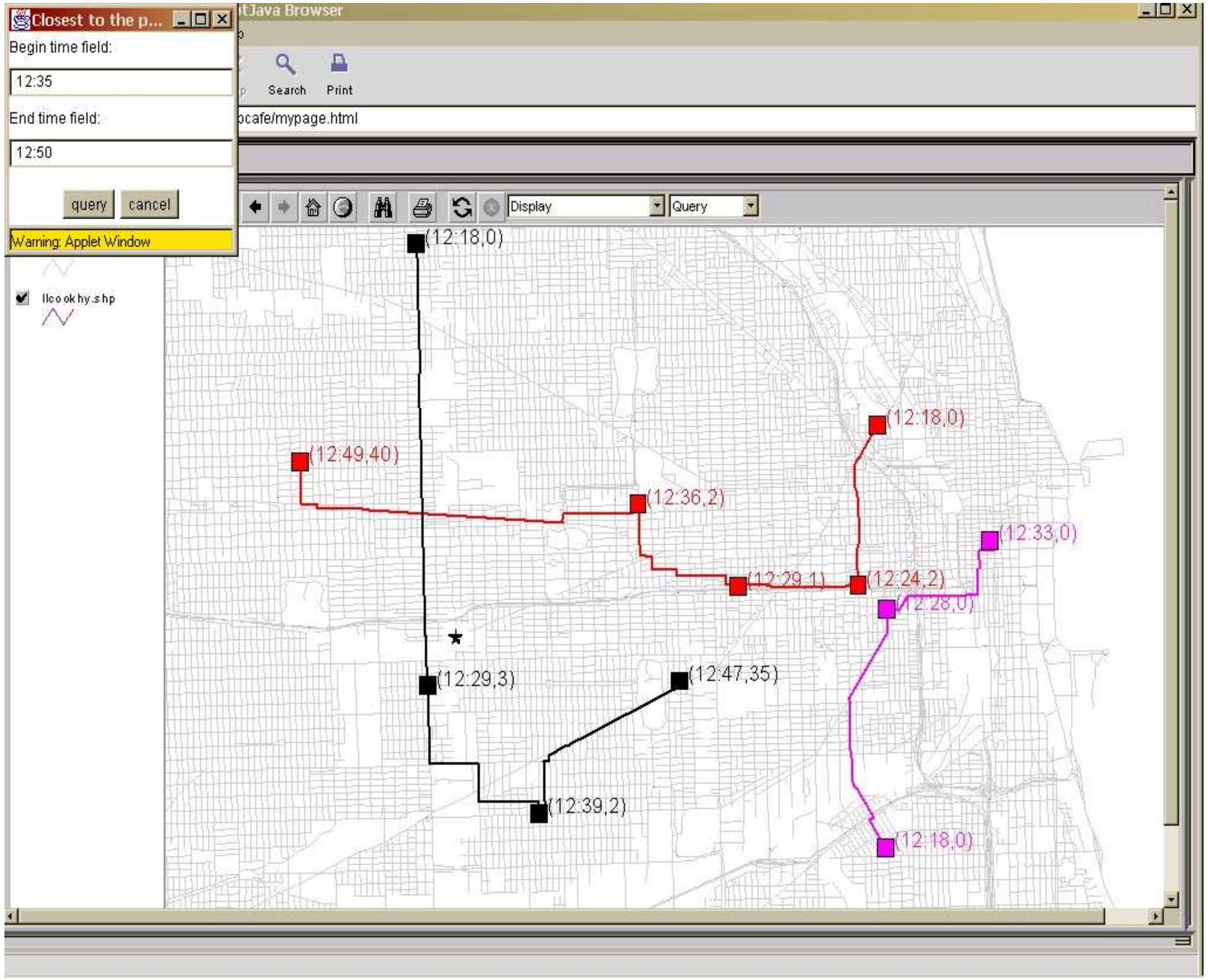
Examples

Q_1 : Show all vehicle's locations at 12:35



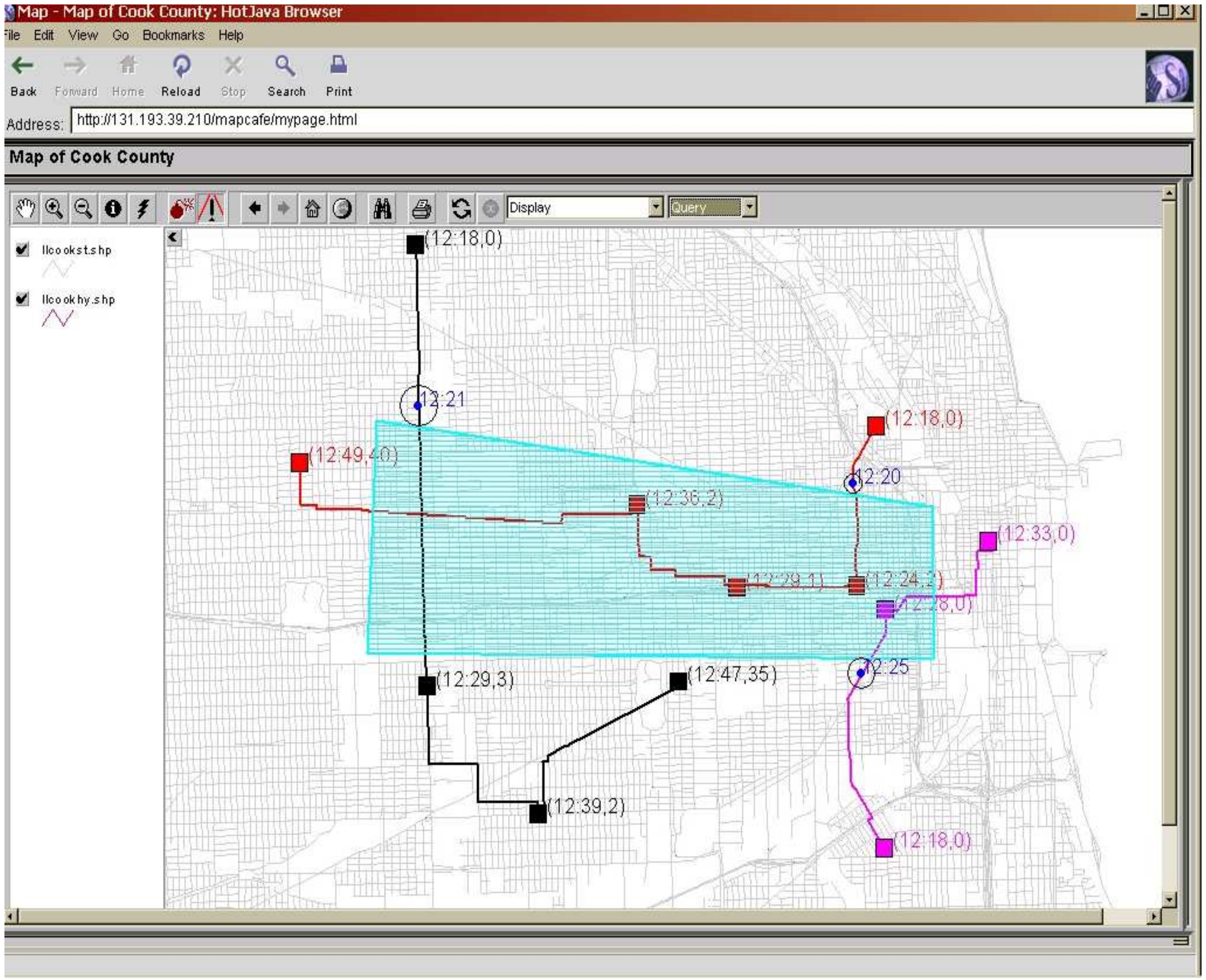
Examples

Q_3 : Which vehicle will be closest to the "star" between 12:35 and 12:50?



Examples

Q_3 : When will each vehicle enter the specified sector?



Definitions

- *A **spatio-temporal** data base is (a new type of) database system that manages spatio-temporal objects and supports corresponding query functionalities*
- *A **spatio-temporal object** is a kind of object that dynamically changes spatial location and/or extents along with time.*
  - *A typical example of a spatiotemporal object is a **moving object** whose location continuously changes*.

*Jespersen and Fitz-Randolph, **From Sundials to Atomic Clocks***

Managing Time in Standard DB

- Databases managed by standard DBMS reflect current state of the world as far as known in the database
- UPDATE → previous state is lost. However, keeping track of history needed by many applications. In standard DBMS possible if ***application*** manages time itself.
- Standard DBMS/SQL offer limited temporal support in the form of data types
  - **`date`** (a particular day from a year in the range 1 through 9999 A.D.)
  - **`time`** (a particular second within a range of 24 hours)
  - **`timestamp`** (a particular fraction of a second of a particular day)
  - **`interval`** (a directed relative duration of time – time interval of known length with unspecified start and end instants)

Managing Time in Standard DB

- Example:

  ```
  parcel (city: string, number: integer, geometry: polygon)
  ```

  Add attributes for managing time:

  ```
  parcel (city: string, number: integer, geometry: polygon,
  from: timestamp, to: timestamp)
  ```

- Disadvantages:
  - Temporal semantic, operations and integrity constraints built into application(s)
  - Difficult, error-prone, complex queries
  - Inefficient query execution

How to Implement ST Applications ?

- **Use built-in SQL data types and build temporal support in application(s)**
  - Disadvantages
    - … *discussed on the previous slide*
- **Implement an abstract data type (ADT) for time**
  - Disadvantages
    - Not possible using a pure relational model
    - … *similar to the previous approach*
- **Extend non-temporal model to temporal data model**
  - Extend non-temporal schema with special temporal attributes
  - Extend algebra and query language wit additional operators to express a *temporal join*, *temporal selection*, *temporal projection*, etc.
  - Disadvantages
    - Proposed extended temporal usually concentrate on special features:
      - Temporal data structures
      - Query language design
      - Temporal algebra
      - Temporal integrity constraints

How to Implement ST Applications ?

- **Generalize non-temporal data model to a temporal data model**
  All three components have to be generalized:
  - Data structures, operations and integrity constraints

    Type or schema of objects is not simply extended, but a new, ***simple*** and ***orthogonal*** concepts needs to be found
  - ***simple***: easily implementable
  - ***orthogonal***:
    - concept is ***not restricted*** to specific constructs of the data model (tuples, atributes, …)
    - User should decide which granularity of data and even constructs of the model (types, collections, integrity constraints or even DB) shall be temporal
    - Temporal operations (including updates) shall refer to this new concept

The Goal of Spatio-Temporal Research

- Integrate spatio-temporal concepts deeply into DBMS data model and query language
- Extend the system implementation accordingly for efficient execution
- Built-in temporal support provides :
  - higher-fidelity data modeling
  - More efficient application development
  - Potential increase in performance

Time

- *It's presented everywhere, but occupies no space*
- *We can measure it, but we can't see it, touch it, get rid of it, or put it in a container*
- *Everyone knows what it is and uses it every day, but no one has been able to define it.*

*Jespersen and Fitz-Randolph, **From Sundials to Atomic Clocks***

The Time Domain

- Time is generally perceived as a one-dimensional space extending from the past to the future:

  Past ⟷ Future

- There are some options:
  - ***bounded/infinite***
  - ***discrete/dense/continuous***
    - Discrete models are isomorphic to the natural numbers or integers
      - Each natural number corresponds to an atomic time interval – ***chronon***
      - Chronons can be grouped into larger units – ***granules*** (hours, days, …)
    - Dense models are isomorphic to either the rationals or the reals
      - between any two instants of time another instant of time exists
    - Continuous models are isomorphic the real numbers
      - Each real number corresponds to a ***point in time***
  - ***absolute/relative***
    - November 20, 2008, 15:45
    - two weeks

Temporal Data Types

- Previous concepts of time can be captured in a number of data types:
  - **`instant`**

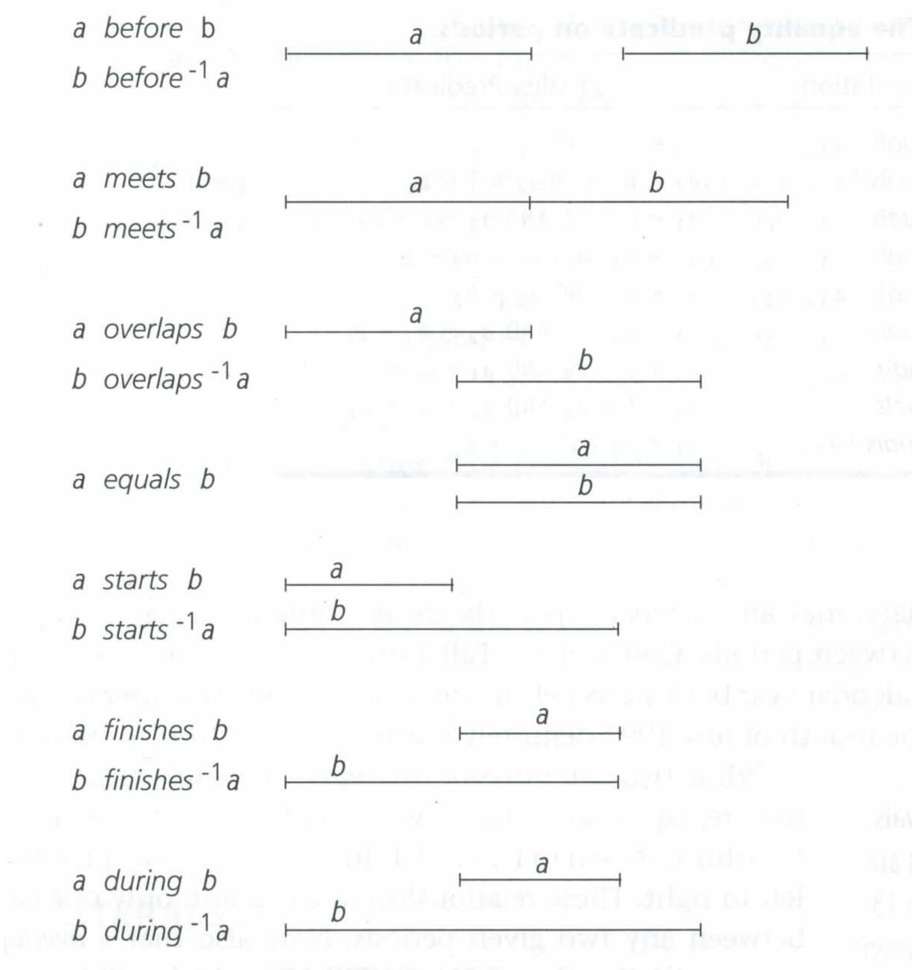
    A particular *chronon* on the time line in the discrete model or a point on the timeline in a continuous model
  - **`period`**

    An anchored interval on the time line
    - Unlike standard SQL temporal types, period is ***not*** ordered – there is only a ***partial order*** between periods
    - Although is not part of SQL standard, period is relatively easy to simulate with **`date/time`**.
  - **`periods`**

    A set of disjoint anchored intervals on the timeline - also called a *temporal element* in the literature.

Relationships/Predicates Between Two Periods

- There are 13 possible relationships/predicates between two periods:



Relationships/Predicates Between Two Periods

- Semantics of Allen's operators

Comparison Predicate	Equivalent Predicates on Endpoints
I_1 before I_2	$end(I_1) < begin(I_2)$
I_1 after I_2	$end(I_2) < begin(I_1)$
I_1 during I_2	$(begin(I_1) > begin(I_2) \wedge end(I_1) \leq end(I_2)) \vee$ $(begin(I_1) \geq begin(I_2) \wedge end(I_1) < end(I_2))$
I_1 contains I_2	$(begin(I_2) > begin(I_1) \wedge end(I_2) \leq end(I_1)) \vee$ $(begin(I_2) \geq begin(I_1) \wedge end(I_2) < end(I_1))$
I_1 overlaps I_2	$begin(I_1) < begin(I_2) \wedge end(I_1) > begin(I_2) \wedge end(I_1) < end(I_2)$
I_1 overlapped_by I_2	$begin(I_2) < begin(I_1) \wedge end(I_2) > begin(I_1) \wedge end(I_2) < end(I_1)$
I_1 meets I_2	$end(I_1) = begin(I_2)$
I_1 met_by I_2	$end(I_2) = begin(I_1)$
I_1 starts I_2	$begin(I_1) = begin(I_2) \wedge end(I_1) < end(I_2)$
I_1 started_by I_2	$begin(I_1) = begin(I_2) \wedge end(I_2) < end(I_1)$
I_1 finishes I_2	$begin(I_1) > begin(I_2) \wedge end(I_1) = end(I_2)$
I_1 finished_by I_2	$begin(I_2) > begin(I_1) \wedge end(I_1) = end(I_2)$
I_1 equals I_2	$begin(I_1) = begin(I_2) \wedge end(I_1) = end(I_2)$

Time Dimensions

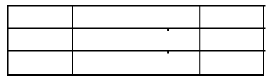
- In the context of databases, two ***orthogonal*** time dimensions are of general interests:
  - ***Valid time***

    Time in real world when an event occurs or a fact is valid, independent of the recording of that event/fact in some database.
  - ***Transaction time***

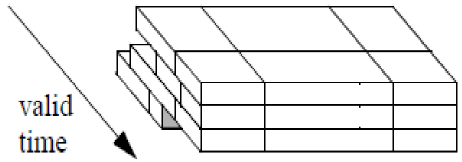
    Time when a change is recorded in the database, or the time interval during which particular state of the database exists.

Time Dimensions

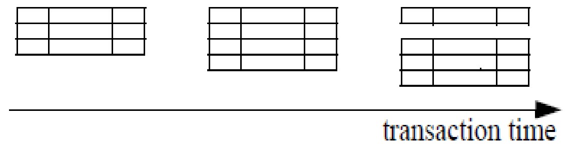
snapshot relation



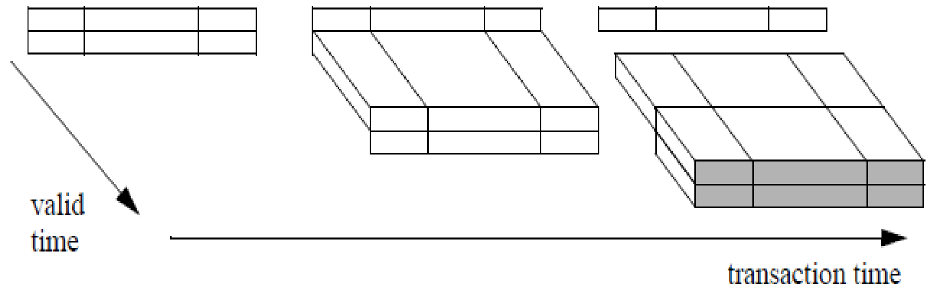
valid-time relation



transaction time relation



bitemporal relation



Time Dimensions

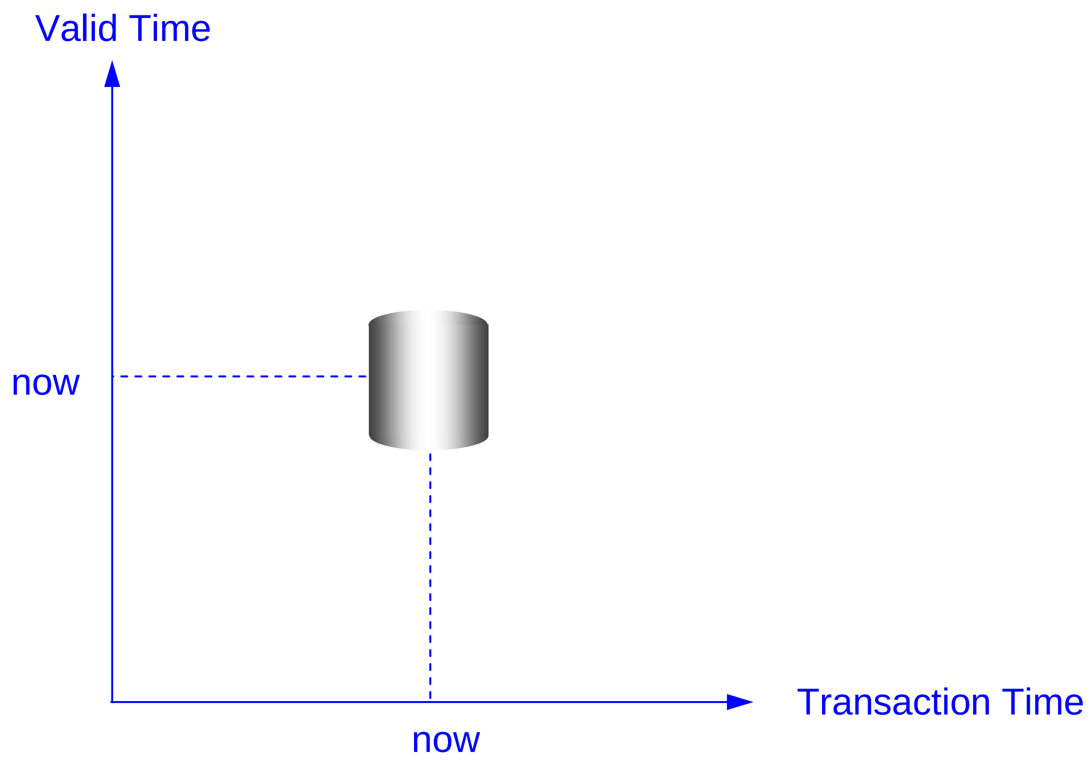
- Valid time and transaction time are:
  - ***Orthogonal***
  - ***but, not homogeneous***

    Transaction time has different semantic than valid time.

    *Valid time may be extended into the future, transaction time is defined only until **now**.*

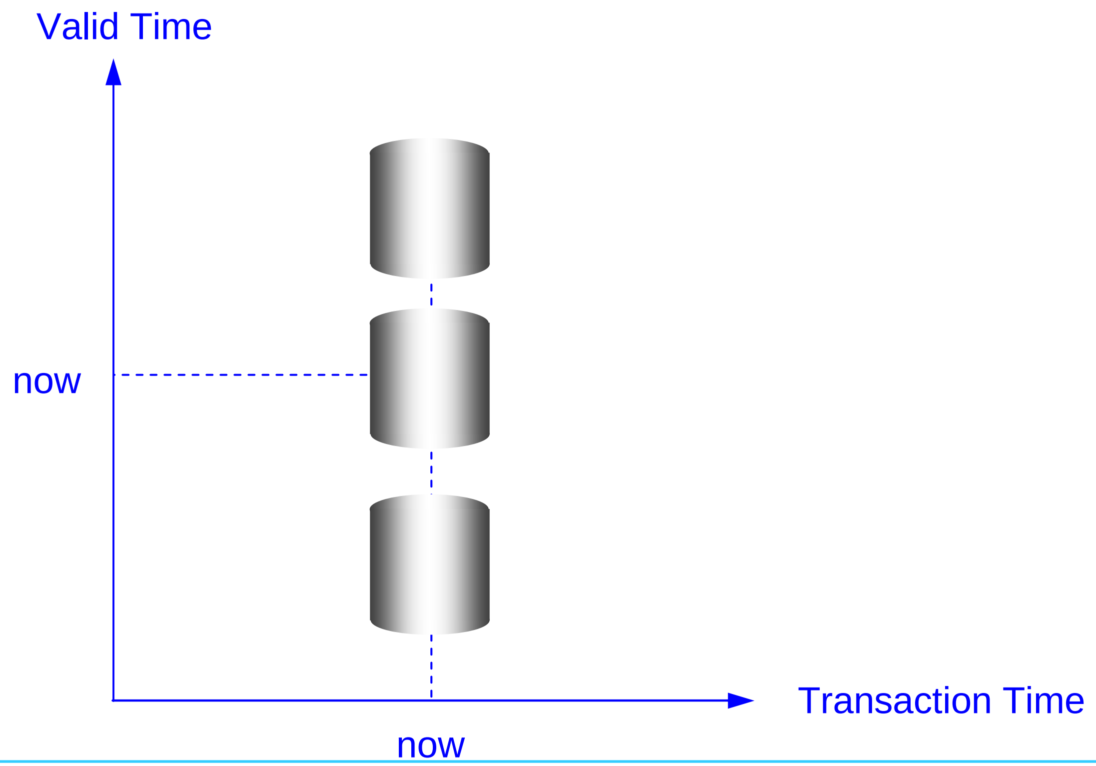
Taxonomy of Temporal Databases

- ***Snapshot Database***
  - Captures a single snapshot of the real world, usually current one
  - This is what currently available commercial DBMS support
  - Modifying the state of DB is done by INSERT, DELETE or UPDATE operations
  - Past states of the database are overwritten



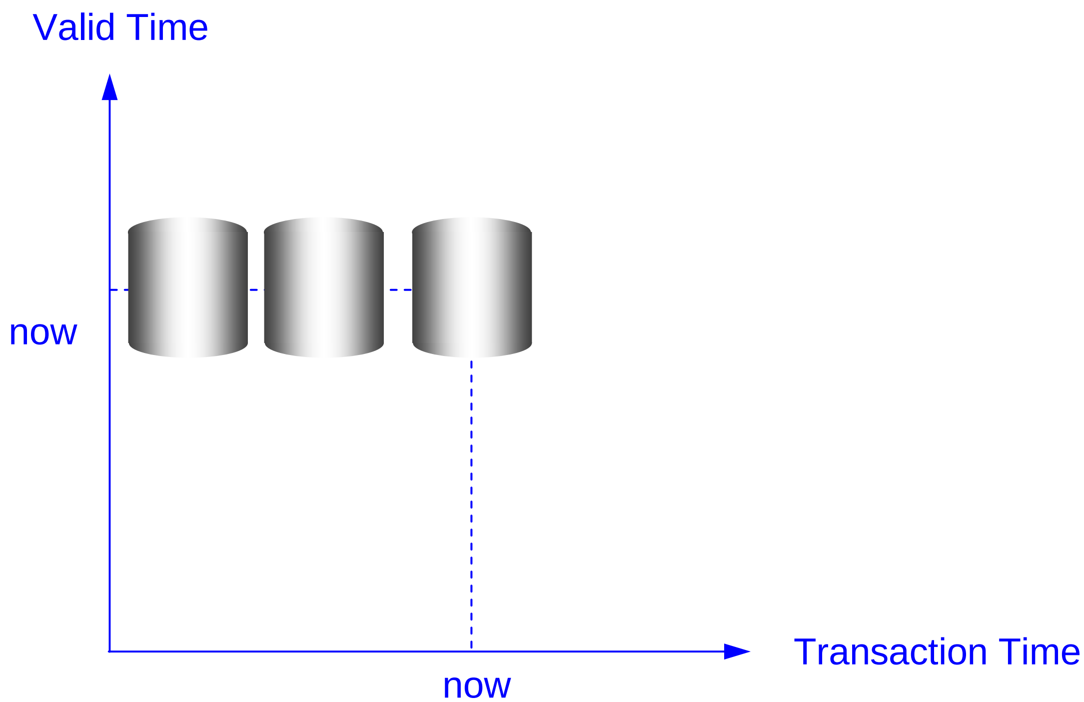
Taxonomy of Temporal Databases

- ***Historical Database***
  - Records the history of data wit the respect to real world, i.e. records database state along the ***valid time*** axis.
  - The user must supply the valid time of a fact
  - Data is still physically deleted when corrected



Taxonomy of Temporal Databases

- ***Rollback Database***
  - Records the changes to DB itself, i.e. records database state along the ***transaction time*** axis.
  - An *append-only* DB, i.e., no data is ever physically deleted

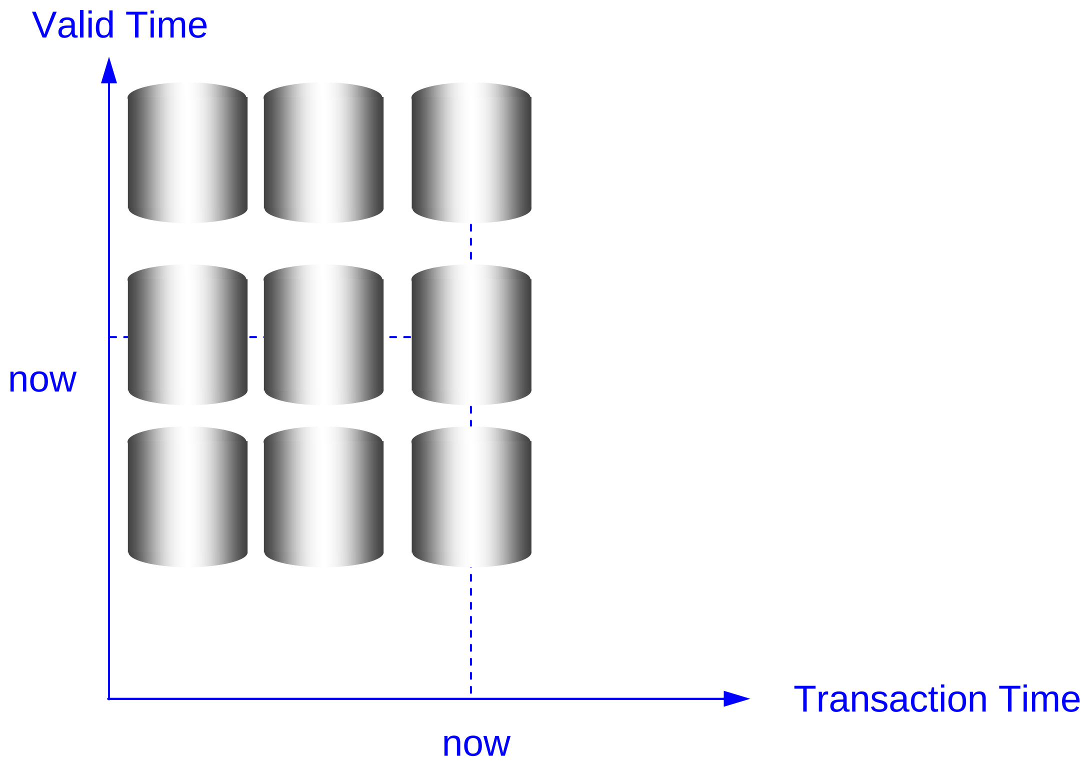


*We consider **linear transaction time** only.*

*An alternative - **branching transaction time**, provides a useful model for **versioning** (**long transaction**).*

Taxonomy of Temporal Databases

- ***Bitemporal Database***
  - A combination of a historical and a rollback DB.
  - Records database states with respect to both ***valid time*** and ***transaction time***



The Temporal Structured Query Language - TSQL2

- Based on the ***Bitemporal Conceptual Data Model*** (BCDM)

1. On the 6th of January, the administration was informed that Lisa had started to work in the toys department on the 1st and was going to work there until the 15th.
2. On the 10th it became known and entered into the database that Lisa had moved to the

Name	*Dept.*	*Time*
Lisa	Toys	{(6, 1), ..., (6, 15), ..., (9, 1), ..., (9, 15), (10, 1), ..., (10, 7), ..., (19, 1), ..., (19, 7), (*uc*, 1), ..., (*uc*, 7), (12, 14), ..., (12, 20), ..., (19, 14), ..., (19, 20), (*uc*, 14), ..., (*uc*, 16)}
Lisa	Books	{(10, 8), ..., (10, 13), ..., (19, 8), ..., (19, 13), (*uc*, 8), ..., (*uc*, 13), (10, 14), (10, 15), (11, 14), (11, 15)}
John	Books	{(12, 11), (12, 12), ..., (12, ∞), (13, 11), ..., (13, ∞), ..., (19, 11), ..., (19, ∞), (*uc*, 11), ..., (*uc*, ∞)}

Bitemporal space

3. On the 12th it was decided that Lisa would move back to toys on the 14th and would stay there a while longer, until the 20th. Also a new employee John had started the day before in the books department.
4. On the 20th, it was entered that Lisa had actually quit the company on the 16th.

The Temporal Structured Query Language - TSQL2

- An example of bitemporal relation:

```
CREATE TABLE parcel (
    city:       STRING,
    number:     INTEGER,
    geometry:   POLYGON)
AS VALID STATE DAY AND TRANSACTION
```

  - **STATE*** – relation/DBMS records facts that are true over certain periods of time
  - **DAY** – valid time granularity (in this example - one day)
  - Transaction time granularity is system dependent (like milliseconds)

* *An alternative clause* **EVENT** *records events that occurred at certain instants of time.*

The Temporal Structured Query Language - TSQL2

- There are six different kinds of relations in TSQL2:
  - Snapshot relations (no temporal support)
  - Valid-time state relations (… **`AS VALID [STATE]`**)
  - Valid-time event relations (… **`AS VALID EVENT`**)
  - Transaction-time relations (… **`AS TRANSACTION`**)
  - Bitemporal state relations (… **`AS VALID STATE AND TRANSACTION`**)
  - Bitemporal event relations (… **`AS VALID EVENT AND TRANSACTION`**)

The Temporal Structured Query Language - TSQL2

- Display geometry of parcels (now or in the past):

```
SELECT SNAPSHOT p.geometry
FROM parcel AS p;
```

- Display juridical geometry changes of parcel 456 in the City of Vienna:

```
SELECT geometry
FROM parcel
WHERE city = "Vienna"
AND   number = 456;
```

The Temporal Structured Query Language - TSQL2

- Display juridical geometry changes of parcel 456 in the City of Vienna, during 2000:

```
SELECT p.geometry
VALID INTERSECT(VALID(p), PERIOD '[2000]' DAY)
FROM parcel AS p
WHERE p.city = "Vienna"
AND   p.number = 456;
```

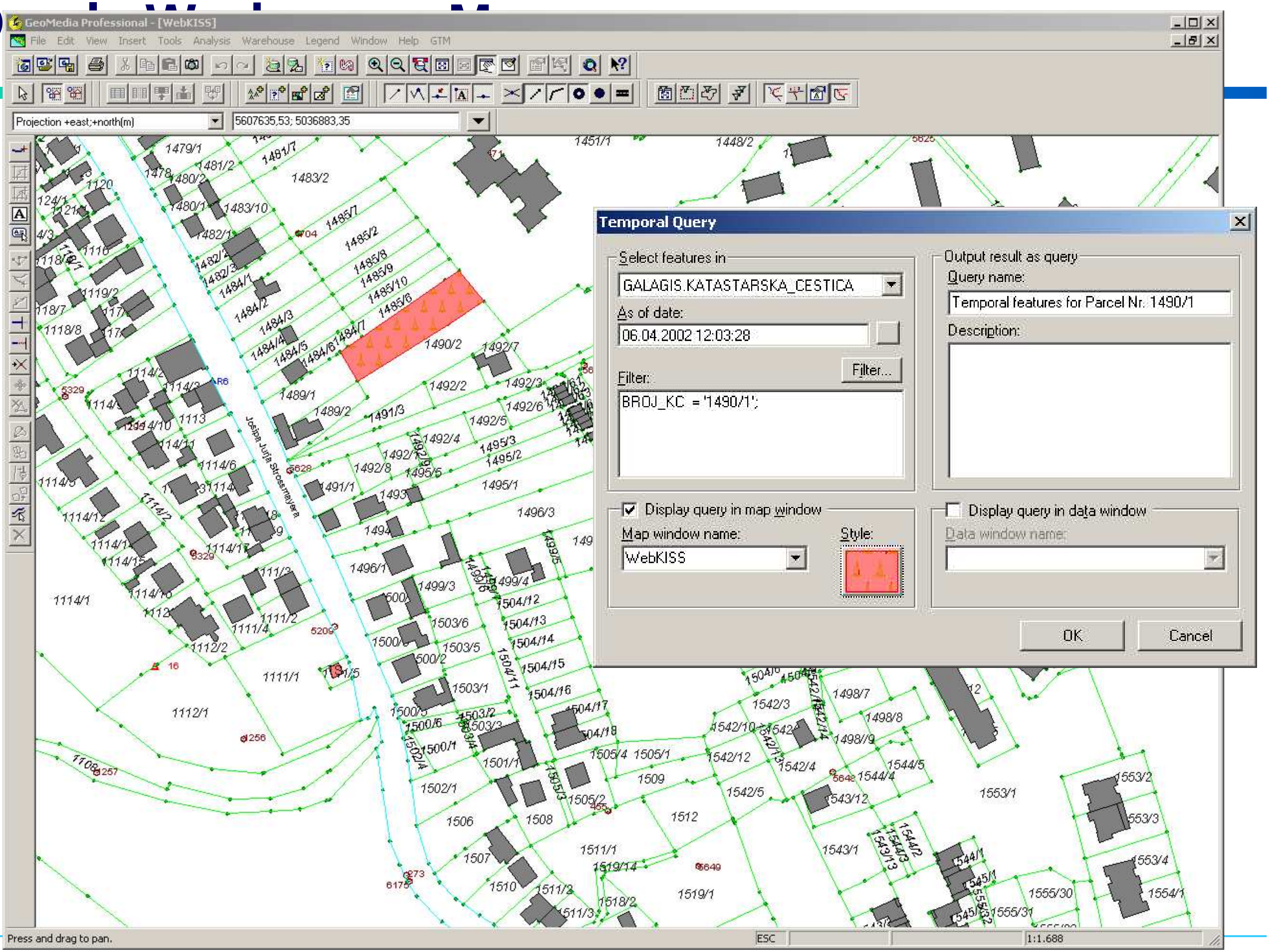
- Display adjacent parcels of parcel 456 in the City of Vienna as of March 17, 1977:

```
SELECT ap.geometry
FROM parcel AS p, parcel AS ap
WHERE VALID(p) OVERLAPS DATE '[1977-03-17]'
AND   p.city = "Vienna"
AND   p.number = 456
AND   p.geometry.touches(ap.geometry);
```

Oracle Workspace Manager

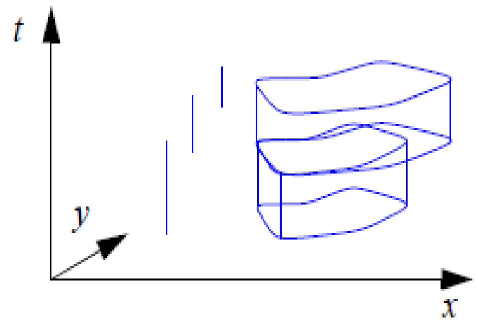
- Wrapper/Simulator on top of Oracle DBMS
- ***Black box*** which cannot be changed (wrapped package)
- Does'not support SQL-like temporal queries, but

```
EXECUTE dbms_wm.GoToDate('06-APR-2002');
SELECT number, geometry
FROM parcel_hist
WHERE number = '1490/1';
```

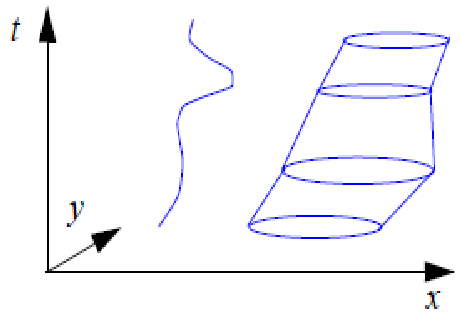

GeoMedia Professional - [WebKISS]
File Edit View Insert Tools Analysis Warehouse Legend Window Help GTM
Projection +east;+north(m)
5607635,53; 5036883,35
Temporal Query
Select features in
GALAGIS.KATASTARSKA_CESTICA
As of date:
06.04.2002 12:03:28
Filter:
Filter...
BROJ_KC = '1490/1';
Output result as query
Query name:
Temporal features for Parcel Nr. 1490/1
Description:
Display query in map window
Map window name:
WebKISS
Style:
Display query in data window
Data window name:
OK
Cancel
Press and drag to pan.
ESC
1:1.688

The Temporal Structured Query Language - TSQL2

- Supports ***discrete*** spatio-temporal changes



- … but not ***continuous*** spatio-temporal changes



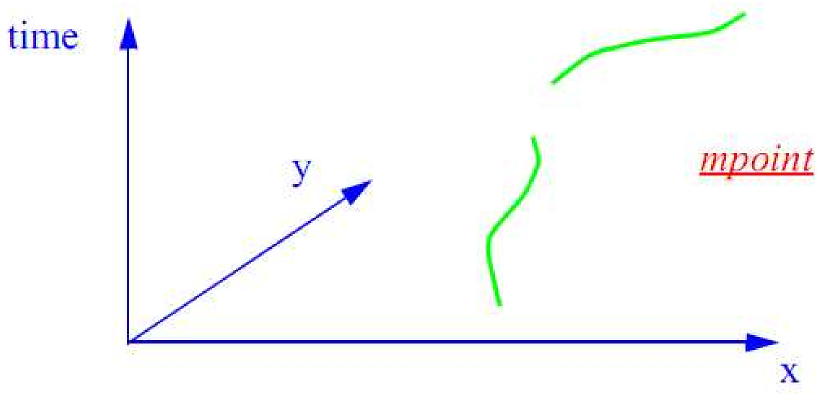
Moving Objects Databases – Current Movement

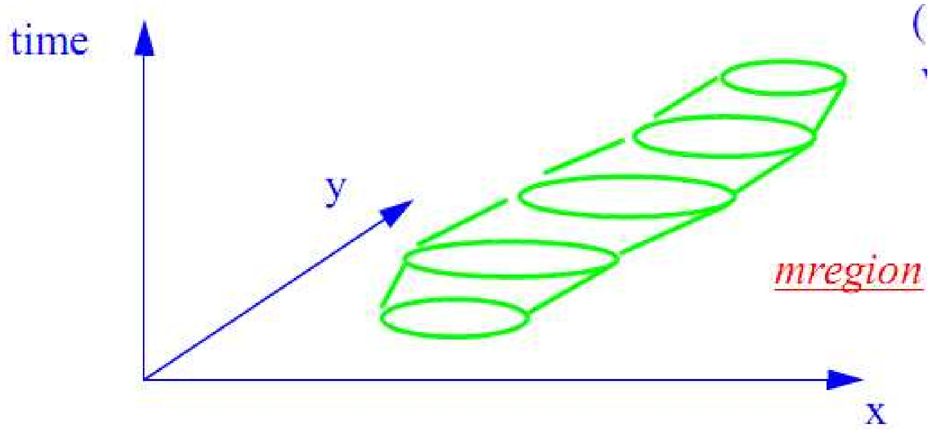
- Moving objects spatio-temporal model - MOST
  - ***Trajectory location management prospective***, i.e. management the locations of a set mobile objects that are moving ***right now***
  - Supports current and expected future movement
  - FTL query language, based on ***future temporal logic***
  - Basically SQL extended with temporal operators
    - **UNTIL, NEXTTIME, EVENTUALLY, ALWAYS**
  - Restricted on moving points
  - Does not address any complex geometries
  - Query example:

```
SELECT id
FROM helicopters h
WHERE EVENTUALLY_WITHIN (inside (h, Valley), 10)
AND ALWAYS (inside (h, Valley), 2);
```

Moving Objects Databases – History of Movement

- Spatio-temporal data perspective
  - Geospatial objects stored in geospatial DB which change continuously over time
  - Spatio-temporal data types
    - Time dependent geometries with operations
    - ***moving point*** (people, cars, animals, planes, ships, …)
    - ***moving region*** (forests, forest fires, countries, hurricanes, armies, …)





Moving Objects Databases – History of Movement

- Assume some available operators

moving(point)	$\rightarrow$	*line*	***trajectory***
moving(region)	$\rightarrow$	*region*	***traversed***
moving(α)	$\rightarrow$	*periods*	***deftime***
moving(point) x moving(region)	$\rightarrow$	*moving(point)*	***intersection***
moving(α) x instant	$\rightarrow$	*intime(α)*	***atinstant***
intime(α)	$\rightarrow$	*instant*	***inst***
intime(α)	$\rightarrow$	α	***val***
periods(α)	$\rightarrow$	*int*	***duration***
intime(α)	$\rightarrow$	α	***val***

moving - a type constructor that transforms a type α into a time dependent version of that type, *moving(α)*

Moving Objects Databases – History of Movement

- Examples

```
flight (id:STRING,from:STRING,to:STRING,route:mPoint)
weather (id : STRING, kind : STRING, area : mRegion)
```

All flights from Vienna that are longer than 5000 miles

```
SELECT id
FROM flight
WHERE from = "ZAG"
AND length(trajectory(route)) > 5000;
```

Moving Objects Databases – History of Movement

- Examples

  What was the route taken by flight 'OS 7052‘ ?

```
SELECT trajectory (route)
FROM flight
WHERE id = "OS 7052";
```

  Which flights went through a snow storm ?

```
SELECT f.id
FROM flight f, weather w
WHERE w.kind = "snow storm"
AND duration(visits(f.route, w.area)) > 0
```

  Which flights were in snow storm for more than 5 minutes ?

```
SELECT f.id
FROM flight f, weather w
WHERE w.kind = "snow storm"
AND duration(deftime(intersection(f.route, w.area))) > 300
```

Moving Objects Databases – History of Movement

- Implementation concept

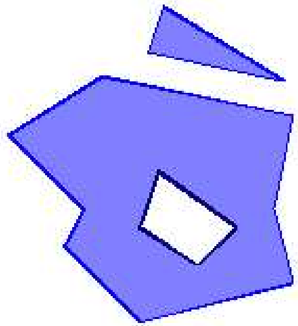
  ADT extension package to an extensible (OR, OO) DBMS
- Work program
  - Design a system of types and operations (***many sorted algebra***)
    - Which level of abstraction ?
    - Abstract model (infinite representation)
      - A continuous function from time into the Euclidean plane **$f: \mathfrak{R} \rightarrow \mathfrak{R}^2$**
      - Mathematically simple, elegant, and uniform, but not directly implementable
    - Discrete model (finite representation)
      - A polyline in the 3D space representing such a function
      - More complex and heterogeneous, but can be implemented
  - Design data structures for the types and algorithms for the operations
  - Implement DBMS extension package

Moving Objects Databases – History of Movement

- Discrete model (part)

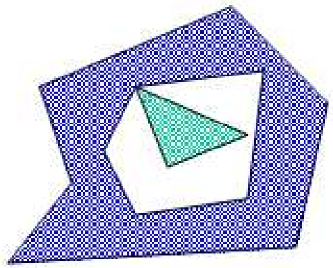
region

abstract

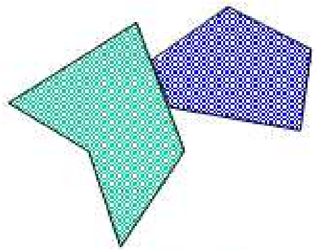


discrete

a finite set of polygons, each with polygonal holes



edge-disjoint



not edge-disjoint

$Seg = \quad \ldots$ (def. of line segments)

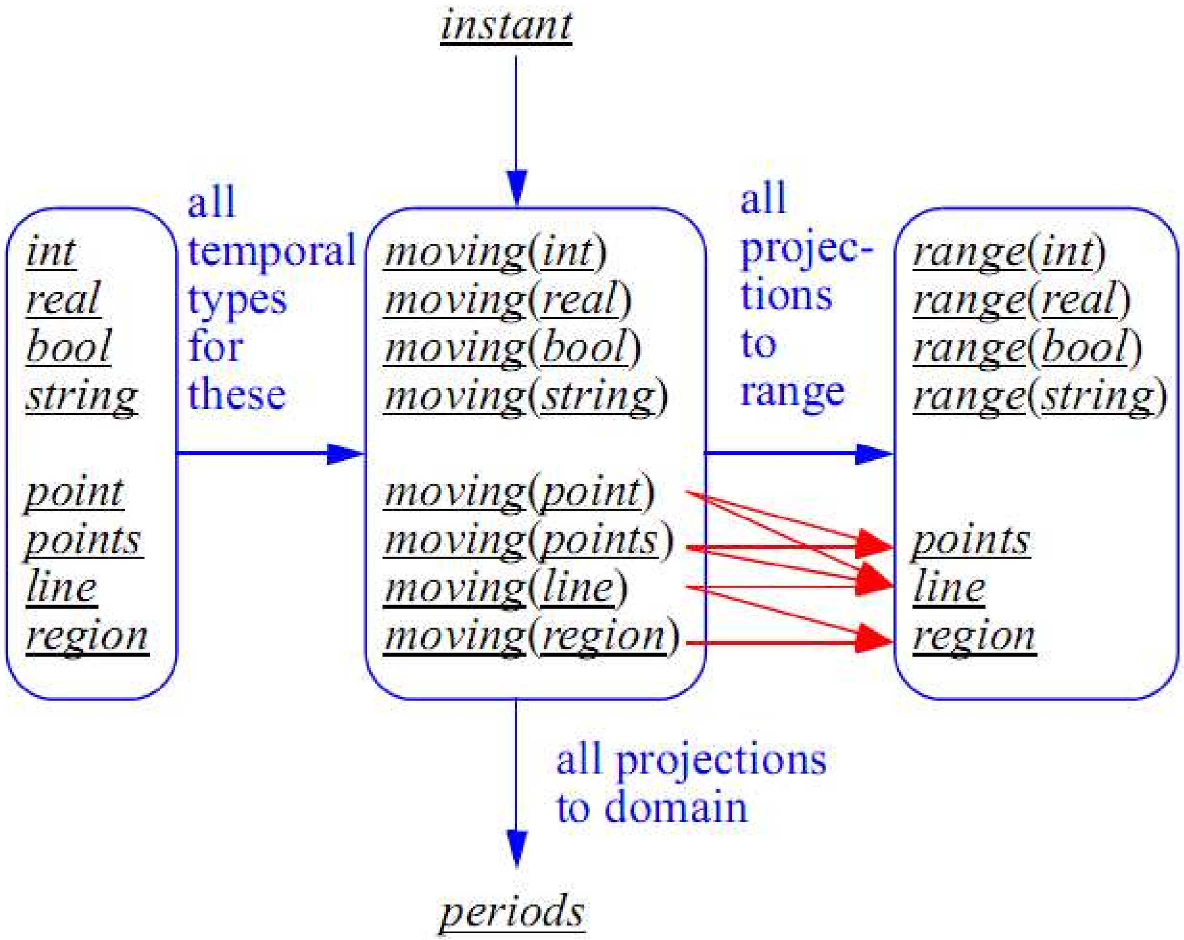
$Cycle = \{S \subset Seg \mid \ldots\}$ (def. of simple polygon)

$Face = \{(c, H) \mid c \in Cycle, H \subset Cycle$, such that ...}

$D_{\underline{region}} = \{F \subset Face \mid f_1, f_2 \in F \wedge f_1 \neq f_2$
$\Rightarrow \text{edge-disjoint}(f_1, f_2)\}$

Moving Objects Databases – History of Movement

- Type System



Moving Objects Databases – History of Movement

- Type System as a ***signature***

Type constructor	*Signature*	
int, *real*, *string*, *bool*		$\rightarrow$ BASE
point, *points*, *line*, *region*		$\rightarrow$ SPATIAL
instant		$\rightarrow$ TIME
range	BASE $\cup$ TIME	$\rightarrow$ RANGE
moving, *intime*	BASE $\cup$ SPATIAL	$\rightarrow$ TEMPORAL

  - A signature has ***sorts*** and ***operations***
  - Sorts – collections of types
  - Operations – type constructors

Moving Objects Databases – History of Movement

- Operations and QL – Design Goals
  - As generic as possible
  - Consistency between operations on non-temporal and temnporal types
    - ***point x point*** $\rightarrow$ ***real*** ***distance***
    - ***mpoint x mpoint*** $\rightarrow$ ***mreal*** ***mdistance***

1. Define operations on non-temporal types
2. "Lift" them all to temporal types – lifting
3. Add specific operations for temporal types
4. To obtain a powerful query language, it is necessary to include operations from various domains:
   - *Simple set theory*
   - *FOL*
   - *Order relationships*
   - *Topology*
   - *Metric spaces*
   - *…*

Moving Objects Databases – History of Movement

- Operations on Non-Temporal Types

Generic view: *point* and *point set* in some space

	Space	point type (π)	point set type (σ)
1D Spaces	Integer	*int*	*range*(*int*)
	Real	*real*	*range*(*real*)
	Bool	*bool*	*range*(*bool*)
	String	*string*	*range*(*string*)
	Time	*instant*	*periods*
2D Space	2D	*point*	*points*, *line*, *region*

$\pi \times \sigma$	$\rightarrow$ *bool*	**inside**	instantiates to
int $\times$ *range*(*int*)	$\rightarrow$ *bool*	**inside**	
bool $\times$ *range*(*bool*)	$\rightarrow$ *bool*		
instant $\times$ *periods*	$\rightarrow$ *bool*		
point $\times$ *line*	$\rightarrow$ *bool*		etc.

Moving Objects Databases – History of Movement

- Operations on Non-Temporal Types …

Class	Operations
Predicates	**isempty** =, /=, **intersects**, **inside** <, <=, >=, >, **before** **touches**, **attached**, **overlaps**, **on_border**, **in_interior**
Set Operations	**intersection**, **union**, **minus** **crossings**, **touch_points**, **common_border**
Aggregation	**min**, **max**, **avg**, **center**, **single**
Numeric	**no_components**, **size**, **perimeter**, **duration**, **length**, **area**
Distance and Direction	**distance**, **direction**
Base Type Specific	**and**, **or**, **not**

Moving Objects Databases – History of Movement

- Operations on Non-Temporal Types - Predicates

Unary

Operation	*Signature*	*Semantics*
isempty[undefined]	$\pi \rightarrow \underline{bool}$	$u = \perp$
	$\sigma \rightarrow \underline{bool}$	$U = \varnothing$

Binary

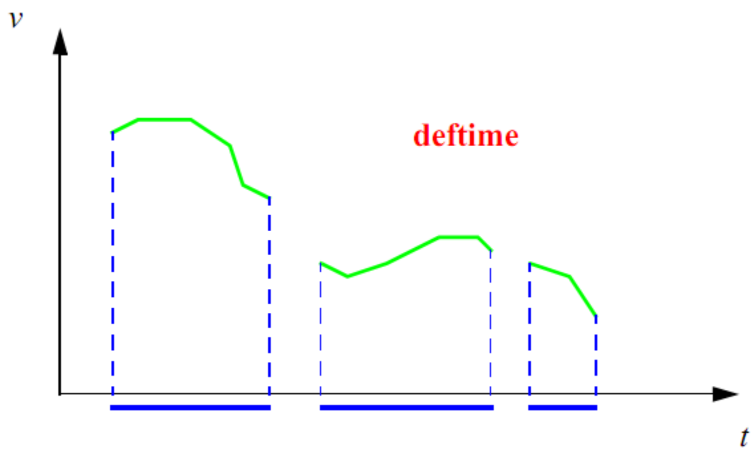
	Sets	*Order (1D Spaces)*	*Topology*
point vs. point	$u = v$, $u \neq v$	$u < v$, $u \leq v$, $u \geq v$, $u > v$	
point set vs. point set	$U = V$, $U \neq V$ $U \cap V \neq \varnothing$ **(intersects)** $U \subseteq V$ **(inside)**	$\forall u \in U, \forall v \in V : u \leq v$ **(before)**	$\partial U \cap \partial V \neq \varnothing$ **(touches)** $\partial U \cap V^{\circ} \neq \varnothing$ **(attached)** $U^{\circ} \cap V^{\circ} \neq \varnothing$ **(overlaps)**
point vs. point set	$u \in V$ **(inside)**	$\forall u \in U : u \leq v$ **(before)** $\forall v \in V : u \leq v$	$u \in \partial V$ **(on_border)** $u \in V^{\circ}$ **(in_interior)**

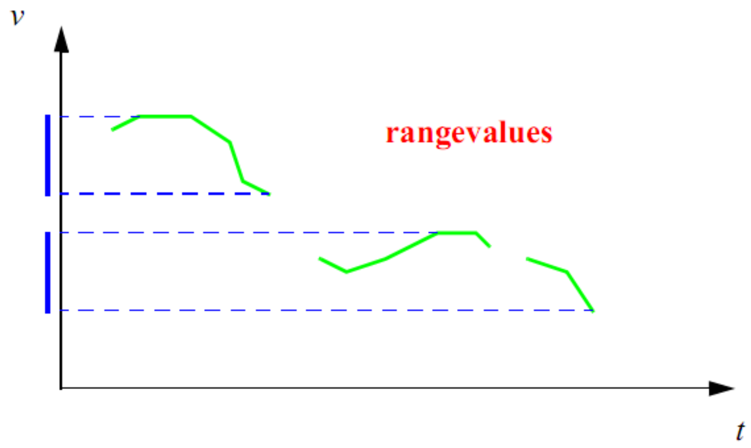
Moving Objects Databases – History of Movement

- Operations on Temporal Types
  - Values of temporal types (i.e. *moving*(α)) are partial functions

$$f: A_{\underline{instant}} \rightarrow A_{\alpha}$$

- Projection to Domain and Range
  - ***deftime*** - define set of time intervals when a temporal function is defined
  - ***rangevalues*** - define set of time intervals when a temporal function is defined





Moving Objects Databases – History of Movement

- Operations on Temporal Types

moving(α)	$\rightarrow$	*periods*	***deftime***
moving(α)	$\rightarrow$	*range(α)*	***rangevalues***
moving(point)	$\rightarrow$	*points*	***locations***
moving(points)	$\rightarrow$	*points*	***locations***
moving(point)	$\rightarrow$	*line*	***trajectory***
moving(points)	$\rightarrow$	*line*	***trajectory***
moving(line)	$\rightarrow$	*line*	***routes***
moving(line)	$\rightarrow$	*region*	***traversed***
moving(region)	$\rightarrow$	*region*	***traversed***

Moving Objects Databases – History of Movement

- Operations on Temporal Types

  How large was the area of Austria affected by hurricane “Lizzy" ?

```
LET Austria = …
LET Lizzy = ELEMENT (SELECT id
FROM weather WHERE id = "Lizzy");
area(intersection(traversed(Lizzy), Austria);
```

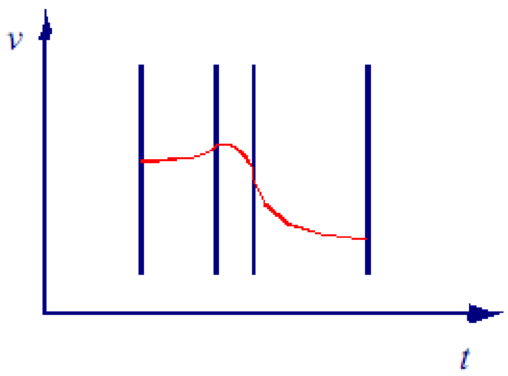
  Where was flight "OS 7052“ while hurricane “Lizzy" was over Austria ?

```
LET Austria = …
trajectory(
   atperiods   (ELEMENT(SELECT route FROM flight
               where id = "OS 7052"),
   deftime(at(Lizzy, Austria)));
```

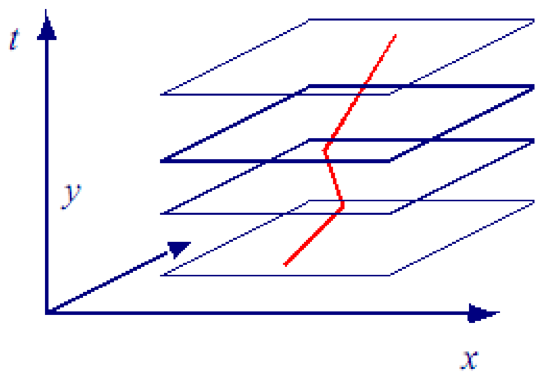
Moving Objects Databases – History of Movement

- Implementation
  - Based on discrete model based on ***sliced representation***:

    Temporal development of the value of type α by decomposing the time dimension into a set of disjoint time intervals (“slices”) such that within each slice the development can be described by some “simple” function.



<u>moving</u>(<u>real</u>)



<u>moving</u>(<u>point</u>)

References

- Allen J.F.: “*Maintaining Knowledge about Temporal Intervals*", Communications of the ACM,, 26(11), pp. 832-843. ACM Press, 1983.
- Güting R.H., Schneider M.: "*Moving Objects Databases*", Morgan Kaufmann Publishers, 2005.
- Koubarakis M., Frank A.U., Grumbach S, Güting R.H, Jensen C.S., Lorentzos N.A., Manolopoulos Y., Nardelli E., Pernici B., Schek H.J., Scholl M., Theodoulidis B., Tryfona N. : “*Spatio-Temporal Databases: The CHOROCHRONOS Approach*", LNCS 2520, Springer, 2003
- Snodgrass R. T.: "*Developing Time-Oriented Database Applications in SQL*", Morgan Kaufmann Publishers, 2000
- Steiner A.: “*A Generalisation Approach to Temporal Data Models and their Implementations*", Swiss Federal Institute of Technology, 1998
- Zaniolo C., Ceri S., Faloutsos C., Snodgrass R.T., Zicari, R.: "*Advanced Database Systems*", Morgan Kaufmann Publishers, 1997
- TSQL2: http://www.cs.arizona.edu/~rts/tsql2.html
- SQL3 http://www.cs.arizona.edu/~rts/sql3.html
- TimeCenter: http://timecenter.cs.aau.dk/index.htm