

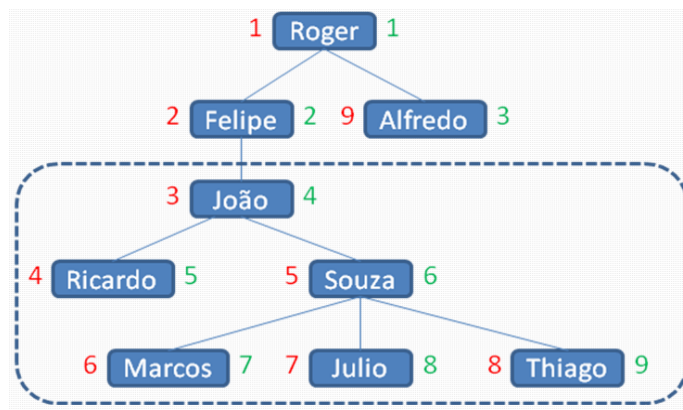
Adabas

1 - Noções Básicas

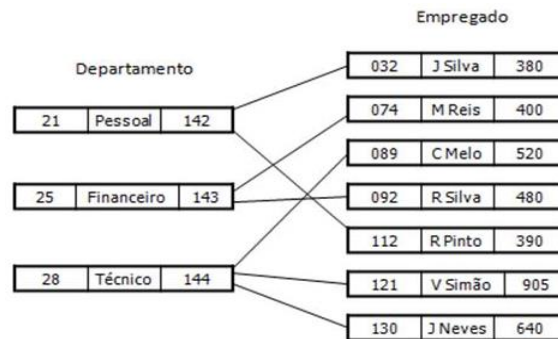
Quase todos os grandes bancos de dados governamentais mantidos pelo BNDES, BACEN, SERPRO e Dataprev estão em Adabas. Eles contêm informações de praticamente todos os brasileiros, como o banco de dados do SUS. Também o Banco do Brasil habilita sua utilização em paralelo com VSAM e DB2. A BrasilTelecom, outros bancos e grandes empresas de diversos setores utilizam Adabas. Por serem sistemas legados, suas mudanças para os modelos de bancos de dados relacionais implicariam investimentos altíssimos e pouca segurança no resultado final. O Adabas é utilizado no Bradesco num sistema de financiamento que é proveniente do antigo FINASA, é o único sistema em Adabas no Bradesco. É também utilizado no Departamento Estadual de Trânsito de Sergipe onde mantém todas informações sobre veículos e condutores do Estado. Na Educação, ele é utilizado no sistema de gestão acadêmica da Universidade Presbiteriana Mackenzie.

Diferente de um banco de dados relacional o Adabas usa o conceito de listas invertidas, sendo classificado como um banco de dados hierárquico por alguns autores e às vezes como um banco de dados em rede. A diferença desses dois tipos de SGBD é bastante sutil, no banco hierárquico você tem o conceito de nós pais-filhos como em uma árvore genealógica e no banco em rede um nó pode ter vários pais.

Hierárquico



Modelo em rede



Essa classificação nem sempre é precisa, assim como acontece com outros SGBD's de mercado quando as características dos modelos são incorporadas simultaneamente, por exemplo o SGBD PostgreSQL que é um banco objeto-relacional. Para fins didáticos vamos dizer que o Adabas é um banco hierárquico mas a sua principal característica é o uso de lista invertida, esse conceito é a mais popular estratégia de sistemas de busca, usada em serviços como o Google.

Realizar pesquisa de um termo em uma lista tradicional ou em um campo de uma tabela relacional pode exigir muito tempo devido a busca sequencial, entretanto com uma lista invertida a estrutura do Adabas pode se avançar diretamente ao termo procurado. Por isso, o uso deste recurso permite que os resultados sejam obtidos de forma consideravelmente mais rápida, a diferença de tempo de resposta tende a aumentar quanto maior for o arquivo ou lista pesquisada.

Veja a diferença no exemplo abaixo, primeiro um exemplo de Lista normal com chave e valor, e logo abaixo a lista invertida mapeando as ocorrências pelos registros.

Lista Normal:

- 1: "Sei que sou"
- 2: "Sou o que sei"
- 3: "Sou uma banana"

Obtemos a seguinte lista invertida:

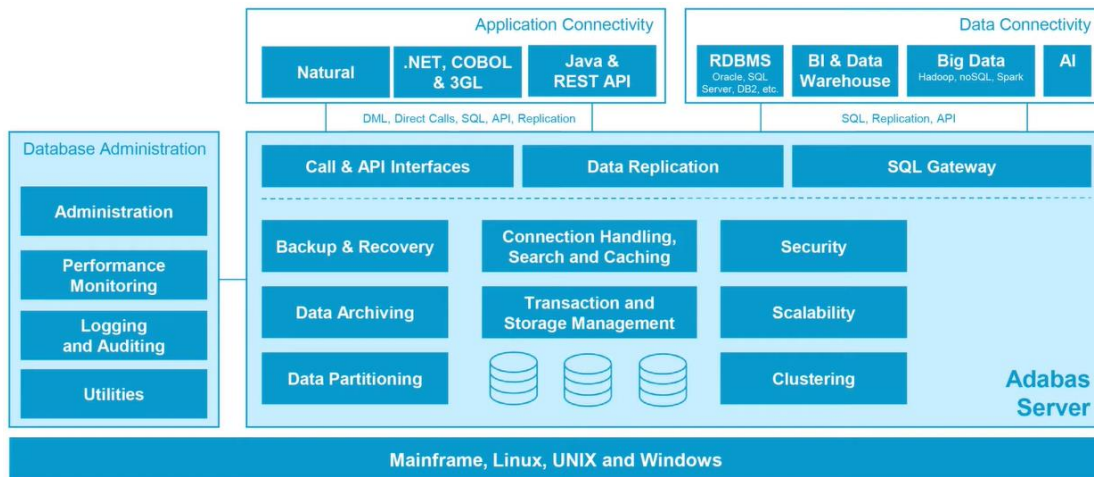
"sei" : {1, 2}
"que" : {1, 2}
"sou" : {1, 2, 3}
"o" : {2}
"uma" : {3}
"banana" : {3}

Além desse conceito chave, o Adabas possui uma estrutura próxima de um modelo relacional mas com algumas diferenças básicas, como:

- Arquivos, e não tabelas, como as principais unidades de organização.
- Registros, e não células, como menores unidades de organização.
- Campos, e não colunas, como componentes de uma unidade.
- Não baseado no sistema SQL, precisando de um mecanismo de busca externo.

O Adabas possui diversos mecanismos para conexão com outros SGBD's, por exemplo o uso de ODBC. O ODBC é um protocolo que você pode usar para conectar um banco de dados do Microsoft Access a uma fonte de dados externa, como o Microsoft SQL Server.

O Adabas é um banco de dados muito consolidado e de alto desempenho que pode ser executado em todas as plataformas de servidor mais populares com baixo custo de propriedade. Ele fornece conectividade com aplicativos de todas as principais plataformas de desenvolvimento e conectividade de dados com ferramentas de dados conhecidas. O Adabas fornece todos os recursos de administração, monitoramento e segurança necessários para um sistema de banco de dados corporativo.



Um banco de dados do Adabas pode conter vários arquivos ou tabelas. O Adabas não é um banco de dados relacional, o que pode ser um benefício significativo quando se trata de desempenho e de estrutura de registro flexível.

Neste exemplo você pode ver um layout de registro com campos para o arquivo "cruises".

Periodic Group (Sailors)			
			Multiple Value Field
Cruise-ID	Start-harbor	Sailors(*) Name	Sailors (*) Languages(*)
5192	ELBA	MANN, GEOFF (1)	ENG (1,1) SPA (1,2)
		MANN, LINDA (2)	ENG (2,1) FRE (2,2)
		MARTINES, ALFREDO (3)	SPA (3,1) FRE (3,2)
		HERNANDO, EUGENIO (4)	SPA (4,1) FRE (4,2) ITA (4,3)

Um registro do arquivo "cruises" pode conter campos como "Cruise-ID", "Start-harbor", "Sailors-name" e "Sailors-languages". Para dados que tem relação entre si podemos criar um grupo periódico (Periodic Group). No nosso exemplo, cada tripulante pode falar vários idiomas, então criamos um campo multivalorado "Sailors-languages" que pode conter vários idiomas para cada tripulante.

Em um banco de dados relacional, esse tipo de informação precisaria ser normalizada e provavelmente distribuída em três tabelas diferentes. No Adabas, isso pode se encaixar em um único tipo de registro, portanto não há necessidade de **"joins"** ao recuperar ou salvar os dados. O Adabas também compactará automaticamente os campos vazios antes que o registro seja armazenado.

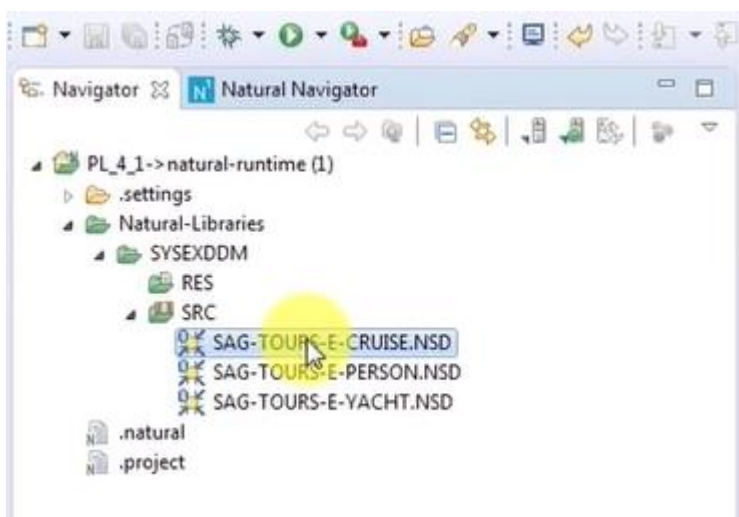
O Adabas suporta tipos de campo padrão como "alphanumeric" que vêm em 3 tipos diferentes. Pequeno, médio e grande de até 2 gigabytes. Essas "Strings" de caracteres longas são especialmente úteis para imagens, documentos em PDF ou vídeos. Há também um tipo especial de campo para alfabéticos e unicode. Para campos numéricos as opções são binário, inteiro, ponto flutuante e decimais compactados ou descompactados.

Format	Format Description	Maximum Length
A	Alphanumeric - With Long Alphanumeric (LA) option - With Large Object (LB) option	253 bytes ~16KB ~2GB
B	Binary	126 bytes
F	Fixed point (Integer)	8 bytes (always exactly 2, 4, or 8 bytes)
G	Floating point	8 bytes (always exactly 4 or 8 bytes)
P	Packed decimal (~2 digits per byte)	15 bytes
U	Unpacked decimal (~1 digit per byte)	29 bytes

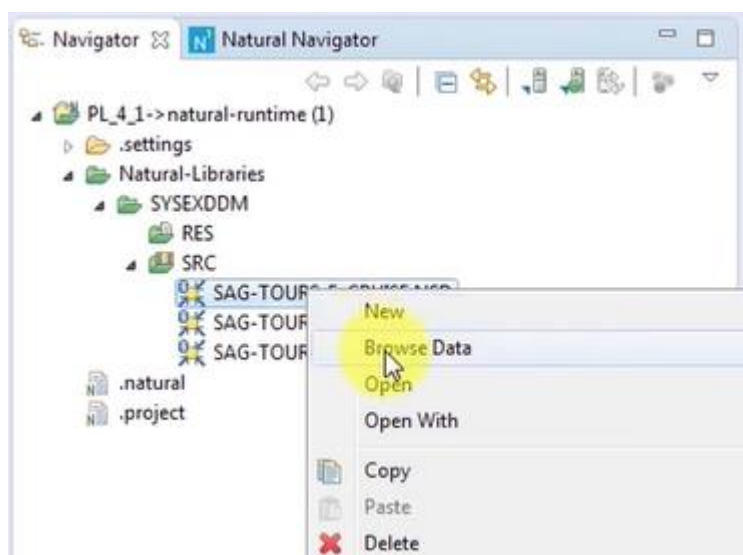
Vamos ver como isso fica no NaturalOne da perspectiva de um desenvolvedor. As visualizações dos arquivos ou tabelas do Adabas são definidas nas DDM's (data definition module).

DDM: O módulo de definição de dados contém informações sobre os campos individuais do arquivo. Essas informações são relevantes para o uso desses campos em um programa Natural. Um DDM constitui uma visão lógica de um arquivo de banco de dados físico.

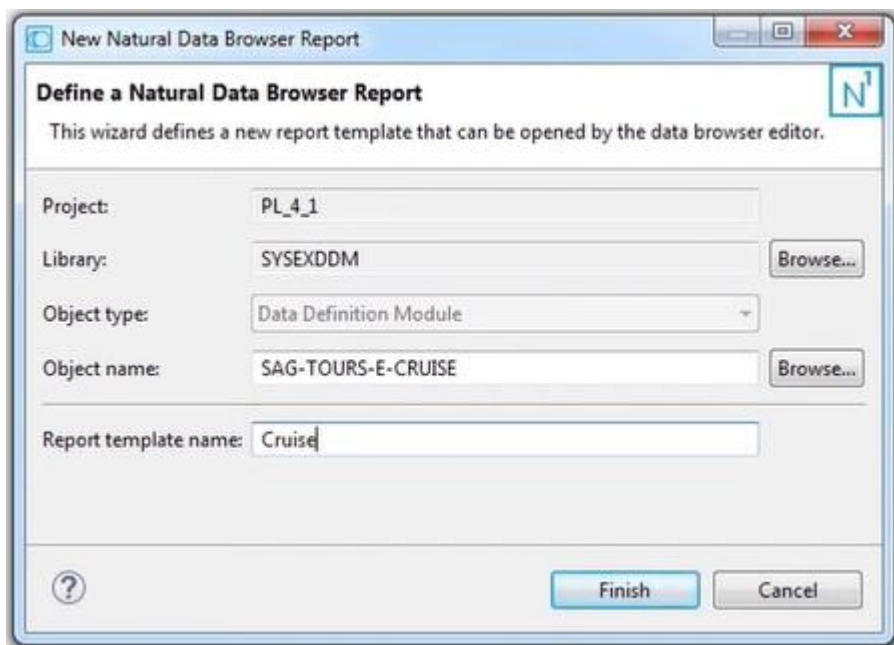
Temos três exemplos aqui:



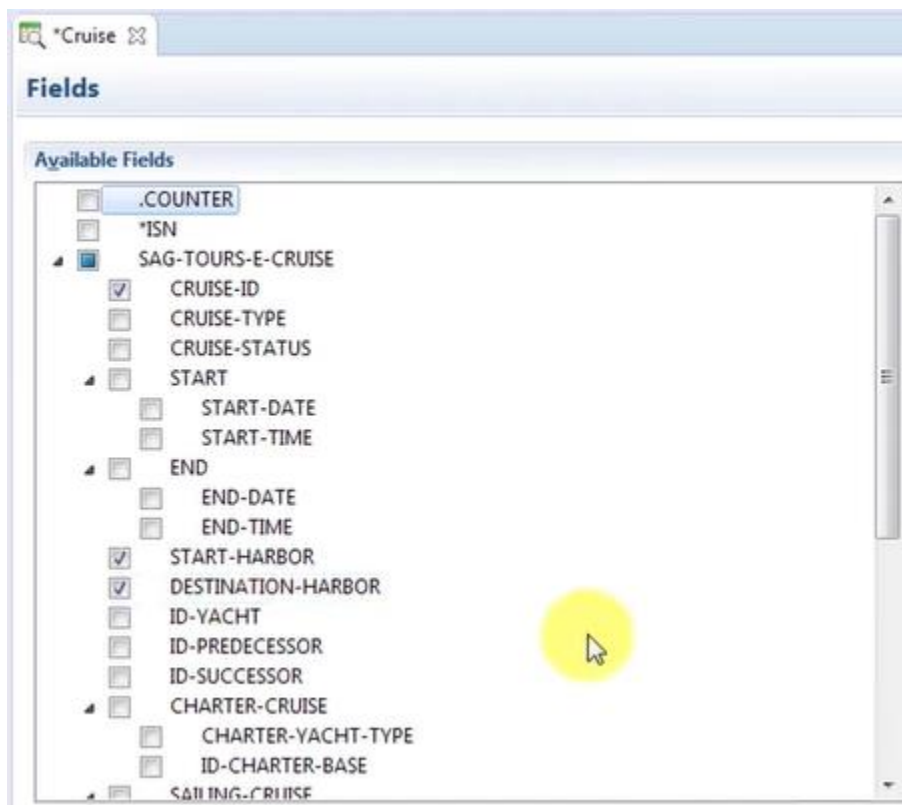
Podemos dar uma olhada nelas e nos dados que elas contém ao clicar com o botão direito e selecionar "browse data".



O relatório que estamos criando precisa de um nome como "Cruise".



Podemos seleccionar os campos relevantes. Para este relatório simples, escolhemos "CRUISE-ID", "START-HARBOR", "DESTINATION-HARBOR", "S-NAME" e "LANGUAGES".



Selected Fields

- CRUISE-ID
- START-HARBOR
- DESTINATION-HARBOR
- S-NAME (1-1)
- LANGUAGES (1-1, 1-1)

Na guia "Report", podemos selecionar a ordem na qual queremos que nossos dados apareçam.

SAILOR (*)

- ☐ AGE (*)
- ☒ S-NAME (*)
- ☐ EXPERIENCE (*)
- ☒ LANGUAGES (*, *)
- ☐ ID-CONTRACT (*)
- ☐ CRUISE-TIMESTAMP
- ☐ CRUISE-TYPE-SP
- ☐ CHARTER-PRICES-SP

Additional DDM Fields

Fields **Report**

Console Problems Tasks Details Code Coverage

Report

Filter

Field: >> Physical Read <<

Start value:

End value:

Maximum results: 100

Options

Edit mask: None

Field header: None

Report Information

Description: Report for DDM SAG-TOURS-E-CRUISE

Created at: 18. december 2018 11:18:39

Neste caso escolheremos "START-HARBOR" e gostaríamos de partir de um lugar ensolarado como as "Bahamas", e clicamos no botão de execução.

Report

Filter

Field: START-HARBOR

Start value: BAHAMAS

End value:

Maximum results: 100

Options

Edit mask: None

Field header: None

Report Information

Description: Report for DDM SAG-TOURS-E-CRUISE

Created at: 18. december 2018 11:18:39

DDM and Server Information

DDM: /PL_4_1/Natural-Libraries/SYSEXDDM/SRC/SAG-TOURS-E-CRUISE.NSC

DDM name: SAG-TOURS-E-CRUISE

Host name: localhost

Port number: 2800

Actions

Calculate Report: not requested

Create Report

Aqui temos o relatório com a lista de cruzeiros com os campos que selecionamos.

SAG-TOURS-E-CRUISE (Cruise) 11:19:13				
CRUISE-ID	START-HARBOR	DESTINATION-HARBOR	S-NAME[1:1]	LANGUAGES[1:1, 1:1]
5044	BAHAMAS	ST. CROIX	[1]	[1,1]
5070	BAHAMAS	MIAMI	[1] SUZANNE DURAND	[1,1] FRE
5086	BAHAMAS	VIRGIN ISLANDS	[1]	[1,1]
5085	BAHAMAS	BAHAMAS	[1] HARRIET ESSEX	[1,1] ENG
5068	BAHAMAS	BAHAMAS	[1] GEORGES ROLLET	[1,1] FRE
5069	BAHAMAS	BAHAMAS	[1] JEAN-MICHEL PICAMOLES-REBERGA	[1,1] FRE
5029	BARBADOS	ST. LUCIA	[1]	[1,1]
5102	BODRUM	KUSADASI	[1] FRANCOISE HSIAO	[1,1] ENG
5123	BODRUM	HERAKLEION	[1] SALIA BARES	[1,1] GER
5082	BUENOS AIRES	PUNTA ARENAS	[1] FERNANDEZ, ELOY	[1,1] SPA
5081	BUENOS AIRES	BUENOS AIRES	[1] MARION SCHOTT	[1,1] ENG
5008	CADIZ	VILAMOURA	[1]	[1,1]
5176	CAGLIARI	OLBIA	[1] JACQUELINE YACKX-COLTEAU	[1,1] FRE
5135	CHANIA	SPETSAI	[1]	[1,1]
5084	CHARLESTON	BAHAMAS	[1] TREVOR EAVES	[1,1] ENG
5179	CHIAVARI	SAN REMO	[1]	[1,1]

Type	Description
Descriptor	Standard index for one elementary field or a field in a Multiple Value / Periodic Group.
Hyperdescriptor	Based on a User-defined algorithm.
Phonetic Descriptor	A descriptor based on the "sound" of an alphanumeric field.
Subdescriptor	A subdescriptor is part of a single field.
Superdescriptor	Combines key-values from all or parts of 2-20 fields.

O nome de uma chave ou índice no Adabas é chamado de "descriptor". É possível adicionar descritores a todos os tipos de campos, incluindo os "grupos periódicos" ou "campos multivalorados". Isso significa, por exemplo, que adicionar um descriptor aos idiomas dos tripulantes nos permitiria procurar por todos os tripulantes que falam espanhol.

O "hiperdescriptor" permite endereçar eventuais requisitos especiais em relação aos descritores definidos pelo usuário.

O "descriptor fonetico" é muito útil para pesquisas de nome em que o som do campo alfanumérico é mais relevante do que a ortografia.

O "subdescriptor" ou "superdescriptor" permite pesquisar partes de um campo ou combinar partes de vários campos.

2 - Acesso e Consulta

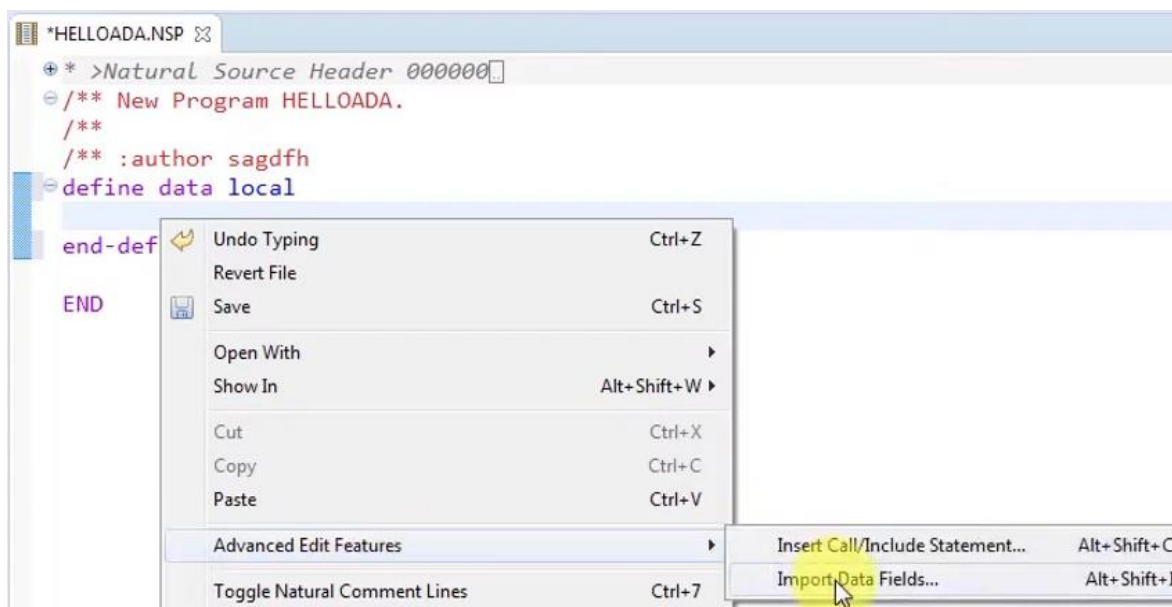
Esta unidade é uma breve introdução ao acesso a dados do Adabas através do Natural. Estas são as 4 instruções básicas do Natural para acessar dados no Adabas.

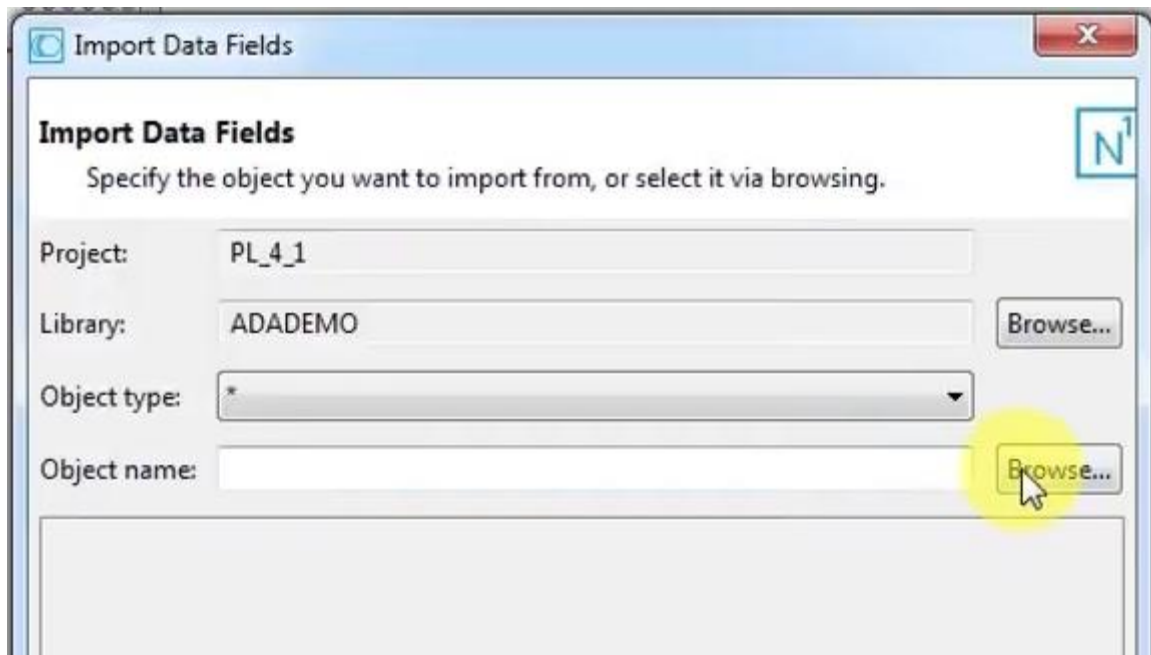
Natural Database Access Statements
READ
FIND
GET
HISTOGRAM

"Read" para acesso sequencial, "Find" para selecionar um conjunto de registros, "Get" para obter um registro usando uma chave interna do Adabas e finalmente "Histogram" para uma contagem de registros com valores de chaves específicas.

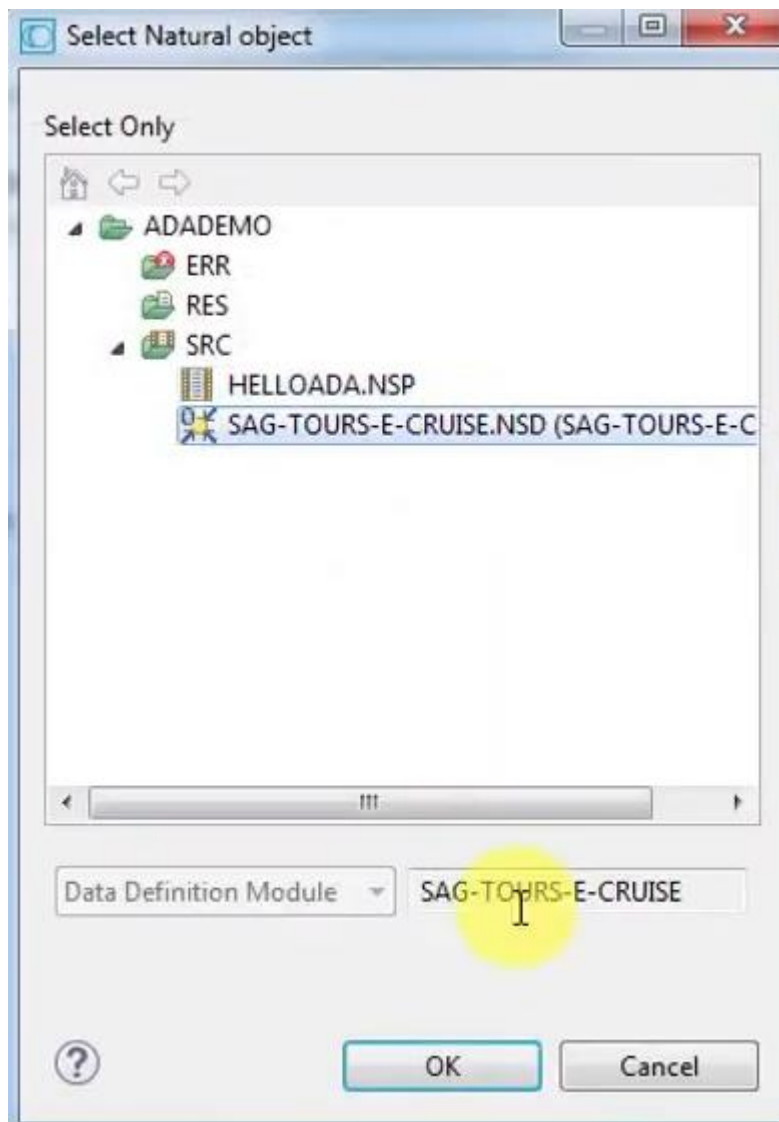
Primeiro vamos falar sobre a instrução "Read" que é uma maneira rápida de ler diversos registros sequencialmente.

Mas antes de podermos acessar os dados, precisaremos definir a visão dos dados que queremos analisar. Assim vamos usar "define data local" e o "end-define". Agora vamos clicar com o botão direito para acessar "Advanced Edit Features" em seguida "Import Data Fields".





Neste caso da DDM "Cruise".



Vamos escolher o nível superior “Sag-tours-e-cruise”, “Cruise-id”, “Start-date”, “Start-harbor” e “Destination-harbor”.

Import Data Fields

Select data field(s) to import

Project: PL_4_1

Library: ADADEMO Browse...

Object type: Data Definition Module

Object name: SAG-TOURS-E-CRUISE Browse...

- ☒ SAG-TOURS-E-CRUISE
 - ☒ CRUISE-ID (N8.0)
 - ☐ CRUISE-TYPE (A1)
 - ☐ CRUISE-STATUS (A1)
 - ☐ START
 - ☒ START-DATE (N8.0)
 - ☐ START-TIME (N6.0)
 - ☐ END
 - ☐ END-DATE (N8.0)
 - ☐ END-TIME (N6.0)
 - ☒ START-HARBOR (A20)
 - ☒ DESTINATION-HARBOR (A20)
 - ☐ ID-YACHT (N8.0)
 - ☐ ID-PREDECESSOR (N8.0)

Select All Deselect All

☐ Select all parent fields

☐ Select all child fields

? OK Cancel

Vamos mudar o nome padrão da nossa visão local para “cruise”.

```
*HELLOADA.NSP
+* >Natural Source Header 000000
-/** New Program HELLOADA.
/**
/** :author sagdfh
-define data local
/*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
-1 SAG-TOURS-E-CRUISE VIEW OF SAG-TOURS-E-CRUISE
2 CRUISE-ID (N8.0)
2 START-DATE (N8.0)
2 START-HARBOR (A20)
2 DESTINATION-HARBOR (A20)
/*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

end-define

END
```

```
*HELLOADA.NSP
+* >Natural Source Header 000000
-/** New Program HELLOADA.
/**
/** :author sagdfh
-define data local
/*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
-1 cruise VIEW OF SAG-TOURS-E-CRUISE
2 CRUISE-ID (N8.0)
2 START-DATE (N8.0)
2 START-HARBOR (A20)
2 DESTINATION-HARBOR (A20)
/*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

end-define

END
```


Agora vamos criar um loop de leitura (read) onde especificamos a visão que no nosso caso é “cruise”. Dentro do loop queremos uma instrução então vamos dizer “display cruise”.

```
*HELLOADA.NSP
+ * >Natural Source Header 000000
- /** New Program HELLOADA.
  /**
  /** :author sagdfh
- define data local
  /*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
- 1 cruise VIEW OF SAG-TOURS-E-CRUISE
    2 CRUISE-ID (N8.0)
    2 START-DATE (N8.0)
    2 START-HARBOR (A20)
    2 DESTINATION-HARBOR (A20)
  /*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

  end-define
- read cruise |
    display cruise
  end-read
END
```

Esse loop simples seria suficiente para percorrer todos os registros do arquivo “cruise” no banco de dados. Mas vamos restringir um pouco por meio da inclusão de uma chave de um valor inicial. Então vamos ler o “cruise” pelo descritor “Start-harbor” com o valor “Bahamas”.

```
*HELLOADA.NSP
+ * >Natural Source Header 000000
- /** New Program HELLOADA.
  /**
  /** :author sagdfh
- define data local
  /*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
- 1 cruise VIEW OF SAG-TOURS-E-CRUISE
    2 CRUISE-ID (N8.0)
    2 START-DATE (N8.0)
    2 START-HARBOR (A20)
    2 DESTINATION-HARBOR (A20)
  /*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

  end-define
- read cruise by start-harbor = 'BAHAMAS'
    display cruise
  end-read
END
```


Ao executar teremos uma lista de todos os cruzeiros que partem das "Bahamas" no topo da lista.

Natural Web I/O Output

Page 1

18-12-20 12:35:07

CRUISE

CRUISE-ID	START-DATE	START-HARBOR	DESTINATION-HARBOR
-----------	------------	--------------	--------------------

-----	-----	-----	-----
-------	-------	-------	-------

5044	20181103	BAHAMAS	ST. CROIX
5070	20190412	BAHAMAS	MIAMI
5086	20190111	BAHAMAS	VIRGIN ISLANDS
5085	20181221	BAHAMAS	BAHAMAS
5068	20190315	BAHAMAS	BAHAMAS
5069	20190329	BAHAMAS	BAHAMAS
5029	20181208	BARBADOS	ST. LUCIA
5102	20200415	BODRUM	KUSADASI
5123	20200812	BODRUM	HERAKLEION
5082	20181222	BUENOS AIRES	PUNTA ARENAS
5081	20181208	BUENOS AIRES	BUENOS AIRES
5008	20190118	CADIZ	VILAMOURA
5176	20190614	CAGLIARI	OLBIA
5135	20200610	CHANIA	SPETSAI
5084	20181207	CHARLESTON	BAHAMAS
5179	20190726	CHIAVARI	SAN REMO

MORE

Vamos restringir ainda mais adicionando um valor final. Vamos dizer “ending at ‘CADIZ’”.

```
HELLOADA.NSP  ⓘ
+ * >Natural Source Header 000000
- /** New Program HELLOADA.
  /**
  /** :author sagdfh
- define data local
  /*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
- 1 cruise VIEW OF SAG-TOURS-E-CRUISE
    2 CRUISE-ID (N8.0)
    2 START-DATE (N8.0)
    2 START-HARBOR (A20)
    2 DESTINATION-HARBOR (A20)
  /*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

  end-define
- read cruise by start-harbor = 'BAHAMAS' ending at 'CADIZ'
  display cruise
end-read
END
```

Ao executar teremos uma lista de resultados ainda menor.

Natural Web I/O Output

Page 1 18-12-20 12:46:27

CRUISE

CRUISE-ID	START-DATE	START-HARBOR	DESTINATION-HARBOR
5044	20181103	BAHAMAS	ST. CROIX
5070	20190412	BAHAMAS	MIAMI
5086	20190111	BAHAMAS	VIRGIN ISLANDS
5085	20181221	BAHAMAS	BAHAMAS
5068	20190315	BAHAMAS	BAHAMAS
5069	20190329	BAHAMAS	BAHAMAS
5029	20181208	BARBADOS	ST. LUCIA
5102	20200415	BODRUM	KUSADASI
5123	20200812	BODRUM	HERAKLEION
5082	20181222	BUENOS AIRES	PUNTA ARENAS
5081	20181208	BUENOS AIRES	BUENOS AIRES
5008	20190118	CADIZ	VILAMOURA

MORE

Agora vamos alterar este exemplo de “Read” para uma instrução “Find”. Vamos alterar “Read” e “And-read” para “Find” e “And-find”. Alterar “By” para “With” e remover o valor final. A grande diferença entre as instruções “Read” e “Find” é que “Read” pode ter apenas um critério pois percorre os dados sequencialmente, enquanto uma instrução “Find” pode ter vários critérios de pesquisa para selecionar o conjunto de resultados. Assim podemos adicionar outro critério de pesquisa. Então adicionaremos “and Start-data > 20190101”. A última coisa que quero fazer é adicionar a “*isn” que é a chave única de um registro.

```

HELLOADB.NSP  HELLOADA.NSP
+ * >Natural Source Header 000000
- /** New Program HELLOADB.
  /**
  /** :author sagdfh
- define data local
  /*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
- 1 cruise VIEW OF SAG-TOURS-E-CRUISE
    2 CRUISE-ID (N8.0)
    2 START-DATE (N8.0)
    2 START-HARBOR (A20)
    2 DESTINATION-HARBOR (A20)
  /*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

  end-define
- find cruise with start-harbor = 'BAHAMAS' AND start-date >20190101
  display cruise *isn
end-find
END

```

Ao executar, vemos os poucos registros que correspondem aos dois critérios de pesquisa.

Natural Web I/O Output				
Page	1	18-12-20 15:24:27		
CRUISE				ISN
CRUISE-ID	START-DATE	START-HARBOR	DESTINATION-HARBOR	
5070	20190412	BAHAMAS	MIAMI	2050
5086	20190111	BAHAMAS	VIRGIN ISLANDS	2419
5068	20190315	BAHAMAS	BAHAMAS	3940
5069	20190329	BAHAMAS	BAHAMAS	4457

Você deve ter notado que no nosso exemplo “Read” os registros foram classificados em ordem alfabética pelo valor da chave usada. Porém, a instrução find permite várias chaves, o que significa que os resultados são por padrão sempre classificados pela “isn”. Aqui, “2050” é a menor.

Vamos agora ver como obter um único registro se já temos disponível a “isn” como, por exemplo, 2050. Assim removeremos “Find” e “End-find” e escreveremos “Get Cruise 2050”.

```
HELLOADC.NSP
+ * >Natural Source Header 000000
- /** New Program HELLOADC.
  /**
  /** :author sagdfh
- define data local
  /*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
- 1 cruise VIEW OF SAG-TOURS-E-CRUISE
    2 CRUISE-ID (N8.0)
    2 START-DATE (N8.0)
    2 START-HARBOR (A20)
    2 DESTINATION-HARBOR (A20)
  /*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

  end-define
  get cruise 2050
    display cruise *isn

  END
```

Ao executar o programa temos:

Natural Web I/O Output					
Page	1	18-12-20 15:41:39			
CRUISE			ISN		
CRUISE-ID	START-DATE	START-HARBOR	DESTINATION-HARBOR		

5070	20190412	BAHAMAS	MIAMI	2050	

No nosso último exemplo veremos como acessar o contador das ocorrências de cada descritor ou índice. Isso é muito semelhante a uma instrução “select count” no SQL. Essa visualização só pode funcionar com a variável do descritor, então removeremos os três campos adicionais. Vamos mudar para “histogram cruise start-harbor”. Em seguida, “display start-harbor *number” para mostrar o número de ocorrências. Fechamos o loop com “and-histogram”.

```
HELLOADD.NSP  ⌕
+ * >Natural Source Header 000000...
- /** New Program HELLOADC.
  /**
  /** :author sagdfh
- define data local
  /*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
- 1 cruise VIEW OF SAG-TOURS-E-CRUISE

    2 START-HARBOR (A20)

  /*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

  end-define
- histogram cruise start-harbor
  display start-harbor *number
  end-histogram
END
```

Ao executarmos o programa podemos ver o número de ocorrências de cada “start-harbor”.

Natural Web I/O Output	
Page	1
	18-12-20 16:18:44
START-HARBOR	NMBR

	2831
ACAPULCO	1
ADEN	1
AEGINA	1
ALGHERO	2
ALICANTE	2
ALMERIA	1
ANDRAITX	1
ANNAPOLIS	385
ANTIGUA	430
AQUILEIA	1
ATHEN	4
AZOREN	3
BAHAMAS	6
BARBADOS	1
BODRUM	2
BUENOS AIRES	2
CADIZ	1
MORE	

Resumindo, “Read” é usado para acesso sequencial, “Find” para selecionar um conjunto de registros, “Get” para um único registro e “Histogram” para o número de ocorrências dos valores do descritor.

Statement	Description	Example
READ	Reads records sequentially from start-value to end-of-file or to a specified end-value	READ cruise BY start-harbor = 'BAHAMAS' ENDING AT = 'CADIZ' DISPLAY cruise END-READ
FIND	Selects a set of records based on one or more criteria and returns one at a time in the loop	FIND cruise WITH start-harbor = 'BAHAMAS' AND start-date > 20190130 DISPLAY cruise END-FIND
GET	Returns one record based on ISN	GET cruise
HISTOGRAM	Returns number of occurrences for each index value	HISTOGRAM cruise start-harbor DISPLAY start-harbor *number END-HISTOGRAM

3 - Atualizações e Transações

Este vídeo é uma introdução rápida a atualização dos dados do Adabas através do Natural. A unidade anterior cobriu o acesso a dados no Adabas. Para atualizar os dados é possível usar uma das seguintes instruções.

Statements
STORE
UPDATE
DELETE
END TRANSACTION
BACKOUT TRANSACTION

Temos o “store” para adicionar um novo registro e “update” ou “delete” para alterar ou excluir registros existentes. Se as alterações não forem confirmadas usando o “end-transaction” ou explicitamente descartadas por um “backout transaction” as alterações no banco de dados serão desfeitas quando o tempo limite “time out” for atingido. Vejamos essas instruções mais de perto.

Para armazenar um novo registro precisamos primeiro inserir alguns valores nos campos da visão. Como você pode ver, já adicionei alguns valores, como “cruise-id := 5202” e assim por diante. Assim, tudo que precisamos é uma instrução “store” contendo o nome da visão local, neste caso, “cruise”. E por fim precisamos confirmar as alterações usando o “end-transaction”. Eu também adicionei um pequeno bloco “read display” para que possamos verificar se os novos dados foram realmente adicionados.

*HELLOADE.NSP

```
* >Natural Source Header 000000
/** New Program HELLOADE.
/**
/** :author sagdfh
define data local
/*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
1 cruise VIEW OF SAG-TOURS-E-CRUISE
2 CRUISE-ID (N8.0)
2 START-DATE (N8.0)
2 START-HARBOR (A20)
2 DESTINATION-HARBOR (A20)
/*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

end-define
CRUISE-ID           := 5202
START-DATE          := 20190520
START-HARBOR        := 'BAHAMAS'
DESTINATION-HARBOR  := 'ADABAS'

store cruise
end transaction

read cruise by cruise-id = 5202
display cruise *isn
end-read

I

END
```

Vamos salvar e executar e poderemos ver o novo registro retornado do Adabas com uma nova “isn”.

Natural Web I/O Output				
Page 1		18-12-21 14:41:36		
CRUISE			ISN	
CRUISE-ID	START-DATE	START-HARBOR	DESTINATION-HARBOR	

5202	20190520	BAHAMAS	ADABAS	5154

Se quisermos alterar os valores e atualizar os registros, primeiro teremos que acessá-los. Podemos fazer isso usando uma instrução “find” como esta. Eu adicionei uma instrução “display” para que possamos ver os valores do registro antes de alterar qualquer coisa. E então temos que lembrar de confirmar a transação com “end-transaction”.

```
HELLOADE.NSP  HELLOADF.NSP
+ * >Natural Source Header 000000
- /** New Program HELLOADF.
  /**
  /** :author sagdfh
- define data local
  /*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
- 1 cruise VIEW OF SAG-TOURS-E-CRUISE
    2 CRUISE-ID (N8.0)
    2 START-DATE (N8.0)
    2 START-HARBOR (A20)
    2 DESTINATION-HARBOR (A20)
  /*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

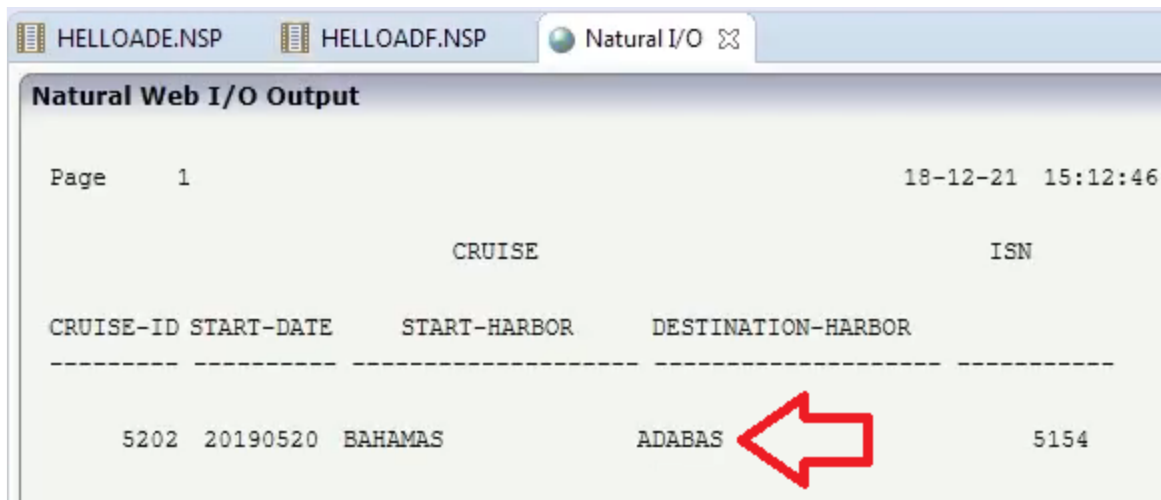
  end-define

- find cruise with cruise-id = 5202
    display cruise *isn
    DESTINATION-HARBOR := 'NATURAL'
    update

  end-find
end transaction

END
```

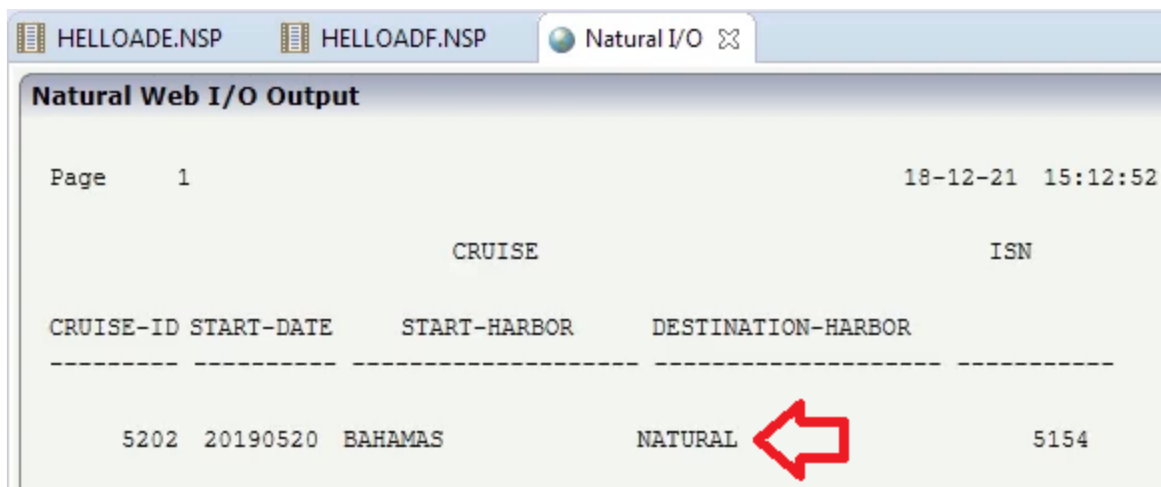
Ao salvamos e executamos, veremos que o valor de “destination-harbor” era “Adabas”.



The screenshot shows the 'Natural Web I/O Output' window. At the top, there are tabs for 'HELLOADE.NSP', 'HELLOADF.NSP', and 'Natural I/O'. The window displays 'Page 1' and the timestamp '18-12-21 15:12:46'. Below this, the word 'CRUISE' is centered. A table header follows, with columns: 'CRUISE-ID', 'START-DATE', 'START-HARBOR', 'DESTINATION-HARBOR', and 'ISN'. A dashed line separates the header from the data row. The data row contains the values: '5202', '20190520', 'BAHAMAS', 'ADABAS', and '5154'. A red arrow points to the 'ADABAS' value in the 'DESTINATION-HARBOR' column.

CRUISE-ID	START-DATE	START-HARBOR	DESTINATION-HARBOR	ISN
5202	20190520	BAHAMAS	ADABAS	5154

Para verificar se as alterações foram realmente feitas apenas, executaremos novamente o programa para exibir o valor atual que agora é “Natural”.



The screenshot shows the 'Natural Web I/O Output' window after a second execution. The tabs at the top remain the same. The window displays 'Page 1' and the timestamp '18-12-21 15:12:52'. The table structure is identical to the previous screenshot, but the value in the 'DESTINATION-HARBOR' column has changed to 'NATURAL'. A red arrow points to the 'NATURAL' value.

CRUISE-ID	START-DATE	START-HARBOR	DESTINATION-HARBOR	ISN
5202	20190520	BAHAMAS	NATURAL	5154

Digamos que queremos excluir este registro. Adicionei uma verificação extra, no caso do registro não ser encontrado. Então vamos adicionar a instrução “delete”. Para ilustrar como podemos cancelar uma transação, podemos adicionar a instrução “backout transaction”.

```
HELLOADG.NSP
* >Natural Source Header 000000
/** New Program HELLOADF.
/**
/** :author sagdfh
define data local
/*( imported data fields from Data Definition Module SAG-TOURS-E-CRUISE
1 cruise VIEW OF SAG-TOURS-E-CRUISE
2 CRUISE-ID (N8.0)
2 START-DATE (N8.0)
2 START-HARBOR (A20)
2 DESTINATION-HARBOR (A20)
/*) imported data fields from Data Definition Module SAG-TOURS-E-CRUISE

end-define

find cruise with cruise-id = 5202
if no records found
write 'cruise-id' cruise-id 'not found'
end-norec
display cruise *isn
delete

end-find
backout transaction

END
```

Salvamos e executamos. Executamos novamente e veremos que a exclusão nunca é efetivada.

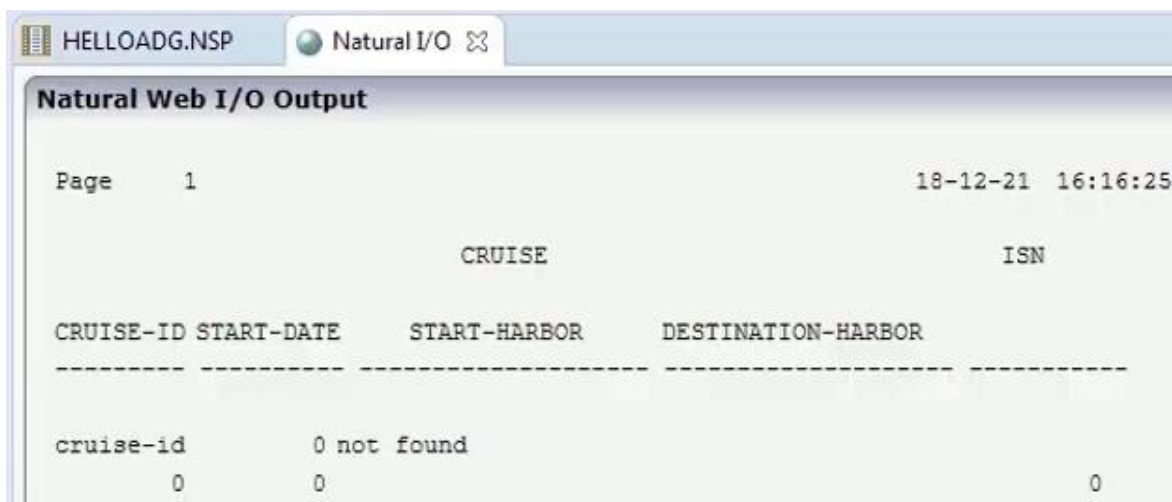
HELLOADG.NSP Natural I/O

Natural Web I/O Output

Page 1 18-12-21 16:16:11

CRUISE				ISN
CRUISE-ID	START-DATE	START-HARBOR	DESTINATION-HARBOR	
5202	20190520	BAHAMAS	NATURAL	5154

Então vamos alterar o “backout transaction” para “end transaction”. Salvamos e executamos. E executamos novamente. Agora podemos ver que o registro finalmente desapareceu.



Resumindo, para armazenar um registro, preencha os devidos valores e use a instrução “store” seguida pelo nome da visão. Lembre-se de confirmar usando “end transaction”.

Para atualização você precisa acessar o registro. Por exemplo, com uma instrução “find”, alterar os valores e atualizá-lo com “update”.

Para exclusão, você também precisa acessar um registro. Use a instrução “delete” e lembre-se da instrução “end transaction”. Ou no caso de um erro exigir o cancelamento da transação, a instrução “backout transaction”.

Statement	Description	Example
STORE	Adds a new record with the data in the view.	<pre> cruise-id := 5202 .. /* assign values to relevant fields in view STORE cruise END TRANSACTION </pre>
UPDATE	Updates a record after accessing it.	<pre> FIND cruise WITH cruise-id = 5202 destination-harbor := 'NATURAL' /* change relevant fields UPDATE END-FIND END TRANSACTION </pre>
DELETE	Deletes a record after accessing it.	<pre> FIND cruise WITH cruise-id = 5202 DELETE END-FIND END TRANSACTION </pre>
END/BACKOUT TRANSACTION	Commits or backs out a logical transaction.	<pre> END TRANSACTION BACKOUT TRANSACTION </pre>