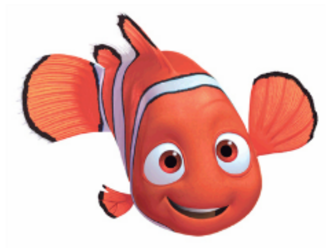
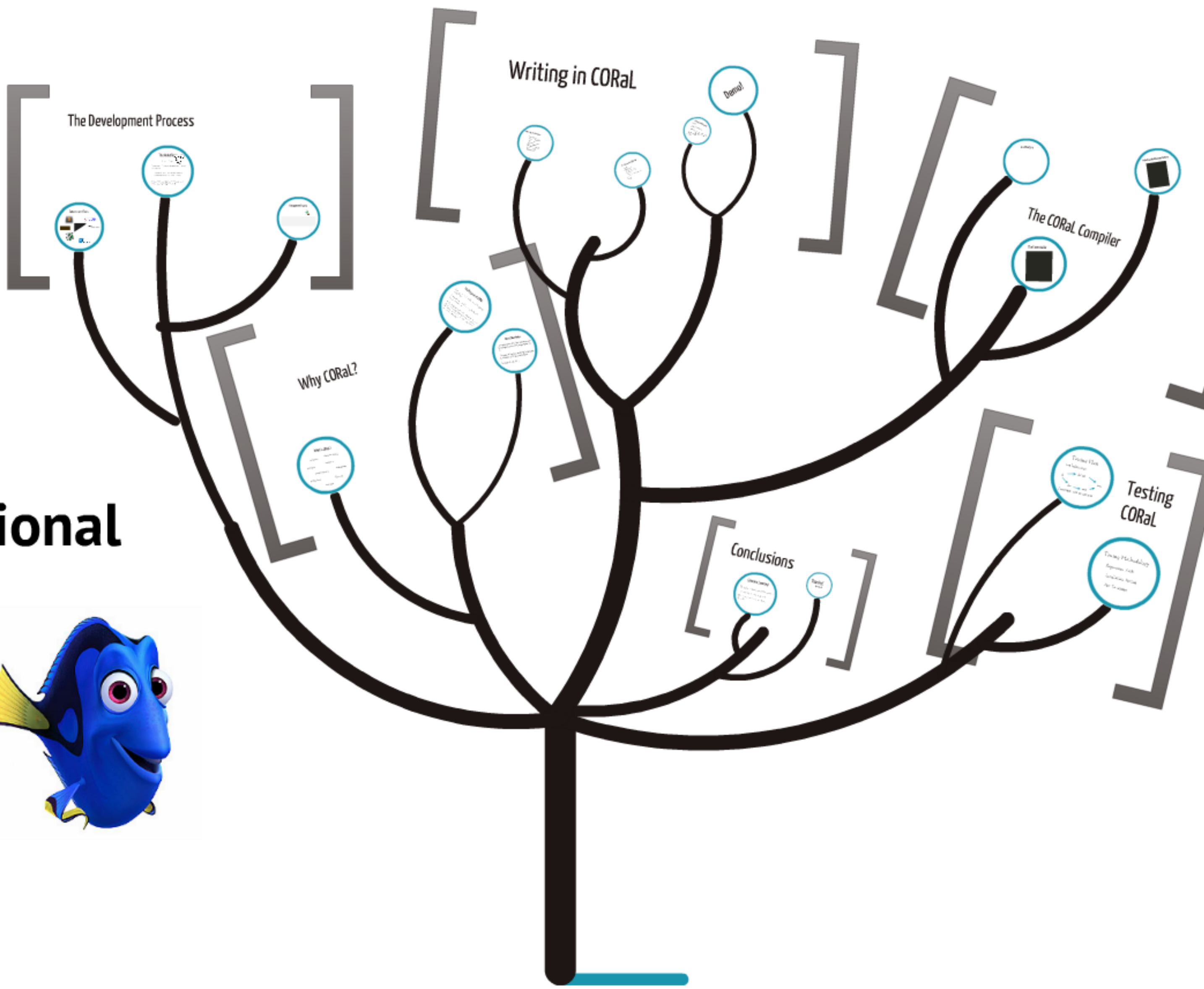


CORaL

C-like Object Relational Language



Shane Chin - System Tester
Molly Karcher - Language Guru
Luis Peña - System Architect
Miguel Yanez - System Integrator
Brian Wagner - Project Manager

Prezi

Why CORaL?

The Purpose of CORaL

Have you ever found yourself struggling with SQL?

Databases are used for everything, but they're hard to use

Some languages have nice database support but they're scripting languages whose features known by a lot of people who only know one or two languages

How CORaL helps

We provide a familiar environment for programmers with experience in C.

No need to learn a scripting language to interact with your database

No need to use SQL

What is CORaL?

intuitive

object-relational

portable

versatile

vendor-neutral

easy-peasy

productive

familiar

universal

The Purpose of CORaL

Have you ever found yourself struggling with SQL?

Databases are used for everything, but they're hard to use

Some languages have nice database support they're scripting languages whose features known by a lot of people who only know one or two languages

What is CORaL?

object-relational

intuitive

versatile

portable

easy-peasy

vendor-neutral

productive

familiar

universal

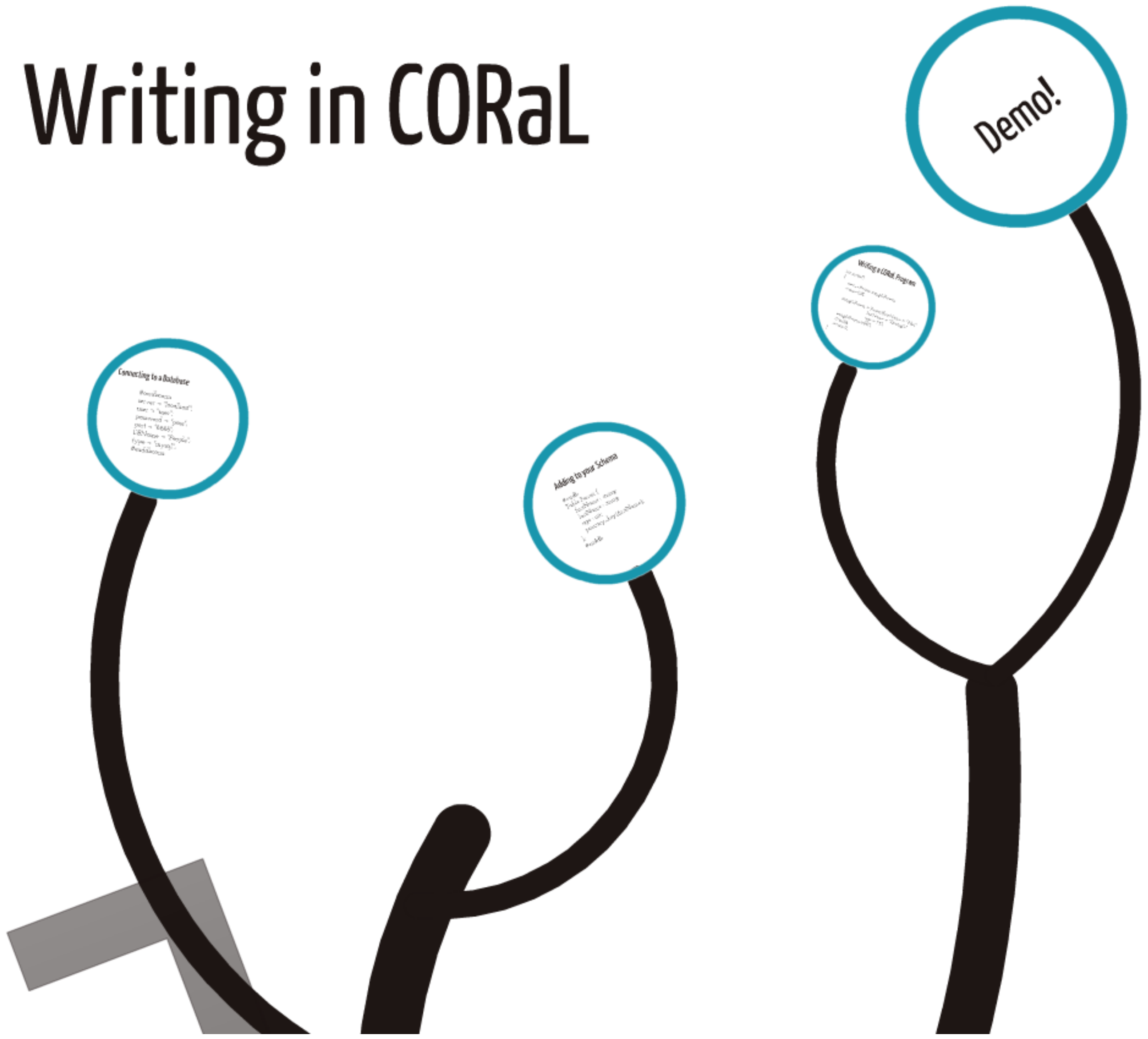
How CORaL helps

We provide a familiar environment for programmers with experience in C.

No need to learn a scripting language to interact with your database.

No need to use SQL.

Writing in CORaL



Connecting to a Database

```
#cordbconn
server = "localhost";
user = "user";
password = "pass";
port = "8888";
DBName = "People";
type = "mysql";
#enddbconn
```

Adding to your Schema

```
#cordb
Table Person {
    firstName : string;
    lastName : string;
    age : int;
    primary_key(firstName);
};
#enddb
```

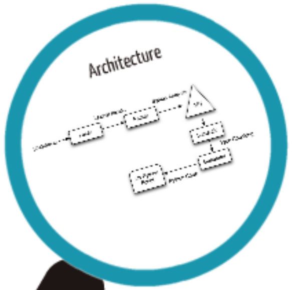
Prezi

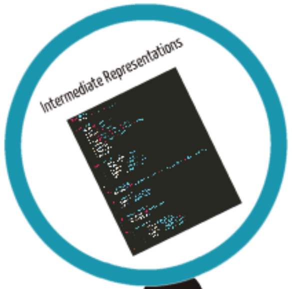
Writing a CORaL Program

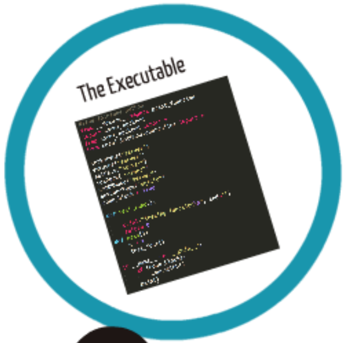
```
int main()
{
    user_t Person samplePerson;
    connectDB;

    samplePerson = Person(firstName = "John",
                          lastName = "Example",
                          age = 25);
    samplePerson.add();
    closeDB;
    return 0;
}
```

Demo!

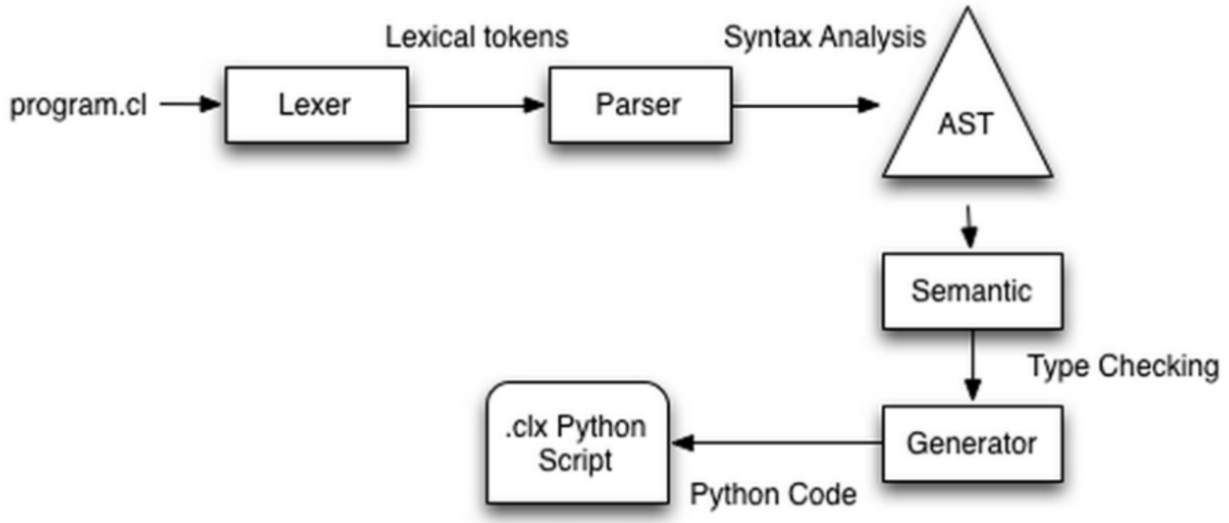
The CORaL Compiler



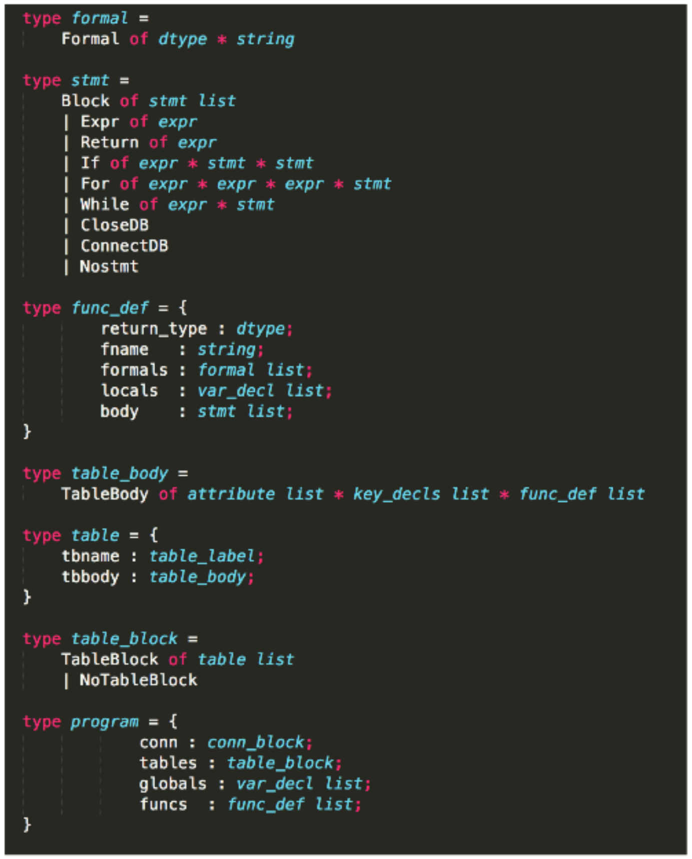




Architecture

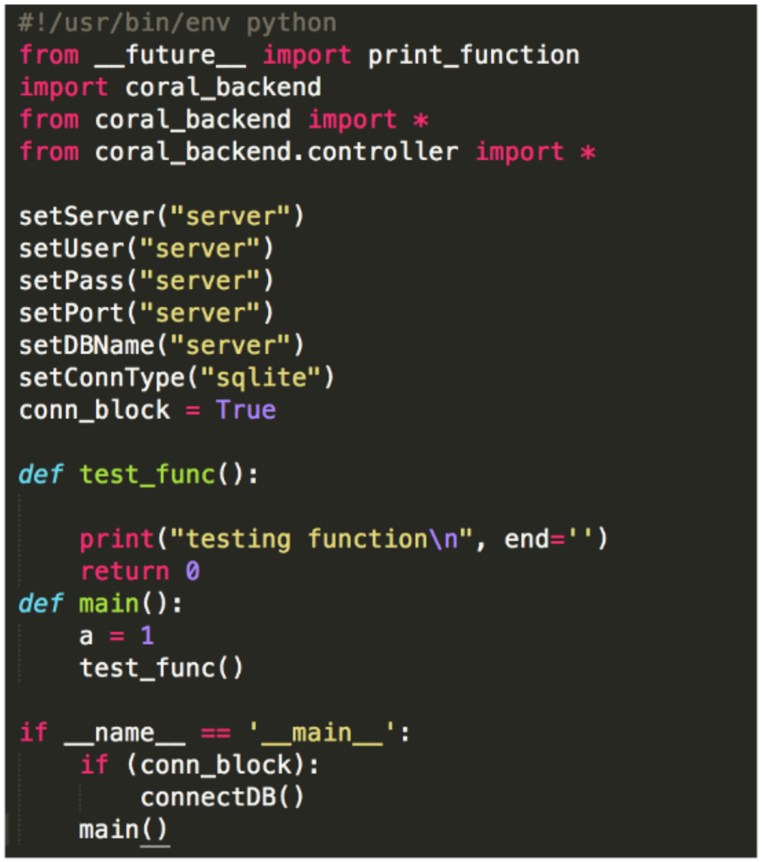


Intermediate Representations



```
type formal =
    Formal of dtype * string

type stmt =
    Block of stmt list
    | Expr of expr
    | Return of expr
    | If of expr * stmt * stmt
    | For of expr * expr * expr * stmt
    | While of expr * stmt
    | CloseDB
    | ConnectDB
    | Nostmt

type func_def = {
        return_type : dtype;
        fname    : string;
        formals : formal list;
        locals  : var_decl list;
        body    : stmt list;
}

type table_body =
    TableBody of attribute list * key_decls list * func_def list

type table = {
    tbname : table_label;
    tbbody : table_body;
}

type table_block =
    TableBlock of table list
    | NoTableBlock

type program = {
            conn : conn_block;
            tables : table_block;
            globals : var_decl list;
            funcs  : func_def list;
}
```

The Executable



```
#!/usr/bin/env python
from __future__ import print_function
import coral_backend
from coral_backend import *
from coral_backend.controller import *

setServer("server")
setUser("server")
setPass("server")
setPort("server")
setDBName("server")
setConnType("sqlite")
conn_block = True

def test_func():

    print("testing function\n", end='')
    return 0
def main():
    a = 1
    test_func()

if __name__ == '__main__':
    if (conn_block):
        connectDB()
    main()
```

The Development Process

Development Tools

git

SQLAlchemy

SQLite

The Makefile

GNU Make

Top-level Makefile to execute all other makefiles.

Different build stages for testing, development, and installation

Once installed you can run coralc as it is already in your PATH.

Management Process

github

Prezi

Development Tools

OCaml

git

SQLAlchemy

Vim

SQLite

The Makefile

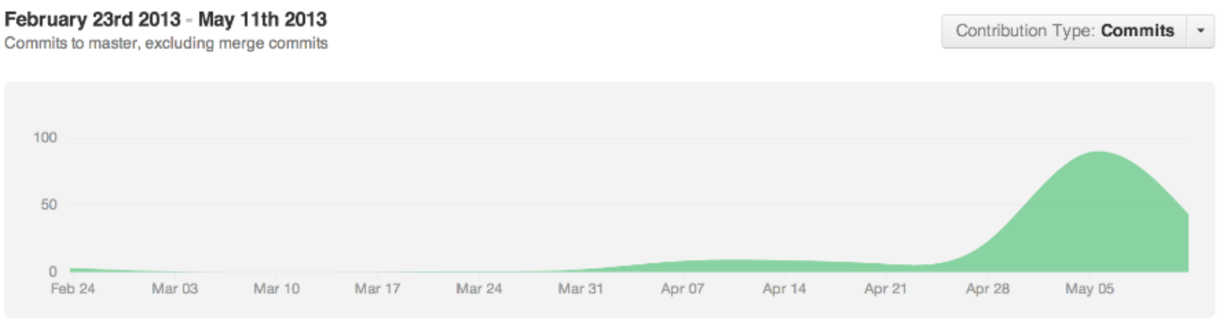
GNU Make

Top-level Makefile to execute all other makefiles.

Different build stages for testing, development, and installation

Once installed you can run coralc as it is already in your PATH.

Management Process



Testing CORaL

Testing Plan

Test before code

$$\begin{tikzcd}
 & \text{.clx.err} \arrow[rrd] & & \\
\text{.cl} \arrow[ru] \arrow[rd] & & & \text{exp} \\
 & \text{.clx} \arrow[r] & \text{.out} \arrow[ru] &
\end{tikzcd}$$

Regression Test on commit

Testing Methodology

Regression Tests

Installation Testing

Test Coverage

Testing Plan

Test before code

$$\begin{tikzcd}
 & \text{.clx.err} \arrow[drr] & & \\
\text{.cl} \arrow[ur] \arrow[dr] & & & \text{exp} \\
 & \text{.clx} \arrow[r] & \text{.out} \arrow[ur] &
\end{tikzcd}$$

Regression Test on commit

Testing Methodology

Regression Tests

Installation Testing

Test Coverage

Conclusions

Lessons Learned

The power of functional languages

Integration is easy in-person

The team matters more than the roles

Thanks!

Questions?

Lessons Learned

The power of functional languages

Integration is easy in-person

The team matters more than the roles

Thanks!

Questions?