Apéndice A: Documentación Base de Datos.

A1. Derivación

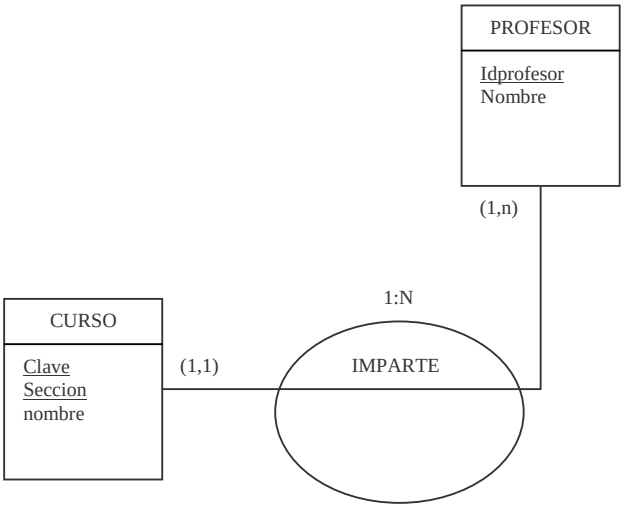

fig 3.2.1.2 Relación Profesor- Curso

Se puede ver en la figura 3.2.1.2 que la cardinalidad **(1,1)** indica que la entidad CURSO se le debe asignar a través de la relación IMPARTE un sólo profesor. Por el otro lado, la entidad PROFESOR puede estar relacionado mediante IMPARTE con uno o varios cursos **(1,n)**. De acuerdo con lo anterior, para este caso, se puede decir que sólo es necesario propagar la llave primaria de PROFESOR como llave extranjera en CURSO, evitando crear la tabla con respecto a la relación IMPARTE.

En pocas palabras como hay una entidad con orden n y otra de orden 1, solo se deben crear dos tablas (una por entidad), en donde la entidad de orden n propague la relación mediante su llave primaria a la otra entidad.

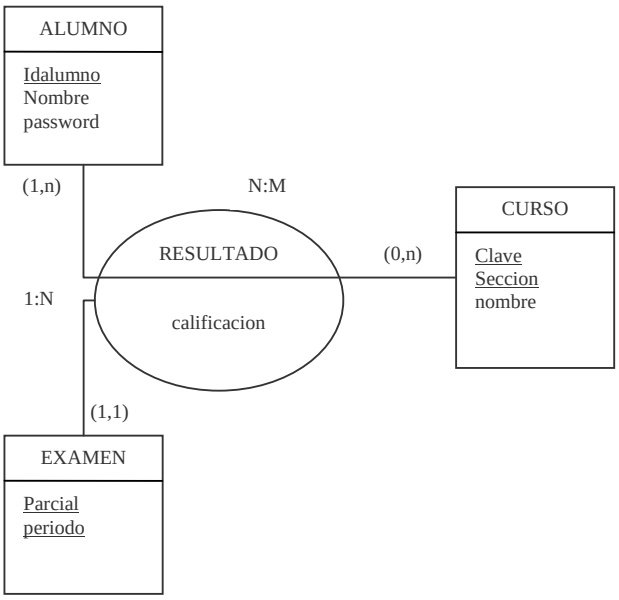

fig 3.2.1.3 Relación Alumno- Curso-Examen

Según la Fig. 3.2.1.3, aquí nos encontramos con tres entidades (ALUMNO, EXAMEN, CURSO) conectadas a la misma relación (RESULTADO), debido a las siguientes cardinalidades:

- Sólo puede haber un examen para cada parcial **(1,1)** en cierto curso por lo que únicamente hay un resultado (calificación) de cierto alumno en un examen.
- Por otro lado, un curso puede tener varios exámenes o ninguno **(0,n)**, al igual puede tener resultados de alumnos o no(también quiere decir que el hecho de que exista un curso no implica que existan: resultados de cualquier alumno y/o exámenes).
- Por último un alumno puede tener de uno a n resultados **(1,n)** en uno o varios cursos según corresponda al examen o exámenes que tome.

Teniendo en cuenta estos tres puntos y conociendo que hay un atributo “calificacion” en la relación RESULTADO que depende directamente de las tres entidades, lo más recomendable es hacer una tabla donde se propague la relación con las llaves primarias de las tres entidades como llaves extranjeras, evitando con esto tener mucha redundancia de información en cualquiera de las otras tablas de las entidades.

Este análisis se obtuvo a partir de que se sabe que dos entidades son de orden n y la otra de orden 1, lo que a primera vista sugiere que sólo se propague la relación mediante las llaves primarias de las entidades de orden n a la entidad de orden 1, pero hay un gran detalle que es el atributo “calificacion” el cual provocaría que existiera demasiada redundancia en la tabla resultante, por lo que la segunda opción es crear 4 tablas, una para la relación RESULTADO donde se propagarían las llaves primarias de las otras tablas para no perder la relación.

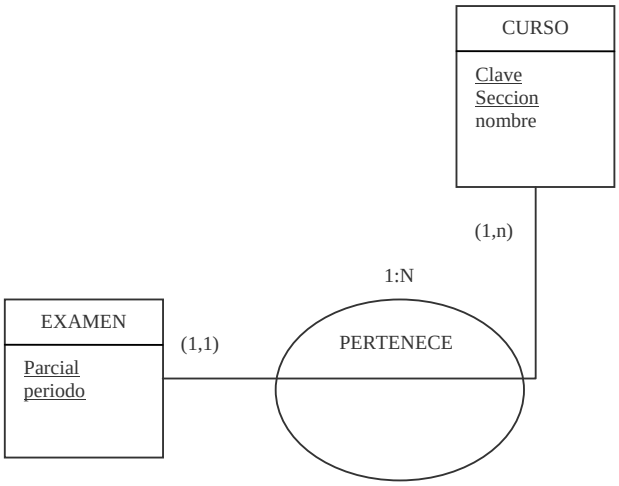

fig 3.2.1.4 Relación Examen- Curso

Si vemos la Fig. 3.2.1.4, encontramos una interrelación 1:N por lo cual se recomienda crear sólo dos tablas y propagar la relación mediante la llave primaria de la entidad CURSO a la entidad EXAMEN como una llave extranjera.

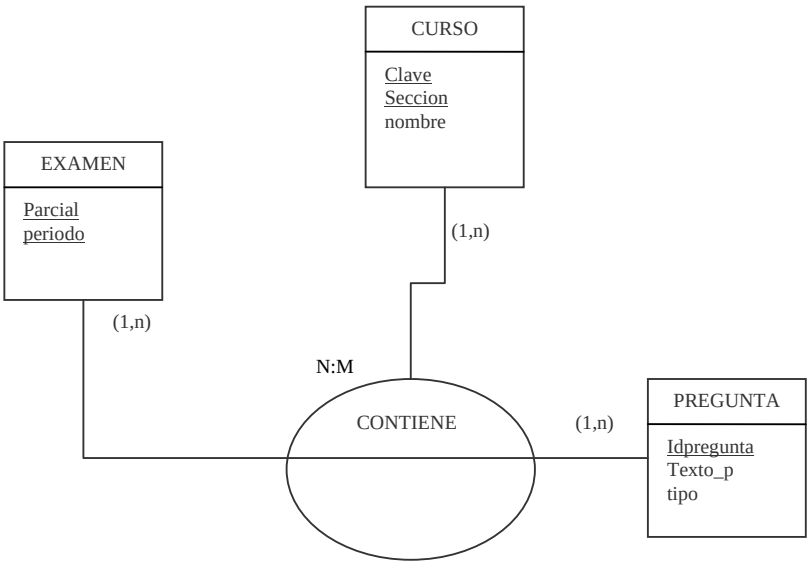

Fig 3.2.1.5 Relación Examen- Curso-Profesor

Para la situación de la fig.3.2.1.5, nos volvemos a encontrar con tres entidades para una relación la transformación es más simple ya que como todas las entidades tienen cardinalidad N:M pues directamente se crean una tabla por cada entidad, y una para la relación que tendrá como atributos son las llaves primarias de las entidades.

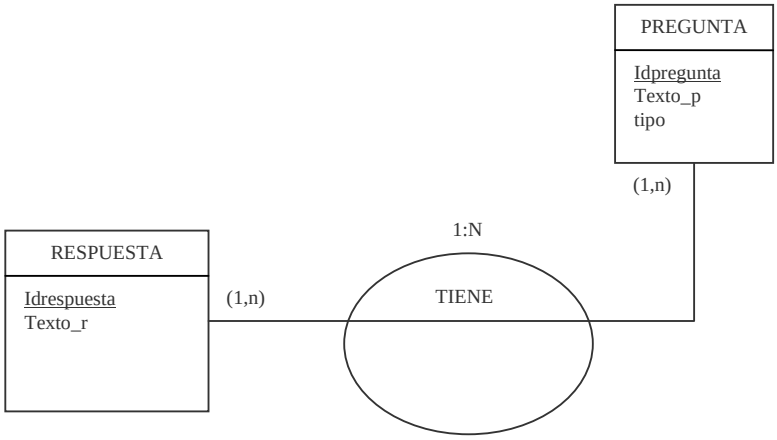

Fig 3.2.1.6 Relación Pregunta- Respuesta

Por último en el diagrama de la Fig.3.2.1.6, tenemos que la interrelación de la entidades PREGUNTA y RESPUESTA es de N:N por lo que se recomienda crear una tabla para la relación TIENE, y propagar las llaves primarias a la para TIENE como llaves foránea, además se debe agregar el atributo "correctaIncorrecta" en esta misma tabla ya que esté define la relación de estas dos tablas.

A.2 Formas Normales

La Primera Forma Normal (**1FN**) Consiste en la prohibición de que en una relación existan grupos repetitivos, es decir, un atributo no puede tomar más de un valor del dominio subyacente, esto también es conocido como atomicidad de los atributos. Esto se refiere a que por ejemplo en la tabla de ALUMNO no puede tener un atributo donde se guarden todas las calificaciones acumuladas, y/o que tenga otro atributo donde estén todos los cursos a los que esta inscrito.

La Segunda Forma Normal (**2FN**), consiste en que si además de estar en 1FN, todos los atributos que no forman parte de ninguna clave candidata suministran información acerca de la clave completa, es decir, para evitar tener información irrelevante, por ejemplo en la tabla RESULTADO si tuviera otro atributo adicional como *"nombrecurso"* la BD dejaría de estar en **2FN** ya que este atributo constituye información acerca del curso y no de un resultado, por otro lado, el atributo *"calificación"* no forma parte de las claves candidatas pero suministra información acerca de la clave completa.

La Tercera Forma Normal (**3FN**), se refiere a que si además de estar en 2FN, los atributos que no forman parte de ninguna clave candidata facilitan información sólo acerca de la(s) clave(s) y no acerca de otros atributos. Para la tabla PREGUNTA existe un atributo "tipo" que forma parte de ninguna clave candidata y no facilita información acerca del atributo "textP" lo cual se elimina la duda de que la Base de Datos no esta en 3FN.

Para este diseño tendríamos un problema en la tabla INSCRITO si existiera el atributo *"ultimoparcial"* el cual se refiere al último parcial que tomo el alumno, por lo que se

estaría entregando información del alumno el cual ya tiene asociada otra tabla lo que nos haría pensar que la Base de Datos no estaría en 3FN, pero como esto no sucede en ninguna tabla no tenemos ningún problema.

Cuarta Forma Normal (4 FN). Una relación se encuentra en **4FN**, si y solo si, las únicas dependencias multivaluadas no triviales son aquellas en las cuales una clave multidetermina un atributo, es decir, toda dependencia multivaluada viene determinada por una clave candidata. La clave candidata es el conjunto de los todos los atributos de una tabla. En pocas palabras para cada tabla sólo debe existir una clave candidata que defina la tabla.

A.3 Creación Física de la Base de datos

Una vez que se tiene definida la estructura de la base de datos sólo resta la creación física de esta en una base de datos real, lo cual se logra utilizando ciertos querys que en este caso deben ser insertados en el interprete SQL.

Primero se deben crear las tablas para cada entidad dando prioridad a las entidades que no tienen ningún atributo propagado por otra entidad o relación, debido a que como no pueden declarar el atributo que ha sido propagado sin especificar la tabla origen, se puede producir un error porque dicha tabla origen no exista o simplemente la consistencia en los datos de esta tabla será ineficaz.

Por lo anterior las primeras tablas a crear serán ACCESO, RESPUESTA, PREGUNTA

```
CREATE TABLE acceso(login char(8) not null PRIMARY KEY,
                    password char(10) not null, activo integer);

CREATE TABLE respuesta(idrespuesta char(10)NOT NULL PRIMARY KEY,
                    textR Text NOT NULL);

CREATE TABLE pregunta(idpregunta char(10) NOT NULL PRIMARY KEY,
                    textP Text NOT NULL,
                    tipo int NOT NULL);
```

Las segundas TIENE, ALUMNO, PROFESOR

```
CREATE TABLE tiene(idpregunta char(10),
                    idrespuesta char(10),
                    correctaincorrecta integer NOT NULL,
                    FOREIGN KEY (idpregunta) REFERENCES
pregunta(idpregunta) ON UPDATE CASCADE ON DELETE CASCADE,
                    FOREIGN KEY (idrespuesta) REFERENCES
respuesta(idrespuesta) ON UPDATE CASCADE ON DELETE CASCADE);


CREATE TABLE alumno(id char(10) not null primary key,
                    login char(8),
                    nombre char(50) NOT NULL,
                    FOREIGN KEY (login) REFERENCES acceso(login) ON
UPDATE CASCADE ON DELETE CASCADE);


CREATE TABLE profesor(idprofesor char(10) not null primary key,
                    login char(8),
                    nombre char(50) NOT NULL,
                    FOREIGN KEY (login) REFERENCES acceso(login) ON
UPDATE CASCADE ON DELETE CASCADE);
```

La tercera CURSO

```
CREATE TABLE curso(clavecurso char(6)NOT NULL PRIMARY KEY,
                    seccion integer NOT NULL,
                    nombrecurso char(50),
                    idprofesor char(9)NOT NULL,
```

FOREIGN KEY (idprofesor) REFERENCES profesor(idprofesor) ON UPDATE CASCADE);

Las cuartas INSCRITO, EXAMEN

```
CREATE TABLE inscrito(idalumno char(9)NOT NULL,
                      clavecurso char(6)NOT NULL,
                      seccion integer NOT NULL,
                      FOREIGN KEY (clavecurso,seccion) REFERENCES
                      curso(clavecurso,seccion) ON UPDATE CASCADE ON
                      DELETE CASCADE, FOREIGN KEY (idalumno)
                      REFERENCES alumno(id) ON UPDATE CASCADE ON
                      DELETE CASCADE);


CREATE TABLE examen(clavecurso char(6) NOT NULL,
                     seccion integer NOT NULL,
                     activo integer,
                      parcial integer NOT NULL,
                      periodo char(10) NOT NULL,
                     FOREIGN KEY (clavecurso,seccion) REFERENCES
                     curso(clavecurso,seccion) ON UPDATE CASCADE ON
                     DELETE CASCADE);
```

Las quintas RESULTADO, CONTIENE

```
CREATE TABLE resultado(idalumno char(9)NOT NULL,
                       clavecurso char(50) NOT NULL,
                       parcial integer NOT NULL,
                       periodo char(10) NOT NULL,
                       seccion integer NOT NULL,
```

```
        calificacion float,
        FOREIGN KEY (clavecurso,seccion) REFERENCES
    cursos(clavecurso,seccion),
        FOREIGN KEY (idalumno) REFERENCES alumno(id) ON
    UPDATE CASCADE,
        FOREIGN KEY (parcial,periodo) REFERENCES
    examen(parcial,periodo)  );


CREATE TABLE contiene(idpregunta char(9)NOT NULL,
        clavecurso char(50) NOT NULL,
        parcial integer NOT NULL,
        periodo char(10) NOT NULL,
        seccion integer NOT NULL,
        FOREIGN KEY (clavecurso,seccion) REFERENCES
        cursos(clavecurso,seccion) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (idpregunta) REFERENCES
        pregunta(idpregunta) ON UPDATE CASCADE ON DELETE
        CASCADE,
        FOREIGN KEY (parcial,periodo) REFERENCES
        examen(parcial,periodo) ON UPDATE CASCADE ON DELETE
        CASCADE );
```