

# **Seis Graus de Kevin Bacon**

**Equipe de documentação de Boca Raton**



## Seis Graus de Kevin Bacon: Exemplo de Programação em ECL

Equipe de documentação de Boca Raton

Copyright © 2022 HPCC Systems®. All rights reserved

Sua opinião e comentários sobre este documento são muito bem-vindos e podem ser enviados por e-mail para <docfeedback@hpccsystems.com>

Inclua a frase **Feedback sobre documentação** na linha de assunto e indique o nome do documento, o número das páginas e número da versão atual no corpo da mensagem.

LexisNexis e o logotipo Knowledge Burst são marcas comerciais registradas da Reed Elsevier Properties Inc., usadas sob licença.

HPCC Systems® é uma marca registrada da LexisNexis Risk Data Management Inc.

Os demais produtos, logotipos e serviços podem ser marcas comerciais ou registradas de suas respectivas empresas.

Todos os nomes e dados de exemplo usados neste manual são fictícios. Qualquer semelhança com pessoas reais, vivas ou mortas, é mera coincidência.

2022 Version 8.6.82-1

<b>Trabalhando com Dados</b> .....	4
Introdução .....	4
Processando os Dados .....	5
Obtendo Informações Úteis dos Dados .....	18
<b>“Próximos passos”</b> .....	22

# Trabalhando com Dados

## Introdução

Este exercício mostra a metodologia para extrair informações úteis dos dados. Localizar links e relações interessantes em datasets grandes ou massivos faz parte da rotina de uso da plataforma HPCC Systems – HPCC.

Neste exemplo, faremos o download dos arquivos de dados do Internet Movie Database (Base de dados de filmes na Internet ou IMDB) e analisaremos uma técnica para extrair links e localizar relações.

Uma vez que o conceito de atores e filmes é conceitualmente simples, todos devem compreender os dados e as relações de forma intuitiva. No entanto, os dados são abrangentes o suficiente para fornecer um exemplo sólido e inspiração para que novos usuários obtenham habilidades para “atacar” seus próprios problemas do mundo real com um HPCC.

Neste exemplo, iremos:

- Fazer o download de arquivos de dados brutos e as documentações de apoio a esses dados
- Analisar o arquivo de dados para compreender seu formato e conteúdo
- Fazer o spray do arquivo para o cluster da refinaria de dados (Thor)
- Examinar os dados e determinar o pré-processamento necessário
- Realizar o pré-processamento dos dados para gerar um novo arquivo de dados



Apesar de esse exemplo ser executado em um HPCC de nó único, você verá uma diferença significativa no desempenho em um sistema de vários nós. O verdadeiro poder do HPCC é a sua capacidade de trabalhar nas diferentes partes de um arquivo de dados em paralelo. Isso é o que chamamos de Processamento Paralelo Massivo (MPP).

# Processando os Dados

## Nós temos um arquivo de dados

A Base de Dados de Filmes na Internet (IMDB) constitui de um conjunto de arquivo de dados sobre filmes que pode ser baixado livremente da Internet.

A IMDB pode ser baixada em vários formatos, incluindo em arquivo de texto. O conjunto de dados inclui aproximadamente 48 arquivos sobre atores, atrizes, diretores, produtores e outros aspectos dos filmes cinematográficos.

Seu tamanho é gerenciável (~400MB) e suficiente para ser usado na plataforma HPCC , porém não é muito grande para download.

Os arquivos de dados de texto simples estão disponíveis nos seguintes sites ftp:

- <ftp://ftp.fu-berlin.de/pub/misc/movies/database/> (Alemanha)
- <ftp://ftp.funet.fi/pub/mirrors/ftp.imdb.com/pub/> (Finlândia)
- <ftp://ftp.sunet.se/pub/tv+movies/imdb/> (Suécia)

Os arquivos são compactados através do GNUzip para economizar espaço e largura de banda.

Inicialmente, vamos focar em dois dos maiores datasets no banco de dados IMDB

- O dataset de atores (com aproximadamente 4 milhões de registros)
- O dataset de atrizes (com aproximadamente 2 milhões de registros)
- Faça o download dos arquivos de dados de texto simples (*actors.list.gz* e *actresses.list.gz* ) em sua unidade local usando a interface ftp de sua preferência.
- Extraia os dois arquivos de dados (*actors.list* e *actresses.list* ) usando qualquer interface GNUzip.

## Analisar o arquivo de dados para compreender seu formato e conteúdo

Aqui está o exemplo dos dados do arquivo Actors.list da IMDB

```
Koolout' Starks, Johnny Nothing Like the Holidays (2008) [Alexis' Thug] <35>
Subtle Seduction (2008) [Officer Ward]
The Godfather of Green Bay (2005) (as Johnny Starks) [Marcus] <18>

La Chispa', Tony Caceria de judiciales (1997) <11>
Violencia en la sierra (1995) [Victoriano] <4>
```

Observe que o arquivo de texto dos atores está estruturado da seguinte forma

```
Blankline
Actorname_i Moviename (year) [role] <listing position>
      Moviename (year) [role] <listing position>
      Moviename (year) [role] <listing position>
      :
Blankline
Actorname_j \t Moviename (year) [role] <listing position>
```

Blankline :

## Enviando os arquivos de dados de entrada para sua Zona de Entrada de Arquivos

Nesta etapa, você copiará os arquivos de dados para um local onde eles possam ser distribuídos aos nós de seu cluster HPCC. Uma zona de entrada de arquivos é um local de armazenagem anexado ao seu HPCC. Ela possui um utilitário em execução para facilitar o spraying (processo de distribuir dados aos nós) para um cluster.

Para arquivos de dados menores, com tamanho máximo de 2GB, você pode usar o utilitário upload/download do arquivo no ECL Watch. Os arquivos de dados de amostra possuem aproximadamente 400 mb.

Em seguida, você distribuirá (ou fará o spray) o dataset para todos os nós no cluster do HPCC. O poder do HPCC vem da sua capacidade de atribuir vários processadores para trabalhar nas diferentes partes do arquivo de dados em paralelo.

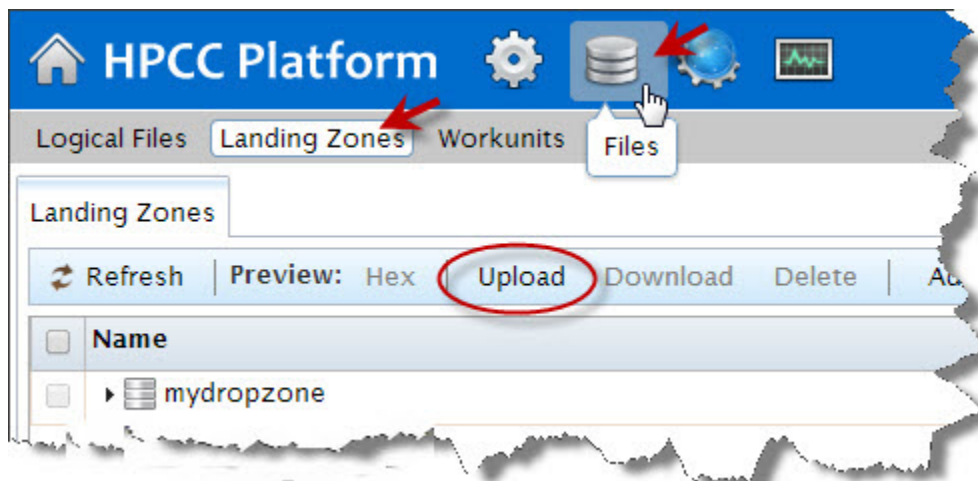
1. Caso ainda não tenha feito, baixe os arquivos de dados de amostra do site ftp, como mencionado na seção anterior.
2. Extraia os arquivos para uma pasta em seu computador local.
3. Em seu navegador, acesse a URL do **ECL Watch** . Por exemplo, <http://nnn.nnn.nnn.nnn:8010>, onde nnn.nnn.nnn.nnn é o endereço IP do seu ESP Server.



Seu endereço IP poderá ser diferente dos endereços fornecidos nas imagens de exemplo. Use o endereço IP fornecido pela sua instalação **sua** installation.

4. Na página do ECL Watch, clique no **ícone Files** , e em seguida no link **Landing Zones (Zona de entrada de arquivos)** .

**Figure 1. Upload/Download**

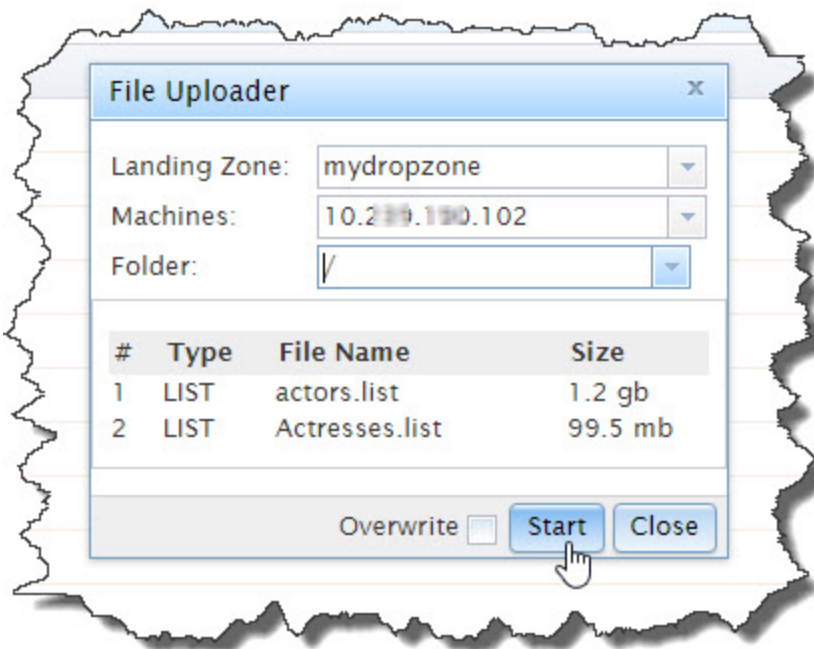


Após clicar no link **Upload** , uma caixa de diálogo do arquivo será exibida.

5. Localize os arquivos em seu computador local e use a função de multi-seleção para selecionar os arquivos que deseja enviar, pressionando em seguida o botão **Open** .

Os arquivos selecionados aparecerão. Os arquivos de dados possuem os seguintes nomes: *actors.list* e *actresses.list*

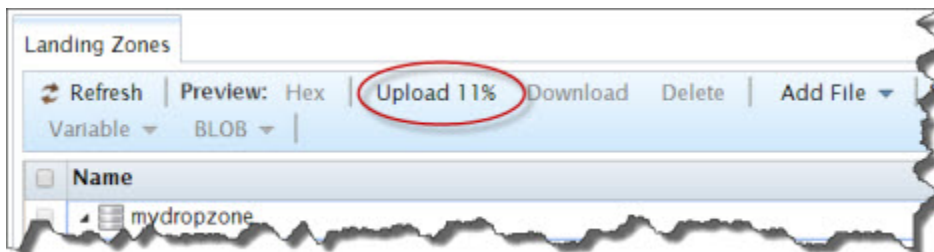
**Figure 2. Dropzones e Arquivos**



6. Pressione o botão **Start** para enviar os arquivos.

É possível acompanhar o progresso do envio.

**Figure 3. Progresso de envio**



## Spray do arquivo de dados para o seu *Cluster da Refinaria de Dados (Thor)*

Para usar o arquivo de dados em seu HPCC System, é preciso fazer o "spray" (distribuir) desse arquivo para todos os nós. O spray ou importação é a transferência de um arquivo de dados de um local (como a zona de entrada de arquivos) para diversas partes do arquivo ou nós em um cluster.

O arquivo distribuído passa a ter um *nome de arquivo lógico* como segue: `~thor::in::IMDB::actors.list` O sistema mantém uma lista de arquivos lógicos e as localizações do arquivo físico correspondente das partes do arquivo.

- Abra o ECL Watch usando a URL:

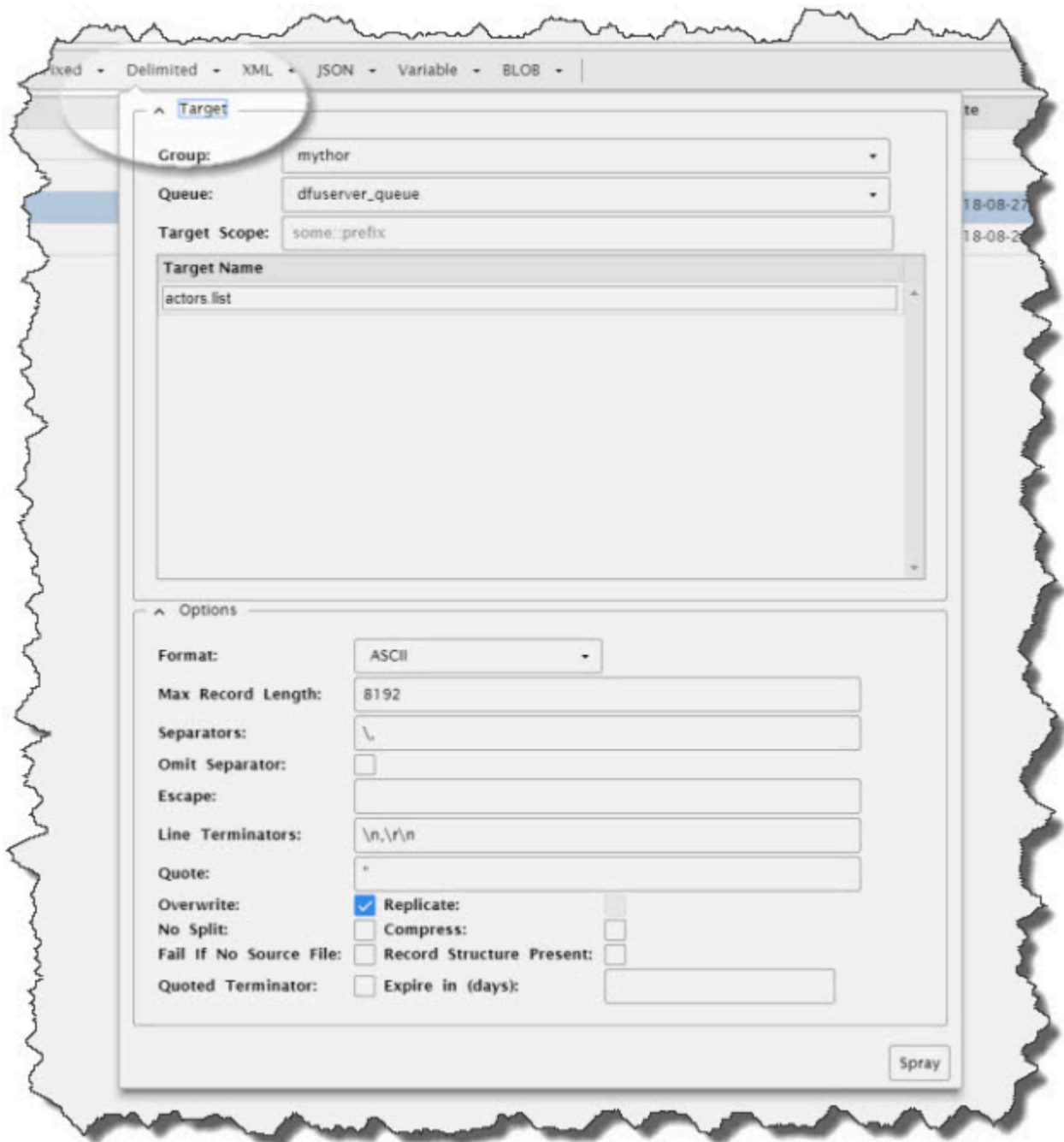


<http://nnn.nnn.nnn.nnn:pppp> (onde nnn.nnn.nnn.nnn é o endereço IP do seu ESP Server e pppp é a porta. A porta padrão é 8010)

- Clique no ícone **Files** , e em seguida no link **Landing Zone** (Zona de entrada de arquivos) na navegação.
- Selecione os dois arquivos (actors.list e actresses.list ) e pressione o botão Delimited.

A caixa de diálogo **Delimited Spray** será exibida.

**Figure 4. Spray delimitado**



- Selecione mythor na lista suspensa **Group** .

O endereço IP é preenchido automaticamente e o Caminho local é parcialmente preenchido com o nome da pasta padrão na Landing Zone. Observação: Community Edition normalmente possui apenas uma landing zone definida.

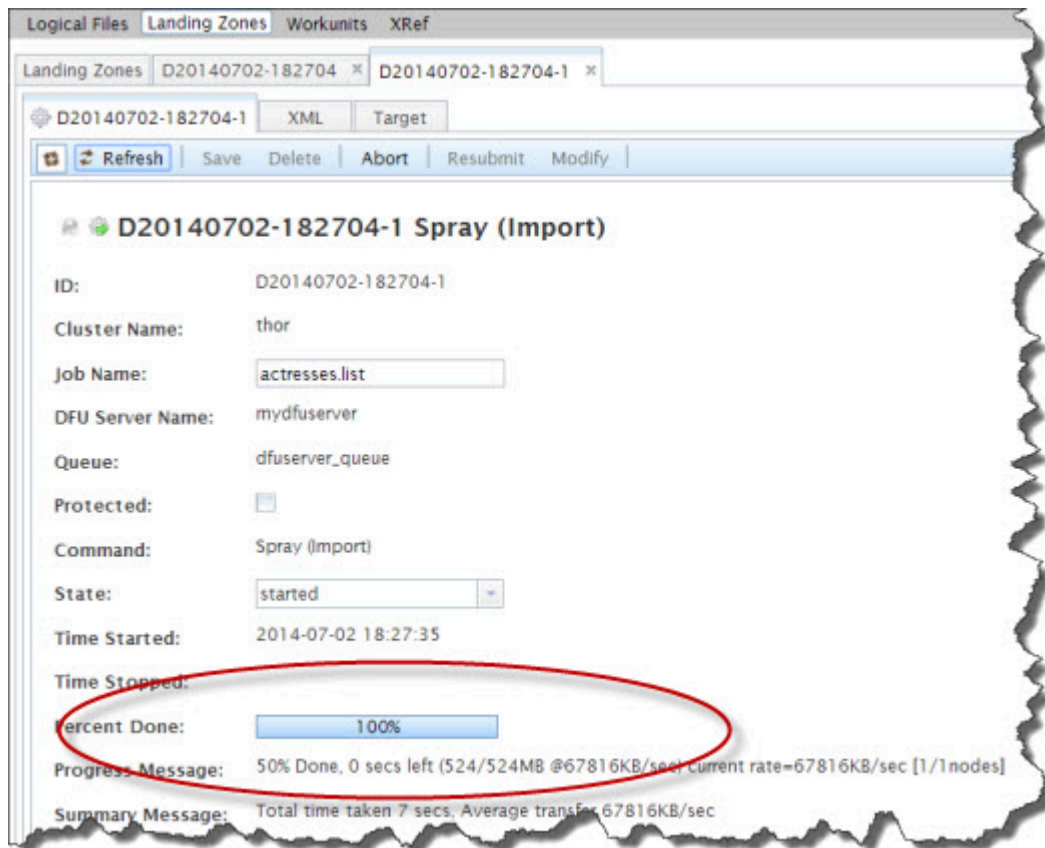
- Preencha o Escopo de destino (Target Scope) **~thor::in::IMDB**
- Preencha os demais parâmetros (caso ainda não tenham sido preenchidos).
  - Max Record Length 8192 (Máximo tamanho do registro 8192)
  - Separator \, (Separador \,)
  - Line Terminator \n,\r\n (Terminador de linhas \n,\r\n)
  - Quote: ‘ (Aspas: ‘)
- Não se esqueça de marcar a caixa **Overwrite** (Substituir).

Se disponível, certifique-se de que a caixa **Replicate** esteja marcada. (A opção replicar está disponível apenas em sistemas em que a replicação tenha sido ativada.)

- Pressione o botão **Spray** .

Uma aba será aberta para cada arquivo. Nessas abas, é possível monitorar o progresso de cada spray DFU .

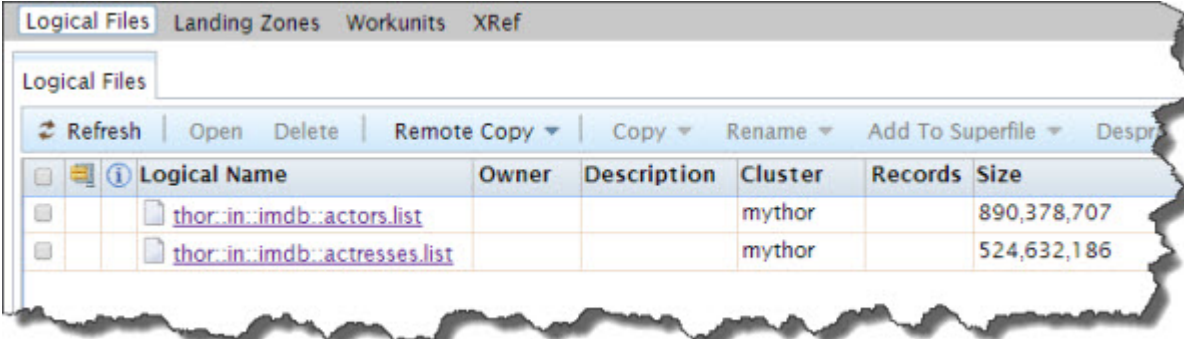
**Figure 5. Ver progresso**



- Após os sprays tiverem sido concluídos, podemos consultar os arquivos lógicos no HPCC para ver os arquivos que foram distribuídos aos nós.
- Clique no link **Logical Files**

Os arquivos serão exibidos na lista de arquivos lógicos:

Figure 6. Exibir arquivos lógicos



The screenshot shows the 'Logical Files' window in the ECL IDE. It features a toolbar with buttons for Refresh, Open, Delete, Remote Copy, Copy, Rename, Add To Superfile, and Despr. Below the toolbar is a table with the following columns: Logical Name, Owner, Description, Cluster, Records, and Size. Two files are listed:

Logical Name	Owner	Description	Cluster	Records	Size
<a href="#">thor.in:imdb:actors.list</a>			mythor		890,378,707
<a href="#">thor.in:imdb:actresses.list</a>			mythor		524,632,186

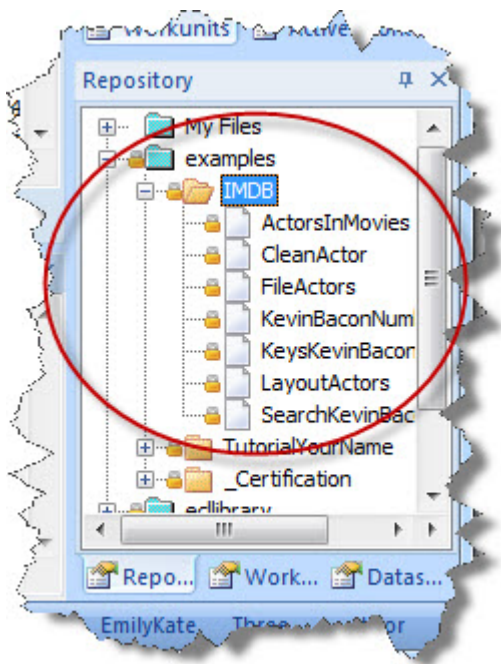
## Trabalhando com Dados

Nesta parte do exemplo, vamos programar o código ECL para que possamos ler e os arquivos de dados distribuídos. Vamos definir e executar consultas simples para que possamos avaliá-lo e determinar qualquer pré-processamento necessário.

- Inicie o ECL IDE (Start >> All Programs >> HPCC Systems >> ECL IDE )
- Faça login em seu ambiente.
- Expanda a pasta ECL de **exemplos** na caixa de ferramentas Repository.
- Expanda a pasta **IMDB**

Todos os arquivos ECL que precisam concluir este tutorial estão localizados na pasta IMDB.

Figure 7. Arquivos ECL IMDB



- Abra o arquivo ECL CleanActor e examine o código.

Esse código lê e processa o arquivo de texto bruto. Os comentários abaixo descrevem o processo:

```
IMPORT Std;

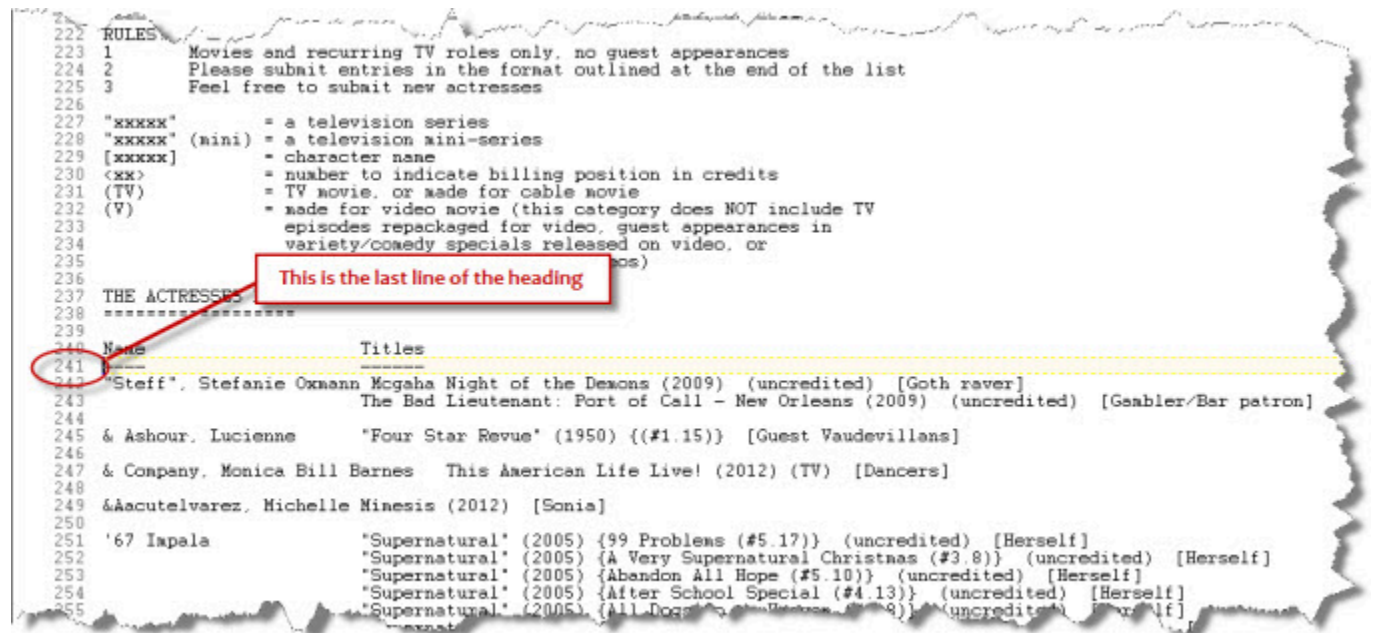
EXPORT STRING CleanActor(String infl) := FUNCTION
  //this can be refined later
  s1 := Std.Str.FindReplace(infl, '\', ''); // replace apostrophe
  s2 := Std.Str.FindReplace(s1, '\t', '');  //replace tabs
  s3 := Std.Str.FindReplace(s2, '----', ''); // replace multiple ----
  return TRIM(s3, LEFT, RIGHT);
END;
```

## Examinando os dados

Nesta seção, vamos analisar os dados e determinar se é há algum pré-processamento para ser realizado. Essa é a etapa no processo de desenvolvimento na qual convertemos dados brutos para um formato realmente utilizável.

**Observação:** O arquivo da IMDB, `FileActors.ecl` file especifica o tamanho do cabeçalho nos arquivos (`actors.list` e `actresses.list`.) O valor do `HEADING()` no código de exemplo estava correto no momento em que baixamos os dados IMDB, mas poderia mudar a qualquer momento. Sugerimos abrir um editor de textos e verificar o número da linha onde o cabeçalho termina e os dados começam (como mostrado abaixo).

Figure 8. `actors.list` no editor de texto



- Abra uma nova Janela do compilador (CTRL+N) e escreva o seguinte código:

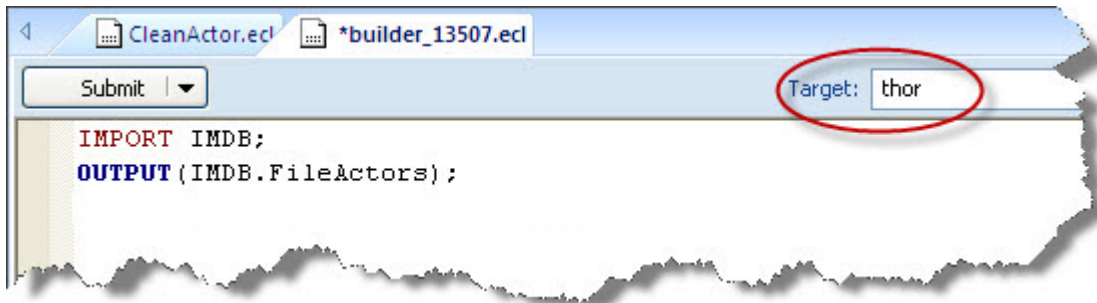
```
IMPORT IMDB; OUTPUT(IMDB.FileActors);
```

- Pressione o botão de verificação de sintaxe localizado na barra de ferramentas principal (ou pressione F7).

É sempre recomendado verificar a sintaxe antes do envio.

- Verifique se o cluster selecionado é o *thor* e pressione o botão **Submit**.

**Figure 9. Enviar para o Thor**

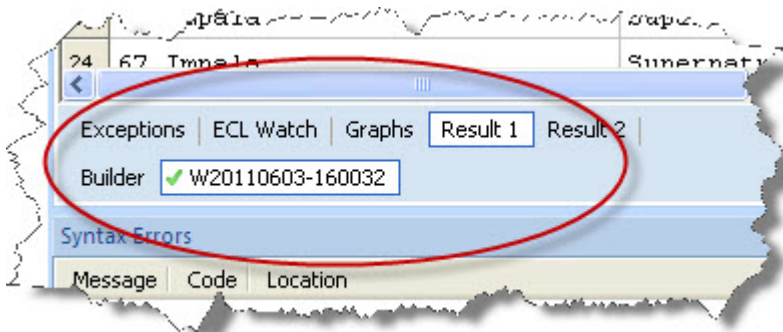


- Quando a workunit for concluída, ela aparece com uma marca de verificação verde. ✓

**Observação:** Dependendo do tamanho do seu cluster e da velocidade do(s) seu(s) servidor(es), esse processo pode levar vários minutos. Se estiver sendo executado em uma máquina virtual, pode demorar até 45 minutos para ser concluído.

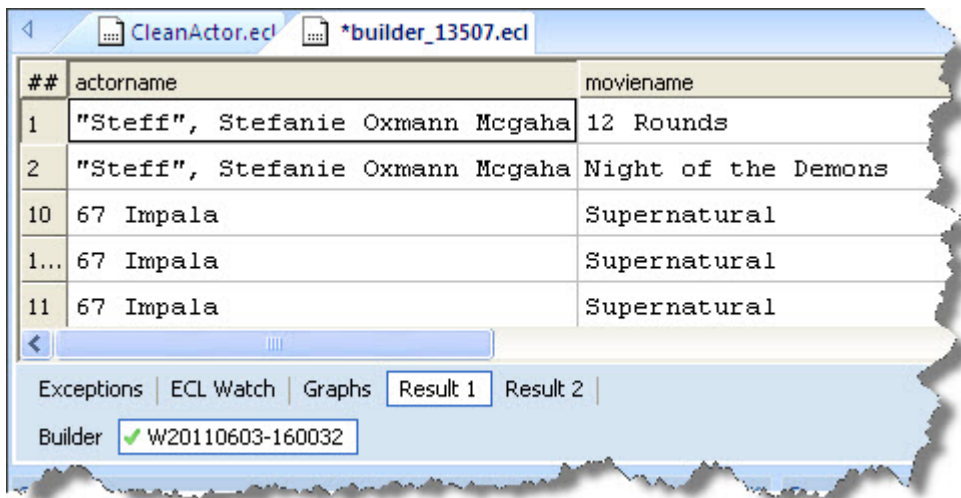
- Selecione a aba Workunit (a que possui um número e uma marca de verificação ao lado) e selecione a aba **Result 1**

**Figure 10. Selecionar Workunit**



- Role a página para baixo para ver mais registros.

**Figure 11. Ver mais registros**



##	actorname	moviename
1	"Steff", Stefanie Oxmann Mcgaha	12 Rounds
2	"Steff", Stefanie Oxmann Mcgaha	Night of the Demons
10	67 Impala	Supernatural
1...	67 Impala	Supernatural
11	67 Impala	Supernatural

- Feche a janela do Compilador.

## Processando os dados Extract, Transform e Load

Nesta seção, nós vamos programar para transformar os dados originais do ator, como segue:

- A partir dos dados brutos de atores, faremos uma operação ETL (Extrair, Transformar, Carregar) para criar um conjunto de relação **actor\_movie** .
- Vamos construir um conjunto de pesquisa dos graus de separação de Kevin Bacon. Essa é a estrutura que consultaremos para responder à pergunta:

*Quantos graus de separação existem entre o Ator X e Kevin Bacon?*

**Por exemplo:** Usando Jon Lovitz como o ator, queremos as informações da seguinte forma:

Jon Lovitz ( (estava no) Filme X ( (com) Ator2 ((que estava no) Filme Y ( (com) Kevin Bacon

Vamos então gravar esse arquivo novo em nosso cluster Thor para que ele seja usado em consultas de parâmetro.

- No ECL IDE , vá para o painel Repository e expanda a pasta IMDB .
- Abra o arquivo ECL ActorsInMovies.

O código neste arquivo ECL é mais ou menos assim:

```
/* *****  
## Copyright 2011 HPCC Systems®. All rights reserved.  
***** */  
**
```



## Seis Graus de Kevin Bacon: Exemplo de Programação em ECL

### Trabalhando com Dados

---

```
* Produce a slimmed down version of the IMDB actor AND actress files to
* permit more efficient join operations.
* Filter out the movie records we do not want in building our KBacon Number sets.
*
*/

IMPORT $ AS IMDB;
IMPORT Std;

// Filter out TV movies, Videos AND some documentary type collections
ds_IMDB := IMDB.FileActors(actorname!='' AND moviename != '' AND
                          Std.Str.Find(moviename,'Boffo',1) = 0 AND
                          Std.Str.Find(moviename,'Slasher Film',1) = 0 AND
                          movie_type != 'Video' AND isTVseries = 'N' AND
                          movie_type != 'For TV');

//Slim the records down to bare essentials for searching AND joining
slim_IMDB_rec := RECORD
  STRING50 actor;
  STRING150 movie;
END;

slim_IMDB_rec slim_it(ds_IMDB L):= TRANSFORM
  SELF.actor := Std.Str.FindReplace(L.actorname,'(I)','');
  SELF.movie := L.moviename;;
END;

IMDB_names := PROJECT(ds_IMDB, slim_it(LEFT));

export ActorsInMovies := IMDB_Names : persist('~temp::IMDB::ActorsInMovies');
```

Isso define um dataset relacional:-- actor:movie. Usaremos essa definição mais tarde.

# Obtendo Informações Úteis dos Dados

## Links e Graus de Separação

Agora que temos nossos dados em um formato útil, definimos a relação, e o arquivo está salvo, podemos escrever mais códigos para usar o novo arquivo de dados.

Queremos saber quantos atores estão a uma distância  $N$  de Kevin Bacon. Para isso, vamos construir conjuntos dos coadjuvantes de Kevin Bacon separados por um número KBacon.

- Abra o arquivo ECL KevinBaconNumberSets.

Esse código ECL conta o número de atores com "*números bacon (bacon number)*" de 1 a 7, sendo os que estão no nível 7 aqueles com maior grau de separação. Usaremos isso mais tarde para realizar buscas através da criação de um índice.

```
/* *****  
ATTRIBUTE PURPOSE:  
  Produce a series of sets for Actors and Movies that are : distance-0  
  away (KBacons Direct movies ), distance-2 Away KBacon's Costars Movies ,  
  distance-3 away - Movies of Costars of Costars etc all the way upto level 7  
  
The nested attributes below are shown here together for the benefit of the reader.  
  
Notes on variable naming convention used for costars and movies  
KBMovies          : Movies Kevin Bacon Worked in      (distance 0)  
KBCoStars          : Stars who worked in KBMovies      (distance 1)  
KBCoStarMovies     : Movies worked in by KBCoStars  
                    except KBMovies (distance 1)  
KBCo2Stars         : Stars(Actors) who worked in KBCoStarMovies (distance 2)  
KBCo2StarMovies    : Movies worked in by KBCo2Stars  
                    except KBCoStarMovies (distance 2)  
KBCo3Stars         : Stars(Actors) who worked in KBCo2StarMovies (distance 3)  
KBCo3StarMovies    : Movies worked in by KBCo3Stars  
                    except KBCo2StarMovies (distance 3)  
etc..  
***** */  
  
IMPORT Std;  
IMPORT IMDB;  
  
EXPORT KevinBaconNumberSets := MODULE  
  // Constructing a proper name match function is an art within itself  
  // For simplicity we will define a name as matching if both first and last name  
  //are found within the string  
  
  NameMatch(string full_name, string fname,string lname) :=  
    Std.Str.Find(full_name,fname,1) > 0 AND  
    Std.Str.Find(full_name,lname,1) > 0;  
  
  //----- Get KBacon Movies  
  AllKBEntries := IMDB.ActorsInMovies(NameMatch(actor,'Kevin','Bacon'));  
  EXPORT KBMovies := DEDUP(AllKBEntries, movie, ALL); // Each movie should ONLY occur once  
  
  //----- Get KBacon CoStars  
  CoStars := IMDB.ActorsInMovies(Movie IN SET(KBMovies,Movie));  
  EXPORT KBCoStars := DEDUP( CoStars(actor<>'Kevin Bacon'), actor, ALL);  
  
  //----- Get KBacon Costars' Movies  
  // CSM = First find all of the movies that a KBCoStar has been in  
  
  CSM := DEDUP(JOIN(IMDB.ActorsInMovies,KBCoStars, LEFT.actor=RIGHT.actor,
```

## Seis Graus de Kevin Bacon: Exemplo de Programação em ECL

### Trabalhando com Dados

---

```
        TRANSFORM(LEFT), LOOKUP),
        movie,ALL);

// Now we need to remove all of those that KB was in himself
// We can use a set; KB has not been in (quite!) that many movies

EXPORT KBCoStarMovies := CSM(movie NOT IN SET(KBMovies,movie));

//----- Bacon # 2 Actors
// To be a Co2Star of Kevin Bacon you must have appeared in a movie with a
//CoStar of Kevin Bacon
// This corresponds to having a Bacon number of 2
// We are now getting towards the expensive part of the process
KBCo2S := DEDUP(JOIN(IMDB.actorsInMovies, KBCoStarMovies, LEFT.movie=RIGHT.movie,
        TRANSFORM(LEFT), LOOKUP),
        actor, ALL);

// KCCo2S = ALL Actors appearing in Movies of KBacon's CoActors
// The above is all the people in the movies; but some will have been co-stars of KB
//directly - these must be removed
// The LEFT ONLY join removes items in one list from another

EXPORT KBCo2Stars := JOIN(KBCo2S, KBCoStars, LEFT.actor=RIGHT.actor,
        TRANSFORM(LEFT), LEFT ONLY);

//----- bacon # 2 Movies
// Co2SM = what movies have all the Co2Stars been in?
Co2SM := DEDUP(JOIN(IMDB.actorsInMovies, KBCo2Stars, LEFT.actor=RIGHT.actor,
        TRANSFORM(LEFT), LOOKUP),
        movie, ALL);
// Co2SM = ALL Movies KBCo2Stars have been in
// Of course some of these movies will have CoStars in too and thus will already have
//been listed. Note this list will not contain any Kevin Bacon movies OR the movie would
//have been reached earlier!

Export KBCo2StarMovies := JOIN(Co2SM, KBCoStarMovies, LEFT.movie=RIGHT.movie,
        TRANSFORM(LEFT),LEFT ONLY);

//----- bacon #3 Actors
// Find people with a Bacon number of 3
// This code is very similar to KBCo2Stars; one might be tempted to common up into a
// function or macro. However it is worth looking at the attribute counts first; we may be
// down to a small enough set that we can start using in-memory functions (e.g.,SET) again.

KBCo3S := DEDUP(JOIN(IMDB.actorsInMovies, KBCo2StarMovies, LEFT.movie=RIGHT.movie,
        TRANSFORM(LEFT), LOOKUP),
        actor, ALL);

// KBCo3S = ALL CoStars in KBCo2Star Movies
// The above is all the people in the movies; but some will have been co2stars of KB
// directly - these must be removed. The LEFT ONLY join removes items in one list from
// another. There should not be any direct CoStars in this list (or the movie would have
// been a CoStarMovie not a CoCoStarMovie)

EXPORT KBCo3Stars := JOIN(KBCo3S, KBCo2Stars, LEFT.actor=RIGHT.actor,
        TRANSFORM(LEFT),LEFT ONLY);

//----- bacon #3 Movies
// So what movies have all the KBCo3Stars been in?

Co3SM := DEDUP(JOIN(IMDB.actorsInMovies, KBCo3Stars, LEFT.actor=RIGHT.actor,
        TRANSFORM(LEFT), LOOKUP),
        movie, ALL);

// Co3SM = ALL Movies KBCo3Stars have been in
```

## Seis Graus de Kevin Bacon: Exemplo de Programação em ECL

### Trabalhando com Dados

---

```
// Of course some of these movies will have KBCo2Stars in too and thus will already have
// been listed. Note We ONLY have to remove one level back from the list; previous levels
// cannot be reached by definition

EXPORT KBCo3StarMovies := JOIN(Co3SM, KBCo2StarMovies, LEFT.movie=RIGHT.movie,
                                TRANSFORM(LEFT),LEFT ONLY);

//-----bacon #4 Actors
KBCo4S := DEDUP(JOIN(IMDB.ActorsInMovies, KBCo3StarMovies, LEFT.movie=RIGHT.movie,
                    TRANSFORM(LEFT), LOOKUP),
               actor, ALL);

EXPORT KBCo4Stars := JOIN(KBCo4S, KBCo3Stars, LEFT.actor=RIGHT.actor,
                          TRANSFORM(LEFT),LEFT ONLY);

//----- bacon #4 Movies
// So what movies have all the Co4Stars been in?

Co4SM := DEDUP(JOIN(IMDB.ActorsInMovies, KBCo4Stars, LEFT.actor=RIGHT.actor,
                   TRANSFORM(LEFT), LOOKUP),
               movie, ALL);

// Co4SM = ALL Movies KBCo4Stars have been in
// Of course some of these movies will have Co3Stars in too and thus will already have
// been listed. Note We ONLY have to remove one level back from the list; previous levels
// cannot be reached by definition

EXPORT KBCo4StarMovies := JOIN(Co4SM, KBCo3StarMovies, LEFT.movie=RIGHT.movie,
                                TRANSFORM(LEFT),LEFT ONLY);

//----- bacon #5 Stars
KBCo5S := DEDUP(JOIN(IMDB.ActorsInMovies, KBCo4StarMovies, LEFT.movie=RIGHT.movie,
                    TRANSFORM(LEFT), LOOKUP),
               actor, ALL);

EXPORT KBCo5Stars := JOIN(KBCo5S, KBCo4Stars, LEFT.actor=RIGHT.actor,
                          TRANSFORM(LEFT),LEFT ONLY);

//----- bacon #5 Movies
Co5SM := DEDUP(JOIN(IMDB.ActorsInMovies, KBCo5Stars, LEFT.actor=RIGHT.actor,
                   TRANSFORM(LEFT), LOOKUP),
               movie,ALL);

EXPORT KBCo5StarMovies := JOIN(Co5SM, KBCo4StarMovies, LEFT.movie=RIGHT.movie,
                                TRANSFORM(LEFT),LEFT ONLY);

//----- bacon #6 Stars
// Find people with a Bacon number of 6
// KBCo5 is getting small again - can move back down to the SET?

KBCo6S := DEDUP(IMDB.ActorsInMovies(movie IN SET(KBCo5StarMovies, movie)),
               actor, ALL);

EXPORT KBCo6Stars := JOIN(KBCo6S, KBCo5Stars, LEFT.actor=RIGHT.actor,
                          TRANSFORM(LEFT),LEFT ONLY);

//----- bacon #6 Movies
Co6SM := DEDUP(IMDB.ActorsInMovies(actor IN SET(KBCo6Stars, actor)), movie, ALL);

EXPORT KBCo6StarMovies := Co6SM(movie NOT IN SET(KBCo5StarMovies, movie));

//----- bacon #7 Movies
// Find people with a Bacon number of 7
KBCo7S := DEDUP(IMDB.ActorsInMovies(movie IN SET(KBCo6StarMovies,movie)), actor, ALL);
EXPORT KBCo7Stars := KBCo7S(actor NOT IN SET(KBCo6Stars, actor));
```

## Seis Graus de Kevin Bacon: Exemplo de Programação em ECL

### Trabalhando com Dados

```
//----- We just have to count them all !! (How many holes in Albert Hall?)
EXPORT doCounts := PARALLEL(
  OUTPUT(COUNT(KBMovies), NAMED('KBMovies')),
  OUTPUT(COUNT(KBCoStars), NAMED('KBCoStars')),
  OUTPUT(COUNT(KBCoStarMovies), NAMED('KBCoStarMovies')),
  OUTPUT(COUNT(KBCo2Stars), NAMED('KBCo2Stars')),
  OUTPUT(COUNT(KBCo2StarMovies), NAMED('KBCo2StarMovies')),
  OUTPUT(COUNT(KBCo3Stars), NAMED('KBCo3Stars')),
  OUTPUT(COUNT(KBCo3StarMovies), NAMED('KBCo3StarMovies')),
  OUTPUT(COUNT(KBCo4Stars), NAMED('KBCo4Stars')),
  OUTPUT(COUNT(KBCo4StarMovies), NAMED('KBCo4StarMovies')),
  OUTPUT(COUNT(KBCo5Stars), NAMED('KBCo5Stars')),
  OUTPUT(COUNT(KBCo5StarMovies), NAMED('KBCo5StarMovies')),
  OUTPUT(COUNT(KBCo6Stars), NAMED('KBCo6Stars')),
  OUTPUT(COUNT(KBCo6StarMovies), NAMED('KBCo6StarMovies')),
  OUTPUT(COUNT(KBCo7Stars), NAMED('KBCo7Stars')),
  OUTPUT(KBCo7Stars)
);
END;
```

- Abra uma nova Janela do Compilador e digite:

```
IMPORT IMDB; IMDB.KevinBaconNumberSets.doCounts;
```

- Verifique a sintaxe e pressione o botão **Submit**.

**Observação:** Dependendo do tamanho do seu cluster e da velocidade do(s) seu(s) servidor(es), esse processo pode levar vários minutos. Se estiver sendo executado em uma máquina virtual, isso pode demorar até uma hora para ser concluído.

- Quando o processo tiver sido concluído, cada linha exibida abaixo se transforma em sua própria aba de resultado. Você obterá uma amostra do resultado como segue:

**Observação:** Os arquivos de dados deste tutorial mudam com frequência; seus resultados podem ser diferentes dos resultados mostrados neste documento.

KB Movies	71
KB Co Stars	3520
KB Co Star Movies	33504
KB Co 2 Stars	430145
KB Co 2 Star Movies	251867
KB Co 3 Stars	896009
KB Co 3 Star Movies	51650
KB Co 4 Stars	102729
KB Co 4 Star Movies	2634
KB Co 5 Stars	6080
KB Co 5 Star Movies	190
KB Co 6 Stars	450
KB Co 6 Star Movies	14
KB Co 7 Stars	22

## “Próximos passos”

Agora que os dados foram processados com sucesso e os links foram determinados, qual é o próximo passo?

Dois arquivos ECL adicionais foram incluídos na pasta IMDB e podem ser usados juntamente com os exemplos que você já trabalhou através deste tutorial:

- **KeysKevinBacon** -- Cria um índice de atores/atrizes e dos filmes que eles atuaram

Esse índice deve ser criado antes de executar as consultas para que seja possível encontrar o grau de separação entre Kevin Bacon e um ator de sua escolha.

Para criar o índice, abra a janela do compilador e digite o seguinte código:

```
IMPORT IMDB; IMDB.KeysKevinBacon.BuildAll;
```

Pressione o botão **Submit** para executar o código ECL code e criar o índice.

**SearchKevinBaconLinks** -- Realiza uma busca no índice que você criou para informar o grau de separação entre um ator e Kevin Bacon.

Por exemplo, para encontrar o grau de separação entre Kevin Bacon e Andi Everingham, abra a janela do compilador e digite o código:

```
IMPORT IMDB; IMDB.SearchKevinBaconLinks('Everingham, Andi');
```

Verifique se o cluster selecionado é o seu cluster *hThor*, e em seguida pressione o botão **Submit** para executar a consulta.

Quando a consulta estiver sido concluída, clique na guia ID da workunit.

Dois resultados serão mostrados.

**Result1** mostra o grau de separação entre o ator e Kevin Bacon.

Os resultados devem ser interpretados da seguinte forma:

Ator se encontra no nível 1 - O ator escolhido e Kevin Bacon atuaram em um filme juntos.

Ator se encontra no nível 2 - O ator escolhido atuou em um filme com um ator que atuou em um filme com Kevin Bacon.

Quanto mais alto for o nível, maior será o grau de separação entre o ator que você escolheu e Kevin Bacon.

Neste exemplo, o ator se encontra no nível 6; isso indica que existem 6 graus de separação entre Andi Everingham e Kevin Bacon.

**Result2** mostra o nível (grau de separação), o nome do ator e o filme em que ele atuou.

Cada linha mostra um ator e o filme em que ele atuou, vinculando cada ator entre si e, eventualmente, com Kevin Bacon.

Divirta-se descobrindo os graus de separação entre qualquer ator e Kevin Bacon.

Lembre-se de primeiramente criar o índice.