Trabajo Práctico: Bases de Datos I

Primer Semestre 2024

1 Modelo de Datos

A continuación se presenta el modelo de datos que se utiliza para almacenar información relativa a la inscripción de alumnes a materias y al registro de sus notas, para la carrera de *Tecnicatura Universitaria en Informática* de la *UNGS*.

```
alumne(
    id_alumne:int,
    nombre:text,
    apellido:text,
    dni:int;
    fecha_nacimento:date,
    telefono:char(12),
    email:text --válido
)

materia(
    id_materia:int,
    nombre:text
)

correlatividad(
    id_materia:int,
    id_mat_correlativa:int
)

comision(
    id_materia:int,
    id_comision:int, --1, 2, 3,… por cada materia
    cupo: int --máxima cantidad de alumnes que pueden cursar
)

cursada(
    id_materia:int,
    id_alumne:int,
    id_comision:int,
    f_inscripcion:timestamp,
    nota:int, --inicialmente en null: no hay nota
    estado:char(12) --`ingresade',`aceptade',`en espera',`dade de baja'
)

periodo(
    semestre:text, --ejemplo: `2024-1'
    estado:char(12) --`inscripcion',`cierre inscrip',`cursada',`cerrado'
)
```

```
historia_academica(
    id_alumne:int,
    semestre:text,
    id_materia:int,
    id_comision:int,
    estado:char(15), --`ausente',`reprobada',`regular',`aprobada'
    nota_regular:int,
    nota_final:int
)

error(
    id_error:int,
    operacion:char(15),
           --`apertura',`alta inscrip',`baja inscrip',`cierre inscrip',
           --`aplicacion cupo',`ingreso nota',`cierre cursada'
    semestre:text,
    id_alumne:int,
    id_materia:int,
    id_comision:int,
    f_error:timestamp,
    motivo:varchar(80)
)

envio_email(
    id_email:int,
    f_generacion:timestamp,
    email_alumne:text,
    asunto:text,
    cuerpo:text,
    f_envio:timestamp,
    estado:char(10) --`pendiente', `enviado'
)

-- Esta tabla *no* es parte del modelo de datos, pero se incluye para
-- poder probar las funciones.
entrada_trx(
    id_orden:int, --en qué orden se ejecutarán las transacciones
    operacion:char(15),
           --`apertura',`alta inscrip',`baja inscrip',`cierre inscrip',
           --`aplicacion cupo',`ingreso nota',`cierre cursada'
    año:int,
    nro_semestre:int,
    id_alumne:int,
    id_materia:int,
    id_comision:int,
    nota:int
)
```

El sistema debe administrar la apertura y el cierre de los períodos de inscripción, el alta y la baja de inscripciones a materias, la aplicación de cupos, el ingreso de notas y el cierre de cursadas, y mantener toda la información de les alumnes, de las materias y comisiones, así como de sus correlatividades.

Se debe asegurar en todo momento la consistencia de la información mantenida—e.g. no puede haber más alumnes aceptades en una comisión que lo que indica su cupo, el proceso de cierre de aplicación de cupos no puede quedar a medio hacer, etc.

Les alumnes deben ser informades vía email cuando ocurran eventos importantes sobre su situación en cada materia.

2 Creación de la Base de Datos

La base de datos deberá nombrarse con el apellido de cada integrante del equipo en orden alfabético, separados con underscores, y seguidos del string `_db1`, por ejemplo:

`giunta_maradona_palermo_riquelme_db1`

No respetar esto, automáticamente implica la desaprobación del trabajo práctico.

Se deberán crear las tablas respetando ***exactamente*** *los nombres de tablas, atributos, y tipos de datos especificados.*

Se deberán agregar las PK's y FK's de todas las tablas, por separado de la creación de las mismas. Además, se deberá tener la posibilidad de borrar todas las PK's y FK's.

3 Instancia de los Datos

Se deberán cargar 20 alumnes y 21 materias. Todas las materias tendrán una comisión cada una, excepto una materia que tendrá dos comisiones, y otra que tendrá tres comisiones. La tabla `periodo` deberá tener todos los semestres informados en la historia académica.

> Observación:
>
> Los datos de les alumnes, de las materias y sus comisiones, de las correlatividades, de los períodos, de la historia académica, así como los datos de entrada para las transacciones de prueba, deberán cargarse en las tablas correspondientes a partir de los siguientes documentos JSON:
>
> - `alumnes.json`
> - `materias.json`
> - `comisiones.json`
> - `correlatividades.json`
> - `periodos.json`
> - `historia_academica.json`
> - `entradas_trx.json`
>
> Estos archivos `json` se encuentran en el directorio compartido de Google Drive de la comisión, junto a este archivo `pdf`.

4 Stored Procedures y Triggers

El trabajo práctico deberá incluir los siguientes stored procedures ó triggers:

- **apertura de inscripción:** se deberá proveer la lógica que reciba un año y un número de semestre, y que devuelva `true` si se logra abrir la inscripción para el período ó `false` en caso contrario. El procedimiento deberá validar los siguientes elementos antes de confirmar la apertura:

- Que el año sea mayor o igual al año actual. En caso de que no se cumpla, se debe cargar un error con el mensaje `?no se permiten inscripciones para un período anterior`.
- Que el número de semestre sea 1 ó 2. En caso de que no se cumpla, se debe cargar un error con el mensaje `?número de semestre no válido`.
- Si el año y semestre solicitado ya existe en la tabla `periodo`, que su estado sea `cierre inscrip`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?no es posible reabrir la inscripción del período, estado actual:[estado]`, reemplazando en el mensaje `[estado]` por el valor correspondiente que generó el error.
- Que no exista otro período (diferente al solicitado) en estado de `inscripcion` o `cierre inscrip`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?no es posible abrir otro período de inscripción, período actual:[semestre]`, reemplazando en el mensaje `[semestre]` por el valor correspondiente que generó el error.

Si las validaciones pasan correctamente, se deberá insertar o actualizar la fila correspondiente en la tabla `periodo`, dejando su estado en `inscripcion`.

- **inscripción a materia:** se deberá incluir la lógica que reciba un id de alumne, un id de materia y un id de comisión, y que devuelva `true` si se logra ingresar la inscripción, ó `false` si se rechaza. El procedimiento deberá validar los siguientes elementos antes de confirmar el alta:
  - Que exista un período en estado de `inscripcion`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?período de inscripción cerrado`.
  - Que el id de le alumne exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de alumne no válido`.
  - Que el id de la materia exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de materia no válido`.
  - Que el id de comisión exista para la materia. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de comisión no válido para la materia`.
  - Que le alumne no esté inscripte previamente en la materia (en cualquiera de sus comisiones). En caso de que no se cumpla, se debe cargar un error con el mensaje `?alumne ya inscripte en la materia`.
  - Que le alumne tenga en su historia académica todas las materias correlativas en estado `regular` o `aprobada`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?alumne no cumple requisitos de correlatividad`.

  Si se aprueba la solicitud de inscripción, se deberá insertar una fila en la tabla `cursada` con los datos de le alumne, la materia, la comisión y la fecha y hora de inscripción, dejando su estado como `ingresade`.
- **baja de inscripción:** se deberá proveer la lógica que permita anular la inscripción de une alumne. El procedimiento debe recibir un id de alumne y un id de materia, y retornar `true` si se logra dar de baja la inscripción ó `false` en caso contrario. El procedimiento deberá validar los siguientes elementos antes de confirmar la baja:
  - Que exista un período en estado de `inscripcion` o de `cursada`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?no se permiten bajas en este período`.
  - Que el id de le alumne exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de alumne no válido`.

- Que el id de la materia exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de materia no válido`.

- Que le alumne esté inscripte en la materia (en cualquiera de sus comisiones). En caso de que no se cumpla, se debe cargar un error con el mensaje `?alumne no inscripte en la materia`.

Si las validaciones pasan correctamente, se deberá actualizar la fila correspondiente de la tabla `cursada` con el estado `dade de baja`.

En caso de que el período se encuentre en estado de `cursada`, deberá además actualizarse el estado de une alumne de la misma comisión que se encuentre `en espera` (si existe algune), cambiándolo por `aceptade`. Elegir para esto a le alumne que tenga la menor fecha de inscripción.

- **cierre de inscripción:** se deberá proveer la lógica que reciba un año y un número de semestre, y que devuelva `true` si se logra cerrar la inscripción para el período ó `false` en caso contrario. El procedimiento deberá validar los siguientes elementos antes de confirmar el cierre:

  - Que el año y semestre solicitado exista en la tabla `periodo`, y que su estado sea `inscripcion`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?el semestre no se encuentra en período de inscripción`.

  Si las validaciones pasan correctamente, se deberá actualizar la fila correspondiente en la tabla `periodo`, dejando su estado en `cierre inscrip`.

- **aplicación de cupos:** se deberá proveer la lógica que reciba un año y un número de semestre, y que retorne `true` si se logra aplicar los cupos de cursada a cada comisión ó `false` en caso contrario. El procedimiento deberá realizar las siguientes validaciones antes de aplicar los cupos:

  - Que el año y semestre solicitado exista en la tabla `periodo`, y que su estado sea `cierre inscrip`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?el semestre no se encuentra en un período válido para aplicar cupos`.

  Si las validaciones pasan correctamente, se deberá asegurar que se ejecuten las siguientes acciones de forma completa (en caso de que se produzca algún error o inconveniente, las acciones que se hayan realizado deberán deshacerse):

  - Por cada comisión de materia que tenga alumnes inscriptes, se actualizarán con el estado `aceptade` en la tabla `cursada`, como máximo la cantidad de alumnes que esté definida en el cupo para la comisión. Al resto de les alumnes de esa comisión, que excedan el cupo, se les actualizará con el estado `en espera`. La prioridad estará dada por el orden de inscripción. A les alumnes que estuvieran dades de baja no se les deberá cambiar su estado, ni se les deberá contar para la aplicación del cupo.

  - Luego de que se hayan aplicado los cupos a todas las comisiones, se actualizará la fila de la tabla `periodo` que tenga el estado `cierre inscrip`, cambiándolo al estado `cursada`.

- **ingreso de nota de cursada:** se deberá incluir la lógica que reciba un id de alumne, un id de materia, un id de comisión y una nota, y que devuelva `true` si se logra ingresar la nota, ó `false` si se rechaza. El procedimiento deberá validar los siguientes elementos antes de confirmar la grabación de la nota:

  - Que exista un período en estado de `cursada`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?período de cursada cerrado`.

  - Que el id de le alumne exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de alumne no válido`.

- Que el id de la materia exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de materia no válido`.
- Que el id de comisión exista para la materia. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de comisión no válido para la materia`.
- Que le alumne esté inscripte en la comisión de la materia, en estado `aceptade`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?alumne no cursa en la comisión`.
- Que la nota se encuentre en el rango de `0 (cero)` a `10 (diez)`, ambos valores inclusive. En caso de que no se cumpla, se debe cargar un error con el mensaje `?nota no válida:[nota]`, reemplazando en el mensaje `[nota]` por el valor que se recibió como parámetro.

Si las validaciones pasan correctamente, se deberá actualizar la nota de le alumne en la fila correspondiente de la tabla `cursada`.

- **cierre de cursada:** se deberá proveer la lógica que reciba un un id de materia y un id de comisión, y que devuelva `true` si se logra completar el cierre de cursada de la comisión ó `false` en caso contrario. El procedimiento deberá validar los siguientes elementos antes de proceder al cierre:
  - Que exista un período en estado de `cursada`. En caso de que no se cumpla, se debe cargar un error con el mensaje `?período de cursada cerrado`.
  - Que el id de la materia exista. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de materia no válido`.
  - Que el id de comisión exista para la materia. En caso de que no se cumpla, se debe cargar un error con el mensaje `?id de comisión no válido para la materia`.
  - Que la comisión tenga al menos une alumne en la tabla `cursada`, no importa cuál sea su estado de inscripción. En caso de que no se cumpla, se debe cargar un error con el mensaje `?comisión sin alumnes inscriptes`.
  - Que todes les alumnes de la comisión en estado `aceptade` tengan informada su nota de cursada. En caso de que no cumpla, se debe cargar un error con el mensaje `?la carga de notas no está completa`.

  Si las validaciones pasan correctamente, se deberá asegurar que se ejecuten las siguientes acciones de forma completa (en caso de que se produzca algún error o inconveniente, las acciones que se hayan realizado deberán deshacerse):
  - Se deberá insertar en la tabla `historia_academica` una fila por cada alumne de la comisión que se encuentre en estado `aceptade`. El semestre corresponde al período con estado de `cursada`, y el estado de la historia académica dependerá de la nota de regularidad (0:`ausente`, 1-3:`reprobada`, 4-6:`regular`, 7-10:`aprobada`). La nota final sólo se grabará si el estado es `aprobada`, y en ese caso será igual a la nota de regularidad.
  - Luego de que se hayan insertado en la historia académica todes les alumnes de la comisión en estado `aceptade`, se deberán eliminar de la tabla `cursada` todos los registros de esa comisión, cualquiera sea su estado.
- **envío de emails a alumnes:** el trabajo práctico deberá proveer la funcionalidad de generar emails para ser enviados a la dirección de email de le alumne—en la tabla `envio_emails`—cuando sucedan las siguientes novedades:

- Cada vez que se confirme el alta de la inscripción a una materia, se debe ingresar automáticamente un email con el asunto '`Inscripción registrada`', e indicando en el cuerpo del email los datos de la materia, la comisión y de le alumne.

- Cada vez que se confirme la solicitud de baja de inscripción, se debe ingresar automáticamente un email con el asunto '`Inscripción dada de baja`', e indicando en el cuerpo del email los datos de la materia, la comisión y de le alumne.

- Una vez que se hayan aplicado los cupos, se debe generar un email para cada une de les alumnes y para cada una de las materias en las que no estén dades de baja. De acuerdo a cuál sea la situación en la que hayan quedado en cada comisión, el asunto deberá ser '`Inscripción aceptada`' o '`Inscripción en espera`', incluyendo en el cuerpo del email los datos de la materia, de la comisión y de le alumne, y un texto breve informando el estado de la inscripción.

- Cada vez que une alumne salga de la lista de espera y sea aceptade en la cursada (por la baja de otre alumne), se debe ingresar automáticamente un email con el asunto '`Inscripción aceptada`', e indicando en el cuerpo del email los datos de la materia, la comisión y de le alumne.

- Cada vez que se cierre la cursada de una comisión, se debe ingresar automáticamente un email para cada alumne aceptade en esa comisión, con el asunto '`Cierre de cursada`', e indicando en el cuerpo del email los datos de la materia y la comisión, la situación registrada en su historia académica, y su nota de regularidad o de aprobación, según corresponda.

Se deberá crear una tabla con datos de entrada de las transacciones para probar el sistema, que deberá contener los siguientes atributos: `id_orden`, `operacion`, `año`, `nro_semestre`, `id_alumne`, `id_materia`, `id_comision`, `nota`. A partir de esta tabla, se deberá hacer un procedimiento de testeo que invoque a las transacciones correspondientes de acuerdo a la operación. Los datos deberán tomarse del archivo `entradas_trx` provisto en el directorio compartido de Google Drive.

Todo el código SQL escrito para este trabajo práctico, deberá poder ejecutarse desde una aplicación CLI escrita en Go.

5 JSON y Bases de datos NoSQL

Por último, para poder comparar el modelo relacional con un modelo no relacional NoSQL, se pide guardar los datos de alumnes, materias, comisiones e inscripciones a cursada (al menos de dos materias, mínimo de tres alumnes por comisión) en una base de datos NoSQL basada en JSON. Para ello, utilizar la base de datos BoltDB. Este código también deberá ejecutarse desde una aplicación CLI escrita en Go.

6 Condiciones de Entrega y Aprobación

No respetar alguna de las siguientes condiciones, representa la desaprobación del trabajo práctico.

- El trabajo es grupal, en grupos de, *exactamente*, cuatro integrantes. Se debe realizar el trabajo práctico en un repositorio privado `git`, hosteado en **GitLab** con el apellido de les cuatro integrantes, separados con guiones, seguidos del string '-db1' como nombre del proyecto, e.g. `giunta-maradona-palermo-riquelme-db1`. Agregar como `owner`'s a les docentes de la materia, los usernames de Gitlab `hdr` y `ximeebertz` al momento de la creación del repositorio.

- La fecha de entrega *máxima* es el 24 de junio de 2024 en horario de clase, con una defensa presencial del trabajo práctico por cada grupo, en la cual les docentes de la materia van a mirar lo que se encuentre en el repositorio `git` hasta ese momento, y van a hacer distintas preguntas a cada integrante del grupo.

- El informe del trabajo práctico debe incluirse en el directorio principal del repositorio con el nombre de archivo `README.adoc`, y debe realizarse en formato Asciidoc. Para ello, cuentan con una guía en **hdr.gitlab.io/adoc**.

- Usar apropiadamente `git` y GitLab. **No usar branches, no usar rebase, no borrar el repo una vez creado**. Asegurarse de configurar merge, nombre y apellido, y email:

  ```
  $ git config --global pull.rebase false
  $ git config --global user.email 'cristina@kirchner.com'
  $ git config --global user.name 'Cristina Kirchner'
  ```

  Durante la creación de la cuenta de GitLab, configurarla para **uso personal**.

- *Trabajo de investigación:* En este trabajo práctico van a tener que investigar por su cuenta cómo se hacen algunas cosas en PostgreSQL. **No usen ChatGPT, ni busquen en Stack Overflow ó sitios similares, para eso tienen la documentación oficial de PostgreSQL.**