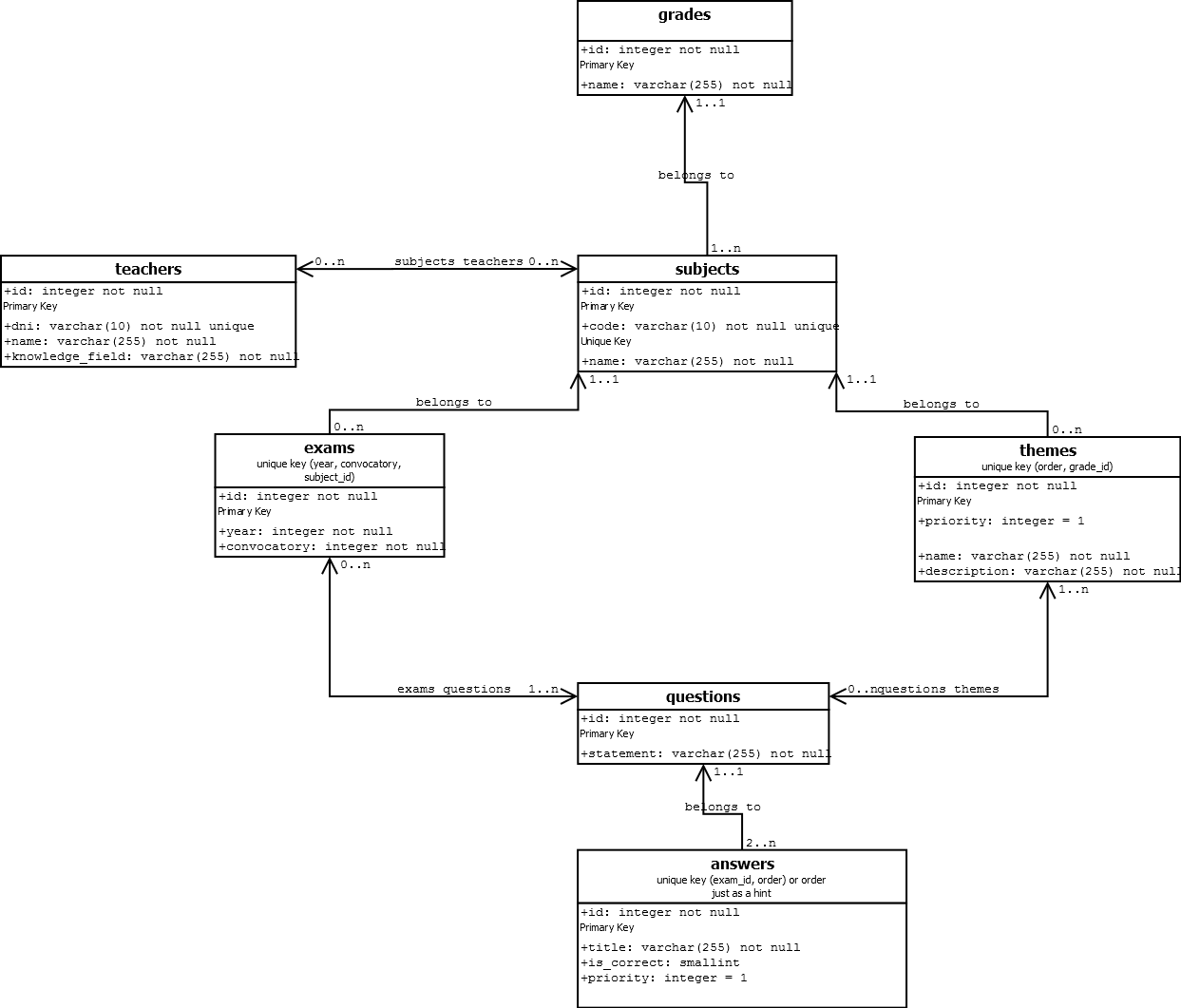
# DESCRIPCIÓN DE LAS TABLAS GENERADAS

Diseño conceptual (modelo E/R)

Nota previa:

* El diagrama con el diseño conceptual (modelo E/R) ha sido generado con la herramienta GNU “DIA”. Esta herramienta no deja usar flechas dobles en las asociaciones entre clases UML. Es, por ello, que se usará una flecha para asociaciones con cardinalidad máxima ‘n’, y no se usará flecha en el lado de la asociación con cardinalidad máxima ‘1’.

Convenciones:

* Si un atributo tiene varias palabras en su nombre, se separarán con barras bajas.
* Las tablas que representan una entidad se llamarán como la entidad, pluralizado.
* Las tablas para relaciones “n-m” se designarán mediante el nombre de la primera entidad ordenada alfabéticamente, tras una barra baja, seguido del nombre de la otra entidad.

Respecto a la elección de claves identificativas de las entidades, hemos considerado los siguientes aspectos:

* Para las entidades “grades”, “subjects”, “teachers”, “themes”, “questions” y “exams” (entidades fuertes) hemos elegido como clave “id” para todas, que garantiza, respectivamente en cada tabla, asociar dicho atributo clave con un valor único.
* Además, las claves externas de dichas entidades son:
* Para la entidad “subjects”, la clave “grade\_id”, que referencia a la clave “id” de la tabla “grades”.
* Para la entidad “themes”, la clave “subject\_id”, que referencia a la clave “id” de la tabla “subjects”.
* Para la entidad “answers”, la clave “question\_id”, que referencia a la clave “id” de la tabla “questions”.
* Para la entidad “exams”, la clave “subject\_id”, que referencia a la clave “id” de la tabla “subjects”.
* Respecto a las entidades débiles “questions\_themes”, “exams\_questions” y “subjects\_teachers” hemos elegido como claves las que, en realidad, son sus claves externas. Es decir:
* Para la entidad “questions\_themes”, sus claves son “question\_id”, que referencia a la clave “id” de la tabla “questions”, y “themes\_id”, que referencia a la clave “id” de la tabla “themes”.
* Para la entidad “exams\_questions”, sus claves son “question\_id”, que referencia a la clave “id” de la tabla “questions”, y “exam\_id”, que referencia a la clave “id” de la tabla “exams”.
* Para la entidad “subjects\_teachers”, sus claves son “subject\_id”, que referencia a la clave “id” de la tabla “subjects”, y “teacher\_id”, que referencia a la clave “id” de la tabla “teachers”.

Diseño lógico relacional

grades(id, name)

Clave primaria: id

teachers(id, dni, name, knowledge\_field)

Clave primaria: id

Clave candidata: dni

subjects(id, code, name, grade\_id)

Clave primaria: id

Clave candidata: code

Clave externa: grade\_id

themes(id, priority, name, description, subject\_id)

Clave primaria: id

Clave externa: subject\_id sobre subjects

questions(id, statement)

Clave primaria: id

answers(id, title, is\_correct, priority, question\_id)

Clave primaria: id

Clave externa: question\_id sobre questions

exams(id, year, convocatory, subject\_id)

Clave primaria: id

Clave externa: subject\_id sobre subjects

questions\_themes(question\_id, theme\_id)

Clave primaria: question\_id, theme\_id

Clave externa: question\_id sobre questions

Clave externa: theme\_id sobre themes

exams\_questions(question\_id, exam\_id, correct\_answer\_count, incorrect\_answer\_count, unreplied\_answer\_count)

Clave primaria: question\_id, exam\_id

Clave externa: question\_id sobre questions

Clave externa: exam\_id sobre exams

subjects\_teachers(subject\_id, teacher\_id)

Clave primaria: subject\_id, teacher\_id

Clave externa: subject\_id sobre subjects

Clave externa: teacher\_id sobre teachers

# Breve descripción de la estructura de la aplicación generada

# 1.Estructura del código C

El código C relativo a la funcionalidad está todo en el directorio src/. El programa está organizado por subcomandos, declarados todos en el fichero principal “app.sc”:

/\* The list of our supported commands \*/

struct command commands[] = {

{ "question", question, "manages questions",

"Usage: question [args...]\n"

"\t-a\tcreates a new question\t-a <statement>\n"

"\t-d\tdeletes a question\t-d <id>\n"

"\t-e\tedits a question\t-e <id> <statement>\n"

"\t-s\tshows a question\t-s <id>\n"

"\t-l\tlists all questions\t-l [<theme\_id>] [--detailed]\n"

"\t-t\tlists all questions by theme and subject\t-t [--detailed]\n"

"Prints the question id on success to stdout." },

{ "answer", answer, "manages answers",

"Usage: answer [args...]\n"

"\t-a\tcreates an answer\t-a <question\_id> <statement> [--correct]\n"

"\t-l\tlist answers for a question\t-l <question\_id>\n"

"\t-d\tdeletes a question\t-d <id>" },

{ "exam", exam, "manages exams",

"Usage: answer [args...]\n"

"\t-l\tlist exams for a question\t-l <question\_id>\n" },

{ "questions\_themes", questions\_themes, "Relation between questions and themes",

"\t-a\tads a new relationship\t-a <question\_id> <theme\_id>\n" },

{ "interactive", interactive, "Open an interactive session" },

{ NULL, NULL, NULL, NULL }

};

El código relativo a la ejecución de los comandos, al ser algo no relacionado con el funcionamiento en sí de la práctica, está en el directorio lib/.

Cada estructura de la lista contiene el alias del comando (la forma de llamarlo), la función que ejecuta (siendo de la forma int (\*) (int argc, char\*\* argv)), una descripción corta que se escribe al ejecutar el comando autogenerado help, y una larga que se escribe al ejecutar el comando help <comando>.

Cada función es encargada de leer el resto de argumentos, y hacer lo que crea conveniente con ellos, y está declarada en un fichero .h con el mismo nombre, y definida en el fichero .sc correspondiente.

Las funciones interaccionan entre ellas usando el macro “CALL” (“common.h”), que manipula los argumentos convenientemente para simular una llamada desde la consola.

Así, escribir:

CALL(question, "-l");

Es equivalente a ejecutar:

./target/app question -l

# 2.Estructura del código SQL

El código sql se encuentra en la carpeta etc/sql/. Bajo el directorio etc/sql/src/ se encuentran:

* **schema.sql**: El esquema de la base de datos.
* **derived\_and\_triggers.sql**: El atributo derivado y los disparadores necesarios para gestionarlo automáticamente.
* **seeds.sql**: Inserción de valores de prueba de la base de datos.
* **view.sql**: La vista requerida por el enunciado.
* **oracle-drops.sql**: Sentencias “drop” de las tablas en Oracle (en postgresql borramos la base de datos directamente).

**Nota**: El esquema de la base de datos varía un poco con respecto al propuesto. Cada tema pertenece exclusivamente a una asignatura, y en vez de forzar órdenes, usamos un campo priority, que es más flexible. Se puede ver la estructura en el esquema de la carpeta doc/

# 3.Compilación y ejecución

Desde la carpeta del proyecto ejecutar:

$ ./configure

$ make

$ cd etc/sql && make

**Nota**: en Olivo, “make” debe ser sustituido por “gmake”

El primer “make” compilará la aplicación en el directorio target/, mientras que el segundo creará las tablas, el atributo derivado, los disparadores, la vista e insertará los datos de prueba.

Usamos un pequeño archivo (“oracle-prepro”) para poder compilar con una interfaz similar a la de gcc o ecpg (compila el archivo pasado como segundo argumento con las opciones deseadas). Usamos la opción DYNAMIC=ORACLE ya que con esta configuración el mensaje de error en sqlca.sqlerrm.sqlerrmc funciona como se espera.

Los warnings generados por Pro\*C acerca del macro CALL deberán ser ignorados. Se deben a que usa argumentos variables (C99), y el preprocesador analiza código según C90.

## 4.Ejecución en modo interactivo

El comando interactive es un comando especial, ya que permite al usuario interaccionar directamente en lugar de mediante consola. Para usar la sesión interactiva usaremos:

$ ./target/app interactive

Donde se podrá ver un menú con el que el usuario puede interaccionar.

# 5.Funcionalidad no requerida

Se puede apreciar en la ayuda de los comandos que hay funcionalidad no requerida para el enunciado. Algunos ejemplos son:

./target/app question -d <id> # Borra una pregunta

./target/app question -e <id> <statement> # Cambia el enunciado de una pregunta

./target/app question -l [--detailed] # Muestra todas las preguntas, opcionalmente con sus detalles

./target/app exam -l <question\_id> # Lista todos los exámenes para una pregunta

./target/app answer -l <question\_id> # Lista todas las respuestas para una pregunta

./target/app answer -d <id> # Borra una respuesta

No se han hecho entradas interactivas de algunas de ellas para evitar incrementar la complejidad.

# 6.Detalles curiosos

-Se ha evitado el tener que especificar la id al insertar los datos mediante un uso de disparadores y secuencias. Los disparadores son autogenerados por el script etc/sql/scripts/generate\_triggers.sh.

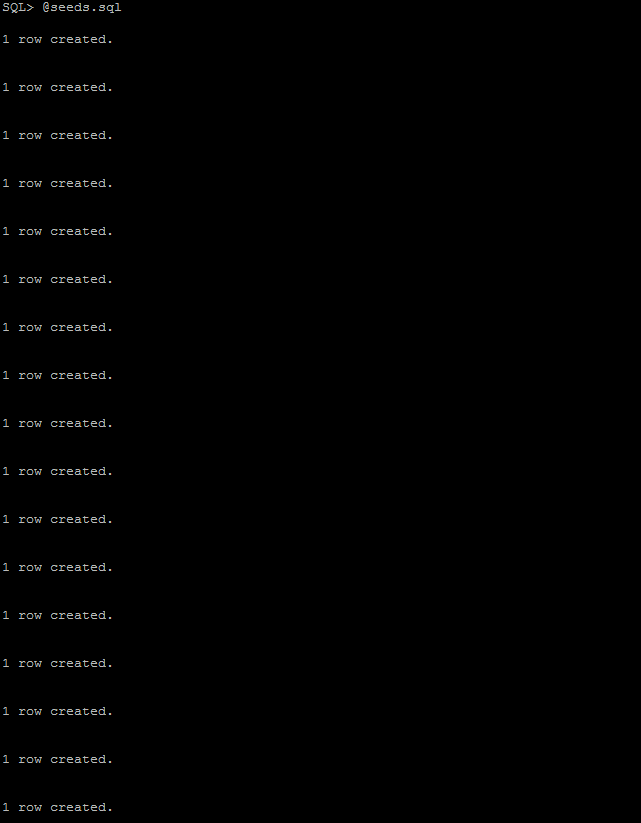
-La vista cuenta los temas que no tienen ninguna pregunta asignada, eso se consigue con una “subselect”. El rendimiento probablemente sea peor que sin ella, pero tenemos el juego de datos esperado.

# Capturas de pantalla

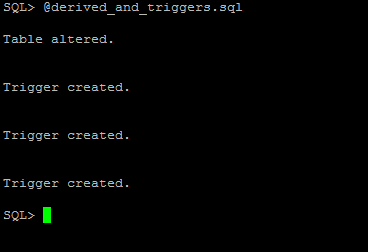
Creación de las tablas



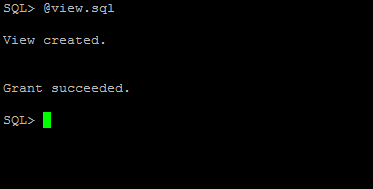
Inserción de un pequeño conjunto de datos en las tablas



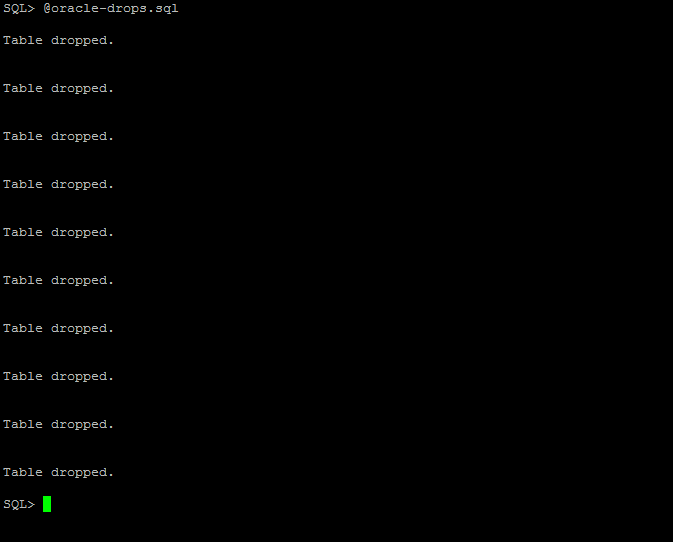
Creación del atributo derivado y los disparadores



Creación de la vista



Borrado de las tablas



# Código fuente de la aplicación en C, así como de los script SQL

Nota previa:

-En este apartado incluiremos el código de todos los archivos, tanto los script SQL, como los ficheros de SQL inmerso en C. No así lo relativo a los ficheros necesarios para la correcta compilación de los mismos.

Script SQL

**“schema.sql”**

CREATE TABLE grades (

id INTEGER NOT NULL,

name VARCHAR(255) NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE teachers (

id INTEGER NOT NULL,

dni VARCHAR(10) NOT NULL UNIQUE,

name VARCHAR(255) NOT NULL,

knowledge\_field VARCHAR(255) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE subjects (

id INTEGER NOT NULL,

code VARCHAR(10) NOT NULL UNIQUE,

name VARCHAR(255) NOT NULL,

grade\_id INTEGER NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (grade\_id) REFERENCES grades(id) ON DELETE CASCADE

);

CREATE TABLE themes (

id INTEGER NOT NULL,

priority INTEGER DEFAULT 1,

name VARCHAR(255) NOT NULL,

-- VARCHAR below should be TEXT, but it's not supported in oracle

description VARCHAR(255) NOT NULL,

subject\_id INTEGER NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (subject\_id) REFERENCES subjects(id) ON DELETE CASCADE

);

CREATE TABLE questions (

id INTEGER NOT NULL,

statement VARCHAR(255) NOT NULL,

PRIMARY KEY (id)

);

CREATE TABLE answers (

id INTEGER NOT NULL,

title VARCHAR(255) NOT NULL,

is\_correct SMALLINT CHECK (is\_correct in (0,1)),

priority INTEGER DEFAULT 1,

question\_id INTEGER NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (question\_id) REFERENCES questions(id) ON DELETE CASCADE

);

CREATE TABLE exams (

id INTEGER NOT NULL,

year INTEGER NOT NULL,

convocatory INTEGER NOT NULL,

subject\_id INTEGER NOT NULL,

PRIMARY KEY (id),

FOREIGN KEY (subject\_id) REFERENCES subjects(id) ON DELETE CASCADE,

CONSTRAINT unique\_exam UNIQUE (subject\_id, year, convocatory)

);

CREATE TABLE questions\_themes (

question\_id INTEGER NOT NULL,

theme\_id INTEGER NOT NULL,

FOREIGN KEY (question\_id) REFERENCES questions(id) ON DELETE CASCADE,

FOREIGN KEY (theme\_id) REFERENCES themes(id) ON DELETE CASCADE,

PRIMARY KEY (question\_id, theme\_id)

);

CREATE TABLE exams\_questions (

question\_id INTEGER NOT NULL,

exam\_id INTEGER NOT NULL,

correct\_answer\_count INTEGER DEFAULT 0,

incorrect\_answer\_count INTEGER DEFAULT 0,

unreplied\_answer\_count INTEGER DEFAULT 0,

FOREIGN KEY (question\_id) REFERENCES questions(id) ON DELETE CASCADE,

FOREIGN KEY (exam\_id) REFERENCES exams(id) ON DELETE CASCADE,

PRIMARY KEY (question\_id, exam\_id)

);

CREATE TABLE subjects\_teachers (

subject\_id INTEGER NOT NULL,

teacher\_id INTEGER NOT NULL,

FOREIGN KEY (subject\_id) REFERENCES subjects(id) ON DELETE CASCADE,

FOREIGN KEY (teacher\_id) REFERENCES teachers(id) ON DELETE CASCADE,

PRIMARY KEY (subject\_id, teacher\_id)

);

**“seeds.sql”**

--Table "grades"

INSERT INTO grades VALUES(1, 'Grado en Ingeniería Informática');

INSERT INTO grades VALUES(2, 'Ingeniería Técnica en Informática de Gestión');

INSERT INTO grades VALUES(3, 'Grado en Ingeniería Informática en Sistemas de Información');

INSERT INTO grades VALUES(4, 'Ingeniería Técnica en Informática de Sistemas');

INSERT INTO grades VALUES(5, 'Grado en Ingeniería del Software');

--Table "teachers"

INSERT INTO teachers VALUES(1, '70015789L', 'Jacinto Pérez Hermoso', 'Automática');

INSERT INTO teachers VALUES(2, '71085749F', 'Brígida Ferreira Santos', 'Robótica');

INSERT INTO teachers VALUES(3, '78014787Y', 'Tomás Fernández Marín', 'Inteligencia Artifical');

INSERT INTO teachers VALUES(4, '70032427T', 'Laura Galende Hernández', 'Informática');

INSERT INTO teachers VALUES(5, '70011497H', 'José de Pedro Santos', 'Nanotecnología');

--Table "subjects"

INSERT INTO subjects VALUES(1, '00015A', 'Algoritmia', 1);

INSERT INTO subjects VALUES(2, '00128E', 'Estructuras de Datos', 2);

INSERT INTO subjects VALUES(3, '00021I', 'Informática Teórica', 3);

INSERT INTO subjects VALUES(4, '00052O', 'Organización y gestión de empresas', 4);

INSERT INTO subjects VALUES(5, '00073U', 'Sistemas Operativos', 5);

--Table "themes"

--First subject

INSERT INTO themes VALUES(1, 1, 'Algoritmos de ordenación', 'Ordenación de un elemento específico dentro de un conjunto cualquiera de datos.', 1);

INSERT INTO themes VALUES(2, 1, 'Algoritmos de búsqueda', 'Búsqueda de un elemento específico dentro de un conjunto cualquiera de datos.', 1);

INSERT INTO themes VALUES(3, 1, 'Notación asintótica', 'Diferentes métodos para expresar el tiempo de ejecución de un algoritmo.', 1);

INSERT INTO themes VALUES(4, 1, 'Esquemas algorítmicos', 'Agrupación de los algoritmos dentro de distintos esquemas conocidos', 1);

INSERT INTO themes VALUES(5, 1, 'Análisis algorítmico', 'Realizar estimaciones teóricas para los recursos que necesita cualquier algoritmo que resuelva un problema computacional dado.', 1);

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

--Second subject

INSERT INTO themes VALUES(6, 1, 'TAD: Pilas', 'Se utilizan generalmente para simplificar ciertas operaciones de programación.', 2);

INSERT INTO themes VALUES(7, 1, 'TAD: Árboles Binarios de búsqueda', 'Ampliación del TAD Árboles Binarios, que nos permiten recuperar una información determinada.', 2);

INSERT INTO themes VALUES(8, 1, 'TAD: Grafos', 'Conjunto de nodos (también llamados vértices) y un conjunto de arcos (aristas) que establecen relaciones entre los nodos.', 2);

INSERT INTO themes VALUES(9, 1, 'TAD: Conjuntos Disjuntos', 'Conjunto basado en la idea de representación de relaciones entre los elementos del conjunto.', 2);

INSERT INTO themes VALUES(10, 1, 'TAD: Montículos Binarios', 'Tipo de implementación de colas de prioridad.', 2);

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

--Third subject

INSERT INTO themes VALUES(11, 1, 'Autómatas Finitos', 'Modelos computacionales que realiza cómputos en forma automática sobre una entrada para producir una salida.', 3);

INSERT INTO themes VALUES(12, 1, 'Lenguajes', 'Lo referente a símbolos primitivos y reglas para unir esos símbolos que están formalmente especificados.', 3);

INSERT INTO themes VALUES(13, 1, 'Gramáticas formales', 'Estructuras matemáticas con un conjunto de reglas de formación que definen las cadenas de caracteres admisibles en un determinado lenguaje formal o lengua natural.', 3);

INSERT INTO themes VALUES(14, 1, 'Máquinas abstractas', 'Modelos teóricos de un sistema computador de hardware o software usado en la teoría de autómatas.', 3);

INSERT INTO themes VALUES(15, 1, 'Máquina de Turing', 'Dispositivo que manipula símbolos sobre una tira de cinta de acuerdo a una tabla de reglas.', 3);

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

--Fourth subject

INSERT INTO themes VALUES(16, 1, 'Análisis de costo-beneficio', 'Es una lógica o razonamiento basado en el principio de obtener los mayores y mejores resultados al menor esfuerzo invertido.', 4);

INSERT INTO themes VALUES(17, 1, 'La empresa y el sistema económico', 'La empresa como estructura de producción, de asignación de recursos económicos, distribución y consumo de bienes y servicios en una economía.', 4);

INSERT INTO themes VALUES(18, 1, 'La empresa como realidad económica', 'Conjunto ordenado de factores destinados a la producción.', 4);

INSERT INTO themes VALUES(19, 1, 'La empresa como sistema', 'Teoría de sistemas aplicada a la empresa. Comportamiento, procesos de control y adaptación al entorno.', 4);

INSERT INTO themes VALUES(20, 1, 'Inversión y financiación', 'La supervivencia de una empresa depende de su acertada planificación en cuanto a su política de inversiones, financiaciones y la continuidad de éstas en el tiempo.', 4);

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

--Fivth subject

INSERT INTO themes VALUES(21, 1, 'Planificación del procesador', 'Mecanismos más comunes que poseen los sistemas operativos actuales para realizar la gestión del procesador.', 5);

INSERT INTO themes VALUES(22, 1, 'Introducción a los Sistemas Operativos', 'Funciones, objetivos, evolución y tipos, así como componentes y estructura de un Sistema Operativo.', 5);

INSERT INTO themes VALUES(23, 1, 'Descripción y control de procesos', 'Estados, implementación y control de los procesos.', 5);

INSERT INTO themes VALUES(24, 1, 'Gestión de la memoria principal', 'Conceptos fundamentales, asignación contigua de memoria, paginación y segmentación.', 5);

INSERT INTO themes VALUES(25, 1, 'Gestón de la memoria virtual', 'Paginación bajo demanda, prepaginación, reemplazo de páginas, gestión del conjunto residente, control de carga y ejemplos de gestión de memoria.', 5);

--Table "questions"

--First subject

INSERT INTO questions (id, statement) VALUES(1, '¿Cuál de estas funciones representa una cota superior asintótica para el archiconocido algoritmo de ordenación QuickSort?');

INSERT INTO questions (id, statement) VALUES(2, '¿Cuál de estas funciones representa una cota superior asintótica para el archiconocido algoritmo de búsqueda Burbuja');

INSERT INTO questions (id, statement) VALUES(3, '¿Qué métodos diferentes de notaciones asintóticas hemos estudiado en clase?');

INSERT INTO questions (id, statement) VALUES(4, '¿A qué esquema algorítmico pertenece el famoso problema de ajedrez de Las Ocho Reinas?');

INSERT INTO questions (id, statement) VALUES(5, '¿Con qué sencilla fórmula matemática se puede analizar un bucle for?');

----------------------------------------------------------------------------------------------------------------------------------------------------------------

--Second subject

INSERT INTO questions (id, statement) VALUES(6, 'Indica qué tipo de esquema sigue este TAD:');

INSERT INTO questions (id, statement) VALUES(7, '¿En qué se diferencia este TAD de los Árboles Binarios normales?');

INSERT INTO questions (id, statement) VALUES(8, '¿Qué es un Árbol de Expansión mínima?');

INSERT INTO questions (id, statement) VALUES(9, '¿Mediante qué tipo de estructuras se pueden implementar un Conjunto Disjunto?');

INSERT INTO questions (id, statement) VALUES(10, '¿Qué dos propiedades rigen cualquier Montículo Binario?');

----------------------------------------------------------------------------------------------------------------------------------------------------------------

--Third subject

INSERT INTO questions (id, statement) VALUES(11, '¿Qué es un AFD?');

INSERT INTO questions (id, statement) VALUES(12, '¿Qué es un lenguaje?');

INSERT INTO questions (id, statement) VALUES(13, '¿Qué es una gramática formal?');

INSERT INTO questions (id, statement) VALUES(14, '¿Para qué son usadas las máquinas abstractas?');

INSERT INTO questions (id, statement) VALUES(15, 'Sobre la Máquina de Turing, señala la afirmación correcta:');

----------------------------------------------------------------------------------------------------------------------------------------------------------------

--Fourth subject

INSERT INTO questions (id, statement) VALUES(16, 'Señala cuál de las siguientes afirmaciones sobre las empresas es correcta:');

INSERT INTO questions (id, statement) VALUES(17, 'Señala la verdadera:');

INSERT INTO questions (id, statement) VALUES(18, '¿Cuáles son las funciones de la empresa dentro de una economía de mercado?');

INSERT INTO questions (id, statement) VALUES(19, '¿Cuáles son los principios básicos de la Teoría de Sistemas?');

INSERT INTO questions (id, statement) VALUES(20, '¿Cuál es el objetivo único de la inversión?');

----------------------------------------------------------------------------------------------------------------------------------------------------------------

--Fivth subject

INSERT INTO questions (id, statement) VALUES(21, '¿En qué consiste la planificación del procesador?');

INSERT INTO questions (id, statement) VALUES(22, '¿Cuáles son las funciones de un Sistema Operativo?');

INSERT INTO questions (id, statement) VALUES(23, '¿Qué es un proceso?');

INSERT INTO questions (id, statement) VALUES(24, '¿Qué problemas puede ocasionar la multiprogramación?');

INSERT INTO questions (id, statement) VALUES(25, '¿En qué consiste la paginación bajo demanda?');

--Table "answers"

--First question

INSERT INTO answers VALUES(100, 'n\*log(n)', 1, 1, 1);

INSERT INTO answers VALUES(101, 'n^2', 0, 2, 1);

INSERT INTO answers VALUES(102, 'n', 0, 3, 1);

INSERT INTO answers VALUES(103, 'n^3', 0, 4, 1);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Second question

INSERT INTO answers VALUES(104, 'n\*log(n)', 0, 1, 2);

INSERT INTO answers VALUES(105, 'n^2', 1, 2, 2);

INSERT INTO answers VALUES(106, 'n^3', 0, 3, 2);

INSERT INTO answers VALUES(107, 'n', 0, 4, 2);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Third question

INSERT INTO answers VALUES(108, 'El orden de y theta', 0, 1, 3);

INSERT INTO answers VALUES(109, 'El orden de y omega', 0, 2, 3);

INSERT INTO answers VALUES(110, 'El orden de, omega y theta', 1, 3, 3);

INSERT INTO answers VALUES(111, 'Ninguna de las anteriores', 0, 4, 3);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Fourth question

INSERT INTO answers VALUES(112, 'Esquema Voraz', 0, 1, 4);

INSERT INTO answers VALUES(113, 'Divide y vencerás', 0, 2, 4);

INSERT INTO answers VALUES(114, 'Las tres restantes no son ciertas', 0, 3, 4);

INSERT INTO answers VALUES(115, 'Backtracking', 1, 4, 4);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Fifth question

INSERT INTO answers VALUES(116, 'Serie aritmética', 1, 1, 5);

INSERT INTO answers VALUES(117, 'Serie geométrica', 0, 2, 5);

INSERT INTO answers VALUES(118, 'Integrales', 0, 3, 5);

INSERT INTO answers VALUES(119, 'Derivadas', 0, 4, 5);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Sixth question

INSERT INTO answers VALUES(200, 'FIFO', 0, 1, 6);

INSERT INTO answers VALUES(201, 'LIFO', 1, 2, 6);

INSERT INTO answers VALUES(202, 'LILO', 0, 3, 6);

INSERT INTO answers VALUES(203, 'Ninguna de las anteriores', 0, 4, 6);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Seventh question

INSERT INTO answers VALUES(204, 'En nada, son iguales, pero con distinto nombre', 0, 1, 7);

INSERT INTO answers VALUES(205, 'Poseen un campo extra con un puntero', 0, 2, 7);

INSERT INTO answers VALUES(206, 'Poseen un campo extra que contiene una clave', 1, 3, 7);

INSERT INTO answers VALUES(207, 'Todas las anteriores son correctas', 0, 4, 7);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Eigth question

INSERT INTO answers VALUES(208, 'Un tipo de Conjunto Disjunto', 0, 1, 8);

INSERT INTO answers VALUES(209, 'Un tipo de Montículo Binario', 0, 2, 8);

INSERT INTO answers VALUES(210, 'Un tipo de Árbol Binario', 0, 3, 8);

INSERT INTO answers VALUES(211, 'Un grado cuyos vértices están unidos por las aristas de menor coste (de menos peso)', 1, 4, 8);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Ninth question

INSERT INTO answers VALUES(212, 'Listas enlazadas, árboles binarios y matrices', 1, 1, 9);

INSERT INTO answers VALUES(213, 'Listas enlazadas y árboles binarios', 0, 2, 9);

INSERT INTO answers VALUES(214, 'Únicamente mediantes matrices', 0, 3, 9);

INSERT INTO answers VALUES(215, 'listas enlazadas y matrices', 0, 4, 9);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Tenth question

INSERT INTO answers VALUES(216, 'Propiedas de recorrido y orden', 0, 1, 10);

INSERT INTO answers VALUES(217, 'Propiedas de estructura y orden', 1, 2, 10);

INSERT INTO answers VALUES(218, 'Propiedad de recorrido y estructura', 0, 3, 10);

INSERT INTO answers VALUES(219, 'Todas las demás son correctas', 0, 4, 10);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Eleventh question

INSERT INTO answers VALUES(300, 'Autómata Finito No Determinsita', 0, 1, 11);

INSERT INTO answers VALUES(301, 'Máquina de Turing', 0, 2, 11);

INSERT INTO answers VALUES(302, 'Autómata Finito Determinista', 1, 3, 11);

INSERT INTO answers VALUES(303, 'Conjunto Universal', 0, 4, 11);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Twelfth question

INSERT INTO answers VALUES(304, 'Ninguna de las siguientes', 0, 1, 12);

INSERT INTO answers VALUES(305, 'Forma de representar información basada en un conjunto finito de sólo símbolos', 0, 2, 12);

INSERT INTO answers VALUES(306, 'Forma de representar información basada en un conjunto infinito de signos o símbolos', 0, 3, 12);

INSERT INTO answers VALUES(307, 'Forma de representar información basada en un conjunto finito de signos o símbolos', 1, 4, 12);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Thirteenth question

INSERT INTO answers VALUES(308, 'Lenguaje descrito mediante un formalismo matemático', 1, 1, 13);

INSERT INTO answers VALUES(309, 'Lenguaje descrito mediante un formalismo físico', 0, 2, 13);

INSERT INTO answers VALUES(310, 'Lenguaje abstracto sobre cualquier cosa', 0, 3, 13);

INSERT INTO answers VALUES(311, 'Ninguna de las anteriores', 0, 4, 13);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Fourteenth question

INSERT INTO answers VALUES(312, 'Para compilar programas escritos en lenguaje de programación', 0, 1, 14);

INSERT INTO answers VALUES(313, 'En experimentos de pensamiento', 1, 2, 14);

INSERT INTO answers VALUES(314, 'Para poder escribir e-mails', 0, 3, 14);

INSERT INTO answers VALUES(315, 'Todas son ciertas', 0, 4, 14);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Fifteenth question

INSERT INTO answers VALUES(316, 'No puede simular ningún algoritmo. No está diseñada para eso', 0, 1, 15);

INSERT INTO answers VALUES(317, 'Puede simular casi cualquier algoritmo', 0, 2, 15);

INSERT INTO answers VALUES(318, 'Puede simular cualquier algoritmo', 1, 3, 15);

INSERT INTO answers VALUES(319, 'Todas las anteriores son ciertas', 0, 4, 15);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Sixteenth question

INSERT INTO answers VALUES(400, 'Debe de maximizar beneficios minimizando costes', 1, 1, 16);

INSERT INTO answers VALUES(401, 'Debe de minimizar beneficios maximizando costes', 0, 2, 16);

INSERT INTO answers VALUES(402, 'Debe de maximizar beneficios maximizando costes', 0, 3, 16);

INSERT INTO answers VALUES(403, 'Debe de minimizar beneficios minimizando costes', 0, 4, 16);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Seventeenth question

INSERT INTO answers VALUES(404, 'La empresa no forma parte de la Economía', 0, 1, 17);

INSERT INTO answers VALUES(405, 'La empresa está integrada enteramente dentro de la Economía', 1, 2, 17);

INSERT INTO answers VALUES(406, 'La empresa está integrada en la Economía únicamente a nivel de producción', 0, 3, 17);

INSERT INTO answers VALUES(407, 'Todas son ciertas', 0, 4, 17);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Eighteenth question

INSERT INTO answers VALUES(408, 'Generar bienes y servicios', 0, 1, 18);

INSERT INTO answers VALUES(409, 'Anticipar producto obtenido', 0, 2, 18);

INSERT INTO answers VALUES(410, 'Todas son verdaderas', 1, 3, 18);

INSERT INTO answers VALUES(411, 'Asumir riesgos', 0, 4, 18);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Nineteenth question

INSERT INTO answers VALUES(412, 'Independencia', 0, 1, 19);

INSERT INTO answers VALUES(413, 'Transformación', 0, 2, 19);

INSERT INTO answers VALUES(414, 'Entropía', 0, 3, 19);

INSERT INTO answers VALUES(415, 'Todas las anterires son correctas', 1, 4, 19);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Twentieth question

INSERT INTO answers VALUES(416, 'Determinar todo lo necesario para poner en marcha una empresa', 1, 1, 20);

INSERT INTO answers VALUES(417, 'Tener en cuenta las amortizaciones en el plan de viabilidad de una empresa', 0, 2, 20);

INSERT INTO answers VALUES(418, 'Calcular los costes de financiación', 0, 3, 20);

INSERT INTO answers VALUES(419, 'Todas son válidas', 0, 4, 20);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Twenty first question

INSERT INTO answers VALUES(500, 'Formatear los datos contenidos en un Disco Duro', 0, 1, 21);

INSERT INTO answers VALUES(501, 'Varias personas utilizando un mismo Sistema a la vez', 0, 2, 21);

INSERT INTO answers VALUES(502, 'Gestión de los mecanismos IPC del Sistema Operativo', 0, 3, 21);

INSERT INTO answers VALUES(503, 'Dar un buen servicio a los procesos de un Sistema', 1, 4, 21);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Twenty second question

INSERT INTO answers VALUES(504, 'Todas las restantes', 1, 1, 22);

INSERT INTO answers VALUES(505, 'Abstracción del hardware', 0, 2, 22);

INSERT INTO answers VALUES(506, 'Detección de errores así como su posible solución', 0, 3, 22);

INSERT INTO answers VALUES(507, 'Puesta en ejecución de programas', 0, 4, 22);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Twenty third question

INSERT INTO answers VALUES(508, 'Algo no inherente al Sistema Operativo', 0, 1, 23);

INSERT INTO answers VALUES(509, 'Una instancia de ejecución de un programa', 1, 2, 23);

INSERT INTO answers VALUES(510, 'Un componente del Disco Duro', 0, 3, 23);

INSERT INTO answers VALUES(511, 'Todas las anteriores son respuestas válidas', 0, 4, 23);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Twenty fourth question

INSERT INTO answers VALUES(512, 'Compartición de código y/o datos', 0, 1, 24);

INSERT INTO answers VALUES(513, 'Reubicación del código', 0, 2, 24);

INSERT INTO answers VALUES(514, 'Todas las demás', 1, 3, 24);

INSERT INTO answers VALUES(515, 'Protección de la memoria de cada proceso', 0, 4, 24);

-----------------------------------------------------------------------------------------------------------------------------------------------------------

--Twenty fivth question

INSERT INTO answers VALUES(516, 'En dividir el programa en páginas/marcos y cargar en memoria secundaria todos', 0, 1, 25);

INSERT INTO answers VALUES(517, 'En dividir el programa en páginas/marcos y cargar en memoria secundaria sólo los que se estén usando', 0, 2, 25);

INSERT INTO answers VALUES(518, 'En dividir el programa en páginas/marcos y cargar en memoria principal todos', 0, 3, 25);

INSERT INTO answers VALUES(519, 'En dividir el programa en páginas/marcos y cargar en memoria principal sólo los que se estén usando', 1, 4, 25);

--Table "exams"

INSERT INTO exams VALUES(1, 2007, 1, 1);

INSERT INTO exams VALUES(2, 2008, 2, 2);

INSERT INTO exams VALUES(3, 2005, 1, 3);

INSERT INTO exams VALUES(4, 2010, 1, 4);

INSERT INTO exams VALUES(5, 2015, 2, 5);

--Table "questions\_themes"

INSERT INTO questions\_themes VALUES(1, 1);

INSERT INTO questions\_themes VALUES(2, 2);

INSERT INTO questions\_themes VALUES(3, 3);

INSERT INTO questions\_themes VALUES(4, 4);

INSERT INTO questions\_themes VALUES(5, 5);

INSERT INTO questions\_themes VALUES(6, 6);

INSERT INTO questions\_themes VALUES(7, 7);

INSERT INTO questions\_themes VALUES(8, 8);

INSERT INTO questions\_themes VALUES(9, 9);

INSERT INTO questions\_themes VALUES(10, 10);

INSERT INTO questions\_themes VALUES(11, 11);

INSERT INTO questions\_themes VALUES(12, 12);

INSERT INTO questions\_themes VALUES(13, 13);

INSERT INTO questions\_themes VALUES(14, 14);

INSERT INTO questions\_themes VALUES(15, 15);

INSERT INTO questions\_themes VALUES(16, 16);

INSERT INTO questions\_themes VALUES(17, 17);

INSERT INTO questions\_themes VALUES(18, 18);

INSERT INTO questions\_themes VALUES(19, 19);

INSERT INTO questions\_themes VALUES(20, 20);

INSERT INTO questions\_themes VALUES(21, 21);

INSERT INTO questions\_themes VALUES(22, 22);

INSERT INTO questions\_themes VALUES(23, 23);

INSERT INTO questions\_themes VALUES(24, 24);

INSERT INTO questions\_themes VALUES(25, 25);

--Table "exams\_questions"

INSERT INTO exams\_questions VALUES(1, 1, 1, 0, 0);

INSERT INTO exams\_questions VALUES(2, 2, 0, 1, 0);

INSERT INTO exams\_questions VALUES(3, 3, 0, 0, 1);

INSERT INTO exams\_questions VALUES(4, 4, 1, 0, 0);

INSERT INTO exams\_questions VALUES(5, 1, 0, 1, 0);

INSERT INTO exams\_questions VALUES(6, 2, 0, 0, 1);

INSERT INTO exams\_questions VALUES(7, 3, 1, 0, 0);

INSERT INTO exams\_questions VALUES(8, 4, 0, 1, 0);

INSERT INTO exams\_questions VALUES(9, 5, 0, 0, 1);

INSERT INTO exams\_questions VALUES(10, 1, 1, 0, 0);

INSERT INTO exams\_questions VALUES(11, 2, 0, 1, 0);

INSERT INTO exams\_questions VALUES(12, 3, 0, 0, 1);

INSERT INTO exams\_questions VALUES(13, 4, 1, 0, 0);

INSERT INTO exams\_questions VALUES(14, 5, 0, 1, 0);

INSERT INTO exams\_questions VALUES(15, 1, 0, 0, 1);

INSERT INTO exams\_questions VALUES(16, 2, 1, 0, 0);

INSERT INTO exams\_questions VALUES(17, 3, 0, 1, 0);

INSERT INTO exams\_questions VALUES(18, 4, 0, 0, 1);

INSERT INTO exams\_questions VALUES(19, 5, 1, 0, 0);

INSERT INTO exams\_questions VALUES(20, 1, 0, 1, 0);

INSERT INTO exams\_questions VALUES(21, 2, 0, 0, 1);

INSERT INTO exams\_questions VALUES(22, 3, 1, 0, 0);

INSERT INTO exams\_questions VALUES(23, 4, 0, 1, 0);

INSERT INTO exams\_questions VALUES(24, 5, 0, 0, 1);

INSERT INTO exams\_questions VALUES(25, 1, 1, 0, 0);

--Table "subjects\_teachers"

INSERT INTO subjects\_teachers VALUES(1, 1);

INSERT INTO subjects\_teachers VALUES(2, 2);

INSERT INTO subjects\_teachers VALUES(3, 3);

INSERT INTO subjects\_teachers VALUES(4, 4);

INSERT INTO subjects\_teachers VALUES(5, 5);

**“derived\_and\_triggers.sql”**

ALTER TABLE questions ADD exam\_count INTEGER DEFAULT 0;

CREATE OR REPLACE TRIGGER question\_exam\_insert AFTER INSERT ON exams\_questions

FOR EACH ROW

BEGIN

UPDATE questions SET exam\_count = exam\_count + 1 WHERE id = :new.question\_id;

END;

/

CREATE OR REPLACE TRIGGER question\_exam\_delete AFTER DELETE ON exams\_questions

FOR EACH ROW

BEGIN

UPDATE questions SET exam\_count = exam\_count - 1 WHERE id = :old.question\_id;

END;

/

CREATE OR REPLACE TRIGGER question\_exam\_insert AFTER UPDATE OF question\_id ON exams\_questions

FOR EACH ROW

BEGIN

UPDATE questions SET exam\_count = exam\_count - 1 WHERE id = :old.question\_id;

UPDATE questions SET exam\_count = exam\_count + 1 WHERE id = :new.question\_id;

END;

/

**“view.sql”**

--View 'questions\_counter'

CREATE OR REPLACE VIEW questions\_counter AS

SELECT subjects\_teachers.teacher\_id, themes.subject\_id, themes.id AS theme\_id,

-- We want to count all the themes, not just those who have at least one question

(SELECT COUNT(qt.question\_id) FROM questions\_themes qt WHERE qt.theme\_id = themes.id) AS question\_count

FROM subjects\_teachers, themes

WHERE subjects\_teachers.subject\_id = themes.subject\_id

GROUP BY subjects\_teachers.teacher\_id, themes.subject\_id, themes.id;

GRANT ALL PRIVILEGES ON questions\_counter TO PUBLIC;

**“oracle-drops.sql”**

DROP TABLE questions\_themes CASCADE CONSTRAINTS PURGE;

DROP TABLE exams\_questions CASCADE CONSTRAINTS PURGE;

DROP TABLE subjects\_teachers CASCADE CONSTRAINTS PURGE;

DROP TABLE exams CASCADE CONSTRAINTS PURGE;

DROP TABLE answers CASCADE CONSTRAINTS PURGE;

DROP TABLE questions CASCADE CONSTRAINTS PURGE;

DROP TABLE themes CASCADE CONSTRAINTS PURGE;

DROP TABLE subjects CASCADE CONSTRAINTS PURGE;

DROP TABLE teachers CASCADE CONSTRAINTS PURGE;

DROP TABLE grades CASCADE CONSTRAINTS PURGE;

Ficheros de SQL inmerso en C

**“commands.c”**

#include "commands.h"

// Gets a pointer to a command, to allow implementing custom behavior

// returns null if not found.

//

// NOTE: Command arrays must end in a command with no name

const struct command\*

command\_by\_name(const struct command\* commands,

const char\* const name) {

size\_t i = 0;

if ( ! commands )

return NULL;

while ( 1 ) {

// The list of commands is over and we didn't find

// anything

if ( ! commands[i].name )

return NULL;

if ( strcmp(commands[i].name, name) == 0 )

return commands + i;

++i;

}

return NULL;

}

// Default help command if implementation doesn't provide one

int command\_default\_help(const char\* const program\_name,

const struct command\* commands,

int argc,

char\*\* argv) {

size\_t i = 0;

const struct command\* command;

if ( ! commands )

return 1;

if ( argc == 0 ) {

printf("Usage: %s <command> [args...]\n", program\_name);

printf("Available commands:\n");

while ( 1 ) {

if ( ! commands[i].name )

break;

printf("\t%s\t%s\n", commands[i].name, commands[i].description);

++i;

}

printf("To see extended info on each command run:\n");

printf("\t%s help <command>\n", program\_name);

return 1;

} else if ( argc > 1 ) {

printf("%s: unrecognized argument: \"%s\"\n", program\_name, argv[1]);

return 1;

}

command = command\_by\_name(commands, argv[0]);

if ( ! command ) {

printf("%s: command \"%s\" not found\n", program\_name, argv[0]);

return 1;

}

printf("%s: %s\n", command->name, command->description);

if ( command->longdesc )

printf("\n%s\n", command->longdesc);

return 1;

}

// Tries to execute a command in the command list. The command name must be

// in argv[1]. Example usage:

//

// int main(int argc, char\*\* argv) {

// return exec\_comand(my\_commands, argc, argv);

// }

int command\_exec(const struct command\* commands, int argc, char\*\* argv) {

const struct command\* command;

if ( argc == 1 ) {

command = command\_by\_name(commands, "help");

if ( command )

return command->executor(argc - 1, argv + 1);

return command\_default\_help(argv[0], commands, argc - 1, argv + 1);

}

command = command\_by\_name(commands, argv[1]);

if ( command )

return command->executor(argc - 2, argv + 2);

if ( strcmp(argv[1], "help") == 0 )

return command\_default\_help(argv[0], commands, argc - 2, argv + 2);

printf("%s: command \"%s\" not found\n", argv[0], argv[1]);

printf("\trun `%s help` for a list of options\n", argv[0]);

return 1;

}

**“commands.h”**

/\*\*

\* Simple command/subcommand library

\*

\*

\* @license MIT

\*

\*

\*/

#ifndef COMMANDS\_H\_

#define COMMANDS\_H\_

#include <stdio.h>

#include <string.h>

typedef int (\*command\_callback\_t)(int, char\*\*);

// A command is described by:

// - name: The name which should be passed as a second parameter

// - executor: The function that will be executed if the command

// matches. Receives the argc of the program minus two, and the rest

// of argv

// - description: Brief description, to show in help messages.

// - longdesc: Long description to show when you run `help <command>

struct command {

const char\* const name;

int (\*executor)(int, char\*\*);

const char\* const description;

const char\* const longdesc;

};

// Gets a pointer to a command, to allow implementing custom behavior

// returns null if not found.

//

// NOTE: Command arrays must end in a command with no name

const struct command\*

command\_by\_name(const struct command\* commands,

const char\* const name);

// Default help command if implementation doesn't provide one

int command\_default\_help(const char\* const program\_name,

const struct command\* commands,

int argc,

char\*\* argv);

// Tries to execute a command in the command list. The command name must be

// in argv[1]. Example usage:

//

// int main(int argc, char\*\* argv) {

// return exec\_comand(my\_commands, argc, argv);

// }

int command\_exec(const struct command\* commands, int argc, char\*\* argv);

#endif

**“answer.h”**

#ifndef ANSWER\_H\_

#define ANSWER\_H\_

int answer(int argc, char\*\* argv);

#endif

**“answer.sc”**

#include "answer.h"

#include "common.h"

int list\_answers(int argc, char\*\* argv);

int answer(int argc, char\*\* argv) {

EXEC SQL WHENEVER SQLERROR DO handle\_error();

EXEC SQL BEGIN DECLARE SECTION;

int id;

int is\_correct = 0;

int question\_id;

varchar title[255] = {0};

EXEC SQL END DECLARE SECTION;

if ( argc < 1 || argc > 4 )

ARGUMENT\_ERROR();

/// Add a new question

if ( strcmp(argv[0], "-a") == 0 ) {

argc--; argv++;

if ( argc < 2 || argc > 3 )

ARGUMENT\_ERROR();

if ( argc == 3 ) {

if ( strcmp(argv[2], "--correct") == 0 || strcmp(argv[2], "-c") == 0 )

is\_correct = 1;

else

ARGUMENT\_ERROR();

}

question\_id = atoi(argv[0]);

COPY\_TO\_VARCHAR(title, argv[1], 255);

EXEC SQL INSERT INTO answers (question\_id, title, is\_correct) VALUES (:question\_id, :title, :is\_correct)

RETURNING id INTO :id;

APPCOM\_RET\_INT(id);

return 0;

/\* Delete \*/

} else if ( strcmp(argv[0], "-d") == 0 ) {

argc--; argv++;

if ( argc != 1 )

ARGUMENT\_ERROR();

id = atoi(argv[0]);

EXEC SQL DELETE FROM answers WHERE id = :id;

return 0;

} else if ( strcmp(argv[0], "-l") == 0 ) {

argc--; argv++;

return list\_answers(argc, argv);

}

ARGUMENT\_ERROR();

}

int list\_answers(int argc, char\*\* argv) {

EXEC SQL WHENEVER SQLERROR DO handle\_error();

EXEC SQL BEGIN DECLARE SECTION;

int question\_id;

int id;

char title[256] = {0};

short is\_correct;

EXEC SQL END DECLARE SECTION;

if ( argc != 1 )

ARGUMENT\_ERROR();

question\_id = atoi(argv[0]);

EXEC SQL DECLARE answers\_cursor CURSOR FOR

SELECT id, title, is\_correct FROM answers

WHERE question\_id = :question\_id

ORDER BY priority ASC;

EXEC SQL OPEN answers\_cursor;

while ( 1 ) {

EXEC SQL FETCH answers\_cursor INTO :id, :title, :is\_correct;

if ( SQLCODE == NOT\_FOUND )

break;

printf(" [%c] %s (%d)\n", is\_correct ? 'x' : ' ', str\_trim\_right(title), id);

}

EXEC SQL CLOSE answers\_cursor;

return 1;

}

**“app.sc”**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "question.h"

#include "answer.h"

#include "exam.h"

#include "questions\_themes.h"

#include "interactive.h"

#include "common.h"

#include "../lib/commands.h"

#ifndef POSTGRES

char SQLSTATE[6];

#define SQLCODE sqlca.sqlcode

#endif

/\* The list of our supported commands \*/

struct command commands[] = {

{ "question", question, "manages questions",

"Usage: question [args...]\n"

"\t-a\tcreates a new question\t-a <statement>\n"

"\t-d\tdeletes a question\t-d <id>\n"

"\t-e\tedits a question\t-e <id> <statement>\n"

"\t-s\tshows a question\t-s <id>\n"

"\t-l\tlists all questions\t-l [<theme\_id>] [--detailed]\n"

"\t-t\tlists all questions by theme and subject\t-t [--detailed]\n"

"Prints the question id on success to stdout." },

{ "answer", answer, "manages answers",

"Usage: answer [args...]\n"

"\t-a\tcreates an answer\t-a <question\_id> <statement> [--correct]\n"

"\t-l\tlist answers for a question\t-l <question\_id>\n"

"\t-d\tdeletes a question\t-d <id>" },

{ "exam", exam, "manages exams",

"Usage: answer [args...]\n"

"\t-l\tlist exams for a question\t-l <question\_id>\n" },

{ "questions\_themes", questions\_themes, "Relation between questions and themes",

"\t-a\tads a new relationship\t-a <question\_id> <theme\_id>\n" },

{ "interactive", interactive, "Open an interactive session" },

{ NULL, NULL, NULL, NULL }

};

void handle\_error() {

fprintf(stderr, "[ERROR] %s | SQLSTATE(%s), SQLCODE(%ld)\n", sqlca.sqlerrm.sqlerrmc, SQLSTATE, SQLCODE);

exit(1);

}

int main(int argc, char\*\* argv) {

EXEC SQL WHENEVER SQLERROR DO handle\_error();

#ifndef POSTGRES

EXEC SQL BEGIN DECLARE SECTION;

char oracle\_id[] = "/";

EXEC SQL END DECLARE SECTION;

EXEC SQL CONNECT :oracle\_id;

#else

EXEC SQL CONNECT TO exams;

#endif

int ret = command\_exec(commands, argc, argv);

if ( appcom.ret.int\_value )

printf("%d\n", appcom.ret.int\_value);

EXEC SQL COMMIT;

return ret;

}

**“common.c”**

#include "common.h"

#include <stdlib.h>

#include <stdio.h>

#include <ctype.h>

struct appcom appcom = { { 0, "" } };

void get\_str(char\* buffer, size\_t size) {

printf("> ");

fgets(buffer, size, stdin);

buffer[size - 1] = '\0';

str\_trim\_right(buffer);

}

char\* str\_trim\_right(char\* input) {

size\_t len = strlen(input);

while ( len-- )

if ( ! isspace(input[len]) )

break;

input[len + 1] = '\0';

return input;

}

char\* str\_copy(const char\* input) {

size\_t len = strlen(input);

char\* ret = (char\*) malloc(len + 1);

ret[len] = '\0';

memcpy(ret, input, len);

return ret;

}

char get\_bool() {

char resp;

do {

printf("(y/n): ");

resp = tolower(getchar());

FLUSH\_STDIN();

} while ( ! ( resp == 'y' || resp == 'n' ) );

if ( resp == 'y' )

return 1;

return 0;

}

int get\_int() {

int ret;

printf("> ");

scanf("%d", &ret);

FLUSH\_STDIN();

return ret;

}

**“common.h”**

#ifndef MACROS\_H\_

#define MACROS\_H\_

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <sqlca.h>

#ifndef SQLCODE

#define SQLCODE sqlca.sqlcode

#endif

#define APPCOM\_RET\_STR\_MAX 70

struct appcom {

struct {

int int\_value;

char str\_value[APPCOM\_RET\_STR\_MAX];

} ret;

};

void handle\_error();

extern struct appcom appcom;

#define APPCOM\_RET\_INT(val) do { \

appcom.ret.int\_value = val; \

snprintf(appcom.ret.str\_value, APPCOM\_RET\_STR\_MAX, "%d", val); \

} while (0)

#ifdef POSTGRES

# define NOT\_FOUND ECPG\_NOT\_FOUND

#else

# define NOT\_FOUND 1403

#endif

#define ARGUMENT\_ERROR() do { \

fprintf(stderr, "Error: Unexpected number of arguments.\n"); \

fprintf(stderr, "Run with `help` to know more\n"); \

exit(1); \

} while (0)

#define FLUSH\_STDIN() do { \

while ( getchar() != '\n' ) {}; \

} while ( 0 )

#define COPY\_TO\_VARCHAR(vchar, str, \_len) do { \

strncpy(vchar.arr, str, \_len); \

vchar.arr[\_len - 1] = '\0'; \

vchar.len = strlen(vchar.arr); \

} while ( 0 )

char\* str\_trim\_right(char\*);

char\* str\_copy(const char\*);

void get\_str(char\*, size\_t);

char get\_bool();

int get\_int();

/\* This must be the last macro to keep PRO\*C happy \*/

#define CALL(fn, ...) do { \

int argc\_\_ = 0; \

char\* argv\_\_[] = { \_\_VA\_ARGS\_\_, NULL }; \

char\*\* argvp\_\_ = argv\_\_; \

while ( \*argvp\_\_++ ) argc\_\_++; \

fn(argc\_\_, argv\_\_); \

} while ( 0 )

#endif

**“exam.h”**

#ifndef EXAM\_H\_

#define EXAM\_H\_

int exam(int argc, char\*\* argv);

#endif

**“exam.sc”**

#include "exam.h"

#include "common.h"

int list\_exams(int argc, char\*\* argv);

int exam(int argc, char\*\* argv) {

if ( argc != 2 )

ARGUMENT\_ERROR();

if ( strcmp(argv[0], "-l") == 0 ) {

argc--; argv++;

return list\_exams(argc, argv);

}

ARGUMENT\_ERROR();

}

int list\_exams(int argc, char\*\* argv) {

EXEC SQL WHENEVER SQLERROR DO handle\_error();

EXEC SQL BEGIN DECLARE SECTION;

int question\_id;

int id;

int exam\_year;

int convocatory;

int correct;

int incorrect;

int unreplied;

char subject\_name[256] = {0};

EXEC SQL END DECLARE SECTION;

if ( argc != 1 )

ARGUMENT\_ERROR();

int count = 0;

question\_id = atoi(argv[0]);

EXEC SQL DECLARE exams\_cursor CURSOR FOR

SELECT exams.id, exams.year, exams.convocatory, subjects.name, exams\_questions.correct\_answer\_count, exams\_questions.incorrect\_answer\_count, exams\_questions.unreplied\_answer\_count

FROM exams, exams\_questions, subjects

WHERE exams\_questions.question\_id = :question\_id AND

exams\_questions.exam\_id = exams.id AND subjects.id = exams.subject\_id

ORDER BY exams.year, exams.convocatory ASC;

EXEC SQL OPEN exams\_cursor;

while ( 1 ) {

EXEC SQL FETCH exams\_cursor INTO :id, :exam\_year, :convocatory, :subject\_name, :correct, :incorrect, :unreplied;

if ( SQLCODE == NOT\_FOUND )

break;

++count;

printf("%d %s (%d)\n", exam\_year, str\_trim\_right(subject\_name), convocatory);

printf(" - Total: %d\n", correct + incorrect + unreplied);

printf(" - Correct: %d\n", correct);

printf(" - Incorrect: %d\n", incorrect);

printf(" - Unreplied: %d\n", unreplied);

}

EXEC SQL CLOSE exams\_cursor;

printf("Total: %d\n", count);

return 0;

}

**“interactive.h”**

#ifndef INTERACTIVE\_H\_

#define INTERACTIVE\_H\_

int interactive(int argc, char\*\* argv);

#endif

**“interactive.sc”**

#include "interactive.h"

#include "common.h"

#include "question.h"

#include "answer.h"

#include "questions\_themes.h"

#include <stdio.h>

#include <ctype.h>

void crear\_pregunta() {

char buffer[256] = {0};

char resp;

char\* question\_id = NULL;

printf("Introduce un enunciado: \n");

get\_str(buffer, sizeof(buffer));

CALL(question, "-a", buffer);

printf("Pregunta creada correctamente (id: %d)\n", appcom.ret.int\_value);

question\_id = str\_copy(appcom.ret.str\_value);

while ( 1 ) {

printf("Quieres introducir una respuesta? ");

resp = get\_bool();

if ( ! resp )

break;

printf("Introduce la respuesta:\n");

get\_str(buffer, sizeof(buffer));

printf("Es una respuesta correcta?\n");

resp = get\_bool();

if ( resp )

CALL(answer, "-a", question\_id, buffer, "--correct");

else

CALL(answer, "-a", question\_id, buffer);

printf("Respuesta creada correctamente (id: %d)\n", appcom.ret.int\_value);

}

free(question\_id);

}

void listar\_por\_tema() {

CALL(question, "-t");

}

void consultar\_pregunta() {

char id[20];

printf("Introduce el id de la pregunta: \n");

get\_str(id, sizeof(id));

CALL(question, "-s", id);

}

void asignar\_tema() {

char question\_id[20];

char theme\_id[20];

printf("Introduce la id de la pregunta: \n");

get\_str(question\_id, sizeof(question\_id));

printf("Introduce la id del tema: \n");

get\_str(theme\_id, sizeof(theme\_id));

CALL(questions\_themes, "-a", question\_id, theme\_id);

printf("Relación añadida correctamente\n");

}

void commit() {

EXEC SQL WHENEVER SQLERROR DO handle\_error();

EXEC SQL COMMIT;

}

void exit\_() {

char resp;

printf("Guardar los cambios? ");

resp = get\_bool();

if ( resp )

commit();

exit(0);

}

struct menu\_option {

const char\* text;

void (\*fn)(void);

};

const struct menu\_option options[] = {

{ "Dar de alta una pregunta", crear\_pregunta },

{ "Listar preguntas por tema y asignatura", listar\_por\_tema },

{ "Consulta una pregunta", consultar\_pregunta },

{ "Asignar un tema a una pregunta", asignar\_tema },

{ "Confirmar cambios", commit },

{ "Salir", exit\_ },

{ NULL, NULL }

};

int interactive(int argc, char\*\* argv) {

size\_t i;

int chosen;

if ( argc != 0 )

ARGUMENT\_ERROR();

while ( 1 ) {

i = 0;

while ( options[i].text ) {

i++;

printf("%zu) %s\n", i, options[i - 1].text);

}

chosen = 0;

do {

chosen = get\_int();

} while ( chosen < 1 || chosen > i );

options[chosen - 1].fn();

printf("\n\n");

}

return 0;

}

**“question.h”**

#ifndef QUESTION\_H\_

#define QUESTION\_H\_

int question(int argc, char\*\* argv);

#endif

**“question.sc”**

#include "common.h"

#include "question.h"

#include "answer.h"

#include "exam.h"

int list\_questions(int, char\*\*);

int show\_question(int argc, char\*\* argv);

int list\_questions\_by\_theme(int argc, char\*\* argv);

int question(int argc, char\*\* argv) {

EXEC SQL WHENEVER SQLERROR DO handle\_error();

EXEC SQL BEGIN DECLARE SECTION;

int id;

varchar statement[255] = {0};

EXEC SQL END DECLARE SECTION;

if ( argc < 1 || argc > 4 )

ARGUMENT\_ERROR();

/\* Add a new question \*/

if ( strcmp(argv[0], "-a") == 0 ) {

argc--; argv++;

if ( argc != 1 )

ARGUMENT\_ERROR();

COPY\_TO\_VARCHAR(statement, argv[0], 255);

EXEC SQL INSERT INTO questions (statement) VALUES (:statement)

RETURNING id INTO :id;

APPCOM\_RET\_INT(id);

return 0;

/\* Delete it \*/

} else if ( strcmp(argv[0], "-d") == 0 ) {

argc--; argv++;

if ( argc != 1 )

ARGUMENT\_ERROR();

id = atoi(argv[0]);

EXEC SQL DELETE FROM questions WHERE id = :id;

return 0;

/\* Update \*/

} else if ( strcmp(argv[0], "-e") == 0 ) {

argc--; argv++;

if ( argc != 2 )

ARGUMENT\_ERROR();

id = atoi(argv[0]);

COPY\_TO\_VARCHAR(statement, argv[1], 255);

EXEC SQL UPDATE questions SET statement = :statement WHERE id = :id;

return 0;

/\* List \*/

} else if ( strcmp(argv[0], "-l") == 0 ) {

argc--; argv++;

return list\_questions(argc, argv);

/\* List by theme \*/

} else if ( strcmp(argv[0], "-t") == 0 ) {

argc--; argv++;

return list\_questions\_by\_theme(argc, argv);

} else if ( strcmp(argv[0], "-s") == 0 ) {

argc--; argv++;

return show\_question(argc, argv);

}

ARGUMENT\_ERROR();

}

/// Lists all questions

int list\_questions(int argc, char\*\* argv) {

EXEC SQL WHENEVER SQLERROR DO handle\_error();

EXEC SQL BEGIN DECLARE SECTION;

int id;

int theme\_id;

char statement[256] = {0};

char question\_id\_str[20];

EXEC SQL END DECLARE SECTION;

unsigned char detailed\_mode = 0;

EXEC SQL WHENEVER SQLERROR DO handle\_error();

int count = 0;

if ( argc > 2 )

ARGUMENT\_ERROR();

if ( argc && strcmp(argv[argc - 1], "--detailed") == 0 )

detailed\_mode = 1;

// If we passed two args and the detailed flag is not found...

else if ( argc == 2 )

ARGUMENT\_ERROR();

EXEC SQL DECLARE questions\_cursor CURSOR FOR

SELECT id, statement FROM questions;

EXEC SQL DECLARE questions\_themes\_cursor CURSOR FOR

SELECT questions.id, questions.statement FROM questions, questions\_themes WHERE questions\_themes.question\_id = questions.id AND questions\_themes.theme\_id = :theme\_id;

if ( argc == 0 || (argc == 1 && detailed\_mode) ) {

EXEC SQL OPEN questions\_cursor;

} else {

theme\_id = atoi(argv[0]);

EXEC SQL OPEN questions\_themes\_cursor;

}

while ( 1 ) {

if ( argc == 0 || (argc == 1 && detailed\_mode) )

EXEC SQL FETCH questions\_cursor INTO :id, :statement;

else

EXEC SQL FETCH questions\_themes\_cursor INTO :id, :statement;

if ( SQLCODE == NOT\_FOUND )

break;

++count;

if ( detailed\_mode ) {

snprintf(question\_id\_str, sizeof(question\_id\_str), "%d", id);

CALL(question, "-s", question\_id\_str);

} else {

printf("%d\t%s\n", id, str\_trim\_right(statement));

}

}

if ( argc == 0 || (argc == 1 && detailed\_mode) )

EXEC SQL CLOSE questions\_cursor;

else

EXEC SQL CLOSE questions\_themes\_cursor;

printf("Total records: %d\n", count);

return 0;

}

int list\_questions\_by\_theme(int argc, char\*\* argv) {

EXEC SQL WHENEVER SQLERROR DO handle\_error();

EXEC SQL BEGIN DECLARE SECTION;

int theme\_id;

int subject\_id;

int theme\_priority;

char theme\_name[256] = {0};

char theme\_id\_str[20];

char subject\_name[256] = {0};

EXEC SQL END DECLARE SECTION;

if ( argc > 1 )

ARGUMENT\_ERROR();

EXEC SQL DECLARE themes\_cursor CURSOR FOR

SELECT themes.id, themes.priority, themes.name, subjects.id, subjects.name FROM themes, subjects WHERE themes.subject\_id = subjects.id ORDER BY subjects.id, themes.priority;

EXEC SQL OPEN themes\_cursor;

while ( 1 ) {

EXEC SQL FETCH themes\_cursor INTO :theme\_id, :theme\_priority, :theme\_name, :subject\_id, :subject\_name;

if ( SQLCODE == NOT\_FOUND )

break;

printf("#%d - %d: %s (%s)\n", theme\_id, theme\_priority, str\_trim\_right(theme\_name), str\_trim\_right(subject\_name));

snprintf(theme\_id\_str, sizeof(theme\_id\_str), "%d", theme\_id);

// argv[argc - 1] is required to be null

CALL(question, "-l", theme\_id\_str, argv[0]);

}

EXEC SQL CLOSE themes\_cursor;

return 0;

}

int show\_question(int argc, char\*\* argv) {

EXEC SQL WHENEVER SQLERROR DO handle\_error();

EXEC SQL BEGIN DECLARE SECTION;

int id;

char statement[256] = {0};

EXEC SQL END DECLARE SECTION;

if ( argc == 0 || argc > 1 )

ARGUMENT\_ERROR();

id = atoi(argv[0]);

EXEC SQL SELECT statement INTO :statement FROM questions WHERE id = :id;

if ( SQLCODE == NOT\_FOUND ) {

fprintf(stderr, "Record not found: %d\n", id);

return 1;

}

printf("%s\n", str\_trim\_right(statement));

printf("-----------------------------------\n");

printf("Options:\n");

CALL(answer, "-l", argv[0]);

printf("Exams:\n");

CALL(exam, "-l", argv[0]);

return 0;

}

**“questions\_themes.h”**

#ifndef QUESTIONS\_THEMES\_H\_

#define QUESTIONS\_THEMES\_H\_

int questions\_themes(int argc, char\*\* argv);

#endif

**“questions\_themes.sc”**

#include "questions\_themes.h"

#include "common.h"

int questions\_themes(int argc, char\*\* argv) {

EXEC SQL BEGIN DECLARE SECTION;

int question\_id;

int theme\_id;

EXEC SQL END DECLARE SECTION;

EXEC SQL WHENEVER SQLERROR DO handle\_error();

if ( argc != 3 )

ARGUMENT\_ERROR();

if ( strcmp(argv[0], "-a") == 0 ) {

argc--; argv++;

question\_id = atoi(argv[0]);

theme\_id = atoi(argv[1]);

EXEC SQL INSERT INTO questions\_themes (question\_id, theme\_id)

VALUES (:question\_id, :theme\_id);

return 0;

}

ARGUMENT\_ERROR();

}