Ús dels Entity Graphs de JPA com a mecanisme per reduir el nombre d'accessos a una base de dades

Publicat per Victor Manuel Cerdeira López [1] el 11/05/2020 - 23:42 | Última modificació: 12/05/2020 - 10:06

1. Introducció

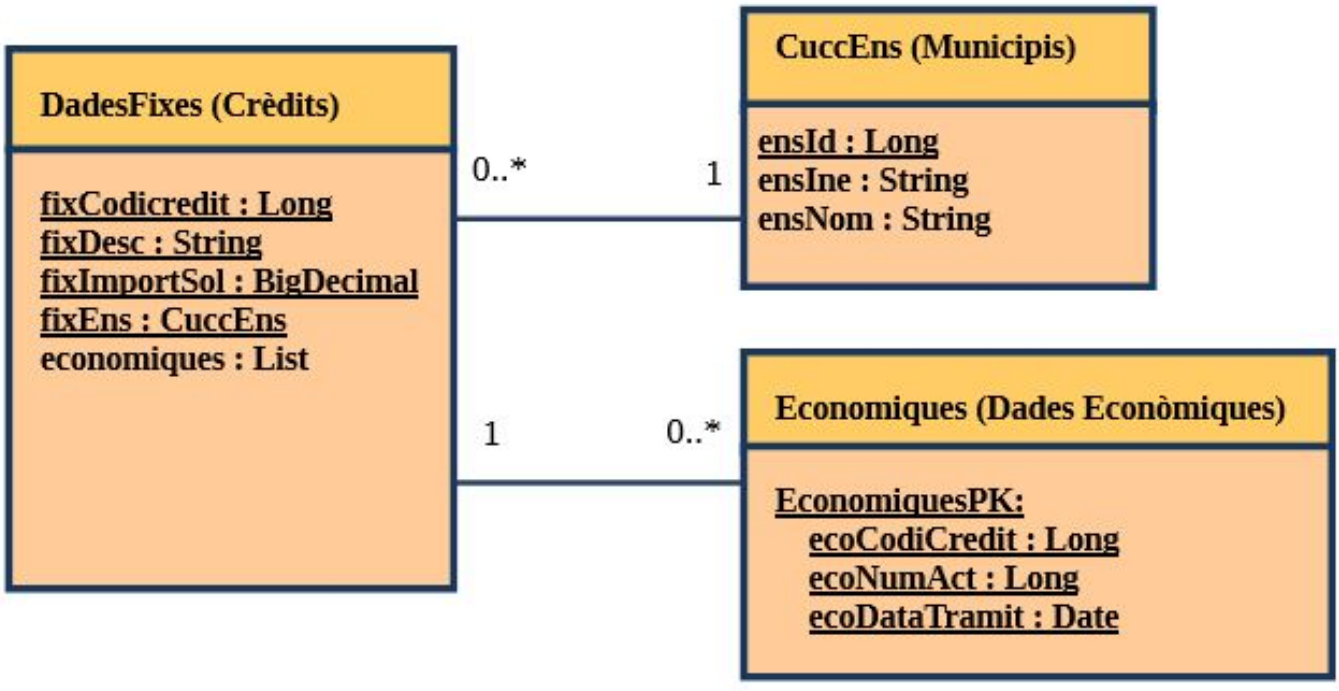
En aquest article descriurem l'ús del mecanisme tècnic anomenat *Entity Graph* (present des de la versió JPA 2.1) amb l'objectiu d'evitar el problema dels *accessos N+1*, el qual és un dels inconvenients més coneguts i recurrents quant a la utilització de JPA per accedir a una base de dades relacional.

Aquest problema consisteix en què, per obtenir una llista d'ocurrències d'una determinada entitat, a més d'executar la consulta que retorna aquesta llista de resultats (N), si l'entitat en qüestió referència una altra, s'executen N consultes addicionals per obtenir les dades de l'entitat referenciada.

Un Entity Graph consisteix en una especificació del subconjunt del model de dades -a nivell d'entitats i atributs necessaris- perquè un determinat servei pugui resoldre la funcionalitat desitjada.

2. Exemple de model de dades

Per il·lustrar l'ús de la tècnica descrita, ens basarem en el següent fragment del model de model de dades utilitzat per l'aplicació PSF (*Programa de Suport al Finançament)*:



Com es pot veure, en aquest model de dades apareix un exemple de relació 1-N i un altre de relació N-1.

La definició de les entitats implicades és:

- public class DadesFixes {

  @Id

  @Column(name = "FIX_CODICREDIT")

  private Long fixCodiCredit;

  @ManyToOne

  @JoinColumn(name = "FIX_CODIENS", referencedColumnName = "ENS_INE")

  private CuccEns fixEns;

  @OneToMany(mappedBy="dadesFixes", fetch=FetchType.LAZY)

  private List<Economiques> economiques;

  ...

}

- public class CuccEns {

  @Id

  @Column(name = "ENS_ID")

  private Long ensId;

  @Column(name = "ENS_INE")

  private String ensIne;

  @Column(name = "ENS_NOM")

  private String ensNom;

```
    ...
}
```

- public class Economiques {

```
    @EmbeddedId
    private EconomiquesPK economiquesPK;

    @ManyToOne(fetch=FetchType.LAZY)
    // Nota: És important declarar com a LAZY la relació inversa d'una relació 1-N.
    // En aquest cas, evita la càrrega de l'entitat DadesFixes per a cada
    // registre Economiques obtingut.
    @JoinColumn(name = "ECO_CODICREDIT", insertable = false, updatable = false)
    private DadesFixes dadesFixes;
    ...
}

public class EconomiquesPK
{
    @Column(name = "ECO_CODICREDIT")
    private Long ecoCodiCredit;

    @Column(name = "ECO_NUMACT", insertable = false, updatable = false)
    private Long ecoNumAct;

    @Temporal(TemporalType.DATE)
    @Column(name = "ECO_DATATRAMIT")
    private Date ecoDataTramit;
}
```

3. Identificació del problema

Per tal de detectar l'execució dels '*accessos N+1'*, por ser convenient activar la traça de les consultes SQL generades. Amb aquest objectiu, pot ser útil descomentar el següent fragment del fitxer *persistence.xml*:

```
<property name="eclipselink.logging.level.sql" value="FINE"/>

<property name="eclipselink.logging.timestamp" value="true"/>

<property name="eclipselink.logging.parameters" value="true"/>

<property name="eclipselink.logging.logger" value="DefaultLogger"/>
```

En el cas de l'aplicació PSF, es pot veure que es generen les següents consultes en el moment d'obtenir el llistat de crèdits:

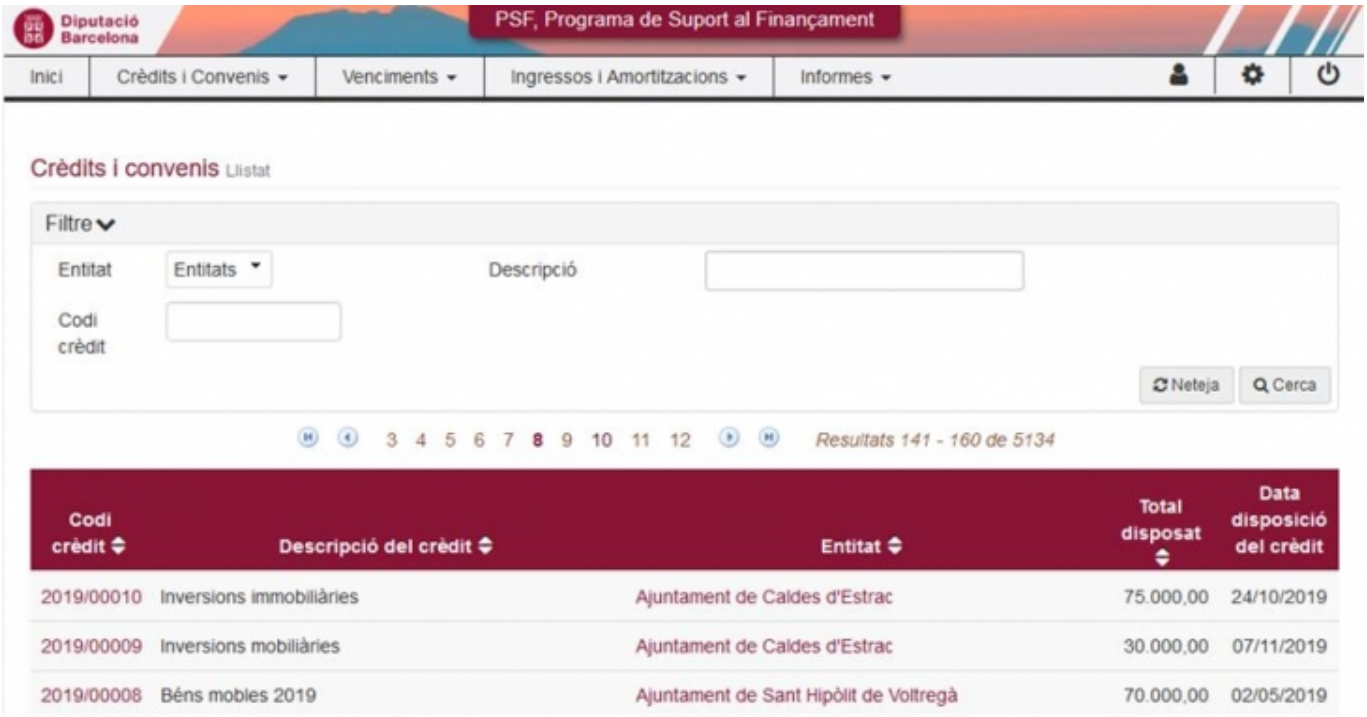
```
// 1 query per obtenir la llista de crèdits:

SELECT FIX_CODICREDIT AS a1, FIX_CODIENS AS a37, .. FROM PSF_DADESFIXES


// N queries per obtenir els municipis de cadascun dels crèdits recuperats:

SELECT ENS_ID, ENS_INE, ENS_NOM FROM CUCC_ENS WHERE (ENS_INE = codi_crèdit_1)

..

SELECT ENS_ID, ENS_INE, ENS_NOM FROM CUCC_ENS WHERE (ENS_INE = codi_crèdit_N)


// N queries per obtenir les dades econòmiques de cadascun dels crèdits recuperats:

SELECT ECO_CODICREDIT, ECO_NUMACT, ... FROM PSF_ECONOMIQUES WHERE (ECO_CODICREDIT = codi_crèdit_1)

...

SELECT ECO_CODICREDIT, ECO_NUMACT, ... FROM PSF_ECONOMIQUES WHERE (ECO_CODICREDIT = codi_crèdit_N)
```

Es pot apreciar que en aquest cas s'estan executant 2N+1 consultes, una per obtenir la llista de crèdits, N per

obtenir cadascun dels municipis referenciats, i N per obtenir les dades econòmiques de cadascun dels crèdits obtinguts.

4. Procediment de definició de l'Entity Graph per reduir el nombre de consultes generades

Per explicar com definir un Entity Graph, utilitzarem com a exemple la consulta que s'executa al polsar el botó 'Cerca' de la pantalla llista de crèdits de l'aplicació PSF: (*dadesFixesLlista.xhtml*)



En aquesta consulta intervenen els següents atributs, els quals s'han identificat a partir de les **columnes** i **filtres** referenciats des d'aquesta pantalla:

Filtre	**Atribut referenciat pel filtre**
Entitat	dadesFixes.fixEns.ensId
Codi crèdit	dadesFixes.fixCodiCredit
Descripció	dadesFixes.fixDesc

Columna	**Atribut referenciat per la columna**
Codi crèdit	dadesFixes.fixCodiCredit
Descripció del crèdit	dadesFixes.fixDesc
Entitat	dadesFixes.fixEns.ensNom
Total disposat	dadesFixes.fixImportSol
Data disposició del crèdit	dadesFixes.economiques[0].economiquesPK.ecoDataTramit

A partir de l'anàlisi dels atributs que intervenen en la consulta, es desprén que la definió de l'Entity Graph corresponent a la part del model de dades a accedir és la següent:

```
@NamedEntityGraph(
   name="graph.dadesFixes",
   attributeNodes={
      @NamedAttributeNode("fixCodiCredit"),
      @NamedAttributeNode("fixDesc"),
      @NamedAttributeNode("fixImportSol"),
      @NamedAttributeNode(value="fixEns", subgraph="subgraph.fixEns"),
      @NamedAttributeNode(value="economiques")
 },
 subgraphs={
      @NamedSubgraph(
         name="subgraph.fixEns",
         attributeNodes={
               @NamedAttributeNode("ensIne"),
               @NamedAttributeNode("ensNom")
         }
      )
  }
```

```
)

// Nota: Com que no s'ha definit cap subgraph per l'atribut 'economiques', es recuperaran

// tots els atributs de l'entitat Economiques.

// Nota: Els camps que pertanyen a la clau primària, per exemple 'ensId', no calen ser especificats.

public class DadesFixes {

    ...
```

Nota: Un entity graph es defineix sobre l'entitat principal a tractar per la consulta que es pretén optimitzar, en aquest cas 'DadesFixes'.

5. Ús des de la capa de servei de l'entity graph definit

Per referenciar a un determinat entity graph des d'un servei, es pot sobre escriure el mètode 'setHints' del servei en qüestió tal i com es mostra a continuació:

Nota: Aquest mètode es cridat des de 'DibaService.findAll()' just abans de fer el "fetch" que recupera les dades.

```
/**

 * Estableix l'estratègia d'accèss a usar.

*/

@Override

public void setHints( JPAQuery<DadesFixes> query )

{

query.setHint("javax.persistence.fetchgraph", entityManagerHelper.getEntityGraph("graph.dadesFixes"));

query.setHint("eclipselink.left-join-fetch", "dadesFixes.fixEns");

query.setHint("eclipselink.left-join-fetch", "dadesFixes.economiques");

}
```

Nota: L'objectiu dels dos últims "hints" és obtenir la llista de resultats amb una única consulta.

Veure la resta de hints proporcionats per EclipseLink a:

https://www.eclipse.org/eclipselink/documentation/2.6/jpa/extensions/queryhints.htm [2]

6. Consultes SQL generades

Amb la utilització de l'entity graph definit, la consulta resultant és la següent:

```
SELECT t1.FIX_CODICREDIT AS a1,
  t1.FIX_DESC          AS a2,
  t1.FIX_IMPORTSOL     AS a3,
  t1.FIX_CODIENS       AS a4,
  t0.ENS_ID            AS a5,
  t0.ENS_INE           AS a6,
  t0.ENS_NOM           AS a7,
  t2.ECO_DATATRAMIT    AS a8,
  t2.ECO_CODICREDIT    AS a9,
  t2.ECO_NUMACT        AS a10,
  t2...
FROM PSF_DADESFIXES t1
LEFT OUTER JOIN CUCC_ENS t0
    ON (t0.ENS_INE = t1.FIX_CODIENS)
LEFT OUTER JOIN PSF_ECONOMIQUES t2
    ON (t2.ECO_CODICREDIT = t1.FIX_CODICREDIT)
ORDER BY t1.FIX_CODICREDIT DESC
```

Notes:

- L'ús d'un Entity Graph associat a una consulta amb paginació que inclou una relació 1-N no funciona del tot bé en la versió d'EclipseLink utilitzada (2.6.5), ja que en determinat casos, el servei termina retornant excepció, o bé, termina correctament però amb un nombre de resultats menor de l'esperat. Suposo que això es degut a que existeixen "bugs" en la implementació d'EclipseLink.
- Com a solució a aquest problema, es pot seguir una de les següents estratègies:

1. Evitar la paginació en la consulta a executar: Aquesta tècnica es factible si s'estima que el nombre de registres a retornar no es massa elevat.

Per implementar aquesta estratègia, comentar la següent línia del servei en qüestió:

```
if (rowCount > 0) {

// Evita la limitació del nombre de registres a retornar:

// query = query.limit(rowCount);

}
```

2. Prescindir de fer el JOIN amb l'entitat referenciada:

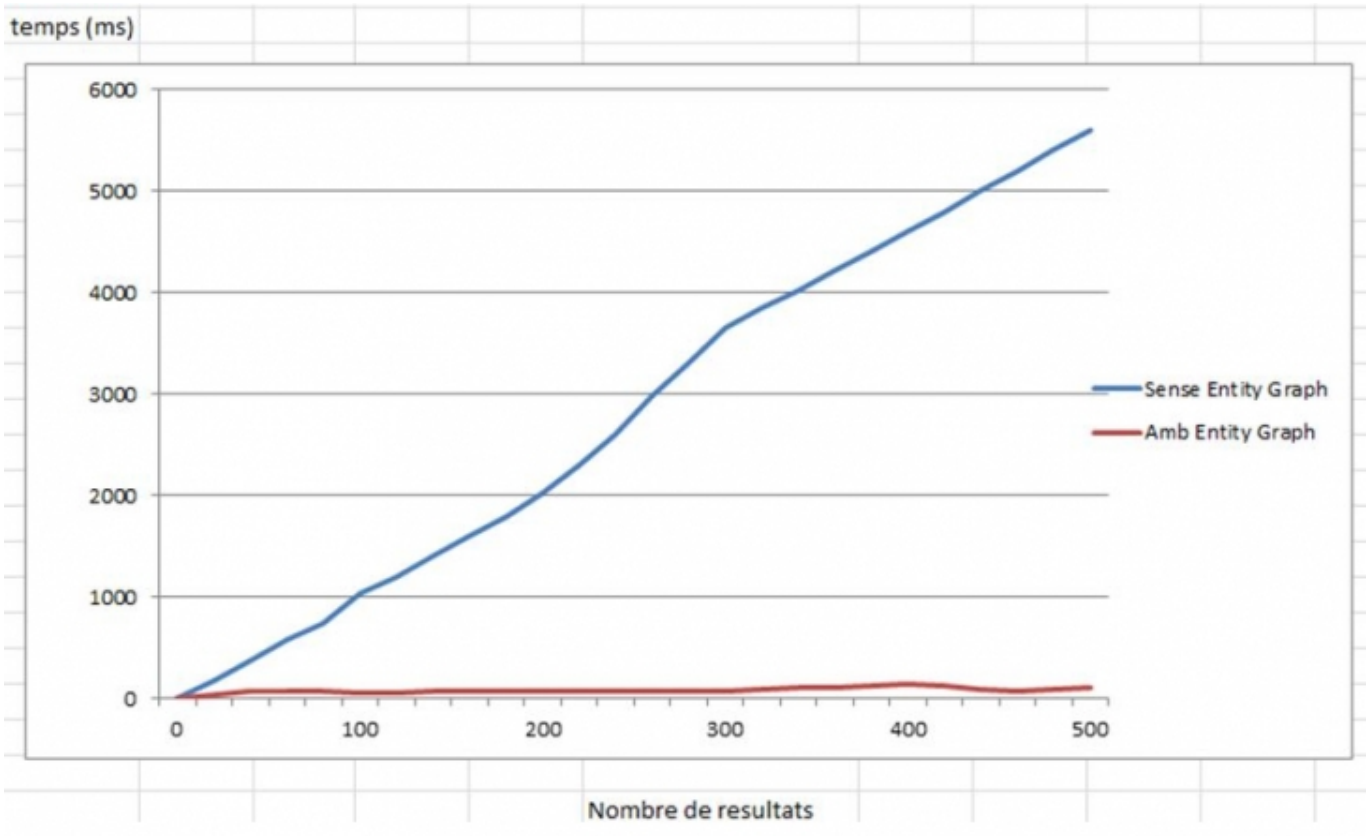
Es pot implementar comentant la següent línia del servei en qüestió:

```
//query.setHint("eclipselink.left-join-fetch", "dadesFixes.economiques");
```

En aquest cas, s'executen N consultes sobre l'entitat referenciada (Economiques), encara que el temps total d'execució pot ser menor que si no s'especifiqués la relació amb 'Economiques' des de l'entity graph definit.

7. Comparativa entre temps d'execució

La següent gràfica compara el temps (en mil·lisegons) emprat per la consulta que obté el llistat de crèdits quan s'usa l'Entity graph definit, respecte del temps emprat quan no s'utilitza:



S'observa que, en el cas de no usar l'Entity Graph, el comportament de la gràfica és lineal mentre que en el cas d'usar-lo és gairebé costant, la qual cosa implica que la utilització dels Entity Graphs resulta més convenient en aquelles situacions en què el nombre de registres a recuperar és relativament gran.

Extracte dels valors obtinguts:

Nombre de resultats	**Temps sense Entity Graph (en mil·lisegons)**	**Temps amb Entity Graph (en mil·lisegons)**
20	172	45
40	365	64
60	580	74
80	743	64
100	1040	59

8. Conclusió

La tècnica descrita pot permetre reduir significativament el nombre de consultes necessàries per accedir a un

model de dades quan s'usa JPA, la qual cosa pot impactar positivament, no tant sols quant a la reducció del temps de resposta d'una determinada funcionalitat, sinó també pel que fa a evitar la sobre càrrega de recursos compartits entre diverses funcionalitats o aplicacions, com ara, el servidor de base de dades, el servidor d'aplicacions o la xarxa de comunicacions utilitzada.

Categories: Plataforma JEE

- [3]

URL d'origen: https://comunitatdstsc.diba.cat/wiki/us-dels-entity-graphs-de-jpa-com-mecanisme-per-reduir-nombre-d-accessos-base-de-dades

Enllaços:
[1] https://comunitatdstsc.diba.cat/members/cerdeiralv
[2] https://www.eclipse.org/eclipselink/documentation/2.6/jpa/extensions/queryhints.htm
[3] https://comunitatdstsc.diba.cat/node/1345