CSCI585

Introduction to Temporal Database Research

by Cyrus Shahabi

from

Christian S. Jensen's

Chapter 1

C. Shahabi

1

CSCI585

Outline

- Introduction & definition
- Modeling
- Querying
- Database design
    - Logical design
    - Conceptual design
- DBMS implementation
    - Query processing
    - Implementation of algebraic operators
    - Indexing structures
- Summary
- Open problems

C. Shahabi

2

CSCI585

Introduction

- Most applications of database technology are *temporal* in nature:
  - Financial apps.: portfolio management, accounting & banking
  - Record-keeping apps.: personnel, medical-record and inventory management
  - Scheduling apps.: airline, car, hotel reservations and project management
  - Scientific apps.: weather monitoring

C. Shahabi 3

CSCI585

Definitions

- Temporal DBMS manages time-referenced data, hence, times are associated with database entities
- Two types of time: *valid* time and *transaction* time
- Valid time, vt, of a *fact* (any logical statement that is either true or false) is the collected times (possibly spanning the past, present & future) when the fact is true
- Although all facts have a valid time, the valid time of a fact may not necessarily be recorded in the database (unknown or irrelevant to the app.)
  - If a database models different worlds, database facts might have several valid times, one for each world

C. Shahabi 4

CSCI585

Definitions …

- Transaction time, tt: the time that a fact is *current* in the database
- Tt may be associated with any database entity, not only with facts
- Although all entities can be assigned a tt, the database designer may decide to not capture this aspect for some entities
- Tt aspect of an entity has a duration: from insertion to deletion, with multiple insertions and deletions being possible for the same entity ➔
- Hence, deletion is pure logical (not physically removed but ceased to be part of the database's current state

CSCI585

Definitions …

- Tt captures time varying states of the db & apps. that demand accountability and tractability rely on dbs that record Tt
- Tt, unlike vt, is well-behaved and may be supplied automatically by the DBMS
- Both tt and vt values are drawn from a time domain, which may or may not stretch infinitely into the past and future
- Time domain may be discrete or continuous
- In databases, a finite and discrete time domain is typically assumed

CSCI585

Definitions ...

- Time is assumed to be totally ordered, but various partial orders and cyclic time has also been suggested
- Uniqueness of "*Now*":
  - the current time is ever-increasing,
  - all activity is trapped at the current time, and
  - current time separates the past from the future
- The spatial equivalent "here" doesn't have the above properties; the biggest difference between time and space is that time cannot be reused!
- The uniqueness of now is one of the reasons why techniques from other research areas are not readily (or not at all) applicable to temporal data
- Now offers new data management challenges particular to temporal databases

CSCI585

Modeling

- To extend a DBMS to become temporal, mechanisms must be provided for capturing valid and transaction times of the facts recorded by relations (temporal relations)
- More than 24 extended relational models proposed to add time to relational model, most of which supported only valid time
- We consider three *bitemporal* ones for a video rental applications: customers check out tapes for certain durations of time and dates.

CSCI585

Modeling …

- **Bitemporal Conceptual Data Model (BCDM): timestamps tuples with sets of (tt, vt) values**

cID	TapeNum	
C101	T1234	{(2,2), (2,3), (2,4), (3,2), (3,3), (3,4), …, (UC,2), (UC,3), (UC,4)}
C102	T1245	{(5,5), (6,5), (6,6), (7,5), (7,6), (7,7), (8,5), (8,6), (8,7),…, (UC,5), (UC,6), (UC,7)}
C102	T1234	{(9,9), (9,10), (9,11), (10,9), (10,10), (10,11), (10,12), (10,13),…, (13,9), (13,10), (13,11), (13,12), (13,13), (14,9), …, (14,14), (15,9), …, (15,15), (16,9), …, (16,15), …, (UC,9), …, (UC,15)}

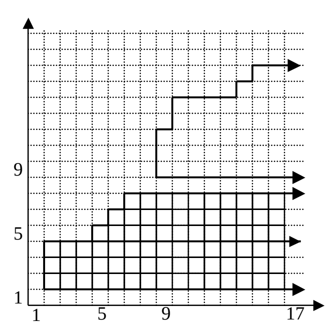
- **C101 rents T1234 on May 2nd for 3 days, & returns it on 5th**
- **C102 rents T1245 on 5th open-ended, & returns it on 8th**
- **C102 rents T1234 on 9th to be returned on 12th. On 10th the rent is extended to include 13th but tape is not returned until 16th.**

C. Shahabi 9

CSCI585

Modeling …

- **Bitemporal Conceptual Data Model (BCDM): timestamps tuples with sets of (tt, vt) values**



- **C101 rents T1234 on May 2nd for 3 days, & returns it on 5th**
- **C102 rents T1245 on 5th open-ended, & returns it on 8th**
- **C102 rents T1234 on 9th to be returned on 12th. On 10th the rent is extended to include 13th but tape is not returned until 16th.**

C. Shahabi 10

CSCI585

Modeling …

- **BCDM pros:**
  - **Since no two tuples with mutually identical explicit values are allowed in BCDM relation instance, the full history of a fact is contained in exactly one tuple**
  - **Relation instances that are syntactically different have different information content and vice versa**
- **BCDM cons:**
  - **Bad internal representation and display to users of temporal info**
  - **Varying length and voluminous timestamps of tuples are impractical to manage directly**
  - **Timestamp values are hard to comprehend in BCDM format**

CSCI585

Modeling …

- **Fixed-length format for tuples, where each tuple's timestamp encodes a rectangular or stair-based bitemporal region**
- **Several tuples may be needed to represent a single fact**

cID	TapeNum	Ts	Te	Vs	Ve
C101	T1234	2	UC	2	4
C102	T1245	5	7	5	now
C102	T1245	8	UC	5	7
C102	T1234	9	9	9	11
C102	T1234	10	13	9	13
C102	T1234	14	15	9	now
C102	T1234	16	UC	9	15

- **C101 rents T1234 on May 2nd for 3 days, & returns it on 5th**
- **C102 rents T1245 on 5th open-ended, & returns it on 8th**
- **C102 rents T1234 on 9th to be returned on 12th. On 10th the rent is extended to include 13th but tape is not returned until 16th.**

CSCI585

Modeling …

- Non-first-normal-form representation
- Relation is thought of as recording information about some types of objects (e.g., information about customers)

CustomerID		TapeNum	
[2, Now] x [2,4]	C101	[2, Now] x [2,4]	T1234
[5, 7] x [5, inf]	C102	[5, 7] x [5, inf]	T1245
[8, Now] x [5, 7]		[8, Now] x [5, 7]	
[9,9] x [9, 11]		[9,9] x [9, 11]	T1234
[10,13] x [9, 13]		[10,13] x [9, 13]	
[14,15] x [9, inf]		[14,15] x [9, inf]	
[16, Now] x [9, 15]		[16, Now] x [9, 15]	

- C101 rents T1234 on May 2nd for 3 days, & returns it on 5th
- C102 rents T1245 on 5th open-ended, & returns it on 8th
- C102 rents T1234 on 9th to be returned on 12th. On 10th the rent is extended to include 13th but tape is not returned until 16th.

CSCI585

Modeling …

- Note that 2nd tuple records two facts: rental information for customer C102 for the two tapes
- Pros of the two latter models:
  - No need to update the relation at every tick, it is achieved by introducing “now” variable that assume the current value
- Two choices to enter time values into relations
  1. At the level of tuples (tuple timestamping)
  2. At the level of attribute values (attribute timestamping)

CSCI585

Modeling …

- Relation instances that all three models may record are *snapshot equivalent* (corresponding to a *point-based* view of data), e.g.,

A	Vs	Ve
a	2	8
b	2	8

A	Vs	Ve
a	2	4
a	5	8
b	2	8

A	Vs	Ve
a	2	8
b	2	4
b	5	8

- The first relation is coalesced version of the other two, but they are snapshot equiv.
- Coalescing operation merges value equivalent tuples with same non-timestamp attributes and adjacent or overlapping time intervals

CSCI585

Modeling …

- BCDM only allows coalesced relation instances, i.e., relations are only different if they are not snapshot equivalent
  - The last two relations are not legal in BCDM
- However, the three relations are not equivalent from an *interval-based view:*
  - First relation: a tape was checked out for 7 days
  - Second relation: the tape was checked out for 3 days initially and then for 4 more days

CSCI585

Querying

- Temporal queries "can" be expressed via conventional query languages such as SQL (e.g., current temporal applications); however, with great difficulty

cID	TapeNum
C101	T1234
C102	T1425
C102	T1324
C103	T1243

S-CheckedOut

cID	TapeNum	Vs	Ve
C101	T1234	2	now
C101	T1245	5	10
C102	T1245	22	25
C102	T1425	9	19
C102	T1434	4	14
C102	T1324	9	now
C103	T1243	7	21

V-CheckedOut

- At time 17, the first relation is a snapshot of the second

C. Shahabi 17

CSCI585

Querying …

- Number of current checkouts:
  - SELECT COUNT (TapeNum) FROM S-CHeckedOut
- Temporal generalization of the above query: time-varying count of tapes checked out
  - If now is replaced with a fixed time value, this can be done in SQL in 6 steps and 35 lines!
- Specifying a key constraint:
  - ALTER TABLE S-CheckedOut ADD PRIMARY KEY (TapeNum)
- TapeNum is also a key for V-CheckedOut at each point in time
  - It takes 12 lines and a complex SQL statement to express this constraint

C. Shahabi 18

CSCI585

Querying …

- Hence, some 40 temporal query languages have been proposed (most with their own data model), e.g., TSQL2
- Simple queries should remain simple:
  - VALIDTIME
    SELECT COUNT (TapeNum) FROM V-CheckedOut
  - CONSTRAINT temporalkey VALIDTIME UNIQUE TapeNum
- Early languages based on: relational algebra
- Later: calculus-based, Datalog-based and OO
- Recent: extensions to SQL

CSCI585

Querying …

- Many modeling issues impact the language design, e.g., time stamping tuples or attributes
- Language design must consider:
  - time-varying nature of data,
  - predicated on temporal values,
  - temporal constructs,
  - supporting states and/or events,
  - supporting multiple calendars,
  - modification of temporal relations,
  - cursors, views, integrity constraints, handling now, aggregates, schema versioning, periodic data

CSCI585

Querying …

- Desired properties of temporal query languages:
  1. Temporal upward compatibility: conventional queries and modifications of temporal relations should act on the current state
  2. Pervasive support for sequence queries: that request the history of something, e.g., temporal aggregation above
  3. Support for point-based and interval-based view of data
  4. Adequate expressive power
  5. Ability to be efficiently implemented

C. Shahabi

21

CSCI585

DBMS Design

- Database schemas capturing time-referenced data are complex
- Two traditional contexts of database design:
  - Data model of DBMS at 3 levels: view, logical, physical (e.g., relational model for the first two)
  - A high-level conceptual design model: ER model
- Then, mappings bring a conceptual design into a schema that conforms to the specific implementation data model (e.g., ER to relational mapping)
- Here: we consider temporal database "logical" and "conceptual" design

C. Shahabi

22

Logical Design

- Need for guidelines such as formalization guidelines, but conventional normalization concepts are not applicable to temporal relational data models
- A range of temporal normalization concepts have been proposed: temporal dependencies, keys and normal forms
- Conventional dependencies do not apply: TapeNum does not determine cID, (go through 3 examples, but it should!)
- But it should: at any point in time, a tape can only be checked out by a single customer
  - ➔ TapeNum temporally determines cID, but the reverse does not hold

Logical Design …

1. A temporal relation satisfies a temporal dependency if all its snapshots satisfy the corresponding conventional dependency

- How to determine snapshots? Timeslice operators:
  - Temporal predicate as argument: e.g., contain
  - A time point as parameter: e.g., (tt, vt)
  - Returns snapshot of the relation corresponding to the specified time point, omitting the timestamp attribute
- Problem: an atemporal approach! which applies to each snapshot of a temporal relation in isolation and hence fails to account for "temporal" aspects of data

CSCI585

Logical Design …

2. **Consider dependencies and associated normal forms that hold *between* time points**

- **Build in the notion of time granularity into the normalization concepts**
- **Not only consider snapshots computed at non-decomposable time points, but also at coarser granularities:**
  - **Video rental examples: day as finest granularity, weeks and months may also be considered**

CSCI585

Logical Design …

3. **Introducing new concepts that capture the temporal aspects of data and may form the basis for new database design guidelines**

- **Most prominent candidate: *time patterns***
  - **Video rental example: since the set of tapes checked out by a customer changes more frequently than the customer's address, they should be stored in separate relations**
- **Another candidate: *lifespan***
- **Attributes with different lifespan (to avoid null values) or with different precision (hour vs. day) should be stored separately**

CSCI585

Conceptual Design

- ER diagrams become obscure and cluttered when an attempt is made to capture temporal aspects (see example)
- CheckedOut relationship should become ternary by introducing an artificial entity set to capture time of rental
- However, still issues remain: varying rental price over time, transaction time inclusion, …
- Some industrial solution: ignore temporal aspects in the ER diagram and supplement it with textual phrases, e.g., “full temporal support”
  - ➔ no automatic mapping from ER to model
- Dozens of temporally enhanced ER models proposed

C. Shahabi

27

CSCI585

Conceptual Design …

1. Give all existing ER constructs temporal semantics, similar to “applies to all snapshots” for normalization
   - ⇨ Does not result in any new syntactical constructs
   - ⇨ Rules out databases with non-temporal parts: while the syntax of legacy diagrams remain valid their semantics have changed!
2. Devise new notational shorthand for frequent temporal aspects in ER diagram (e.g., time varying attributes)
   - ⇨ Both non-temporal and *mixed* databases can be modeled
   - ⇨ More difficult to understand

C. Shahabi

28

CSCI585

Conceptual Design ...

- All existing models assume mapping to relational model
- None tries to map to one of the several time-extended relational models
- Also mapping to emerging models (e.g., SQL3/ORDBMS) are missing.

C. Shahabi 29

CSCI585

DBMS Implementation

- Integrated approach: internal modules of a DBMS are modified or extended to support time-varying data
  - Efficiency
- Layered approach: a software layer interposed between the user applications and DBMS that converts temporal query language statements to conventional statements
  - Realistic for short and medium term
- Popular approach: integrated, utilizing timestamping tuples with time intervals

C. Shahabi 30

CSCI585

Query Processing

- Temporal queries are large and complex
- Also, the predicates might be temporal, e.g., overlap among two time intervals
- Unlike equality predicate in conventional joins, temporal joins require multiple inequality predicates to be examined: two intervals I and j overlap iff *st(i) <= end(j) and st(j) <= end(i)*
- Coalescing of data should be implemented efficiently: interactions among coalescing, duplicate removal and ordering

CSCI585

Query Processing ...

- Opportunities for temporal query optimization:
  - Time advances continuously, hence for transaction time, time value used most recently in updates is the largest value used so far

    ➔ natural sorting and clustering: if current and logically deleted tuples are stored separately, then
    - Current clustered on st(tt)
    - Deleted clustered on end(tt)
  - Integrity constraint st(j)<end(j)
  - Intervals associated with a key value are contiguous in time (end of one interval is the beginning of the other)

CSCI585

Implementation of Algebraic Operators

- Efficient implementation of temporal selection, joins, aggregates, and duplicate elimination ➔ temporal index structures
- Variety of binary temporal joins have been proposed: time-join, time-equijoin, … as extensions of nested loop or merge join that exploits orders or local workspace as well as partitioning based joins
- Also, incremental techniques for implementing operators on relations capturing transaction time have been discussed
  - Caching the results of previous computations to be reused later (easy to do since the records of updates, I.e., changes to previously cached results, are already contained in a temporal DBMS)

CSCI585

Imp. Of Algebraic Ops…

- Efficient implementation of time-varying aggregates
- Efficient implementation of <u>coalescing</u>:
  1. Sorting the argument relation on the explicit attribute values as well as the valid time
  2. Perform the merging in the subsequent scan

CSCI585

Indexing Structures

- Similar to spatial index structures can be based on traditional indexes such as B+-tree or multidimensional ones such as R-tree
- Index structures usually used for selection operators
- Active research investigation: use index structures for temporal joins, coalescing and aggregates

CSCI585

Summary

- Popular approaches:
  - Snapshot-based semantics for database design
  - BCDM for modeling
  - TSQL2 as a query language
- Well understood issues (some with efficient implementation):
  - Semantics of the time domain: its structure, dimensionality, and indeterminacy
  - Representational issues and operations on timestamps
  - Temporal joins, aggregates and coalescing
  - Temporal index structures supporting vt, tt, or both
  - Prototype implementations of temporal DBMS

CSCI585

Open Problems

- Legacy awareness
- Architecture awareness
- Visualization of temporal data
- Conceptual design
- Performance (cost models for temporal operators and maintaining statistics for query optimizer)

CSCI585

Open Problems ...

- Related research that can benefit from and/or challenge temporal DBMS research:
  - Active databases
  - Spatiotemporal databeses
  - Moving objects
  - Multimedia, virtual reality, immersive apps.
  - Temporal data mining
  - Warehousing