

Universidad de Cantabria – Facultad de Ciencias
Ingeniería en Informática
Bases de Datos Avanzadas - Teoría

Ejercicios del Tema 3, parte b

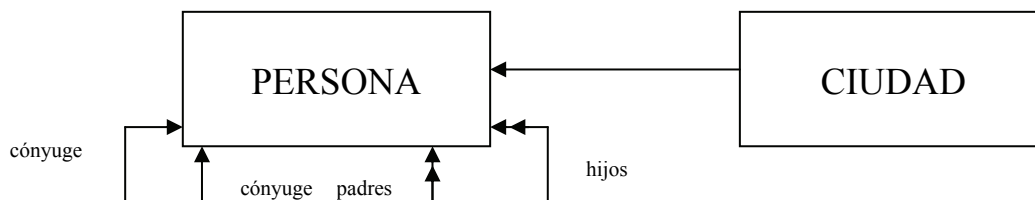
Ejercicio 3.1:

Se quiere modelar en ODMG 3.0 una base de datos que maneja información sobre personas y sus relaciones familiares y de parentesco: matrimonios, hijos, etc.

Las principales características a modelar son:

1. Persona tiene una extensión denominada gente.
2. Una Persona tiene un nombre, una dirección, un cónyuge, hijos y padres.
3. Las operaciones nacimiento, matrimonio, ancestros y mover son también características de Persona.
4. Nacimiento añade un nuevo hijo a la lista de hijos de una instancia de Persona.
5. Matrimonio define un cónyuge para una instancia Persona.
6. Ancestros computa el conjunto de instancias Persona que son ancestros de una instancia Persona.
7. Mover cambia la dirección de una instancia Persona.
8. Una dirección es una estructura cuyas propiedades son: número, calle y ciudad. Número es un unsigned short, calle y ciudad son de tipo string.
9. Ciudad tiene las propiedades código (unsigned short), nombre(string) y población (conjunto de referencias a objetos Persona).
10. Cónyuge es una ruta cónyuge:cónyuge con cardinalidad 1:1 recursiva.
11. Hijos es una de las rutas de hijos:padres de cardinalidad n:m recursiva.
12. Padres es otra ruta de la relación hijos:padres de cardinalidad m:n recursiva.

SOLUCIÓN:



El esquema en ODL es:

```
Class Persona (extent gente) {
  attribute string name;
  attribute sttstruct Address {unsigned short number, string street,
string cioty_name}address;
  relationship persona cónyuge
    inverse Persona::cónyuge;
  relationship set<Persona> hijos inverse Persona::Padres;
  relationship list<Persona> padres inverse Persona::Hijos;
  void birth(in string name);
  boolean marriage(in string person_name)
    raises(no_persona);
  unsigned short ancestors(out set<Persona>all_ancestros)
    raises(no_persona);
  void move(in string nueva_dir);
```

```

};

class Ciudad (extent ciudades) {
    attribute unsigned short código;
    attribute string nombre;
    attribute set <Persona>población;
};

```

Ejercicio 3.2:

Dado el siguiente esquema objeto-relacional en SQL:2003 transformar a un esquema orientado a objetos puro en ODM 3.0:

```

CREATE SCHEMA video_and_music
    AUTHORIZATION m_s_enterprises
    DEFAULT CHARACTER SET "Latin_1";

CREATE DOMAIN price DECIMAL (7,2)
    CHECK (VALUE IS NOT 0);
CREATE DISTINCT TYPE money AS DECIMAL (9,2);

CREATE TYPE movie AS(
    movie_id INTEGER,
    title CHARACTER VARYING (100),
    languages MULTISSET ['English', 'French', 'Spanish',
        'Portuguese', 'Italian'],
    genre CHARACTER VARYING (20) ARRAY [10],
    run_time INTEGER
    )
    INSTANTIABLE
    NOT FINAL
    METHOD length_interval ()
    RETURNS INTERVAL HOUR (2) TO MINUTE;

CREATE INSTANCE METHOD length_interval ()
    RETURNS INTERVAL HOUR (2) TO MINUTE FOR MOVIE
    RETURN CAST (CAST (SELF.run_time AS INTERVAL (4) )
    AS INTERVAL HOUR (2) TO MINUTE );

CREATE TABLE movies (
    stock_number CHARACTER(10)
    CONSTRAINT movies_stock_number_not_null NOT NULL,
    movie movie,
    our_tape_cost price,
    tapes_in_stock INTEGER
    CONSTRAINT movies_primary_key PRIMARY KEY (stock_number)
    );

CREATE TABLE movies_stars (
    movie_title CHARACTER (30)
    CONSTRAINTmovies_stars_movie_title_not_null NOT NULL,
    movie_year_released DATE,
    movie_number CHARACTER (10),
    actor_last_name CHARACTER (35)
    CONSTRAINT movies_stars_actor_last_name_not _null NOT NULL,
    actor_first_name CHARACTER (25)
    CONSTRAINT movies_stars_actor_first_name_not _null NOT NULL,
    actor_middle_name CHARACTER (25),
    CONSTRAINT movies_stars_unique

```

```

        UNIQUE (movie_title, actor_last_name, actor_first_name,
actor_middle_name)
        NOT DEFERRABLE,
        CONSTRAINT movies_stars_fk_movie
        FOREIGN KEY (movie_number)
        REFERENCES movies (stock_number)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    );

CREATE TYPE music_distributors AS (
    distributor_id CHARACTER (15),
    distributor_name CHARACTER (25)
);

CREATE TABLE music_distributors OF music_distributors (
    REF IS dist_ref SYSTEM GENERATED,
    distributor_id WITH OPTIONS
    CONSTRAINT music_distributors_distributor_id_not_null NOT NULL,
    distributor_name WITH OPTIONS
    CONSTRAINT music_distributors_distributor_name_not_null NOT
    NULL,
);

CREATE TYPE address AS(
    street CHARACTER VARYING (35),
    city CHARACTER VARYING (40),
    country character (3)
);

CREATE TYPE US_address UNDER address AS(
    state CHARACTER (2),
    zip ROW (
        Basic INTEGER,
        Plus4 SMALLINT)
    )
    METHOD zipcode ()
    RETURNS CHARACTER VARYING (10);

CREATE INSTANCE METHOD zipcode ()
    RETURNS CHARACTER VARYING (10)
    FOR US_address
    BEGIN
        IF SELF.zip.plus4 IS NULL
        THEN RETURN CAST (SELF.zip.basic AS CHARACTER VARYING (5));
        ELSE RETURN CAST (SELF.zip.basic AS CHARACTER VARYING (5))
        || '-'
        || CAST (SELF.zip.basic AS CHARACTER VARYING (4))
        ENDIF;
    END;

CREATE TABLE customers(
    nr_of_customer INTEGER GENERATED ALWAYS AS IDENTITY (START WITH
    1 INCREMENTED BY 1MINVALUE
    1),
    cust_last_name CHARACTER (35)
    CONSTRAINT customers_cust_last_name_not_null NOT NULL,
    cust_first_name CHARACTER (35)
    CONSTRAINT customers_cust_first_name_not_null NOT NULL,
    cust_complete_name GENERATED ALWAYS AS (cust_first_name ||
    cust_last_name),

```

```

    cust_address US_address,
    cust_current_charges money,
    number_of_problems SMALLINT
);

CREATE VIEW problem_customers ( last, first)
AS
SELECT cust_last_name, cust_first_name
FROM customers
WHERE number_of_problems
0.8 * (SELECT MAX(number_of_problems)
FROM customers);

CREATE ASSERTION limit_total_movie_stock_value
CHECK (( SELECT COUNT(*)
FROM customers
WHERE number_of_problems > 5 AND
cust_current_charges > 150,00 AND
cust_current_charges < 1000,00)
<10 );

```