Entangled Transactions

Nitin Gupta, Milos Nikolic, Sudip Roy, Gabriel Bender, Lucja Kot, Johannes Gehrke, Christoph Koch
In the age of Web 2.0, users increasingly coordinate on data-driven tasks

Example: travel planning

- Mickey and Minnie want to travel to Seattle on the same flight
Students want to enroll in classes with their friends

Interesting scenarios:

- **Negative constraints**
  - avoid the section my ex is in

- **Strong mutual dependencies**
  - I will take this tough class only if my friend takes it too
Players want alliances based on shared or complementary goals

- I will attack from the North if someone else attacks from the South

Alliances often formed with strangers for the purpose of achieving one goal
Room Sharing among attendees of the 2011 ACM SIGMOD Conference

The conference officers have set up a web page where interested attendees of the conference can register their interest in sharing rooms at the conference hotel. Through this service attendees can enter their details so that interested people can contact each other.

To register your interest, please submit your information at: http://bit.ly/sigm_share_room (URL shortener service forwarding to a Google Spreadsheets form). This service is provided solely as a convenience to participants that seek to share accommodation costs. Please contact directly participants that have expressed interest. The organizers will not be involved in the process nor are they responsible for possible abuse of the information you provide.
Sharing a room at the Conference Hotel?

This form allows people who want to stay at the conference hotel to express their interest in sharing rooms. Please fill out the following form, all people expressing interest in sharing a room can then contact each other by looking at the following page: [http://bit.ly/sigmod_share_room_list](http://bit.ly/sigmod_share_room_list)

* Required

**Name** *

**email** *

**Period you wish to stay at the hotel** *

**Please add any constraints on sharing a room (gender, etc)**

**Submit**

Powered by Google Docs
<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Name</th>
<th>email</th>
<th>Period you wish to stay at the hotel</th>
<th>Please add any constraints on sharing a room (gender, etc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/10/2011 6:15:25</td>
<td>Greg Ju</td>
<td><a href="mailto:gregju@gmail.com">gregju@gmail.com</a></td>
<td>12/06/2011-16/06/2011</td>
<td>Female</td>
</tr>
<tr>
<td>5/10/2011 6:38:39</td>
<td>Stephen Duke</td>
<td><a href="mailto:stephen.duke@gmail.com">stephen.duke@gmail.com</a></td>
<td>June 13 - June 17 (4 nights)</td>
<td>I will be interested to share a room (only females) during the conference. thanks!</td>
</tr>
<tr>
<td>5/10/2011 16:38:58</td>
<td>Ryan Johnson</td>
<td><a href="mailto:ryjohnson@gmail.com">ryjohnson@gmail.com</a></td>
<td>12-17 June</td>
<td>Males only. I already have a room reservation -- looking to fill the other bed and split the cost.</td>
</tr>
<tr>
<td>5/10/2011 18:34:49</td>
<td>John Doe</td>
<td><a href="mailto:john.doe@gmail.com">john.doe@gmail.com</a></td>
<td>5 nights June 12-16 (inclusive)</td>
<td>prefer females (i'm a girl)</td>
</tr>
<tr>
<td>5/12/2011 13:03:30</td>
<td>Bob Smith</td>
<td><a href="mailto:bsmith@gmail.com">bsmith@gmail.com</a></td>
<td>13th-16th June</td>
<td>n/a</td>
</tr>
<tr>
<td>5/13/2011 12:45:54</td>
<td>Jane Doe</td>
<td><a href="mailto:janedoe@gmail.com">janedoe@gmail.com</a></td>
<td>12-17 June</td>
<td>I'm easy going :)</td>
</tr>
<tr>
<td>5/14/2011 12:20:15</td>
<td>John Doe</td>
<td><a href="mailto:john.doe@gmail.com">john.doe@gmail.com</a></td>
<td>Check-in 11, Check-out 15</td>
<td>n/a</td>
</tr>
<tr>
<td>5/23/2011 22:47:20</td>
<td>Alice Black</td>
<td><a href="mailto:alice.black@gmail.com">alice.black@gmail.com</a></td>
<td>12th-17th</td>
<td>n/a</td>
</tr>
<tr>
<td>5/25/2011 23:16:36</td>
<td>Tom White</td>
<td><a href="mailto:tom.white@gmail.com">tom.white@gmail.com</a></td>
<td>June 12 - June 17</td>
<td>male</td>
</tr>
<tr>
<td>6/4/2011 13:10:08</td>
<td>Jack Brown</td>
<td><a href="mailto:jack.brown@gmail.com">jack.brown@gmail.com</a></td>
<td>June 12th</td>
<td>Gender: male</td>
</tr>
</tbody>
</table>
Mickey expresses his intention to coordinate
  • “I want to travel to Seattle on the same flight as Minnie”
Minnie expresses a symmetric intention
System takes care of the rest

To make this a reality, need:
  • a basic primitive for coordination - entangled queries (SIGMOD 2011)
  • an understanding of how entangled queries fit into transactions (this paper)
SELECT ‘Mickey’, Flightno INTO ANSWER Booking
WHERE
(‘Minnie’, Flightno)
AND Flightno
IN SELECT Flightno FROM Flights F
WHERE F.Destination=‘Seattle’

CHOOSE 1

• ANSWER Booking is an ephemeral relation
• exists only when the queries are answered
• used to collect the answers to all “participating” queries
• allows the expression of cross-constraints between answers
SELECT ‘Mickey’, Flightno INTO ANSWER Booking
WHERE
(‘Minnie’, Flightno)
AND Flightno

CHOOSE 1

SELECT ‘Minnie’, Flightno INTO ANSWER Booking
WHERE
(‘Mickey’, Flightno)
AND Flightno

IN SELECT Flightno FROM Flights F
WHERE F.Destination=‘Seattle’

AND Flightno

IN SELECT F.Flightno FROM Flights F, Airlines A
WHERE F.Destination=‘Seattle’
AND F.Flightno = A.Flightno
AND A.Airline = ‘United’

CHOOSE 1
**Evaluation Example**

<table>
<thead>
<tr>
<th>Flightno</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO83</td>
<td>Seattle</td>
</tr>
<tr>
<td>CO82</td>
<td>Paris</td>
</tr>
<tr>
<td>UA211</td>
<td>Seattle</td>
</tr>
<tr>
<td>TH244</td>
<td>Chicago</td>
</tr>
<tr>
<td>UA112</td>
<td>Seattle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flightno</th>
<th>Airline</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO83</td>
<td>Continental</td>
</tr>
<tr>
<td>CO82</td>
<td>Continental</td>
</tr>
<tr>
<td>UA211</td>
<td>United</td>
</tr>
<tr>
<td>TH244</td>
<td>Thai</td>
</tr>
<tr>
<td>UA112</td>
<td>United</td>
</tr>
</tbody>
</table>

**UA211 and UA112 satisfy all constraints**

<table>
<thead>
<tr>
<th>Mickey’s query</th>
<th>Minnie’s query</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Answer</strong></td>
<td>(Mickey, UA112) (Minnie, UA112)</td>
</tr>
<tr>
<td><strong>Constraint</strong></td>
<td>(Minnie, UA112) (Mickey, UA112)</td>
</tr>
</tbody>
</table>
Entangled queries typically embedded in transactions

1. coordinate on flight number
2. book ticket based on result from step 1
3. commit

More interesting scenario:

1. coordinate on flight number
2. book ticket
3. coordinate on hotel based on date of flight chosen
4. book hotel
5. commit
BEGIN TRANSACTION;

SELECT `Mickey`, fno, fdate AS @ArrivalDay INTO ANSWER FlightRes
WHERE fno, date IN (SELECT fno, fdate FROM Flights WHERE dest=`Seattle')
AND (`Minnie', fno, fdate) IN ANSWER FlightRes
CHOOSE 1;

-- (Code to perform flight booking omitted)

SELECT `Mickey`, hid, @ArrivalDay, `2011-09-02’ INTO ANSWER HotelRes
WHERE hid IN (SELECT hid FROM Hotels WHERE location=`Seattle')
AND (`Minnie', hid, @ArrivalDay, `2011-09-02’) IN ANSWER HotelRes
CHOOSE 1;

-- (Code to perform hotel booking omitted)

COMMIT;
Research Challenges

What kind of “transaction” is this?
• a classical transaction is a standalone, coherent unit of work
• an entangled transaction is not standalone - requires an entanglement partner!

What happens to isolation?
• there is communication, so classical isolation is broken
• but some sort of “residual isolation” is desirable

Need a formal semantic model for entangled transactions
How do we actually run entangled transactions?

- how do we enforce “correct” execution as defined in semantic model?
  - locking, optimistic cc?

- what if something goes wrong?
  - Minnie never submits her matching transaction
  - an entangled query fails
  - entanglement succeeds, but then one of the transactions aborts

Need an execution model for entangled transactions

- one size will likely not fit all
How do we run entangled transactions in a real system?

- is entangled transaction support implemented in the middle tier or within the DBMS?
- what is the overall system architecture?
- how do we make this fast and scalable?
Our Contributions

A semantic model for entangled transactions
  • formalizes the entangled equivalents of the ACID properties

A practical execution model
  • suitable for realistic scenarios like travel planning
  • (ongoing research)

A concrete system design and prototype implementation
  • middle-tier support for entangled transactions
  • integrates with existing DBMS functionality

Experimental evaluation
Our Contributions

A semantic model for entangled transactions

- formalizes the entangled equivalents of the ACID properties

A practical execution model

- suitable for realistic scenarios like travel planning
- (ongoing research)

A concrete system design and prototype implementation

- middle-tier support for entangled transactions
- integrates with existing DBMS functionality

Experimental evaluation
What is a transaction?
- a standalone, coherent unit of work

Formalized in the *consistency* assumption

Every transaction, if executed on an initially consistent database by itself, will produce another consistent database.
What is an **entangled** transaction?

- a standalone, coherent unit of work *modulo* its need for entangled query answers

Formalized in the **oracle-consistency** assumption

- an entangled query oracle is a process that (only) answers entangled queries

Every **entangled** transaction, if executed on an initially consistent database by itself *except for an entangled query oracle* that returns valid query answers, will produce another consistent database.
Classically, two ways to formalize isolation:

- exclusion of anomalies (dirty reads etc.)
- serializability - equivalence to a serial schedule
- results that link the two notions

Challenges in the entangled case:

- serializability no longer makes sense
- new isolation anomalies unique to entangled setting
New Isolation Anomaly #1

Widowed transaction

- what if one transaction aborts?
- entanglement is a kind of dirty read (on the system state)
Unrepeatable quasi-read

- information flows through entanglement to a transaction, even if it does not read a table directly
Two definitions of isolation for an entangled schedule:

- **anomaly-based entangled isolation**
  - exclude all the classical anomalies plus widowed transactions and unrepeatable quasi-reads

- **oracle-serializability**
  - (final state) equivalence to schedule where the same transactions execute serially along a suitable oracle

**Theorem:** Anomaly-based entangled isolation implies oracle-serializability

- so list of anomalies is “complete”
- see paper for details!
Consistency
- a transaction executing on its own with an oracle takes DB from one consistent state to another

Isolation
- anomaly-based and oracle-serializability definitions
- Theorem: the former implies the latter

Atomicity
- transaction must complete or be rolled back

Durability
- if a transaction commits, changes must persist
We ran several experiments using our prototype
  • implemented in Java
  • uses JDBC to connect to a MySQL database system (InnoDB)

Experiments investigate:
  • the overhead of providing transactional guarantees
    - “How much slower is the running time if we enclose the code in BEGIN TRANSACTION; and COMMIT;?”
  • the performance impact of different workloads (transactions match well or badly, in a simple or complex way)
  • what happens when we vary parameters in our execution model
Three workload types

- **NoSocial** - a user books a flight
- **Social** - a user books a flight based on a friend’s booking
- **Entangled** - a user coordinates with a friend to book a flight using entangled query

For each of these, generate a non-transactional (-Q) and a transactional (-T) workload

- 10000 transactions generated using Slashdot social network data

Determine running time for each workload

- this is a function of the number of concurrent connections
Results (10K-transaction Workloads)

- NoSocial-T
- Social-T
- Entangled-T
- NoSocial-Q
- Social-Q
- Entangled-Q
Entangled transactions are a powerful, clean and declarative way to support data-driven coordination

- formal semantic model with analogues of the classic ACID properties
- end-to-end solution with a practical execution model and an implemented prototype

Many exciting challenges for future work

- more execution models
- language and model extensions
- privacy issues
- ...
A simple execution for noninteractive transactions

Isolation achieved with appropriate locking and group commit requirement

Run-based scheduling:
• transactions scheduled in batches or runs
• entangled queries are blocking points in evaluation
• run ends when every transaction is either ready to commit or blocked waiting for a partner
Transactions in a Run

Mickey's transaction ready to commit, waits for Minnie

Donald's transaction blocked, waits for retry of flight query

Mickey
- System evaluates all three flight queries

Books flight

Minnie
- Books flight

System evaluates Mickey and Minnie's hotel queries and Donald's flight query

Books hotel

Donald

Books hotel

COMMIT

COMMIT

ABORT
Results: Pending Transactions

![Graph showing the relationship between pending transactions and time for different $f$ values.]
Results: Coordinating Set

The graph shows the time (in seconds) as a function of the size of the coordinating set for different configurations:

- Spoke-hub, $f=10$
- Spoke-hub, $f=50$
- Cycle, $f=10$
- Cycle, $f=50$

The x-axis represents the size of the coordinating set, ranging from 2 to 10. The y-axis represents time, ranging from 0 to 400 seconds.