Temporal Data Models

Fabio Grandi
fabio.grandi@unibo.it
DISI, Università di Bologna

A short course on Temporal Databaes for DISI PhD students, 2016
Credits: most of the materials used is taken from slides prepared by Prof. M. Böhlen (Univ. of Zurich, Switzerland)

Temporal Data Models

- Data model: *DM = (DS, QL)*
  - *DS* is a set of data structures
  - *QL* is a language for querying and updating the data structures
- Example: the relational data model is composed of relations and SQL (or relational algebra)
- Many extensions of the relational data model to support time have been proposed

Temporal Data Models

- Several modeling aspects have to be considered
  - Different Time dimensions
  - Different Timestamp types
  - Tuple versus Attribute timestamping
    (Ungrouped versus Grouped model?)
  - Point-based versus Period-based model
    (Atelic versus Telic data?)
- The different modeling aspects lead to subtle and difficult issues. There are pros and cons in all cases (no consensus can be reached)

Time Dimensions

- Time in a TDB is multi-dimensional:
  - valid time, transaction time, event/decision time, publication time, efficacy time, “user-defined” time
- Different time dimensions are of practical interest in different application fields
- The key question is: which time aspects are *sufficiently important* so that they should be supported by the database system?
- There is a broad consensus that transaction time and valid time are the most important time dimensions

Valid Time

- Valid time is the time a fact was/is/will be true in the modeled reality or mini-world
  - A fact is a statement that is either true or false
  - A relation is a collection of facts
  - Example: John has been hired on October 1, 2014
  - Valid time captures the time-varying states of the mini-world
  - All facts have a valid time by definition, however, it might not be recoreded in the database
  - Valid time is independent of the recording of the fact in a database
  - Valid time is either bounded (does not extend until infinity) or unbounded (extends until infinity)
  - Future facts can be represented (stated or forecasted)

Transaction Time

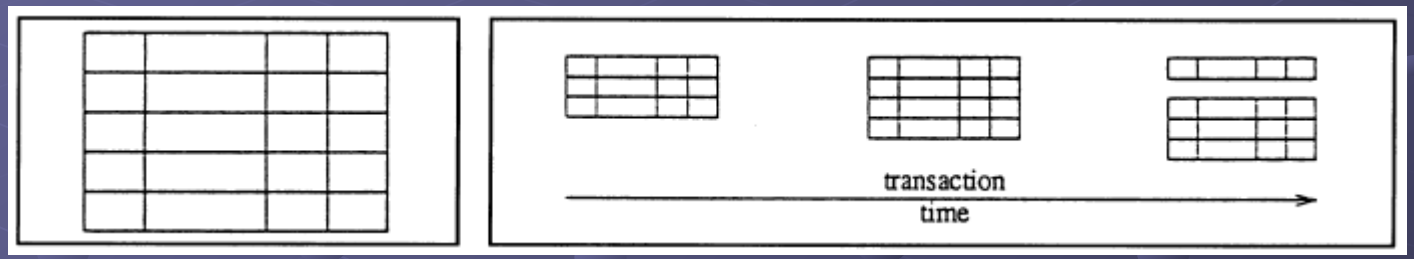
- Transaction time is the time when a fact is current/present in the database as stored data
  - Example: the fact “John was hired on October 1, 2014” was stored in the DB on October 5, 2014, and has been deleted on March 31, 2015
  - Transaction time has a duration: from insertion to deletion, with multiple insertions and deletions being possible for the same fact
  - With transaction time deletions of facts are purely logical
    - the fact remains in the database, but ceases to be part of the database current state.
  - Transaction time captures the time-varying states of the database
  - Always bounded on both ends
    - Starts when the database is created (nothing was stored before)
    - Does not extend past now (no facts are known to have been stored in the future)
  - Basis for supporting accountability and “traceability” requirements, e.g. in financial, medical, legal applications
  - Should be supplied and managed automatically by the DBMS

Dimensions of Time

- A data model can support none, one, two, or more of these time dimensions
  - Snapshot data model: None of the time dimensions is supported
    - Represents a single snapshot of the reality and the database
  - Valid time data model: Supports only valid time
  - Transaction time data model: Supports only transaction time
  - Bitemporal data model: Supports valid time and transaction time
- In a former terminology [Snodgrass & Ahn 1986]:
  - Historical DB → valid-time DB
  - Rollback DB → transaction-time DB
  - Temporal DB → bitemporal DB
- A DB where snapshot, transaction-time, valid-time and bitemporal relations coexist can be called a multi-temporal database

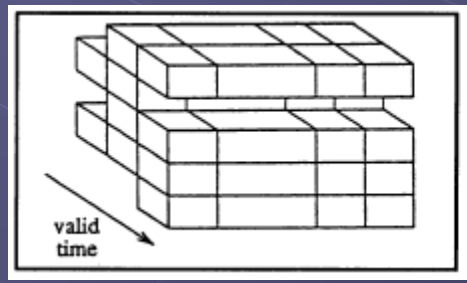
Temporal Relations

- A pictorial representation of the 4 kinds of temporal table and evolution along the time axes

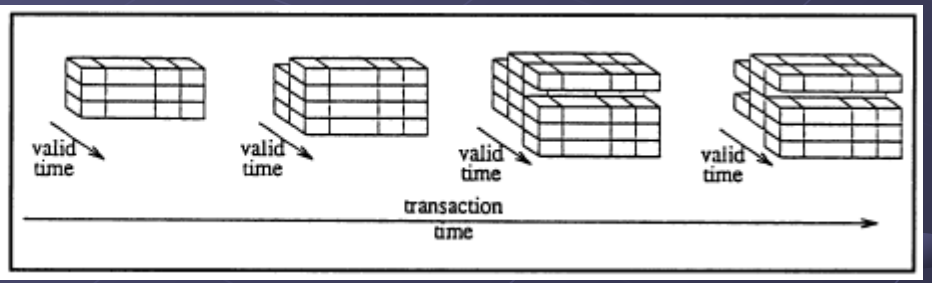


snapshot table

transaction-time table



valid-time table



bitemporal table

Dimensions of Time

- Which time dimensions are needed by an application? (what can be done and what cannot be done?)
- Consider the following example involving the career of an employee:
  1. John was hired as a programmer (PRG) with initial salary 2000 at time 1;
  2. John's salary was raised to 3000 at time 3 (but recorded in the DB at time 4);
  3. John became a database administrator (DBA) at time 6.
- Notice that 2. involves a *retroactive* update

In a Transaction-time DB

1. John was hired as a programmer (PRG) with initial salary 2000 at time 1;
2. John's salary was raised to 3000 at time 3 (but recorded in the DB at time 4);
3. John became a database administrator (DBA) at time 6.

Emp

Name	Job	Salary	TT
John	PRG	2000	[1,Now]
John	PRG	3000	[4,Now]
John	DBA	3000	[6,Now]

The time of the change 2. is incorrectly represented

In a Valid-time DB

1. John was hired as a programmer (PRG) with initial salary 2000 at time 1;
2. John's salary was raised to 3000 at time 3 (but recorded in the DB at time 4);
3. John became a database administrator (DBA) at time 6.

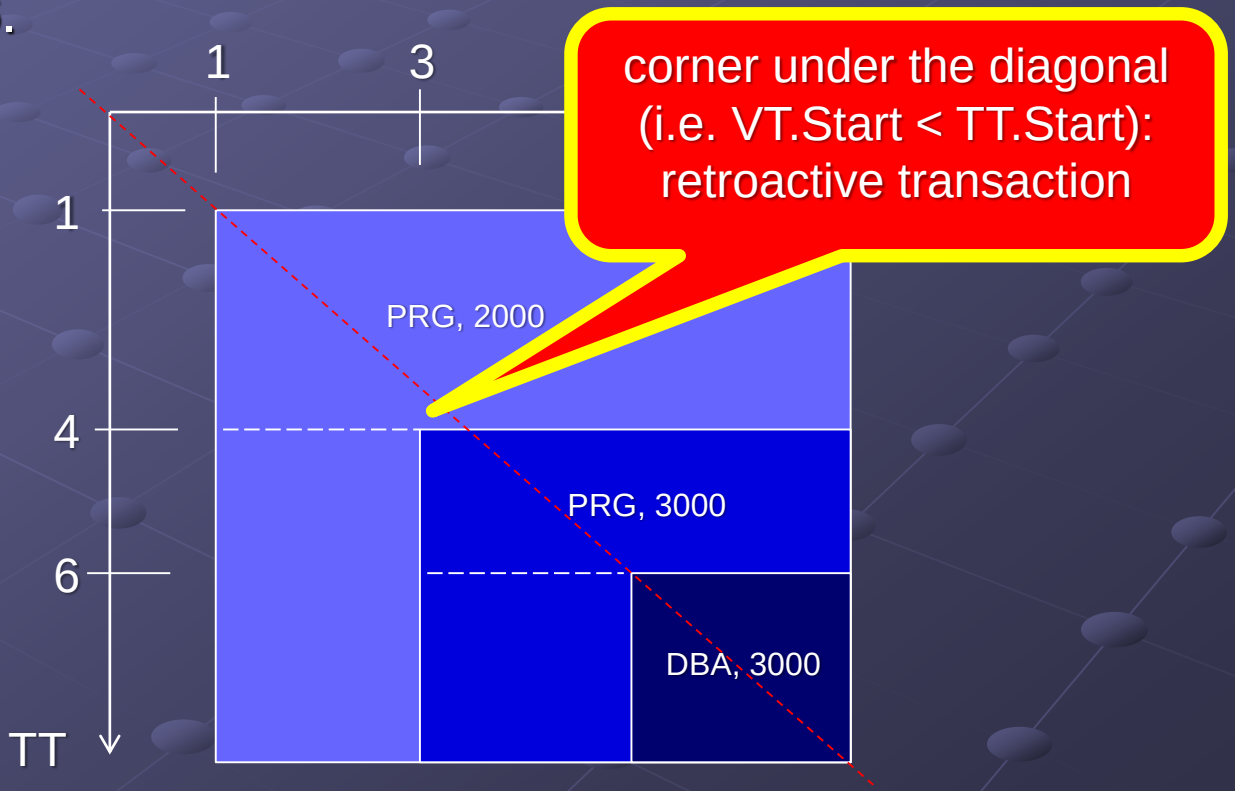
Emp

Name	Job	Salary	VT
John	PRG	2000	[1,Now]
John	PRG	3000	[3,Now]
John	DBA	3000	[6,Now]

The validity of changes is correctly represented but there is no way to know that change 2. was retroactive

In a Bitemporal DB

1. John was hired as a programmer (PRG) with initial salary 2000 at time 1;
2. John's salary was raised to 3000 at time 3 (but recorded in the DB at time 4);
3. John became a database administrator (DBA) at time 6.



In a Bitemporal DB

1. John was hired as a programmer (PRG) with initial salary 2000 at time 1;
2. John's salary was raised to 3000 at time 3 (but recorded in the DB at time 4);
3. John became a database administrator (DBA) at time 6.

Emp

Name	Job	Salary	TT	VT
John	PRG	2000	[1,Now]	[1,Now]
John	PRG	2000	[4,Now]	[1,2]
John	PRG	3000	[4,Now]	[3,Now]
John	PRG	3000	[6,Now]	[3,5]
John	DBA	3000	[6,Now]	[6,Now]

Choice of Temporal Dimensions

- A Transaction-time DB allows user to only effect immediate (on-time) transactions; proactive transactions are physically impossible and retroactive transactions store data histories with a wrong “validity”

- A Valid-time DB allows users to execute retro-/pro-active transactions (validity of modifications aka *applicability period* is expressed by users via the DML); after its execution, there is no way to know whether a transaction was immediate or retro/pro-active

- A Bitemporal DB allows users to execute retro-/proactive transactions and to keep track of their execution in the DB

Other Time Dimensions

- Event Time [Chakravarthy & Kim 94] aka Decision Time [Nascimento & Eich 95]
- Considering the event E causing the change of some data with some validity (in the mini-world and in the DB):
  - E occurs at time T in the mini-world (Decision/Event Time)
  - E occurs at time T' ≥ T in the DB (Transaction Time)
- E/D-T vs VT (=,<,>): current, futuristic, past due
- TT vs VT (=,<,>): immediate, proactive, retroactive
- E/D-T vs TT (=,<,>): instantaneous, late, N/A

- In specific application domains, other time dimensions can be of interest (e.g. Efficacy time in the legal field)

Event vs State Temporal Relations

- Moreover, in a TDB there can be two kinds of temporal relations:
  - Event tables, with instant timestamps (store information about facts without duration)
  - State tables, with period or element timestamps
- Event table are suitable to store measures, sensor data, departure/transit/arrival times

Departures

Flight	Time
100	2015-08-01 12:30
55	2015-09-10 11:15
256	2016-01-01 16:40

Temporal Relations

- In the following, we focus on state tables

- An implicit continuity assumption is often done *(data values as produced by an insertion or update are assumed to persist until they are changed or deleted, e.g. salary of an employee)*

Emp

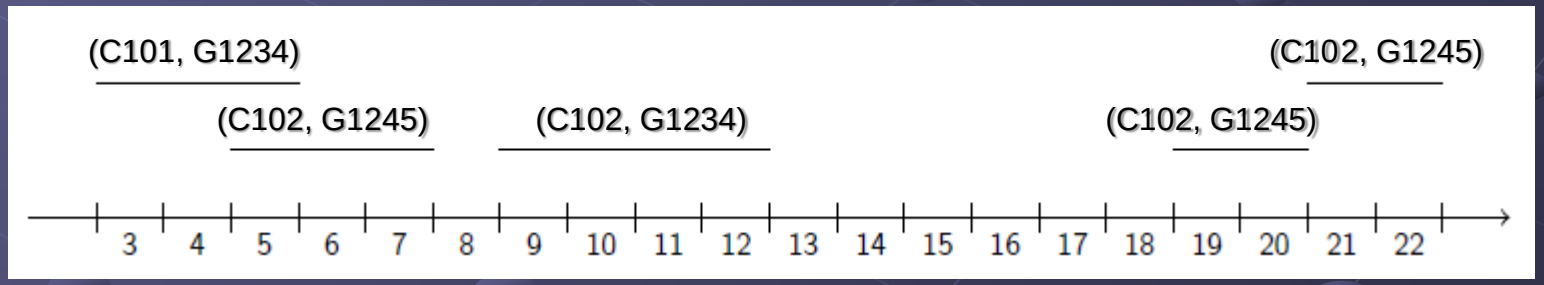
Name	Dept	Salary	Time
Tom	SE	2300	[1/1/12, 1/1/16)
Ann	DB	3200	[1/1/10, 1/1/15)
Ann	DB	3400	[1/1/15, Now]

Timestamping

- A timestamp is a value that is associated with data in a database
  - Captures some temporal aspect, e.g. valid time, transaction time
  - Represented as one or more attributes/columns of a relation
- Three different types of timestamps are widely used
  - Time points
  - Time periods
  - Temporal elements
- Two different ways of timestamping
  - Tuple timestamping
  - Attribute timestamping
- Temporally grouped models are not based on timestamping (but adopt a functional approach similar to attribute timestamping though)

Timestamping

- **Example:** Videogame store where customers, identified by a CustID, rent videogames, identified by a GameNo. Consider the following rentals during May 2015:
  - On 3rd of May, customer C101 rents game G1234 for three days
  - On 5th of May, customer C102 rents game G1245 for 3 days
  - From 9th to 12th of May, customer C102 rents game G1234
  - From 19th to 20th of May, and again from 21st to 22nd of May, customer C102 rents game G1245
- These rentals are stored in a relation CheckOut which is graphically illustrated below



Tuple Timestamping with Points

- Point-based data model: each tuple is timestamped with a time point/instant
  - Most basic and simple data model
  - Timestamps are atomic values that can be easily compared, using =, <>, >, <, >=, <=
  - Multiple tuples are used if a fact is valid at several time points
  - Syntactically different relations store different information
  - Provides an *abstract view* of a DB and is not meant for physical implementation
  - Conceptual simplicity and computational complexity make it popular for theoretical studies

CustID	GameNo	Time
C101	G1234	3
C101	G1234	4
C101	G1234	5
C102	G1245	5
C102	G1245	6
C102	G1245	7
C102	G1234	9
C102	G1234	10
C102	G1234	11
C102	G1234	12
C102	G1245	19
C102	G1245	20
C102	G1245	21
C102	G1245	22

Tuple Timestamping with Points

- The reconstruction of the original relation is not always possible
- The table on the previous slide makes it impossible to determine if C102 rented G1245 once or twice in the period from 19 to 22
- Additional attributes are required, e.g. Rental to represent the individual rentals
- It is difficult to predict when an additional attribute is needed

Rental	CustID	GameNo	Time
R1	C101	G1234	3
R1	C101	G1234	4
R1	C101	G1234	5
R2	C102	G1245	5
R2	C102	G1245	6
R2	C102	G1245	7
R3	C102	G1234	9
R3	C102	G1234	10
R3	C102	G1234	11
R3	C102	G1234	12
R4	C102	G1245	19
R4	C102	G1245	20
R5	C102	G1245	21
R5	C102	G1245	22

Tuple Timestamping with Periods

- Period-based (interval-based) data model: each tuple is timestamped with a time period

CheckOut

CustID	GameNo	Time
C101	G1234	[3,5]
C102	G1245	[5,7]
C102	G1234	[9,12]
C102	G1245	[19,20]
C102	G1245	[21,22]

- Timestamps are atomic values that can be compared using Allen's 13 basic relationships between periods (before, meets, during, etc.)
  - More convenient than comparing the endpoints of the periods
  - The benefits of Allen's predicates are relatively small

Tuple Timestamping with Periods

- The start and end of an interval are distinguished change points
- The Rental attribute is not needed to distinguish different
- checkouts
- Multiple tuples are used if a fact is valid over disjoint time periods
- Cannot model a single checkout with a gap
- The most popular model from an implementation perspective (even in SQL89, with two columns Start, End)
- Time periods are not closed under all set operations
  - Ex. subtracting [5, 7] from [1, 9] returns a set of periods { [1, 4], [8, 9] }

Tuple Timestamping with Temporal Elements

- Data model with temporal elements:
  each tuple is timestamped with a temporal element, that is a finite set of time periods

CheckOut

CustID	GameNo	Time
C101	G1234	{ [3,5] }
C102	G1245	{ [5,7], [19,22] }
C102	G1234	{ [9,12] }

CheckOut

CustID	GameNo	Time
C101	G1234	[3,5]
C102	G1245	[5,7] U [19,22]
C102	G1234	[9,12]

- The full history of a fact is stored in one tuple
- Usually the periods of a temporal element must be disjoint and non-adjacent (i.e. element = union of maximal disjoint periods). This makes it similar to point timestamps

Attribute Timestamping

- Attribute value timestamping: each attribute value is timestamped with a set of time points/periods
- All information about a real-world object is captured in a single tuple
  - e.g. all information about a customer in a tuple of the relation below; each tuple is timestamped with a temporal element, that is a finite set/union of time periods

CheckOut

CustID	Rental	GameNo
C101 { [3,5] }	R1 { [3,5] }	G1234 { [3,5] }
C102 { [5,7], [9,12], [19,22] }	R2 { [5,7] } R3 { [9,12] } R4 { [19,20] } R5 { [21,22] }	G1245 { [5,7], [19,22] } G1234 { [9,12] }

Attribute Timestamping

- Notice that a single tuple may record multiple facts
  - e.g. the second tuple records the following facts: rental information for customer C102 for the games G1245 and G1234, and four different checkouts

CheckOut

CustID	Rental	GameNo
C101 { [3,5] }	R1 { [3,5] }	G1234 { [3,5] }
C102 { [5,7], [9,12], [19,22] }	R2 { [5,7] } R3 { [9,12] } R4 { [19,20] } R5 { [21,22] }	G1245 { [5,7], [19,22] } G1234 { [9,12] }

- Non-first-normal-form (N1NF) data model
- In a previous terminology:
  - Homogeneous model → tuple timestamping
  - Inhomogeneous model → attribute timestamping

Attribute Timestamping

- Different groupings of the information into tuples are possible for attribute-value timestamping
  - Information about other objects is spread across several tuples (e.g. information about videogames)
  - e.g. regrouping the CheckOut table on GameNo in the example below

CheckOut

CustID	Rental	GameNo
C101 { [3,5] } C102 { [9,12] }	R1 { [3,5] } R3 { [9,12] }	G1234 { [3,5], [9,12] }
C102 { [5,7], [19,22] }	R2 { [5,7] } R4 { [19,20] } R5 { [21,22] }	G1245 { [5,7], [19,22] }

(such an operation is, in general, problematic!)

Temporally Grouped Model

- In a temporally grouped (or history-oriented) data model
  - the temporal dimension is implicit in the structure of data representation
  - data objects are substituted by their *histories* (ID not necessary)
  - attributes can be regarded as *partial functions* that map time into data domains

Rental	CustID	GameNo
R1	{ [3,5] } → C101	{ [3,5] } → G1234
R2	{ [5,7] } → C102	{ [5,7] } → G1245
R3	{ [9,12] } → C102	{ [9,12] } → G1234
R4	{ [19,20] } → C102	{ [19,20] } → G1245
R5	{ [21,22] } → C102	{ [21,22] } → G1245

- Temporal models based on addition of timestamping columns can be considered *ungrouped*

Temporally Grouped Model

- A temporally grouped model is strictly more *expressive* than an ungrouped data model
  - Ex. If we project the CheckOut relation on CustID:

CustID
{ [3,5] } → C101
{ [5,7] } → C102
{ [9,12] } → C102
{ [19,20] } → C102
{ [21,22] } → C102

We still know that such tuples involve 5 different rentals: the last two tuples do not merge as they belong to different groups (i.e. checkouts)

In an ungrouped models the last two tuples can be coalesced and we lose such information

- A temporally grouped model is difficult to implement
  - History IDs (e.g. surrogates) are needed to represent grouped data in a 1NF relation
  - Operations (e.g. join) are problematic to define with HIDs
  - A N1NF (e.g. XML) database would be needed

Point- versus Period-based Data Model

- In a point-based data model, truth value of facts is associated to time points
- Tuple timestamping with periods (or elements) can be used as a compact representation or normalization tool
  - Adjacent or overlapping value-equivalent tuples can be coalesced to obtain a canonical representation
- A fact true in [S,E] is true at any instant $t \in [S,E]$

- In a period-based (or interval-based) data model, period timestamps are first-class objects and truth value of facts can be associated to whole time periods

Period-based Data Model

- In a *weak interpretation*, period timestamps are first-class objects
- Although the truth value of facts is point-based, it is important to preserve (e.g. for lineage/provenance management) the individuality of period boundaries through operations, as they are reminiscent of change events (initiation and termination)
  - Ex. promotion or retirement for salary changes
- In a *strong interpretation*, period timestamps are used to represent telic facts

Atelic versus Telic Temporal Data

- Atelic data is temporal data that describe facts that do not involve a goal or culmination (e.g. have a job, salary)
- Atelic data enjoy the downward and upward inheritance properties
  - Downward inheritance: fact valid in period T is also valid in any subset of T (and at any instant of T)
  - Upward inheritance: a fact valid in consecutive or overlapping periods T1 and T2 is also valid in T1 U T2
- Telic data are temporal data for which downward and upward inheritance properties do not hold
- Telic data represent accomplishments or achievements
- Examples of telic facts:
  - the Golden Gate bridge was built from January 1933 to April 1937
  - John had a phleboclysis of 500mg of drug X from 10:30 to 11:45

The Bitemporal Conceptual Data Model

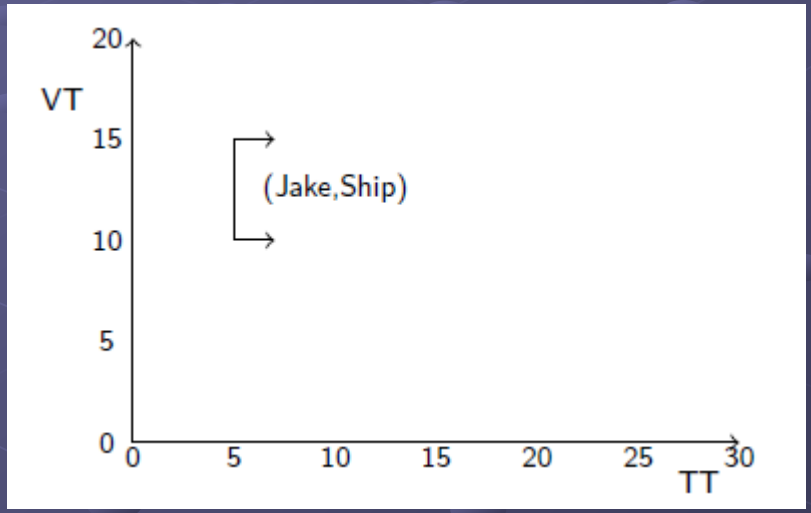
- The goal of the Bitemporal Conceptual Data Model (BCDM) is to capture the essential semantics of time-varying relations
  - The BCDM is not intended for presentation, storage, or query evaluation purposes
  - The goal of the BCDM is similar to the goal of abstract temporal databases
  - Chomicki [2009] proposed the notions of abstract and concrete temporal databases to separate semantics and representation
- Semantics associated with periods is not possible in the BCDM (it is a point-based data model)

The Bitemporal Conceptual Data Model

- Bitemporal Conceptual Data Model (BCDM)
  - Supports valid time and transaction time
  - Both time domains are linear and discrete
    - Valid-time domain: $D_{VT} = \{ t_1, t_2, \ldots, t_k \}$
    - Transaction-time domain: $D_{TT} = \{ t'_1, t'_2, \ldots, t'_j \} \cup \{now\}$
  - A bitemporal chronon is a pair of a transaction-time chronon and a valid-time chronon
    - $(t_i , t_j) \in D_{TT} \times D_{VT}$
    - "tiny rectangle" in the two-dimensional space
  - A bitemporal element is a set of bitemporal chronons
  - Timestamp attribute T with domain of bitemporal elements
  - Explicit (non-timestamp) attributes
    - Names: $D_A = \{ A_1, A_2, \ldots, A_n \}$
  - BCDM schema: $(A_1, A_2, \ldots, A_n, T)$
  - BCDM tuple: $(a_1, a_2, \ldots, a_n, t_b)$
  - Value-equivalent tuples (tuples with identical explicit attributes) are not allowed
    - the full history of a fact is contained in a single tuple

The Bitemporal Conceptual Data Model

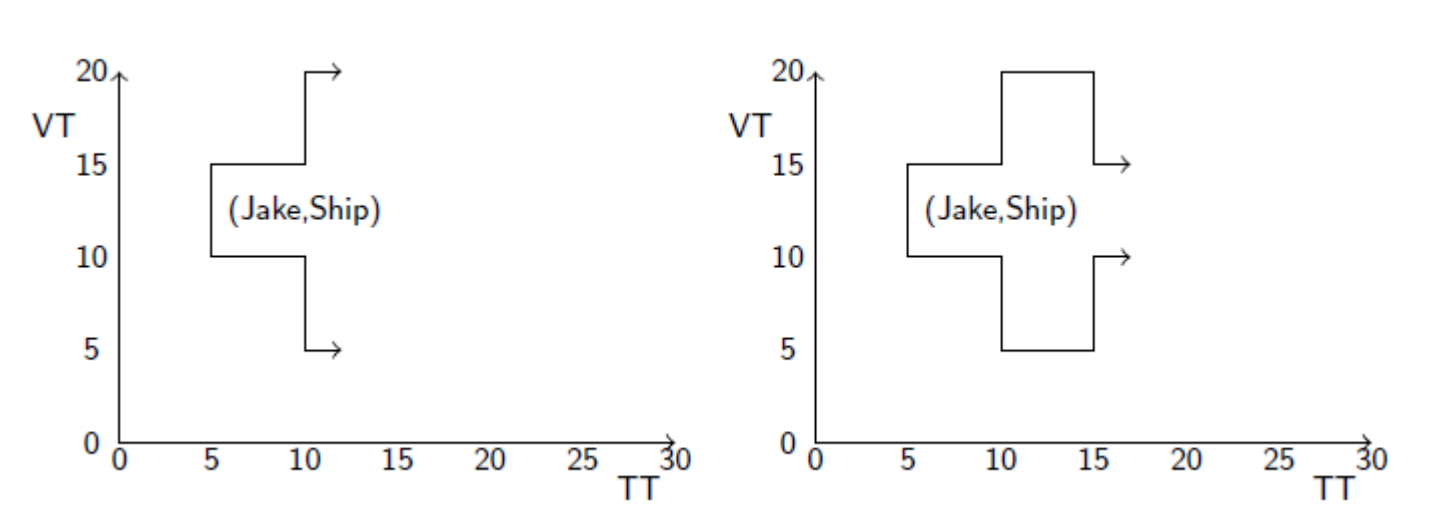
- Example: Consider a relation recording empolyee/department information
  - Employee Jake was hired in the shipping department for the period from time 10 to time 15
  - This fact became current in the database at time 5



  - Arrows indicate that the tuple has not been deleted yet

The Bitemporal Conceptual Data Model

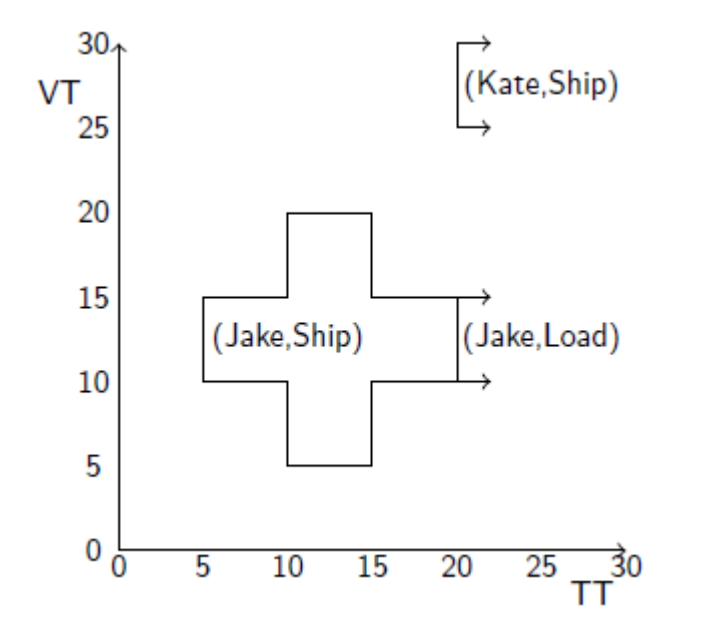
- Example (contd.)
  - The personnel office discovers that Jake had really been hired from time 5 to time 20
  - The database is corrected beginning at time 10



  - Later on at time 15 the HR department has been informed that the original time was correct

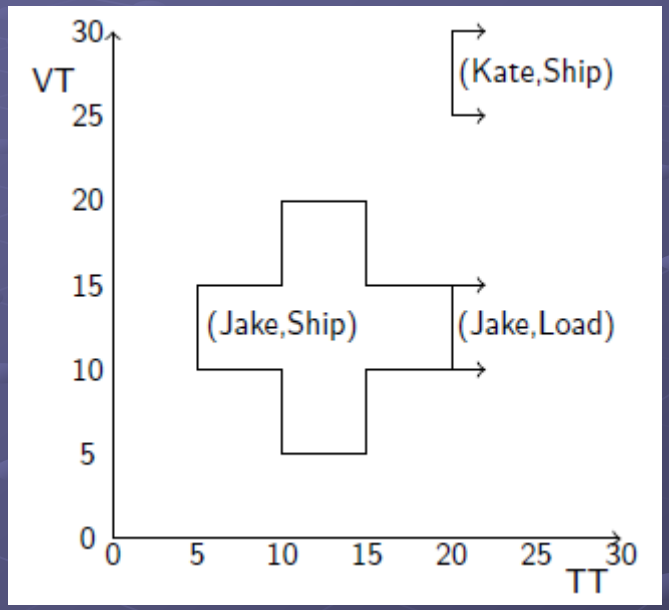
The Bitemporal Conceptual Data Model

- Example (contd.) At time point 19 the following updates are performed (updates shall become effective at time 20):
  - Jake was not in the shipping department, but in the loading department
    - The fact (Jake,Ship) is removed from the current state, and the fact (Jake,Load) is inserted
  - A new employee Kate is hired for the shipping department for the time from 25 to 30



The Bitemporal Conceptual Data Model

- After the updates the bitemporal relation contains 3 facts and is given below



Emp	Dept	T
Jake	Ship	{(5,10),…,(5,15),…,(9,10),…,(9,15), (10,5),…,(10,20),…,(14,5),…,(14,20), (15,10),…,(15,15),…,(19,10),…,(19,15)}
Jake	Load	{(now,10),…,(now,15)}
Kate	Ship	{(now,25),…,(now,30)}

Updates in the BCDM

- Update operations
  - New facts with a given valid timestamp are inserted to a relation with now as transaction time chronon
  - As time passes by, the bitemporal elements associated with current facts are updated
  - Facts are (logically) deleted by removing the chronons containing now

Updates in the BCDM

- Insert: Record in a relation r a currently unrecorded fact $(a_1, a_2, \ldots, a_n)$ with validity t_v
- Three cases are distinguished:
  1. If $(a_1, a_2, \ldots, a_n)$ was never recorded, a new tuple is appended
  2. If $(a_1, a_2, \ldots, a_n)$ was part of some previously current state, the tuple recording is updated
  3. If $(a_1, a_2, \ldots, a_n)$ is already current in the database, a modification is required (and the insertion is rejected)

$$
insert(r, (a_1, \ldots, a_n), t_v) =
$$

$$
\begin{cases}
r \cup \{(a_1, \ldots, a_n | \{now\} \times t_v)\} & \text{if } \nexists t_b((a_1, \ldots, a_n | t_b) \in r) \\
r - \{(a_1, \ldots, a_n | t_b)\} & \\
\quad \cup \{(a_1, \ldots, a_n | t_b \cup \{\{now\} \times t_v\})\} & \text{if } \exists t_b((a_1, \ldots, a_n | t_b) \in r \wedge \nexists (now, c_v) \in t_b) \\
r & otherwise
\end{cases}
$$

Updates in the BCDM

- ts_update: Special routine to add new chronons as time goes by
  - Applied to all bitemporal relations at each clock tick
  - Updates the timestamps to include the new transaction-time value
  - Each bitemporal chronon with a transaction time of now produces an appended bitemporal chronon with now replaced with the current transaction time

$$
\begin{aligned}
&ts\_update(r, c_t): \\
&\quad \texttt{for each } x \in r \\
&\quad\quad \texttt{for each } (now, c_v) \in x[T] \\
&\quad\quad\quad x[T] \leftarrow x[T] \cup \{(c_t, c_v)\}
\end{aligned}
$$

  - Example: Department relation at time 19 and 20

dept

Emp	Dept	T
Jake	Ship	{(5,10),...,(5,15),...,(9,10),...,(9,15), (10,5),...,(10,20),...,(14,5),...,(14,20), (15,10),...,(15,15),...,(19,10),...,(19,15)}
Jake	Load	{(now,10),...,(now,15)}
Kate	Ship	{(now,25),...,(now,30)}

dept

Emp	Dept	T
Jake	Ship	{(5,10),...,(5,15),...,(9,10),...,(9,15), (10,5),...,(10,20),...,(14,5),...,(14,20), (15,10),...,(15,15),...,(19,10),...,(19,15)}
Jake	Load	{(20,10),...,(20,15),(now,10),...,(now,15)}
Kate	Ship	{(20,25),...,(20,30),(now,25),...,(now,30)}

Updates in the BCDM

- Delete: Logical removal of a tuple from the current valid-time state
  - Delete all chronons (now, c_v) from the timestamp of the tuple (c_v is some valid-time chronon)
  - The timestamp is not expanded by subsequent invocations of ts_update, and the tuple will not appear in future valid-time states

$$delete(r, (a_1, \ldots, a_n)) = \begin{cases} r - \{(a_1, \ldots, a_n | t_b)\} \\ \quad \cup \{(a_1, \ldots, a_n | t_b - \mathtt{uc\_ts}(t_b))\} & \text{if } \exists t_b((a_1, \ldots, a_n | t_b) \in r) \\ r & otherwise \end{cases}$$

$$\text{where } \mathtt{uc\_ts}(t_b) = \{(now, c_v) | (now, c_v) \in t_b\}$$

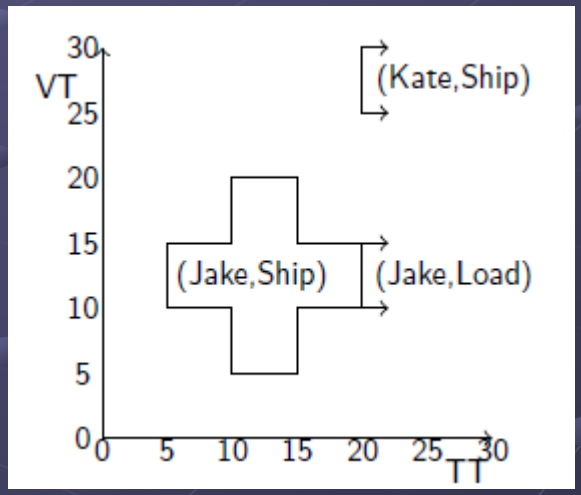
- Modify: Modification of a current tuple

$$modify(r, (a_1, \ldots, a_n), t_v) = insert(delete(r, (a_1, \ldots, a_n)), (a_1, \ldots, a_n), t_v)$$

Updates in the BCDM

- Example: The istance of the department relation dept is created by the following sequence of commands

Operation	TT
insert(dept, ("Jake","Ship"), [10,15])	5
modify(dept, ("Jake","Ship"), [5,20])	10
modify(dept, ("Jake","Ship"), [10,15])	15
delete(dept, ("Jake","Ship"))	20
insert(dept, ("Jake","Load"), [10,15])	20
insert(dept, ("Kate","Ship"), [25,30])	20



Concrete Temporal Data Models

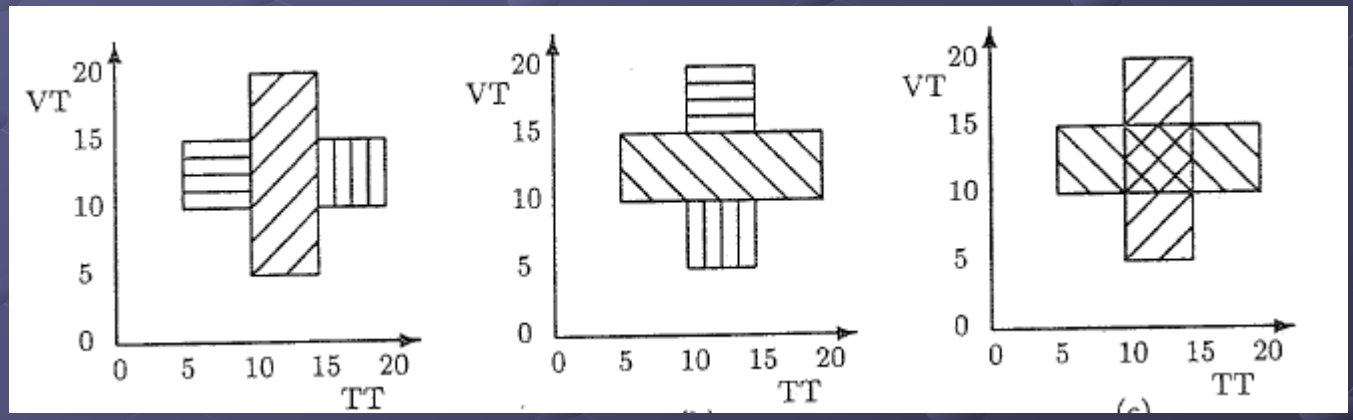
- The *abstract* Bitemporal Conceptual Data Model needs conversion into a representational or *concrete* temporal data model to be implemented in a DBMS
- The BCDM is a unifying framework for studying and comparing different temporal data models
- Mappings have been provided for most of the concrete temporal data models proposed in the literature

Tuple Timestamped Model [Snodgrass]

- Supports valid time and transaction time
- Adds four atomic-valued attributes to each relation
  - Start and end point of the valid time: Vs ,Ve
  - Start and end point of the transaction time: Ts ,Te
- Schema: R = (A1, . . . , An, Ts ,Te ,Vs ,Ve)
- Timestamping attributes Ts, Te, Vs, Ve have been also called differently (e.g. In, Out, From, To, respectively)
- Ts, Te (Vs, Ve, resp.) represent the endpoints of a transaction (valid, resp.) time period, which is usually considered open to the right
- Hence Ts ,Te ,Vs ,Ve represent the bitemporal chronons (a bitemporal chronon is a two-dimensional time point) of the corresponding rectangular region [Ts,Te) x [Vs,Ve)
- 1NF relations

Tuple Timestamped Model

- A closed region in a two dimensional space (TT x VT) must be represented by a set of rectangles
  - any bitemporal chronon in x.T is contained in at least one rectangle
  - each bitemporal chronon in a rectangle is contained in x.T
- Various coverings of a 2D area are possible:
  - Overlapping versus non-overlapping rectangles
  - Partitioning by transaction time versus partitioning by valid time



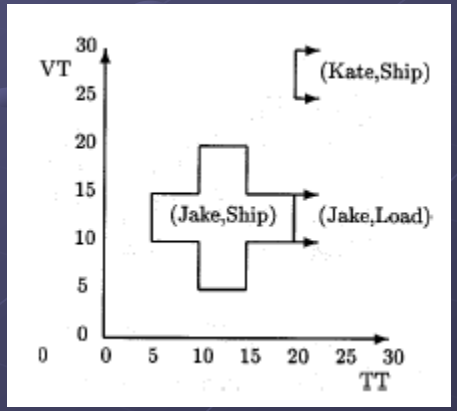
Partitioning by TT (VT) yields maximal segments in VT (TT) direction

Tuple Timestamped Model

- Example: Department relation in the tuple timestamped data model, using partitioning by transaction time

dept

Emp	Dept	Ts	Te	Vs	Ve
Jake	Ship	5	10	10	15
Jake	Ship	10	15	5	20
Jake	Ship	15	20	10	15
Jake	Load	20	Now	10	15
Kate	Ship	20	Now	25	30



Once the partitioning criterion has been chosen, a unique mapping from the BCDM is defined

Backlog-based Data Model [Jensen]

- Supports valid time and transaction time
- Adds four atomic-valued attributes to each relation
  - Start and end point of the valid time (Vs ,Ve)
  - Transaction time when the tuple was inserted into the backlog (T)
  - An operation which is either insert or delete (Op)
- Schema: R = (A1, . . . , An, Vs ,Ve ,T, Op)
- Tuples in backlogs are never updated, i.e. backlogs are append-only 1NF relations
- The fact in an insertion request is current starting at the transaction's timestamp and until a matching delete request is recorded

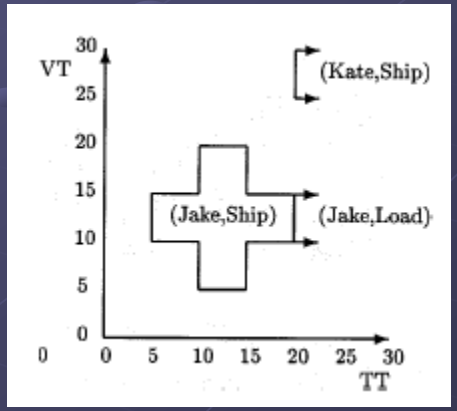
T is the commit time of the transaction executing Op

Backlog-based Data Model

- Example: Department relation in the backlog-based data model

dept

Emp	Dept	Ts	Te	T	Op
Jake	Ship	10	15	5	I
Jake	Ship	10	15	10	D
Jake	Ship	5	20	10	I
Jake	Ship	5	20	15	D
Jake	Ship	10	15	15	I
Jake	Ship	10	15	20	D
Jake	Load	10	15	20	I
Kate	Ship	25	30	20	I



Implicitly does partitioning by TT in mapping from the BCDM

Attribute Timestamped Data Model [Gadia]

- Supports valid time and transaction time
- Schema:

  R = ({(TT1 x VT1, A1)}, . . . , {(TTn x VTn, An)})
- A tuple is composed of n sets
  - Each set element is composed of a bitemporal period (e.g. [Ts,Te) x [Vs,Ve)) and an attribute value
- N1NF relations (e.g. suitable to OODB or XML)

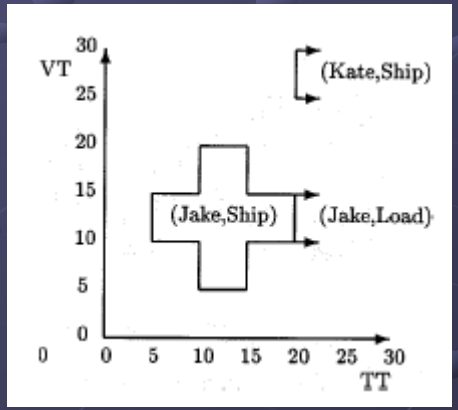
- A relation might be restructured (regrouped) on different attributes
  - For example, group by department rather than employee yields facts for each department

Attribute Timestamped Data Model

- Example: Department relation in the attribute timestamped data model

dept

Emp		Dept	
[5,10) × [10,15)	Jake	[5,10) × [10,15)	Ship
[10,15) × [5,20)	Jake	[10,15) × [5,20)	Ship
[15,20) × [10,15)	Jake	[15,20) × [10,15)	Ship
[20,Now) × [10,15)	Jake	[20,Now) × [10,15)	Load
[20,Now) × [25,30)	Kate	[20,Now) × [25,30)	Ship

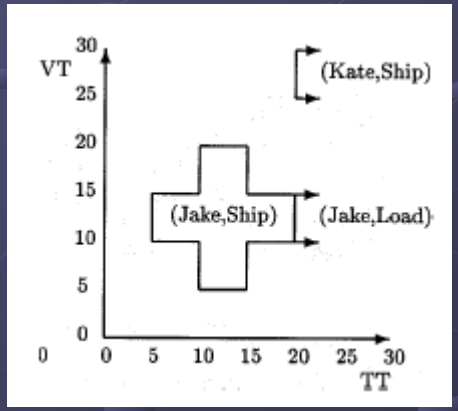


Attribute Timestamped Data Model

- Temporal elements can also be used as timestamps

dept

Emp		Dept	
{ [5,10) × [10,15), [10,15) × [5,20), [15,20) × [10,15), [20,Now) × [10,15) }	Jake	{ [5,10) × [10,15), [10,15) × [5,20), [15,20) × [10,15) }	Ship
		{ [20,Now) × [10,15) }	Load
{ [20,Now) × [25,30) }	Kate	{ [20,Now) × [25,30) }	Ship



The mapping from the BCDM is univocally defined once we have chosen the form of the periods (e.g. closed or open to the right)