ReStore Block Analyzer – Predicting the Future

John Aspinall

Freelance Smalltalk Developer

https://github.com/rko281

ReStore Block Analyzer – Predicting the Future

- **ReStore**
  - Object-Relational Interface (MySQL, Postgres, SQLite etc.)
  - Simplicity > Flexibility
  - Highly Transparent
  - Open Source
    - https://github.com/rko281/restore

- **Dolphin Smalltalk**
  - Windows-native Smalltalk
  - Deploys to Windows executable or DLL
  - Open Source
    - https://github.com/dolphinsmalltalk

ReStore Block Analyzer – Predicting the Future

- **Example – Customer and Orders**

```
Object subclass: #Customer
        instanceVariableNames: 'firstName surname emailAddress dateOfBirth address orders'
```

```
reStoreDefinition

        ^super reStoreDefinition
                define: #surname as: (String maxSize: 100);
                define: #firstName as: (String maxSize: 100);
                define: #emailAddress as: (String maxSize: 100);
                define: #dateOfBirth as: Date;
                define: #address as: Address dependent;
                define: #orders as: (OrderedCollection of: Order dependent owner: #customer);
                yourself.
```

ReStore Block Analyzer – Predicting the Future

- **Example – Creating the Database**

```
ReStore
    dsn: 'ReStoreTest';
    connect;
    addClasses: {Customer. Address. Order};
    synchronizeAllClasses
```

```
CREATE TABLE CUSTOMER (ID_ INTEGER PRIMARY KEY, SURNAME VARCHAR(100),
FIRST_NAME VARCHAR(100), EMAIL_ADDRESS VARCHAR(100), DATE_OF_BIRTH
DATE, ADDRESS INTEGER);

CREATE TABLE ADDRESS (ID_ INTEGER PRIMARY KEY, LINE1 VARCHAR(100),
POSTCODE VARCHAR(16), COUNTRY VARCHAR(100));

CREATE TABLE ORDER_TABLE (ID_ INTEGER PRIMARY KEY, PRODUCT
VARCHAR(100), QUANTITY INTEGER, CUSTOMER INTEGER);
```

ReStore Block Analyzer – Predicting the Future

- **Example – Creating an Object**

```
customer := Customer new.
customer
        surname: 'Smith';
        firstName: 'John';
        emailAddress: 'john.smith@somewhere.net';
        address: (Address new
                        line1: '123 Oxford Street';
                        postcode: 'W1 1AA';
                        country: 'UK';
                        yourself);
        commit.
```

```
INSERT INTO CUSTOMER (SURNAME, FIRST_NAME, EMAIL_ADDRESS, ADDRESS, ID_)
VALUES ('Smith', 'John', 'john.smith@somewhere.net', 1, 1);

INSERT INTO ADDRESS (LINE1, POSTCODE, COUNTRY, ID_)
VALUES ('123 Oxford Street', 'W1 1AA', 'UK', 1);
```

ReStore Block Analyzer – Predicting the Future

- **Example – Updating an Object**

```
customer
        emailAddress: 'john.smith@somewhereelse.com';
        addOrder: (Order new product: 'widgets'; quantity: 4; yourself);
        commit.
```

```
UPDATE CUSTOMER SET EMAIL_ADDRESS = 'john.smith@somewhereelse.com'
WHERE CUSTOMER.ID_ = 1;

INSERT INTO ORDER_TABLE (PRODUCT, QUANTITY, CUSTOMER, ID_)
VALUES ('widgets', 4, 1, 1);
```

ReStore Block Analyzer – Predicting the Future

- **Example – Reading Objects**

```
customers := Customer storedInstances.

customers size.

customers first.

customers asOrderedCollection.
```

ReStore Block Analyzer – Predicting the Future

- **Example – Querying Objects**

```
customers := Customer storedInstances.

customers select: [ :each | each surname = 'Smith'].

customers select: [ :each | each address country = 'UK'].

customers select: [ :each | each orders anySatisfy:
                        [ :order | order product = 'widgets']].
```

ReStore Block Analyzer – Predicting the Future

- **Query Block Analysis**

```
customers select: [ :each | each address country = 'UK'].
```

```
BlockAnalyzer doesNotUnderstand: #address
BlockAnalyzer doesNotUnderstand: #country
BlockAnalyzer = 'UK'

SELECT * FROM CUSTOMER
LEFT JOIN ADDRESS ON ADDRESS.ID_ = CUSTOMER.ADDRESS
WHERE ADDRESS.COUNTRY
 = 'UK'
```

ReStore Block Analyzer – Predicting the Future

- **A Small Problem**

```
customers select: [ :each | each address country isNil].
```

```
BlockAnalyzer doesNotUnderstand: #address
BlockAnalyzer doesNotUnderstand: #country
???

SELECT * FROM CUSTOMER
LEFT JOIN ADDRESS ON ADDRESS.ID_ = CUSTOMER.ADDRESS
WHERE ADDRESS.COUNTRY
???
```

```
customers select: [ :each | each address country = nil].
```

ReStore Block Analyzer – Predicting the Future

- **A Solution**

```
customers select: [ :each | each address country isNil].
```

```
4    Push Temp[0]; Send[1]: #address with 0 args
6    Send[2]: #country with 0 args
7    Special Send #isNil
8    Return From Block
```

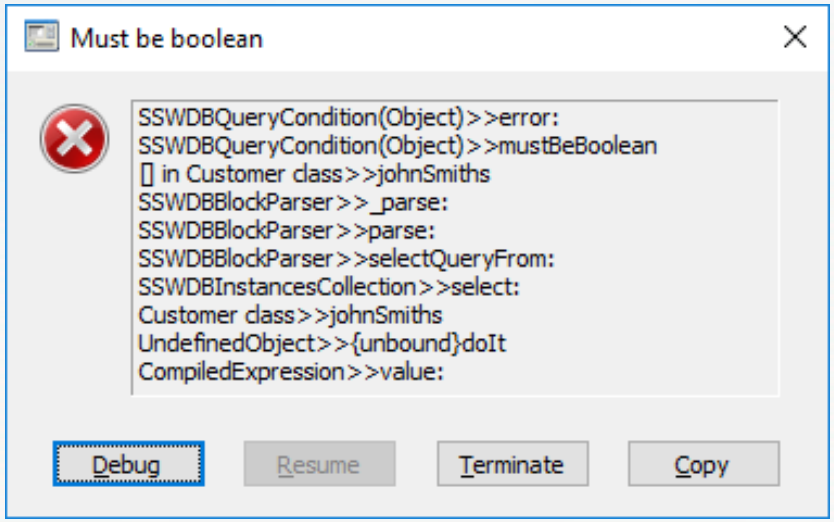
```
doesNotUnderstand: aMessage

    self convertMessageToSQL: aMessage.

    senderFrame nextBytecode = SpecialSendIsNil ifTrue:
        [self = nil.
        senderFrame skipNextBytecode]
```

```
WHERE ADDRESS.COUNTRY IS NULL
```

ReStore Block Analyzer – Predicting the Future

- **Another Problem**

```
[ :each | each firstName = 'John' and: [each surname = 'Smith']]
        "SQLCondition, not a boolean"
```




```
[ :each | (each firstName = 'John') & (each surname = 'Smith')]
```

ReStore Block Analyzer – Predicting the Future

- **Another Solution**

```
[ :each | each firstName = 'John' and: [each surname = 'Smith']]
```

```
5    Push Temp[0]; Send[1]: #firstName with 0 args
7    Push Const[2]: 'John'
8    Special Send #=
9    Jump If False +5 to 15
10   Push Temp[0]; Send[4]: #surname with 0 args
12   Push Const[5]: 'Smith'
13   Special Send #=
14   Return From Block
15   Push false
16   Return From Block
```

mustBeBoolean

```
"Examine sender's bytecodes"
"Determine and: / or: "

lastBytecode = JumpIfFalse ifTrue: [^true].
lastBytecode = JumpIfTrue ifTrue: [^false].

self error: 'unexpected bytecode'
```

```
WHERE FIRST_NAME = 'JOHN' AND  SURNAME
```

ReStore Block Analyzer – Predicting the Future

- **Yet Another Problem and Solution**

```
[ :each | each address postcode isNil or: [each address country isNil]]

    5    Push Temp[0]; Send[1]: #address with 0 args
    7    Send[2]: #postcode with 0 args
    8    ~~Jump If Not Nil +2 to 12~~ Jump If False +2 to 12   → mustBeBoolean
   10    Push true
   11    Return From Block
   12    Push Temp[0]; Send[1]: #address with 0 args
   14    Send[4]: #country with 0 args
```

```
doesNotUnderstand: aMessage

    self convertMessageToSQL: aMessage.
    self handleIsNilBytecodes ifTrue: [^self].

    senderFrame nextBytecode = JumpIfNotNil ifTrue:
        [senderFrame replaceNextBytecodeWith: JumpIfFalse.
        ^self = nil "SQLCondition, not a boolean"].
```

```
WHERE ADDRESS.POSTCODE IS NULL OR ADDRESS
```

ReStore Block Analyzer – Predicting the Future

…though it's a bit more complicated than that:

- isNil and: […]
- isNil or: […]
- notNil and: […]
- notNil or: […]

```
"There are 8 known possibilities to consider when deciding if this is an isNil or notNil test:

- JumpIfNil to ^True = isNil
- JumpIfNil to ^False = notNil
- JumpIfNil to other bytecode with ^True in the jumped-over code = notNil
- JumpIfNil to other bytecode without ^True in the jumped-over code = isNil

- JumpIfNotNil to ^True = notNil
- JumpIfNotNil to ^False = isNil
- JumpIfNotNil to other bytecode with ^True in the jumped-over code = isNil
- JumpIfNotNil to other bytecode without ^True in the jumped-over code = notNil"
```

ReStore Block Analyzer – Predicting the Future

- **An Included Bonus**

```
customers select: [ :each | #('UK' 'France') includes: each address country]
```

```
doesNotUnderstand: aMessage

    "Do everything we did earlier"

    senderFrame nextMessage = #includes: ifTrue:
        [self includedIn: senderFrame nextMessageReceiver.
        senderFrame skipNextBytecode]
```

```
WHERE ADDRESS.COUNTRY  IN ('UK' 'France')
```