Entrega Final
Calidad de Datos e Información

Context-aware Data Quality Methodology (CaDQM)

Grupo 11

Bruno Ottonelli (4.954.242-1)
Gabriel Rode (4.535.978-1)

Universidad de la República
Facultad de Ingeniería

Junio 2025

Índice general

Capítulo 1

Introducción

En este trabajo presentamos la aplicación práctica del método **CaDQM** (Context-aware Data Quality Methodology). El objetivo fue, mediante dicha metodología, evaluar y tratar de mejorar la calidad de los datos en un caso particular que simula la fusión de dos librerías, L1 y L2, en una nueva entidad llamada NL.

Durante el desarrollo seguimos las fases definidas por CaDQM: *Data Quality Planning*, *Data Quality Assessment* y *Data Quality Improvement*, lo que nos permitió abordar el trabajo de forma estructurada y ordenada.

En este informe detallamos el proceso seguido, las decisiones que tomamos en cada etapa y cómo la metodología CaDQM nos ayudó a obtener un diagnóstico preciso del estado de calidad de los datos tratados.

Nota: la fase de *Data Quality Improvement* fue aplicada de forma parcial.

Capítulo 2

Fase 1 - Data Quality Planning

2.1. ST1: Elicitation

2.1.1. Entradas y salidas

Entradas y salidas	
Entradas	**Salidas**
	Base de datos integrada (Data at hand) (2.1.5)
	Reporte con problemas de CD (2.1.4)
	Modelo de contexto (2.1.3)

CD: Calidad de datos.

2.1.2. Descripción de la realidad

El análisis de la calidad de los datos se centrará en información relacionada con libros, proveniente de dos librerías (L1 y L2) que se fusionarán en una nueva librería NL.

Los encargados de la librería reportan saber de la existencia de muchos problemas de calidad (sin explicitar cuáles) y que éstos se verán potenciados luego de la fusión. Es por esto que la situación genera la necesidad de evaluar la calidad de los datos en la base resultante de la integración de los datos proporcionados por ambas librerías.

Los datos proporcionados por la librería L1 estan distribuidos en dos archivos CSV, uno con detalles de los libros y otros con detalles de las reviews sobre los libros. Ambas tablas se relacionan entre si mediante el atributo *Title*.

Tablas de la librería L1	
Nombre de tabla	**Atributos**
Books_rating	'Id', 'Title', 'Price', 'User_id', 'profileName', 'review/helpfulness', 'review/score', 'review/time', 'review/summary', 'review/text'
books_data	'Title', 'description', 'authors', 'image', 'previewLink', 'publisher', 'publishedDate', 'infoLink', 'categories', 'ratingsCount'

De la librería L2 se obtienen los datos en tres archivos CSV, uno con detalles de los libros, otro con detalle de los usuarios y el último con detalle de las valoraciones de dichos usuarios sobre los libros. Entre ellas se relacionan mediante los atributos *ISBN*, que es un identificador para los libros, y *User-ID*, que es un identificador para los usuarios.

Tablas de la librería L2	
Nombre de tabla	**Atributos**
Books	'ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher', 'Image-URL-S', 'Image-URL-M', 'Image-URL-L'
Users	'User-ID', 'Location', 'Age'
Ratings	'User-ID', 'ISBN', 'Book-Rating'

2.1.3. Contexto

Identificación de componentes del contexto

Componentes de Contexto	
Dominio	D: Libros.
Fuentes de datos	Datos obtenidos de ambas librerías que se fusionarán y los proporcionados por el cliente sobre sus realidades.
Tipos de usuario	U1: Administrador. U2: Publicista digital. U3: Analista de datos.
Tareas	T1: Gestión. T2: Análisis. T3: Consulta.
Reglas de negocio	RN1: Cada libro deberá tener asociado un ISBN, un título, al menos un autor y un editor.
Requerimientos de calidad	RQ1: Frescura de datos: la base debe actualizarse todos los viernes. RQ2: Al menos el 80 % de los usuarios que califican los libros deben ser mayores de 18 años. RQ3: Al menos el 95 % de los libros deben cumplir simultáneamente con los siguientes requisitos: contar con un ISBN, tener el título correctamente escrito y que el nombre del autor incluya al menos un nombre y un apellido. RQ4: Al menos el 60 % de los libros tengan al menos un score mayor o igual a 5. RQ5: La librería pretende tener al menos 500 libros y poseer al menos el 20 % de la lista de los 100 mejores libros de Goodreads. RQ6: Los libros deben contar con fecha de publicación. RQ7: Los libros deben tener editorial. RQ8: Los libros deben tener asignado un valor de score.
Requerimientos del sistema	RS1: Los tiempos de respuesta del sitio Web de la NL no pueden superar los 3 segundos.
Problemas de calidad ya reportados	Ninguno en particular.
Necesidades de filtrado	F1: Libros por fecha (en particular, del año actual). F2: Libros por editorial. F3: Top de libros según su score.

Todos los componentes de contexto surgen de analizar la realidad planteada por el cliente (letra proporcionada para la entrega), sin embargo, los requerimientos RQ6, RQ7 y RQ8 no están detallados explícitamente, sino que surgen de las necesidades de filtrado para que los usuarios puedan realizar correctamente sus labores.

Contexto

Contexto				
Componente de contexto	**Todos los usuarios**	**U1: Administrador**	**U2: Publicista**	**U3: Analista**
Dominio	D			
Tareas	T3	T1	T2	
Reglas de negocio	RN1			
Requerimientos de sistema	RS1			
Requerimientos de calidad de datos	RQ5	RQ6, RQ7, RQ8	RQ1, RQ2, RQ4	RQ3
Necesidades de filtrado		F1, F2, F3		
Metadatos				
Metadatos de calidad de datos				
Otros datos				

2.1.4. Primeros problemas de calidad de datos

Al momento de la integración de la base de datos se identifica como problema de calidad que los datos provenientes de las librerías no contienen los mismos atributos, mencionamos algunos de los ejemplos más importantes:

- En el caso de la librería 1 la información referente a libros (*booksData*) no contiene el ISBN que identifica al mismo y solo aparece si los libros poseen una entrada en el libro de ratings (*booksRating*).
- La librería 2 tiene identificado a los usuarios con un entero autogenerado y solo registra la edad y ubicación del mismo mientras que la librería 1 registra mas datos (como el nombre) pero no registra la edad del mismo.

2.1.5. Data at hand

El *Data at hand* sobre el que se evaluará la calidad será la base de datos resultante de integrar las bases existentes de ambas librerías. Para ello, se identificaron atributos de ambas bases que representen lo mismo y se ideó una nueva estructura en la que se presentará la información, sin modificar los datos existentes.

La base de datos integrada para la librería NL consta de las tablas y estructura que se muestran en la Figura 2.1.

La migración de las tablas de L1 y L2 a NL se hizo como se muestra en las tablas de la presente sección, donde la columna izquierda son los atributos en las tablas originales de L1 y L2, y la columna derecha es en qué atributos se incluyeron de la base de NL.

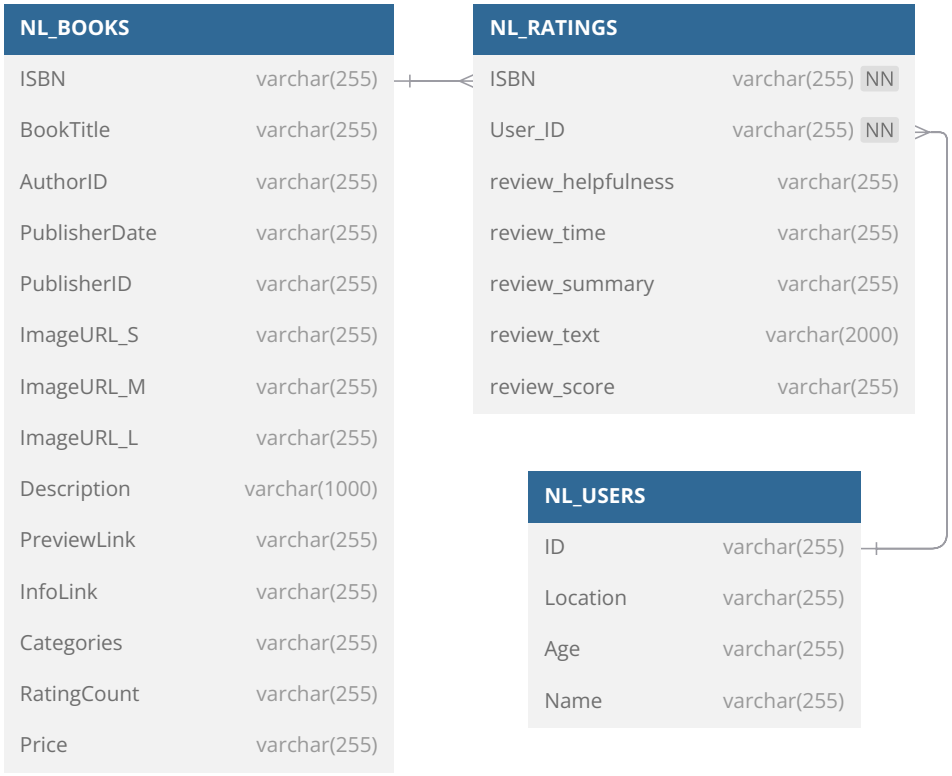

Figura 2.1: Estructura de la base de datos creada

Tabla 2.1: Migración: L1_bookData → NL_BOOKS

Books	
Entradas (L1_bookData)	**Salidas (NL_BOOKS)**
ISBN	NL_Ratings.ID (relación con BookTitle)
Title	BookTitle
description	Description
authors	AuthorID
image	ImageURL_S
previewLink	PreviewLink
publisher	PublisherID
publishedDate	PublisherDate
infoLink	InfoLink
categories	Categories
ratingsCount	RatingCount
Price	Price

Tabla 2.2: Migración: L1_booksRating → NL_USERS

Users	
Entradas (L1_bookRating)	**Salidas (NL_USERS)**
User_ID	ID
profileName	Name

Tabla 2.3: Migración: L1_booksRatings → NL_RATINGS

Users	
Entradas (L1_booksRatings)	**Salidas (NL_RATINGS)**
Title	BookTitle
Price	Price
User_id	ID
profileName	Name
review_helpfulness	review_helpfulness
review_score	review_score
review_time	review_time
review summary	review_summary
review text	review_text

Tabla 2.4: Migración: L2_books → NL_BOOKS

Ratings	
Entradas (L2_books)	**Salidas (NL_BOOKS)**
ISBN	ISBN
Book-Title	BookTitle
Book-Author	AuthorID
Year-Of-Publication	PublisherDate
Publisher	PublisherID
Image-URL-S	ImageURL_S
Image-URL-M	ImageURL_M
Image-URL-L	ImageURL_L

Tabla 2.5: Migración: L2_users → NL_USERS

Entradas y salidas	
Entradas (L2_users)	**Salidas (NL_USERS)**
User-ID	ID
Location	Location
Age	Age

Tabla 2.6: Migración: L2_ratings → NL_RATINGS

Entradas y salidas	
Entradas (L2_ratings)	**Salidas (NL_RATINGS)**
User-ID	User_ID
ISBN	ISBN
Book-Rating	review_score

2.2. ST2: Data Analysis

2.2.1. Entradas y salidas

Entradas y salidas	
Entradas	**Salidas**
Base de datos integrada (Data at hand) (2.1.5)	Reporte de análisis de datos (2.2.2)
Reporte con problemas de CD (2.1.4)	Reporte con problemas de CD (Tabla 2.7)
Modelo de contexto (2.1.3)	Modelo de contexto (2.2.3)

CD: Calidad de datos.

2.2.2. Reporte del análisis de datos

Descripción de las herramientas y técnicas utilizadas para el data profiling

En el proceso de data profiling, se utilizaron diversas herramientas según cada etapa del flujo de trabajo:

- *Python* y *Pandas* para el preprocesamiento, formateo de los archivos en formato .csv y análisis de los datos.
- *DBDiagram.io* para el análisis y diseño de la estructura de la base de datos, facilitando su visualización y validación.
- *SQL Server* para la creación y gestión de la base de datos, consolidando los datos unificados.

Data profiling

Para analizar los datos del data at hand, el primer paso fue un estudio estadístico, que se resume en las siguientes tablas:

NL_Books

Column	Total	Not Null	Nulls	Unique	Repeated
ISBN	483,783	483,783	0	464,312	19,471
BookTitle	483,783	483,782	1	435,056	48,726
AuthorID	483,783	452,364	31,419	229,313	223,051
PublisherDate	483,783	458,477	25,306	11,744	446,733
PublisherID	483,783	407,902	75,881	29,912	377,990
ImageURL_S	483,783	271,379	212,404	271,063	316
ImageURL_M	483,783	431,709	52,074	420,449	11,260
ImageURL_L	483,783	271,379	212,404	271,063	316
Description	483,783	143,964	339,819	133,227	10,737
PreviewLink	483,783	188,566	295,217	188,093	473
InfoLink	483,783	188,565	295,218	184,498	4,067
Categories	483,783	171,210	312,573	10,969	160,241
RatingCount	483,783	49,813	433,970	538	49,275
Price	483,783	47,699	436,084	8,117	39,582

NL_Users

Column	Total	Not Null	Nulls	Unique	Repeated
ID	3,278,859	2,717,056	561,803	1,287,821	1,429,235
Location	3,278,859	278,858	3,000,001	57,339	221,519
Age	3,278,859	168,097	3,110,762	282	167,815
Name	3,278,859	2,438,094	840,765	854,151	1,583,943

NL_Ratings

Column	Total	Not Null	Nulls	Unique	Repeated
ISBN	4,149,780	4,149,780	0	545,325	3,604,455
User_ID	4,149,780	3,587,977	561,803	1,114,247	2,473,730
review_helpfulness	4,149,780	2,999,966	1,149,814	12,145	2,987,821
review_time	4,149,780	2,999,972	1,149,808	12,407	2,987,565
review_summary	4,149,780	2,999,593	1,150,187	1,592,295	1,407,298
review_text	4,149,780	2,999,992	1,149,788	2,062,655	937,337
review_score	4,149,780	4,149,780	0	74	4,149,706

En ellas se puede apreciar una gran cantidad de nulos y duplicados. Muchos de estos ya existían previamente en los datos originales proporcionados por ambas librerias, pero, sobre todo en el caso de los nulos, muchos se generaron debido a la integración, ya que ambas bases de datos poseian distintos atributos.

A continuación, se muestra de manera gráfica y porcentual los valores nulos y repetidos:

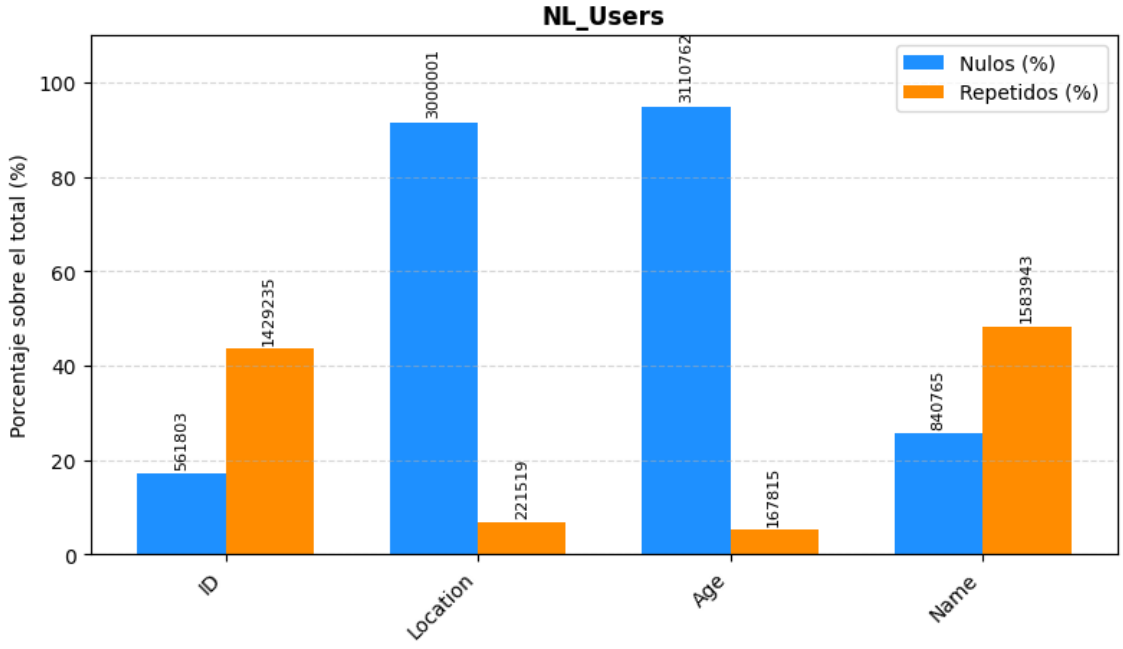

Figura 2.2: Gráfica de los valores nulos en la tabla `NL_Users`

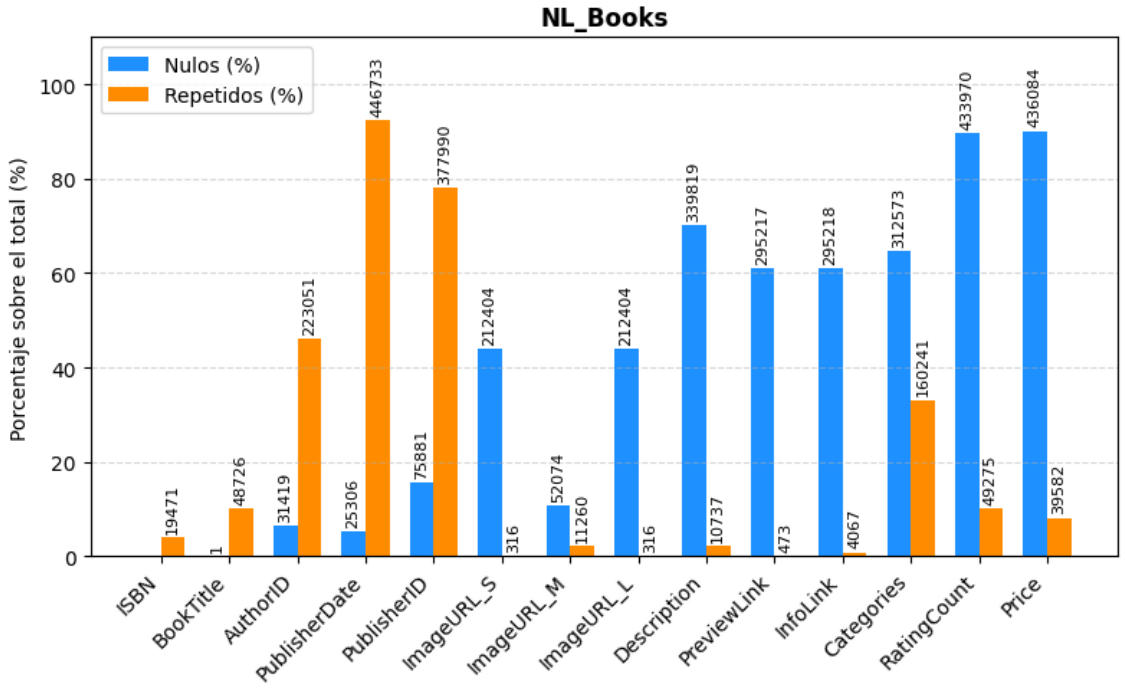

Figura 2.3: Gráfica de los valores nulos en la tabla `NL_Books`

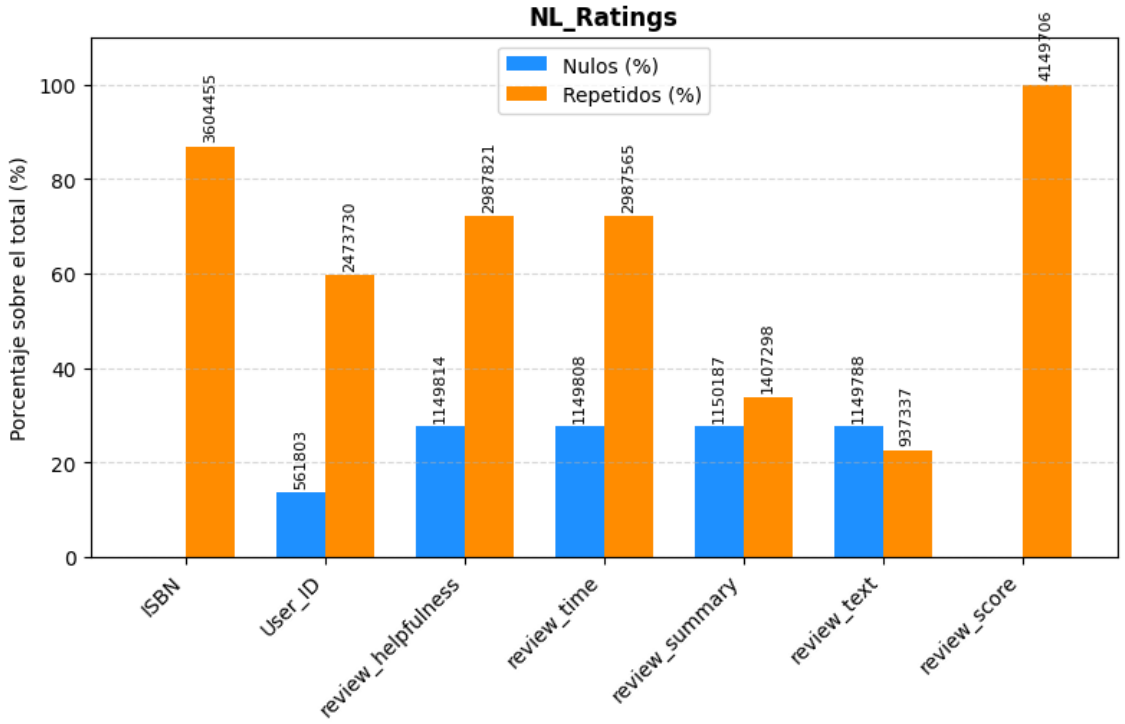

Figura 2.4: Gráfica de los valores nulos en la tabla `NL_Ratings`

Listado de los problemas de calidad detectados

Durante la ejecución del *Data Profiling* se detectaron múltiples errores sobre los datos, los cuales se detallan en la siguiente tabla junto a los ya reportados durante la ST1.

Cabe destacar que, si bien no fueron detectados explícitamente, se consideró oportuno incluir los errores P14 y P15 ya que es de interés del equipo de trabajo medirlos debido a que podrían afectar la calidad y coherencia de los datos.

Tabla 2.7: Problemas detectados en la Calidad de los Datos

Problemas detectados en la Calidad de los Datos	
Campo	**Problema de calidad**
NL_Books.ISBN	P1: Entradas no respetan el formato ISBN ya que algunos están en formato ASID.
NL_Books.PublisherDate	P2: Las fechas tienen distinto formato.
NL_Books.PublisherID	P3: Mismo publisher escrito de forma distinta.
NL_Books.Title	P4: Títulos mal escritos.
NL_Books.AuthorID	P5: Mismo autor escrito de forma distinta.
NL_USERS.Age	P6: Valores de edad poco coherentes (por ejemplo 0).
NL_USERS.Location	P7: Ciudades mal escritas.
NL_RATINGS.review_time	P8: Formato de fecha/hora inconsistente.
NL_RATINGS.review_score	P9: Valores no numéricos en la puntuación.
NL_RATINGS.review_score	P10: Los valores importados entre L1 y L2 manejan distintas escalas (L1 puntúa de 0 a 5 y L2 de 0 a 10).
Base de datos	P11: Gran cantidad de nulos en muchos de los atributos.
NL_Books	P12: No hay campo de rating promedio del libro (podría calcularse).
NL_RATINGS.Helpfullness	P13: Este atributo en realidad debería ser dos atributos diferentes: cantidad de votaciones en esa review y cantidad de votaciones que consideraron útil esa review.
NL_Books.AuthorID	P14: Libros indican autores de forma distinta cuando tienen más de uno (dos libros con autores {A,B} y {B,A} deben considerarse con los mismos autores).
NL_RATINGS.review_score	P15: Podría haber valores fuera del rango 0 a 10.

2.2.3. Contexto

Nuevos componentes de contexto

Dado que se busca a futuro tener una base de datos con un funcionamiento correcto y siguiendo la línea de los problemas de calidad detectados, durante el análisis de datos se identificaron los componentes de contexto RN2, RN3, RN4 y RN5.

Se presenta a continuación la lista completa de componentes de contexto, incluyendo las nuevas componentes mencionadas.

Componentes de Contexto	
Dominio	D: Libros.
Fuentes de datos	Datos obtenidos de ambas librerías que se fusionarán y los proporcionados por el cliente sobre sus realidades.
Tipos de usuario	U1: Administrador. U2: Publicista digital. U3: Analista de datos.
Tareas	T1: Gestión. T2: Análisis. T3: Consulta.
Reglas de negocio	RN1: Cada libro deberá tener asociado un ISBN, un título, al menos un autor y un editor. RN2: El atributo ISBN en NL_Books debe ser único a cada libro. RN3: El atributo Price en NL_Books debe ser un real positivo. RN4: El atributo Age en NL_Users debe ser un entero positivo. RN5: El atributo ID en NL_Users debe ser único y no vacío.
Requerimientos de calidad	RQ1: Frescura de datos: la base debe actualizarse todos los viernes. RQ2: Al menos el 80 % de los usuarios que califican los libros deben ser mayores de 18 años. RQ3: Al menos el 95 % de los libros deben cumplir simultáneamente con los siguientes requisitos: contar con un ISBN, tener el título correctamente escrito y que el nombre del autor incluya al menos un nombre y un apellido. RQ4: Al menos el 60 % de los libros tengan al menos un score mayor o igual a 5. RQ5: La librería pretende tener al menos 500 libros y poseer al menos el 20 % de la lista de los 100 mejores libros de Goodreads. RQ6: Los libros deben contar con fecha de publicación. RQ7: Los libros deben tener editorial. RQ8: Los libros deben tener asignado un valor de score.
Requerimientos del sistema	RS1: Los tiempos de respuesta del sitio Web de la NL no pueden superar los 3 segundos.
Necesidades de filtrado	F1: Libros por fecha (en particular, del año actual). F2: Libros por editorial. F3: Top de libros según su score.

Contexto

Contexto				
Componente de contexto	**Todos los usuarios**	**U1: Administrador**	**U2: Publicista**	**U3: Analista**
Dominio	D			
Tareas	T3	T1	T2	
Reglas de negocio	RN1, RN2, RN3, RN4, RN5			
Requerimientos de sistema	RS1			
Requerimientos de calidad de datos	RQ5	RQ6, RQ7, RQ8	RQ1, RQ2, RQ4	RQ3
Necesidades de filtrado		F1, F2, F3		
Metadatos				
Metadatos de calidad de datos				
Otros datos				

2.3. ST3: User requirements analysis

2.3.1. Entradas y salidas

Entradas y salidas	
Entradas	**Salidas**
Base de datos integrada (Data at hand) (2.1.5)	Reporte de análisis de requerimientos de usuarios (2.3.2)
Reporte con problemas de CD (Tabla 2.7)	Reporte con problemas de CD (Tabla 2.7)
Modelo de contexto (2.2.3)	Modelo de contexto (2.3.3)

CD: Calidad de datos.

2.3.2. Reporte de requerimientos de usuario

En esta etapa, luego de realizadas consultas al cliente, se identificaron los siguientes requerimientos de calidad.

Nuevas componentes de Contexto	
Requerimientos de calidad	RQ9: los nombres de las editoriales deben estar estandarizados. RQ10: Las reglas de formato para nombres (autores, libros, editoriales) son: primera letra del nombre propio en mayúsculas y sin punto al final. RQ11: El formato para las fechas será dd/mm/aaaa.

2.3.3. Contexto

Nuevas componentes de contexto

Se presenta a continuación la lista completa de componentes de contexto, incluyendo los requerimientos de calidad RQ9, RQ10 y RQ11 identificados durante la ejecución de esta etapa.

Componentes de Contexto	
Dominio	D: Libros.
Fuentes de datos	Datos obtenidos de ambas librerías que se fusionarán y los proporcionados por el cliente sobre sus realidades.
Tipos de usuario	U1: Administrador. U2: Publicista digital. U3: Analista de datos.
Tareas	T1: Gestión. T2: Análisis. T3: Consulta.
Reglas de negocio	RN1: Cada libro deberá tener asociado un ISBN, un título, al menos un autor y un editor. RN2: El atributo ISBN en NL_Books debe ser único a cada libro. RN3: El atributo Price en NL_Books debe ser un real positivo. RN4: El atributo Age en NL_Users debe ser un entero positivo. RN5: El atributo ID en NL_Users debe ser único y no vacío.
Requerimientos de calidad	RQ1: Frescura de datos: la base debe actualizarse todos los viernes. RQ2: Al menos el 80 % de los usuarios que califican los libros deben ser mayores de 18 años. RQ3: Al menos el 95 % de los libros deben cumplir simultáneamente con los siguientes requisitos: contar con un ISBN, tener el título correctamente escrito y que el nombre del autor incluya al menos un nombre y un apellido. RQ4: Al menos el 60 % de los libros tengan al menos un score mayor o igual a 5. RQ5: La librería pretende tener al menos 500 libros y poseer al menos el 20 % de la lista de los 100 mejores libros de Goodreads. RQ6: Los libros deben contar con fecha de publicación. RQ7: Los libros deben tener editorial. RQ8: Los libros deben tener asignado un valor de score. RQ9: los nombres de las editoriales deben estar estandarizados. RQ10: Las reglas de formato para nombres (autores, libros, editoriales) son: primera letra del nombre propio en mayúsculas y sin punto al final. RQ11: El formato para las fechas será dd/mm/aaaa.
Requerimientos del sistema	RS1: Los tiempos de respuesta del sitio Web de la NL no pueden superar los 3 segundos.
Necesidades de filtrado	F1: Libros por fecha (en particular, del año actual). F2: Libros por editorial. F3: Top de libros según su score.

Contexto

Contexto				
Componente de contexto	**Todos los usuarios**	**U1: Administrador**	**U2: Publicista**	**U3: Analista**
Dominio	D			
Tareas	T3	T1	T2	
Reglas de negocio	RN1, RN2, RN3, RN4, RN5			
Requerimientos de sistema	RS1			
Requerimientos de calidad de datos	RQ5, RQ9, RQ10, RQ11	RQ6, RQ7, RQ8	RQ1, RQ2, RQ4	RQ3
Necesidades de filtrado		F1, F2, F3		
Metadatos				
Metadatos de calidad de datos				
Otros datos				

2.4. Entradas y salidas de la Fase 1 completa

Como preparación para abordar la Fase 2 del modelo CaDQM, se presentan a continuación las entradas y salidas generales de la Fase 1.

Entradas y salidas	
Entradas	**Salidas**
	Base de datos integrada (Data at hand) (2.1.5)
	Reporte de análisis de datos (2.2.2)
	Reporte de análisis de requerimientos de usuarios (2.3.2)
	Reporte con problemas de CD (Tabla 2.7)
	Modelo de contexto (2.3.3)

CD: Calidad de datos.

2.5. Descripción del desarrollo

Siguiendo la metodología CaQDM, el primer paso fue identificar el contexto de la realidad planteada, identificando los elementos de contexto en ella para armar luego el modelo de contexto en base a estos. Luego se revisó la documentación disponible sobre las bases de datos proporcionadas, para entender su estructura y contenido, pero sin indagar en los datos.

Dado que los archivos estaban en formato CSV y su contenido tenia ciertas particularidades, fue necesario realizar algunas modificaciones tales como cambiar el caracter separador y ver que el contenido de las celdas estaba entre comillas, para que pudieran abrirse correctamente en *SQL Server*. Una vez adaptados y cargados correctamente, se diseñó la nueva estructura de la base de datos de NL donde se integraron las distintas fuentes de datos para conformar el data at hand.

Para la realización del proceso, el análisis de las tablas se realizó utilizando *Python* y la biblioteca *Pandas*, lo que permitió explorar y procesar los datos de manera rápida y sencilla. Para el diseño de la nueva base, se utilizaron herramientas visuales como *DBDiagram.io*, que luego se implementarían en *SQL server*.

2.6. Conclusiones de la fase 1

La implementación de la primer fase de CaQDM en este proyecto fue una experiencia interesante en el manejo de datos donde pudimos aplicar el conocimiento y la metodología vistos en clase.

Uno de los principales retos fue la migración de la información proveniente de los conjuntos **L1** y **L2**, los cuales estaban originalmente en archivos **csv** y, en el caso particular de uno de ellos, tenia un tamaño grande que hizo difícil su tratamiento por lo que fue necesario una etapa previa de tratamiento de los datos, además de particularidades en la forma que estaban almacenados los datos, que generaron complicaciones a la hora de importar. No obstante, estos problemas pudieron ser sorteados con éxito.

Por otro lado, al ser nuestra primer experiencia aplicando el modelo sobre un caso tan particular como este, donde se debian integrar datos, tuvimos ciertas dificultades al inicio en delimitar el alcance de cada una de las etapas del modelo, pero que luego fueron aclaradas y ejecutadas correctamente.

Finalmente consideramos un logro importante el haber podido aplicar de forma satisfactoria la **Fase 1 del marco CaQDM** (Calidad y Gestión de Datos), aplicando en un caso con información real.

Capítulo 3

Fase 2 - Data Quality Assessment

3.1. ST4: DQ Model Definition

3.1.1. Entradas y salidas

Entradas y salidas	
Entradas	**Salidas**
Reporte del análisis de requerimientos de usuarios (2.3.2)	Reporte de problemas de CD priorizados (3.1.2)
Reporte del análisis de datos (2.2.2)	Modelo de CD (3.1.4)
Reporte de problemas de CD (Tabla 2.7)	
Modelo de Contexto (2.3.3)	

CD: Calidad de datos.

3.1.2. Priorización de los problemas de calidad de datos

La siguiente tabla muestra los problemas de calidad de datos detectados durante la *Fase 1*, ya priorizados.

Para la priorización, se consideró que todos los problemas de calidad asociados a reglas de negocio tuvieran alta prioridad, dado que representan un interés central para el cliente o constituyen requisitos esenciales para el correcto funcionamiento de cualquier sistema de bases de datos.

Los errores de formato o que tienen que ver con una posible futura reestructuración de la base de datos fueron clasificados con una prioridad media, ya que, si bien no impiden el uso de los datos, pueden dificultar su utilización, presentación o interpretación.

Por otro lado, aquellos problemas relacionados con errores ortográficos, tipográficos o inconsistencias en la escritura se asignaron con prioridad baja. La única excepción a esta regla es el caso del *P3* con prioridad media, ya que un error en el *Publisher* interfiere directamente con la necesidad de filtrado *F2*, imposibilitando parte de las tareas del usuario *U1*.

Problemas detectados en la Calidad de los Datos		
Campo	**Problema de calidad**	**Prioridad**
NL_Books.ISBN	P1: Entradas no respetan el formato ISBN ya que algunos están en formato ASID.	Alta
NL_Books.PublisherDate	P2: Las fechas tienen distinto formato.	Media
NL_Books.PublisherID	P3: Mismo publisher escrito de forma distinta.	Media
NL_Books.Title	P4: Títulos mal escritos.	Baja
NL_Books.AuthorID	P5: Mismo autor escrito de forma distinta.	Baja
NL_USERS.Age	P6: Valores de edad poco coherentes (por ejemplo 0).	Alta
NL_USERS.Location	P7: Ciudades mal escritas.	Baja
NL_RATINGS.review_time	P8: Formato de fecha/hora inconsistente.	Media
NL_RATINGS.review_score	P9: Valores no numéricos en la puntuación.	Media
NL_RATINGS.review_score	P10: Los valores importados entre L1 y L2 manejan distintas escalas (L1 puntúa de 0 a 5 y L2 de 0 a 10).	Media
Base de datos	P11: Gran cantidad de nulos en muchos de los atributos, incluyendo claves.	Alta
NL_Books	P12: No hay campo de rating promedio del libro (podría calcularse).	Media
NL_RATINGS.Helpfullness	P13: Este atributo en realidad debería ser dos atributos diferentes: cantidad de votaciones en esa review y cantidad de votaciones que consideraron útil esa review.	Media
NL_Books.AuthorID	P14: Libros indican autores de forma distinta cuando tienen mas de uno (dos libros con autores {A,B} y {B,A} deben considerarse con los mismos autores).	Baja
NL_RATINGS.review_score	P15: Podría haber valores fueron del rango 0 a 10.	Media

3.1.3. Métricas

check_ISBN

Métrica	
Nombre	**check_ISBN**
Descripción	Controla si el valor es un ISBN válido.
Granularidad	Celda
Dominio del Resultado	{0, 1}

Método	
Descripción	Implementa la métrica Check_ISBN teniendo en cuenta la estructura de un código ISBN.
Tipos de datos de entrada	String
Tipos de datos de salida	Boolean

Algoritmo	`def Check_ISBN(codigo):`
	`  size = len(codigo)`
	`  # Chequea para el caso de un ISBN10`
	`  if (size == 10):`
	`  verifica que los primeros 9 caracteres sean dígitos`
	`  verifica que el último caracter sea un dígito o una "X"`
	`  verifica que el dígito verificador sea correcto`
	`  # Chequea para el caso de un ISBN13`
	`  if (size == 13):`
	`  verifica que todos los caracteres sean dígitos`
	`  verifica que el dígito verificador sea correcto`

Método aplicado	
Tipo	Medición
Descripción	Utiliza el algoritmo de cálculo de ISBN para verificar la validez del dato.
Aplicado a	Atributo ISBN de la tabla `NL_Books`

check_edades

Métrica	
Nombre	check_edades
Descripción	Controla si la edad es válida.
Granularidad	Celda
Dominio del Resultado	{0, 1}

Método	
Descripción	Implementa la métrica check_edades verificando si la edad tiene formato correcto y es un valor razonable.
Tipos de datos de entrada	String
Tipos de datos de salida	Boolean
Algoritmo	`IsNumeric(edad) AND edad >0 AND edad <100`

Método aplicado	
Tipo	Medición
Descripción	Verifica que tenga un valor numérico entre 1 y 99.
Aplicado a	Atributo Age de la tabla `NL_Users`

check_price

Métrica	
Nombre	check_price
Descripción	Controla si el precio es válido.
Granularidad	Celda
Dominio del Resultado	{0, 1}

Método	
Descripción	Implementa la métrica check_price verificando si el valor tiene formato correcto.
Tipos de datos de entrada	String
Tipos de datos de salida	Boolean
Algoritmo	`IsNumeric(price) AND price >= 0`

Método aplicado	
Tipo	Medición
Descripción	Verifica que tenga un valor numérico positivo con una consulta SQL.
Aplicado a	Atributo Price de la tabla `NL_Books`

duplicate_ratio

Métrica	
Nombre	duplicate_ratio
Descripción	Da el porcentaje de valores duplicados.
Granularidad	Columna
Dominio del Resultado	[0...1]

Método	
Descripción	Implementa la métrica duplicate_ratio sobre un atributo dado.
Tipos de datos de entrada	String
Tipos de datos de salida	Float

Algoritmo	`def duplicate_ratio(data, column):`
	`  total = len(data)`
	`  duplicated = data.duplicated(subset=[column]).sum()`
	`  return duplicated / total`

Método aplicado	
Tipo	Medición
Descripción	Dados los datos y el atributo, calcula el porcentaje de valores duplicados en ese atributo.
Aplicado a	Atributos ISBN, AuthorID, PublisherID, User_ID de la tabla `NL_Books`. Atributo ID de la tabla `NL_Users`.

check_RN1

Métrica	
Nombre	check_RN1
Descripción	Da el porcentaje de entradas de la tabla que tienen al menos un campo vacío entre los atributos `isbn`, `titulo`, `autor` y `editor`.
Granularidad	Conjunto de columnas
Dominio del Resultado	[0...1]

Método	
Descripción	Implementa la métrica check_RN1 sobre una tabla dada.
Tipos de datos de entrada	String
Tipos de datos de salida	Float

Algoritmo	`def Check_RN1(datos):`
	`  # Seleccionar solo las columnas relevantes (isbn, titulo, autor, editor)`
	`  datos_relevantes = datos[['isbn', 'titulo', 'autor', 'editor']]`
	`  # Verificar cuántas filas tienen al menos un campo NULL entre estas columnas`
	`  incompletos = datos_relevantes.isnull().any(axis=1).sum()`
	`  # Calcular la proporción de filas incompletas`
	`  return incompletos / len(datos)`

Método aplicado	
Tipo	Medición
Descripción	Dada una tabla, calcula el porcentaje de entradas que tienen al menos un campo vacío entre los atributos `isbn`, `titulo`, `autor` y `editor`.
Aplicado a	Tabla `NL_Books`.

contar_nulls

Métrica	
Nombre	contar_nulls
Descripción	Calcula el porcentaje de entradas vacías en una columna.
Granularidad	Columna
Dominio del Resultado	[0...1]

Método	
Descripción	Implementa la métrica contar_nulls.
Tipos de datos de entrada	String
Tipos de datos de salida	Float

Algoritmo	`def porcentaje_nulos(atributo):`
	`  nulos = atributo.isnull()`
	`  contador_nulos = nulos.sum()`
	`  porcentaje = contador_nulos / len(atributo)`
	`  return porcentaje`

Método aplicado	
Tipo	Medición
Descripción	Dado un atributo de una tabla, calcula el porcentaje de entradas vacías.
Aplicado a	Cualquier atributo de cualquier tabla.

concistencia_ratings

Métrica	
Nombre	concistencia_ratings
Descripción	Verifica que la cantidad de ratings entre las distintas tablas sea coherente.
Granularidad	Conjunto de columnas
Dominio del Resultado	{0, 1}

Método	
Descripción	Implementa la métrica concistencia_ratings contando la cantidad de ocurrencias de un ISBN válido en la tabla NL_ratings y lo compara con lo declarado en el atributo rating_count de NL_Books.
Tipos de datos de entrada	String
Tipos de datos de salida	Boolean

Algoritmo	`def funcion_consistencia_ratings(libros, valid_isbn, ratings):`
	`  libros_validos = libros[valid_isbn==True]`
	`  for libro in libros_validos:`
	`  isbn = libro['isbn']`
	`  cantidad = (ratings['isbn'] == isbn).sum()`
	`  contados[isbn] = cantidad`
	`  rating_esperado = libros_validos['rating_counts']`
	`  return rating_esperado == contados`

Método aplicado	
Tipo	Agregación
Descripción	Dada una tabla de reviews, una tabla de libros (ambas con el campo isbn) y una lista que indiquen si son válidos o no, determina si hay congruencia entre lo registrado en ambas tablas sobre los ratings.
Aplicado a	Conjunto de atributos (NL_ratings.ISBN, NL_books.ISBN, NL_books.Rating_Count)

consistencia_fechas

Métrica	
Nombre	consistencia_fechas
Descripción	Verifica que la fecha de un rating sea posterior a la fecha de publicación del libro.
Granularidad	Conjunto de columnas
Dominio del Resultado	{0, 1}

Método	
Descripción	Implementa la métrica concistencia_fechas comparando las fechas de review_time de la tabla NL_reviews y la de PublisherDate de NL_Books.
Tipos de datos de entrada	String
Tipos de datos de salida	Boolean

Algoritmo	`def consistencia_fechas(NL_reviews, NL_books):`
	`  result = []`
	`  for i in range(len(NL_reviews)):`
	`  isbn = NL_reviews['isbn'][i]`
	`  fila_libro = NL_books[NL_books['isbn'] == isbn]`
	`  review_time, PublishedDate = fila_libro[['review_time', 'PublishedDate']].values[0]`
	`  result.append(review_time >= PublishedDate)`
	`  return result`

Método aplicado	
Tipo	Medición
Descripción	Dada una tabla de reviews y una tabla de libros (ambas con el campo isbn), determina si la fecha de la review es posterior a la publicación del libro.
Aplicado a	Conjunto de atributos $\mathrm{NL}_{R}eviews.review_{t}ime, NL_{B}ooks.PublisherDate$

missing_rating_books

Métrica	
Nombre	missing_rating_books
Descripción	Para ratings sobre libros, controla la existencia de estos en la base de datos.
Granularidad	Celda
Dominio del Resultado	{0, 1}

Método	
Descripción	Implementa la métrica check_rating_books para ver si los libros indicados en los ratings existen.
Tipos de datos de entrada	String
Tipos de datos de salida	Boolean
Algoritmo	`SELECT COUNT(*) FROM NL_Ratings WHERE NOT EXISTS(SELECT * FROM NL_Books WHERE NL_Ratings.ISBN = NL_Books.ISBN)`

Método aplicado	
Tipo	Medición
Descripción	Verifica que el libro en NL_Ratings exista en NL_Books
Aplicado a	Atributos ISBN en las tablas NL_Books y NL_Ratings

date_format

Métrica	
Nombre	date_format
Descripción	Controla el formato de fecha de todas las columnas tipo fecha
Granularidad	Celda
Dominio del Resultado	{0, 1}

Método	
Descripción	Implementa la métrica date_format para controlar que el formato de fecha sea correcto
Tipos de datos de entrada	String
Tipos de datos de salida	Boolean
Algoritmo	`TRY_CONVERT(DATE,date,103)`

Método aplicado	
Tipo	Medición
Descripción	Verifica que el campo PublisherDate en Books y el campo review_time en Ratings respeten el formato correcto de fecha dd/mm/yyyy
Aplicado a	Atributo PublisherDate en la tabla NL_Books y atributo review_time en la tabla NL_Ratings

duplicated_authors

MÉTRICA	
Nombre	duplicated_authors
Descripción	Indica cuántas tuplas nombran más de un autor
Granularidad	Columna
Dominio del Resultado	[0...1]

MÉTODO	
Descripción	Implementa la métrica duplicated_authors para obtener el porcentaje de tuplas que indican más de un autor
Tipos de datos de entrada	String
Tipos de datos de salida	float

Algoritmo	`DEFINE @tuplasDuplicadas = SELECT COUNT(*) FROM NL_Authors WHERE AuthorID LIKE '%,%'`
	`DEFINE @tuplasTotales = SELECT COUNT(*) FROM NL_Authors`
	`SELECT @tuplasDuplicadas / @tuplasTotales`

MÉTODO APLICADO	
Tipo	Medición
Descripción	Divide las tuplas que indican más de un autor sobre las tuplas totales
Aplicado a	Atributo AuthorID en la tabla NL_Books

3.1.4. Modelo de calidad de datos contextual

DQ Problems	DQ Dimension	DQ Factor	DQ Metric	DQ method	Applied DQ method
	ID: D4_Unicidad **Name:** Unicidad **Description:** Evalúa si los registros en un conjunto de datos son únicos y no presentan duplicaciones no deseadas **Suggested by** = {RN2, RN5}	**ID:** **Name:** No duplicación **Description:** Porcentaje de datos que no están duplicados en forma exacta **Represents** = {RN2, RN5}	**ID:** M_duplicate_ratio **Name:** duplicate_ratio **Description:** Da el porcentaje de valores duplicados. **Influenced by** = {RN2, RN5} **Granularity:** Columna **Result domain** = [0...1]	**ID:** metodo_duplicate_ratio **Name:** met_duplicate_ratio **Description:** Implementa la metrica duplicate_ratio sobre un atributo dado. **Uses** = {RN2, RN5} **Input data types:** String **Output data types:** Float **Algorithm:** "def duplicate_ratio(data, column): total = len(data) duplicated = data.duplicated(subset=[column]).sum() return duplicated / total"	**ID:** MA_duplicate_ratio **Type:** Medición **Descriptinon:** Dados los datos y el atributo, calcula el porcentaje de valores duplicados en ese atributo. **AppliedTo:** Atributos ⟪ISBN⟫, ⟪AuthorID⟫, ⟪PublisherID⟫ de la tabla NL_Books. Atributos ⟪ID⟫ de la tabla NL_USERS.

DQ Problems	DQ Dimension	DQ Factor	DQ Metric	DQ method	Applied DQ method
P11	**ID:** D2_Completitud **Name:** Completitud **Description:** Indica si el sistema de información contiene toda la información de interés. **Suggested by** = {RN1, RN2, RN5, RQ3, RQ7, F2}	**ID:** F2_Dens **Name:** Densidad **Description:** Indica cuanta información se tiene y cuanta falta sobre las entidades del sistema de información. **Represents** = {RN1, RN2, RN5, RQ3, RQ7, F2}	**ID:** M_contar_nulls **Name:** contar_nulls **Description:** Calcula el porcentaje de entradas vacias en una columna **Influenced by** = {RN1, RN5, RQ3, RQ7, F2} **Granularity:** Columna **Result domain** = [0...1]	**ID:** metodo_contar_nulls **Name:** met_contar_nulls **Description:** Implementa la metrica contar_nulls. **Uses** = {RN1, RN5, RQ3, RQ7, F2} **Input data types:** String **Output data types:** Float **Algorithm:** "def porcentaje_nulos(atributo): nulos = atributo.isnull() contador_nulos = nulos.sum() porcentaje = contador_nulos / len(atributo) return porcentaje"	**ID:** MA_contar_nulls **Type:** Medición **Descriptinon:** Dado un atributo de una tabla, calcula el porcentaje de entradas vacias. **AppliedTo:** Cualquier atributo de cualquier tabla.

			ID: M_check_RN1
			Name: check_RN1
			Description: Da el porcentaje de entradas de la tabla que tienen al menos un campo vacio entre los atributos isbn, titulo, autor y editor.
			Influenced by = {RN1, RN2, RQ3, RQ7, F2}
			Granularity: Conjunto de columnas
			Result domain = [0...1]
				ID: metodo_check_RN1
				Name: met_check_RN1
				Description: Implementa la metrica Check_RN1 sobre una tabla dada.
				Uses = {RN1, RN2, RQ3, RQ7, F2}
				Input data types: String
				Output data types: Float
				Algorithm:
				"def Check_RN1(datos):
				# Seleccionar solo las columnas relevantes (isbn, titulo, autor, editor)
				datos_relevantes = datos[['isbn', 'titulo', 'autor', 'editor']]
				# Verificar cuántas filas tienen al menos un campo NULL entre estas columnas
				incompletos = datos_relevantes.isnull().any(axis=1).sum()
				# Calcular la proporción de filas incompletas
				return incompletos / len(datos)"
					ID: MA_check_RN1
					Type: Medición
					Descriptinon: Dada una tabla, calcula el porcentaje de entradas que tienen al menos un campo vacio entre los atributos isbn, titulo, autor y editor.
					AppliedTo:
					Tabla NL_Books.

DQ Problems	DQ Dimension	DQ Factor	DQ Metric	DQ method	Applied DQ method
	ID: D3_Consistencia **Name:** Consistencia **Description:** Captura la satisfacción de reglas semánticas definidas sobre los datos. **Suggested by** = {RQ4, RQ6, RQ8, RQ11, F1, F3}	**ID:** **Name:** Integridad inter-relación **Description:** Captura la satisfacción de reglas entre atributos de tablas distintas. **Represents** = {RQ4, RQ6, RQ8, RQ11, F1, F3}	**ID:** M_consistencia_ratings **Name:** concistencia_ratings **Description:** Verifica que la cantidad de ratings entre las distintas tablas sea coherente. **Influenced by** = {RQ8} **Granularity:** conjunto de columnas **Result domain** = {0, 1}	**ID:** metodo_consistencia_ratings **Name:** met_consistencia_ratings **Description:** Implementa la metrica concistencia_ratings contando la cantidad de ocurrencias de un ISBN valido en la tabla NL_ratings y lo compara con lo declarado en el atributo rating_count de NL_Books. **Uses** = {RQ8} **Input data types:** String **Output data types:** Boolean **Algorithm:** "def función consistencia_ratings(libros, valid_isbn, ratings): libros_validos = libros[valid_isbn==True] for libro in libros_validos: isbn = libro['isbn'] cantidad = (ratings['isbn'] == isbn).sum() contados[isbn] = cantidad rating_esperado = libros_validos['rating_counts'] return rating_esperado==contados"	**ID:** MA_consistencia_ratings **Type:** Agregación **Descriptinon:** Dada una tabla de reviews, una tabla de libros (ambas con el campo isbn) y una lista que indiquen si son validos o no, determina si hay congruencia entre lo registrado en ambas tablas sobre los ratings. **AppliedTo:** Conjunto de atributos (NL_ratings.ISBN, NL_books.ISBN, NL_books.Rating_Count)
			ID: M_consistencia_fechas **Name:** concistencia_fechas **Description:** Verifica que la fecha de un rating sea posterior a la fecha de publicación del libro. **Influenced by** = {RQ6, RQ11, F1} **Granularity:** Conjunto de columnas **Result domain** = {0, 1}	**ID:** metodo_consistencia_fechas **Name:** met_consistencia_fechas **Description:** Implementa la metrica concistencia_fechas comparando las fechas de review_time de la tabla NL_reviews y la de PublisherDate de NL_Books. **Uses** = {RQ6, RQ11, F1} **Input data types:** String **Output data types:** Boolean **Algorithm:** "def consistencia_fechas(NL_reviews, NL_books): result = lista vacia for (i ; i< len(NL_reviews); i++): isbn = NL_reviews['isbn'][i] fila_libro = fila en NL_books donde NL_books['isbn']==isbn obtener los review_time y PublishedDate de las filas correspondientes result[i] = review_time >= PublishedDate: return resut"	**ID:** MA_consistencia_fechas **Type:** Medición **Descriptinon:** Dada una tabla de reviewsy una tabla de libros (ambas con el campo isbn), determina si la fecha de la review es posterior a la publicación del libro. **AppliedTo:** Conjunto de atributos ⟪NL_Reviews.review_time, NL_Books.PublisherDate⟫
			ID: M_missing_rating_books **Name:** missing_rating_books **Description:** Para ratings sobre libros, controla la existenia de estos en la base de datos. **Influenced by** = {RQ4, RQ8, F3} **Granularity:** Celda **Result domain** = {0, 1}	**ID:** metodo_missing_rating_books **Name:** met_missing_rating_books **Description:** Implementa la metrica check_rating_books para ver si los libros indicados en los ratings existen. **Uses** = {RQ4, RQ8, F3} **Input data types:** String **Output data types:** Boolean **Algorithm:** "SELECT COUNT(*) FROM NL_Ratings WHERE NOT EXISTS(SELECT * FROM NL_Books WHERE NL_Ratings.ISBN = NL_Books.ISBN)"	**ID:** MA_missing_rating_books **Type:** Medición **Descriptinon:** Verifica que el libro en NL_Ratings exista en NL_Books mediante una consulta SQL. **AppliedTo:** Conjunto de atributos ⟪ISBN⟫ en las tablas NL_Books y NL_Ratings

DQ Problems	DQ Dimension	DQ Factor	DQ Metric	DQ method	Applied DQ method

P1, P2, P5, P6, P8, P14.	**ID:** D1_Exactitud
	Name: Exactitud
	Description: Concierne a la correctitud y la precisión con que los datos del mundo real son representados en un sistema de información.
	Suggested by = {RN1, RN2, RN4, RQ2, RQ3, RQ6, RQ11, F1}
		ID: F1_ExactSint
		Name: Exactitud sintáctica
		Description: Indica que tan libre de errores sintácticos están los datos.
		Represents = {RN1, RN2, RQ3, RN4, RQ2}
			ID: M_check_ISBN
			Name: check_ISBN
			Description: Controla si el valor es un ISBN valido.
			Influenced by = {RN1, RN2, RQ3}
			Granularity: Celda
			Result domain = {0, 1}
				ID: metodo_check_ISBN
				Name: met_check_ISBN
				Description: Implementa la métrica Check_ISBN teniendo en cuenta la estructura de un código ISBN.
				Uses = {RN1, RN2, RQ3}
				Input data types: String
				Output data types: Boolean
				Algorithm:
				"def Check_ISBN(codigo):
				size = len(codigo)
				
				# Chequea para el caso de un ISBN10
				if (size==10):
				verifica que los primeros 9 caracteres sean digitos
				verifica que el ultimo caracter sea un digito o una "X"
				verifica que el digito verificador sea correcto
				
				# Chequea para el caso de un ISBN10
				if (size==13):
				verifica que todos los caracteres sean digitos
				verifica que el digito verificador sea correcto"
					ID: MA_check_ISBN
					Type: Medición
					Descriptinon: Utiliza el algoritmo de calculo de ISBN para verificar la validez del dato.
					AppliedTo:
					Atributo ⟪ISBN⟫ de la tabla NL_Books

			ID: M_check_edades **Name:** check_edades **Description:** Controla si la edad es valida. **Influenced by** = {RN4, RQ2} **Granularity:** Celda **Result domain** = {0, 1}	**ID:** metodo_check_edades **Name:** met_check_edades **Description:** Implementa la metrica Check_edades verificando si la edad tiene formato correcto y es un valor razonable. **Uses** = {RN4, RQ2} **Input data types:** String **Output data types:** Boolean **Algorithm:** IsNumeric(edad) AND edad > 0 AND edad < 100	**ID:** MA_check_edades **Type:** Medición **Descriptinon:** Verifica que tenga un valor numerico entre 1 y 99 **AppliedTo:** Atributo ⟪Age⟫ de la tabla NL_Users
			ID: M_check_price **Name:** check_price **Description:** Controla si la precio es valido. **Influenced by** = {RN3} **Granularity**: Celda **Result domain** = {0, 1}	**ID:** metodo_check_price **Name:** met_check_price **Description:** Implementa la metrica check_price verificando si el valor tiene formato correcto **Uses** = {RN3} **Input data types:** String **Output data types:** Boolean **Algorithm:** IsNumeric(price) AND price >= 0	**ID:** MA_check_price **Type:** Medición **Descriptinon:** Verifica que tenga un valor numerico positivo con una consulta SQL **AppliedTo:** Atributo ⟪Price⟫ de la tabla NL_Books
			ID: M_duplicated_authors **Name:** duplicated_authors **Description:** Indica cuantas tuplas nombran mas de 1 autor. **Influenced by** = {RN1, RQ3} **Granularity**: Columna **Result domain** = [0..1]	**ID:** metodo_duplicated_authors **Name:** met_duplicated_authors **Description:** Implementa la metrica duplicated_author para obtener el porcentaje de tuplas que indican mas de 1 autor **Uses** = {RN1, RQ3} **Input data types:** String **Output data types:** Double **Algorithm:** DEFINE @tuplasDuplicadas = SELECT COUNT(*) FROM NL_Authors WHERE AuthorID LIKE '%,%' DEFINE @tuplasTotales = SELECT COUNT(*) FROM NL_Authors SELECT @tuplasDuplicadas / @tuplasTotales	**ID:** MA_duplicated_authors **Type:** Medición **Descriptinon:** Divide las tuplas que indican mas de 1 autor sobre las tuplas totales **AppliedTo:** Atributo ⟪AuthorID⟫ en la tabla NL_Books

DQ Problems	DQ Dimension	DQ Factor	DQ Metric	DQ method	Applied DQ method
P1, P2, P5, P6, P8, P14.	**ID:** D1_Exactitud **Name:** Exactitud **Description:** Concierne a la correctitud y la precisión con que los datos del mundo real son representados en un sistema de información. **Suggested by** = {RN1, RN2, RN4, RQ2, RQ3, RQ6, RQ11, F1}	**ID:** F3_Precision **Name:** Exactitud precisión **Description:** Indica que tan detallados son los datos. **Represents** = {RQ6, RQ11, F1}	**ID:** M_date_format **Name:** date_format **Description:** Controla el formato de fecha de todas las columnas tipo fecha. **Influenced by** = {RQ6, RQ11, F1} **Granularity**: Celda **Result domain** = {0, 1}	**ID:** metodo_date_format **Name:** met_date_format **Description:** Implementa la metrica date_format para controlar que el formato de fecha sea correcto **Uses** = {RQ6, RQ11, F1} **Input data types:** String **Output data types:** Boolean **Algorithm:** TRY_CONVERT(DATE,date,103)	**ID:** MA_date_format **Type:** Medición **Descriptinon:** Verifica que el campo PublisherDate en Books y el campo review_time en Ratings respeten el formato correcto de fecha dd/mm/yyyy **AppliedTo:** Atributo ⟪PublisherDate⟫ en la tabla NL_Books y atributo ⟪review_time⟫ en al tabla NL_Ratings

3.2. ST5: DQ Measurement

3.2.1. Entradas y salidas

Entradas y salidas	
Entradas	**Salidas**
Reporte de problemas de CD priorizados (3.1.2)	Especificación de la BD de metadatos de CD (3.2.2)
Modelo de CD contextual (3.1.4)	Reporte de medición de la CD (3.2.3)
	Modelo de contexto (3.2.4)

CD: Calidad de datos.
BD: Base de datos.

3.2.2. Diseño de la base de datos de metadatos de calidad

La base de datos de metadatos se diseñó siguiendo el MER de la Figura 3.1, pensando en que todos los tipos de métricas implementadas y futuras métricas posibles pudieran ser almacenadas en ella.

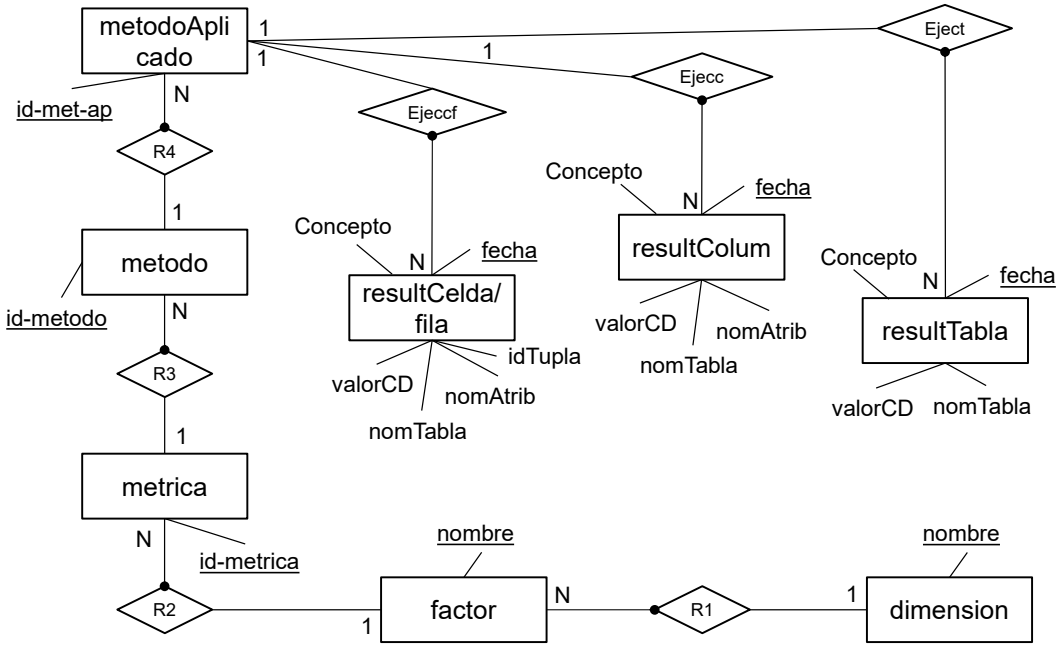

Figura 3.1: MER de la base de datos creada para almacenar los metadatos de calidad

3.2.3. Implementación y ejecución de los métodos propuestos

Los métodos propuestos fueron implementados directamente en SQL y sus resultados se ejecutaron y almacenaron en la base de datos de metadatos de calidad.

A continuación, se presenta un resumen de los resultados obtenidos en las tablas siguientes. Para destacar de forma más significativa los resultados de los métodos con granularidad a nivel de celda y fila, se muestran los porcentajes de resultados positivos, dado que dichos resultados son valores binarios.

Resultados para métodos con granularidad columna y tabla			
NombreTabla	**Atributo**	**ID Método Aplicado**	**% Valor=1**
NL_BOOKS	PublisherDate	MA_date_format	72.29
NL_RATINGS	review_time	MA_date_format	72.29
NL_BOOKS	ISBN	MA_missing_rating_books	87.38
NL_USERS	Age	MA_check_edades	5.13
NL_BOOKS	Price	MA_check_price	9.86

Resultados para métodos con granularidad columna y tabla			
Tabla	**Columna**	**Valor**	**ID Método Aplicado**
NL_BOOKS	ISBN	0.04	MA_duplicate_ratio
NL_BOOKS	AuthorID	0.53	MA_duplicate_ratio
NL_BOOKS	PublisherID	0.94	MA_duplicate_ratio
NL_USERS	ID	0.61	MA_duplicate_ratio
NL_USERS	ID	0.17	MA_contar_nulls
NL_BOOKS	AuthorID	0.08	MA_duplicated_authors
NL_BOOKS	-	0.16	MA_check_RN1

3.2.4. Contexto

Nuevos componentes de contexto

Se presenta a continuación la lista completa de componentes de contexto. En este caso, se agrega como nueva componente de contexto la base de datos diseñada junto a los metadatos de calidad de datos cargados en ella.

Componentes de Contexto	
Dominio	D: Libros.
Fuentes de datos	Datos obtenidos de ambas librerías que se fusionarán y los proporcionados por el cliente sobre sus realidades.
Tipos de usuario	U1: Administrador. U2: Publicista digital. U3: Analista de datos.
Tareas	T1: Gestión. T2: Análisis. T3: Consulta.
Reglas de negocio	RN1: Cada libro deberá tener asociado un ISBN, un título, al menos un autor y un editor. RN2: El atributo ISBN en NL_Books debe ser único a cada libro. RN3: El atributo Price en NL_Books debe ser un real positivo. RN4: El atributo Age en NL_Users debe ser un entero positivo. RN5: El atributo ID en NL_Users debe ser único y no vacío.
Requerimientos de calidad	RQ1: Frescura de datos: la base debe actualizarse todos los viernes. RQ2: Al menos el 80 % de los usuarios que califican los libros deben ser mayores de 18 años. RQ3: Al menos el 95 % de los libros deben cumplir simultáneamente con los siguientes requisitos: contar con un ISBN, tener el título correctamente escrito y que el nombre del autor incluya al menos un nombre y un apellido. RQ4: Al menos el 60 % de los libros tengan al menos un score mayor o igual a 5. RQ5: La librería pretende tener al menos 500 libros y poseer al menos el 20 % de la lista de los 100 mejores libros de Goodreads. RQ6: Los libros deben contar con fecha de publicación. RQ7: Los libros deben tener editorial. RQ8: Los libros deben tener asignado un valor de score. RQ9: los nombres de las editoriales deben estar estandarizados. RQ10: Las reglas de formato para nombres (autores, libros, editoriales) son: primera letra del nombre propio en mayúsculas y sin punto al final. RQ11: El formato para las fechas será dd/mm/aaaa.
Requerimientos del sistema	RS1: Los tiempos de respuesta del sitio Web de la NL no pueden superar los 3 segundos.
Necesidades de filtrado	F1: Libros por fecha (en particular, del año actual). F2: Libros por editorial. F3: Top de libros según su score.
Metadatos de calidad	BDQ: Base de datos de metadatos de calidad de datos.

Contexto

Contexto				
Componente de contexto	**Todos los usuarios**	**U1: Administrador**	**U2: Publicista**	**U3: Analista**
Dominio	D			
Tareas	T3	T1	T2	
Reglas de negocio	RN1, RN2, RN3, RN4, RN5			
Requerimientos de sistema	RS1			
Requerimientos de calidad de datos	RQ5, RQ9, RQ10, RQ11	RQ6, RQ7, RQ8	RQ1, RQ2, RQ4	RQ3
Necesidades de filtrado		F1, F2, F3		
Metadatos				
Metadatos de calidad de datos	BDQ			
Otros datos				

3.3. ST6: DQ Assessment

3.3.1. Entradas y salidas

Entradas y salidas	
Entradas	**Salidas**
Especificación de la BD de metadatos de CD (3.2.2)	Reporte de evaluación de CD (3.2.3)
Reporte de medición de la CD (3.2.3)	
Modelo de contexto (3.2.4)	

CD: Calidad de datos.
BD: Base de datos.

3.3.2. Definición de umbrales de evaluación

Al momento de definir los umbrales para las distintas medidas de calidad, se consideraron las componentes contextuales, especialmente los requerimientos de los usuarios y las reglas de negocio, procurando al mismo tiempo mantener coherencia con el propósito específico de cada métrica.

Bajo esta línea, se establecieron tres criterios distintos, los cuales se presentan en las Tablas 3.1, 3.2 y 3.3. La Tabla 3.4 detalla qué criterio fue aplicado en cada caso, según el método considerado.

Los métodos que producen resultados binarios fueron evaluados con el criterio *binario*. Para los métodos con resultados graduados, se distinguieron dos casos: aquellos sugeridos por una regla de negocio o requerimiento con umbral ya definido, y los que no. Para los del primer caso, se optó por utilizar el criterio *estándar estricto*, mientras que para los del segundo se utilizó el estándar, siguiendo la escala directa o inversa según la naturaleza de dichos métodos.

Tabla 3.1: Criterio estándar		
Concepto	**Rango (directo)**	**Rango (inverso)**
Malo	[0, 0.30)	[0.70, 1)
Bueno	[0.30, 0.60)	[0.40, 0.70)
Muy bueno	[0.60, 0.90)	[0.10, 0.40)
Excelente	[0.90, 1]	[0, 0.10)

Tabla 3.2: Criterio estándar estricto		
Concepto	**Directo**	**Inverso**
Deficiente	(0, 0.30]	[0.70, 1]
Malo	(0.30, 0.60]	[0.40, 0.70)
Aceptable	(0.60, 0.75]	[0.25, 0.40)
Bueno	(0.75, 0.90]	[0.10, 0.25)
Excelente	(0.90, 1]	[0, 0.10)

Tabla 3.3: Criterio binario	
Concepto	**Valor**
Deficiente	0
Excelente	1

Tabla 3.4: Criterios asociados a cada método	
Método	**Criterio**
metodo_check_ISBN	Binario
metodo_check_PublisherID	Binario
metodo_check_edades	Binario
metodo_duplicated_authors	Estándar (inverso)
metodo_date_format	Binario
metodo_duplicate_ratio	Estándar estricto (inverso)
metodo_contar_nulls	Estándar estricto (inverso)
metodo_check_RN1	Estándar estricto (inverso)
metodo_consistencia_ratings	Binario
metodo_consistencia_fechas	Binario
metodo_missing_rating_books	Binario

3.3.3. Ejecución de los umbrales de evaluación

Dados los criterios elegidos en la Subsección 3.3.2, se aplicaron y almacenaron los umbrales mediante consultas SQL y los resultados de los mismos pueden apreciarse en las Figuras 3.2 y 3.3

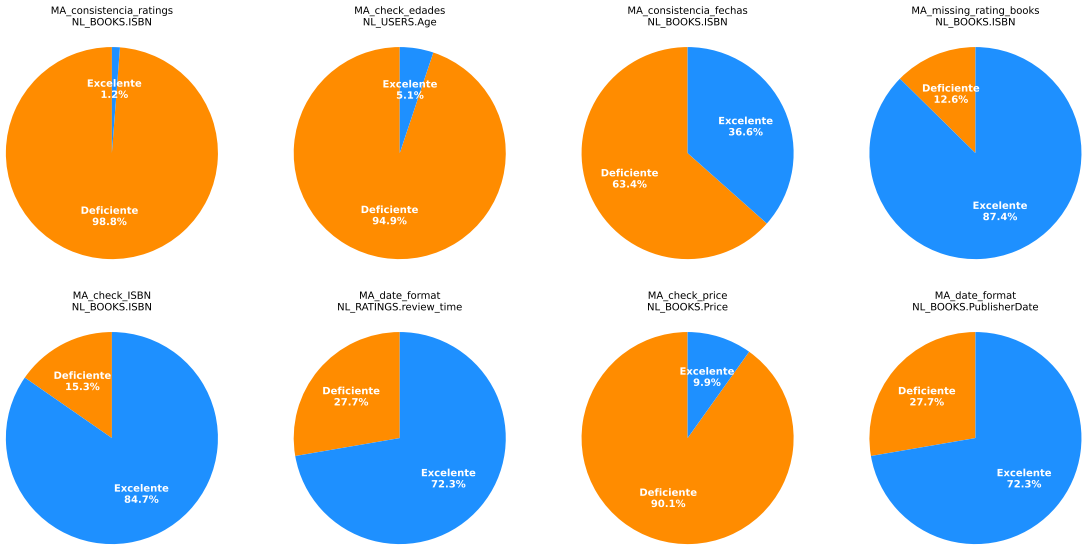

Figura 3.2: Gráficos de tortas con los resultados obtenidos para los métodos con granularidad celda y fila

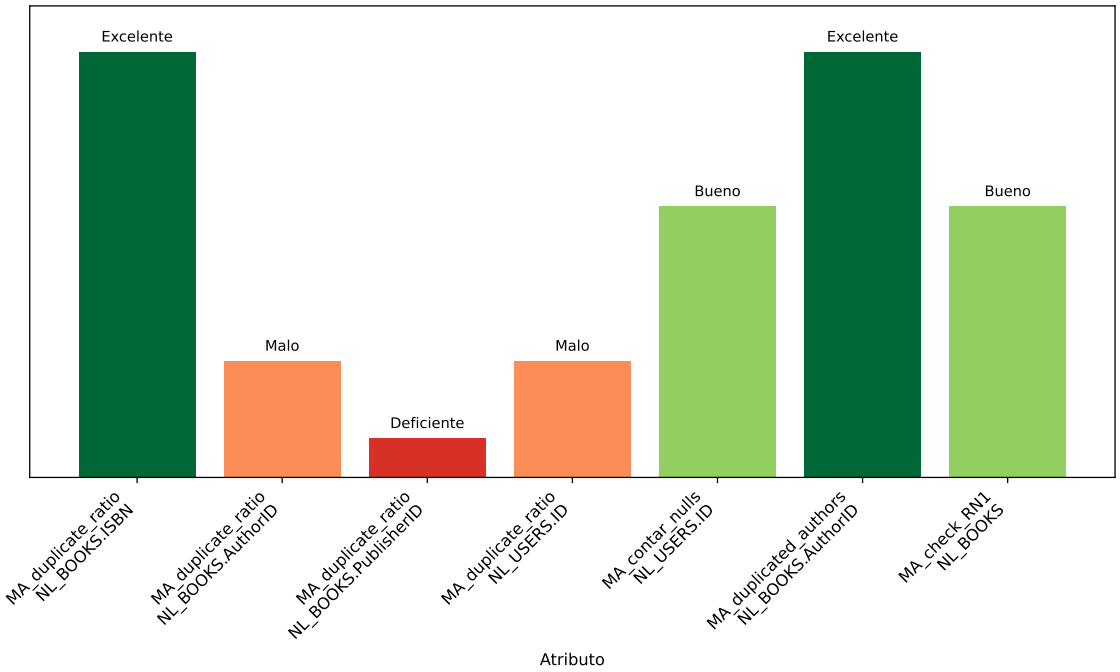

Figura 3.3: Gráficos de barras con los resultados obtenidos para los métodos con granularidad columna y tabla

A partir de estas gráficas, se puede ver que la calidad de los datos es muy dispar dependiendo de la métrica que se analice. En particular, se destaca que el método encargado de medir el desempeño de la regla de negocios RN1 dio un resultado favorable, mientras que hay un problema importante en lo que refiere a los ID de los usuarios.

En cuanto a los resultados de métodos con granularidad celda o fila, sorprende la gran cantidad de resultados negativos encontrados, posiblemente debido a la unión de dos bases de datos con atributos muy diferentes.

3.3.4. Contexto

Durante esta etapa no se identificaron nuevos componentes de contexto, sino que solo se modificaron algunos existentes (BDQ), por lo que el contexto completo sigue siendo el presentado en la sección 3.2.4.

3.4. Entradas y salidas de la Fase 2 completa

Como preparación para abordar la Fase 3 del modelo CaDQM, se presentan a continuación las entradas y salidas generales de la Fase 2.

Entradas y salidas	
Entradas	**Salidas**
Reporte del análisis de requerimientos de usuarios (2.3.2)	Reporte de problemas de CD priorizados (3.1.2)
Reporte del análisis de datos (2.2.2)	Modelo de CD (3.1.4)
Reporte de problemas de CD (Tabla 2.7)	Especificación de la BD de metadatos de CD (3.2.2)
Modelo de Contexto (2.3.3)	Reporte de medición de la CD (3.2.3)
	Modelo de contexto (3.2.4)
	Reporte de evaluación de CD (3.2.3)

CD: Calidad de datos.
BD: Base de datos.

3.5. Descripción del desarrollo del trabajo

Siguiendo la metodología CaQDM en la fase 2 el primer paso fue repasar los problemas de calidad analizados en la fase 1 y poder identificar potenciales nuevos problemas.

Una vez identificados, procedimos a asignar una prioridad a cada uno tomando en cuenta los requerimientos de calidad, las reglas de negocio, los requerimientos del sistema y las necesidades de filtrado del cliente.

Luego definimos la estructura de la base de datos que usaríamos para almacenar las distintas dimensiones, factores, métricas, métodos, métodos aplicados y ejecuciones de las distintas metricas.

Una vez pronta la estructura de la base, procedimos a insertar todos los datos y a crear distintos procedimientos almacenados donde se encuentra la lógica de distintos algoritmos de los métodos aplicados. Con la estructura pronta y los datos cargados solo restaba ejecutar los distintos métodos aplicados y almacenar los resultados.

Finalmente armamos el modelo de calidad con sus distintas métricas y métodos tomando en cuenta las prioridades de los problemas de calidad.

3.6. Conclusiones de la fase 2

La implementación de la segunda fase del modelo CaDQM en este proyecto permitió establecer un enfoque estructurado para diagnosticar el estado actual de la calidad de los datos, considerando los problemas detectados y la prioridad asignada a cada uno.

Mediante las distintas etapas logramos cuantificar la calidad de los datos mediante métricas especificas obtenidas analizando la realidad del problema. Luego aplicando dichas métricas y manejando distintos umbrales pudimos transformar estas métricas en distintas valoraciones de calidad importantes para ayudarnos a entender el estado de calidad de los datos.

Finalmente, logramos entender el origen de los posibles problemas de calidad que pueden presentar un conjunto de datos y, tomando en cuenta las distintas métricas métodos implementados junto a

los umbrales que elegimos, pudimos obtener procedimientos que nos ayudarán a entender estado de la calidad de los datos estudiados de una forma fácil de entender y comunicar.

Capítulo 4

Fase 3 - Data Quality Improvement

4.1. Entradas y salidas de la Fase 3 completa

Dado que la Fase 3 del modelo CaDQM no se ejecutará completamente y siguiendo cada una de sus etapas, sino que se hará de manera resumida, se presentan a continuación las entradas y salidas esperadas de cada una de las etapas de la fase:

Entradas y salidas de la ST7	
Entradas	**Salidas**
Reporte de evaluación de CD (3.2.3)	Reporte con los problemas de CD seleccionados y priorizados de acuerdo a sus causas (4.2)

Entradas y salidas de la ST8	
Entradas	**Salidas**
Reporte con los problemas de CD seleccionados y priorizados de acuerdo a sus causas (4.2)	Reporte de análisis de costos
	Plan de mejora de la CD (4.3)

Entradas y salidas de la ST9	
Entradas	**Salidas**
Reporte de análisis de costos	Reporte de ejecución del plan de mejora de CD
Plan de mejora de la CD (4.3)	Data at hand mejorados

CD: Calidad de datos.
BD: Base de datos.

4.2. Análisis de causas

Los datos disponibles en el *data at hand* provienen de dos bases de datos distintas, cada una con sus propios atributos, rangos de valores y formatos definidos. Estas diferencias provocan diversas inconsistencias en la base unificada, vinculadas a los problemas de calidad P1, P2, P8 y P10. Además, como consecuencia directa de la incompatibilidad entre los atributos, se genera una gran cantidad de valores nulos, lo que da lugar al problema P11. Estos errores pueden considerarse de prioridad media, ya que, si bien no impiden directamente el uso de los datos, sí afectan su integridad y consistencia general.

Por otro lado, ambas tablas ya presentaban errores antes de su unificación, producto de un diseño deficiente y de la aparente falta de restricciones sobre los datos ingresados. Esta causa está asociada a los problemas P6, P9, P12, P13, P14 y P15. Dado que estas fallas impactan directamente en la estructura y confiabilidad del sistema, se consideran de prioridad alta.

Finalmente, una fuente adicional de errores es la presencia de errores de tipeo, derivados del ingreso manual de datos. Estos se reflejan en los problemas P3, P4, P5 y P7, y se consideran de prioridad baja.

4.3. Plan de mejora

Siguiendo el principio de que toda reingeniería o modificación de procesos debe orientarse a corregir y prevenir errores, procurando mantener la mayor calidad de datos posible a futuro, la mejora principal propuesta consiste en una reestructuración de la base de datos. La nueva estructura se presenta en la Figura 4.1.

Esta nueva base fue diseñada con el objetivo de normalizar formatos y nombres, evitar redundancias y reducir al mínimo el ingreso manual de datos por parte del usuario, con el fin de mitigar errores de tipeo.

Además, se incorporan restricciones de integridad y de dominio que permiten controlar la validez de los datos ingresados, mejorando la consistencia general del sistema.

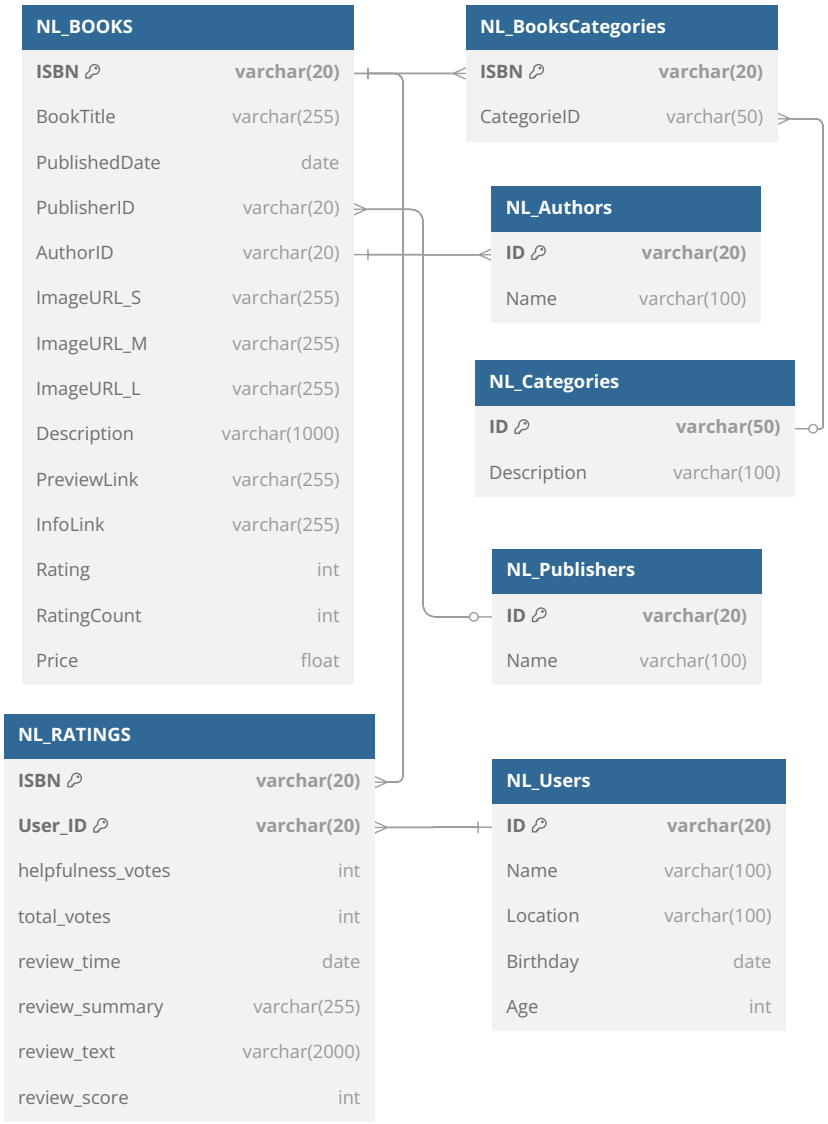

Figura 4.1: Estructura de la nueva base de datos propuesta

Para reducir aún más el ingreso manual, se propone el uso de diccionarios o listas de selección para campos como ciudades, géneros, autores y editoriales, permitiendo la introducción manual únicamente en los casos en que los valores no se encuentren previamente cargados.

Respecto a los datos ya existentes que presentan errores o formatos diferentes al estándar (por ejemplo, nombres en minúscula), se recomienda implementar funciones automáticas de corrección

y estandarización. Para aquellos registros con valores nulos o errores que no puedan corregirse automáticamente, será necesaria una revisión y corrección manual.

Finalmente, se sugiere capacitar al personal encargado del ingreso y mantenimiento de los datos, a fin de garantizar el uso correcto del sistema. Asimismo, se recomienda establecer controles periódicos de calidad de datos, que permitan monitorizar el estado general de la base y tomar acciones correctivas cuando sea necesario.

4.4. Descripción del desarrollo del trabajo propuesto

Si bien no se implementó toda la Fase 3, se entiende que su desarrollo podría haberse llevado a cabo en conjunto con el cliente. A continuación se propone un posible enfoque para su implementación:

- **Validación del plan de mejora:** confirmar que las acciones propuestas están alineadas con los requerimientos de calidad definidos previamente y con los intereses del cliente.
- **Estimación de recursos:** calcular tiempo y costo necesarios para llevar adelante el plan de mejora evaluando su viabilidad.
- **Alineación con el cliente:** mantener una comunicación con el cliente para asegurar que el plan validado responda sus necesidades.
- **Aplicación de mejoras :** ejecutar los distintos algoritmos propuestos (3.2.3). Para cada uno se evaluara el impacto sobre la calidad de los datos utilizando las métricas definidas en la Fase 2 como mecanismo de comparación y validación.

4.5. Conclusiones de la fase 3

La fase 3 es el cierre del ciclo propuesto por CaDQM y si bien no se implementó de forma completa pudimos entender la importancia del mismo y sobre todo el porque de muchos elementos propuestos en fases previas.

En este caso entendemos que es una etapa en la que es crucial la comunicación fluida con el cliente (si bien todas las etapas, en cierta forma, lo son) ya que es donde se cierra el proyecto y donde se dejan ver los resultados y quizá esto complica su implementación.

Nos hubiera gustado poder atacar algunos de los puntos mencionados en esta fase (como la estimación o la aplicación de las soluciones propuestas) aunque logramos dejar en claro la forma en que lo haríamos.

Capítulo 5

Conclusiones finales

Este proyecto nos permitió implementar las distintas fases de CaDQM, desde la planificación inicial hasta la propuesta de acciones de mejora (de forma parcial).

Durante el proceso aplicamos las distintas técnicas y herramientas propuestas para identificar, medir y evaluar problemas de calidad de datos sobre una base integrada proveniente de dos fuentes distintas, simulando un caso real.

Cada una de las fases nos permitió entender distintos problemas sobre la calidad de los datos y como cada etapa se alimentaba de la anterior pudimos ver como es importante tener una base firme para llegar a las etapas mas avanzadas con un proyecto solido (y en concordancia con el cliente).

Si bien no se implemento de forma total la Fase 3, pudimos elaborar un plan de mejora concreto y proponer una estrategia para su desarrollo en conjunto con el cliente.

En resumen valoramos el poder aplicar una metodología completa sobre un caso de un cliente real, con muchas dimensiones y problemas a considerar así como distintas decisiones a tomar. Esta experiencia fue útil para poder bajar a tierra los distintos temas vistos en clase y ademas poder tener criterio propio a la hora de enfrentarnos con posibles problemas de calidad de datos en la vida profesional.

Finalmente el trabajo nos permitió comprender y aplicar el modelo CaDQM en profundidad, destacando tanto sus ventajas como los distintos problemas o desafíos que implica llevarlo a la práctica.

Como complemento al informe dejamos adjuntos una serie de scripts que incluyen la base de datos de metadatos de calidad y todos los algoritmos implementados para la medición y análisis realizados durante el proyecto.