Capítulo 3. Fases de descubrimiento en bases de datos

3.1 Sistema de gestión de bases de datos

La elección del sistema de gestión de bases de datos no es una tarea fácil, si bien todos tienen un mismo objetivo que es definir, construir y manipular una base de datos, es cierto también que algunos poseen características especificas que dependiendo del propósito para que se quieran usar son mejores que otros, por ejemplo algunos sistemas cuentan con una forma más sencilla de generar consultas complejas que otros, en algunos los procedimientos son más avanzados y por mencionar otra característica algunos sistemas de gestión no permiten múltiples desencadenadores, cualquiera de las características antes mencionadas u otras más pueden ser suficiente justificación para elegir un sistema de gestión de bases de datos de otro.

Para el presente trabajo se ha elegido utilizar como principal sistema de gestión de bases de datos a SQL Server 2008, las principales razones son expuestas a continuación:

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL. Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, PostgreSQL o MySQL. [20]

FIGURA 3.1.a *Microsoft SQLServer*

Ventajas: [20]

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.

Desventajas [20]

- MSSQL no maneja compresión de datos (en *SQL Server* 2005 y 2000, solamente la versión 2008 *Enterprise Edition* incluye esta característica), por lo que ocupa mucho espacio en disco.
- MSSQL está atado a la plataforma del sistema operativo sobre la cual se instala.

3.2 Integración, almacenamiento y limpieza (Etapa 1 y 2 del KDD)

Como se mencionó previamente un almacén de datos es un repositorio de información coleccionada desde varias fuentes, almacenada bajo un esquema unificado que normalmente reside en un único emplazamiento[1] y que nos permiten aplicar herramientas que permiten resumir, describir y analizar datos con el fin de ayudar en la toma de decisiones estratégicas.

En el presente trabajo y para buscar la consecución de los objetivos planteados en el primer capítulo del mismo se recolectaron cinco distintas fuentes de información, buscando con ellas y mediante su integración, almacenamiento y limpieza el obtener un almacén de datos que satisfaga la necesidad de contener en una sola base toda la información requerida para comenzar la minería.

Las bases de información recolectadas son:

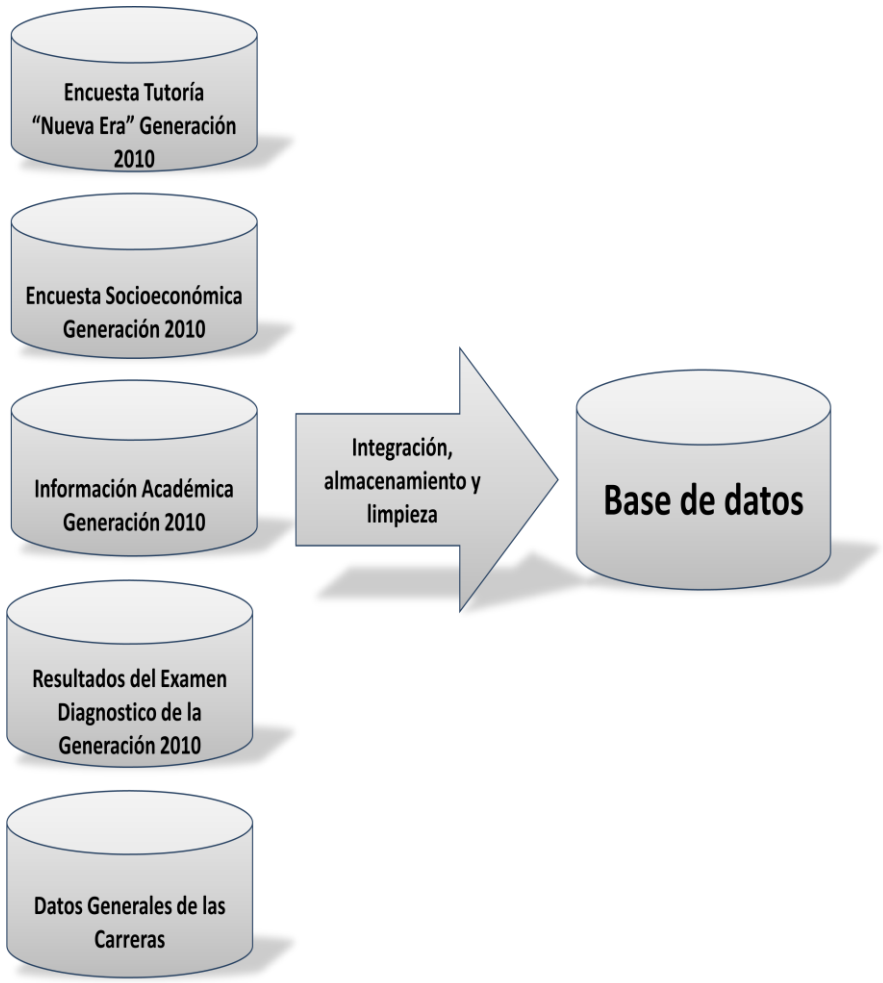

FIGURA 3.2.a Bases de información recolectadas para la minería

[1] A cada uno de los ordenadores que integran un sistema de Bases de Datos distribuido se le conoce como nodo o emplazamiento del sistema y pueden ser administrados de forma diferente.

Los datos de cada una de ellas se muestran en la siguiente tabla:

Base de información	Fuente	Formato	Información contenida
Encuesta tutoría "Nueva Era" Generación 2010	Archivo proporcionado por la Facultad de Ingeniería de la UNAM	Excel	Resultados de la encuesta respecto a la tutoría "Nueva Era" de la generación 2010
Encuesta Socioeconómica Generación 2010	Archivo proporcionado por la Facultad de Ingeniería de la UNAM	Excel	Resultados de la encuesta socioeconómica de la generación 2010
Información Académica Generación 2010	Archivo proporcionado por la Facultad de Ingeniería de la UNAM	Excel	Calificaciones de las materias inscritas de los alumnos de la generación 2010
Datos Generales de las Carreras	Página web de la Facultad de Ingeniería de la UNAM	Contendido web	Información general de cada una de las carreras como nombre, claves, siglas, créditos totales, etc.
Resultados del Examen Diagnostico de la Generación 2010	Archivo proporcionado por la Facultad de Ingeniería de la UNAM	Excel	Resultados del examen diagnostico que realizo la Facultad de Ingeniería a los alumnos de la generación 2010 previo al inicio del semestre

FIGURA 3.2.b Detalle de las bases de información recolectadas para la minería

Hasta este punto se ha hablado acerca de las fuentes de información recolectadas y del sistema de gestión de bases de datos a utilizar, así como las características de los mismos, pero no puede hacerse a un lado uno de los puntos más importantes para que la minería de datos funcione y es la limpieza y transformación de datos.

Partiendo del hecho de que el éxito de un proceso de minería de datos depende, de tener todos los datos necesarios (una buena recopilación) y también una buena limpieza e integración, se puede afirmar que este punto es uno de los pilares más importantes de la minería de dato y es que una mala calidad de la información, trae como consecuencia que los patrones descubiertos contrasten con la realidad, y la realización del proceso sea inútil.

El primer problema al realizar una integración de distintas fuentes de datos son los identificadores únicos. Este problema se conoce como esclarecimiento de la identidad. Para el presente trabajo dicho problema fue resuelto con el número de cuenta del alumno, de esta manera podemos obtener que todas las base de información recolectadas comparten dicho dato, el cual es único e irrepetible.

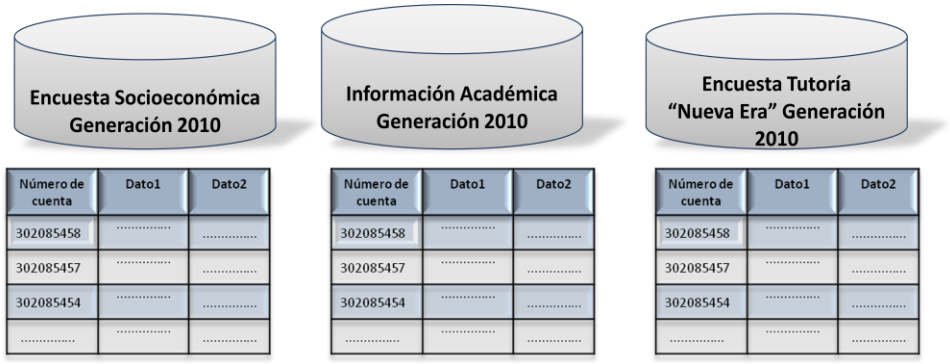

FIGURA 3.2.c Muestras de las bases de información

El siguiente punto a considerar es que al integrar dos fuentes o más suele suceder que aparezcan datos faltantes o diferentes, por ejemplo para esta investigación tanto la encuesta socioeconómica como la encuesta de tutores contaban con el campo de sexo, al momento de realizar la integración se presentaron casos (pocos) en los que una respuesta para esta opción difirió en ambas encuestas para el mismo identificador del alumno, es decir, mientras en una encuesta el sexo era femenino en otra encuesta aparecía como masculino, la solución que se considero en este trabajo es dejar dicho dato al azar.

En ocasiones para la realización de la minería de datos es recomendable utilizar valores numéricos que valores nominales, por ejemplo en lugar de utilizar un SI o un NO para una respuesta en alguna pregunta de la encuesta sería recomendable utilizar un 1 para SI y un 2 para un NO, es por lo anterior que en la integración de datos se podría incluir una transformación de tal forma que se manejen solo valores numéricos, sin embargo para el presente trabajo y por la descripción del almacén de datos, la elección tomada fue manejar valores nominales, y será hasta la creación de la vista minable cuando dichos valores serán transformados en numéricos. En realidad muchas transformaciones en los datos así como la obtención de otros datos derivados están consideradas para ser realizados en la creación de la vista minable y no antes.

Finalmente y tras realizar todo lo antes mencionado es posible obtener un almacén de datos completo y útil, con información coherente y que permita obtener buenos resultados.

3.3 Descripción del almacén de datos

3.3.1 Modelo Entidad/Relación

A continuación se muestra el modelo entidad/relación de la base de datos relacional diseñada:

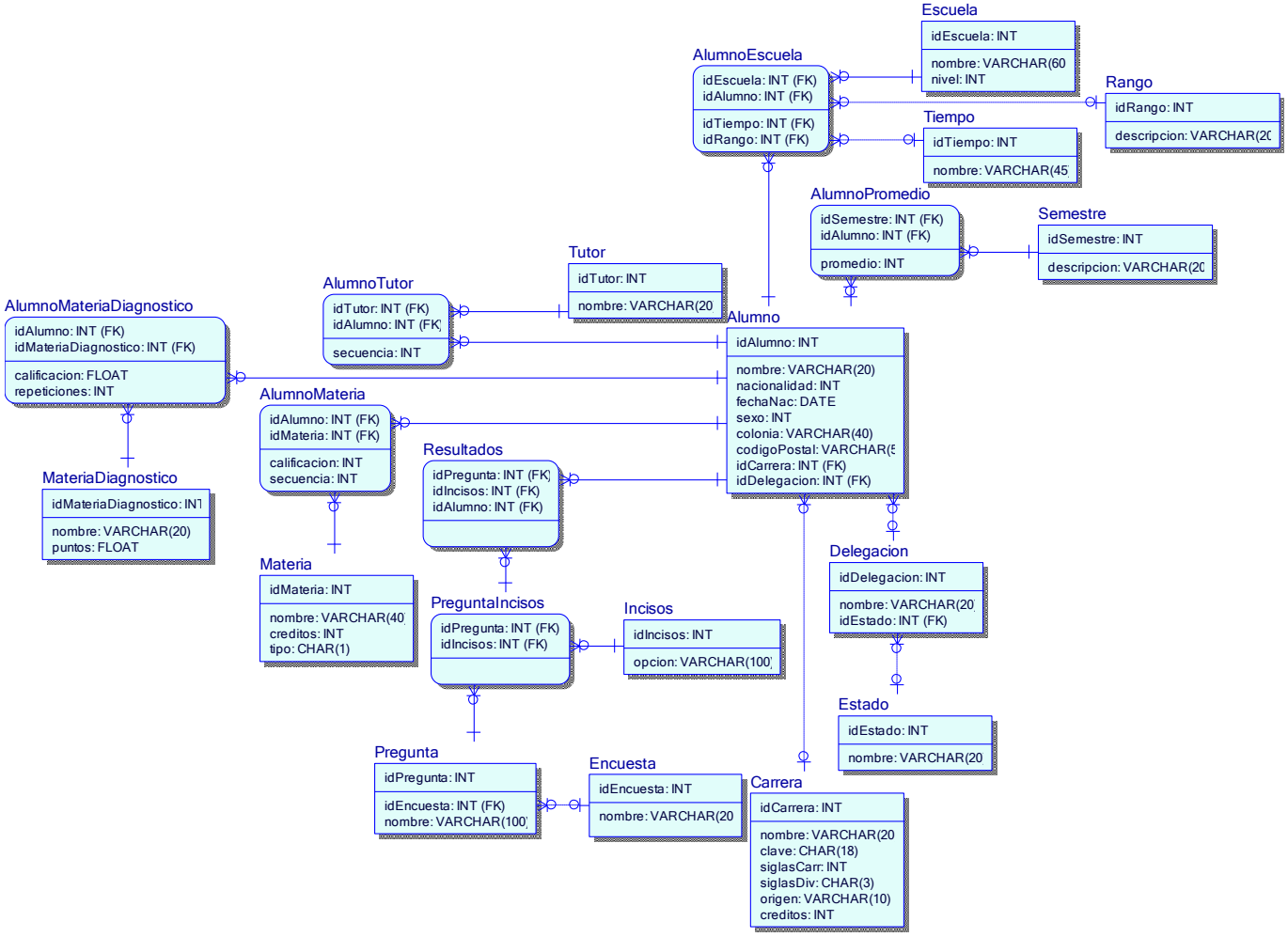

FIGURA 3.3.1.a Modelo Entidad / Relación

3.3.2 Diccionario de datos corporativo

Tabla	Característica
alumno	Tabla que almacena los datos de los alumnos
alumnoescuela	Tabla que almacena alumnos y escuelas a las que asistieron
alumnomateria	Tabla que asocia alumnos y materias
alumnomateriadiagnostico	Tabla que asocia alumnos y materias de diagnostico
alumnopromedio	Tabla que asocia alumnos y semestres (con promedio)
alumnotutor	Tabla que asocia alumnos y tutores
carrera	Catalogo de carreras
delegacion	Catalogo de delegaciones
encuesta	Catalogo de encuestas [1] = En cuesta de tutores [2] = Encuesta Socioeconómica
escuela	Tabla que almacena los datos de las escuelas
estado	Catalogo de estados
incisos	Tabla que almacena los incisos a ser usados en las preguntas
materia	Catalogo de materias
materiadiagnostico	Catalogo de materias de diagnostico
pregunta	Tabla que almacena las preguntas usadas de todas la encuestas
preguntaincisos	Tabla que asocia las preguntas de las encuestas y sus incisos
rango	Tabla que almacena los rango de las calificaciones
resultados	Tabla con las respuestas de los alumnos
semestre	Catalogo de semestres
tiempo	Catalogo con rangos de tiempos en que el alumno curso cada nivel escolar.
tutor	Tabla que almacena los datos de los tutores

Tabla "alumno"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idAlumno	int	✔		✔		Identificador del alumno
idDelegacion	int		✔	✔		Identificador de la delegación
nombre	varchar(20)			✔		Nombre del alumno
nacionalidad	int			✔		[1] = Nacionalidad mexicana [2] = Nacionalidad extranjera
fechaNac	date			✔		En formato YYYY-MM-DD
sexo	int			✔		[1] = Sexo masculino [2] = Sexo femenino
colonia	varchar(40)					Colonia
codigoPostal	varchar(5)					CP del alumno
idCarrera	int		✔	✔		Identificador de su carrera

Definición:

```
CREATE TABLE alumno (
  idAlumno      int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador del alumno',
  idDelegacion  int NOT NULL,-- COMMENT 'Identificador de la delegación ',
  nombre        varchar(20) NOT NULL,-- COMMENT 'Nombre del alumno',
  nacionalidad   int NOT NULL,-- COMMENT ' [1] = Nacionalidad mexicana [2] = Nacionalidad
extranjera',
  fechaNac      date NOT NULL,-- COMMENT 'En formato YYYY-MM-DD',
  sexo          int NOT NULL,-- COMMENT '[1] = Sexo masculino [2] = Sexo femenino',
  colonia       varchar(40),-- COMMENT 'Colonia',
  codigoPostal  varchar(5),-- COMMENT 'CP del alumno',
  idCarrera     int NOT NULL,-- COMMENT 'Identificador de su carrera',
  /* Keys */
  PRIMARY KEY (idAlumno),
  /* Foreign keys */
  CONSTRAINT alumno_ibfk_1
    FOREIGN KEY (idDelegacion)
    REFERENCES delegacion(idDelegacion)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT foreign_key01
    FOREIGN KEY (idCarrera)
    REFERENCES carrera(idCarrera)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);

CREATE INDEX Alumno_FKIndex1
  ON alumno
  (idDelegacion);
CREATE INDEX Alumno_FKIndex2
  ON alumno
  (idCarrera);
```

Tabla "alumnoescuela"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idEscuela	int	✔	✔	✔		Id de la escuela
idAlumno	int	✔	✔	✔		Id del alumno
idTiempo	int		✔	✔		Id del tiempo cursado
idRango	int		✔	✔		Id del rango de cal.

Definición:

```
CREATE TABLE alumnoescuela (
  idEscuela  int  NOT NULL ,--COMMENT 'Id de la escuela',
  idAlumno   int  NOT NULL ,--COMMENT 'Id del alumno',
  idTiempo   int NOT NULL ,--COMMENT 'Id del tiempo cursado',
  idRango    int  NOT NULL ,--COMMENT 'Id del rango de cal.',
  /* Keys */
  PRIMARY KEY (idEscuela, idAlumno),
  /* Foreign keys */
```

```
  CONSTRAINT FK_alumnoescuela_1
    FOREIGN KEY (idAlumno)
    REFERENCES alumno(idAlumno)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT FK_alumnoescuela_2
    FOREIGN KEY (idTiempo)
    REFERENCES tiempo(idTiempo)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT FK_alumnoescuela_3
    FOREIGN KEY (idEscuela)
    REFERENCES escuela(idEscuela)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT FK_alumnoescuela_4
    FOREIGN KEY (idRango)
    REFERENCES rango(idRango)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);

CREATE INDEX AlumnoEscuela_FKIndex3
  ON alumnoescuela
  (idRango);

CREATE INDEX AlumnoEscuela_FKIndex4
  ON alumnoescuela
  (idTiempo);

CREATE INDEX Alumno_has_Escuela_FKIndex1
  ON alumnoescuela
  (idAlumno);

CREATE INDEX Alumno_has_Escuela_FKIndex2
  ON alumnoescuela
  (idEscuela);
```

Tabla "alumnomateria"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idAlumno	int	✔	✔	✔		Identificador del alumno
idMateria	int	✔	✔	✔		Identificador de la materia
calificacion	int					[0] = NP
secuencia	int			✔	1	Vez que el alumno la cursa

Definición:

```
CREATE TABLE alumnomateria (
  idAlumno      int NOT NULL ,--COMMENT 'Identificador del alumno',
  idMateria     int NOT NULL ,--COMMENT 'Identificador de la materia',
  calificacion  int ,--COMMENT '[0] = NP',
  secuencia     int NOT NULL DEFAULT '1' ,--COMMENT 'Vez que el alumno la cursa',
  /* Keys */
```

```
  PRIMARY KEY (idAlumno, idMateria),
  /* Foreign keys */
  CONSTRAINT alumnomateria_ibfk_1
    FOREIGN KEY (idAlumno)
    REFERENCES alumno(idAlumno)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT alumnomateria_ibfk_2
    FOREIGN KEY (idMateria)
    REFERENCES materia(idMateria)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);

CREATE INDEX Alumno_has_Materia_FKIndex1
  ON alumnomateria
  (idAlumno);

CREATE INDEX Alumno_has_Materia_FKIndex2
  ON alumnomateria
  (idMateria);
```

Tabla "alumnomateriadiagnostico"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idAlumno	int	✔	✔	✔		Identificador del alumno
idMateriaDiagnostico	int	✔	✔	✔		Identificador de la materia
calificacion	float					Calificación del alumno
repeticiones	int					Repeticiones que realizo

Definición:

```
CREATE TABLE alumnomateriadiagnostico (
  idAlumno              int  NOT NULL ,--COMMENT 'Identificador del alumno',
  idMateriaDiagnostico  int  NOT NULL ,--COMMENT 'Identificador de la materia',
  calificacion          float ,--COMMENT 'Calificación del alumno',
  repeticiones          int  ,--COMMENT 'Repeticiones que realizo',
  /* Keys */
  PRIMARY KEY (idAlumno, idMateriaDiagnostico),
  /* Foreign keys */
  CONSTRAINT alumnomateriadiagnostico_ibfk_1
    FOREIGN KEY (idAlumno)
    REFERENCES alumno(idAlumno)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT alumnomateriadiagnostico_ibfk_2
    FOREIGN KEY (idMateriaDiagnostico)
    REFERENCES materiadiagnostico(idMateriaDiagnostico)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);

CREATE INDEX Alumno_has_MateriaDiagnostico_FKIndex1
  ON alumnomateriadiagnostico
```

```
  (idAlumno);

CREATE INDEX Alumno_has_MateriaDiagnostico_FKIndex2
  ON alumnomateriadiagnostico
  (idMateriaDiagnostico);
```

Tabla "alumnopromedio"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idSemestre	int	✔	✔	✔		Identificador del semestre
idAlumno	int	✔	✔	✔		Identificador del alumno
promedio	int					Promedio del alumno en el semestre

Definición:

```
CREATE TABLE alumnopromedio (
  idSemestre  int  NOT NULL,--COMMENT 'Identificador del semestre',
  idAlumno int  NOT NULL ,--COMMENT 'Identificador del alumno\r\n',
  promedio int ,--COMMENT 'Promedio del alumno en el semestre',
  /* Keys */
  PRIMARY KEY (idSemestre, idAlumno),
  /* Foreign keys */
  CONSTRAINT alumnopromedio_ibfk_1
    FOREIGN KEY (idAlumno)
    REFERENCES alumno(idAlumno)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT alumnopromedio_ibfk_2
    FOREIGN KEY (idSemestre)
    REFERENCES semestre(idSemestre)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);

CREATE INDEX Semestre_has_Alumno_FKIndex1
  ON alumnopromedio
  (idSemestre);

CREATE INDEX Semestre_has_Alumno_FKIndex2
  ON alumnopromedio
  (idAlumno);
```

Tabla "alumnotutor"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idAlumno	int	✔	✔	✔		Identificador del alumno
idTutor	int	✔	✔	✔		Identificador del tutor
secuencia	in				1	[1] = Primer tutor [2] = Segundo tutor [3] = ...

Definición:

```
CREATE TABLE alumnotutor (
  idAlumno    int  NOT NULL ,--COMMENT 'Identificador del alumno',
  idTutor     int  NOT NULL ,--COMMENT 'Identificador del tutor\r\n',
  secuencia   int  DEFAULT '1' ,--COMMENT '[1] = Primer tutor [2] = Segundo tutor [3] = .....',
  /* Keys */
  PRIMARY KEY (idAlumno, idTutor),
  /* Foreign keys */
  CONSTRAINT alumnotutor_ibfk_1
    FOREIGN KEY (idTutor)
    REFERENCES tutor(idTutor)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT alumnotutor_ibfk_2
    FOREIGN KEY (idAlumno)
    REFERENCES alumno(idAlumno)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);

CREATE INDEX Tutor_has_Alumno_FKIndex1
  ON alumnotutor
  (idTutor);

CREATE INDEX Tutor_has_Alumno_FKIndex2
  ON alumnotutor
  (idAlumno);
```

Tabla "carrera"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idCarrera	int	✔		✔		Identificador de la carrera
nombre	varchar(20)			✔		Nombre de la carrera
clave	int			✔		Clave de la carrera
siglasCarr	char(3)			✔		Siglas de la carrera
siglasDiv	varchar(10)			✔		Siglas de su división
origen	varchar(10)			✔		Origen
creditos	int					Créditos que tiene la carrera

Definición:

```
CREATE TABLE carrera (
  idCarrera   int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador de la carrera\r\n',
  nombre      varchar(20) NOT NULL,-- COMMENT 'Nombre de la carrera\r\n',
  clave       int NOT NULL,-- COMMENT 'Clave de la carrera',
  siglasCarr  char(3) NOT NULL, -- COMMENT 'Siglas de la carrera',
  siglasDiv   varchar(10) NOT NULL,-- COMMENT 'Siglas de su division',
  origen     varchar(10) NOT NULL,-- COMMENT 'Origen',
```

```
  creditos    int -- COMMENT 'Creditos que tiene la carrera',
  /* Keys */
  PRIMARY KEY (idCarrera)
)
```

Tabla "delegacion"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idDelegacion	int	✔		✔		Identificador de la delegación
idEstado	int		✔	✔		Identificador del estado
nombre	varchar(20)			✔		Nombre de la delegación

Definición:

```
CREATE TABLE delegacion (
  idDelegacion  int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador de la delegación\r\n',
  idEstado      int NOT NULL,-- COMMENT 'Identificador del estado',
  nombre        varchar(20) NOT NULL,-- COMMENT 'Nombre de la delegación',
  /* Keys */
  PRIMARY KEY (idDelegacion),
  /* Foreign keys */
  CONSTRAINT delegacion_ibfk_1
    FOREIGN KEY (idEstado)
    REFERENCES estado(idEstado)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);

CREATE INDEX Delegacion_FKIndex1
  ON delegacion
  (idEstado);
```

Tabla "encuesta"

Campo	Tipo de Dato	PK	FK	Not Null	Default	Descripción
idEncuesta	int	✔		✔		Identificador de la encuesta
nombre	varchar(20)			✔		Nombre de la encuesta

Definición:

```
CREATE TABLE encuesta (
  idEncuesta  int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador de la encuenta',
  nombre     varchar(20) NOT NULL,-- COMMENT 'Nombre de la encuesta',
  /* Keys */
  PRIMARY KEY (idEncuesta)
)
```

Tabla "escuela"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idEscuela	int(	✔		✔		Identificador de la escuela
nombre	varchar(60)			✔		Nombre de la escuela
nivel	int			✔		[1] = Primaria [2] = Secundaria [3] = Bachillerato

Definición:

```
CREATE TABLE escuela (
  idEscuela  int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador de la escuela',
  nombre     varchar(60) NOT NULL,--COMMENT 'Nombre de la escuela',
  nivel      int NOT NULL,-- COMMENT '[1] = Primaria  [2] = Secundaria  [3] = Bachillerato',
  /* Keys */
  PRIMARY KEY (idEscuela)
);
```

Tabla "estado"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idEstado	int	✔		✔		Identificador del Estado
nombre	varchar(20)			✔		Nombre del estado

Definición:

```
CREATE TABLE estado (
  idEstado  int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador del Estado\r\n',
  nombre    varchar(20) NOT NULL,-- COMMENT 'Nombre del estado',
  /* Keys */
  PRIMARY KEY (idEstado)
);
```

Tabla "incisos"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idIncisos	int	✔		✔		Identificador del inciso
opcion	varchar(100)			✔		Texto del inciso

Definición:

```
CREATE TABLE incisos (
  idIncisos  int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador del inciso',
  opcion     varchar(100) NOT NULL,-- COMMENT 'Texto del inciso',
  /* Keys */
  PRIMARY KEY (idIncisos)
);
```

Tabla "materia"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idMateria	int	✔		✔		Identificador de la materia
nombre	varchar(40)					Nombre de la materia
creditos	int					Créditos de la materia
tipo	char					Tipo de materia

Definición:

```
CREATE TABLE materia (
  idMateria  int IDENTITY(1,1) NOT NULL, -- COMMENT 'Identificador de la materia'
  nombre     varchar(40) , -- COMMENT 'Nombre de la materia'
  creditos   int, -- COMMENT 'Creditos de la materia'
  tipo       char , -- COMMENT 'Tipo de materia'
  /* Keys */
  PRIMARY KEY (idMateria)
);
```

Tabla "materiadiagnostico"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idMateriaDiagnostico	int	✔		✔		Identificador de la materia de diagnostico
nombre	varchar(20)			✔		Nombre de la materia de diagnostico
puntos	float			✔		Puntos de la materia

Definición:

```
CREATE TABLE materiadiagnostico (
  idMateriaDiagnostico  int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador de la materia de
diagnostico',
  nombre                varchar(20) NOT NULL,-- COMMENT 'Nombre de la materia de diagnostico',
  puntos               float NOT NULL,-- COMMENT 'Puntos de la materia',
  /* Keys */
  PRIMARY KEY (idMateriaDiagnostico)
);
```

Tabla "pregunta"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idPregunta	int	✔		✔		Identificador de la pegunta
idEncuesta	int		✔	✔		Identificador de la encuesta a la que pertenece
nombre	varchar(140)			✔		Texto de la pregunta

Definición:

```
CREATE TABLE pregunta (
  idPregunta  int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador de la pegunta',
  idEncuesta  int NOT NULL,-- COMMENT 'Identificador de la encuesta a la que pertenece',
  nombre      varchar(140) NOT NULL,-- COMMENT 'Texto de la pregunta',
  /* Keys */
  PRIMARY KEY (idPregunta),
  /* Foreign keys */
  CONSTRAINT pregunta_ibfk_1
    FOREIGN KEY (idEncuesta)
    REFERENCES encuesta(idEncuesta)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);

CREATE INDEX Pregunta_FKIndex1
  ON pregunta
  (idEncuesta);
```

Tabla "preguntaincisos"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idPregunta	int	✔	✔	✔		Identificador de la pregunta
idIncisos	int	✔	✔	✔		Identificador del inciso que tendrá la pregunta

Definición:

```
CREATE TABLE preguntaincisos (
  idPregunta  int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador de la pregunta',
  idIncisos   int  NOT NULL,-- COMMENT 'Identificador del inciso que tendra la pregunta',
  /* Keys */
  PRIMARY KEY (idPregunta, idIncisos),
  /* Foreign keys */
  CONSTRAINT preguntaincisos_ibfk_1
    FOREIGN KEY (idPregunta)
    REFERENCES pregunta(idPregunta)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT preguntaincisos_ibfk_2
    FOREIGN KEY (idIncisos)
```

```
    REFERENCES incisos(idIncisos)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
CREATE INDEX Pregunta_has_Incisos_FKIndex1
  ON preguntaincisos
  (idPregunta);

CREATE INDEX Pregunta_has_Incisos_FKIndex2
  ON preguntaincisos
  (idIncisos);
```

Tabla "rango"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idRango	int	✔		✔		Identificador del rango
descripcion	varchar(20)			✔		Rango de promedio de calificación con que salió el alumno de esa escuela

Definición:

```
CREATE TABLE rango (
  idRango      int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador del rango',
  descripcion   varchar(20) NOT NULL,-- COMMENT 'Rango de promedio de calificacion con que
salio el alumno de esa escuela',
  /* Keys */
  PRIMARY KEY (idRango));
```

Tabla "resultados"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idAlumno	int	✔	✔	✔		Identificador del alumno
idIncisos	int	✔	✔	✔		Identificador del inciso
idPregunta	int	✔	✔	✔		Identificador de la pregunta

Definición:

```
CREATE TABLE resultados (
  idAlumno    int  NOT NULL ,--COMMENT 'Identificador del alumno',
  idIncisos   int  NOT NULL ,--COMMENT 'Identificador del inciso',
  idPregunta  int  NOT NULL ,--COMMENT 'Identificador de la pregunta',
  /* Keys */
  PRIMARY KEY (idAlumno, idIncisos, idPregunta),
  /* Foreign keys */
  CONSTRAINT resultados_ibfk_1
    FOREIGN KEY (idPregunta, idIncisos)
    REFERENCES preguntaincisos(idPregunta, idIncisos)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT resultados_ibfk_2
    FOREIGN KEY (idAlumno)
    REFERENCES alumno(idAlumno)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

CREATE INDEX PreguntaIncisos_has_Alumno_FKIndex1
  ON resultados
  (idPregunta, idIncisos);

CREATE INDEX PreguntaIncisos_has_Alumno_FKIndex2
  ON resultados
  (idAlumno);
```

Tabla "semestre"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idSemestre	int	✔		✔		Identificador del semestre
descripcion	varchar(20)					Semestre en cuestión

Definition:

```
CREATE TABLE semestre (
  idSemestre   int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador del semestre\r\n',
  descripcion  varchar(20),-- COMMENT 'Semestre en cuestión',
  /* Keys */
  PRIMARY KEY (idSemestre)
);
```

Tabla "tiempo"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idTiempo	int	✔		✔		Identificador de la tabla
nombre	varchar(45)					Años cursados

Definición:

```
CREATE TABLE tiempo (
  idTiempo  int IDENTITY(1,1) NOT NULL,-- COMMENT 'Identificador de la tabla\r\n',
  nombre    varchar(45),-- COMMENT 'Años cursados',
  /* Keys */
  PRIMARY KEY (idTiempo)
);
```

Tabla "tutor"

Campo	Tipo de Dato	PK	FK	No nulo	Por defecto	Descripción
idTutor	int	✔		✔		Identificador del tutor
Nombre	varchar(20)					Nombre del tutor

Definición:

```
CREATE TABLE tutor (
  idTutor  int IDENTITY(1,1) NOT NULL,--COMMENT 'Identificador del tutor',
  Nombre   varchar(20),-- COMMENT 'Nombre del tutor',
  /* Keys */
  PRIMARY KEY (idTutor)
);
```

3.4 Herramientas y técnicas de análisis de datos y evaluación de las mismas

Microsoft SQL Server proporciona varios algoritmos que se pueden usar en las soluciones de minería de datos. Estos algoritmos son un subconjunto de todos los algoritmos que pueden utilizarse en la minería de datos. También permite utilizar algoritmos de minería de datos desarrollados por terceros que cumplan la especificación OLE DB para minería de datos. [21]

El siguiente diagrama describe las relaciones entre cada paso del proceso y las tecnologías de Microsoft SQL Server que se pueden utilizar para completar cada paso.

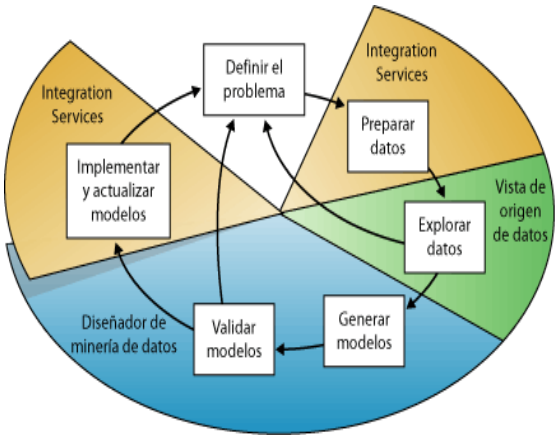

FIGURA 3.4.a Diagrama de procesos SQL Server

Aunque el proceso que se ilustra en el diagrama es circular, esto no significa que cada paso conduzca directamente al siguiente. La creación de un modelo de minería de datos es un proceso dinámico e iterativo. Una vez que ha explorado los datos, puede que descubra que resultan insuficientes para crear los modelos de minería

de datos adecuados y que, por tanto, debe buscar más datos. Pueden ser generados varios modelos y descubrir que no responden al problema planteado cuando fue definido y que, por tanto, se debe volver a definir el problema. Es posible que en ocasiones se tenga la necesidad de actualizar los modelos una vez implementados debido a que haya más datos disponibles.

SQL Server ofrece así un entorno integrado para crear y trabajar con modelos de minería de datos denominado *Business Intelligence Development Studio*. El entorno incluye algoritmos y herramientas de minería de datos que facilitan la generación de una solución completa para diversos proyectos. [21]

Ventajas:

- Permite a los usuarios más avanzados optimizar los modelos al cambiar valores de parámetros de algoritmos.

- Incluye un juego completo de herramientas de visualización.

- Permite graficar la relativa exactitud de todos sus modelos para columnas de predicción.

La siguiente tabla nos proporciona sugerencias sobre qué algoritmos usar en tareas específicas dentro de SQL Server 2008:

Tarea	Algoritmos de Microsoft que se pueden usar
Predecir un atributo discreto. Por ejemplo, predecir si el destinatario de una campaña de correo directo adquirirá un producto.	Algoritmo de árboles de decisión de Microsoft Algoritmo Naive Bayes de Microsoft Algoritmo de clústeres de Microsoft Algoritmo de red neuronal de Microsoft
Predecir un atributo continuo. Por ejemplo, prever las ventas del año próximo.	Algoritmo de árboles de decisión de Microsoft Algoritmo de serie temporal de Microsoft
Predecir una secuencia. Por ejemplo, realizar un análisis del flujo de clics en el sitio Web de una empresa.	Algoritmo de clústeres de secuencia de Microsoft
Buscar grupos de elementos comunes en transacciones. Por ejemplo, utilizar el análisis de la cesta de compra para sugerir a un cliente la compra de productos adicionales.	Algoritmo de asociación de Microsoft Algoritmo de árboles de decisión de Microsoft
Buscar grupos de elementos similares. Por ejemplo, segmentar datos demográficos en grupos a fin de comprender mejor las relaciones entre los atributos.	Algoritmo de clústeres de Microsoft Algoritmo de clústeres de secuencia de Microsoft

FIGURA 3.4.b Tabla de algoritmos [21]

Microsoft SQL Server Analysis Services

Microsoft SQL Server Analysis Services contiene las características y herramientas necesarias para crear complejas soluciones de minería de datos:

- Un conjunto de algoritmos de minería de datos estándar del sector.

- El diseñador de minería de datos sirve para crear, administrar y examinar modelos de minería de datos para, a continuación, crear predicciones a partir de dichos modelos.

- El lenguaje DMX (Extensiones de minería de datos), que sirve para administrar modelos de minería de datos y crear complejas consultas predictivas.

Se pueden usar varias de estas características y herramientas a la vez para detectar las tendencias y los patrones

existentes en los datos; después, se pueden usar esas tendencias y los patrones para tomar decisiones informadas sobre los problemas que se tengan que resolver. [22]

FIGURA 3.4.c *Integration Services* [22]

La imagen anterior muestra los distintos componentes que conformar al *Integration Services,* uno de estos componentes en el *Analysis Services*, el cual contiene las herramientas de minería de datos antes mencionadas.

Business Intelligence Development Studio

SQL Server Business Intelligence Development Studio es un entorno integrado para desarrollar construcciones de inteligencia empresarial, como orígenes de datos, cubos, informes y paquetes de *Integration Services. Business Intelligence Development Studio* incluye plantillas de proyecto que proporcionan un contexto para desarrollar construcciones específicas. También es posible desarrollar proyectos que formen parte de una solución independiente de un servidor concreto. [23]

En *Business Intelligence Development Studio*, el asistente para minería de datos facilita la creación de estructuras y de modelos de minería de datos basados en orígenes de datos OLAP y relacionales. Uno puede utilizar el asistente para definir estructuras y modelos que utilicen técnicas de minería de datos específicas para analizar datos. Así mismo es posible utilizar el diseñador de minería de datos para perfeccionar la definición de modelos de minería datos y explorar y trabajar con los resultados del modelo. [23]

SQL Server Management Studio

SQL Server Management Studio proporciona herramientas que pueden ser utilizadas para administrar y explorar los modelos de minería de datos una vez creados. [24]

Transformaciones y tareas de minería de datos en Integration Services

SQL Server Integration Services proporciona herramientas que pueden ser utilizadas para automatizar tareas comunes de minería de datos, como procesar un modelo de minería de datos y crear consultas de predicción. Por ejemplo, si se dispone de un modelo de minería de datos generado a partir de un conjunto de datos de posibles clientes, se puede crear un paquete de *Integration Services* que actualice automáticamente el modelo cada vez que el conjunto de datos se actualice con nuevos clientes. A continuación se podría utilizar el paquete para crear una predicción, separando los clientes potenciales en dos tablas. Una tabla contendría los clientes probables y la otra los clientes que posiblemente no adquirirán ningún producto. [24]