Big Data Social

Johannes Gehrke
Database Group
Department of Computer Science, Cornell University

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

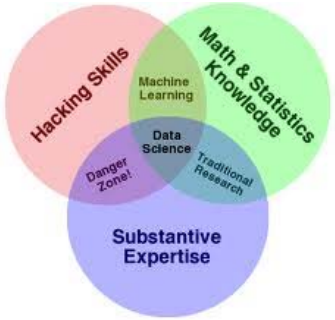
Big Data

Cornell University

- 40% growth in data per year
- Cost of a disk drive to hold the world's music: <$500
- $700B value in personal location data

- What is Big Data?
  Data that is too large to store and analyze using traditional database systems.

- The three "V"s:
  - Volume
  - Velocity
  - Variety



foursquare

Big Data

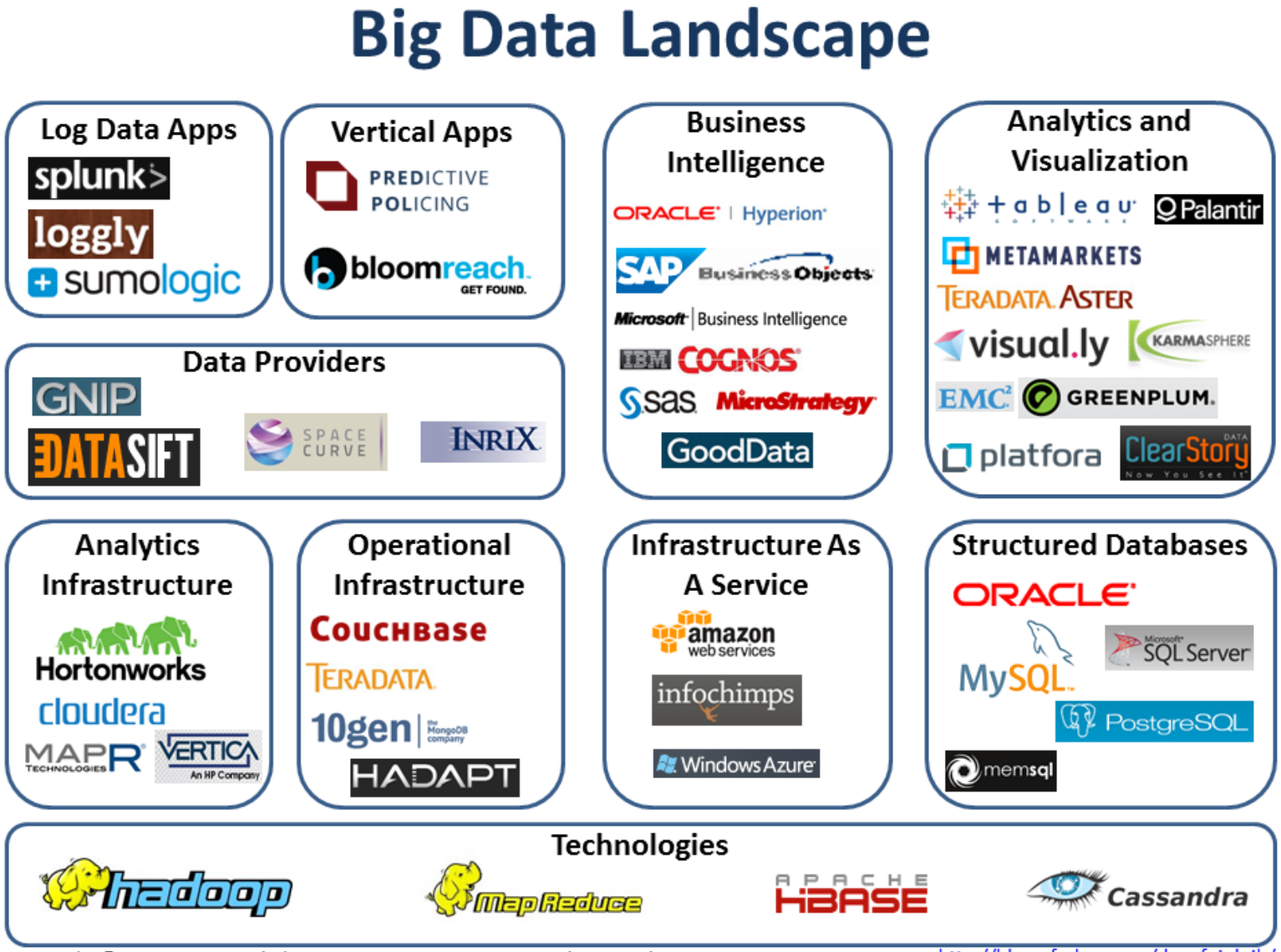
Cornell University

- Opportunities:
  - Turn machine-generated data streams into insights
  - Take the pulse of the world/a sector/an event through social media, location data
  - Create transparency, improve performance of government and health care
  - Augment and replace human decision making with algorithms
- We will lack 140,000 data scientists in 2018

- Two angles:
  - Analysis
  - New experiences

Big Data: Analysis

Cornell University



dave@vcdave.com

http://blogs.forbes.com/davefeinleib/

Big Data: New Experiences

Cornell University

Source: Luma Partners, Terry Kawaja

Coordination: Travel

Cornell University

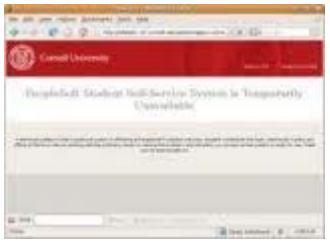
- Assume Tom and Meg want to coordinate itineraries
  - Fly on the same flight, in adjacent seats
  - Also stay in the same hotel if possible

Coordination: Enrollment

Cornell University

Students want to enroll in classes with their friends

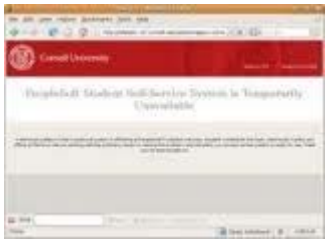
- Help with homework/moral support
- Already happens with out-of-band communication
- CourseRank
  - "Connect to Facebook to find out who of your friends is enrolled"



Coordination: Enrollment

Cornell University

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 Robson takes it too

Coordination: MMOs

Cornell University

- 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

Coordination: SIGMOD 2011

Cornell University

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.

Coordination: Other Examples

Cornell University

- Scheduling meeting times with students/advisees
- 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

Data-Driven Coordination

Cornell University

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 they call each other to jointly enrolling in a course

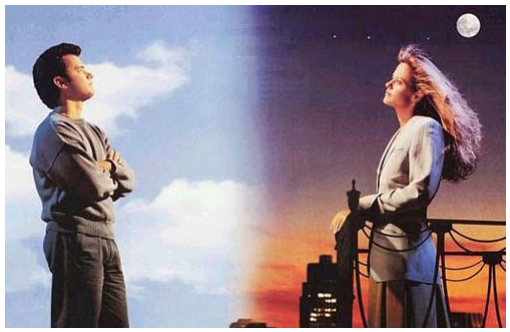
Today typically achieved with out-of-band communication

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

D3C: Declarative Data-Driven Coordination

Cornell University

- 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



D3C and The Legacy of Transactions

Cornell University

ACID Properties of a transaction

- Atomicity
- Consistency
- Isolation
- Durability

D3C requires relaxing isolation

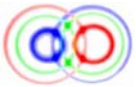
- For semantic reasons, not for performance (such as lower isolation levels, eventual consistency)
- We still want atomicity and durability

And the communication due to coordination should be “controlled“

- “Residual" isolation

Existing Abstractions

- 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



Existing Abstractions (Contd.)

Cornell University

Sagas/nested transactions

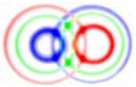
- Transactions inside transactions; only commit when outermost transaction commits

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 …

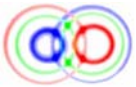
But: Only unidirectional information flow, no matchmaking.



Why A New Abstraction?

Cornell University

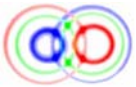
- Need an abstraction that is at the “right” level
  - Data-centric, not process-centric
  - Should not require users to manipulate low-level constructs like channels
- Hide the implementation of the coordination logic
  - The matching that must take place between different users‘ coordination constraints
- 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
- First in a line of other data-driven abstractions for large-scale social



Outline

Cornell University

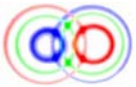
- Introduction
- Entangled queries
  - Language
  - Query Evaluation
- Entangled transactions
- Experimental results
- Open Problems



Outline

Cornell University

- Introduction
- Entangled queries
  - Language
  - Query Evaluation
- Entangled transactions
- Experimental results
- Open Problems



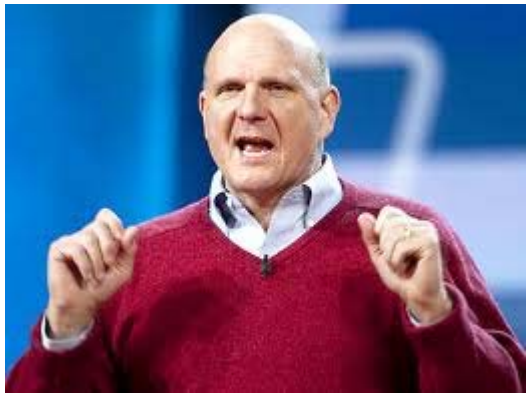
Entangled Queries

Cornell University

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



UNITED

Steve's Entangled Query

Cornell University

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

Steve's Entangled Query

Cornell University


```
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

Larry's Entangled Query

Cornell University

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

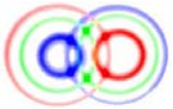
Cornell University

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

Cornell University

Flight

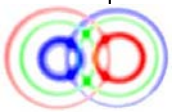
Flightno	Destination
122	JFK
123	JFK
134	JFK
136	Brussels

Airlines

Flightno	Airline
122	United
123	United
134	Lufthansa
136	Alitalia

United Flights 122 and 123 satisfy the constraints.

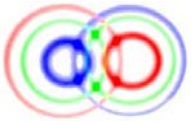
	Steve's query	Larry's query
Answer	(Steve, 122)	(Larry, 122)
Constraint	(Larry, 122)	(Steve, 122)



Query Evaluation

Cornell University

- 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

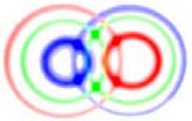


Entangled Queries: Language

Cornell University

```
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, …



Outline

Cornell University

- Introduction
- Entangled queries
  - Language
  - Query Evaluation
- Entangled transactions
- Experimental results
- Open Problems

Evaluating Entangled Queries

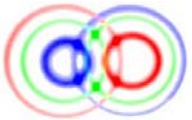
How do we evaluate entangled queries?

Problem: Evaluation is NP-hard in the general case

- Not that surprising: Entangled queries can encode CSP

More than one source of complexity:

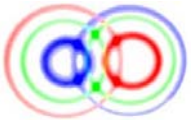
- Matching up the entangled queries
- Finding data values that satisfy coordination constraints



Query Evaluation

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

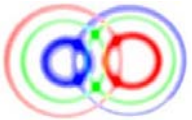


Query Evaluation

Cornell University

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

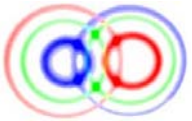


Our Approach

Cornell University

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



Intermediate Represenation

Cornell University

A Datalog-like representation (without recursion)

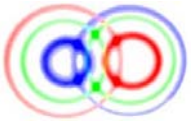
$$\{ C \} \quad H :- B$$

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

- { Booking(Larry,x) } Booking(Steve,x) :- Flight(x, JFK)
- { Booking(Steve,y) } Booking(Larry,y) :- Flight(y, JFK)
  Λ Airline(y, United)



Safety

Cornell University

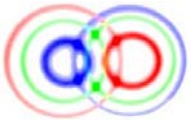
- A set of queries is unsafe if there a query with more than one potential coordination partner


{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



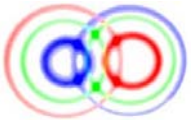
Safety and Unifiability

Cornell University

- 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.

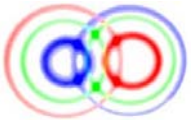
Uniqueness: Either all or none of the users coordinate.



Query Evaluation

Stages of query evaluation

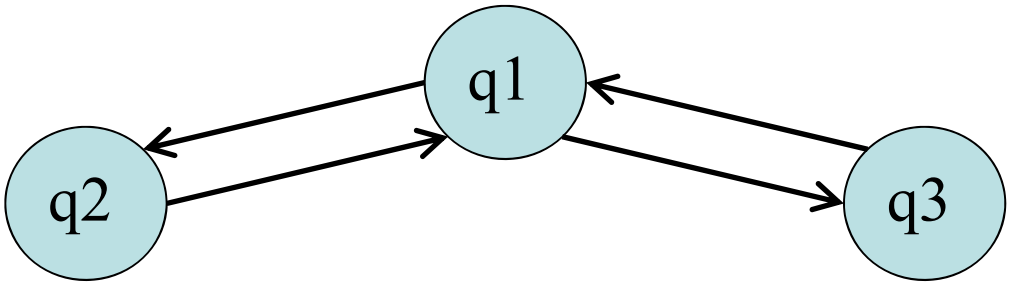
1. Check queries for safety
2. Partition queries into subsets and match queries
3. Create and evaluate the combined query and construct the individual answers

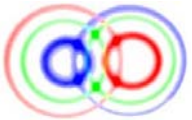


Unifiability Graph

Cornell University

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





Matching

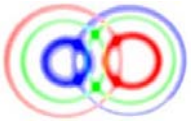
Cornell University

Unifiability graph gives overall structure of how queries match up

But we know more information:

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

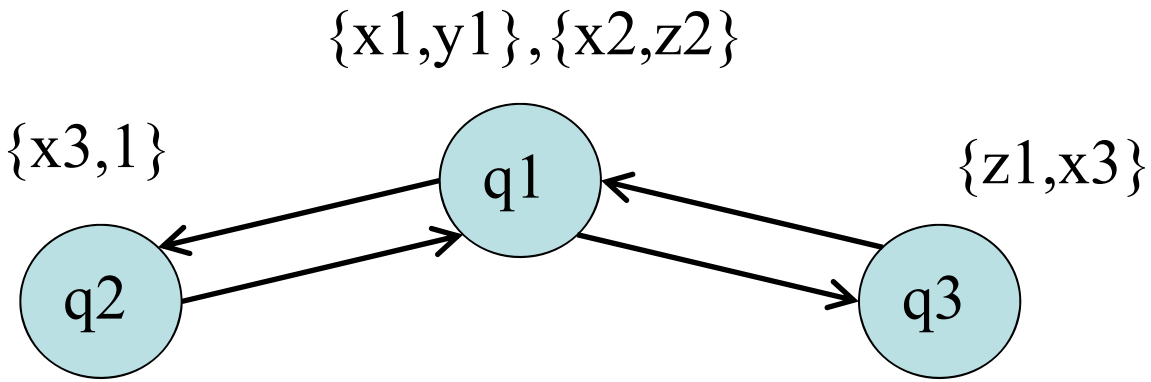
- The head of q1 only satisfies the postcondition of q2 if x3=1
  - Eventually, all the variables will be associated with values from the DB, so we will have a valuation
  - We know coordination is only possible for valuations that assign x3 the value 1



Matching (Contd.)

Cornell University

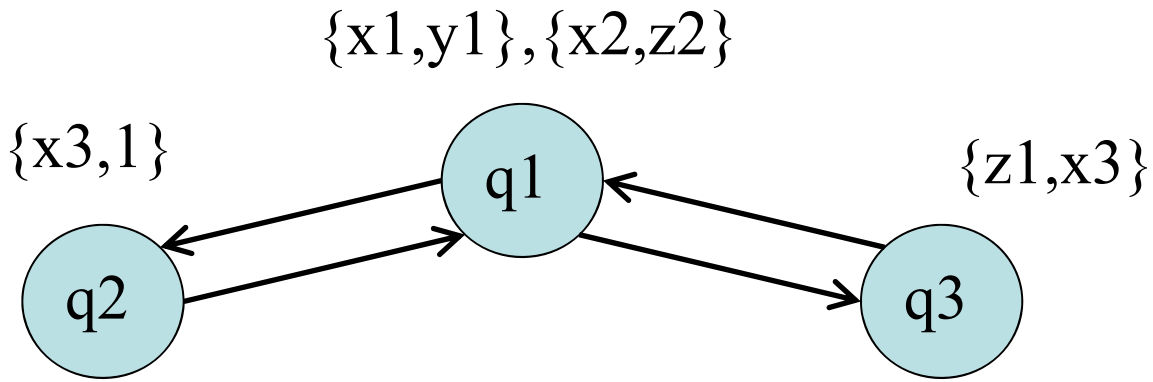
- Represent this information as unifiers associated with nodes in the graph
- A unifier is a constraint imposed by a particular query
  - q1 : {R(x1) ∧ S(x2)} T(x3) :- D1(x1, x2, x3)
  - q2 : {T(1)} R(y1) :- D2(y1)
  - q3 : {T(z1)} S(z2) :- D3(z1, z2)

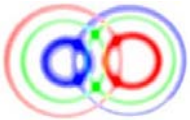


Matching (Contd.)

Cornell University

- 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!



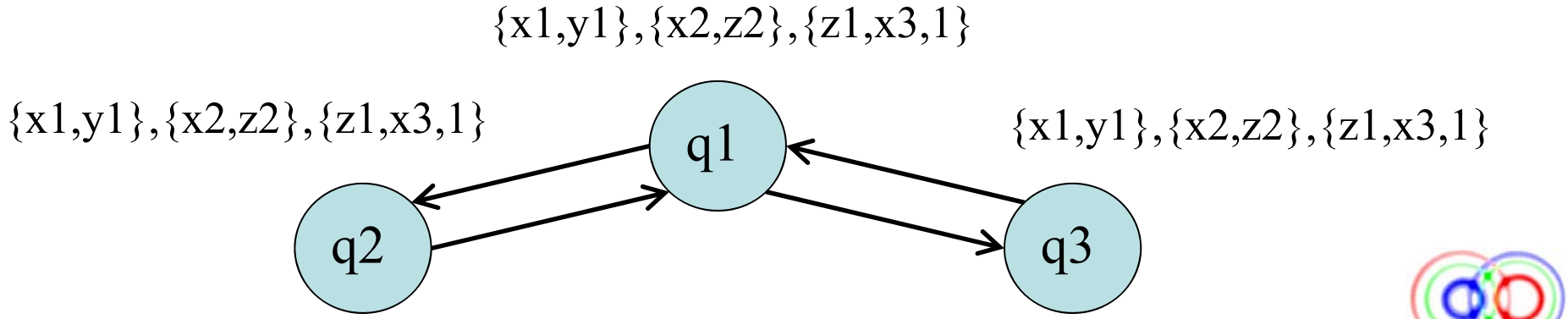


Matching (Contd.)

Cornell University

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 → matching

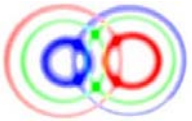


Query Evaluation

Cornell University

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

Cornell University

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

 {Booking(Steve, y)} Booking(Larry, y) :- Flight(y, JFK)
Λ Airline(y, United)

Gets rewritten to:

Booking(Larry, x) Λ Booking(Steve, x) :-
Flight(x, JFK) Λ Airline(x, United)

- Are we done now?

What We Want: Transactions

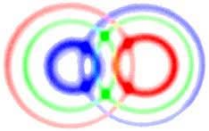
Cornell University

- 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
  - All of this should happen or nothing at all

Outline

Cornell University

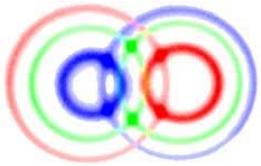
- Introduction
- Entangled queries
- Entangled transactions
- Experimental results
- Open Problems



Entangled Transactions

Cornell University

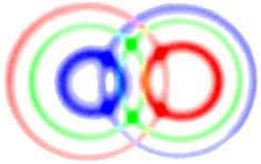
- Goal: Extend transactions to incorporate entangled queries
- Challenge:
  - Relationship of entanglement to classical transactions
- Example scenario
  - Steve and Larry want to fly together to NYC
  - If they can make a flight booking together, then they want to stay in the same hotel



Steve's Transaction

Cornell University

```
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;
```



Consistency

Cornell University

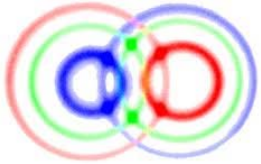
Recall consistency:

- 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;
```



Consistency

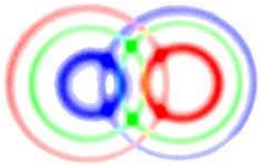
Cornell University

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.



Isolation

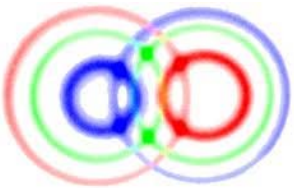
Cornell University

Anomaly-based definition

Two anomalies

- Widowed Transactions
- Unrepeatable quasi-reads

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



Isolation: Basics

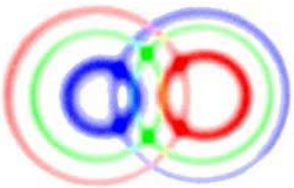
Cornell University

- 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)	

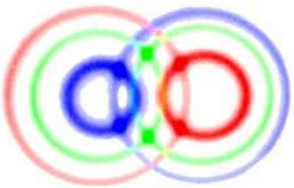


Scheduling Transactions

Cornell University

- 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.



Traditional Anomalies

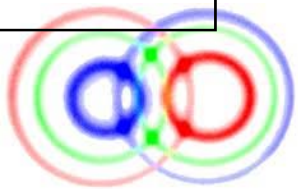
Cornell University

- 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
```

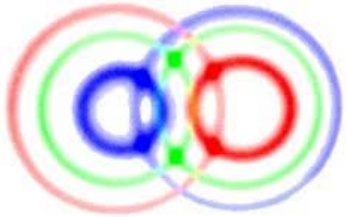


Traditional Anomalies (Contd.)

Cornell University

- Overwriting Uncommitted Data (WW Conflicts):

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

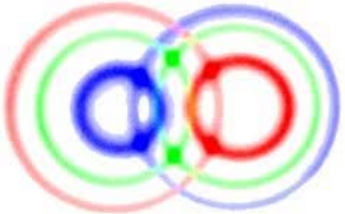


Isolation for D3C

Cornell University

Two new anomalies

- Widowed Transactions
- Unrepeatable quasi-reads



New Anomaly 1: Widowed Transactions

Cornell University

Flight entangled queries

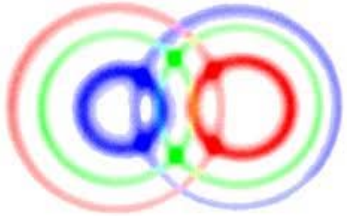
Ticket booking code

Ticket booking code

Hotel entangled queries

Room booking code

ABORT



New Anomaly 2: Unrepeatable Quasi-Reads

Cornell University

Entangled queries

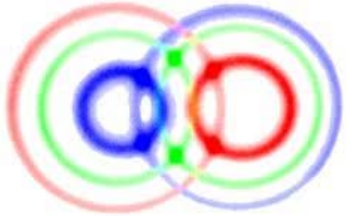
Read **Flights**

Read **Flights**, **Airlines**

Entangled query evaluation

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

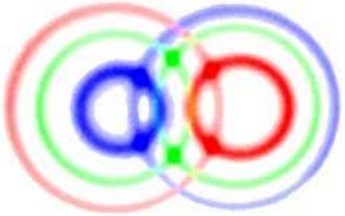
```
INSERT INTO
Airlines
VALUES (`125',
`United')
```



Eliminating These Anomalies

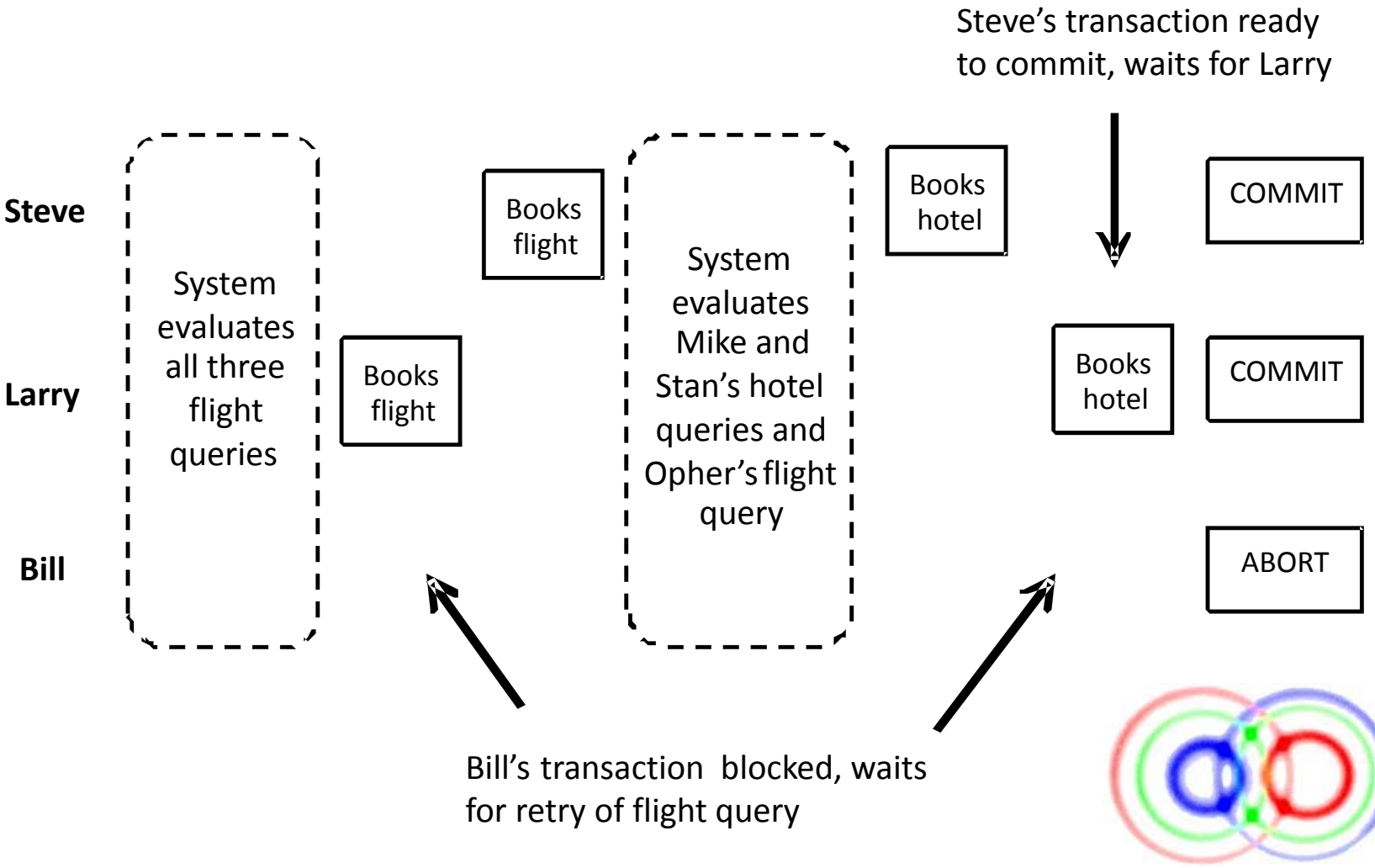
Cornell University

- 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

Cornell University



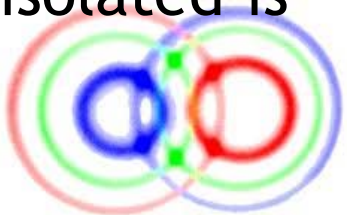
Putting Everything Together

Cornell University

- 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

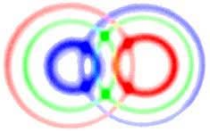
- Main theorem: Any schedule that is entangled-isolated is also oracle-serializable.



Outline

Cornell University

- Introduction
- Entangled queries
- Entangled transactions
- Experimental results
- Open Problems



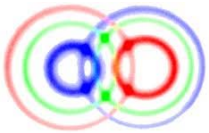
Experimental Setup

Cornell University

- Prototype implemented in Java and uses JDBC to connect to a MySQL database system

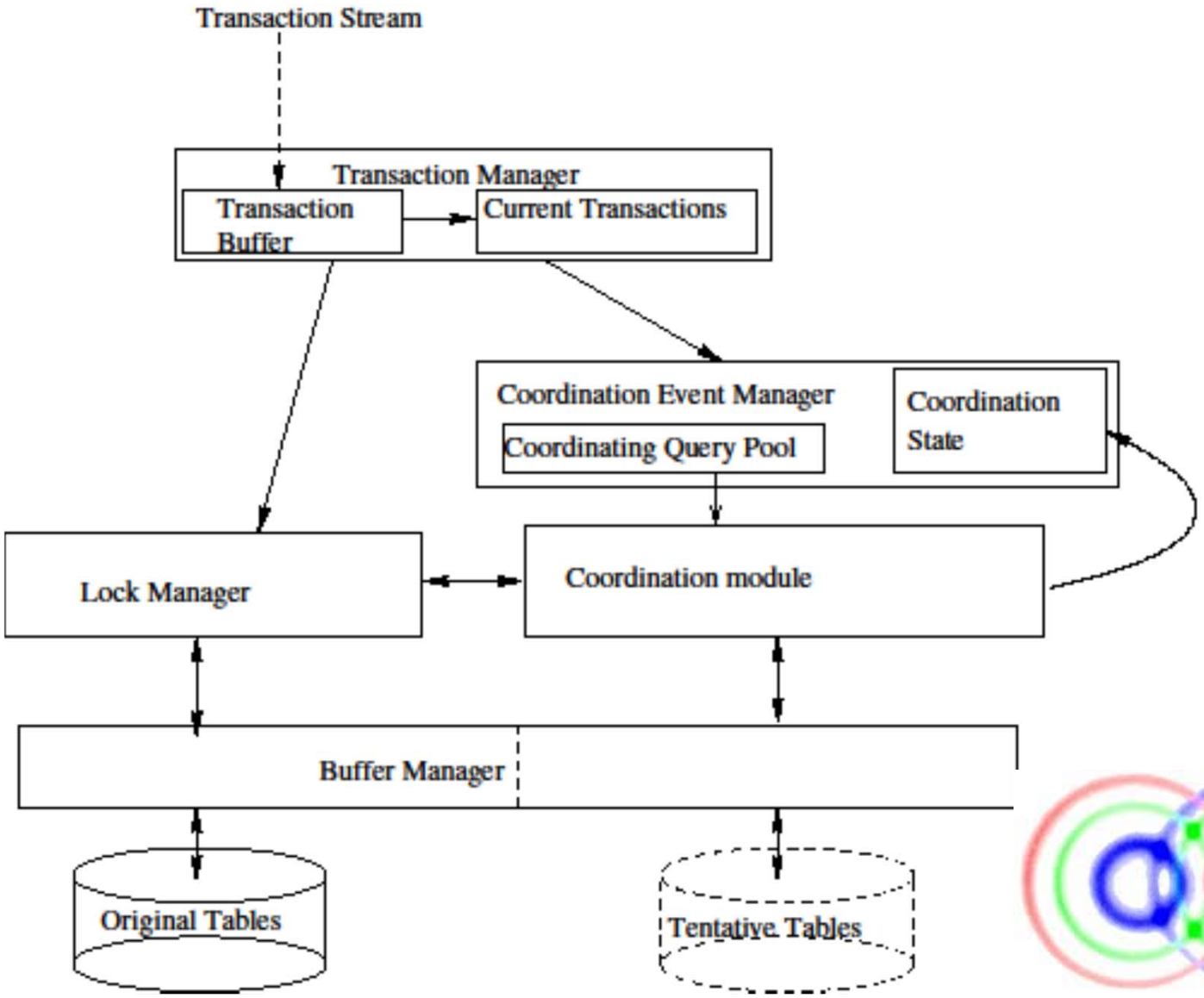
- Dataset:
  - Generate queries that match in pairs or triples
  - Make queries more or less specific (coordinate with a named friend vs. any friend)

- Additional experiments:
  - Increase number of post-conditions per query
  - Stress-test performance of matching algorithm



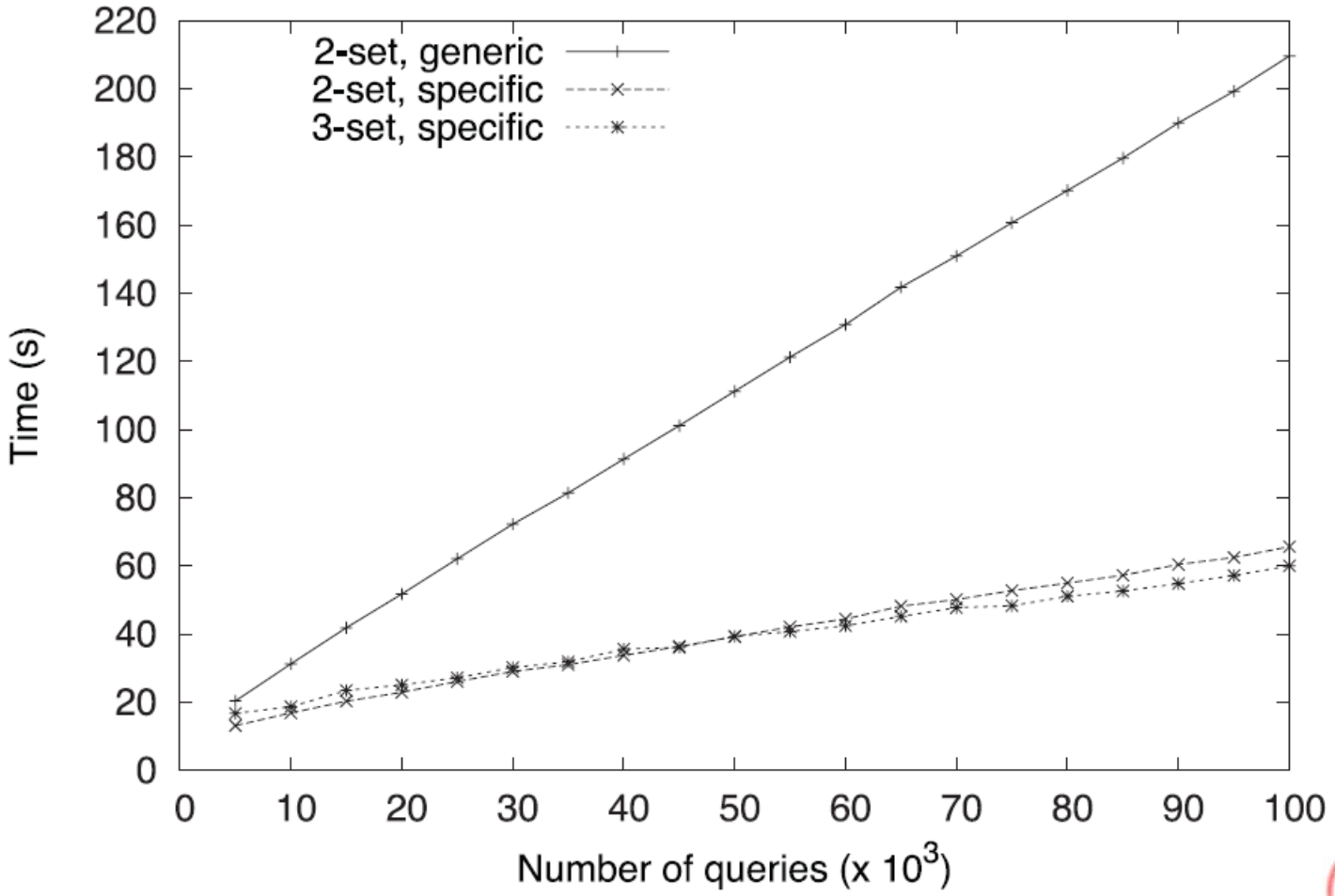
System Architecture

Cornell University



Results: Scalability

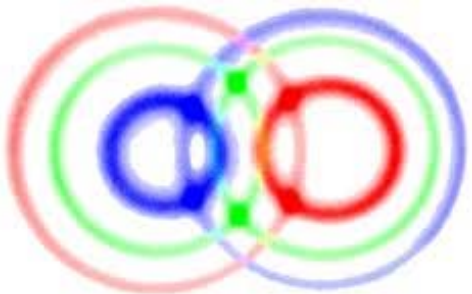
Cornell University



Outline

Cornell University

- Introduction
- Entangled queries
- Entangled transactions
- Experimental results
- Open Problems

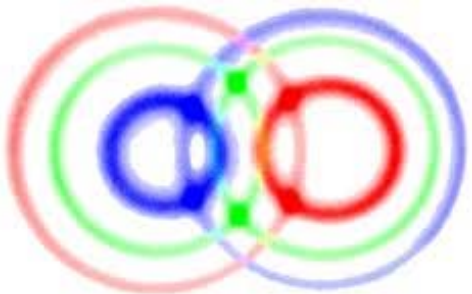


This is Just The Beginning

Cornell University

Many exciting research directions:

- Scaling beyond a single machine
- 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



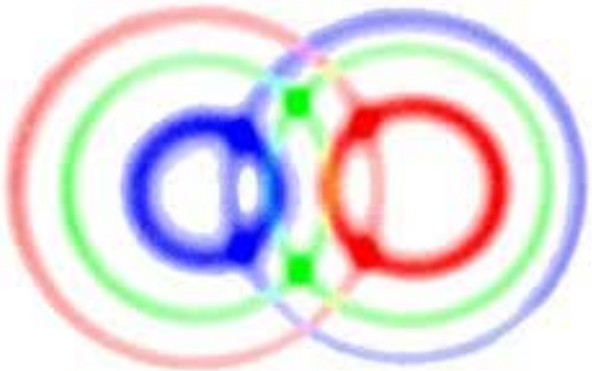
Scaling Beyond A Single Machine

- Scalable graph processing infrastructure
- Transactions/queries are now pending in the system
  - Batch-style execution?
  - Fairness?
- Load balancing across machines
  - But likely small coordination groups
- Groups as a first-class citizen in the system

Language Extensions

Cornell University

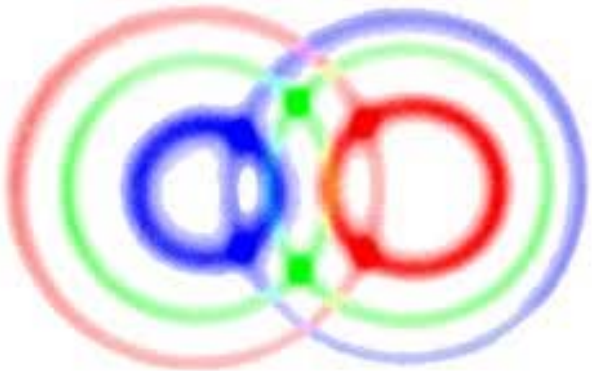
```
SELECT P.partyid, 'Simon' INTO ANSWER Attendance
FROM Parties P
WHERE P.pdate='Friday'
    AND
    (SELECT COUNT(*)
     FROM ANSWER Attendance A, Friend F
     WHERE     P.partyid = A.partyid AND A.name = F.name1
               AND F.name2 = ‘Paul) > 2
CHOOSE 1
```



Language Extensions (Contd.)

Cornell University

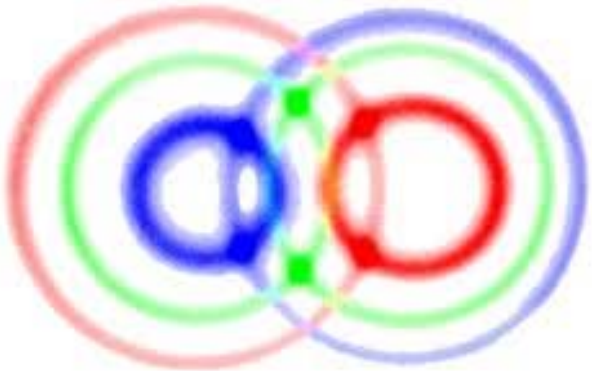
- "Soft" constraints
  - Travel dates should be as close as possible (but need not be identical)
- Preferences
  - Will travel on any US carrier, but prefer United if possible
  - Will travel any day next week, but the earlier the better
- Semantics where more than one record is returned
  - Example: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



Complexity

Cornell University

- 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

Cornell University

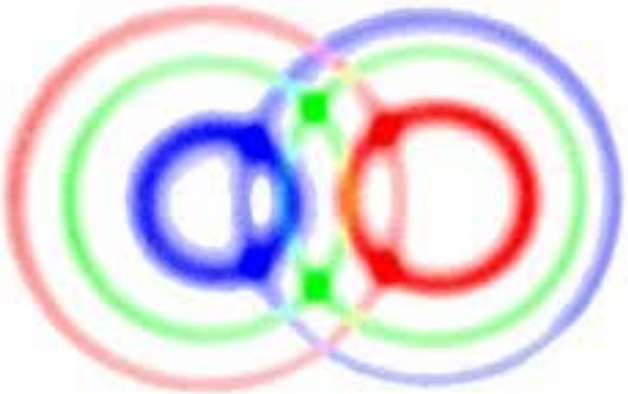
Where does evaluation take place?

- Inside or outside the DBMS?

How to reconcile asynchronous query submission, synchronous query answering?

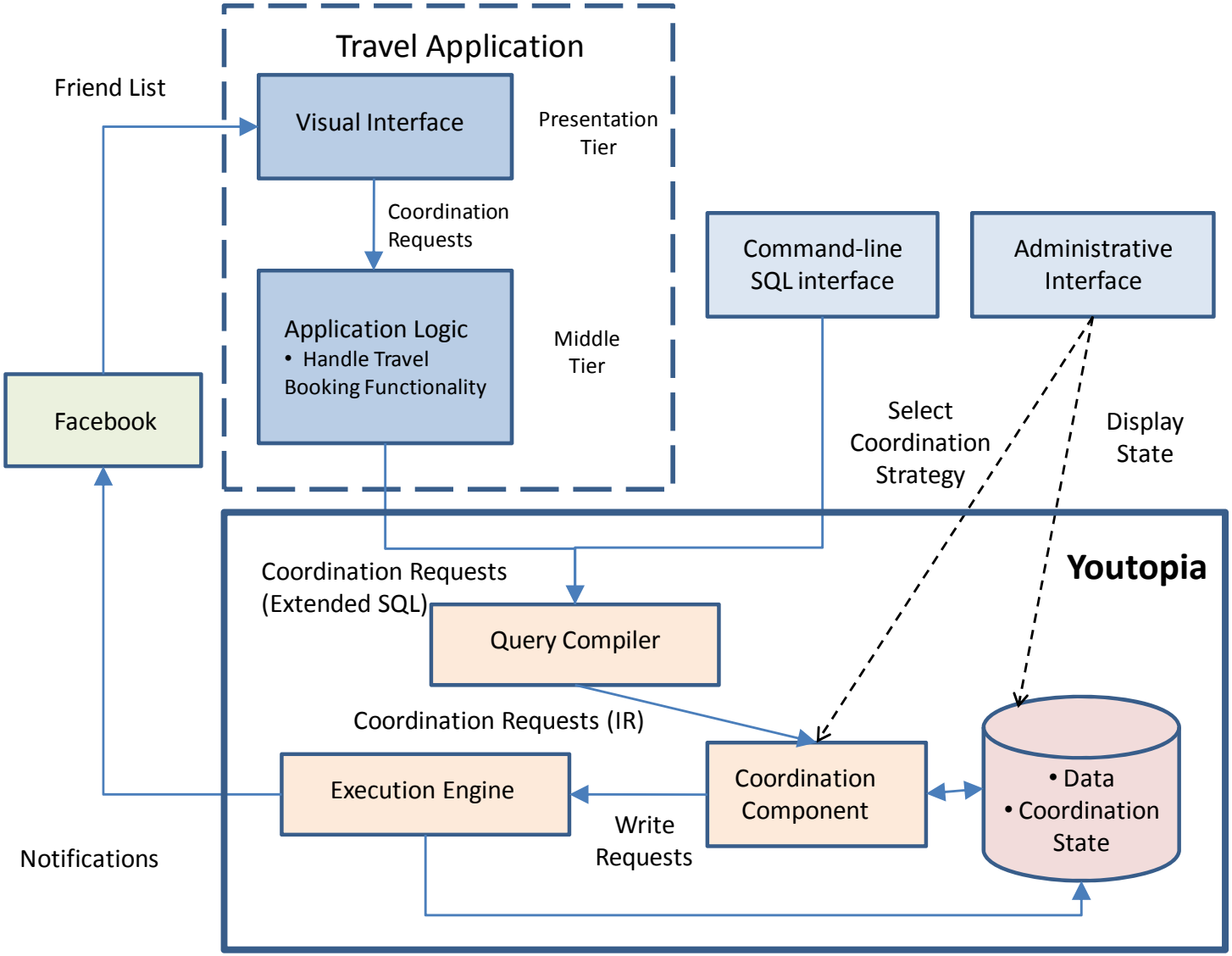
- Staleness
- Incremental evaluation strategies for coordination

Entanglement as a service?



System Support for Entanglement

Cornell University



Privacy and Security

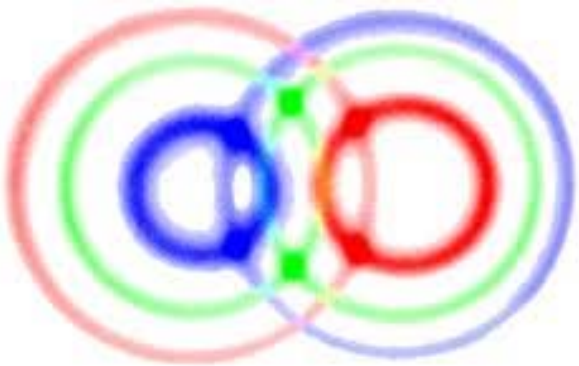
Cornell University

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?

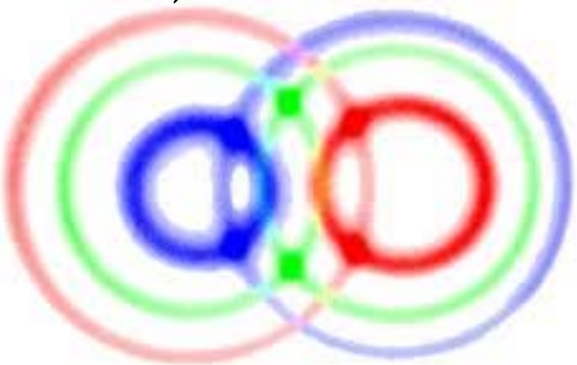
- Make someone coordinate with an unintended partner
- Flood the system with queries to prevent coordination



Beyond Entanglement?

Cornell University

- Note that our model of entangled transactions requires us to know the full transaction in advance
  - No interactive SQL from the middle tier as in all of today's systems
- One idea: Heisenstate™
  - You construct an entangled transaction interactively for a flight and hotel
  - The transaction commits, and thus you are ensured that you have a flight and a hotel, but you do *not* know their values
    - Your transaction has imposed a constraint on the available seats and hotelrooms in the database
  - At a later time, somebody/you reads the value of your seat, and the value gets assigned (the *completion transaction*).
- Of interest beyond entanglement?
  - Database = Records + Constraints?

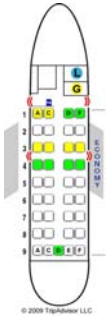


Transactions for Allocation

Cornell University

- Many database applications use transactions to allocate physical or virtual objects based on user requests





- **Resources**: an abstraction for such commodities or objects
  - Represented as data items in database

Transactions for Allocation

Cornell University

- Goal: Allocate resources to users while maximizing global utility
  - Example: Allocate seats in a flight to satisfy the maximum number of user preferences

- Existing transaction models not well-suited for resource allocation
  - Why?

Classical Execution Model

Cornell University

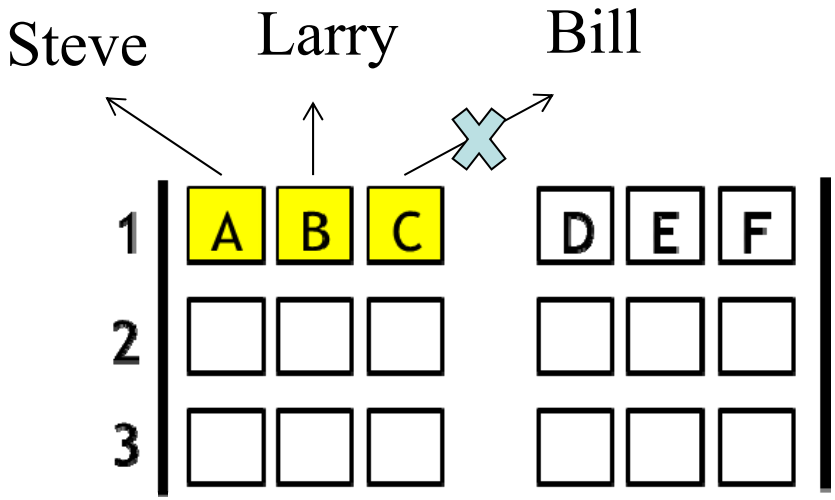
- Steps involved in an allocation transaction
  - User requests resource with constraints
  - System assigns resource to user
  - Transaction commits

- Example: Steve's transaction for booking a seat
  - Steve requests a window seat
  - System checks and assigns one available window seat
  - Transaction commits

Example: Flight Seat Allocation

Cornell University

- Resources: Seats
- Transactions: Book seats
- Scenario
  - 3 seats available
  - Steve: Book any seat
  - Larry: Book any seat
  - Bill: Book any window seat

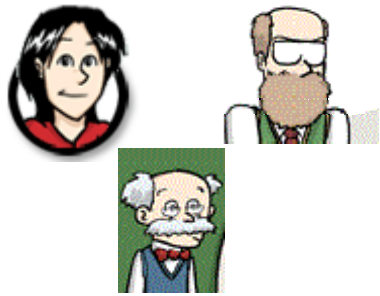


- Result
  - Bill's transaction aborts
- Issue
  - Steve's and Larry's seat assigned before Bill's transaction arrives

Example: Calendar Management

Cornell University

- Resources: Time slots in calendar
- Transactions: Schedule meeting in a particular slot
- Scenario:
  - Prof. Jones: Meet Cecilia at on Monday at noon
  - Dept. Chair: Meet Prof. Jones on Monday at noon

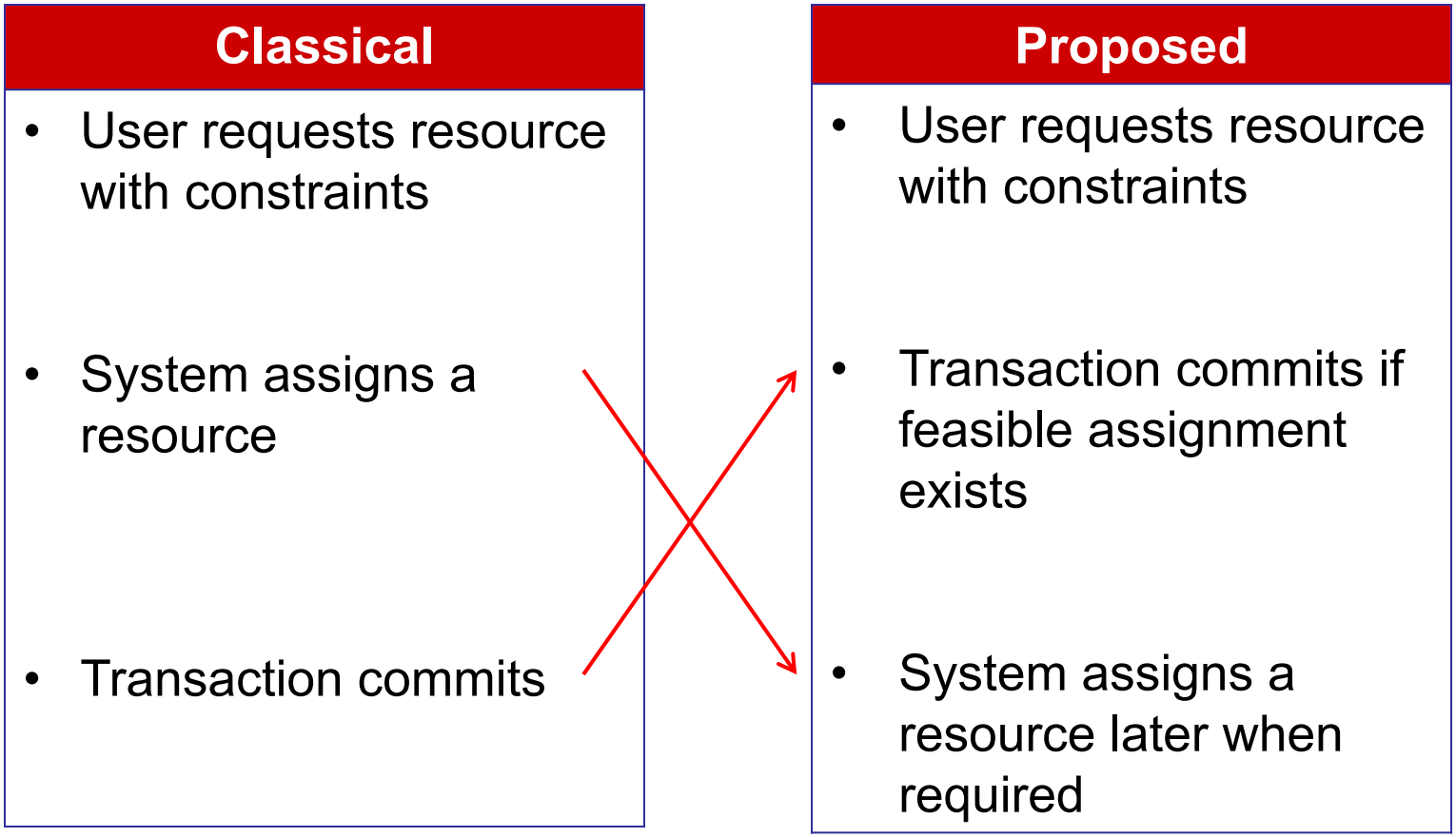


- Result
  - Compensating transaction to manually reschedule meeting with Cecilia to another time on Monday
  - Potentially leads to cascading compensations
- Issue
  - Prof. Jones fixed meeting time with Cecilia before Dept. Chair scheduled a conflicting meeting

Quantum Databases

Cornell University

- Push assignment of resources *beyond* transaction commit



Proposed Execution Model

Cornell University

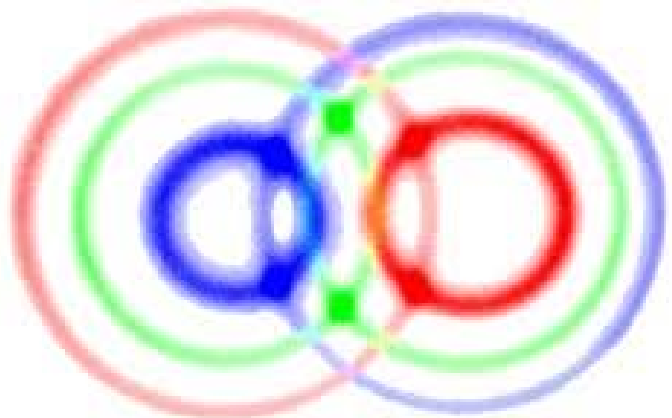
- Example
  - Steve requests a window seat
  - Check availability of a window seat
  - Commit without assigning specific seat
  - System guarantees Steve **some** window seat
  - Assign seat when required
- Assumption: Delay between transaction commit and reading assigned resource
- Core Idea: Late binding of unread values in transactions
  - Allows for more informed and better allocation

Final Thought

Cornell University

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



Summary

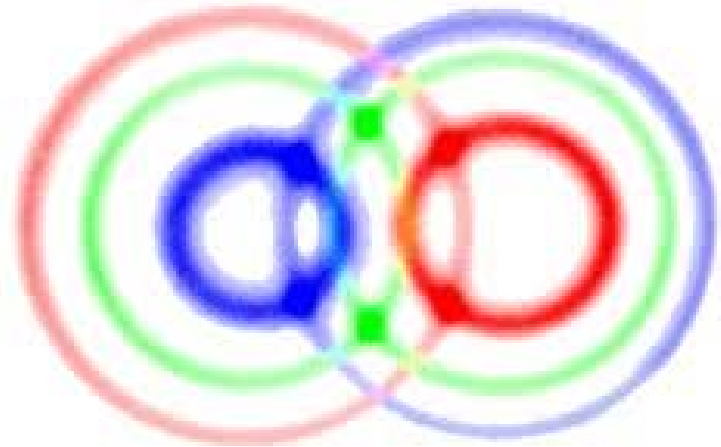
Cornell University

- 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

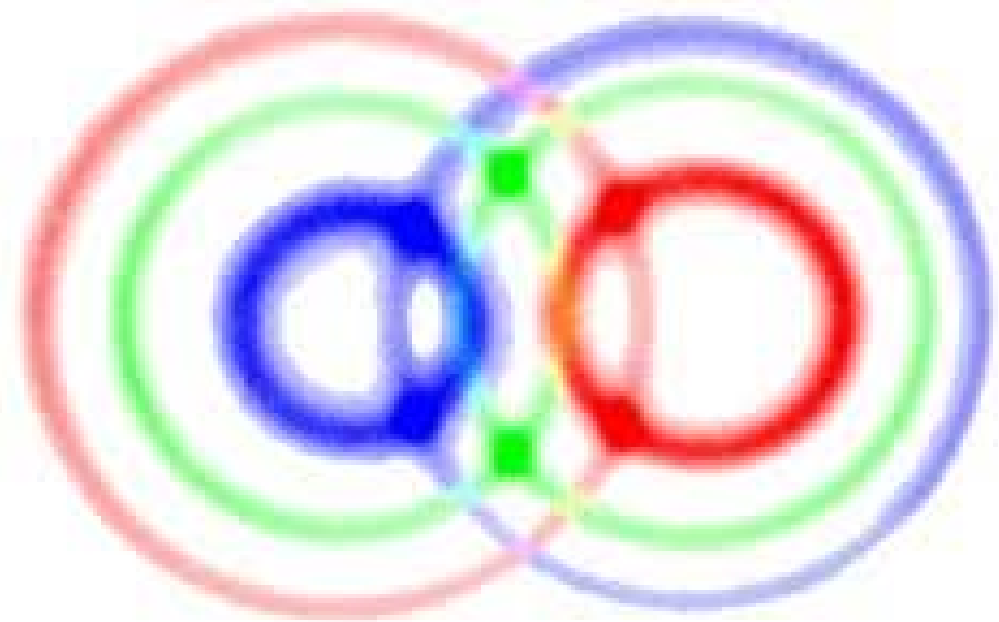
Lots of open research questions.



Related Publications

Cornell University

- Nitin Gupta, Lucja Kot, Sudip Roy, Gabriel Bender, Johannes Gehrke, Christoph Koch. Entangled Queries: Enabling Declarative Data-Driven Coordination. Best paper award, SIGMOD 2011.
- Nitin Gupta, Milos Nikolic, Sudip Roy, Gabriel Bender, Lucja Kot, Johannes Gehrke, Christoph Koch. Entangled Transactions. VLDB 2011.
- Sigal Oren, Konstantinos Mamouras, Lior Seeman, Lucja Kot, and Johannes Gehrke. The Complexity of Social Coordination. VLDB 2012.
- Cheng Li, Daniel Porto, Allen Clement, Rodrigo Rodrigues, Nuno Preguiça, Johannes Gehrke. Making Geo-Replicated Systems Fast if Possible, Consistent when Necessary. OSDI 2012.
- Sudip Roy, Lucja Kot, Christoph Koch: Quantum Databases. CIDR 2013



Questions?

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

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