Google™

Mapping Relational Data Model Patterns To The App Engine Datastore

Max Ross
November 19, 2009

Google™

Agenda

- App Engine Datastore Basics
- Soft Schemas
- Moving To App Engine
- Leaving App Engine
- Questions

Google

The App Engine Datastore

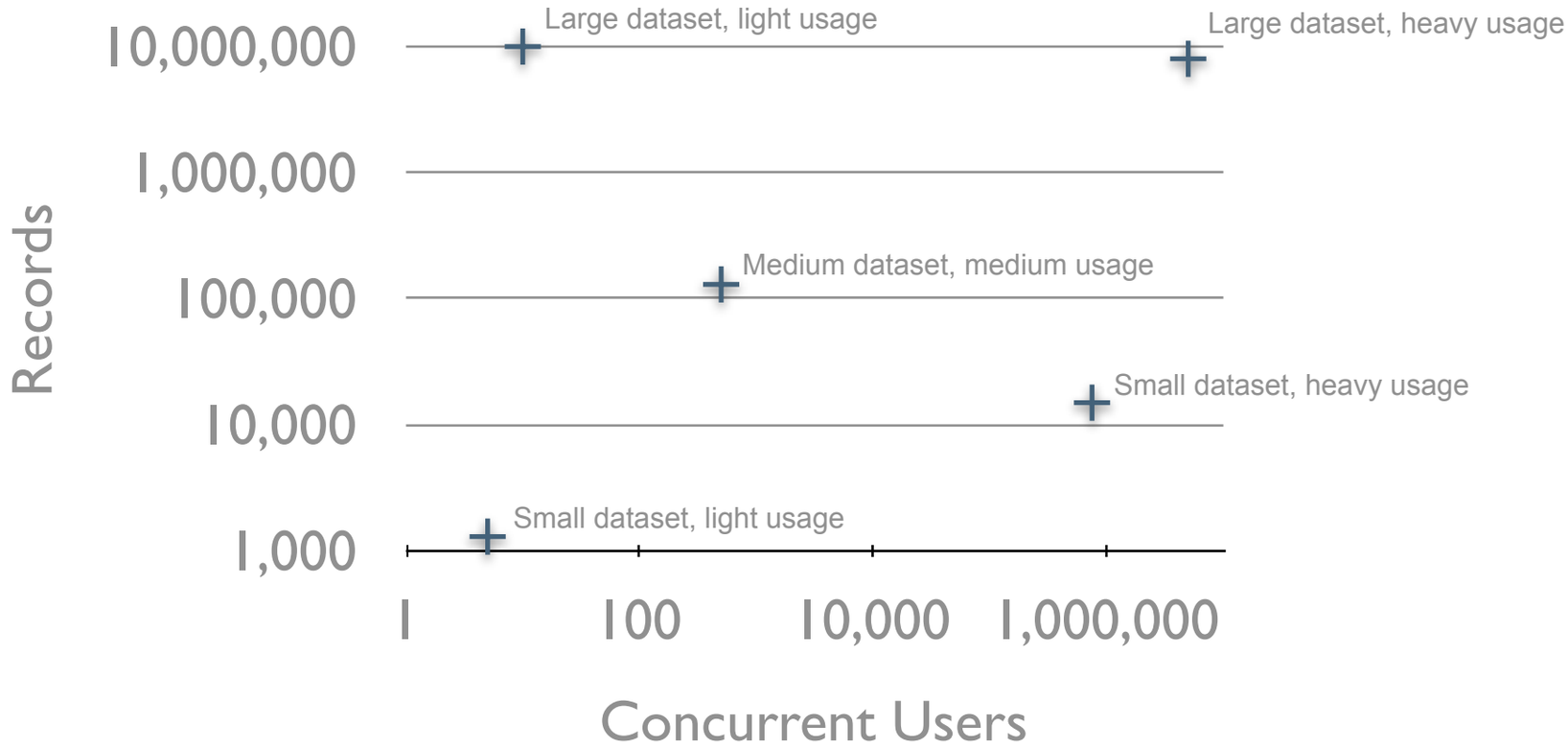
The Datastore Is...

- Transactional
- Natively Partitioned
- Hierarchical
- Schema-less
- Based on Bigtable
- Not a relational database

Google

Simplifying Storage

- Simplify development of apps
- Simplify management of apps
- Scale *always* matters
  - Request volume
  - Data volume



Google

What's The Value Prop?

- Free to get started
- Pay only for what you need
- Let someone else manage
  - upgrades
  - redundancy
  - connectivity
- Let someone else scramble when things go south
- Scale automatically to any point on the scale curve

- Remember this when I'm telling you what you have to give up!

Google

Datastore Storage Model

- Basic unit of storage is an Entity consisting of
  - Kind (table)
  - Key (primary key)
  - Entity Group (partition)
  - 0..N typed Properties (columns)

Kind	Person	
Entity Group	/Person:Ethel	
Key	/Person:Ethel	
Age	Int64: 30	
Best Friend	Key:/Person:Sally	Key:/Person:Dave

Google

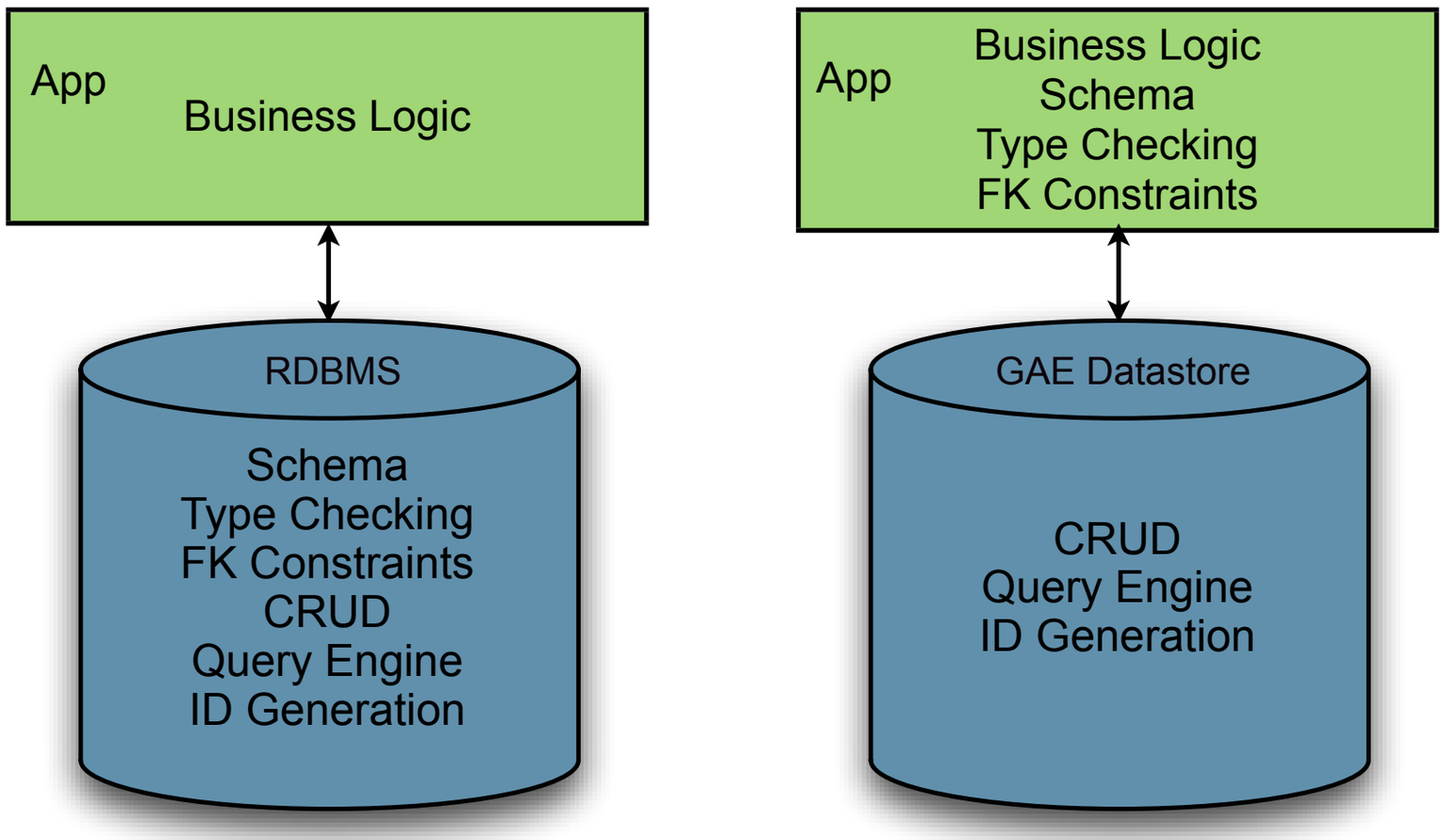
7

Soft Schemas

“A soft schema is a schema whose constraints are enforced purely in the application layer.”

Soft Schemas

- App's expectations define the schema
- Simpler development process
  - Rapid *typesafe* prototyping
- Think about data in a familiar way



Google

10

JPA

- Use JPA to define the soft schema

```
@Entity
class Book {
  @Id
  Long id;
  String author;
  Date publishDate;
  // ...
}

List<Book> getBooksByAuthor(EntityManager em, String author) {
  Query q = em.createQuery(
    "select from Book where author = :a order by publishDate");
  q.setParameter("a", author);
  return q.getResultList();
}
```

- Reuse existing tools, apis, and knowledge
- You're not giving up as much as you think!

Google

Moving To App Engine

Sub-Agenda

- Primary Keys
- Transactions
- Relationships
- Queries

Google

Primary Keys

- What's different?
  - kind (table) is part of the pk
  - hierarchical

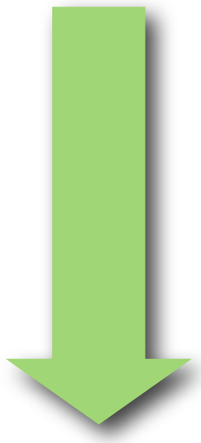
/Person:13/Pet:Ernie

  - Person 13 is the parent of the pet named Ernie

Google

Primary Keys - Composite Example

PET	
PET_ID (pk)	PERSON_ID (pk)(fk)
Ernie	13



Key	/Person:13/Pet:Ernie

Google

Primary Keys - Surrogate Example

PET		
PET_ID (pk)	PET_NAME (u)	PERSON_ID (fk) (u)
88	Ernie	13

Key	/Person:13/Pet:Ernie

Key	/Pet:88
PetName	Ernie
PersonId	/Person:13

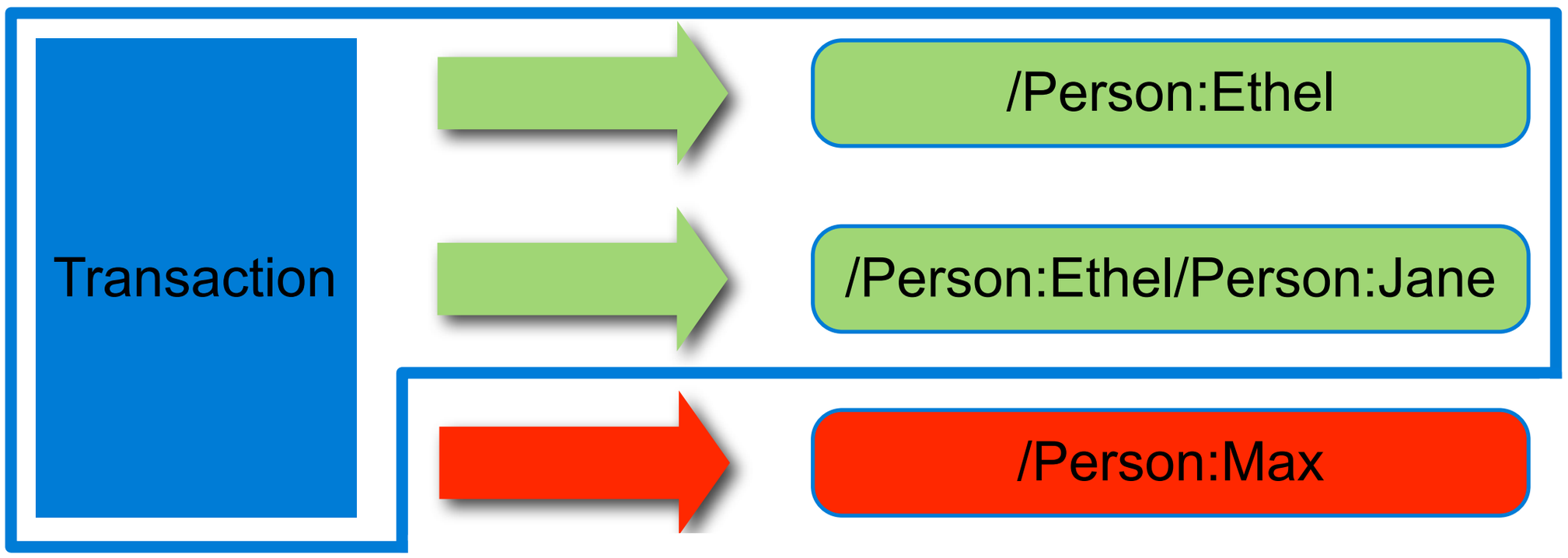
Key	/Person:13/Pet:Ernie
PetId	88

Google

16

Transactions

- What's different?
  - Transactions apply to a single Entity Group

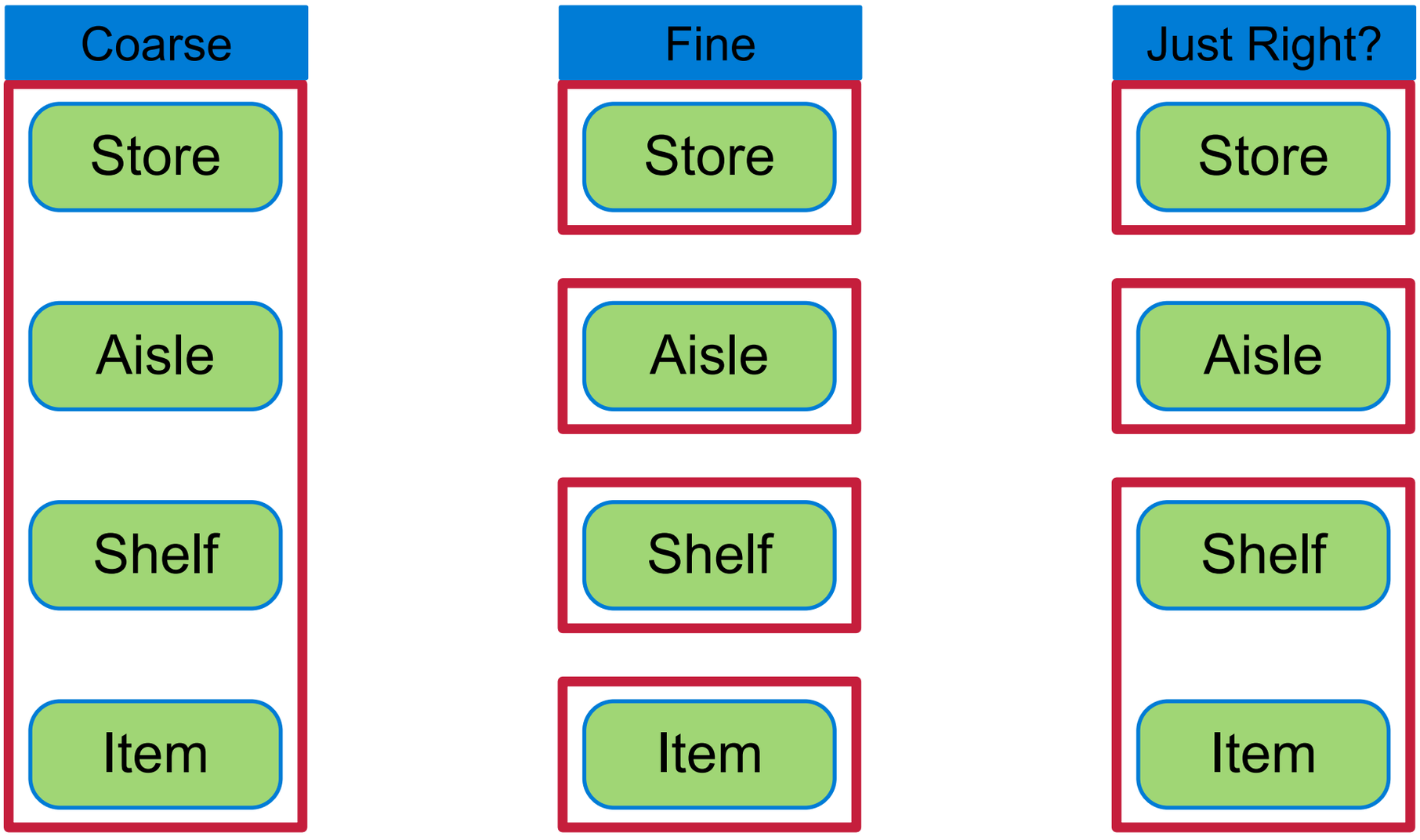


Google

17

Transactions - Entity Group Selection

- Critical design choice
- Too coarse hurts throughput
- Too fine limits usefulness of transactions



Google

18

Transactions - Eventual Consistency

- Use transactional tasks to update multiple entity groups

Google

Transactions - Eventual Consistency

- Use transactional tasks to update multiple entity groups

```
1  void updateBalance(EntityManager em, Account act, int balance,
2      TaskOptions taskOpts) {
3    em.getTransaction().begin();
4    act.setBalance(balance);
5    em.merge(act);
6    if (taskOpts != null) {
7      QueueFactory.getDefaultQueue().add(taskOpts);
8    }
9    em.getTransaction().commit();
10 }
```


Google

Transactions - Eventual Consistency

- Use transactional tasks to update multiple entity groups

```
1  void updateBalance(EntityManager em, Account act, int balance,
2      TaskOptions taskOpts) {
3    em.getTransaction().begin();
4    act.setBalance(balance);
5    em.merge(act);
6    if (taskOpts != null) {
7      QueueFactory.getDefaultQueue().add(taskOpts);
8    }
9    em.getTransaction().commit();
10 }
```

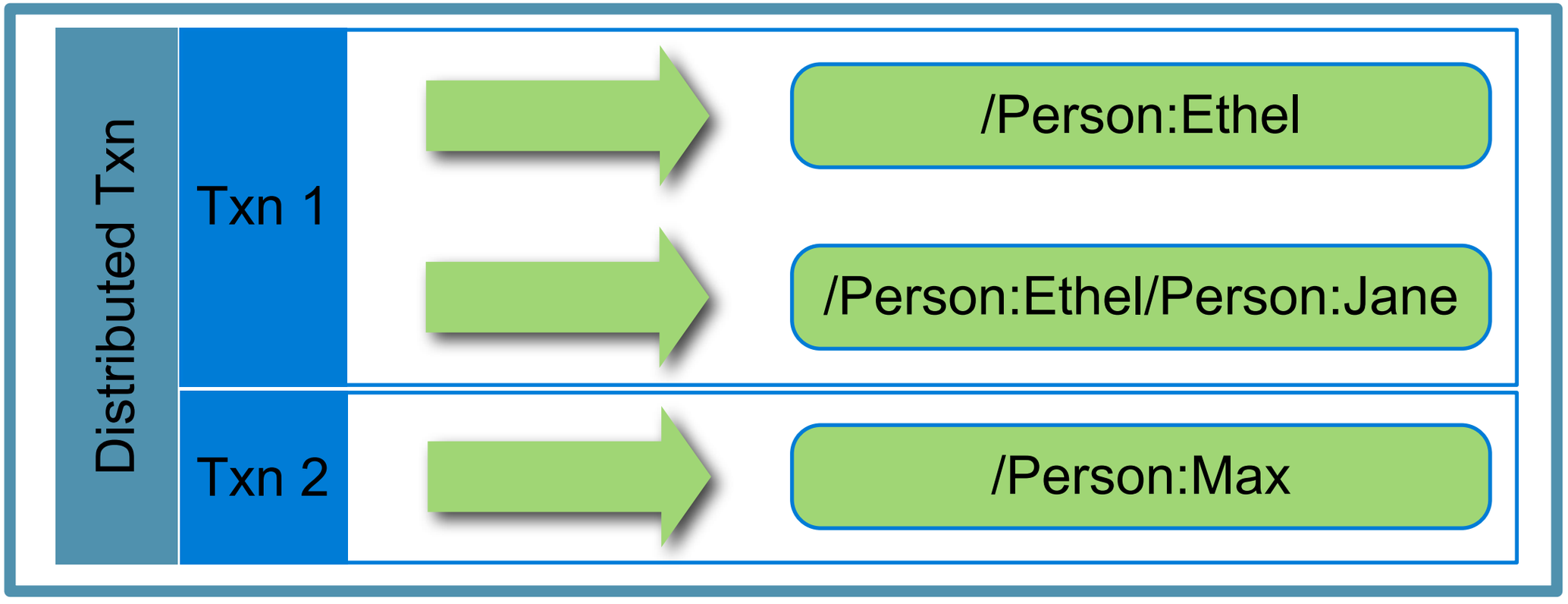
```
11 void transferCash(EntityManager em, Account from, Account to,
12      int amount) {
13   TaskOptions taskOpts = newTask(to, to.getBalance() + amount);
14   updateBalance(em, from, from.getBalance() - amount, taskOpts);
15   updateBalance(em, to, to.getBalance() + amount, null);
16 }

17 TaskOptions newTask(Account act, int newBalance) {...}
```

Google

19

Transactions - What About 2PC?

- Similar limitations in a typical sharded db deployment
- Why not consider a typical sharded db deployment solution?
- Two phase commit
  - Dan Wilkerson (Berkeley) developed the algo
  - Erick Armbrust (Google) implemented it



Google

Relationships

- Letting a framework manage relationships can simplify code
  - True for RDBMS
  - Especially true for App Engine Datastore

- Relationships can be described as “owned” or “unowned”
- Ownership implies co-location within an Entity Group

Google

Owned One To Many

```
@Entity
class Person {
  // ...
  @OneToMany(mappedBy = "owner")
  List<Pet> petList;
}
```

```
@Entity
class Pet {
  // ...
  @ManyToOne
  Person owner;
}
```

```
void createPersonWithPet(EntityManager em) {
  em.getTransaction().begin();
  Person p = new Person("max", "ross");
  p.addPet(new Pet("dog", "ernie"));
  em.persist(p);
  em.getTransaction().commit();
}
```

Kind	Person
Entity Group	/Person:13
Key	/Person:13

Kind	Pet
Entity Group	/Person:13
Key	/Person:13/Pet:18

Google

22

Queries

- Testing set membership (RDBMS)
  - Give me all users who do yoga
    - Requires a join table

```
@Entity
class User {
  // ...
  List<UserHobby> hobbies;
}
```

```
@Entity
class UserHobby {
  // ...
  User user;
  String hobby;
}
```

```
select from User u JOIN u.hobbies h where h.hobby = 'yoga'
```


Google

Queries Continued

- Testing set membership (GAE Datastore)
  - Give me all users who do yoga
    - Use a multi-value property!

```
@Entity
class User {
  // ...
  List<String> hobbies;
}

            select from User where hobbies = 'yoga'
```

- Simpler and more efficient!

Google

Why We Don't Support Joins (yet)

- Our commitment:
  - Query performance scales linearly with the size of the ***result*** set
- Feasible for joins?

```
select * from Student s JOIN s.courses c where
  c.department = 'Biology' and s.grade = 10 order by s.lastName
```

  - How can we return the first result without constructing a complete cross product?
- Making good progress
  - Working algo for a subset of join queries!
  - Based on merge-join
  - Not production ready

Google

In The Meantime...

- RDBMS encourages cheap writes and expensive reads
- Datastore encourages expensive writes and cheap reads
  - Denormalization is not a dirty word!

```
@Entity
class Student {
  // ...
  int grade;
  List<Course> courses;
  List<String> courseDepartments;
}

EntityManager em = getEntityManager();
em.createQuery("select from Student where grade = 10 and
  courseDepartments = 'biology').getResultList();
```

- What happens when a course switches departments?

Google

26

Leaving App Engine

Taking Your Code To Someone Else's Party

- App Engine persistence generally more restrictive
  - Primary Keys
  - Queries
  - Transactions

- Decide what portability means and how important it is
  - To Key or not to Key?
  - Multi-value properties

- Congratulations, you've already sharded your data model!

Google

Portable Root Object

```
@Entity
class Book {
  @Id
  String id;
  String title;
  // ...
}
```

Kind	Book
Entity Group	/Book:2
Key	/Book:2
Title	Vineland

BOOK	
ID (pk)	TITLE
2	Vineland

Google

Portable Child Object

```
@Entity
class Chapter {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Extension(vendorName = "datanucleus", key = “gae.encoded-pk”)
  String id;

  @Extension(vendorName = “datanucleus”, key = “gae.parent-pk”)
  Long bookId;

  String pages;
  // ...
}
```

Kind	Chapter
Entity Group	/Book:2
Key	/Book:2/Chapter:8
Pages	23

CHAPTER		
ID (pk)	BOOK_ID (pk)(fk)	PAGES
8	2	23

Google

30

Key Takeaways

- App Engine Datastore simplifies persistence

- JPA adds typical RDBMS features to the datastore

- Important to understand how the datastore is different
  - Even if you're starting from scratch!

- Easier to move apps off than on

- If portability is important, plan for it!

Google

Questions

More Information

- http://code.google.com/appengine
- http://groups.google.com/group/google-appengine-java
- http://gae-java-persistence.blogspot.com
- http://code.google.com/p/tapioca-orm (dt library)

- App Engine Chat Time
  - irc.freenode.net#appengine
  - First and third Wednesday of each month

- maxr@google.com

Google

33