The EJB-QL Query Language

UML based modeling and analysis

Dániel Varró

Why do we need a new QL?

Why do we need a new QL?

- Standard SQL is not so standard
  - Different vendors → different implementation
  - Queries becomes vendor-specific

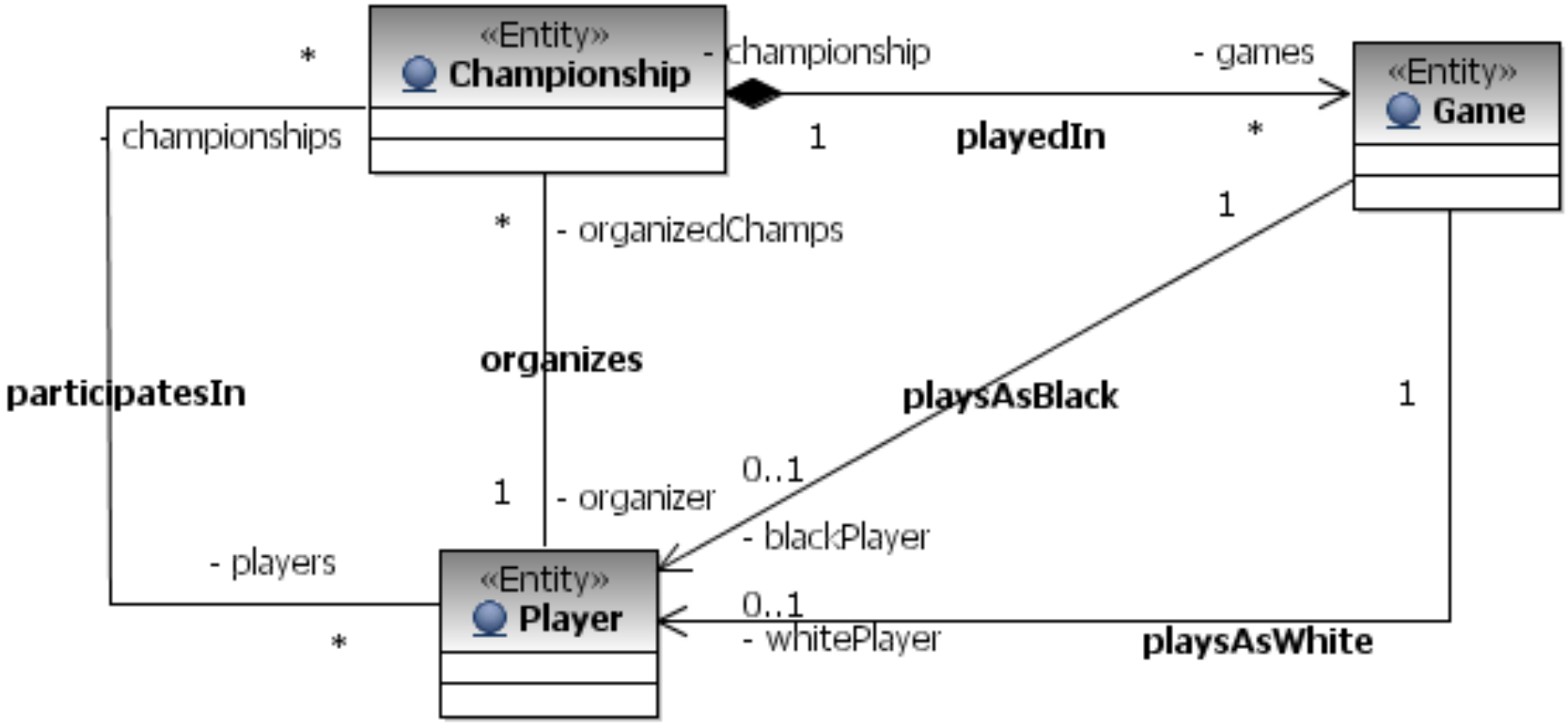
Why do we need a new QL?

- Standard SQL is not so standard
  - Different vendors → different implementation
  - Queries becomes vendor-specific
- Different paradigms (OO vs. Rel)
  - OO2Rel mapping: hide Relations layer
  - Queries should be defined over OO and not over Rel

Definition of EJB-QL

- EJB-QL:
  - an object-oriented query language
  - for querying entities
  - polymorphic (!)
- Contents:
  - SELECT clause
  - FROM clause: domain of the query
  - WHERE clause (optional)
- Declares queries for a find method
- Usage:
  - Written in the deployment descriptor
  - Container generates DB logic (SQL)

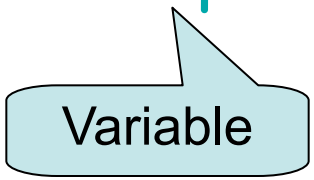
Entity Classes in Championship Management



Simple EJB-QL queries

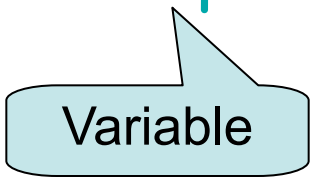
Simple EJB-QL queries

- findAllChampionships()
  - SELECT o FROM Championship o



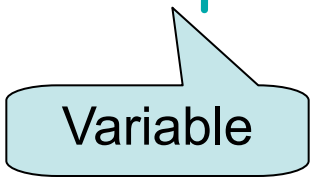
Simple EJB-QL queries

- findAllChampionships()
  - SELECT o FROM Championship o



Simple EJB-QL queries

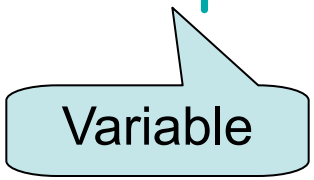
- findAllChampionships()
  - SELECT o FROM Championship o



- findYoungPlayers()
  - SELECT p FROM Player p
    WHERE p.age<30

Simple EJB-QL queries

- findAllChampionships()
  - SELECT o FROM Championship o



- findYoungPlayers()
  - SELECT p FROM Player p
    WHERE p.age<30
- findOrganizerNames()
  - SELECT DISTINCT c.organizer.name
    FROM Championship c

Simple EJB-QL queries

- findAllChampionships()
  - SELECT o FROM Championship o

Selected fields (object this time)

Variable

- findYoungPlayers()
  - SELECT p FROM Player p WHERE p.age<30
- findOrganizerNames()
  - SELECT DISTINCT c.organizer.name FROM Championship c

Distinct names of organizers are selected

Navigation

Navigation

- Navigation is possible
  - along relationships and attributes
  - if the relationship end is navigable, and
  - its multiplicity is at most one

Navigation

- Navigation is possible
  - along relationships and attributes
  - if the relationship end is navigable, and
  - its multiplicity is at most one
- Example: findOrganizers()
  - select c.organizer from Championship c

Navigation

- Navigation is possible
  - along relationships and attributes
  - if the relationship end is navigable, and
  - its multiplicity is at most one
- Example: findOrganizers()
  - select c.organizer from Championship c
- Incorrect is:
  - select c.participants from Championship c

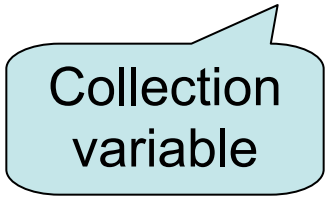
SELECT clause may return single variables and not collections

Collection variables

- Rule:
  Variables represent one value at a time
- IN() keyword (or JOIN)
- Left-to-right evaluation order
  - Declared variables can be used later
- Example:
  findParticipants()
  - SELECT p FROM Championship c, IN (c.players) p
    WHERE c.name = 'Champions League'

Collection variables

- Rule:
  Variables represent one value at a time
- IN() keyword (or JOIN)
- Left-to-right evaluation order
  - Declared variables can be used later
- Example:
  findParticipants()
  - SELECT p FROM Championship c, IN (c.players) p
    WHERE c.name = 'Champions League'



Queries with Relationships

Queries with Relationships

- findPlayersWithChamps() – Version 1
  - SELECT DISTINCT p
    FROM Player p, IN(p.championships) c

Queries with Relationships

- findPlayersWithChamps() – Version 1
  - SELECT DISTINCT p
    FROM Player p, IN(p.championships) c

Does not include players without a championship

Queries with Relationships

- findPlayersWithChamps() – Version 1
  - SELECT DISTINCT p
    FROM Player p, IN(p.championships)
- findPlayersWithChamps() – Version 2
  - SELECT p FROM Player p
    WHERE p.championships IS NOT EMPTY

Does not include players without a championship

Queries with Relationships

- findPlayersWithChamps() – Version 1
  - SELECT DISTINCT p
    FROM Player p, IN(p.championships) c
- findPlayersWithChamps() – Version 2
  - SELECT p FROM Player p
    WHERE p.championships IS NOT EMPTY
- findPlayersWithoutChamps()
  - SELECT p FROM Player p
    WHERE p.championships IS EMPTY

Does not include players without a championship

Queries with Relationships

- findPlayersWithChamps() – Version 1
  - SELECT DISTINCT p
    FROM Player p, IN(p.championships) c
- findPlayersWithChamps() – Version 2
  - SELECT p FROM Player p
    WHERE p.championships IS NOT EMPTY
- findPlayersWithoutChamps()
  - SELECT p FROM Player p
    WHERE p.championships IS EMPTY
- findOrganizingParticipants()
  - SELECT p FROM Player p JOIN (p.championships) c
    WHERE c.organizer=p
  - SELECT p FROM Player p JOIN Championship c

Does not include players without a championship

Queries with Relationships

- findPlayersWithChamps() – Version 1
  - SELECT DISTINCT p
    FROM Player p, IN(p.championships) c
- findPlayersWithChamps() – Version 2
  - SELECT p FROM Player p
    WHERE p.championships IS NOT EMPTY
- findPlayersWithoutChamps()
  - SELECT p FROM Player p
    WHERE p.championships IS EMPTY
- findOrganizingParticipants()
  - SELECT p FROM Player p JOIN (p.championships) c
    WHERE c.organizer=p
  - SELECT p FROM Player p JOI

Does not include players without a championship

LEFT JOIN operation also exists

Subqueries in EJB-QL

- Subqueries
  - SELECT o FROM Championship o
    WHERE o.id =
    (SELECT MAX (c.id) FROM
    Championship c)
- ALL, ANY/SOME
  - ALL: true if a comparison is true for all values in the result of the subquery (or the result of the subquery is empty)
  - ANY: true if the comparison is true for some value in the result of the subquery
  - SELECT c FROM Championship c
    WHERE c.organizer.age < ALL (SELECT p.age FROM

Query parameters

- Can only be used in WHERE (and HAVING) clauses
  - Positional parameter: ?1
  - Named parameter: :param
- Find all championships of an organizer findAllChamps(Integer id1)
  - SELECT c FROM Championship c WHERE c.id = ?1

Calling Queries

- See the UML wiki:
- Named Queries
- Dynamic queries

```
public List findWithName(String name) {
return em.createQuery(
"SELECT c FROM Championship c WHERE c.name
   LIKE :custName")
.setParameter("custName", name)
.setMaxResults(10)
.getResultList();
}
```

Additional Constructs

- GROUP BY
  - enables the aggregation of values according to a set of properties
- HAVING:
  - search conditions over the grouping items or
  - aggregate functions that apply to grouping items
- ORDER BY
  - allows the objects or values that are returned by the query to be ordered

Database Manipulation

- Bulk Update and Delete
  - Applied to a collection of objects in a single step

```
DELETE FROM Championship c
  WHERE c.organizer MEMBER OF c.players
```

- Caution: May introduce inconsistency between the database and the entity
  - Should be performed as a separate transaction, OR
  - At the beginning of transactions

Built-in functions

- Similar use as in SQL
  - CONCAT(Str1, Str2)
  - SUBSTRING(Str, start, length)
  - LENGTH(Str)
  - MAX, ABS, SQRT
  - DISTINCT