Temporal Databases

(Chapter 5)

Peter Revesz

CSCE 413/813
Computer Science and Engineering
University of Nebraska – Lincoln

Allen's Relations

Example Binary Relations between events I and J:

I Before J – when I is over before J starts.

I Contains J – when J occurs only when I occurs.

I Equals J – when I and J occur always simultaneously.

I Meets J – when I just finishes before J starts.

I Overlap J – when I and J overlap in time.

Axioms on Allen's Relations

Transitivity Axioms:

If I_1 Before I_2 and I_2 Before I_3, then I_1 Before I_3.

If I_1 Contains I_2 and I_2 Contains I_3, then I_1 Contains I_3.

If I_1 Equals I_2 and I_2 Equals I_3, then I_1 Equals I_3.

Meets-Before Axiom:

If I_1 Meets I_2 and I_2 Meets I_3, then I_1 Before I_3.

Representing Allen's Relations by Constraints

Let the beginning and the end of event *I* be I^- and I^+ Then we have:

Relation of I and J	I^- J^-	I^- J^+	I^+ J^-	I^+ J^+
After :		$>$		
Before :			$<$	
Meets :			$=$	
Met by :		$=$		
During :	$>$			$<$
Contains :	$<$			$>$
Equal :	$=$			$=$
Finishes :	$>$			$=$
Finished by :	$<$			$=$
Starts :	$=$			$<$
Started by :	$=$			$>$
Overlaps :	$<$		$>$	$<$
Overlapped by :	$>$	$<$		$>$

Temporal Data Abstraction

View Level:

Anderson at AT&T: 1980 ——————————————— 1997
Brown at IBM: 1985 ——————————— 1996
Clark at Lotus: 1990 — 1991

Logical Level:

Employee

Name	Company	Year
Anderson	AT&T	1980
⋮	⋮	⋮
Anderson	AT&T	1997
Brown	IBM	1985
⋮	⋮	⋮
Brown	IBM	1996
Clark	Lotus	1990
Clark	Lotus	1991

Temporal Database Queries (Point-Based Queries)

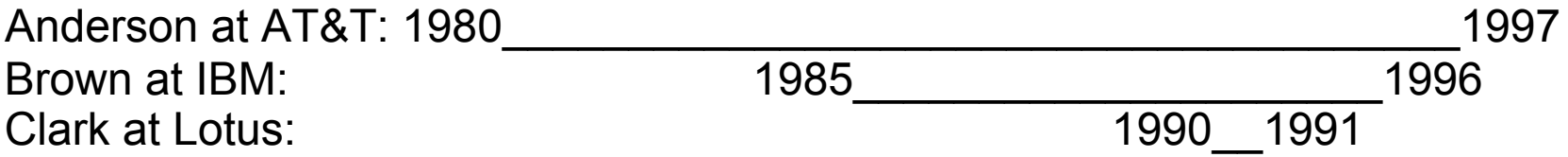
Example: Find the employees who did not work during all the time that Brown worked.

```
SELECT   E1.Name
FROM     Employee AS E1, Employee AS E2
WHERE    E1.Name ≠ "Brown"  AND
         E2.Name = "Brown"  AND
         E2.Year  NOT IN
                  (SELECT   E3.Year
                   FROM     Employee AS E3
                   WHERE    E3.Name = E1.Name)
```

- Point-based queries use the logical level.
- Point-based queries are usually only *standard SQL queries.*

Temporal Data Abstraction

View Level:



Logical Level:

Employee

Name	Company	Year
Anderson	AT&T	1980
⋮	⋮	⋮
Anderson	AT&T	1997
Brown	IBM	1985
⋮	⋮	⋮
Brown	IBM	1996
Clark	Lotus	1990
Clark	Lotus	1991

Constraint Level:

Option 1: Use *order constraints.*

Employee

Name	Company	Year	
Anderson	AT&T	t	$1980 \leq t,\ t \leq 1993$
Anderson	AT&T	t	$1994 \leq t,\ t \leq 1997$
Brown	IBM	t	$1985 \leq t,\ t \leq 1996$
Clark	Lotus	t	$1990 \leq t,\ t \leq 1991$

Option 2 (TQuel): Use *From* and *To.*

Employee

Name	Company	From	To
Anderson	AT&T	1980	1993
Anderson	AT&T	1994	1997
Brown	IBM	1985	1996
Clark	Lotus	1990	1991

Querying TQuel Temporal Databases

Employee

Name	Company	From	To
Anderson	AT&T	1980	1993
Anderson	AT&T	1994	1997
Brown	IBM	1985	1996
Clark	Lotus	1990	1991

Example: Find the employees who worked during all the time that Brown worked.

One difficulty in expressing the above query in standard SQL is that the intervals are broken up into smaller ones. In the above Employee relation, Anderson worked at AT&T from 1980 to 1997, which is a single interval, but it is broken up into two separate records with intervals 1980 to 1993 and 1994 to 1997.

Querying TQuel Temporal Databases

coalesce the temporal intervals in the Employee relation.

coalesce$(2^{[I^-,I^+]}) \rightarrow 2^{[J^-,J^+]}$: This operator takes as input a set of intervals and returns another logically equivalent set of intervals, such that in the returned set no interval meets or overlaps another interval.

```
CREATE VIEW   Coalesce_Employee(Name, Company, From, To)
SELECT        Name, Company, Coalesce(From, To)
FROM          Employee
GROUP BY      Name, Company
```

Example: Find the employees who worked during all the time that Brown worked.

```
CREATE VIEW   Contemporary_of_Brown(Name)
SELECT        DISTINCT E1.Name
FROM          Coalesce_Employee AS E1, Coalesce_Employee AS E2
WHERE         E1.Name ≠ "Brown" AND
              E2.Name = "Brown" AND
              contains(E1.From, E1.To, E2.From, E2.To)
```

Note the use of *contains* in the last line.

- TQuel uses SQL extended with the *coalesce* operator and *Allen's relations.*

Practice

1. Andrew and Barbara are two persons who have some free days next month as shown in the constraint relation $Free$, where d is an integer representing days of the month.

Free

PERSON	DAY	
Andrew	d	$3 \leq d,\ d \leq 6$
Andrew	d	$16 \leq d,\ d \leq 26$
Barbara	d	$6 \leq d,\ d \leq 18$
Barbara	d	$25 \leq d,\ d \leq 30$

   (a) Convert the $Free$ relation to the TQuel temporal data model.

   (b) Convert the $Free$ relation to the relational data model.

2. Answer the following queries using the $Free$ relation converted to the TQuel temporal data model and interval-based temporal database queries.

   (a) Find the days when Andrew and Barbara are both free.

   (b) A common friend is visiting town from days 5 to 15. Find whether either Andrew or Barbara can be with the common friend on each of the days of the visit.

3. Answer the queries in the previous exercise using the $Free$ relation converted to the relational data model.