

Declarative Data-Driven Coordination

Johannes Gehrke

Department of Computer Science, Cornell University

With Gabriel Bender, Nitin Gupta, Lucja Kot, Sudip Roy (Cornell)
and Milos Nikolic, Christoph Koch (EPFL)



Cornell University

- Assume Tom and Meg want to coordinate itineraries
 - Fly on the same flight, in adjacent seats
 - Also stay in the same hotel if possible



Students want to enroll in classes with their friends

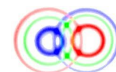
- Help with homework/moral support
- Already happens with out-of-band communication
- CourseRank
 - “Connect to Facebook to find out who of your friends is enrolled”

Interesting coordination 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 to form alliances with others based on shared or complementary goals
 - I will attack from the North if someone else attacks from the South
- These alliances may be completely ad-hoc and formed with total strangers just for the purpose of achieving one goal



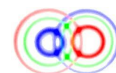


Room Sharing among attendees of the 2011 ACM SIGMOD Conference

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.



← → ↻ <https://spreadsheets.google.com/spreadsheet/viewform?formkey=dEVscThiZkVJMkdJeUFCX1pSZVFPUC6MQ> ☆ 🔍

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/sigm_share_room_list

* Required

Name *

email *

Period you wish to stay at the hotel *

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

Powered by [Google Docs](#)

[Report Abuse](#) - [Terms of Service](#) - [Additional Terms](#)



Coordination: SIGMOD 2011 (Contd.)

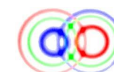


← → ↻ <https://spreadsheets.google.com/spreadsheet/pub?hl=en&hl=en&key=0/>

Sharing a room at the Conference Hotel? : List

Timestamp	Name	email	Period you wish to stay at the hotel	Please add any constraints on sharing a room (gender, etc)
5/10/2011 6:15:25	Pragna	pragna@cornell.edu	12/06/2011-16/06/2011	Female
5/10/2011 6:38:39	Pragna	pragna@cornell.edu	June 13 - June 17 (4 nights)	I will be interested to share a room (only females) during the conference. thanks!
5/10/2011 16:38:58	Pragna	pragna@cornell.edu	12-17 June	Males only. I already have a room reservation -- looking to fill the other bed and split the cost.
5/10/2011 18:34:49	Pragna	pragna@cornell.edu	5 nights June 12-16 (inclusive)	
5/12/2011 13:03:30	Pragna	pragna@cornell.edu	13th-16th June	prefer females (i'm a girl)
5/13/2011 12:45:54	Pragna	pragna@cornell.edu	12-17 June	
5/14/2011 12:20:15	Pragna	pragna@cornell.edu	Check-in 11 , Check-out 15	n/a I'm easy going :)
5/23/2011 22:47:20	Pragna	pragna@cornell.edu	12th-17th	
5/25/2011 23:16:36	Pragna	pragna@cornell.edu	June 12 - June 17	male
6/4/2011 13:10:08	Pragna	pragna@cornell.edu	June 12th	Gender: male

Published by [Google Docs](#) – [Report Abuse](#) – Updated automatically every 5 minutes



- Scheduling meeting times with students/advises
- Wedding gift purchases
 - People can group together to purchase a more expensive item
- Post-disaster emergency management
- Charity fund-raising with matching funds
- Joint gift-giving



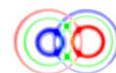
It is not just the applications that are data-driven....

The coordination itself is data-driven too!

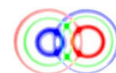
- Users want to agree on a choice of data values
- Not on the time of day of when jointly enrolling in a course

Today typically achieved with out-of-band communication

- Or through an ad-hoc solution for a given scenario...



- Goal: Provide a declarative abstraction and mechanism to support D3C
 - Being declarative is fundamental principle in query and update languages
 - Coordination pertains to data, so should be expressed at the same level
 - Meg says: “Book me a ticket on the same flight as Tom”
 - System takes care of the actual coordination



ACID Properties of a transaction

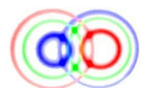
- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

D3C requires relaxing isolation

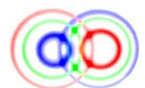
- For semantic reasons, not for performance (such as lower isolation levels, eventual consistency)
- We still want atomicity and durability

And the communication due to coordination should be “controlled“

- “Residual” isolation



- Operating systems:
 - Message passing
 - Shared memory
 - Transactional memory
- Programming languages:
 - Powerful formalisms such as the Π -calculus (channels)
 - Concurrent ML
 - Concurrent Haskell
 - Erlang
- Multi-agent systems



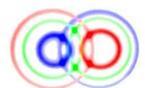
Sagas/nested transactions

- Transactions inside transactions; only commit when outermost transaction commits

Triggers

- Program automatically executed as a response to certain events in the database
- Used instead of or after an INSERT, UPDATE, or DELETE operations on database tables

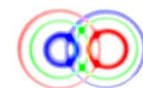
```
CREATE OR REPLACE TRIGGER trigger_name  
BEFORE DELETE OR INSERT OR UPDATE ON table_name  
FOR EACH ROW ...
```



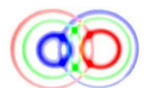
Why A New Abstraction?



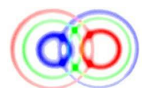
- Need an abstraction that is at the “right” level
 - Data-centric, not process-centric
 - Should not require users to manipulate low-level constructs like channels
- Hide the implementation of the coordination logic
 - The matching that must take place between different users' coordination constraints
- We want to enable complex kinds of coordination
 - Mutual handshake: I will take this tough course but only if my friend does, and vice versa



- Introduction
- **Entangled queries**
 - Language
 - Query Evaluation
- Entangled transactions
- Open Problems



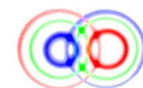
- Introduction
- **Entangled queries**
 - Language
 - Query Evaluation
- Entangled transactions
- Open Problems



Entangles queries: an abstraction and a mechanism for D3C

Example scenario: Steve and Larry want to travel to NYC on the same flight

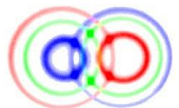
- In addition, Steve wants to travel only on United



Steve's Entangled Query

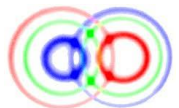


```
SELECT 'Steve', fno INTO ANSWER Reservation
WHERE
    fno IN (SELECT fno FROM Flights WHERE dest='JFK')
    AND ('Larry', fno) IN ANSWER Reservation
CHOOSE 1
```



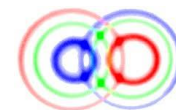
```
SELECT 'Steve', fno INTO ANSWER Reservation
WHERE
    fno IN (SELECT fno FROM Flights WHERE dest='JFK')
    AND ('Larry', fno) IN ANSWER Reservation
CHOOSE 1
```

- **Larry's answer** must also be in the Reservation table



```
SELECT 'Larry', fno INTO ANSWER Reservation
WHERE
    fno IN (SELECT fno FROM Flights F, Airlines A
            WHERE F.dest='JFK' and F.fno = A.fno AND
                 A.airline = 'United' )
    AND ('Steve', fno) IN ANSWER Reservation
CHOOSE 1
```

```
SELECT 'Steve', fno INTO ANSWER Reservation
WHERE
    fno IN (SELECT fno FROM Flights WHERE dest='JFK')
    AND ('Larry', fno) IN ANSWER Reservation
CHOOSE 1
```



Flights Database



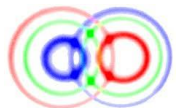
Cornell University

Flight

Flightno	Destination
122	JFK
123	JFK
134	JFK
136	Brussels

Airlines

Flightno	Airline
122	United
123	United
134	Lufthansa
136	Alitalia



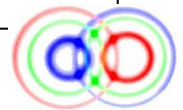
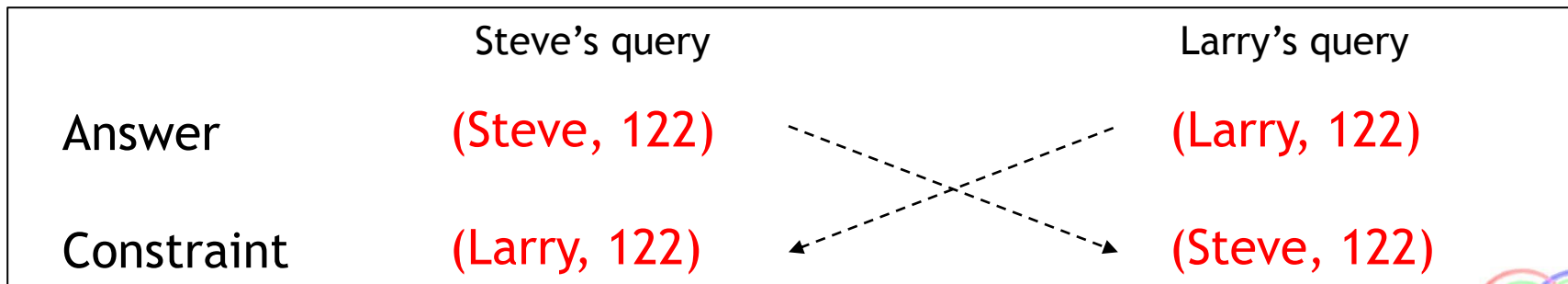
Mutual Constraint Satisfaction



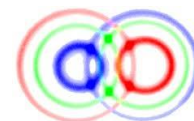
Flightno	Destination
122	JFK
123	JFK
134	JFK
136	Brussels

Flightno	Airline
122	United
123	United
134	Lufthansa
136	Alitalia

United Flights 122 and 123 satisfy the constraints.

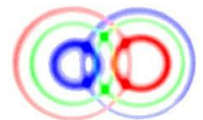


- Individual queries do not see the full ANSWER table, but are guaranteed that their constraints are satisfied
- Both transactions that contain these entangled queries can now proceed and make bookings
- Note that the coordination partner was specified **implicitly** using data values, not explicitly

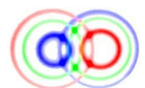


```
SELECT select_expr
INTO ANSWER tbl_name [, ANSWER tbl_name] ...
FROM TABLE
[WHERE answer_condition]
CHOOSE 1
```

- Currently, we allow only SPJ (conjunctive) queries in the WHERE clause
 - Could be extended with disjunction, union, aggregate constraints, ...



- Introduction
- **Entangled queries**
 - Language
 - **Query Evaluation**
- Entangled transactions
- Open Problems



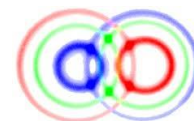
How do we evaluate entangled queries?

Problem: Evaluation is NP-complete in the general case

- Not that surprising: Entangled queries can encode CSP

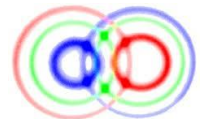
More than one source of complexity:

- Matching up the entangled queries
- Finding data values that satisfy coordination constraints



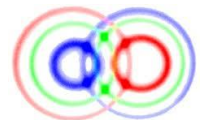
Stages of query evaluation

1. Check queries for safety
2. Partition queries into subsets and match queries
3. Create and evaluate the combined query and construct the individual answers



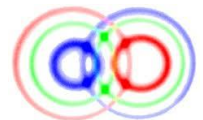
Stages of query evaluation

1. Check queries for safety
2. Partition queries into subsets and match queries
3. Create and evaluate the combined query and construct the individual answers



In many settings, there will be only one way to match up the queries for coordination

- Specify this formally as a notion of **safety** for a set of queries
- Test for safety



A Datalog-like representation (without recursion)

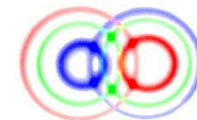
$$\{ C \} \quad H :- B$$

C, H and B are conjunctions of relational atoms

- C and H over answer relations
- B over database (non-answer) relations

Representation of Larry and Steve's queries

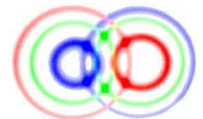
- $\{ \text{Booking}(\text{Larry}, x) \} \quad \text{Booking}(\text{Steve}, x) :- \text{Flight}(x, \text{JFK})$
- $\{ \text{Booking}(\text{Steve}, y) \} \quad \text{Booking}(\text{Larry}, y) :- \text{Flight}(y, \text{JFK})$
 $\quad \wedge \text{Airline}(y, \text{United})$



- A set of queries is unsafe if there a query with more than one potential coordination partner

{Booking (Larry, x)} Booking (Steve, x) :- Flight (x, JFK)
{Booking (Larry, x)} Booking (Bill, x) :- Flight (x, JFK)
{Booking (u, x)} Booking (Larry, x) :- Friend (Larry, u),
Flight (x, JFK)

- Safety is independent of data
 - Asking the system to choose between users is different from asking it to choose between flight numbers
 - Safety is formalized using logical unifiability between heads and postconditions

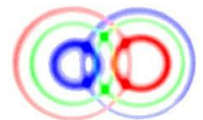


Safety and Unifiability



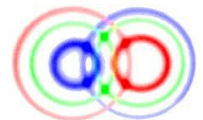
- Two relation atoms (referring to the same relation) are unifiable unless they contain different constants in the same attribute
 - $R(x; y)$ and $R(z; z)$ are unifiable
 - $R(2; y)$ and $R(3; z)$ are not
- Query q is a potential coordination partner for q' if some head atom of q unifies with some postcondition atom of q' .
- A set of queries is **unsafe** if there a query with more than one potential coordination partner

Simple algorithm: Iterate over query set and search for queries with postconditions that unify with heads from more than one query.

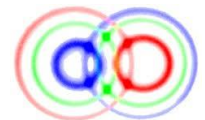
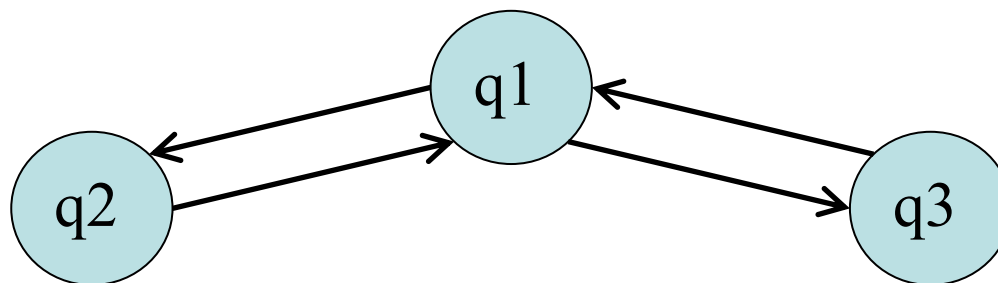


Stages of query evaluation

1. Check queries for safety
2. Partition queries into subsets and match queries
3. Create and evaluate the combined query and construct the individual answers



- Partitioning and query matching use a structure called the **unifiability graph**
 - One node per query
 - Edge from q to q' if a head atom of q unifies with a postcondition atom of q'
- Example:
 - $q_1 : \{R(x_1) \wedge S(x_2)\}$ $T(x_3) \quad :- \quad D_1(x_1; x_2; x_3)$
 - $q_2 : \{T(1)\}$ $R(y_1) \quad :- \quad D_2(y_1)$
 - $q_3 : \{T(z_1)\}$ $S(z_2) \quad :- \quad D_3(z_1, z_2)$

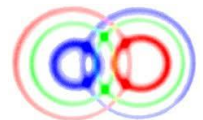


Unifiability graph gives overall structure of how queries match up

But we know more information:

- $q1 : \{R(x1) \wedge S(x2)\}$ $T(x3) \quad :- \quad D1(x1; x2; x3)$
- $q2 : \{T(1)\}$ $R(y1) \quad :- \quad D2(y1)$
- $q3 : \{T(z1)\}$ $S(z2) \quad :- \quad D3(z1, z2)$

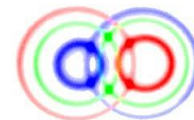
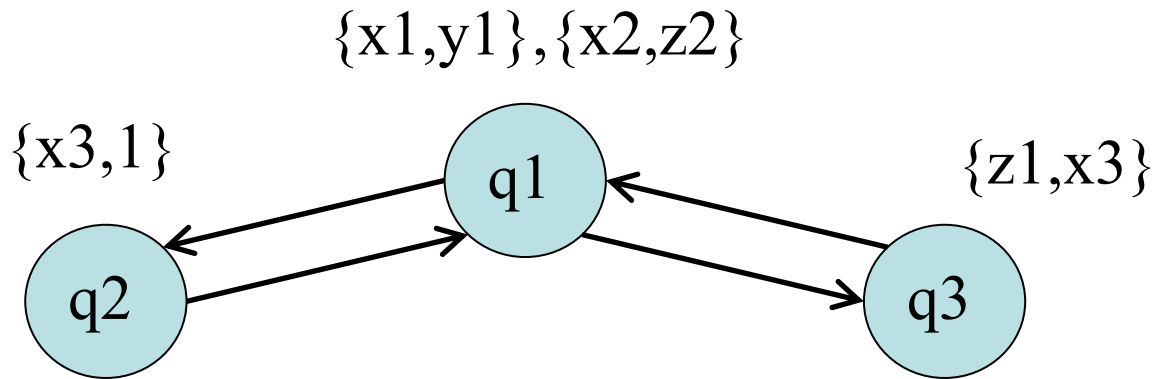
- The head of $q1$ only satisfies the postcondition of $q2$ if $x3=1$
 - Eventually, all the variables will be associated with values from the DB, so we will have a **valuation**
 - We know coordination is only possible for valuations that assign $x3$ the value 1



Matching (Contd.)



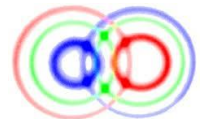
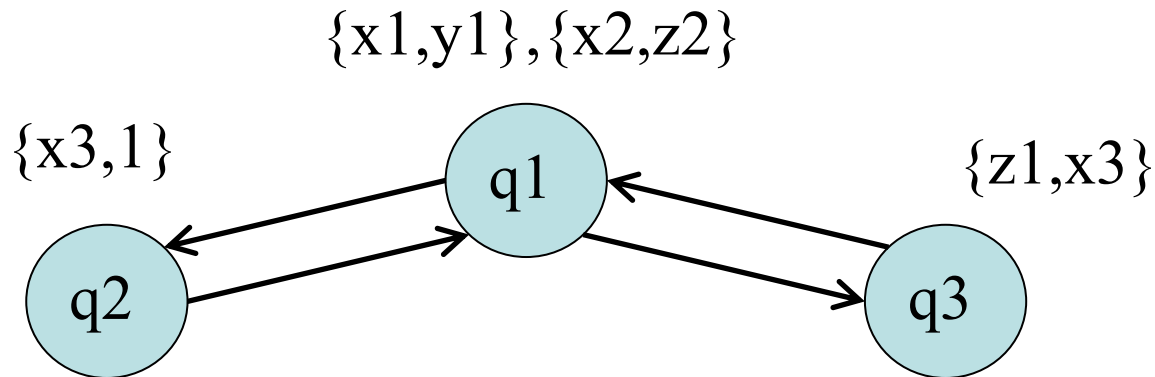
- Represent this information as unifiers associated with nodes in the graph
- A unifier is a constraint imposed by a particular query
 - $q1 : \{R(x1) \wedge S(x2)\}$ $T(x3) \quad :- \quad D1(x1, x2, x3)$
 - $q2 : \{T(1)\}$ $R(y1) \quad :- \quad D2(y1)$
 - $q3 : \{T(z1)\}$ $S(z2) \quad :- \quad D3(z1, z2)$



Matching (Contd.)



- Suppose a head of q unifies with a postcondition of q'
 - q' “relies” on q for satisfaction
 - q is unique for this q' , by safety
 - so, if q' is to receive an answer, q must receive an answer too
 - so, any valuation constraints from q apply to q' as well!

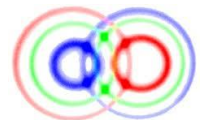
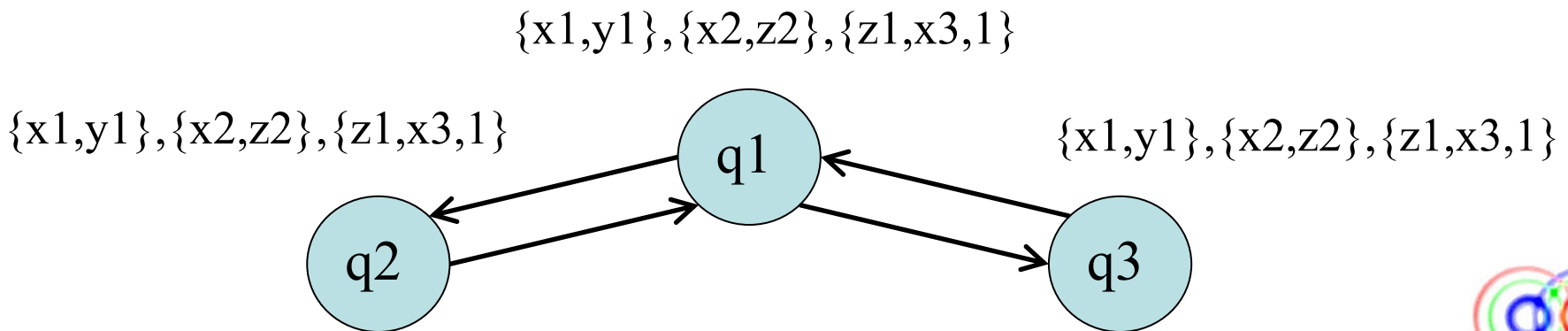


Matching (Contd.)



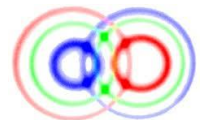
Query matching is an iterative process that propagates these unifiers through the graph

- Related to the chase and to arc-consistency
- May remove nodes from the graph (queries whose postconditions cannot be satisfied)
- Eventually either fails or reaches a fixpoint \rightarrow matching



Stages of query evaluation

1. Check queries for safety
2. Partition queries into subsets and match queries
3. Create and evaluate the combined query and construct the individual answers



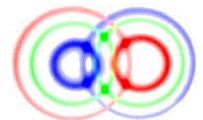
Building the Combined Query



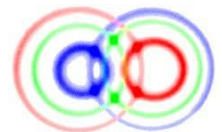
$\{\text{Booking}(\text{Larry}, x)\} \text{Booking}(\text{Steve}, x) \text{ :- Flight}(x, \text{JFK})$
 $\{\text{Booking}(\text{Steve}, y)\} \text{Booking}(\text{Larry}, y) \text{ :- Flight}(y, \text{JFK})$
 $\wedge \text{Airline}(y, \text{United})$

Gets rewritten to:

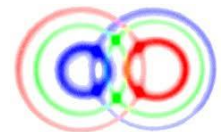
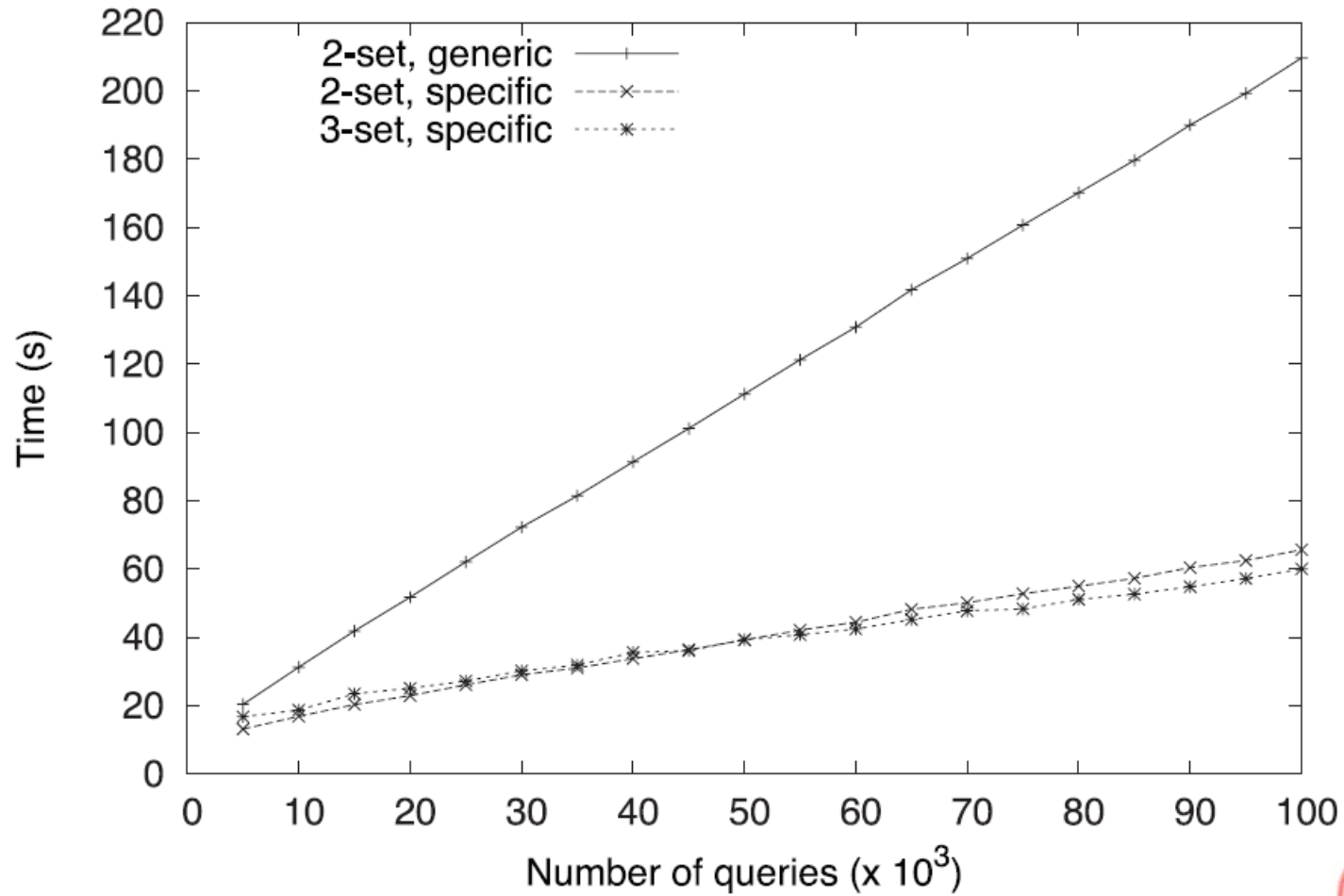
$\text{Booking}(\text{Larry}, x) \wedge \text{Booking}(\text{Steve}, x) \text{ :-}$
 $\text{Flight}(x, \text{JFK}) \wedge \text{Airline}(x, \text{United})$



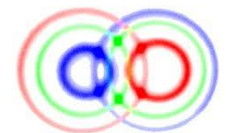
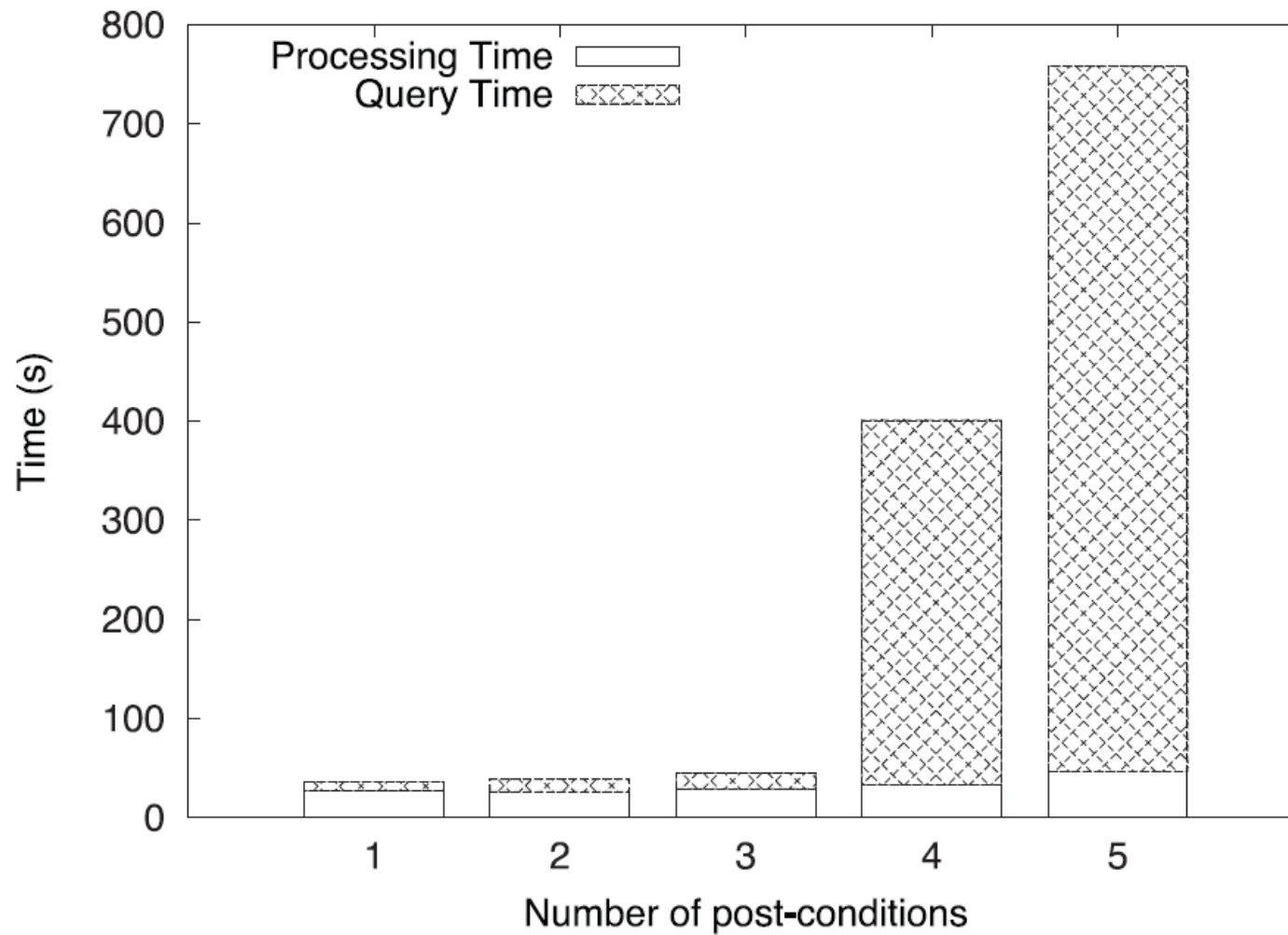
- Prototype implemented in Java and uses JDBC to connect to a MySQL database system
- Dataset:
 - Generate queries that match in pairs or triples
 - Make queries more or less specific (coordinate with a named friend vs. any friend)
- Additional experiments:
 - Increase number of post-conditions per query
 - Stress-test performance of matching algorithm



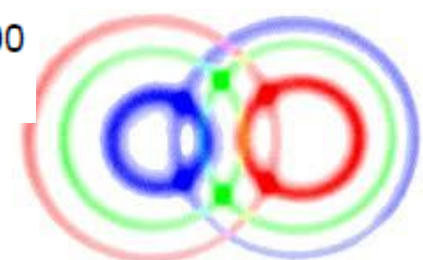
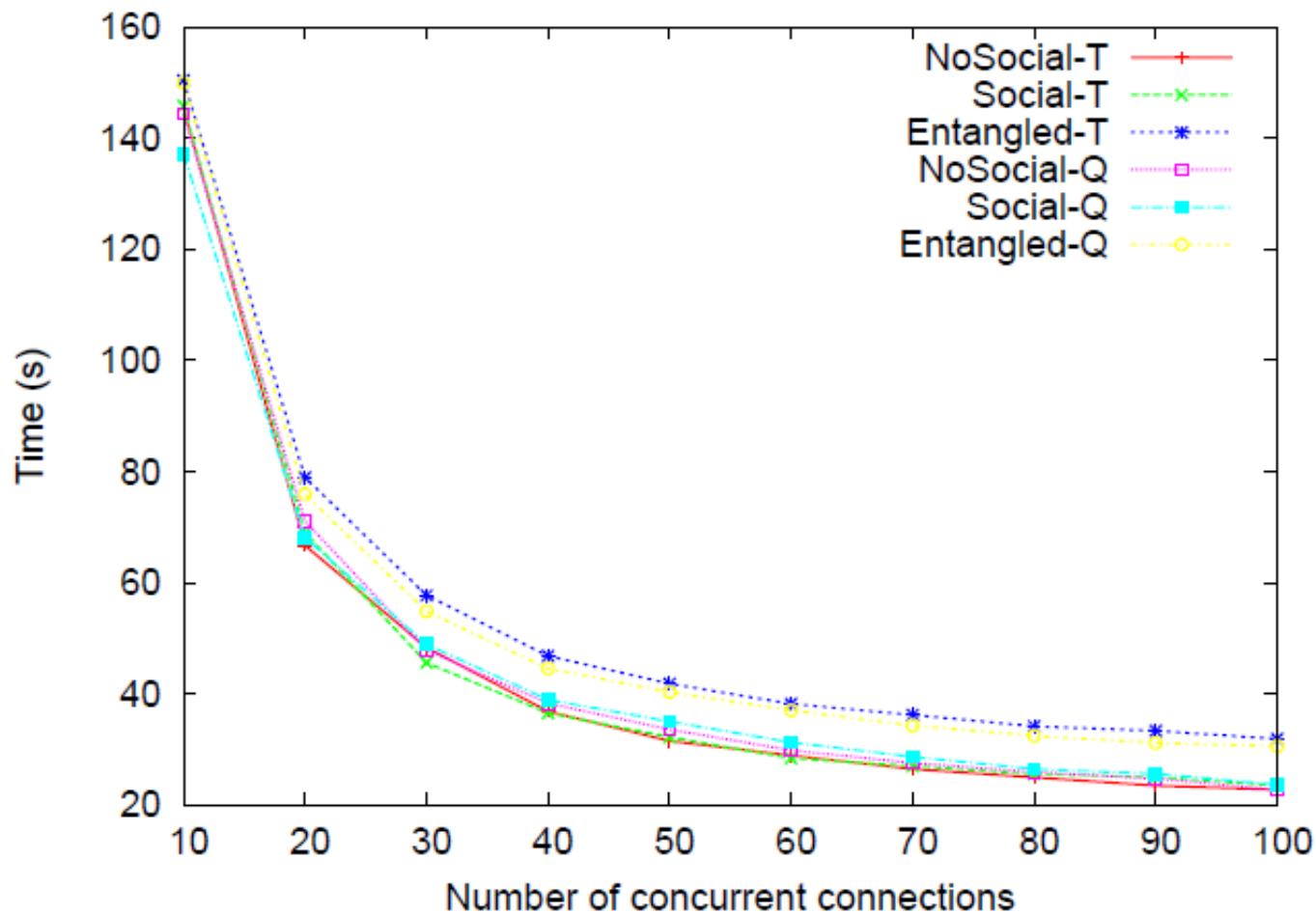
Results: Scalability



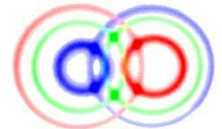
Increasing # of constraints



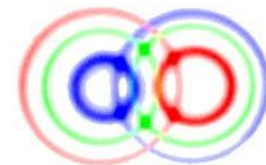
Results: Concurrency



- Introduction
- Entangled queries
- **Entangled transactions**
- Open Problems



- Goal: Extend transactions to incorporate entangled queries
- Challenge:
 - Relationship of entanglement to classical transactions
- Example scenario
 - Steve and Larry want to fly together to NYC
 - If they can make a flight booking together, then they want to stay in the same hotel



Steve's Transaction



```
BEGIN TRANSACTION WITH TIMEOUT 2 DAYS;
```

```
SELECT `Steve`, fno, fdate AS @ArrivalDay INTO ANSWER FlightReservation  
WHERE fno, date IN (SELECT fno, fdate FROM Flights WHERE dest=`JFK')  
AND (`Larry`, fno, fdate) IN ANSWER FlightReservation  
CHOOSE 1;
```

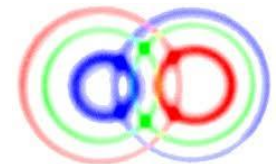
```
-- (Code to perform flight booking omitted)
```

```
SET @StayLength = `2011-10-30` - @ArrivalDay;
```

```
SELECT `Steve`, hid, @ArrivalDay, @StayLength INTO ANSWER HotelReservation  
WHERE hid IN (SELECT hid FROM Hotels WHERE location=`NYC')  
AND (`Larry`, hid, @ArrivalDay, @StayLength) IN ANSWER HotelReservation  
CHOOSE 1;
```

```
-- (Code to perform hotel booking omitted)
```

```
COMMIT;
```



Recall consistency:

- Every transaction, if executed by itself on an initially consistent database, will produce another consistent database.

What is the analogous property for entangled transactions --- what is a unit of work?

- Portions of a transaction?
- A transaction?
- A group of transactions?

```
BEGIN TRANSACTION WITH TIMEOUT 2 DAYS;
```

```
SELECT `Steve`, fno, fdate AS @ArrivalDay INTO  
ANSWER FlightReservation
```

```
WHERE fno, date IN (SELECT fno, fdate FROM Flights  
WHERE dest=`JFK')
```

```
AND (`Larry`, fno, fdate) IN ANSWER FlightReservation  
CHOOSE 1;
```

```
-- (Code to perform flight booking omitted)
```

```
SET @StayLength = `2011-10-30` - @ArrivalDay;
```

```
SELECT `Steve`, hid, @ArrivalDay, @StayLength INTO  
ANSWER HotelReservation
```

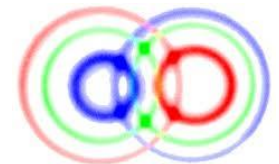
```
WHERE hid IN (SELECT hid FROM Hotels WHERE  
location=`NYC')
```

```
AND (`Larry`, hid, @ArrivalDay, @StayLength) IN  
ANSWER HotelReservation
```

```
CHOOSE 1;
```

```
-- (Code to perform hotel booking omitted)
```

```
COMMIT;
```

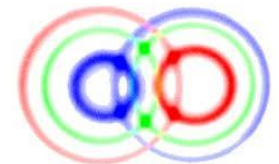


Entangled Query Oracle

- Process that executes alongside an entangled transaction
- For an entangled query, the oracle chooses a valid answer (=ground the query on the database) and returns it to any entangled query
- Has no direct effect on the database's state

Oracle Consistency:

- Suppose an entangled transaction executes by itself on an initially consistent database, **using an entangled query oracle to obtain answers to the entangled queries**. Then the execution will produce another consistent database.

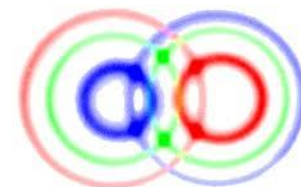


Anomaly-based definition

Two anomalies

- Widowed Transactions
- Unrepeatable quasi-reads

Let us review first what we mean by anomalies for classical transactions.

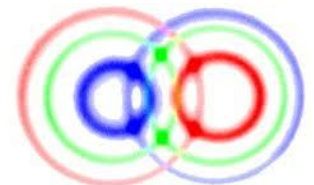


- Consider a possible interleaving (schedule) of two transactions:

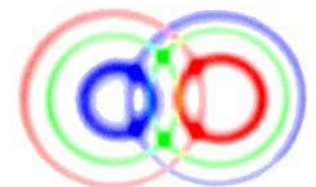
T1:	$A=A+100,$	$B=B-100$
T2:	$A=1.06*A, B=1.06*B$	

- The systems's view of the schedule:

T1:	$R(A), W(A),$	$R(B), W(B)$
T2:	$R(A), W(A), R(B), W(B)$	



- Serial schedule: Schedule that does not interleave the actions of different transactions.
- Equivalent schedules: For any database state
 - The effect (on the set of objects in the database) of executing the schedules is the same
 - The values read by transactions is the same in the schedules
- Serializable schedule: A schedule that is equivalent to some serial execution of the transactions.
- Note: If each transaction preserves consistency, every serializable schedule preserves consistency.

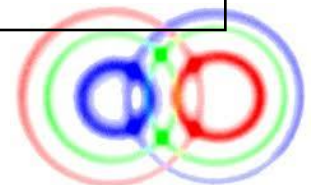


- Reading Uncommitted Data (WR Conflicts, “dirty reads”):

T1:	R(A), W(A),	R(B), W(B), Abort
T2:	R(A), W(A), C	

- Unrepeatable Reads (RW Conflicts):

T1:	R(A),	R(A), W(A), C
T2:	R(A), W(A), C	

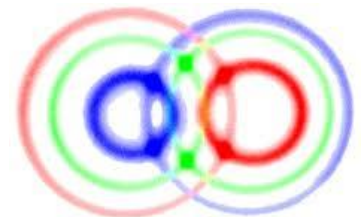


Traditional Anomalies (Contd.)



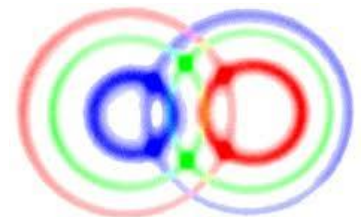
- Overwriting Uncommitted Data (WW Conflicts):

T1:	W(A),	W(B), C
T2:	W(A), W(B), C	

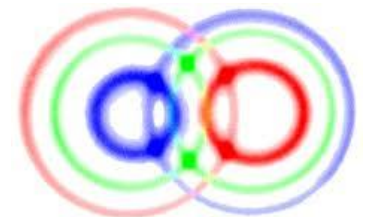
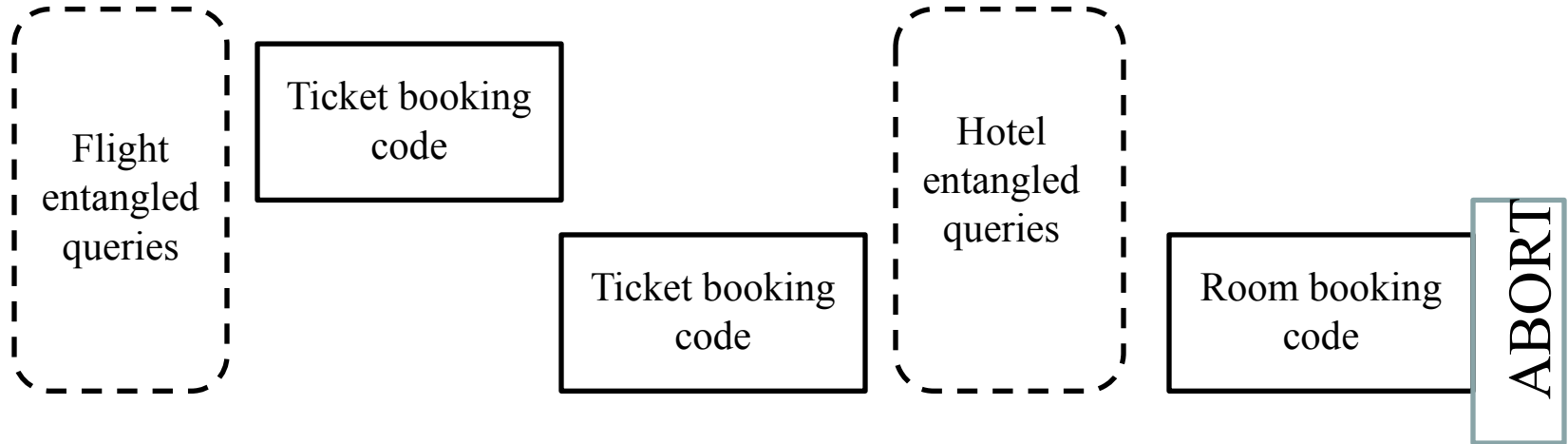


Two new anomalies

- Widowed Transactions
- Unrepeatable quasi-reads



New Anomaly 1: Widowed Transactions



New Anomaly 2: Unrepeatable Quasi-Reads

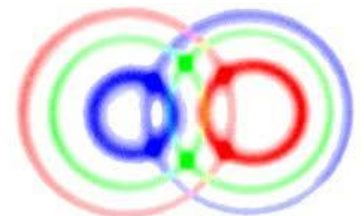


Entangled queries



```
SELECT * FROM  
Airlines  
WHERE airline  
= `United`
```

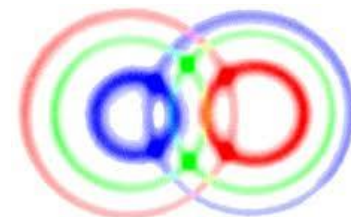
```
INSERT INTO  
Airlines  
VALUES (`125`,  
`United`)
```



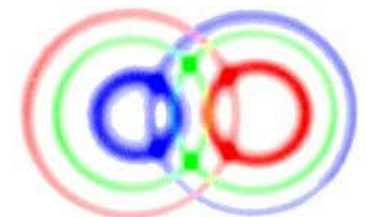
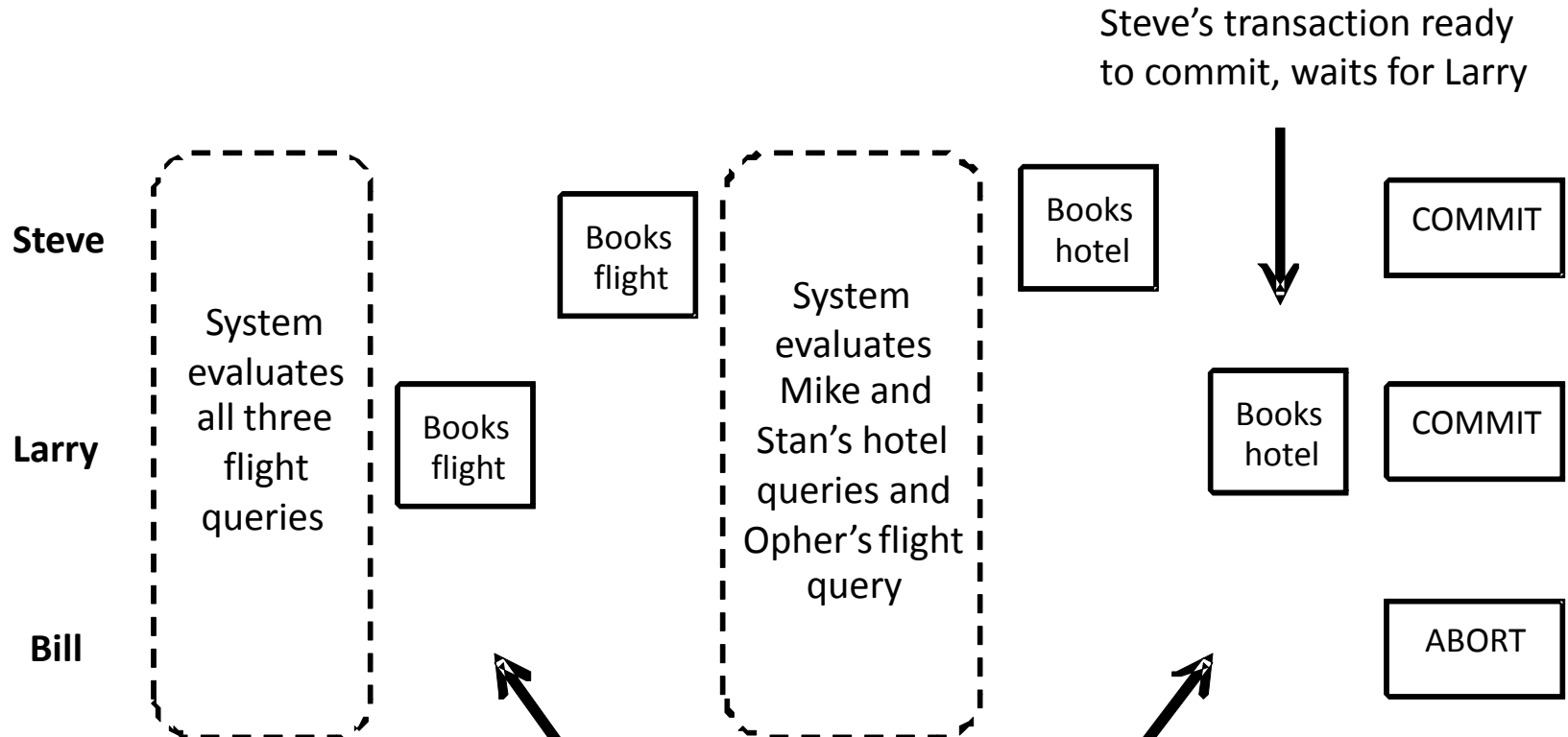
Eliminating These Anomalies



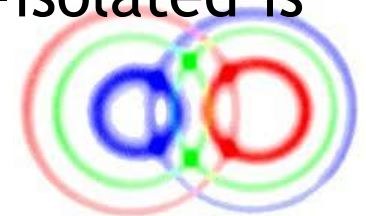
- How to avoid widowed transactions?
 - **Group commit** of all the transactions that are connected through entangled queries
- Unrepeatable quasi-reads
 - **Appropriate locking** of data structures



Scheduling



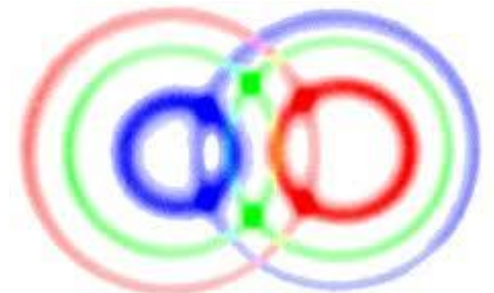
- Traditional ACID Properties:
 - Atomicity
 - Consistency → Oracle Consistency
 - Isolation → Two new phenomena
 - Durability
- We can now define
 - Oracle-serializability: Serial schedule with a suitable oracle that provides answers to entangled queries
 - Entangled isolation: Schedule does not have any anomalies
- Main theorem: Any schedule that is entangled-isolated is also oracle-serializable.



Outline



- Introduction
- Entangled queries
- Entangled transactions
- **Open Problems**



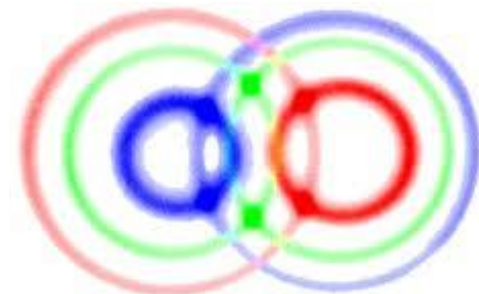
This is Just The Beginning



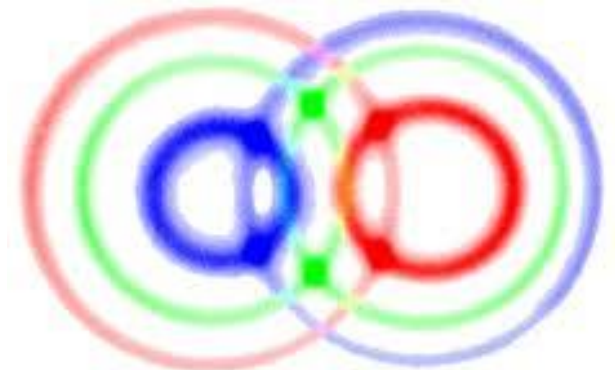
Cornell University

Many exciting research directions:

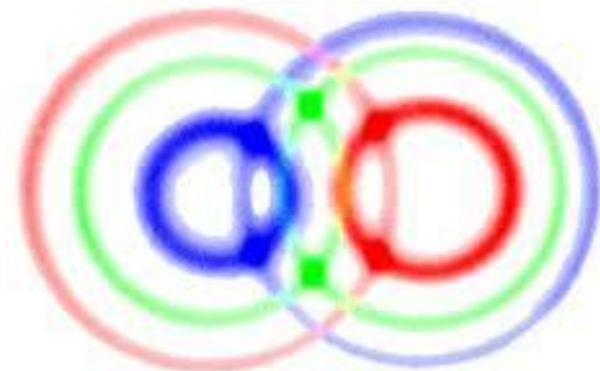
- Extending the language for entangled queries
- Studying the complexity of evaluation
- Modeling entangled transactions
- Designing a system for end-to-end support of entangled queries
- Privacy and security issues
- Wider implications for system design of relaxing isolation
- Entanglement for CEP



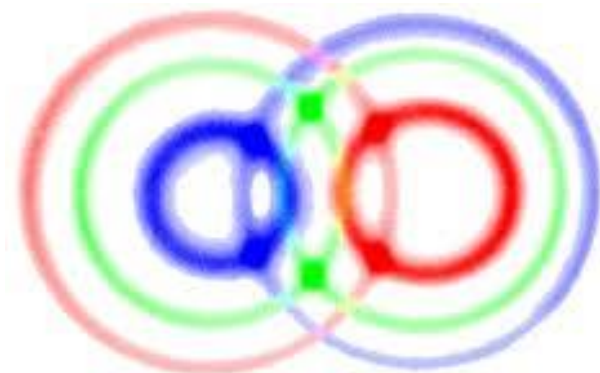
```
SELECT P.partyid, 'Simon' INTO ANSWER Attendance
FROM Parties P
WHERE P.pdate='Friday'
      AND
      (SELECT COUNT(*)
       FROM ANSWER Attendance A, Friend F
       WHERE P.partyid = A.partyid AND A.name = F.name1
            AND F.name2 = 'Paul) > 2
CHOOSE 1
```



- “Soft” constraints
 - Travel dates should be as close as possible (but need not be identical)
- Preferences
 - Will travel on any US carrier, but prefer United if possible
 - Will travel any day next week, but the earlier the better
- Semantics where more than one record is returned
 - Example: Course enrollment



- We need to understand the complexity of evaluation better
 - How do the different sources of complexity interact?
- What answering guarantees can we provide and when?
 - Do we always find an answer if one exists?
 - Do we find an answer that involves a maximal number of queries?
- How do language extensions affect tractability?



Supporting Entanglement in a System

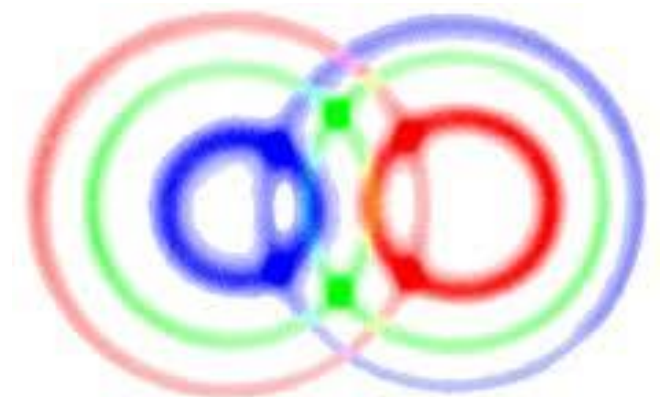


Where does evaluation take place?

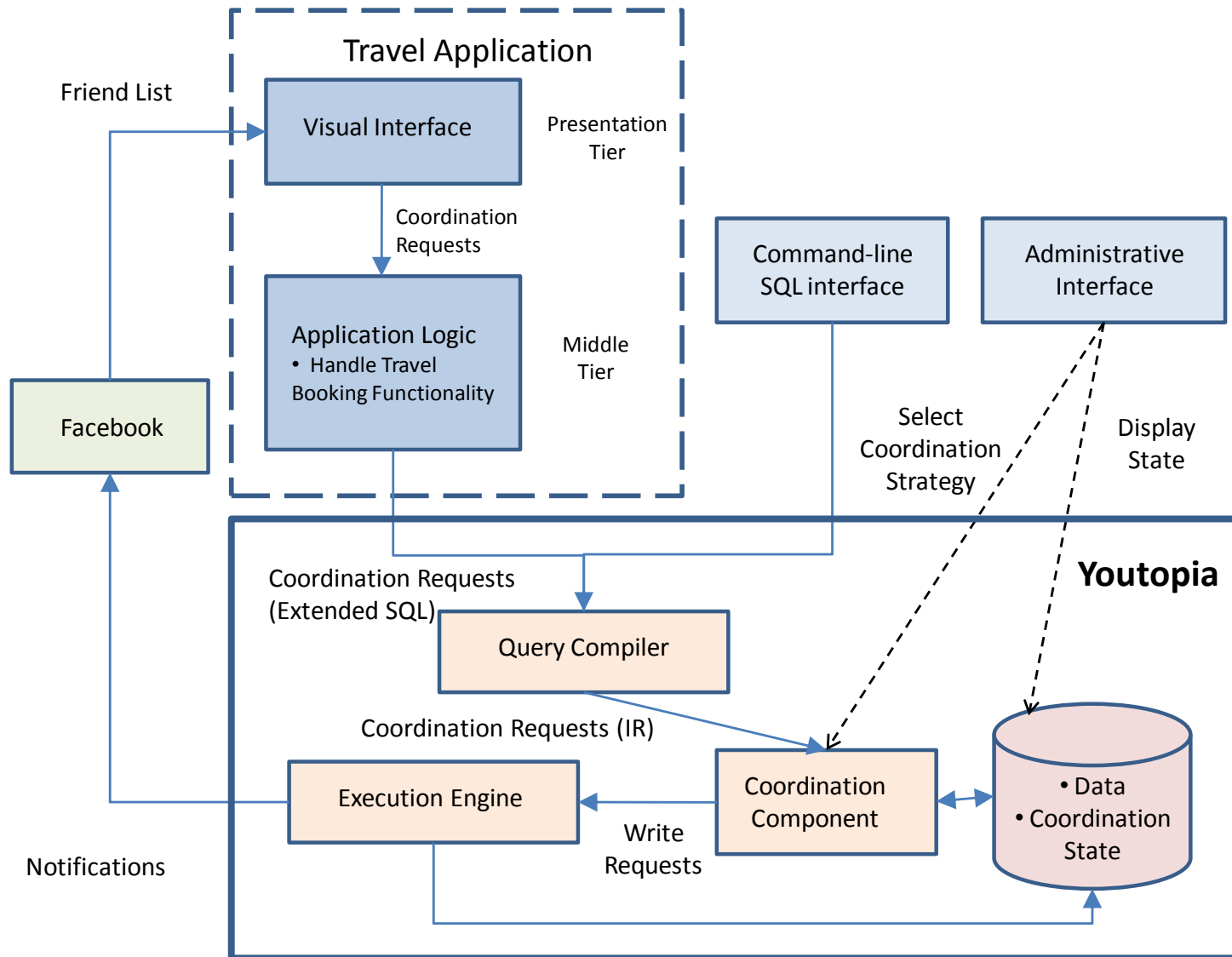
- Inside or outside the DBMS?

How to reconcile asynchronous query submission,
synchronous query answering?

- Staleness
- Incremental evaluation strategies for coordination



System Support for Entanglement



How much of the coordination information should be visible to whom?

- Should the whole answer relation be visible to everyone?
- Should we make the waiting queries visible to other users in order to enable them to join?

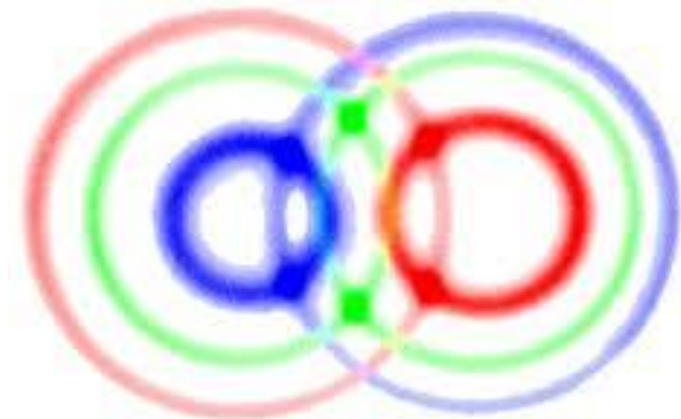
Is it possible to perform malicious attacks involving coordination?

- Make someone coordinate with an unintended partner
- Flood the system with queries to prevent coordination



What is the impact of relaxing isolation on the design of a data-driven systems?

- Isolation has been a cornerstone of the transaction abstraction for a very long time
- It permeates all aspects of database design
- Until today, if isolation has been relaxed, it has been for performance reasons rather than semantic reasons

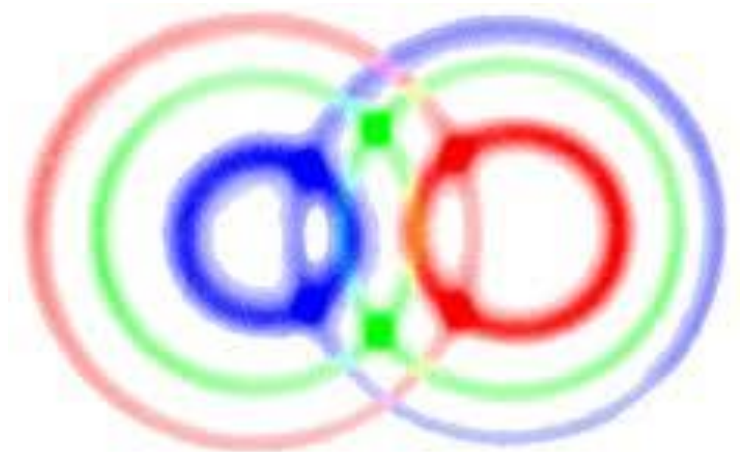


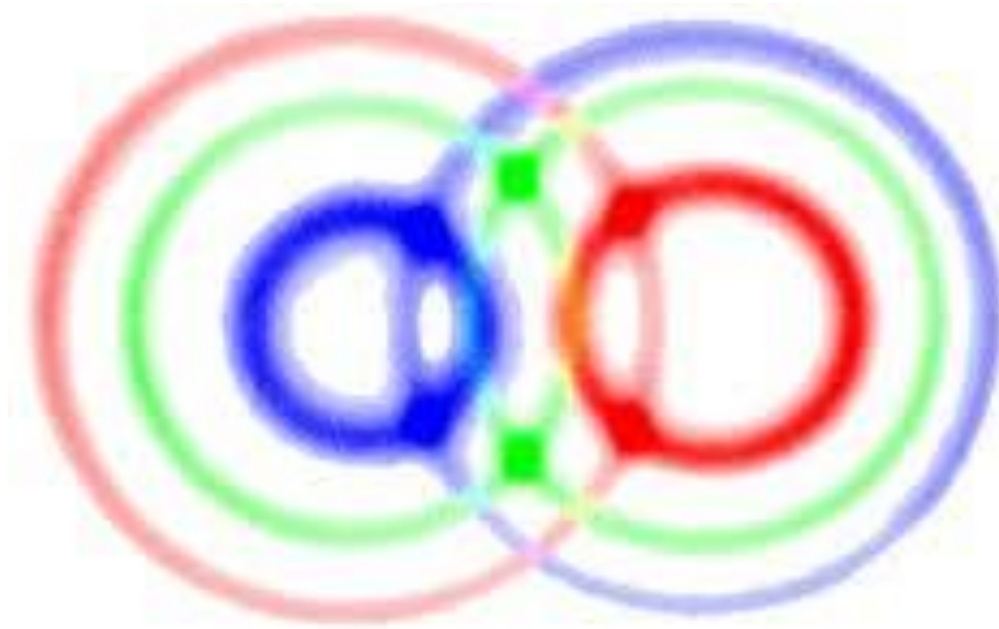
- Many applications require some form of coordination between users
- This coordination should happen at the same level of abstraction as the remainder of the application code

Two abstractions

- Entangled queries
- Entangled transactions

Lots of open research questions





Questions?

<http://www.cs.cornell.edu/johannes>
johannes@cs.cornell.edu

Thank you: AFOSR, IARPA, NSF, Microsoft, Yahoo!, Google, Amazon, NEC.