

Abstract and Concrete Temporal Data Models SL03

- ▶ Modeling temporal data
- ▶ Reduction to snapshots
- ▶ The bitemporal conceptual data model
- ▶ Concrete temporal data models

Table of Contents

- ▶ **Modeling temporal data**
 - ▶ Dimensions of time
 - ▶ Timestamping data
- ▶ Reduction to snapshots
 - ▶ Snapshot equivalence
 - ▶ Snapshot reducibility
- ▶ The bitemporal conceptual data model
 - ▶ The bitemporal conceptual data model (BCDM)
 - ▶ Updates in the BCDM
 - ▶ Relational Algebra for BCDM
- ▶ Concrete temporal data models
 - ▶ Snodgrass' tuple timestamping model
 - ▶ Jensen's backlog model
 - ▶ Ben-Zvi's tuple timestamped model
 - ▶ Gadia's attribute value timestamped model

Temporal Data Models/1

- ▶ **Data model:** $M = (DS, QL)$
 - ▶ DS is a set of data structures
 - ▶ QL is a language for querying and updating the data structures
- ▶ Example: relational data model is composed of relations and SQL
- ▶ Many extensions of the relational data model to support time have been proposed
- ▶ Several modeling aspects have to be considered
 - ▶ Different time dimensions
 - ▶ Different timestamp types
 - ▶ Attribute versus tuple timestamping

Temporal Data Models/2

The different modeling aspects lead to subtle and difficult issues. There are pros and cons in all cases. No single choice seems best and after more than 20 years no consensus has been reached!

- ▶ Which time dimensions do we want?
 - ▶ Transaction time, valid time, decision time, ...
- ▶ Which timestamps shall we use?
 - ▶ points, periods, sets of points, ...
- ▶ Which information shall be timestamped?
 - ▶ objects, facts, part of objects, ...

Dimensions of Time/1

- ▶ Time is **multi-dimensional**
 - ▶ Valid time
 - ▶ Transaction time
 - ▶ Publication time
 - ▶ Efficacy time
 - ▶ Decision time
 - ▶ etc.
- ▶ The key question: 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.
- ▶ Transaction time is about to be incorporated into commercial database systems (law requires accountability, etc).

Dimensions of Time/2

- ▶ **Valid time:** The time a fact was/is/will be true in the modeled reality/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, 2004
 - ▶ Valid time captures the time-varying states of the mini-world
 - ▶ All facts have a valid time by definition, however, it might not be recorded 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)

Dimensions of Time/3

- ▶ **Transaction time:** The time when a fact is current/present in the database as stored data
 - ▶ Example: the fact “John was hired on October 1, 2004” was stored in the DB on October 5, 2004, and has been deleted on March 31, 2005
 - ▶ 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’s 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 automatically by the DBMS

Dimensions of Time/4

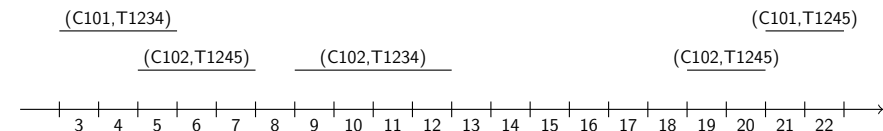
- ▶ A data model can support none, one, two, or more of these time dimensions
 - ▶ **Snapshot** data model: None of the time dimensions is supported
 - ▶ Represents a single snapshot of the reality and the database
 - ▶ **Valid-time** data model: Supports only valid time
 - ▶ **Transaction-time** data model: Supports only transaction time
 - ▶ **Bitemporal** data model: Supports valid time and transaction time
- ▶ An important issue is what shall be the basic element of a temporal data model:
 - ▶ objects, relations, classes, tuples, etc
 - ▶ closure is important

Timestamping/1

- ▶ 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

Timestamping/2

- ▶ **Example:** Video store where customers, identified by a `CustID`, rent video tapes, identified by a `TapeNum`. Consider the following rentals during May 1997:
 - ▶ On 3rd of May, customer C101 rents tape T1234 for three days
 - ▶ On 5th of May, customer C102 rents tape T1245 for 3 days
 - ▶ From 9th to 12th of May, customer C102 rents tape T1234
 - ▶ From 19th to 20th of May, and again from 21st to 22nd of May, customer C102 rents tape T1245
- ▶ These rentals are stored in a relation `CheckOut` which is graphically illustrated below



Tuple Timestamping with Points/1

- ▶ **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 $=, \neq, <, >, \geq, \leq$.
 - ▶ 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 database and is not meant for physical implementation.
 - ▶ Conceptual simplicity and computational complexity make it popular for theoretical studies.

CheckOut		
CustID	TapeNum	T
C101	T1234	3
C101	T1234	4
C101	T1234	5
C102	T1245	5
C102	T1245	6
C102	T1245	7
C102	T1234	9
C102	T1234	10
C102	T1234	11
C102	T1234	12
C102	T1245	19
C102	T1245	20
C102	T1245	21
C102	T1245	22

Tuple Timestamping with Points/2

- ▶ The reconstruction of the original relation is not always possible.
- ▶ The table on the previous slide makes it impossible to determine if C102 rented T1245 once or twice in the period from 19 to 22.
- ▶ Additional attributes are required, e.g., `SeqNo` to represent the 2-day rentals.
- ▶ It is difficult to predict when an additional attribute is needed.

CheckOut			
SeqNo	CustID	TapeNum	T
1	C101	T1234	3
1	C101	T1234	4
1	C101	T1234	5
2	C102	T1245	5
2	C102	T1245	6
2	C102	T1245	7
3	C102	T1234	9
3	C102	T1234	10
3	C102	T1234	11
3	C102	T1234	12
4	C102	T1245	19
4	C102	T1245	20
5	C102	T1245	21
5	C102	T1245	22

Tuple Timestamping with Periods/1

- **Period-based data model** (usually the term interval-based is used): each tuple is timestamped with a **time period**.

CheckOut

CustID	TapeNum	T
C101	T1234	[3,5]
C102	T1245	[5,7]
C102	T1234	[9,12]
C102	T1245	[19,20]
C102	T1245	[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/2

- The start and end of an interval are distinguished **change points**.
- The SeqNo 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.
- Time periods are not closed under all set operations.
 - Example: subtracting [5, 7] from [1, 9] returns a set of periods {[1, 4], [8, 9]}.

Tuple Timestamping with Temporal Elements

- **Data models with temporal elements**: each tuple is timestamped with a **temporal element**, i.e., a finite union of periods

CheckOut

CustID	TapeNum	T
C101	T1234	{[3, 5]}
C102	T1245	{[5, 7], [19, 22]}
C102	T1234	{[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. This makes it similar to point timestamps.

Attribute Timestamping/1

- **Attribute value timestamping**: each **attribute value** is timestamped with a set of time points/periods
- Goal: capture all information about a real-world object in a single tuple
 - e.g., all information about a customer in the relation below

CheckOut

SeqNo	CustID	TapeNum
{[3, 5]}	1	{[3, 5]} C101 {[3, 5]} T1234
{[5, 7]}	2	{[5, 7], [9, 12], [19, 22]} C102 {[5, 7], [19, 22]} T1245
{[9, 12]}	3	
{[19, 20]}	4	
{[21, 22]}	5	

- A single tuple may record multiple facts
 - e.g., the second tuple records the following facts: rental information for customer C102 for the tapes T1245 and T1234, and four different checkouts
- Non-first-normal-form data model

Attribute Timestamping/2

- ▶ **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 tapes.
 - ▶ e.g., grouping on tape number in the example below

CheckOut

SeqNo		CustID		TapeNum
{[3, 5]}	1	{[3, 5]}	C101	{[3, 5], [9, 12]}
{[9, 12]}	3	{[9, 12]}	C102	
{[5, 7]}	2	{[5, 7], [19, 22]}	C102	{[5, 7], [19, 22]}
{[19, 20]}	4			
{[21, 22]}	5			

Table of Contents

- ▶ Modeling temporal data
 - ▶ Dimensions of time
 - ▶ Timestamping data
- ▶ **Reduction to snapshots**
 - ▶ **Snapshot equivalence**
 - ▶ **Snapshot reducibility**
- ▶ The bitemporal conceptual data model
 - ▶ The bitemporal conceptual data model (BCDM)
 - ▶ Updates in the BCDM
 - ▶ Relational Algebra for BCDM
- ▶ Concrete temporal data models
 - ▶ Snodgrass' tuple timestamping model
 - ▶ Jensen's backlog model
 - ▶ Ben-Zvi's tuple timestamped model
 - ▶ Gadia's attribute value timestamped model

Goal of Reduction to Snapshots

- ▶ There is a (close) relationship between a temporal and a nontemporal database.
- ▶ There should be a close relationship between a temporal and a nontemporal statement.
- ▶ This is not the case (remember temporal join versus join).
- ▶ The purpose of snapshot equivalence is to establish a relationship between temporal relations if different timestamping methods are used.
- ▶ The purpose of snapshot reducibility is to establish a relationship between temporal and nontemporal statement.

Setup and Notations

- ▶ Relation schema: $R(A_1, \dots, A_n, T_S, T_E, V_S, V_E)$
- ▶ $A_1, \dots, A_n$ are the explicit attributes
- ▶ T_S, T_E, V_S, V_E are temporal attributes
 - ▶ T_S : transaction time start
 - ▶ T_E : transaction time end
 - ▶ V_S : valid time start
 - ▶ V_E : valid time end
- ▶ z^{n+4} denotes a tuple of arity $n + 4$

Timeslice

- ▶ The timeslice operator maps a temporal to a nontemporal relation.

Timeslide operators

$$\rho_{t_1}^B(r) = \{z^{(n+2)} \mid \exists x \in r(z.A = x.A \wedge z.V = x.V \wedge t_1 \in x.T)\}$$

$$\tau_{t_2}^B(r) = \{z^{(n+2)} \mid \exists x \in r(z.A = x.A \wedge z.T = x.T \wedge t_2 \in x.V)\}$$

- ▶ $\rho_{t_1}^B$ is the transaction timeslice operator. It selects the valid time relation at transaction time t_1 .
- ▶ $\tau_{t_2}^B$ is the valid timeslice operator. It selects the transaction time relation at valid time t_2 .
- ▶ $\rho_{t_1}^T$ is the transaction timeslice of a transaction time relation.
- ▶ $\tau_{t_2}^V$ is the valid timeslice of a valid time relation.

Snapshot Equivalence/1

- ▶ Many semantically equivalent representations of the same conceptual relation may exist.
- ▶ We need a mechanism to determine if two instances
 - ▶ are represented differently but have the same meaning
 - ▶ are represented differently and indeed have different meaning
- ▶ **Snapshot equivalence:** Two bitemporal relations, r and s , are snapshot equivalent, $r \stackrel{s}{=} s$, if for all times t_1 (not exceeding the current time) and for all times t_2 the following holds:

$$\tau_{t_2}^V(\rho_{t_1}^B(r)) = \tau_{t_2}^V(\rho_{t_1}^B(s))$$

Snapshot Equivalence/2

- ▶ **Example:** Two-day and four-day checkouts in the video example

checkout1

CustID	TapeNum	T
C102	T1245	[19,20]
C102	T1245	[21,22]

checkout2

CustID	TapeNum	T
C102	T1245	[19,22]

- ▶ Apply the valid-timeslice operator at each time point

$$\tau_{19}^V(\text{checkout1}) = \{(C102, T1245)\} \quad \tau_{19}^V(\text{checkout2}) = \{(C102, T1245)\}$$

$$\tau_{20}^V(\text{checkout1}) = \{(C102, T1245)\} \quad \tau_{20}^V(\text{checkout2}) = \{(C102, T1245)\}$$

$$\tau_{21}^V(\text{checkout1}) = \{(C102, T1245)\} \quad \tau_{21}^V(\text{checkout2}) = \{(C102, T1245)\}$$

$$\tau_{22}^V(\text{checkout1}) = \{(C102, T1245)\} \quad \tau_{22}^V(\text{checkout2}) = \{(C102, T1245)\}$$

- ▶ checkout1 and checkout2 are snapshot equivalent.
- ▶ checkout1 and checkout2 are syntactically different.

Reducibility of Temporal Operators/1

Task 3.2

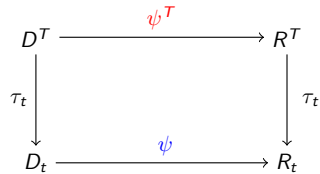
- ▶ Operators in the various data models (snapshot, transaction-time, valid-time, and bitemporal) are related.
 - ▶ The semantics of an operator in a more complex data model can be **reduced** to the semantics of the operator in a simpler data model.
- ▶ **Definition:** Let $r_1, \dots, r_n$ be temporal relations, ψ^T be an n -ary temporal operator and ψ be the corresponding n -ary non-temporal operator, Ω^T be the time domain and τ_t^T be the timeslice operator. The operator ψ^T is *snapshot reducible* to ψ iff $\forall t \in \Omega^T$

$$\tau_t^T(\psi^T(r_1, \dots, r_n)) \equiv \psi(\tau_t^T(r_1), \dots, \tau_t^T(r_n)).$$

- ▶ Reducibility guarantees that the semantics of simpler operators are preserved in their more complex counterparts.

- ▶ Snapshot reducibility reduces the semantics of **temporal operators** to the semantics of the corresponding **nontemporal operators**.
- ▶ Example: A **temporal count** shall correspond to a **count** at each point in time.
- ▶ Illustration of snapshot reducibility:

$$\forall t : \tau_t(\psi^T(D^T)) = \psi(\tau_t(D^T))$$



- ▶ D^T = temporal DB
- ▶ $\psi^T \in \{\sigma^T, \pi^T, \vartheta^T, \times^T, \cup^T, -^T\}$
- ▶ R^T = temporal result relation
- ▶ τ_t = snapshot at time point t
- ▶ D_t = snapshot of D^T at time t
- ▶ $\psi \in \{\sigma, \pi, \vartheta, \times, \cup, \}$
- ▶ R_t = result relation at time t

- ▶ Modeling temporal data
 - ▶ Dimensions of time
 - ▶ Timestamping data
- ▶ Reduction to snapshots
 - ▶ Snapshot equivalence
 - ▶ Snapshot reducibility
- ▶ **The bitemporal conceptual data model**
 - ▶ **The bitemporal conceptual data model (BCDM)**
 - ▶ **Updates in the BCDM**
 - ▶ **Relational Algebra for BCDM**
- ▶ Concrete temporal data models
 - ▶ Snodgrass' tuple timestamping model
 - ▶ Jensen's backlog model
 - ▶ Ben-Zvi's tuple timestamped model
 - ▶ Gadia's attribute value timestamped model

The Bitemporal Conceptual Data Model/1

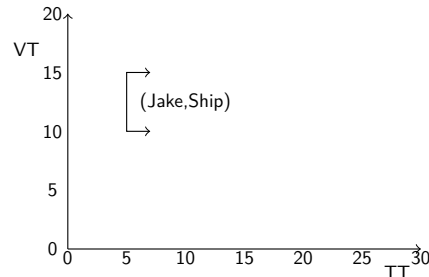
- ▶ The goal of the Bitemporal Conceptual Data Model (BCDM) is to capture the essential semantics of time-varying relations.
- ▶ The BCDM is not intended for presentation, storage, or query evaluation purposes.
- ▶ The goal of the BCDM is similar to the goal of abstract temporal databases.
- ▶ Chomicki proposed the notions of abstract and concrete temporal databases to separate semantics and representation.
- ▶ BUT (unfortunately): semantics associated with periods (intervals) is not possible in the BCDM.

The Bitemporal Conceptual Data Model/2

- ▶ **Bitemporal Conceptual Data Model (BCDM)**
 - ▶ Supports **valid time** and **transaction time**
 - ▶ Both time domains are **linear** and **discrete**
 - ▶ Valid-time domain: $D_{VT} = \{t_1, t_2, \dots, t_k\} \cup \{now\}$
 - ▶ Transaction-time domain: $D_{TT} = \{t'_1, t'_2, \dots, t'_j\} \cup \{now\}$
 - ▶ 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: $\{A_1, \dots, A_n\}$
 - ▶ BCDM schema: $(A_1, A_2, \dots, A_n, T)$
 - ▶ BCDM tuple: $x = (a_1, a_2, \dots, a_n, t_b)$
 - ▶ 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/3

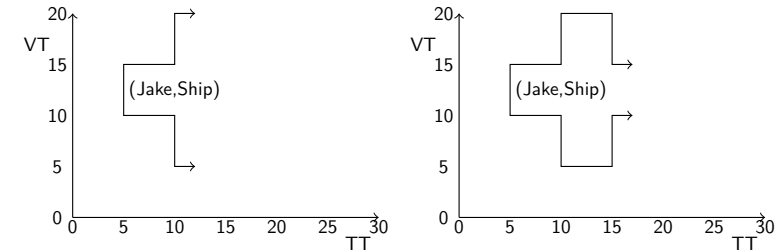
- **Example:** Consider a relation recording employee/department information.
 - Employee Jake was hired in the shipping department for the period from time 10 to time 15.
 - This fact become current in the database at time 5.



- Arrows indicate that the tuple has not yet been deleted

The Bitemporal Conceptual Data Model/4

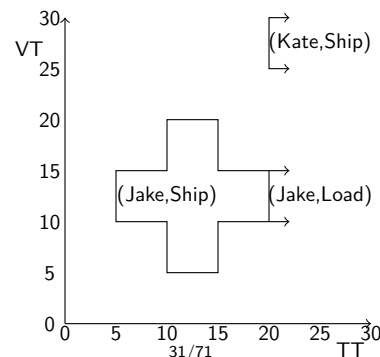
- **Example (contd.)**
 - The personnel office discovers that Jake had really been hired from time 5 to time 20.
 - The database is corrected beginning at time 10.



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

The Bitemporal Conceptual Data Model/5

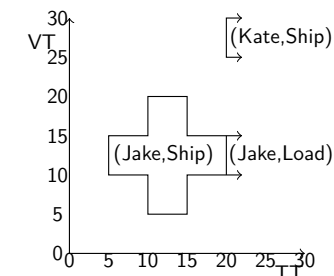
- **Example (contd.)** At time point 20 the following updates are performed:
 - Jake was not in the shipping department, but in the loading department
 - The fact (Jake,Ship) is removed from the current state, and the fact (Jake,Load) is inserted
 - A new employee Kate is hired for the shipping department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



The Bitemporal Conceptual Data Model/6

Task 3.4

- **Example (contd.)** At time 20 the bitemporal relation contains 3 facts and is given below.



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/1

► Update operations

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

Updates in the BCDM/2

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

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

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

Updates in the BCDM/3

- **ts_update:** Special routine to add new chronons as time passes 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.

$ts_update(r, c_t) :$
 for each $x \in r$
 for each $(now, c_v) \in x.T$
 $x.T \leftarrow x.T \cup \{(c_t, c_v)\}$

- **Example:** Department relation before and after $ts_update(dept, 20)$.

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

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

Updates in the BCDM/4

- **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

$delete(r, (a_1, \dots, a_n)) =$

$$\begin{cases} r - \{(a_1, \dots, a_n | t_b)\} \cup \{(a_1, \dots, a_n | t_b - uc_ts(t_b))\} & \text{if } \exists t_b((a_1, \dots, a_n | t_b) \in r) \\ r & \text{otherwise} \end{cases}$$

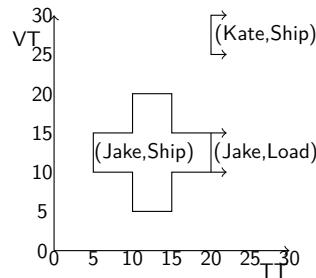
where $uc_ts(t_b) = \{(now, c_v) | (now, c_v) \in t_b\}$

- **Modification:** Modification of a current tuple

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

Updates in the BCDM/5

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



Command	TT
insert(<i>dept</i> , ("Jake", "Ship"), [10,15])	5
modify(<i>dept</i> , ("Jake", "Ship"), [5,20])	10
modify(<i>dept</i> , ("Jake", "Ship"), [10,15])	15
delete(<i>dept</i> , ("Jake", "Ship"))	20
insert(<i>dept</i> , ("Jake", "Load"), [10,15])	20
insert(<i>dept</i> , ("Kate", "Ship"), [25,30])	20

Relational Algebra for BCDM

- Algebra for the standard relational algebra operators in BCDM
 - Schema: $R = (A_1, \dots, A_n, T)$
 - Domains: A_i has domain D_i and T has domain $\mathcal{P}(D_{TT} \times D_{VT})$
 - D_{TT} is the transaction-time domain and D_{VT} is the valid-time domain
 - r is an instance relations of schema R
 - The operators then have the following signature

$$\pi_D^B : r \rightarrow r$$

$$\sigma_P^B : r \rightarrow r$$

$$\cup^B : r \times r \rightarrow r$$

$$\bowtie^B : r \times r \rightarrow r$$

$$-^B : r \times r \rightarrow r$$

$$\rho_t^B : r \rightarrow r_{vt}$$

$$\tau_t^B : r \rightarrow r_{tt}$$

Projection in BCDM

Task 3.5

- **Temporal projection:** Project a relation r with non-timestamp attributes $A_1, \dots, A_n$ to a subset D of attributes

$$\pi_D^B(r) = \{z^{(|D|+1)} \mid \exists x \in r(z.D = x.D) \wedge \forall y \in r(y.D = z.D \Rightarrow y.T \subseteq z.T) \wedge \forall t \in z.T \exists y \in r(y.D = z.D \wedge t \in y.T)\}$$

- **Example:** Temporal projection on the *Emp* attribute: $\pi_{Emp}^B(dept)$

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)}

result		T
Jake		{(5,10),..., (5,15),..., (9,10),..., (9,15), (10,5),..., (10,20),..., (14,5),..., (14,20), (15,10),..., (15,15),..., (19,10),..., (19,15), (now,10),..., (now,15)}
Kate		{(now,25),..., (now,30)}

Selection in BCDM

- **Temporal selection:** Select from relation r with non-timestamp attributes $A_1, \dots, A_n$ all tuples that satisfy a predicate P defined on the non-timestamp attributes

$$\sigma_P^B(r) = \{z \mid z \in r \wedge P(z.A)\}$$

- **Example:** Select all tuples of employee Kate: $\sigma_{Emp=Kate}^B(dept)$

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)}

result		T
Kate	Ship	{(now,25),..., (now,30)}

Union in BCDM/1

- **Temporal union:** Compute the union of tuples from two relations r_1 and r_2 that are instances of the same schema (or union-compatible schemas)

$$r_1 \cup^B r_2 = \{z^{(n+1)} \mid$$

$$\exists x \in r_1 \exists y \in r_2 (z.A = x.A = y.A \wedge z.T = x.T \cup y.T) \vee$$

$$\exists x \in r_1 (z.A = x.A \wedge \neg \exists y \in r_2 (y.A = x.A) \wedge z.T = x.T) \vee$$

$$\exists y \in r_2 (z.A = y.A \wedge \neg \exists x \in r_1 (x.A = y.A) \wedge z.T = y.T) \}$$

- The first clause handles value-equivalent tuples found in r_1 and r_2 .
- The second clause handles those tuples that are found only in r_1 .
- The third clause handles those tuples that are found only in r_2 .

Union in BCDM/2

- **Example:** Compute union of *dept* and *emp*: $dept \cup^B emp$

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)}

emp		T
Jake	Ship	{(5,20),..., (5,25),..., (9,20),..., (9,25) }
Sue	Load	{(5,20),..., (5,25),..., (9,20),..., (9,25) }

result		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), (5,20),..., (5,25),..., (9,20),..., (9,25) }
Jake	Load	{(now,10),..., (now,15)}
Kate	Ship	{(now,25),..., (now,30)}
Sue	Load	{(5,20),..., (5,25),..., (9,20),..., (9,25) }

Difference in BCDM/1

- **Temporal difference:** Computes those tuples that are in r_1 and not in r_2 , where the two relations are instances of the same schema (or union-compatible schemas)

$$r_1 -^B r_2 = \{z^{(n+1)} \mid \exists x \in r_1 ($$

$$z.A = x.A \wedge$$

$$(\exists y \in r_2 (z.A = y.A \wedge z.T = x.T - y.T \wedge z.T \neq \emptyset) \vee$$

$$\neg \exists y \in r_2 (z.A = y.A \wedge z.T = x.T)) \}$$

- The last two lines compute the bitemporal element, depending on whether a value-equivalent tuple may be found in r_2 or not.

Difference in BCDM /2

- **Example:** Compute difference between *dept* and *emp*: $dept -^B emp$

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)}

emp		T
Jake	Ship	{(10,5),..., (10,20),..., (14,5),..., (14,20) }
Jake	Load	{(15,10),..., (15,15),..., (now,10),..., (now,15) }

result		T
Jake	Ship	{(5,10),..., (5,15),..., (9,10),..., (9,15), (15,10),..., (15,15),..., (19,10),..., (19,15) }
Kate	Ship	{(now,25),..., (now,30)}

Join in BCDM/1

- **Temporal join:** Two tuples join if they match on the join attributes $A_1 \dots A_n$ and have overlapping bitemporal-element timestamps.
 - r and s are instances over the following schemas:

$$\begin{aligned} R &= (A_1, \dots, A_n, B_1, \dots, B_l, T) \\ S &= (A_1, \dots, A_n, C_1, \dots, C_m, T) \end{aligned}$$

$$\begin{aligned} r \bowtie^B s &= \{z^{(n+l+m+1)} \mid \exists x \in r \exists y \in s (\\ &\quad x.A = y.A \wedge x.T \cap y.T \neq \emptyset \wedge \\ &\quad z.A = x.A \wedge z.B = x.B \wedge z.C = y.C \wedge \\ &\quad z.T = x.T \cap y.T) \} \end{aligned}$$

- The timestamp of a result tuple is the intersection of the timestamps of the corresponding argument tuples

Timeslice Operators for BCDM/1

- **Transaction-timeslice** operator: selects the relation at transaction-time t_1 (which is not exceeding the current time)
 - Gets a bitemporal relation r as input and returns a valid-time relation

$$\rho_{t_1}^B(r) = \{z^{(n+1)} \mid \exists x \in r (z.A = x.A \wedge z.T_v = \{t_1 \mid (t_1, t_2) \in x.T\})\}$$

- **Valid-timeslice** operator: selects the relation at valid-time t_2
 - Gets a bitemporal relation r as input and returns a transaction-time relation

$$\tau_{t_2}^B(r) = \{z^{(n+1)} \mid \exists x \in r (z.A = x.A \wedge z.T_t = \{t_2 \mid (t_1, t_2) \in x.T\})\}$$

- Timeslice operators can be extended for transaction-time and valid-time relations
 - ρ^T gets as input a transaction-time relation and returns a snapshot relation
 - τ^V gets as input a valid-time relation and returns a snapshot relation

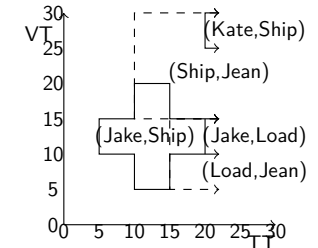
Join in BCDM /2

- **Example:** Bitemporal join of two relations to compute “Who managed whom”?
 - Compute $dept \bowtie^B mgr$

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)}

mgr		T
Ship	Jean	{(10,15),..., (10,30),..., (now,15),..., (now,30), (15,5),..., (15,15),..., (now,5),..., (now,15)}
Load	Jean	{(15,5),..., (15,15),..., (now,5),..., (now,15)}

result			T
Jake	Ship	Jean	{(10,15),..., (10,20),..., (15,15),..., (15,20), (15,5),..., (15,15),..., (now,5),..., (now,15)}
Jake	Load	Jean	{(now,10),..., (now,15)}
Kate	Ship	Jean	{(now,25),..., (now,30)}



- Timestamp is the overlap of timestamp regions of tuples with matching join attributes

Timeslice Operators for BCDM/2

- **Example:** Department relation

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)}

$$\rho_7^B(dept) = \{(Jake, Ship, \{10, 11, 12, 13, 14, 15\})\}$$

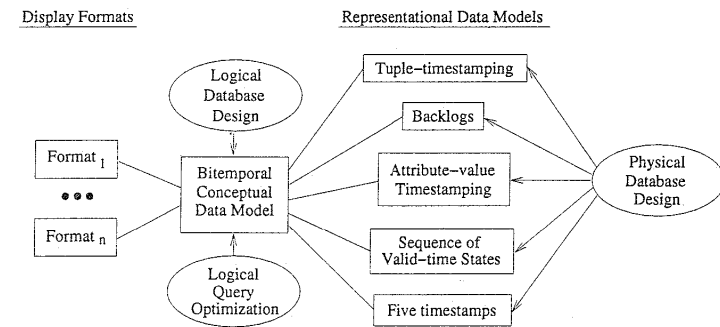
$$\begin{aligned} \tau_{12}^B(dept) &= \{(Jake, Ship, \\ &\quad \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}), \\ &\quad (Jake, Load, \{now\})\} \end{aligned}$$

Table of Contents

- ▶ Modeling temporal data
 - ▶ Dimensions of time
 - ▶ Timestamping data
- ▶ Reduction to snapshots
 - ▶ Snapshot equivalence
 - ▶ Snapshot reducibility
- ▶ The bitemporal conceptual data model
 - ▶ The bitemporal conceptual data model (BCDM)
 - ▶ Updates in the BCDM
 - ▶ Relational Algebra for BCDM
- ▶ **Concrete temporal data models**
 - ▶ **Snodgrass' tuple timestamping model**
 - ▶ **Jensen's backlog model**
 - ▶ **Ben-Zvi's tuple timestamped model**
 - ▶ **Gadia's attribute value timestamped model**

The BCDM versus Concrete Data Models

- ▶ A BCDM relation is conceptually and structurally simple
 - ▶ a set of facts, each timestamped with a bitemporal element
- ▶ Aimed for the design and as a basis for query languages, rather than for implementation
- ▶ Concrete data models are designed with implementation in mind



Terminology

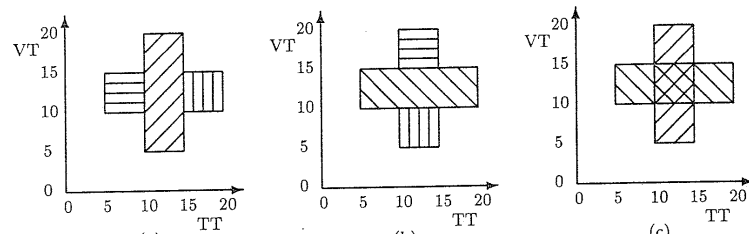
- ▶ The terms *abstract* and *conceptual* have been used for issues that are independent of the implementation.
- ▶ The terms *concrete* and *representational* have been used for issues that consider the implementation.
- ▶ We use the respective terms interchangeably and follow the most commonly used practice.
- ▶ In general it is good practice to separate semantics (abstract, conceptual) and implementation (concrete, representational).

Snodgrass' Tuple Timestamped Model/1

- ▶ **Snodgrass' tuple timestamped data model**
 - ▶ Supports valid time and transaction time
 - ▶ Adds four **atomic-valued attributes** to each relation
 - ▶ Start and end point of the valid time (V_s, V_e)
 - ▶ Start and end point of the transaction time (T_s, T_e)
 - ▶ Schema: $R = (A_1, \dots, A_n, T_s, T_e, V_s, V_e)$
 - ▶ T_s, T_e, V_s, V_e represent the bitemporal chronons of the corresponding **rectangular** region
 - ▶ 1NF relations

Snodgrass' Tuple Timestamped Model/2

- **Representation** of a BCDM tuple x in Snodgrass' concrete model
 1. Divide the region covered by the BCDM timestamp into rectangles
 2. Represent a BCDM tuple by a set of representational tuples, one for each rectangle
- **Covering function**: Divides a bitemporal region $x.T$ into rectangles such that
 - 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 bitemporal element are possible
 - Overlapping versus non-overlapping rectangles
 - Partitioning by transaction time versus partitioning by valid time



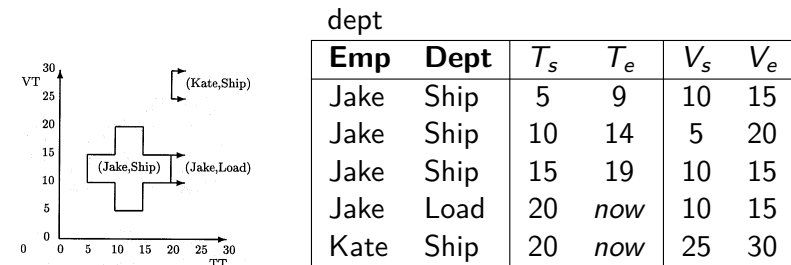
NDBS14, SL03

53/71

M. Böhlen, ifi@uzh

Snodgrass' Tuple Timestamped Model/3

- **Example**: Department relation in the Snodgrass data model, using partitioning by transaction time.



- Partitioning by transaction time yields maximal segments in valid time direction
- Partitioning by valid time yields maximal segments in transaction time direction

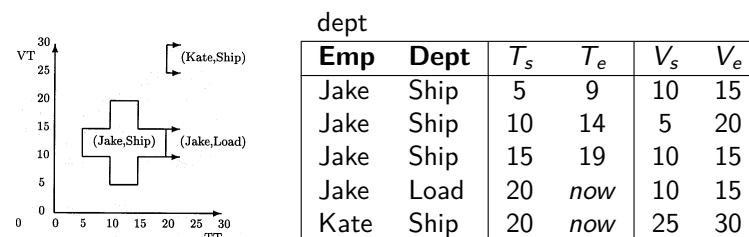
NDBS14, SL03

54/71

M. Böhlen, ifi@uzh

Now in Representational Data Models/3

The concrete relation instance (we assume Snodgrass tuple timestamped model) does not change as time passes by.



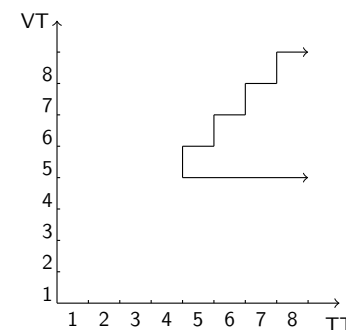
NDBS14, SL03

55/71

M. Böhlen, ifi@uzh

The Timestamp NOW/1

E-ID	EName	Salary	Dept	TT	VT
1	Jane	40	Advtg	5-now	5-now
2	John	20	Sales	5-now	3-now
1	Jane	30	Advtg	1-now	1-4
2	John	20	Sales	1-4	1-now



A growing stair shape:

- Starting at time 5, the valid time extends from 5 to now.
- At each clock tick, the period end grows by one unit.

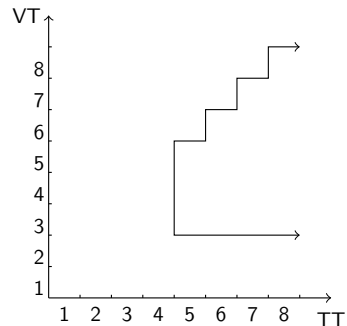
NDBS14, SL03

56/71

M. Böhlen, ifi@uzh

The Timestamp NOW/2

E-ID	EName	Salary	Dept	TT	VT
1	Jane	40	Advtg	5-now	5-now
2	John	20	Sales	5-now	3-now
1	Jane	30	Advtg	1-now	1-4
2	John	20	Sales	1-4	1-now

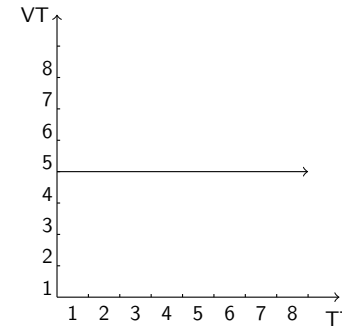


A growing stair shape with an initial big step:

- ▶ Starting at time 5, the valid time extends from 3 to now.
- ▶ At each clock tick, the period end grows by one unit.

The Timestamp NOW/3

E-ID	EName	Salary	Dept	TT	VT
1	Jane	40	Advtg	5-now	5-now
2	John	20	Sales	5-now	3-now
1	Jane	30	Advtg	1-now	1-4
2	John	20	Sales	1-4	1-now

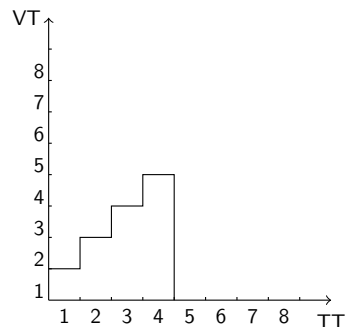


A growing period:

- ▶ Starting at time 1, the valid time extends from 1 to 4.
- ▶ The period remains part of the current state.
- ▶ The right end of the region tracks the current time.

The Timestamp NOW/4

E-ID	EName	Salary	Dept	TT	VT
1	Jane	40	Advtg	5-now	5-now
2	John	20	Sales	5-now	3-now
1	Jane	30	Advtg	1-now	1-4
2	John	20	Sales	1-4	1-now



A static stair shape:

- ▶ Starting at time 1, the valid time extends from 1 to now until time 4.
- ▶ The shape grew until time 4, but is now static.

Implications of Timestamp NOW

Task 3.9

- ▶ When *now* is stored as a timestamp then the database state changes continuously as time passes by.
- ▶ Some facts stop being valid as time passes by.
- ▶ Some facts start being valid as time passes by.
- ▶ Index structures must also be dynamic (cf. TPR-Tree)
- ▶ The database integrity might get violated as time passes by.
- ▶ Materialized views (precomputed results) get outdated as time passes by.
- ▶ The timestamp *now* cannot be approximated by NULL or infinity (e.g., a query that retrieves all periods that are between 5 and 7 years long).

Algebra for Snodgrass' Data Model/1

- ▶ Relation schema: $R = (A_1, \dots, A_n, T_s, T_e, V_s, V_e)$
 - ▶ V is a shorthand for (V_s, V_e) , T is a shorthand for (T_s, T_e)
- ▶ **Temporal projection:** Project a relation r with non-timestamp attributes $A_1, \dots, A_n$ to a subset D of these attributes (columns)

$$\pi_D^B(r) = \{z^{(|D|+4)} \mid \exists x \in r (z.D = x.D \wedge z.T = x.T \wedge z.V = x.V)\}$$
- ▶ Note:
 - ▶ The result of a projection might not be coalesced (even if the argument relation is coalesced).
 - ▶ $\{(C101, T1234, 7, now, 8, 10), (C101, T1245, 7, now, 11, 15)\}$

Algebra for Snodgrass' Data Model/2

- ▶ **Temporal natural join:** Two tuples join if they match on the join attributes and have overlapping bitemporal-element timestamps.
 - ▶ r and s are instances over the following schemas:

$$\begin{aligned} R &= (A_1, \dots, A_n, B_1, \dots, B_l, T_s, T_e, V_s, V_e) \\ S &= (A_1, \dots, A_n, C_1, \dots, C_m, T_s, T_e, V_s, V_e) \end{aligned}$$

$$r \bowtie^B s = \{z^{(n+l+m+4)} \mid (\exists x \in r \exists y \in s (z.A = x.A = y.A \wedge x.T \cap y.T \neq \emptyset \wedge x.V \cap y.V \neq \emptyset \wedge z.B = x.B \wedge z.C = y.C \wedge z.T = x.T \cap y.T \wedge z.V = x.V \cap y.V))\}$$

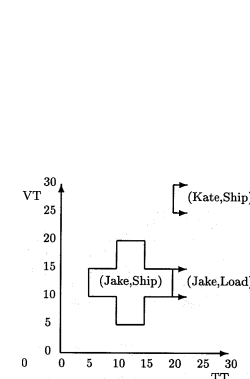
- ▶ Note:
 - ▶ The result of a temporal bitemporal join is again a bitemporal relation.
 - ▶ The original timestamps are not included in the result relation.

Jensen's Backlog-Based Data Model/1

- ▶ **Jensen's backlog-based data model**
 - ▶ Supports valid time and transaction time
 - ▶ Adds four **atomic-valued attributes** to each relation
 - ▶ Start and end point of the valid time (V_s, V_e)
 - ▶ Transaction time when the tuple was inserted into the backlog (T)
 - ▶ An operation which is either insert or delete (Op)
 - ▶ Schema: $R = (A_1, \dots, A_n, V_s, V_e, T, Op)$
 - ▶ 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.

Jensen's Backlog-Based Data Model/2

- ▶ **Example:** Department relation in Jensen's backlog model



dept

Emp	Dept	V_s	V_e	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	Ship	10	15	20	I
Kate	Ship	25	30	20	I

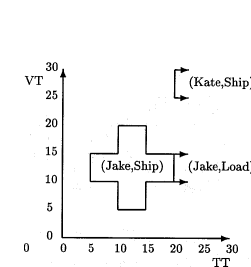
Ben-Zvi's Tuple Timestamped Data Model/1

► Ben-Zvi's Tuple Timestamped Data Model

- Adds five **atomic-valued attributes** to each relation
 - T_{es} is the *effective start* of the non-temporal attributes values
 - T_{ee} is the *effective end* of the non-temporal attributes values
 - T_{rs} is the *registration start* and stores when the T_{es} value was recorded
 - T_{re} is the *registration end* and stores when the T_{ee} value was recorded
 - T_{ee} is not necessarily recorded at the same time as T_{es} is recorded
 - T_d indicates the time when the information in the tuple was logically deleted
- The symbol “—” indicates
 - an unrecorded T_{ee} (T_{re}) value
 - in the T_d field that the tuple contains current information
- Schema: $R = (A_1, \dots, A_n, T_{es}, T_{rs}, T_{ee}, T_{re}, T_d)$
- 1NF relations

Ben-Zvi's Tuple Timestamped Data Model/2

- **Example:** Department relation in Ben-Zvi's tuple timestamped data model



dept		T_{es}	T_{rs}	T_{ee}	T_{re}	T_d
Jake	Ship	10	5	15	5	10
Jake	Ship	5	10	20	10	15
Jake	Ship	10	15	15	15	20
Jake	Load	10	20	15	20	—
Kate	Ship	25	20	30	20	—

Gadia's Attribute Value Timestamped Data Model/1

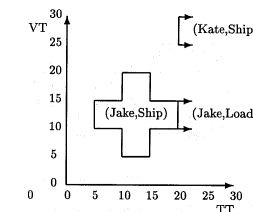
► Gadia's attribute value timestamped data model

- Supports valid time and transaction time
- Schema:

$$R = (\{([T_s, T_e] \times [V_s, V_e] A_1)\}, \dots, \{([T_s, T_e] \times [V_s, V_e] A_n)\})$$
- A tuple is composed of n sets
 - Each set element is a triple of a transaction-time period, a valid-time period, and an attribute value
- Non-1NF relations
- A relation might be restructured (grouped) on different attributes
 - for example, group by department rather than employee yields facts for each department

Gadia's Attribute Value Timestamped Data Model/2

- **Example:** Department relation in Gadia's attribute value timestamped data model



dept		Emp	Dept
$[5,9] \times [10,15]$	Jake	$[5,9] \times [10,15]$	Ship
$[10,14] \times [5,20]$	Jake	$[10,14] \times [5,20]$	Ship
$[15,19] \times [10,15]$	Jake	$[15,19] \times [10,15]$	Ship
$[20,now] \times [10,15]$	Jake	$[20,now] \times [10,15]$	Load
$[20,now] \times [25,30]$	Kate	$[20,now] \times [25,30]$	Ship

History of Temporal Data Models

Task 3.11

Data model	Citation	Time	Remark
Time Oriented Data Base Model	[Wiederhold et al. 75]	valid	Wiederhold was the first temporal model to be implemented; mainly to support medial applications.
DATA	[Kimball 78]	transaction	First implemented transaction-time model
LEGOL 2.0	[Jones & al. 79]	valid	Jones was the first to define a time-oriented algebra.
Time Relational Model	[Ben-Zvi 82]	both	First to incorporate both transaction time and valid time.
Historical Data Model	[Clifford & Warren 83]	valid	Attempted to model the semantics of natural language; associate time with both attribute values and tuples.
–	[Snodgrass & Ahn 86]	both	
Temporally Oriented Data Model	[Ariav 86]	valid	
HQuel	[Tansel 86]	valid	Operators to switch between period and point representation
Historical Relational Data Model	[Clifford & Croker 87]	valid	
TQuel	[Snodgrass 87]	both	
Temporal Data Model	[Segev & Shoshani 87]	valid	
HQL	[Sadeghi 87]	valid	
Postgres	[Stonebraker 87]	transaction	
Homogeneous Relational Model	[Gadia 88]	valid	
Heterogeneous Relational Model	[Gadia & Yeung 88]	valid	
Temporal Relational Model	[Lorentzos 88]	valid	Operators to switch between period and point representation; supports nested granularity timestamps and periodic time.
Temporal Relational Model	[Navathe & Ahmed 89]	valid	
HSQL	[Sarda 90]	valid	Operators to switch between period and point representation
DM/T	[Jensen et al. 91]	transaction	
Accounting Data Model	[Thompson 91]	both	
TempSQL	[Gaida 92]	both	

NDBS14, SL03

69/71

M. Böhlen, ifi@uzh

Summary/1

- ▶ Temporal data models choose
 - ▶ time dimensions (valid time, transaction time, ...)
 - ▶ timestamps (points, intervals, temporal elements)
 - ▶ scope of timestamps (attribute, tuple, relation, ...)
- ▶ No single best choice exist. Assessment based on chosen timestamps is difficult; must be considered together with query language.
- ▶ **Snapshot equivalence** is an important concept to compare temporal relations; it cannot be used to capture interval semantics
- ▶ **Snapshot reducibility** can be used to constrain the semantics of temporal operations and statements
- ▶ It is an open question whether periods should have a meaning beyond a set of points (point-based versus interval-based models).

NDBS14, SL03

70/71

M. Böhlen, ifi@uzh

Summary/2

- ▶ It is good to have a distinction between semantics and representation
 - ▶ conceptual/abstract versus concrete/representational
- ▶ The BCDM attempts to define the semantics independent of presentation, storage, performance. It is point-based and cannot capture interval semantics.
- ▶ There are a number of different concrete models:
 - ▶ Snodgrass' tuple timestamping model
 - ▶ Jensen's backlog model
 - ▶ Ben-Zvi's tuple timestamped model
 - ▶ Gadia's attribute value timestamped model
- ▶ Now is a continuously moving constant
 - ▶ Understand the meaning of timestamps that include now
 - ▶ Often the largest timestamp is used to represent now; leads to subtle issue (duration, etc)

NDBS14, SL03

71/71

M. Böhlen, ifi@uzh