Juan Carizza

Repositor con del TP: https://github.com/jcarizza/tpbbdd

1. Presentación

Durante el año 2010 se llevo a cabo el CENSO Nacional Argentino recopilando información sobre condiciones económicas, sociales, etc. de los habitantes.

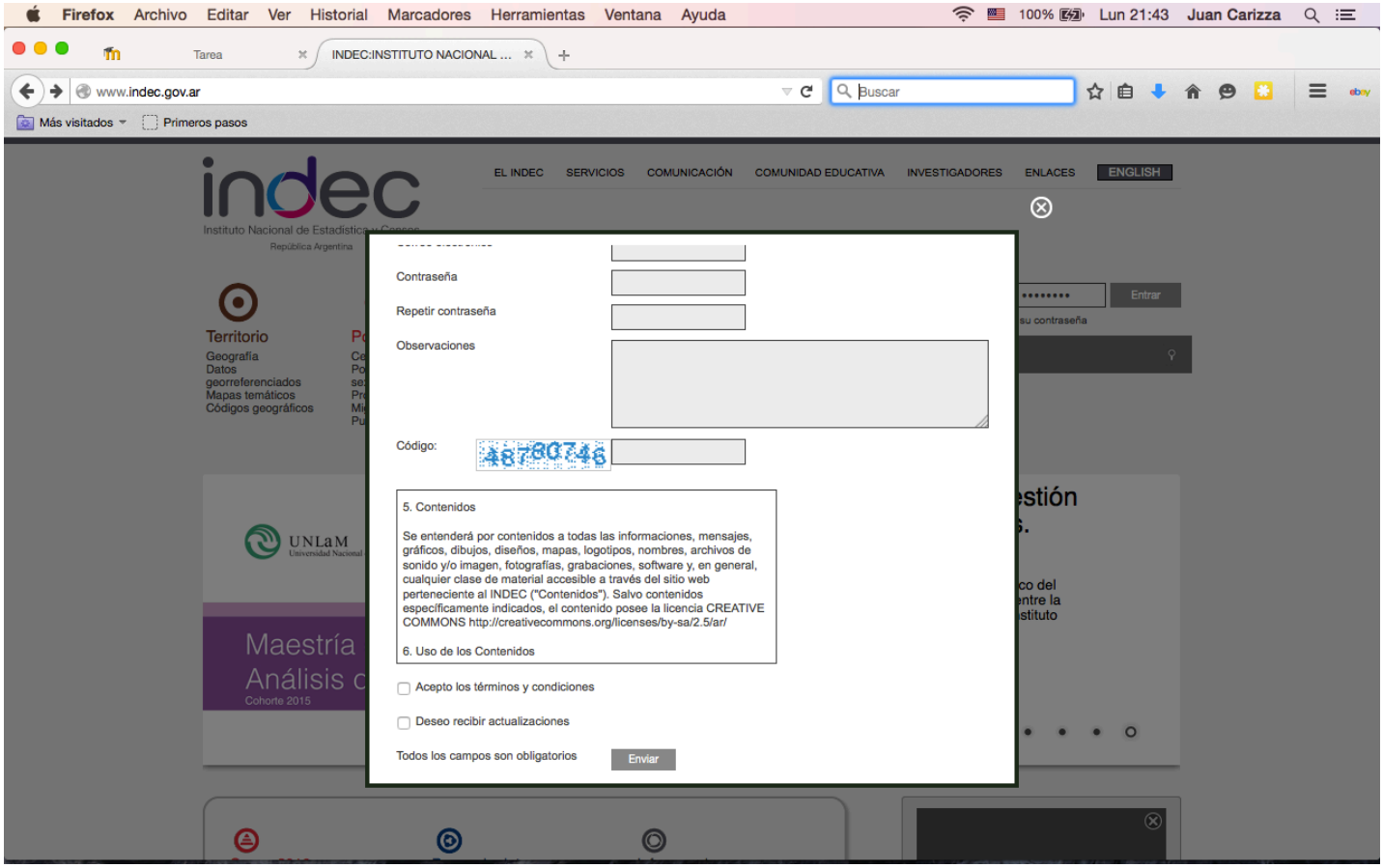
El propósito extraer la información de los archivos en formato PDF obtenidos e insertarlos en un gestor de base de datos para visualizar, exportar, generar reportes, etc.

La información que vamos a usar pertenece a la sección de encuesta de *Hogares y Viviendas*

Correcciones

Los datos se descargaron de la base de datos del INDEC.
http://200.51.91.245/argbin/RpWebEngine.exe/PortalAction?BASE=CPV2010B

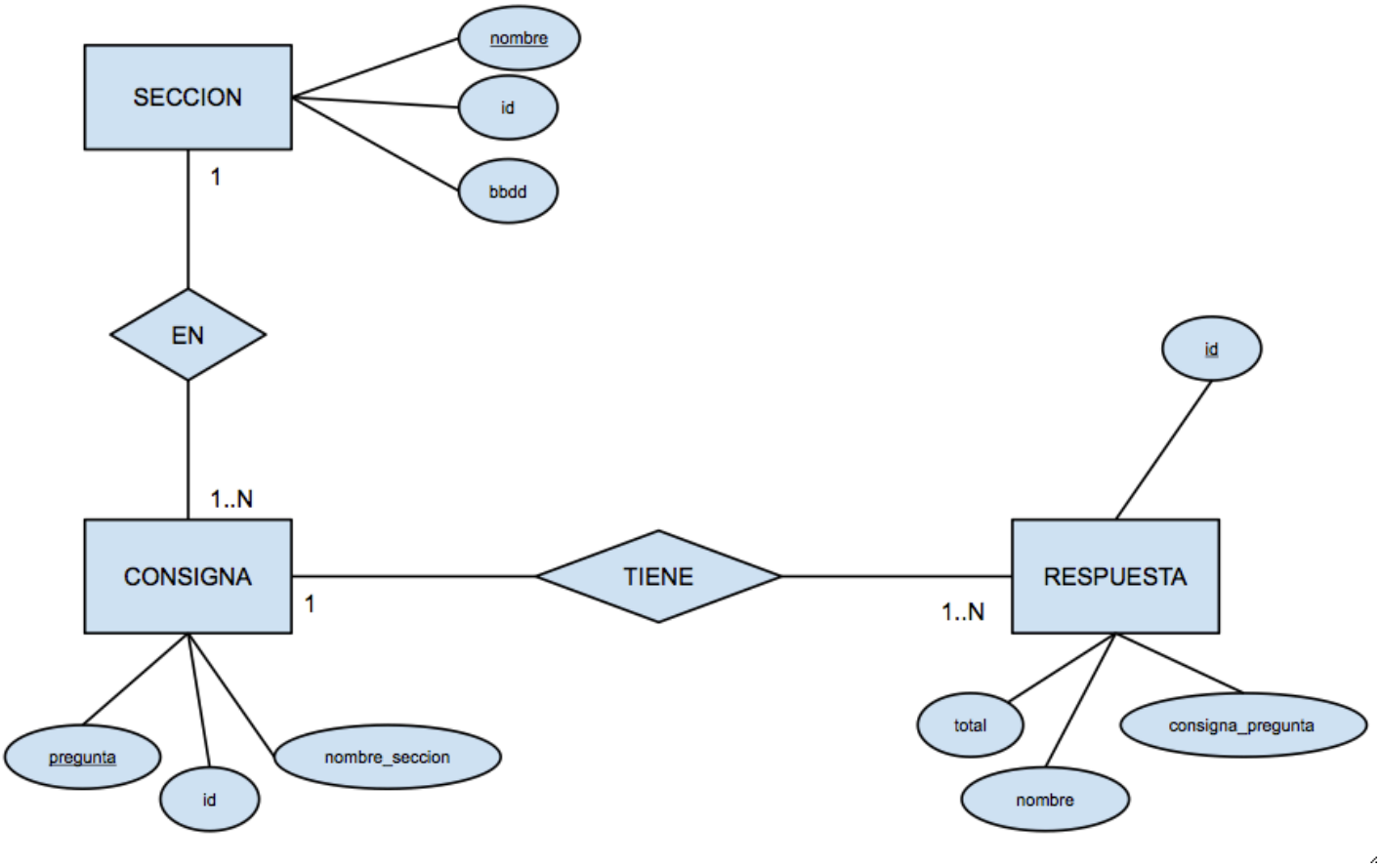
La licencia de los datos es cc-by-sa (Creative Commons by Attribution-ShareAlike 2.5)



Correcciones

2. El modelo relacional

Diagrama ER



correción

Tabla SECCION: Clave candidatas *id*, *nombre*, clave primaria *nombre*

Tabla CONSIGNA: Clave candidatas *id*, *pregunta*, clave primaria elegida *pregunta*

Tabla RESPUESTA Claves candidatas *id*, *consigna_pregunta*, clave primaria seleccionada *id*

Primera forma normal 1FN:

Una tabla está en primera forma normal si cumple con los siguientes requerimientos:

- Todos los atributos son atómicos
- La tabla contiene una clave primaria única
- La clave primaria no contiene atributos nulos
- No debe existir variación en el número de columnas
- Los campos no clave deben identificarse por la clave
- Debe existir una independencia en el orden de columnas y filas

Tabla **SECCION**

Los attributos *bbdd*, *nombre* e *id* son únicos e indivisibles.

Elegí la clave candidata *nombre* y la elegí como clave primaria. Las filas de esta tabla se identificarán únicamente por *nombre*

Todas las secciones tienen la misma cantidad de información por lo tanto no varia el numero de columnas.

La clave primaria elegida (*nombre*) no puede ser nula porque no existe una sección que no tenga nombre.

No hay otros campos no claves en la tabla.

Cada fila de la tabla es única porque el nombre y es la clave primaria; siempre que se haga una consulta por el identificador de la tabla va a devolver una y solo una fila por lo tanto el orden de las filas tiene independencia. El orden de las columnas es independiente porque ninguna depende de la otra en el orden.

Tabla **CONSIGNA** Una *pregunta* contiene un texto que identifica un enunciado interrogativo y es un dato atómico (en los datos no hay dos preguntas en un mismo enunciado) al igual que el campo *id*

Elegí la clave candidata pregunta como clave candidata y luego como clave primaria.

El campo consigna no puede ser nulo porque no se puede hacer una pregunta sin pregunta.

No existe variación en el numero de columnas, cada fila tiene una y sólo una consigna.

Los demas campos no son campos claves, quedando identificables únicamente por la clave primaria pregunta

El orden de columnas y filas es independiente la columna pregunta puede estar a la izquierda o derecha sigue siendo la clave primaria y las filas pueden tener cualquier orden porque se identifica unívocamente por el campo pregunta.

No existe variación en el número de columnas cada respuesta tiene un total y una descripción.

Los campos total y descripción dependen en identificación del campo id.

El orden de las filas y campos es independiente.

Tabla **RESPUESTA**

Los campos total y descripción son atómicos.

La tabla contiene una clave primaria autoincremental que es el campo id el cuál es un número entero.

El campo id no puede ser nulo porque no existe el número nulo.

Segunda forma normal 2FN:

Una tabla está en segunda forma normal si está en 1FN y ademas cumple con el siguiente requisito:

- Todos los atributos que no son clave primaria dependen funcionalmente de la clave principal.

Que dependa funcionalmente de la clave primaria es que para acceder siempre al mismo attributo no clave se use la clave primaria. Teoricamente se expresa como que la clave primaria X define el attributo Y.

Tabla **SECCION**:

En esta tabla la columna *bbdd* depende funcionalmente de la columna *nombre* para ser identificada

Tabla **CONSIGNA**:

Las columnas *id* y *nombre_seccion* dependen de la clave primaria *pregunta* para ser identificadas.

Tabla **RESPUESTA**:

Las columnas *nombre*, *total* y *consigna_pregunta* estan definidas por la columna *id*

Tercera forma normal 3FN:

Una tabla está en tercera forma normal si está en 2FN y ademas cumple con lo siguiente:

- No existe ninguna dependencia funcional transitiva entre un attributos que no son claves.

Tabla **SECCION**:

Las columnas *id* y *bbdd* dependen de la clave *nombre* y no transitivamente. (id no depende de nombre a travez de bbdd ni bbdd depende de nombre a travez de id)

Tabla **CONSIGNA**:

Las columnas *id* y *nombre_seccion* no dependen de *pregunta* de forma transitiva.

Tabla **RESPUESTA**:

Las columnas *total*, *nombre* y *consigna_pregunta* no dependen transitivamente de la columna *id*.

correción

3. Esquema Relacional

Relaciones

Como las únicas relaciones que hay son 1..N no necesitamos crear ninguna tabla intermediaria por lo tanto debajo solo listo las entidades.

SECCION(id, bbdd, nombre)

CONSIGNA(id, pregunta, seccion_nombre)

RESPUESTA(id, nombre, total, consigna_pregunta)

Dominio de los atributos

SECCION:

dominio(id)=integer

dominio(bbdd)=text

dominio(nombre)=text

CONSIGNA:

dominio(id)=integer

dominio(pregunta)=text

dominio(seccion_nombre)=text

RESPUESTA:

dominio(id)=integer

dominio(nombre)=text

dominio(consigna_pregunta)=text

dominio(total)=integer

Claves candidatas y claves primarias: **SECCION:** Claves candidatas: {id, nombre}

Clave primaria: {nombre}

CONSIGNA:

Claves candidatas: {pregunta, id}

Clave primaria: {pregunta}

RESPUESTA:

Claves candidatas: {id, nombre}

Clave primaria: {id}

Claves foraneas:

Las columnas con clave foranea son:

- seccion_nombre establece la relacion con el attributo *nombre* de la tabla SECCION y que ademas es clave primaria.
- consigna_pregunta establece la relacion con el attributo clave primaria *pregunta* de la tabla CONSIGNA

Algebra relacional

Para probar el modelo realizamos unas consultas para ver si nos devuelve los resultados como lo planeamos

Seleccionar todas las filas de la tabla *consigna* donde el campo seccion_nombre sea igual a "Hogar y Vivienda"

- π(id, pregunta)(σ seccion_nombre="Hogar y Vivienda" consigna)

```
indec=#
SELECT id, pregunta FROM consigna WHERE seccion_nombre='Hogar y Vivienda';
  1 | Tipo de vivienda agrupado
  2 | Tipo de vivienda particular
  3 | Tipo de vivienda colectiva
  4 | Area Urbano - Rural
  5 | Condición de ocupación
  6 | Municipio
  7 | Localidad
  8 | Calidad constructiva de la vivienda
  9 | Calidad de los materiales
 10 | Calidad de Conexiones a servicios básicos
 11 | Cantidad de hogares en la vivienda
 12 | Tipo de casa

indec=#
```

Seleccionar las filas de la tabla *respuesta* que hayan donde el total sea mayor a 1000.

- π(id, total, nombre)(σ total>1000 respuesta)

```
indec=# SELECT id, total, nombre FROM respuesta WHERE total>1000;
  1 | 1424571 |  Particular
  2 |    1867 |  Colectiva
  3 |  340368 |  Casa
  5 |    2202 |  Casilla
  6 | 1041759 |  Departamento
  7 |   19571 |  Pieza en inquilinato
  8 |   17082 |  Pieza en hotel familiar o pensión
  9 |    2237 |  Local no construido para habitación
 12 | 1082998 |  Con personas presentes
 13 |  155740 |  Con todas las personas temporalmente
 14 |   52387 |  En alquiler o venta
 15 |    6433 |  En construcción
 16 |   51060 |  Se usa como comercio, oficina o consultorio
 17 |    8508 |  Se usa para vacaciones, fin de semana u otro
 18 |   66847 |  Por otra razón
 28 | 1426438 |  Urbano
 29 |  131382 |  COMUNA1
 30 |  108136 |  COMUNA2
 31 |  101436 |  COMUNA3
 32 |   83033 |  COMUNA4
 33 |   92903 |  COMUNA5
 34 |   93481 |  COMUNA6
 35 |   89688 |  COMUNA7
 36 |   55427 |  COMUNA8
 37 |   63414 |  COMUNA9
 38 |   71664 |  COMUNA10
 39 |   84734 |  COMUNA11
 40 |   93503 |  COMUNA12
```

4, 5 y 6 Sentencias DDL

Crear usuario y base de datos:

```
createuser -P indec;  # Nuevo usuario indec
createdb -O indec; indec  # Crear y asignar base de datos al usuario
```

Crear tablas:

```
CREATE TABLE seccion (
    id SERIAL,
    bbdd text,
    nombre text PRIMARY KEY UNIQUE NOT NULL
);

CREATE TABLE consigna (
    id SERIAL,
    pregunta text PRIMARY KEY UNIQUE NOT NULL,
    seccion_nombre text REFERENCES seccion (nombre) NOT NULL
);

CREATE TABLE respuesta (
    id SERIAL PRIMARY KEY,
    nombre text NOT NULL,
    total integer,
    consigna_pregunta text REFERENCES consigna (pregunta) NOT NULL
);
```

7. Analisis comparativo

Característica	Firebird	MySQL	MSSQL	Postgresql
Licencia	IPL/IDPL	GPL/Comercial	Comercial	BSD
Multiplataforma	Sí	Sí	No	Sí
Clusterizable	No	Sí	Sí	Sí
Comunidad	Buena	Muy buena	(?)	Excelente
Valoración	Buena	Muy buena	Buena	Muy buena
Difusión	Media	Mucha	Mucha	Mucha
Soporte	No	Comercial	Comercial	No

8. Mantenimiento

Generación de backup:

```
pg_dump indec > backup.sql
```

Restaurar base de datos:

```
psql indec < backup.sql
```

Consideraciones de seguridad:

La información que se maneja en esta base de datos es información pública por lo tanto no es necesario implementar medidas de seguridad elevadas (como encriptación). Nos limitamos a crear un usuario que tenga solo permiso para operar la base de datos *indec*.

Crear un usuario que no sea admin evitamos que se pueda robar informacion de otras base de datos o dumpear informacion sensible mediante injectiones SQL, leer archivos con COPY, etc.

Copiar datos iniciales:

La información está procesada y lista para descargar desde estos gist:

- seccion.csv
- consigna.csv
- respuesta.csv

Una vez descargados, copiar con COPY:

```
COPY seccion(nombre) from '/var/lib/pgsql/seccion.csv' CSV;
COPY consigna(seccion_nombre, pregunta) from '/var/lib/pgsql/consigna.csv' CSV;
COPY respuesta(consigna_pregunta, nombre, total) from '/var/lib/pgsql/respuesta.cs
v' CSV;
```

Aclaracion: Los archivos deben estar en el mismo host del servidor postgres y ademas debe tener los permisos para que el usuario postgres pueda leerlos, indicar el camino completo hacia el archivo.

Ademas debe tener permisos de super usuario para poder leer los archivos con COPY (Esto es por una cuestion de seguridad).

9. Aplicación python

https://gist.github.com/jcarizza/93bfae808074cc5db1d1

Exportar informacion de la calidad de los materiales:

```
fox:BBDD jcarizza$ python indec.py exportar
0: Municipio
1: Calidad de los materiales
Elija que exportar
opcion: 0
COMUNA1, 131382
COMUNA2, 108136
COMUNA3, 101436
COMUNA4, 83033
COMUNA5, 92903
COMUNA6, 93481
COMUNA7, 89688
COMUNA8, 55427
COMUNA9, 63414
COMUNA10, 71664
COMUNA11, 84734
COMUNA12, 93502
COMUNA13, 129784
COMUNA14, 141408
COMUNA15, 86446
fox:BBDD jcarizza$
```

Graficar cantidad de viviendas por municipio:

```
pfox:BBDD jcarizza$ python indec.py graficar
0: Municipio
1: Calidad de los materiales
Elija que graficar
opcion :0
Se ha generado un archivo html con el grafico en viviendas_por_municipio.html
fox:BBDD jcarizza$ open viviendas_por_municipio.html
```

10. Análisis DAFO

Analisis DAFO - Fortalezas

PostgreSQL	MySQL
Integridad: PITR - WAL (1) Concurrencia: Multiples usuarios - escritura MVCC (2) Funcionalidad: tipos (OO), reglas, lenguajes embebidos	Escalabilidad: replicacion, clustering Flexibilidad: Multiples motores de almacenamiento Velocidad: Sin integridad / concurrencia

1. PITR: Backup de copia continua
2. MVCC: Manejo de acceso concurrente a informacion

Analisis DAFO - Oportunidades

PostgreSQL	MySQL
MySql -> Sun -> Oracle: forks: MariaDB, Drizzle Estable y Sólido,(MySQL 5.1 != bug free)	Nuevos Motores de Almacenamiento (Maria/Falcon?) Gran aceptación por parte del mercado (11 millones de instalaciones)

Analisis DAFO - Debilidades

PostgreSQL	MySQL
Soporte Comercial Ausencia de replicacion nativa,(roadmap 8.5) Ausencia de cluster nativo	Nuevos Motores de Almacenamiento (Maria/Falcon?) Gran aceptación por parte del mercado (11 millones de instalaciones)

Analisis DAFO - Amenazas

PostgreSQL	MySQL
MariaDB, FireBird, Oracle	(Modelo de Arquitectura) SQL rudimentario Planner Basico MyISAM no soporta FKs (roadmap 5.2)

Licencia

Links:

http://www.postgresql.org.ar/trac/attachment/wiki/PgDayUnnoba/PDF*pgsql*mysql_FODA.pdf http://db-engines.com/en/system/Firebird%3BMySQL%3BPostgreSQL http://es.slideshare.net/nosys/replicacion-y-

haenpostgresql https://es.wikipedia.org/wiki/Modelo_relacional
http://www.ual.es/~mtorres/BD/bdtransp3.pdf http://es.slideshare.net/JosepSalvadorSotoObregon/algebra-relacional-fundamentos-de-base-de-datos https://es.wikipedia.org/wiki/Normalización*de*bases*de*datos
http://www.ub.edu.ar/catedras/ingenieria/Datos/capitulo4/cap43.htm
http://es.slideshare.net/gerardo_gauna/modelo-relacional-15371336
https://es.wikipedia.org/wiki/Modelo_entidad-relación https://es.wikipedia.org/wiki/Análisis_DAFO
https://es.wikipedia.org/wiki/PostgreSQL https://es.wikipedia.org/wiki/Microsoft*SQL*Server
https://es.wikipedia.org/wiki/MySQL https://es.wikipedia.org/wiki/Firebird