Abstract

In this talk we describe the inbuilt tables that comprise the OpenEdge database schema. These are the "metadata" tables that describe the logical and physical structure of a database.

Unlike the Virtual System Tables (aka VST's) which exist only in memory, the schema tables are real tables just like your own. Well, almost.

Topics include logical structure, physical structure, what schema tables exist, how they are related, how they are used, and example queries showing how to produce simple reports about the database.

PUG CHALLENGE AMERICAS EXCHANGE

Secrets Of The OpenEdge RDBMS:
Schema Tables

Gus Björklund, Ronin

PROGRESS BravePoint

Notices

- Please ask questions as we go
- Tell us when we are wrong

About the 4GL examples …

Good artists copy, great artists steal.

-- Pablo Picasso

- Steal the code herein !

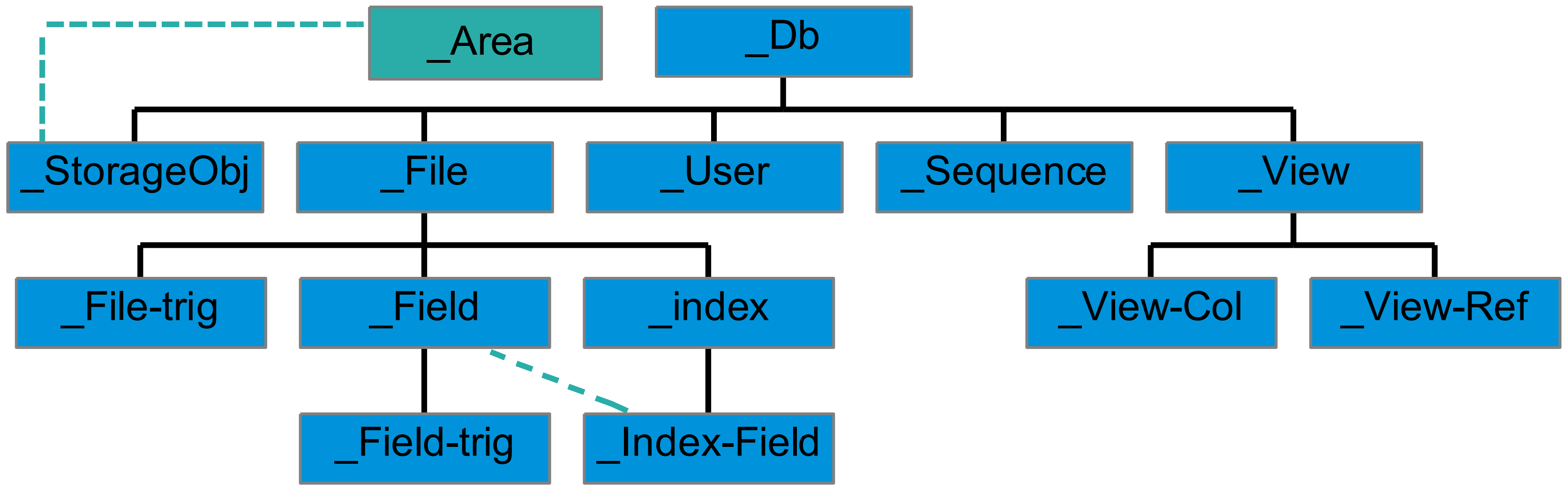
What are schema tables?

- Schema tables are metadata: data about data
- Real tables, stored in the database
- Describe what is stored in the database
  - Tables, indexes, sequences, views, etc., including schema tables
- Describe database physical storage
  - Areas, extents, transaction logs
- Describe users and access rights
- 4GL and SQL share *most* schema tables
- SQL has additional “catalog tables” and views

Virtual System Tables (aka VST’s)

- Are NOT schema tables
- NOT real tables
- VST data are NOT stored anywhere
- VST records are created “on the fly” when retrieved
- VST tables describe database status and activity
- The schema *does* contain descriptions of VST’s

The Schema Tables



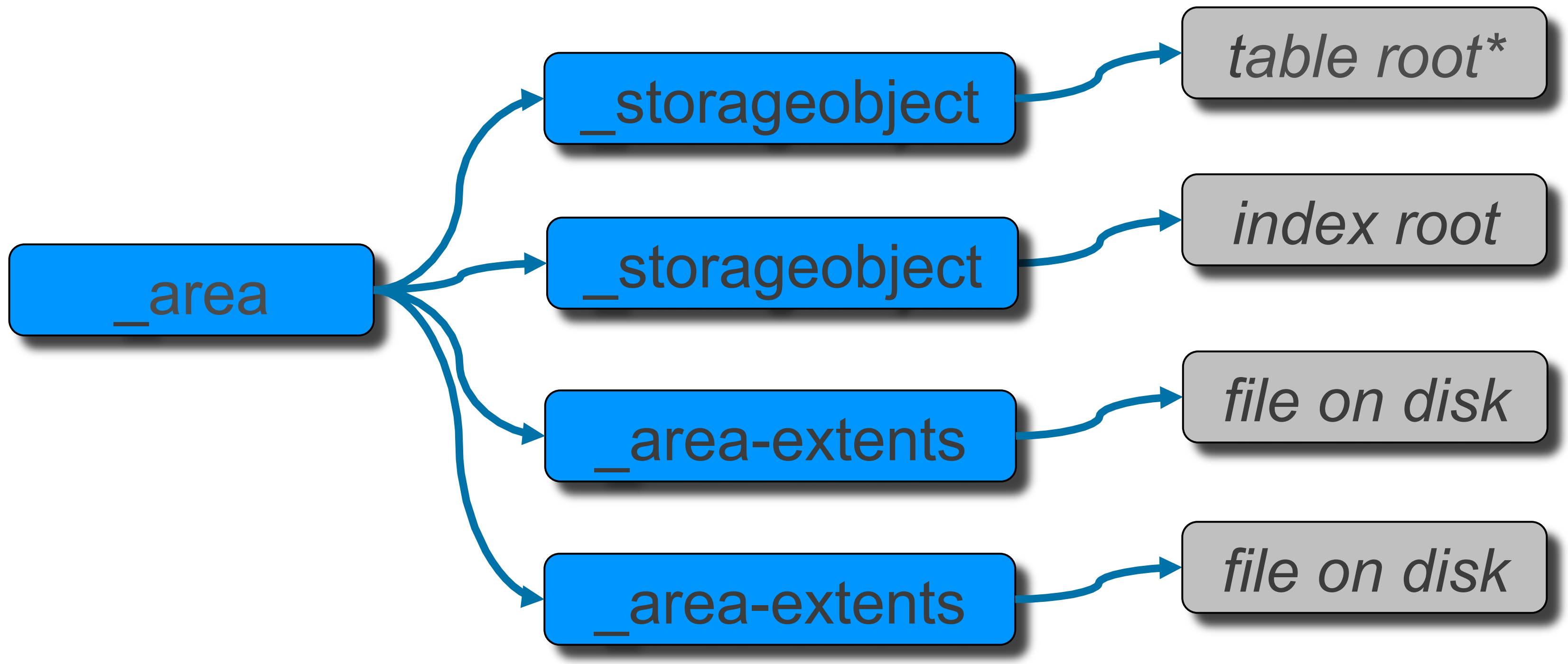
Physical storage related tables

Storage

- Storage defined by .ST file
- Used by prostrct

```
d "Schema Area"  /db/gus/atm.d1
d "atm":7,64;512 /db/gus/atm_7.d1     f 2000000
d "atm":7,64;512 /db/gus/atm_7.d2     f 2000000
d "atm":7,64;512 /db/gus/atm_7.d3     f 2000000
d "atm":7,64;512 /db/gus/atm_7.d4     f 2000000
d "atm":7,64;512 /db/gus/atm_7.d5     f 2000000
d "atm":7,64;512 /db/gus/atm_7.d6     f 2000000
d "atm":7,64;512 /db/gus/atm_7.d7
b /bi/gus/atm.b1
```

Physical Storage Schema Tables



** table root exists only for tables in Type II data areas*

can be multiple storage object rows for partitioned tables

Physical Storage Schema Tables

Table	Contents
_Area	Area definition information, 1 row per area, including AI and BI areas
_Area-extent	Extent definition information, 1 row per extent, including AI and BI extents
_StorageObject	Database object information, 1 row per table, index, and LOB column 1 row for each partition

_area table selected fields

Column	Contents	Type
_area-name	Area name	Char
_area-number	Area number	Integer
_area-recbits	Rows per block = 2 ^ recbits	integer
_area-type	Area type	Integer
_area-extents	Number of extent files	Integer
_area-clustersize	Allocation cluster size or 1	Integer
_area-blocksize	Block size bytes	integer

_area-extent table selected fields

Column	Contents	Type
_extent-path	Pathname of file	Char
_extent-size	Extent size if fixed, max size if variable	Integer
_extent-type	Fixed or variable	Integer
_extent-number	Extent number within area	Integer
_area-number	Which are owns extent	Integer
_area-recid	Location of _area record	recid

_storageobject table selected fields

Column	Contents	Type
_object-number	Object number	Integer
_area-number	Which area object is in	Integer
_object-type	Table, index, lob	Integer
_create-limit	Create limit for tables	Integer
_toss-limit	Toss limit for tables	Integer

List tables and indexes by area

```
FOR EACH _area WHERE _area._area-type EQ 6,
    EACH _storageobject
    WHERE _storageobject._area-number EQ
          _area._area-number:
  DISPLAY _area-name.
  CASE _storageobject._object-type:
    WHEN 1 THEN DO: FIND FIRST _file
        WHERE _file-number EQ
              _storageobject._object-number.
      DISPLAY _file-name.
    END.
    WHEN 2 THEN DO: FIND FIRST _index
        WHERE _idx-num EQ _storageobject._object-number.
      DISPLAY _index-name.
    END.
  END CASE.
END.  /* code by George Potemkin */
```

List indexes by storage area and table

```
for each _area, each _storageobject
     where (_storageobject._area-number = _area._area-number),
     each _index
     where (_index._idx-num = _storageobject._object-number)
               and (_storageobject._object-type eq 2) :

     find _file of _index.

     if (_file._file-number > 0) then
        display _area._area-name _file._file-name _index._index-name.
end.
```

How much space is being used?

```
for each _areastatus where
   ( not _areastatus-areaname matches "*After Image Area*" )
   no-lock:

  display
     _areastatus-areanum  format ">>>" column-label "Num"
     _areastatus-areaname format "x(20)" column-label "Area Name"
     _areastatus-totblocks column-label "Tot blocks"
     _areastatus-hiwater column-label "High water mark"
     _areastatus-hiwater / _areastatus-totblocks * 100 column-label "% use"
     _areastatus-extents  format ">>>" column-label "Num Extents"
     .
end.
```

Note: _areastatus is a VST

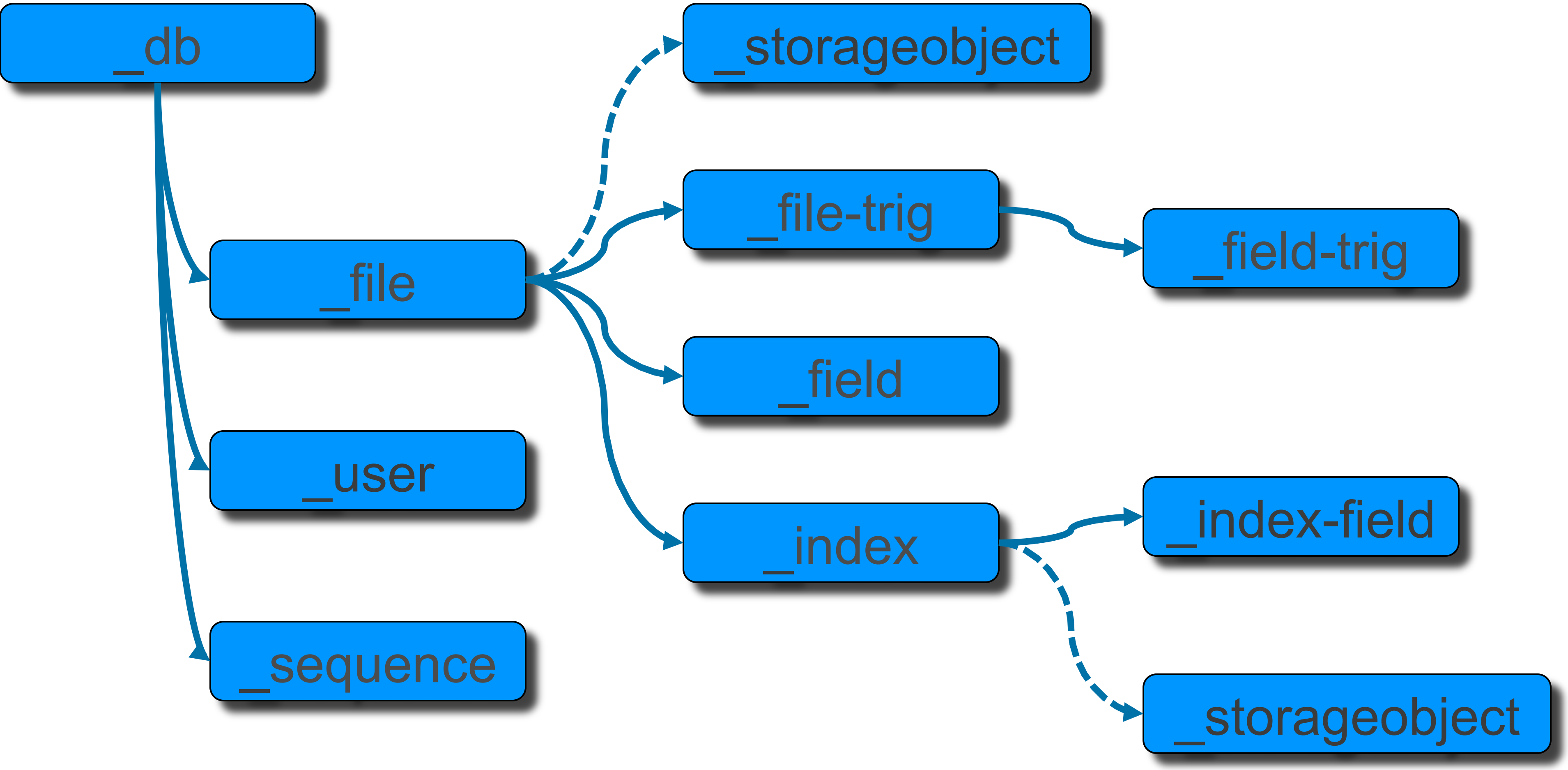
Basic schema tables for 4GL

Data Dictionary

- Schema tables are maintained by the data dictionary program
  - Interactive
  - Load a .df from 4GL
  - Dump a .df from 4GL
- Can also use SQL to
  - CREATE TABLE, CREATE INDEX
- DO NOT UPDATE SCHEMA TABLES DIRECTLY

```
run prodict/load_df.p ("accounts.df").
```

Basic 4GL Schema Tables



Basic 4GL Schema Tables

Table	Contents
_Db	"Owns" the other tables, but could be more in schema holder databases, or autconnect databases
_File	Table definition information, 1 row per table
_Field	Field (column) definition information, 1 row per field for all fields in all tables, Including VST's and Schema tables
_Index	Index definition information, 1 row per index
_Index-field	Key definition information, 1 row per key field for all indexes
_Sequence	Sequence definition information, 1 row per sequence generator

Categories of _file records

Category	How to identify
Your tables	0 > _file-number < 32000
Basic schema tables	-80 < _file-number < 0
Virtual system tables	_file-number < -16384
SQL catalog tables	_file-name begins with "_sys"
SQL-89 view definitions	_file-name begins with "_view" OBSOLETE

Every table, real or virtual, has an _file record

List Your Tables

```
 for each _file
     where (0 < _file-num) and
           (_file-num < 32000):

     display _file-num _file-name
                .
 end.
```

List all schema the tables and category thereof

```
for each _file where _file-num < 0:
   display _file-name _category
end.
```

List the basic schema tables

```
for each _file
    where (_file-num < 0) and
          (_file-num > -80)
    by _file-num descending:

    display _file-num _file-name
            .
end.
```

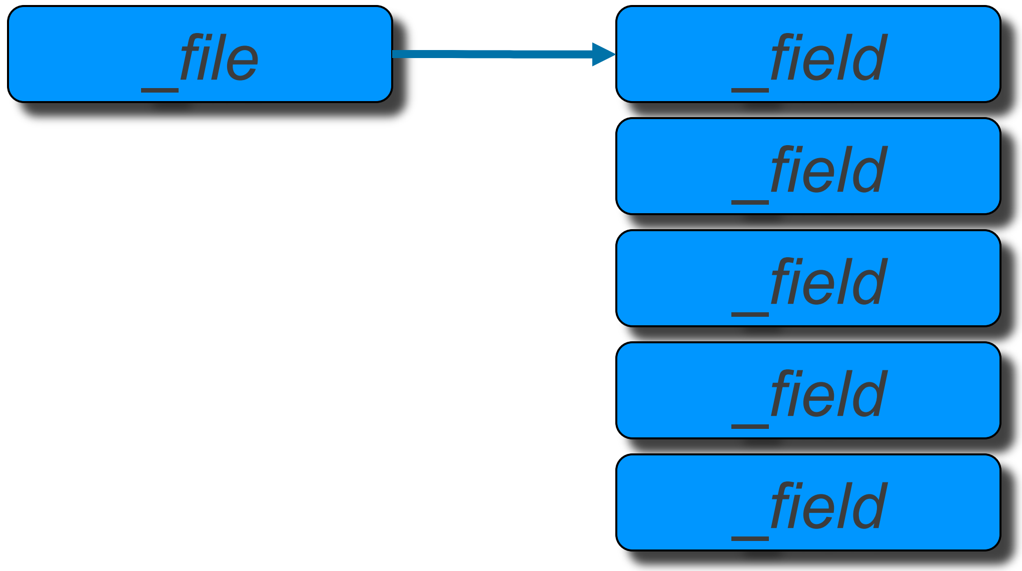
List VST tables

```
  for each _file
      where (_file-num < -16384)
      by _file-num descending:

      display _file-num _file-name
              .
  end.
```

List SQL catalog tables

```
for each _file
    where (_file-name begins "_SYS"):

    display _file-num _file-name
                .
end.
```

Table definitions



One _field record for each field (column) of a table

_file table selected fields

Column	Contents	Type
_file-name	Table name	Char
_file-number	Table number	Integer
_frozen	Can definition be changed	logical
_numfld	Number of fields	Integer
_numkcomp	Number of key components	Integer
_numkey	Number of indexes	Integer
_prime-index	Recid of primary _index	Recid
_crc	Table crc	Integer

_field table selected fields

Column	Contents	Type
_field-name	Name	char
_field-rpos	Logical field order	integer
_field-physpos	Physical field order	integer
_data-type	Field data type	character
_decimals	Decimal digits if decimal	integer
_format	Default display format	char
_can-read	Read permission string	Char
_can-write	Write permission string	Char
_extent	Number of array elem	Integer
_width	Max allowed characters	Integer

List Your Tables and Their Fields

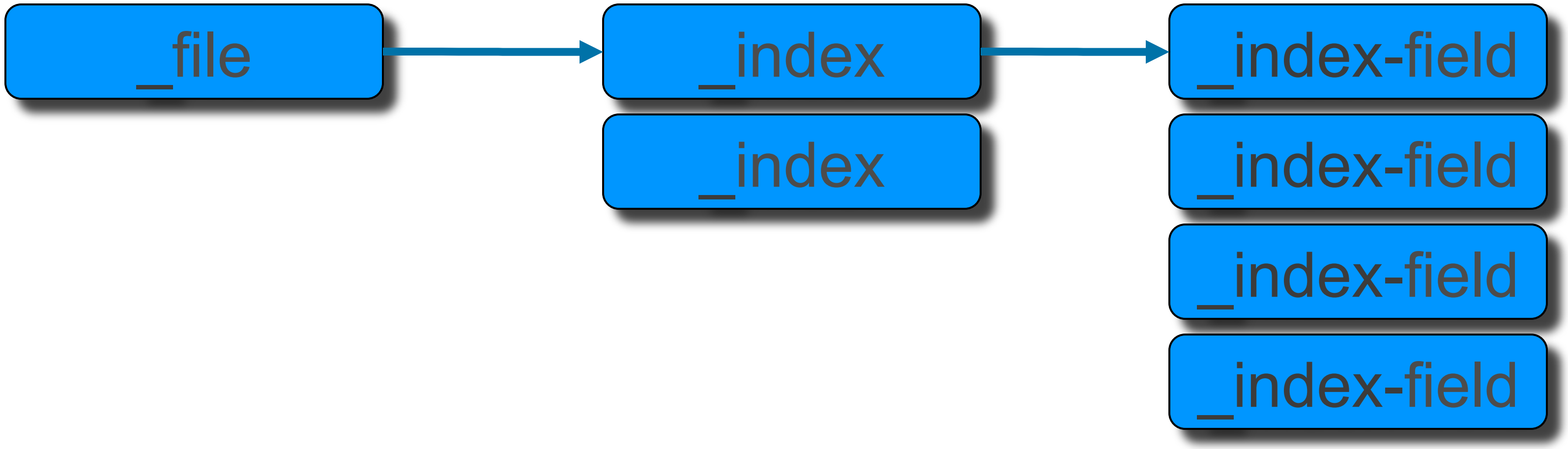
```
output to tables.txt.
for each _file
    where (0 < _file-num):

    put _file-name skip.
    for each _field of _file:
        put "    " _field-name skip.
    end.
    put "" skip.
end.
output close.
```

Table and fields

Invoice
- Adjustment
- Amount
- Cust-Num
- Invoice-Date
- Invoice-Num
- Order-Num
- Ship-Charge
- Total-Paid

Customer
- Address
- Address2
- Balance
- City

Index definitions



One _index record for each index of a table,

One _index-field record for each key component of an index

_index table selected fields

Column	Contents	Type
_index-name	Name	char
_idx-num	Index number	integer
_file-recid	Which file owns index	recid
_idx-crc	Index crc value	integer
_num-comp	Number of key components	integer
_unique	Is index uniqe or nonunique	logical
_wordidx	Regular or word index	integer

_index-field table selected fields

Column	Contents	Type
_index-recid	Which index	Recid
_field-recid	Which field	recid
_ascending	Which order	logical

How 4GL finds records

- Compiler determines which index (perhaps cust-num)
  - R-code has index information
  - Query used to form equality or range bracket
- At runtime, load schema cache into memory
- Look up index number (_index.idx-num, # 113)
- Find _storage-object record for index 113
- Load into "om cache" so we can use it again
- Get location of index root block from _storage-object
- Traverse index b-tree from root to leaf, perhaps cust-num = 20
- Get record's rowid from index leaf block
- Read data block containing record
- Copy record to 4GL buffer or network buffer

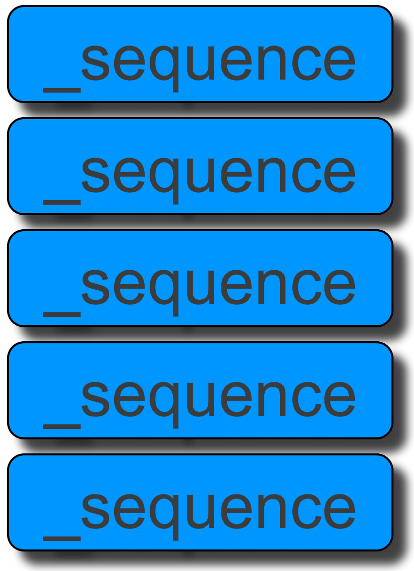
List indexes and key components by table

```
output to index.txt.
for each _file where _file-num > 0:

  put _file-name skip.
  for each _index of _file:
    put "              " _index-name skip.
    for each _index-field of _index:
      find _field where recid(_field) = _field-recid.
      put "          " _field-name skip.
    end.
  end.
  put "" skip.
end.
output close.
```

Index report

```
Customer
    Comments
        Comments
    Country-Post
        Country
        Postal-Code
    Cust-Num
        Cust-Num
    Name
        Name
    Sales-Rep
        Sales-Rep
```

Sequence generators



One _sequence record for each sequence generator

_sequence table selected fields

Column	Contents	Type
_seq-name	Sequence name	char
_seq-num	Sequence number	integer
_seq-min	Minimum value	Integer 64
_seq-max	Maximum value	Integer 64
_seq-incr	Increment amount	Integer

That’s all we have time for today, except

PROGRESS BravePoint

MDBA

Answers

Email:

gus@bravepoint.com