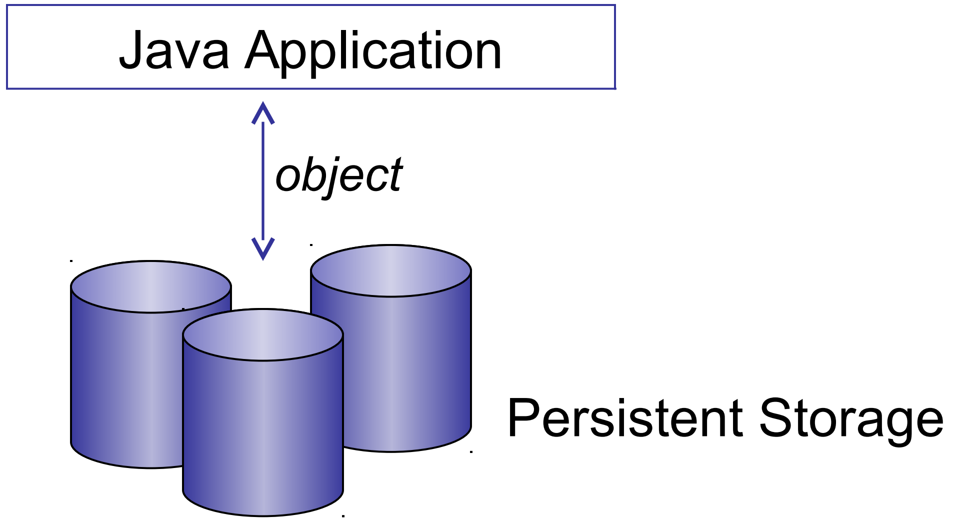
Object Persistence and Object-Relational Mapping

James Brucker

Goal

- Applications need to save data to *persistent storage*.
- Persistent storage can be database, directory service, XML files, spreadsheet, ...
- For O-O programming, we'd like to save and retrieve *objects* to/from storage.



The Problem with Databases

Object-Relational Paradigm Mismatch

- Database stores data as rows in *tables*, which are not like objects.
- Objects have associations and collections
  databases have relations between tables.
- Objects are unique,
  database row data is *copied* each time you query it.

Object-Relational Mapping

Purpose

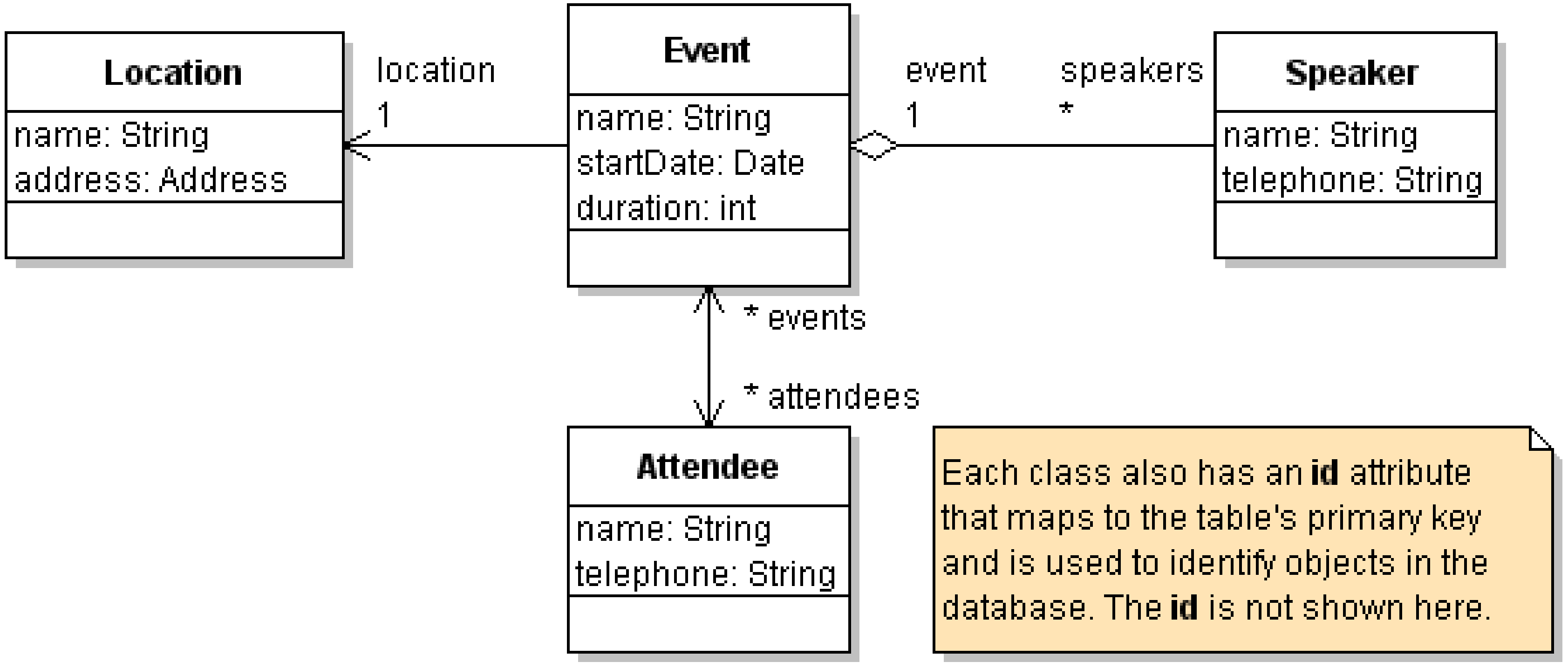
- save object as a row in a database table
- create object using data from table
- save and create *associations* between objects

Design Goals

- separate O-R mapping service from our application
- localize the impact of change in database

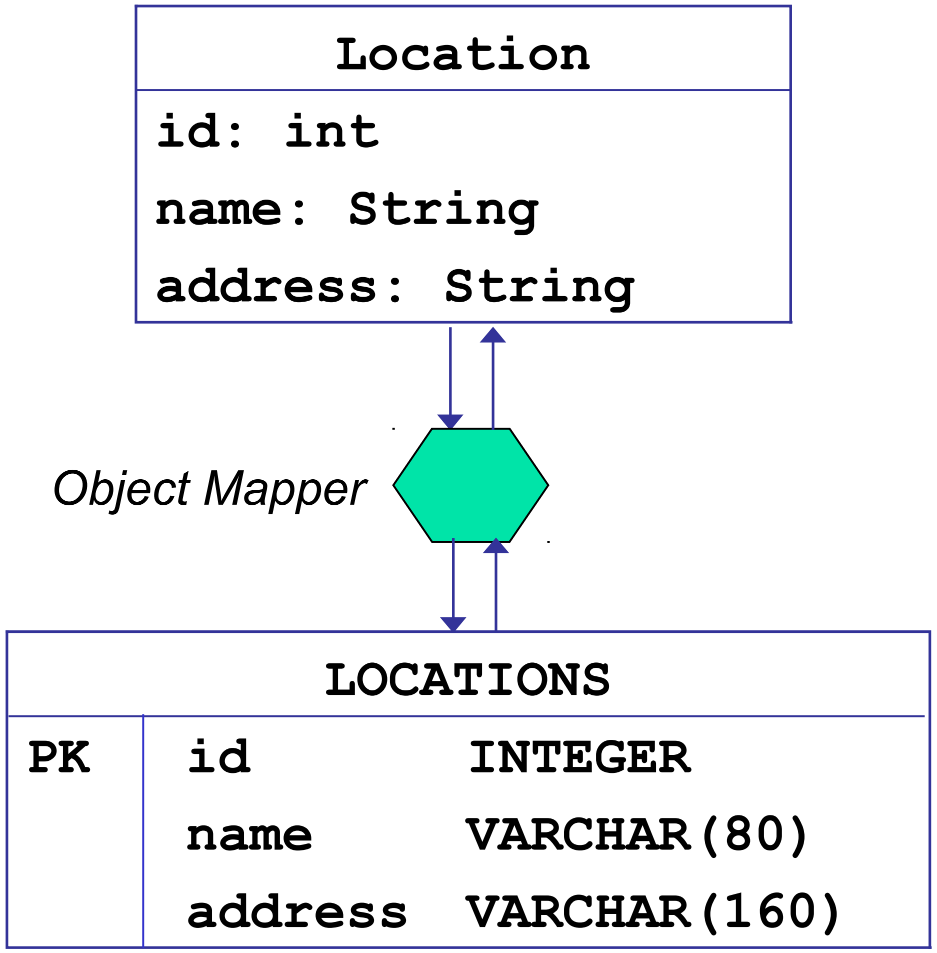
An Example

An Event Manager application with these classes:



Object-Relational Mapping

Map between an object and a row in a database table.



Class

should have an identifier attribute

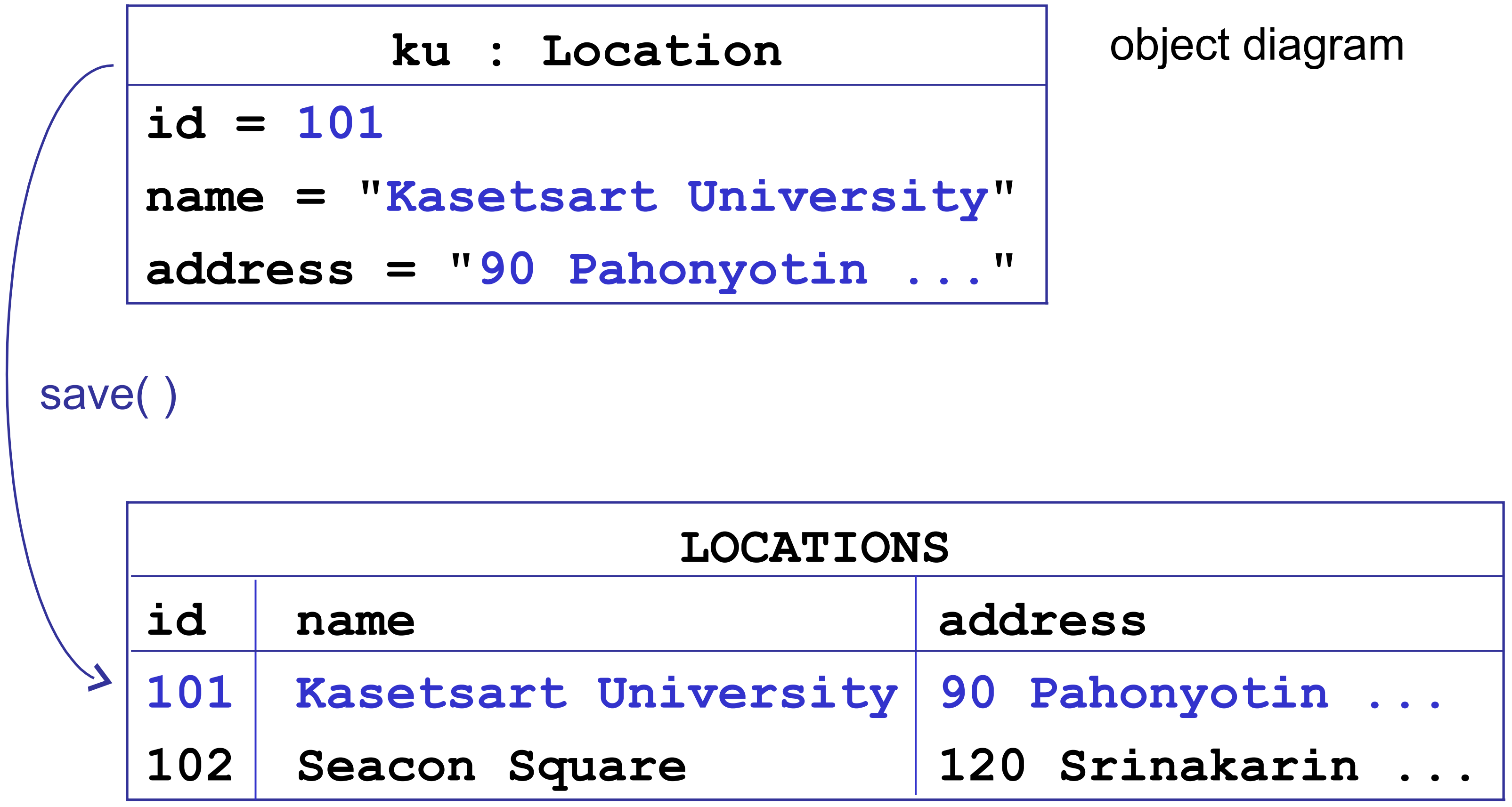
Object Mapper

convert object to table row data, convert data types, instantiates objects

Database Table

identifier is usually the primary key of table

Mapping an Object



```
ku : Location
id = 101
name = "Kasetsart University"
address = "90 Pahonyotin ..."
```

LOCATIONS		
id	name	address
101	Kasetsart University	90 Pahonyotin ...
102	Seacon Square	120 Srinakarin ...

O-R Mapping Code for Location (1)

```
Location ku = new Location( "Kasetsart University" );
ku.setAddress( "90 Pahonyotin Road; Bangkok" );
// save the location
objectMapper.save( ku );
```

Issues:

- mapper should choose a unique ID for saved objects
- what happens if same data is already in the table?

Finding an object

```
// retrieve the location
Location ku1 = objectMapper.find("Kasetsart University");
Location ku2 = objectMapper.find("Kasetsart University");
```

- what field does find( ) search for? id field? name field?
- does mapper always return the same object?

  `( ku1 == ku2 )` => true or false?

Finding an object: Solution

Provide two kinds of "find".

`find( key )` - find object by primary key

`query( string )` - find objects using a flexible query language. May return many matches.

```
// retrieve the location
Location ku1 = objectMapper.find( 111 );
List ku_list = objectMapper.query(
   "'SELECT WHERE name LIKE 'Kasetsart U%'");
```

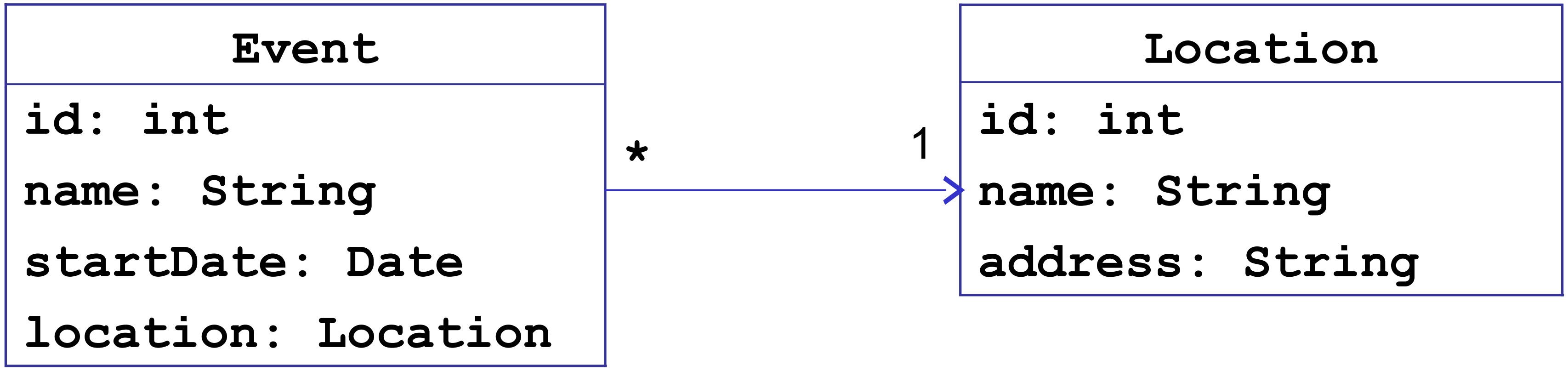
Transparent Persistence

With *transparent persistence*, changes to a "managed" object are automatically propagated to the database.

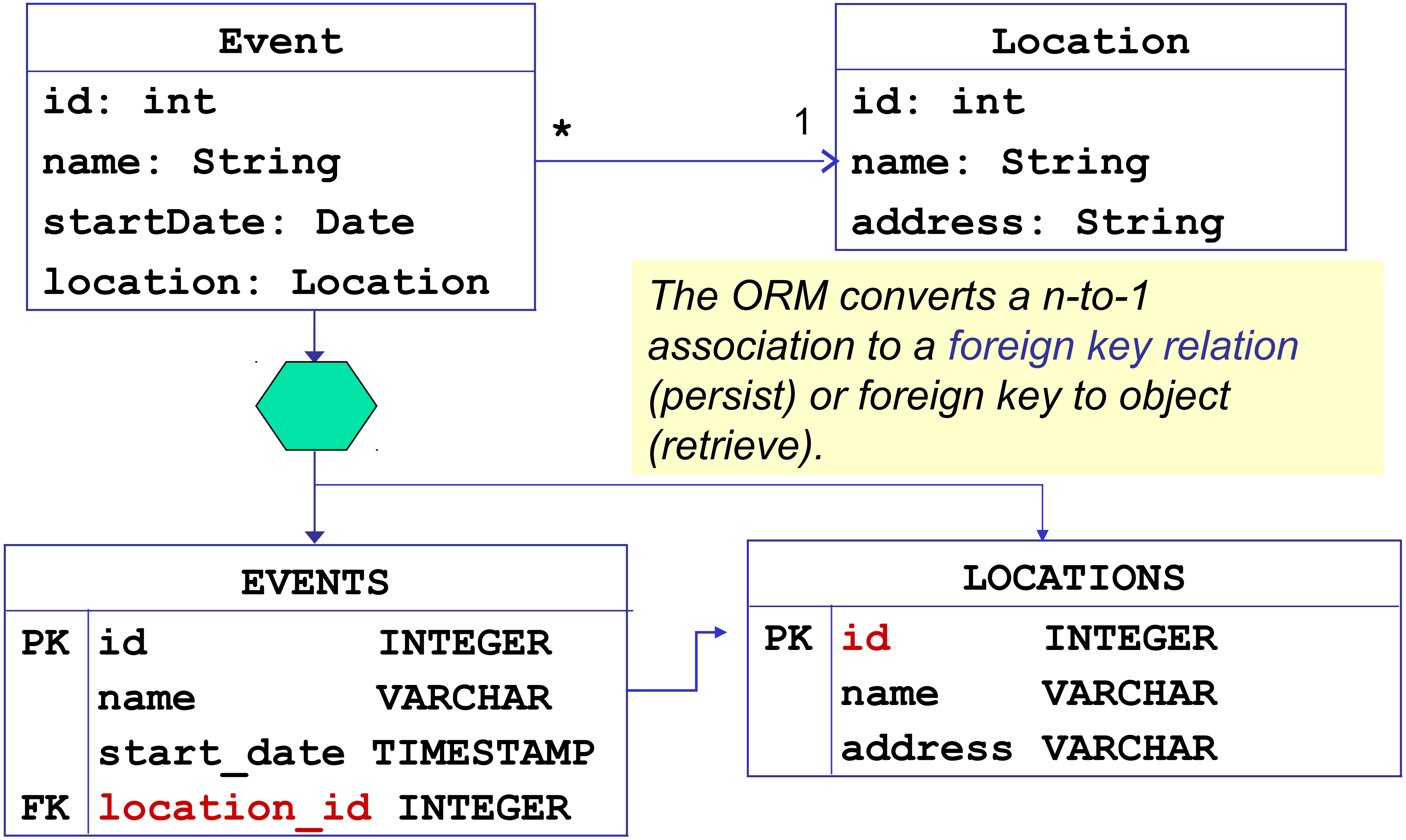
```
Location ku = new Location( "Kasetsart University" );
ku.setAddress( "90 Pahonyotin Road; Bangkok" );
// save the location
objectMapper.save( ku );
// change the address
ku.setAddress( "Kampaengsaen, Nakorn Pathom" );
```

LOCATIONS		
id	name	address
101	Kasetsart University	Kampaengsaen ...
102	Seacon Square	120 Srinakarin ...

O-R Mapping of n-to-1 Associations



O-R Mapping of n-to-1 Associations



The ORM converts a n-to-1 association to a foreign key relation (persist) or foreign key to object (retrieve).

Cascaded Save

Save an Event...

```
Event event = new Event( "Java Days" );
Location ku = new Location( "Kasetsart University" );
ku.setAddress( "90 Pahonyotin Road; Bangkok" );
event.setLocation( ku );
event.setStartDate( new Date(108,Calendar.JULY, 1) );
// save the event
objectMapper.save( event );
```

When we save the event, does it save the location, too?

Is this done automatically?

Deleting an Event

```
// delete the event
Event evt = objectMapper.find( "Java Days" );
objectMapper.delete( evt );
```

Does the dataMapper delete the Location, too?

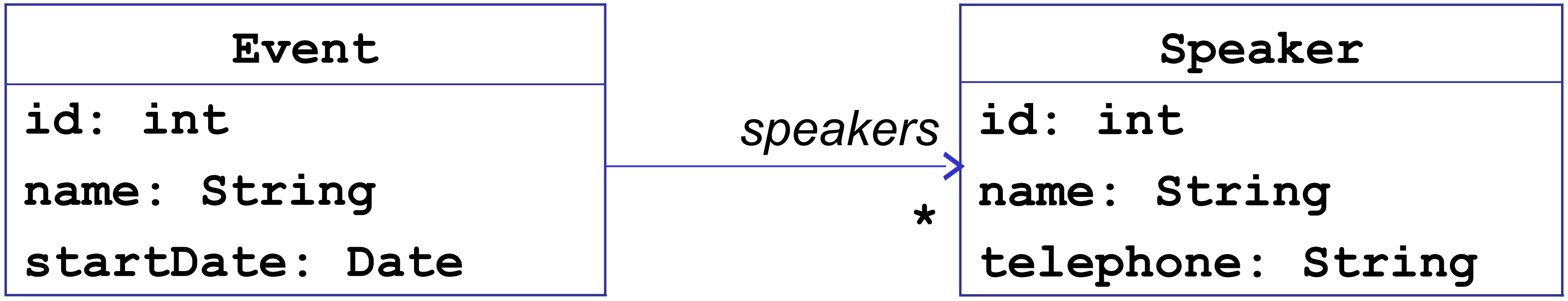
What if other Events (in database) still refer to this Location?

Fetching an Event

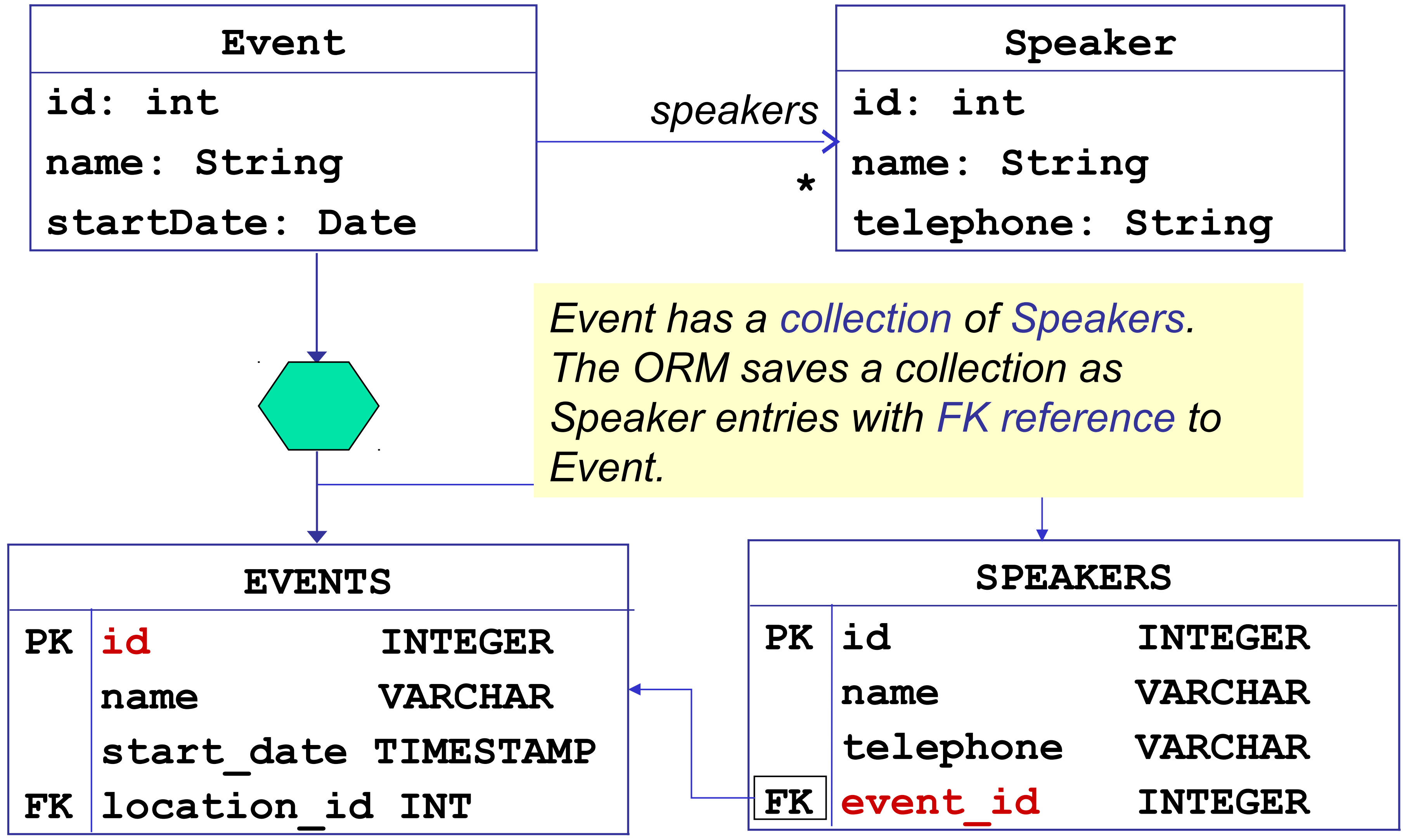
```
// retrieve the event
Event evt = objectMapper.find( "Java Days" );
Location location = evt.getLocation( ); // null?
```

When we get the event, does the ORM fetch the location, too?

O-R Mapping of 1-to-n Associations



O-R Mapping of 1-to-n Associations



Event
id: int
name: String
startDate: Date

Speaker
id: int
name: String
telephone: String

Event has a collection of Speakers. The ORM saves a collection as Speaker entries with FK reference to Event.

EVENTS		
PK	id	INTEGER
	name	VARCHAR
	start_date	TIMESTAMP
FK	location_id	INT

SPEAKERS		
PK	id	INTEGER
	name	VARCHAR
	telephone	VARCHAR
FK	event_id	INTEGER

O-R Mapping for Collections (1)

```
Event event = new Event( "Java Days" );
event.setLocation( ku );
// add event speakers
Speaker gosling = new Speaker( "James Gosling" );
Speaker yuen = new Speaker( "Prof. Yuen" );
event.getSpeakers().add( gosling );
event.getSpeakers().add( yuen );
// save the event
objectMapper.save( event );
```

Issues:

- same issues as many-to-1 association

How to Map Collections?

```
// retrieve the event
Event evt = objectMapper.find("Java Days");
Collection speakers = evt.getSpeakers( );
```

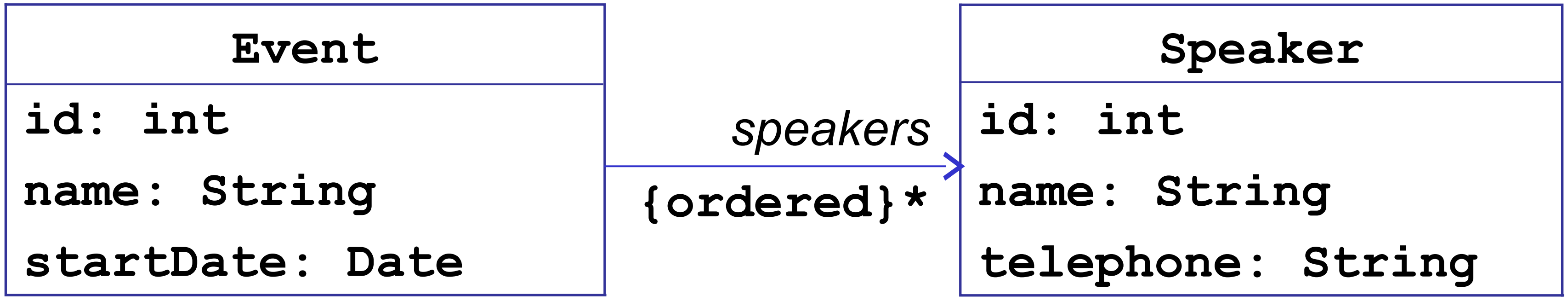
What kind of collection does ORM return?

Can we use *any* collection we want?

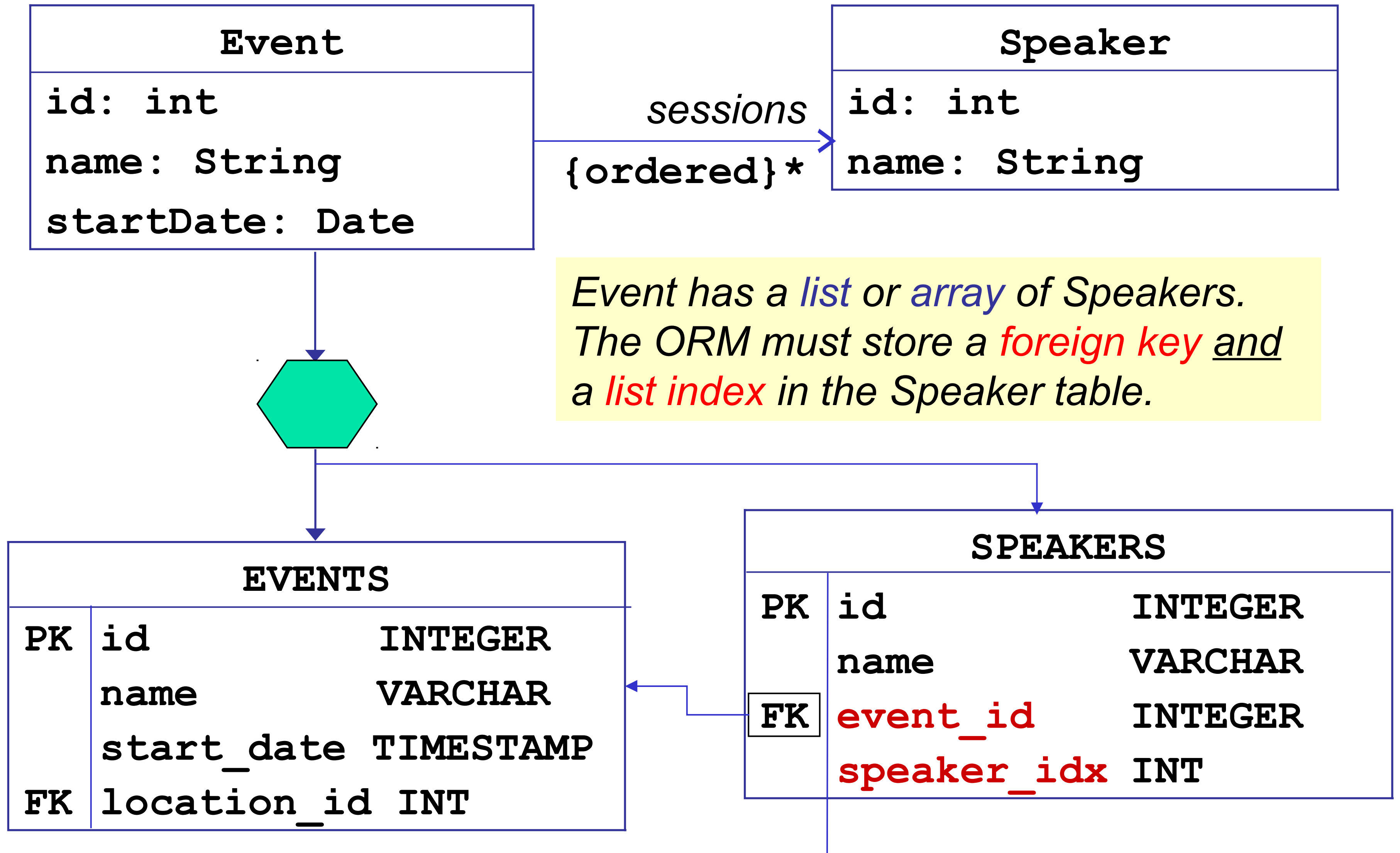
List?

ArrayList?

O-R Mapping of Ordered Collections

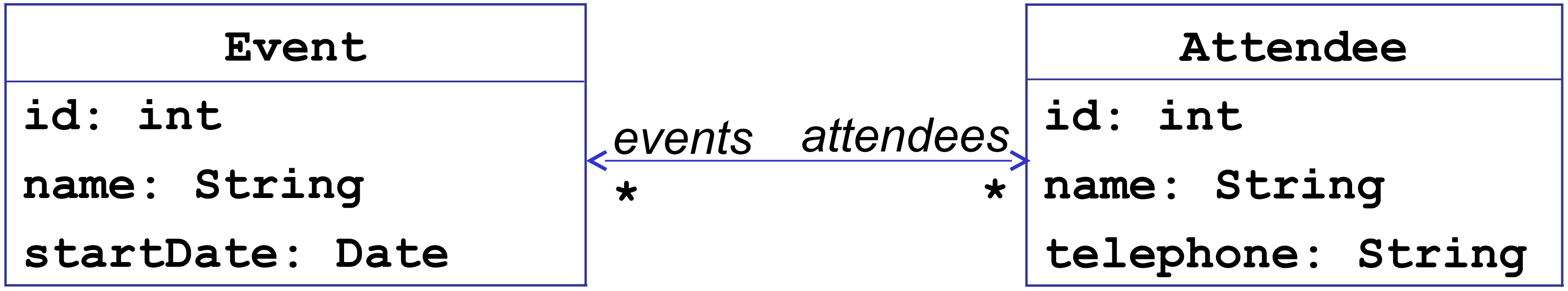


O-R Mapping of Ordered Collections

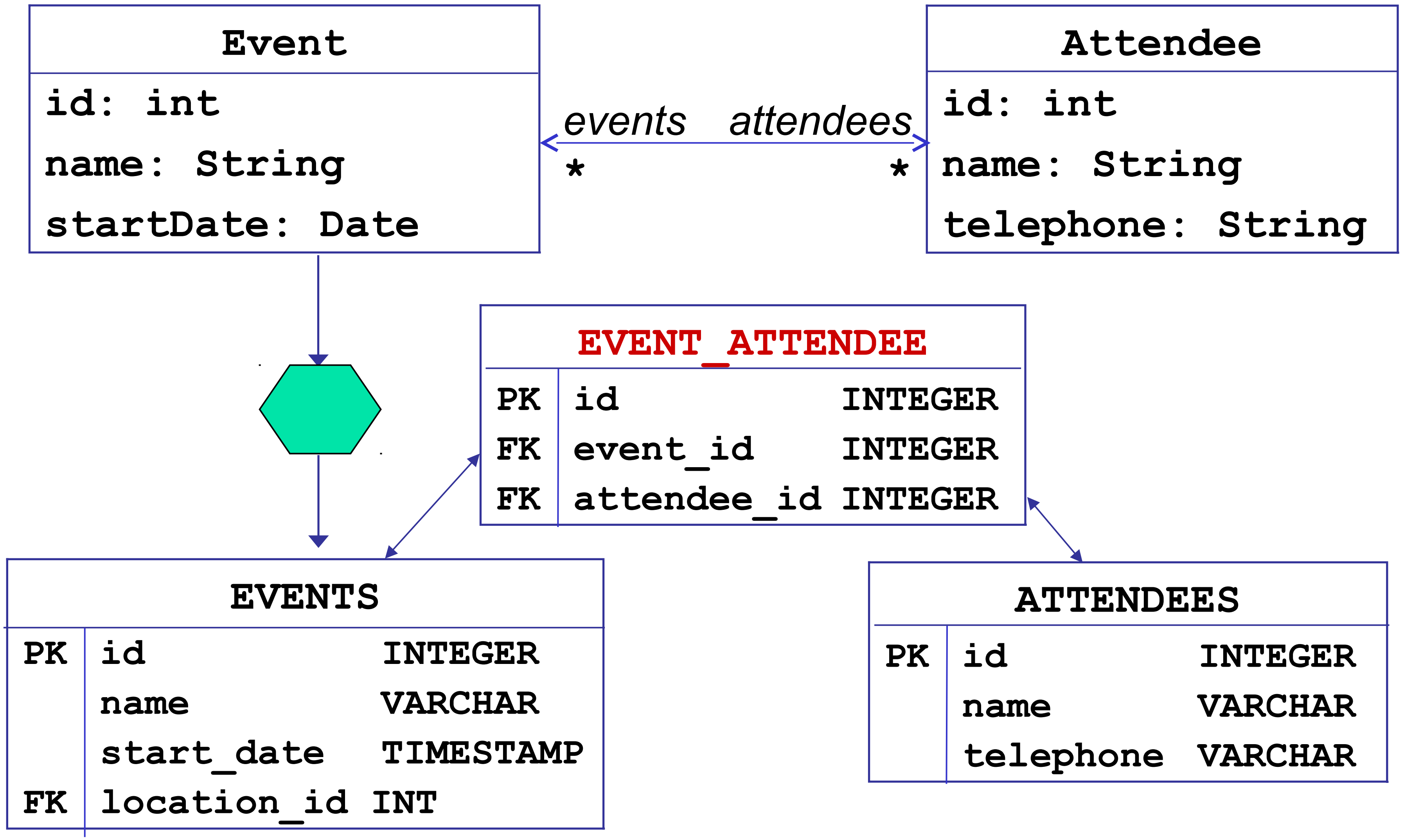


Event has a list or array of Speakers.
The ORM must store a foreign key and a list index in the Speaker table.

O-R Mapping of m-to-n Associations



O-R Mapping of m-to-n Associations



Event
id: int
name: String
startDate: Date

Attendee
id: int
name: String
telephone: String

EVENT_ATTENDEE

PK	id	INTEGER
FK	event_id	INTEGER
FK	attendee_id	INTEGER

EVENTS

PK	id	INTEGER
	name	VARCHAR
	start_date	TIMESTAMP
FK	location_id	INT

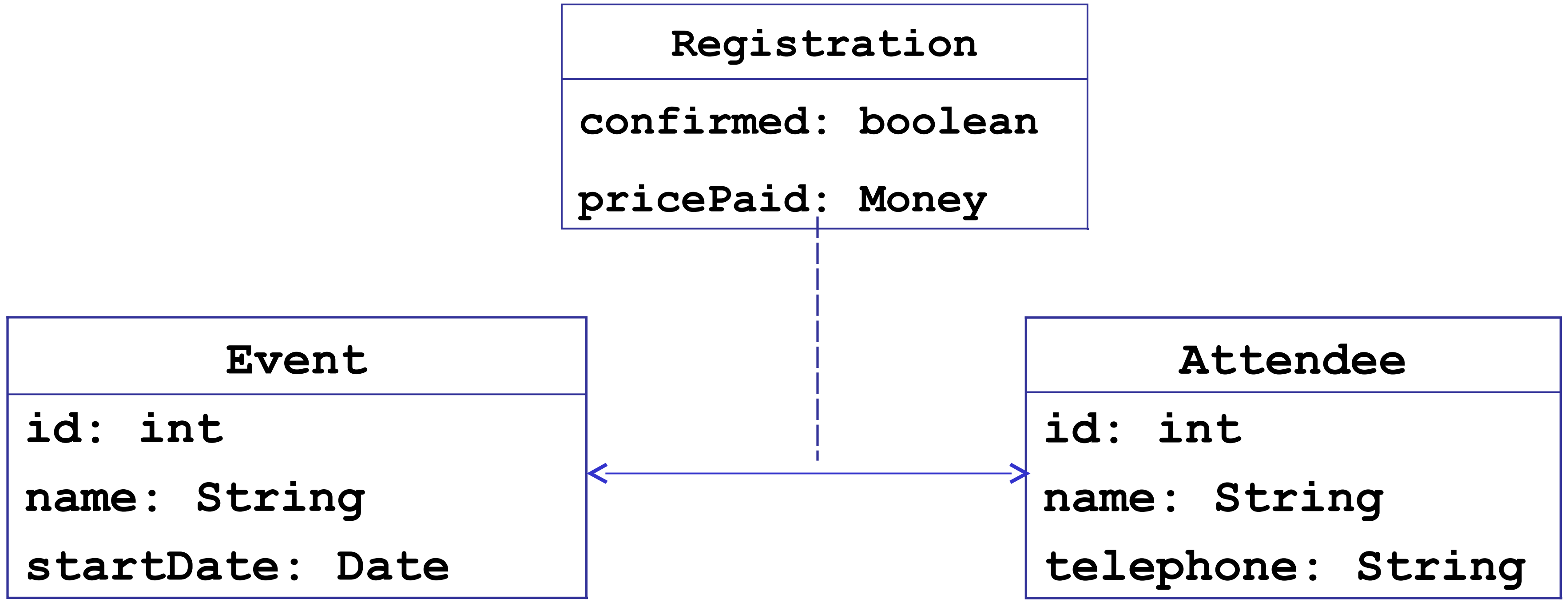
ATTENDEES

PK	id	INTEGER
	name	VARCHAR
	telephone	VARCHAR

Association Class as part of Model

Sometimes the *association* has modeling significance.

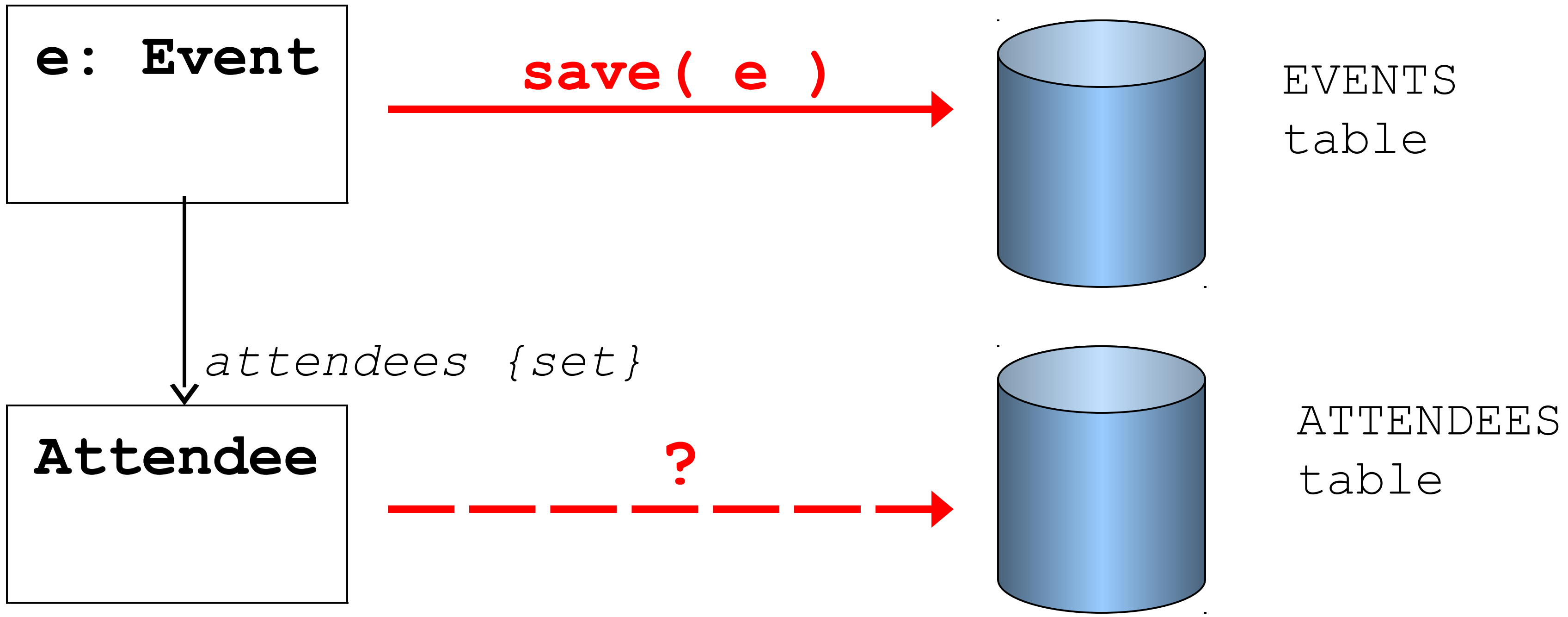
An *Attendee* has a collection of *Registrations*.



What is *Cascading*?

When you save/update/delete an object in database...

are associated objects also saved/updated/deleted?



Frameworks Provide Cascading

In JPA, using annotations:

```
NONE
PERSIST
REFRESH
REMOVE
ALL
```

```
@Entity
class Event {

@OneToMany(mappedBy="event", cascade=PERSIST)
    private List<Person> attendees;
```

Cascading in Hibernate

In Hibernate mapping file for Event:

```
<class name="Event" table="EVENTS" lazy="false">
   <id name="id" column="ID"> </id>
   <property name="name" column="Name"/>
   <set name="attendees" cascade="save-update">
      <key column="event_id"/>
      <one-to-many class="Person"/>
   </set>
```

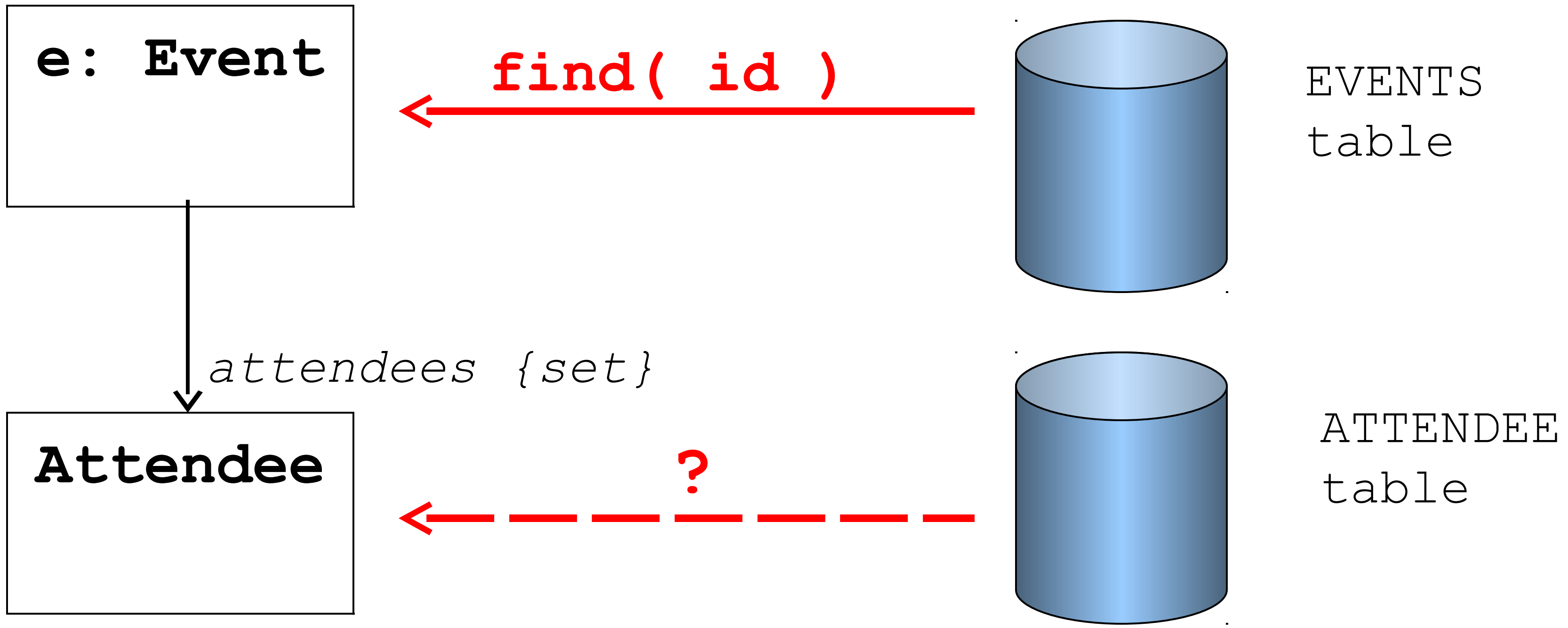
cascade=

"none"	don't cascade operations
"all"	cascade all operations (be careful)
"save-update"	cascade save and updates
"delete-orphan"	cascade all, delete unreferenced orphan children

What are *Eager* and *Lazy Fetching*?

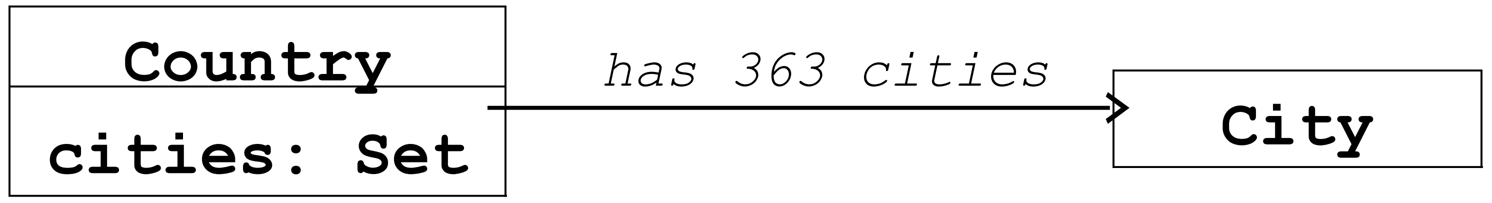
When you create an object from database...

when are associated objects created?



Why is fetching Important?

Example: get a Country from Country database.



```
Country china = orm.query(
   "SELECT c FROM Country c WHERE c.name='China'");
System.out.println(
   "Population is "+china.getPopulation() );
```

How many objects are created?

a) One - just the Country object

b) 364 - Country + all 363 cities

What are Eager and Lazy Fetching?

Eager: create all associated object immediately.

Lazy: create associated objects only when they are referenced.

```
Country china =
         orm.query("SELECT c FROM ...");   <-- EAGER

System.out.println(
  "Population is "+china.getPopulation() );

for(City c: china.getCities() )             <-- LAZY
   Sytem.out.println("has city: "+city);
```

Problem with Lazy Fetching

The query or connection object might be *closed before* the code accesses the cities.

```
// This code uses JPA
em = entityManagerFactory.getEntityManager();
Query q = em.createQuery("SELECT c FROM ...");
Country china = q.getSingleResult();
// close entity manager to free resources
em.close( );

for(City c: china.getCities() )
   Sytem.out.println("has city: "+city);
```

ERROR: not attached to database

Object-Relational Operations: CRUD

Common O-R operations are:

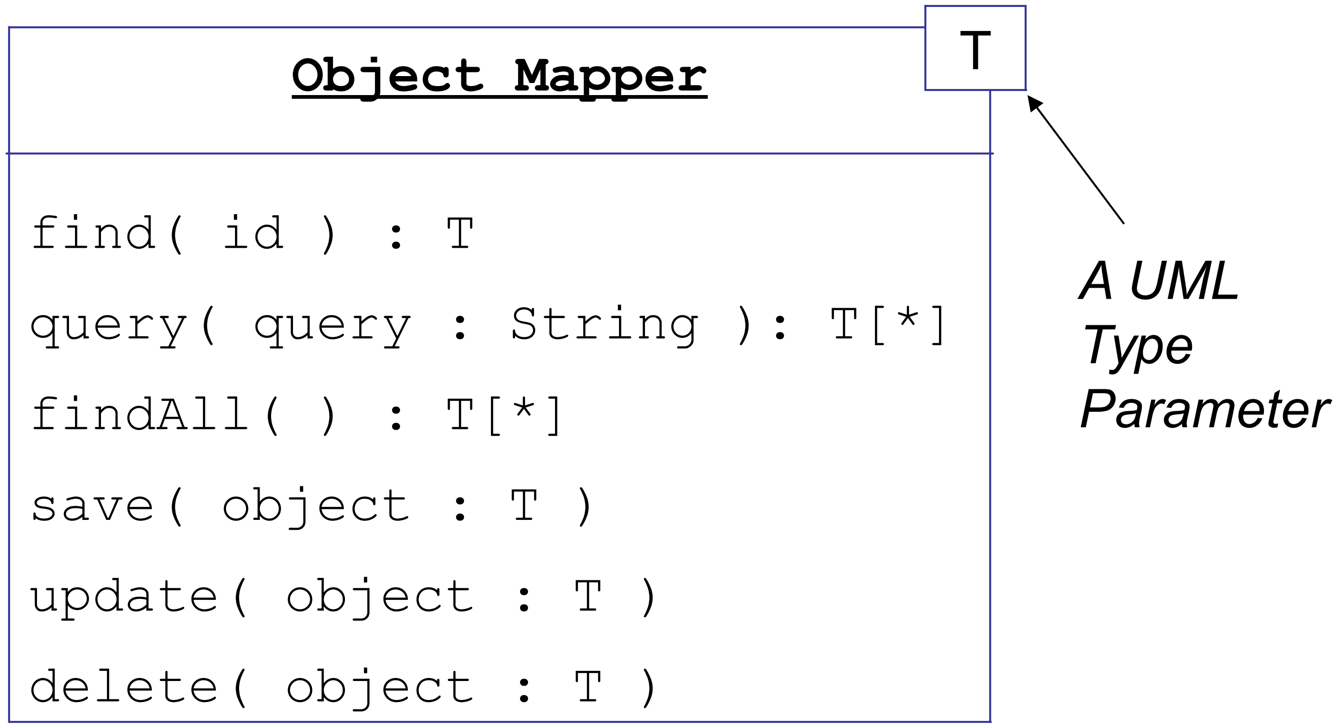
Create - save (persist) a new object in the database

Retrieve an object from the database

Update data for an object already saved in database

Delete object data from the database

Design Model for Object Mapper



The method to "find" an Object by its identifier maybe named:

load(id) the Hibernate and Spring name

find(id, Class) JPA

get(id) similar to load but no exception if id is not found

Object Mapping for Event Class

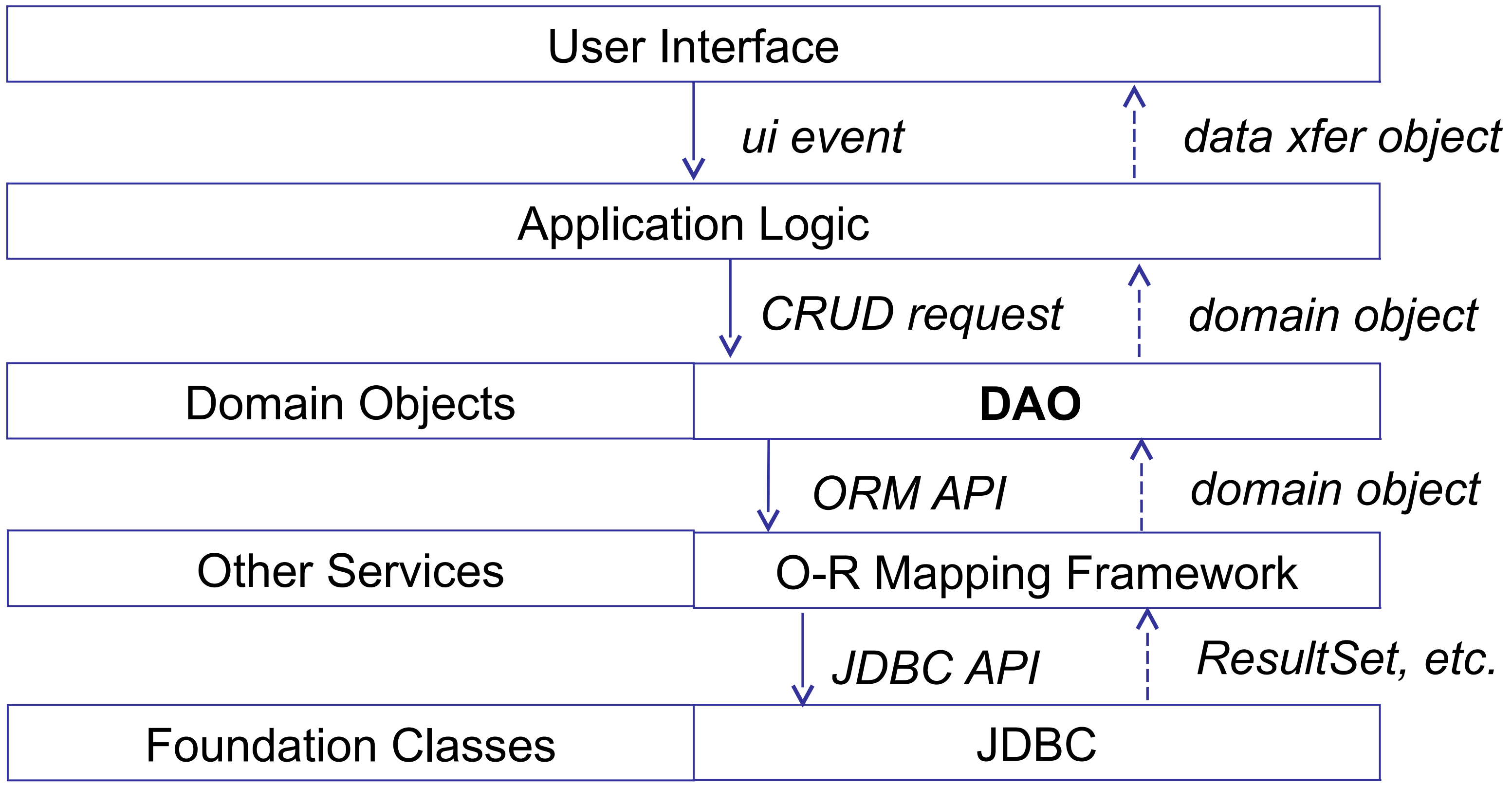
This class is generally called a

Data Access Object (DAO).

- Hibernate uses the term "data access object".
- Append "Dao" to the class name, e.g. **`EventDao`**.

EventDao
`find( id: int ) : Event`
`query( query: String ) : Event[*]`
`save( evt: Event )`
`update( evt: Event )`
`delete( evt: Event )`

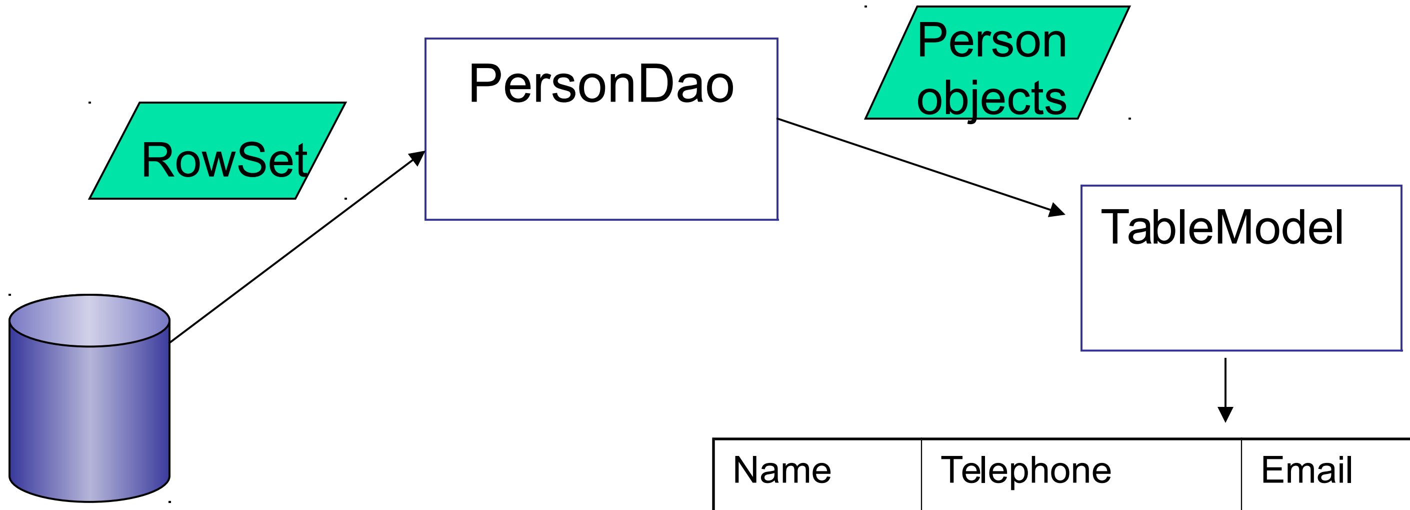
Layered Design



When *Not* to Use O-R Mapping

In some applications, Object-Relational mapping is inefficient.

Example: display a table of attendees



Name	Telephone	Email
Bill Gates	1-215-555-1212	gates@msn.com
B. Obama	1-212-111-1212	president@whitehouse

4 Approaches to ORM

1. No ORM -- JDBC in my code.

   No Layers! Put the JDBC right in your app code.

2. Do It Myself.

   Write your own DAO using JDBC.

3. Use a Framework.

   Hibernate, MyBatis, TopLink, or other.

4. Use a Standard.

   Java Persistence Architecture (JPA) or Java Data Objects (JDO) provide a *standard API* that have *many implementations*.

What's Next?

If you want to...	*Study path:*
do It yourself	❑ SQL Fundamentals ❑ JDBC Fundamentals ❑ Design and Code
use a framework	❑ How to use Hibernate ❑ Configure a Database
use a stardard	❑ How to use JPA ❑ Configure a Database

Persistence Frameworks

Hibernate - most popular open-source persistence framework for Java. NHibernate for .Net.

Uses POJOs and object-query language. Completely decouple Java from database. Can reverse engineer.

MyBatis - simple, uses SQL maps. Database schema not transparent to Java code.

Cayenne - Apache project, has GUI modeler that eliminates need to write xml. Can reverse engineer database or generate database schema & Java code.

TopLink (Oracle)

Torque (Apache DB)

Castor, ...

Persistence Standards

Java Persistence API (JPA)

standard for persistence of plain java objects. Can be used with stand-alone or enterprise apps. Good IDE support.

- EclipseLink, TopLink Essentials (Glassfish project), OpenJPA. DataNucleus, Hibernate Annotations.

Java Data Objects (JDO)

transparent persistence of POJOs; can persist to LDAP, RDBMS, Excel, and other

- Kodo, DataNucleus

Reference for Frameworks

Article: *Adopting a Java Persistence Framework,* *http://today.java.net/pub/a/today/2007/12/18/adopting-java-persistence-framework.html*

No Persistence Framework

Web4J (www.web4j.org)

- web + database in Java *without* O-R mapping. Interesting & educational web site
- Presents arguments why *not* to use a framework (but doesn't mention Hibernate).