Сводная таблица решений N+1 проблемы и кейсов по ней на OneToMany

ПРОБЛЕМА

КОРОТКО:
Проблема более не актуальна, она решена самим Hibernate (выше 6 версии =) шутка

ПОДРОБНЕЕ:

Вместо того, чтобы получить ОДИН запрос, получаем в нагрузку N количество.
Одним запросом достали всех пользователей и потом на каждого инициализируется какая-либо связь сущностей.

Проблема N+1 связана с производительностью. Если делаем множественный запрос (пользователь, платёж, компания), то Hibernate делает дополнительный SELECT запрос.

РЕШЕНИЕ

	UNIdirectional **OneTo**Many		**UNI**directional **Many**ToOne		**BI**directional **Many**ToOne, OneTo**Many**	
solution	JPQL	native	JPQL	native	JPQL	native
join fetch	+	—	+	—	+	—
FetchMode SUBSELECT	+	—	—	—	+ / — **	—
BatchSize	+	+	—	—	+ / — **	+ / — **
Entity Graph	+	—	+	—	+	—
SqlResultSetMapping	—	—	—	+	—	— / + ***
HibernateSpecificMapping	—	— *	—	+	—	+

* если не используем аннотацию @JoinColumn и оставляем связанную третью таблицу
** работает только при выборке владельца связи с коллекцией зависимых сущностей
*** работает только при выборке зависимой сущности со ссылкой на Entity владельца связи

ВЫВОДЫ:

1. Лучшим вариантом решения N+1 проблемы для простых запросов (1-3 уровня вложенности связанных объектов) будет **join fetch** и **JPQL** запрос. Следует придерживаться тактики, когда выбираем из JPQL и нативного запроса JPQL

2. Если имеется нативный запрос и мы не заботимся о слабой связанности кода, то хорошим вариантом будет использование **Hibernate Specific Mapping.** В противном случае стоит использовать **@SqlResultSetMapping**

3. В случаях, когда нужно получить по-настоящему много данных и используется JPQL запрос, лучше всего использовать **Entity Graph**

4. Если мы знаем примерное кол-во коллекций, которые будут использоваться в любом месте приложения, то можно использовать **@BatchSize**

Версия Матвиенко, DMDEV: Сначала бизнес-логика, потом оптимизация!

https://www.youtube.com/watch?v=XH9KMY4jMSQ

избегайте отношение **OneToOne, Bidirectional**
используйте **LAZY** ленивую загрузку везде, где возможно
избегайте аннотаций **@BatchSize, @Fetch**
используйте **Query Fetch** запрос (HQL, Criteria)
предпочитайте **EntityGraph** API (вместо @FetchProfile)

Коррекный ответ: лучше всего решает эту проблему написание нативных SQL запросов