Persistence with Ada Database Objects (ADO)

Stéphane Carrez

FOSDEM 2019

Design Goals

- Support several databases *à la* JDBC
- Type safe interface to create/read/update/delete
- Map SQL rows and results in Ada records
- Generate database mappings
- Still allow to use and tune SQL queries

Database Session Factory

- The session factory manages database connections
- The factory is a limited type to prevent copies
- It is initialized with a connection string:
  - `mysql://localhost:3306/samples?user=test`
  - `sqlite:///samples.db?synchronous=OFF&encoding='UTF-8'`
  - `postgresql://localhost:5432/samples?user=test`

```
with ADO.Sessions.Factory;
with ADO.Drivers;
procedure Select_User is
  Factory : ADO.Sessions.Factory.Session_Factory;
begin
  ADO.Drivers.Initialize ("samples.properties");
  Factory.Create (ADO.Drivers.Get_Config ("ado.database"));
  ...
end Select_User;
```

Database Session

- Holds the database connection
- Two types of database sessions to support database replication:
  - A `Session` type to connect to a read-only database server (replicated server)
  - A `Master_Session` type to connect to a write capable database server (master)
- `Session` and `Master_Session` types can be copied and use reference counting

```
with ADO.Sessions;

  Session : ADO.Sessions.Session := Factory.Get_Session;
  --  Can only run queries

  Session : ADO.Sessions.Master_Session : Factory.Get_Master_Session;
  --  Can run queries, start transactions, insert, update, delete
```

Simple SQL query

- Create a query statement using SQL
- Execute and get manually each row and column

```
with ADO.Statements;
...
  Name    : String := ...;
  Stmt    : ADO.Statements.Query_Statement
    := Session.Create_Statement
          (“SELECT * FROM user WHERE name = :name”);
...
  Stmt.Bind_Param (“name”, Name);
  Statement.Execute;
  while Stmt.Has_Elements loop
    Id := Stmt.Get_Identifier (0);
    ...
    Stmt.Next;
  end loop;
```

But...

- Difficulties with the manual approach
  - Can make errors when getting values,
  - Can make type errors when updating
- Solution
  - Map SQL results in Ada
  - Map database tables in Ada

Mapping SQL to Ada

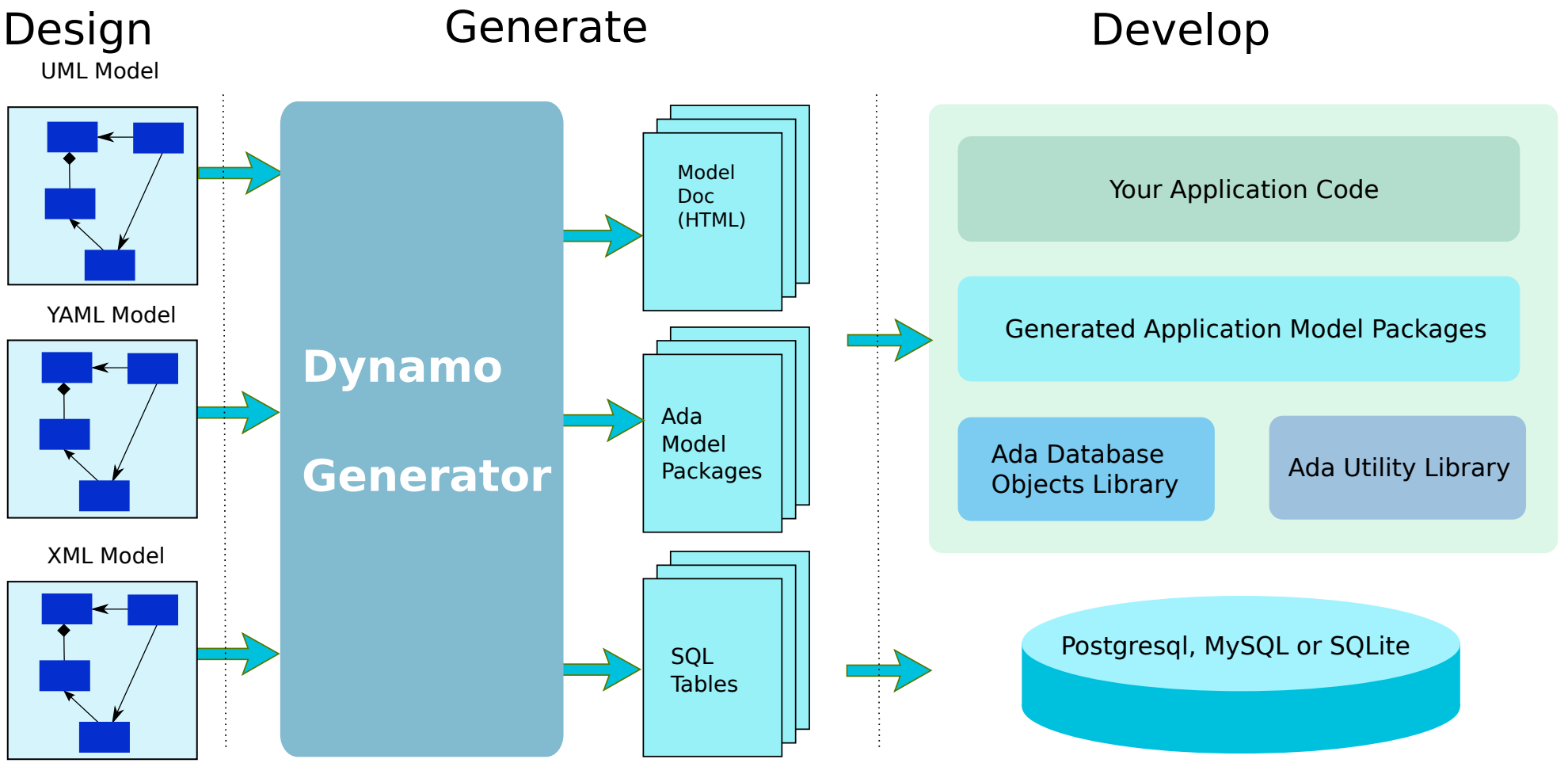
- Map the SELECT row in an Ada record
- Define a Vector of the Ada record
- Procedure to run the query and map the results

```
SELECT
  u.id AS id,
  u.name AS name,
  u.email AS email
FROM
  user AS u
ORDER BY
  u.name ASC
```

```
type User_Info is record
  Id : ADO.Identifier;
  Name : Unbounded_String;
  Email : Unbounded_String;
end record;

package User_Info_Vectors is
  new Ada.Containers.Vectors (Index_Type => Positive,
                              Element_Type => User_Info);

subtype User_Info_Vector is User_Info_Vectors.Vector;

procedure List (Object  : in out User_Info_Vector;
                Session : in out ADO.Sessions.Session'Class;
                Context : in out ADO.Queries.Context'Class);
```

https://github.com/stcarrez/ada-ado

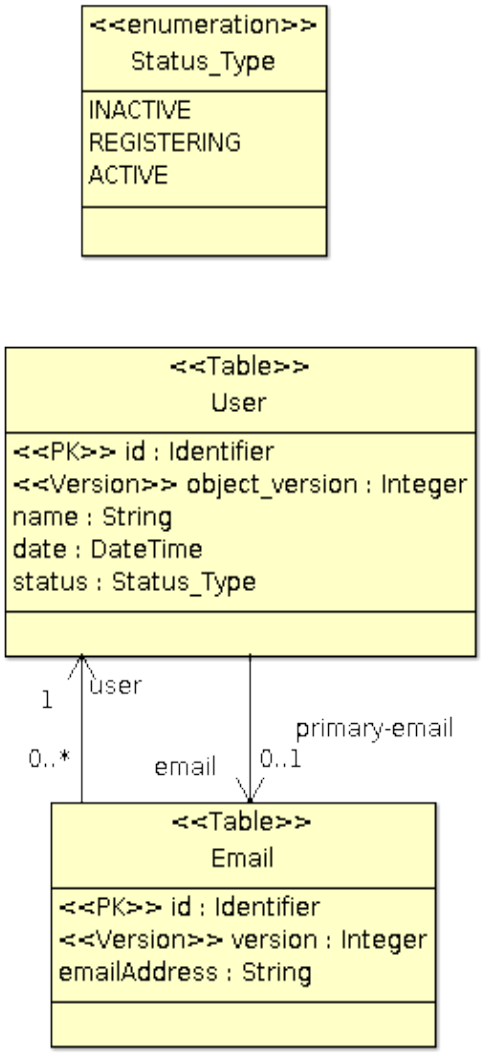
Database Modeling



https://github.com/stcarrez/ada-ado

Database Modeling YAML vs UML

```
Samples.User.Model.User:
  type: entity
  table: user
  description: Record representing a user
  id:
    id:
      type: identifier
      column: id
      not-null: true
      unique: true
      description: the user identifier
      generator:
         strategy: sequence
  fields:
    version:
      type: integer
      column: object_version
      not-null: true
      version: true
    name:
      type: string
      length: 255
      column: name
      not-null: true
      description: the user name
    date:
      type: date
      column: date
      not-null: true
      description: the user registration
...
```



https://github.com/stcarrez/ada-ado

Generated Ada Model

- Public object reference type with accessors
- Private implementation type holds the values
- Load, Save, Delete, Find operations

```
package Samples.User.Model is
  type Status_Type is (INACTIVE, REGISTERING, ACTIVE);

  type Email_Ref is new ADO.Objects.Object_Ref with null record;
  type User_Ref is new ADO.Objects.Object_Ref with null record;

  procedure Set_Name (Object : in out User_Ref; Value : in String);
  function Get_Name (Object : in User_Ref) return String;
  overriding procedure Save (Object : in out User_Ref; ...);
...
private
  type Email_Impl is new ADO.Objects.Object_Record ...;
  type User_Impl is new ADO.Objects.Object_Record ...;
end Samples.User.Model;
```

https://github.com/stcarrez/ada-ado

Using the Ada Model

- Declare *T_Ref* instances
- Use `Get_X` and `Set_X` to access attributes
- Use `Load`, `Find` to retrieve and `Save`, `Delete` to modify

```
User  : User_Ref;
Email : Email_Ref;

User.Set_Name (“Ada Lovelace”);
User.Set_Status (REGISTERING);
User.Save (Session);
...
Email.Set_Emailaddress (“ada@protonmail.com”);
Email.Set_User (User);
Email.Save (Session);
User.Set_Email (Email);
User.Set_Status (ACTIVE);
User.Save (Session);
```

```
INSERT INTO user (id,object_version,name,
email,date,status) VALUES(?, ?, ?, ?, ?, ?)
```

```
INSERT INTO email (id,version,name,
emailAddress, user) VALUES(?, ?, ?, ?)
```

```
UPDATE user SET status = ?, email = ?
WHERE id = ? AND object_version = ?
```

https://github.com/stcarrez/ada-ado

Bonus: database auditing

- Track and record changes in database
- Apply `<<auditable>>` UML stereotype to attributes or `auditable: true` in YAML
- `Audit_Manager` interface called after each `Save` with:
  - Object record
  - List of changes with field, old value, new value

Conclusion

- Generated model simplifies database access
- Strong typing representation of SQL queries
- Dynamo generates Ada model and SQL tables
- Ada Database Objects Programmer's Guide
  - https://ada-ado.readthedocs.io/en/latest/