






Mapping Inheritance

@MappedSuperclass

s_soccer		
	ID	bigint(20)
	DATE_PLAYED	datetime
	AWAY_TEAM_ID	gint(20)
	HOME_TEAM_ID	gint(20)
	TEAM_ID	bigint(20)

s_multiplayergame		
	ID	bigint(20)
	DATE_PLAYED	time
	PLAYER_ID	bigint(20)

@MappedSuperclass






- will you ever need to query across class hierarchies?
- will you ever use a base class as a target of the query?

If either question is yes, @MappedSuperClass strategy is not a good fit

Inheritance

- @Inheritance
 - TABLE_PER_CLASS
 - JOINED
 - SINGLE_TABLE

TABLE_PER_CLASS

s_soccer	
 ID	bigint(20)
 DATE_PLAYED	datetime
 AWAY_TEAM_ID	gint(20)
 HOME_TEAM_ID	gint(20)
 TEAM_ID	bigint(20)

s_multiplayergame	
 ID	bigint(20)
 DATE_PLAYED	time
 PLAYER_ID	bigint(20)

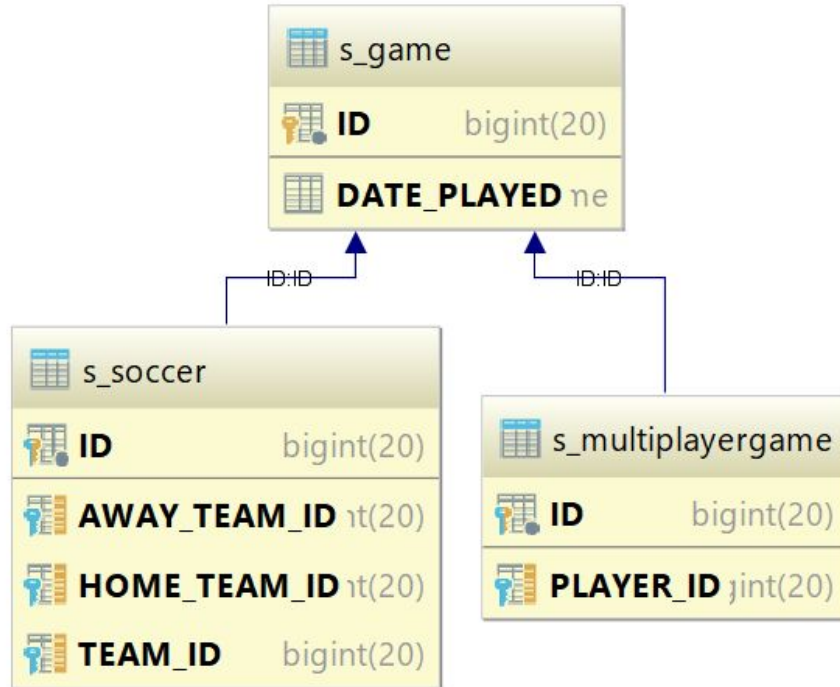
- Poor performance on the polymorphic queries

Single Table Inheritance Strategy

s_game	
ID	bigint(20)
DTYPE	varchar(31)
DATE_PLAYED	datetime
PLAYER_ID	bigint(20)
AWAY_TEAM_ID	bigint(20)
HOME_TEAM_ID	bigint(20)
TEAM_ID	bigint(20)

- attributes of all entities are mapped to the same database table
- can't use *not null* constraints on any column that isn't mapped to all entities

Joined Inheritance Strategy



Choosing a Strategy

- single table - best performance with polymorphic queries
- joined table - preserved data consistency and polymorphic queries allowed
- table per class - if you don't need polymorphic queries