CS-322 Introduction to database systems

Grand Comics Database Project

Team No : 40

Léon Delachaux	**296093**
Ralph El Haddad	**302360**
Tom MERY	**297217**

Spring 2022

Contents

List of Figures

1 Deliverable 1

The main purpose of this first report is to explain and motivate the choices that has been made while designing an ER model and a corresponding relational schema for the Grand Comics Database (see documentation at `docs.comics.org`). The studied dataset is available at `https://drive.switch.ch/index.php/s/I8fQKctZc6POjoc`.

1.1 Assumptions

Every assumptions that has been made on the data is motivated by an inspection on the given csv files using Pandas library. All these assumptions are already represented on the ER diagram (see 1) as they are only about participation and key constraints. But for clarity purpose a brief recap is made in the following subsections.

1.1.1 Story

The data contained in the csv file of Story was found to be very dirty thus some assumptions had to be made on the data. For example, it was observed that some stories do not have an issue id (this has been checked by counting the number of null values for this attribute). However, it can seem logical that a story without an issue can not physically exist. Also the number of story without issue id was clearly negligible and most likely due to dirty data (144 over 1869594 instances). Thus one can assume that a story must have at least one associated issue.

The same phenomenon has been noticed for the type id attribute: 166 over 1869594 instances instances of story table do not have any type id. So in the same way one can assume that every story must have at least one type id.

Finally, it is assumed that the issue in which the story was published should be unique.

1.1.2 Issue and Indicia Publisher

By inspecting the csv file for Issue table, it has been found that some issue does not have a value for the attribute indicia_publisher_id, which might also look like a problem, but it has been decided not to assume that it is correlated with dirty data.

However, one can notice that the field indicia_publisher_id only contains a number and if the indicia_publisher_id exists then it is unique.

1.1.3 Series and Issue

There are three different relationship sets involving Series and Issue entity sets.

The first thing is that every issue must be related to an entity from Series that is unique. One can check on the csv file that every issue has actually an existing attribute series_id of type int64.

The second one is that the series is supposed to have a first and a last issue. It might seem logical to expect that any series have a first issue but maybe not a last. However, some series do not have first and last issue. Actually, they correspond to series that does not have any issue.

1.1.4 Publisher, brand group and indicia publisher

Every brand group from the csv file has an existing publisher_id and therefore is owned by a master publisher. However, Brand group is not declared as a weak entity because one can for example imagine that a brand group can change from one publisher to another. In fact, it is logical to declare brand group as an independent entity set.

It also has been checked that every indicia publisher has a publisher_id, and because Indicia_publisher has a different role from publisher it was natural not to declare it as a weak entity.

Finally it is assumed that a brand group or an indicia publisher must have one master publisher.

1.2 Entity Relationship Model

1.2.1 Schema

Figure 1 shows the proposed ER diagram for the Grand Comics Database. In order to keep the ER diagram readable it has been chosen to show only the entities and the relationship first, the attributes of each entities are given in Figure 2. Section 1.2.2 describes the nomenclature used.

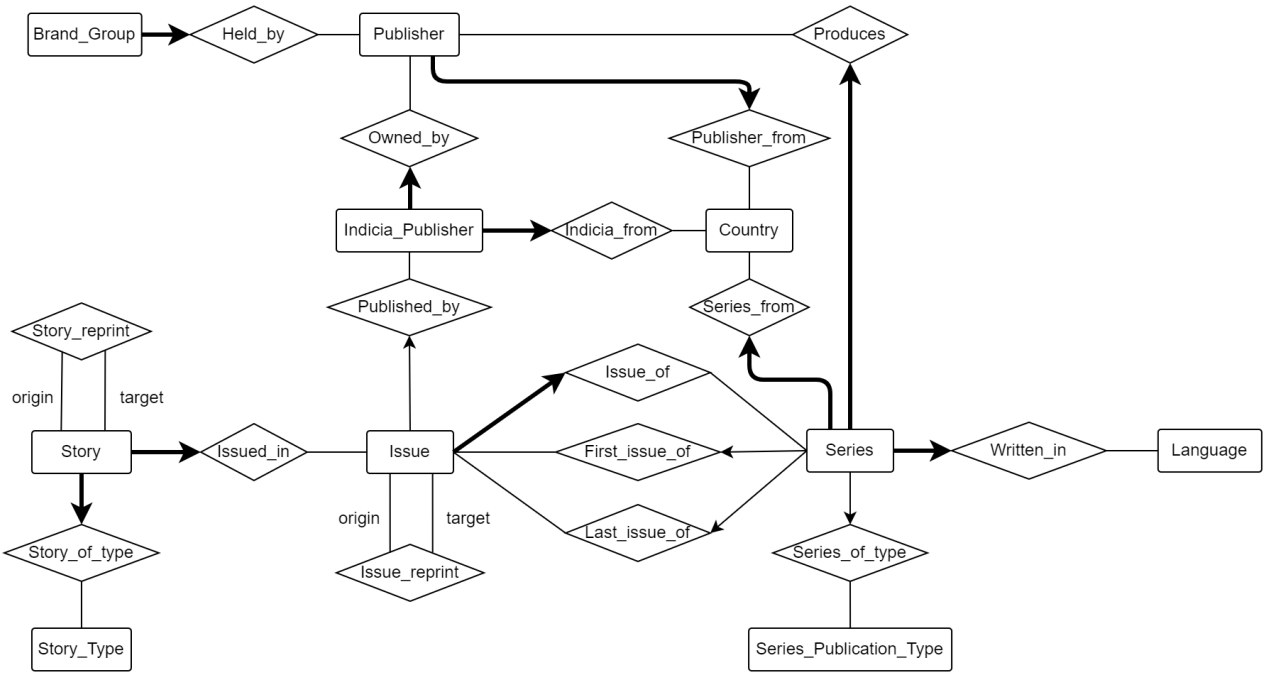

Figure 1: ER Diagram without attributes

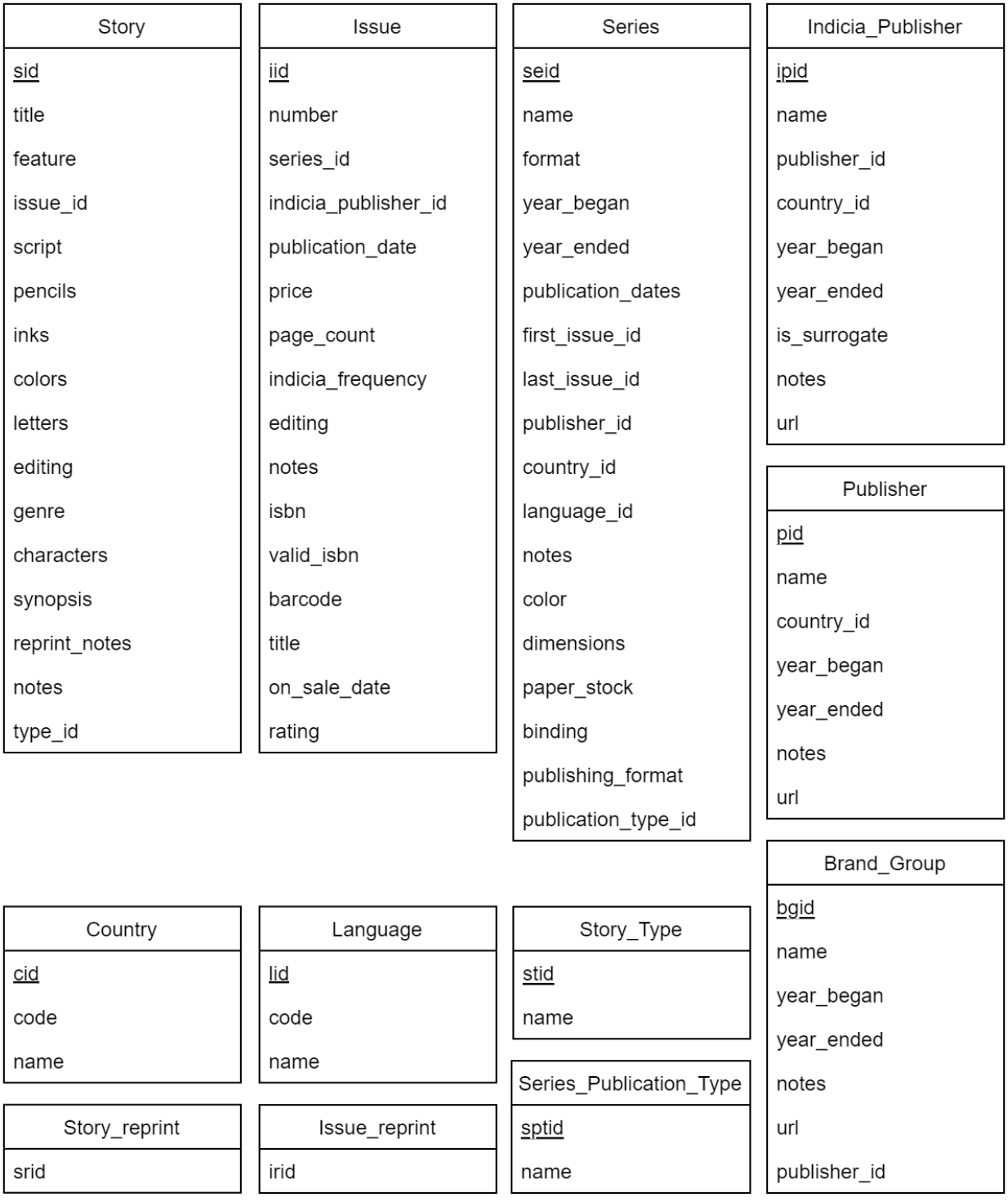

Figure 2: Entities and relationships with their attributes

1.2.2 Description

The nomenclature used to create the ER diagram is taken from [1] where key constraints and participation constraints are well defined in section 2.4 of this book. Here is the most useful information:

- Entity sets are represented by a rectangle shape.
- Relationship sets are represented by a diamond shape.
- Primary key are underlined in Figure 2.
- When two entities are involved in a relationship and that the first entity has a relation with at most one instance of the second entity, then the branch bound to the first entity is replaced with an arrow pointing toward the relationship.
- If the participation of the entity is total (it has a relation with exactly one instance of the second entity), the arrow is bold.

For *Story_reprint* and *Issue_reprint* it has be decided to keep the id as an attributes even though one could drop it (actually we were not sure if we could drop some unnecessary attributes from the given csv files so we kept it).

1.3 Relational Model

In order to avoid redundancy and to not overload the report the relational model is directly given in DDL where the resulting tables, keys, foreign keys, constraints, type, domain constraints and nullable values are explicitly defined. Here are the most useful informations about the proposed DDL :

- Domain constraints are expressed by defining a type for the attributes (built-in types compatible with Oracle).
- Relationships are defined using foreign key in the relevant entity sets as explained in the following paragraph. The name given to these kind of constraints corresponds to the name of the relationship sets in the ER diagram.
- Participation constraints are defined using NOT NULL clause. Primary key has to be NOT NULL as well.
- The size is not specified for INTEGER type as the default size is large enough for the GCD dataset.
- VARCHAR2 type as been preferred than CHAR type as it offers more flexibility by using dynamic allocation memory.

- When specified the size of a VARCHAR2 is expressed in bytes.
- When an attribute can contain an arbitrary long string the size is not specified.
- When the size is known to be small then the size is set by overestimating the max size found in the csv file for the given attributes (assuming that the strings is encoded in utf8).
- When the size is known to be limited but not small the most common length VARCHAR(255) is used since it's large enough and it's the maximum value encoded in 8 bits.
- There is no weak entity (instances of every entity sets can exist by itself) thus for foreign key constraint with total participation noting is specified as ON DELETE NO ACTION and ON UPDATE NO ACTION is already here by default.
- When a foreign key can be null (no total participation) ON DELETE SET NULL is specified.

It is important to mention that in the following relational model, except for *Story_reprint* and *Issue_reprint*, the relationship sets represented in the ER diagram are not translated in the relational model by a table. The relationships set are indeed translated by including the information about the relationship set in the table corresponding to the entity set with the key. For instance, instead of creating a table for the relationship *Held_by* with a foreign key *brand_group_id* from *Brand_group* entity set and foreign key *publisher_id* from *Publisher* entity set, knowing that every instance of *Brand_group* has at most one associated instance of *Publisher* one could simply add an attribute *publisher_id* as a foreign key to *Brand_group*.

This is possible since every relationship sets in the ER model involves at least one entity set which has a key constraint specifying that every instance of it is in relation with at most one instance of the other entity set. This approach has been preferred as none of the relationship set have specific attributes and also this approach avoids creating a distinct table for every relationship set. The only cons of this method is that some space could be wasted if for example several instances of *Brand_group* do not have a *publisher_id* as with this approach one has to store null value for these instances. In the proposed ER model most of the relationships include one entity set with total participation thus this drawback does not apply. For the relationship with partial participation it still has been chosen to use the same approach since it offers faster operations (joining table is time-consuming).

DDL

```
01 | CREATE TABLE 'Story' (
02 |     'sid' INTEGER NOT NULL,
03 |     'title' VARCHAR2(255),
04 |     'feature' VARCHAR2(255),
05 |     'issue_id' INTEGER NOT NULL,
06 |     'script' VARCHAR2,
07 |     'pencils' VARCHAR2(255),
08 |     'inks' VARCHAR2(255),
```

```
09 |     'colors' VARCHAR2(255),
10 |     'letters' VARCHAR2(255),
11 |     'editing' VARCHAR2(255),
12 |     'genre' VARCHAR2(255),
13 |     'characters' VARCHAR2,
14 |     'synopsis' VARCHAR2,
15 |     'reprint_notes' VARCHAR2(255),
16 |     'notes' VARCHAR2(255),
17 |     'type_id' INTEGER NOT NULL,
18 |     PRIMARY KEY ('sid'),
19 |     CONSTRAINT 'Issued_in'
20 |         FOREIGN KEY ('issue_id')
21 |         REFERENCES 'Issue' ('iid'),
22 |     CONSTRAINT 'Story_of_type'
23 |         FOREIGN KEY ('type_id')
24 |         REFERENCES 'Story_Type' ('stid')
25 | );
26 |
27 | CREATE TABLE 'Issue' (
28 |     'iid' INTEGER NOT NULL,
29 |     'number' VARCHAR2(50),
30 |     'series_id' INTEGER NOT NULL,
31 |     'indicia_publisher_id' INTEGER,
32 |     'publication_dates' VARCHAR2(255),
33 |     'price' VARCHAR2(50),
34 |     'page_count' INTEGER,
35 |     'indicia_frequency' VARCHAR2(50),
36 |     'editing' VARCHAR2(255),
37 |     'notes' VARCHAR2(255),
38 |     'isbn' VARCHAR2(50),
39 |     'valid_isbn' VARCHAR2(50),
40 |     'barcode' VARCHAR2(50),
41 |     'title' VARCHAR2(255),
42 |     'on_sale_date' VARCHAR2(255),
43 |     'rating' VARCHAR2(50),
44 |     PRIMARY KEY ('iid'),
45 |     UNIQUE ('isbn', 'valid_isbn', 'barcode'),
46 |     CONSTRAINT 'Issue_of'
47 |         FOREIGN KEY ('series_id')
48 |         REFERENCES 'Series' ('seid'),
49 |     CONSTRAINT 'Published_by'
50 |         FOREIGN KEY ('indicia_publisher_id')
51 |         REFERENCES 'Indicia_Publisher' ('ipid')
52 |         ON DELETE SET NULL
53 | );
54 |
55 | CREATE TABLE 'Series' (
56 |     'seid' INTEGER NOT NULL,
57 |     'name' VARCHAR2(255) NOT NULL,
58 |     'format' VARCHAR2(255),
59 |     'year_began' INTEGER,
60 |     'year_ended' INTEGER,
```

```
61 |      'publication_dates' VARCHAR2(255),
62 |      'first_issue_id' INTEGER,
63 |      'last_issue_id' INTEGER,
64 |      'publisher_id' INTEGER NOT NULL,
65 |      'country_id' INTEGER NOT NULL,
66 |      'language_id' INTEGER NOT NULL,
67 |      'notes' VARCHAR2(255),
68 |      'color' VARCHAR2(255),
69 |      'dimensions' VARCHAR2(255),
70 |      'paper_stock' VARCHAR2(255),
71 |      'binding' VARCHAR2(255),
72 |      'publishing_format' VARCHAR2(255),
73 |      'publication_type_id' INTEGER,
74 |      PRIAMRY KEY ('seid'),
75 |      CONSTRAINT 'First_issue_of'
76 |          FOREIGN KEY ('first_issue_id')
77 |          REFERENCES 'Issue' ('iid')
78 |          ON DELETE SET NULL,
79 |      CONSTRAINT 'Last_issue_of'
80 |          FOREIGN KEY ('last_issue_id')
81 |          REFERENCES 'Issue' ('iid')
82 |          ON DELETE SET NULL,
83 |      CONSTRAINT 'Produces'
84 |          FOREIGN KEY ('publisher_id')
85 |          REFERENCES 'Publisher' ('pid'),
86 |      CONSTRAINT 'Series_from'
87 |          FOREIGN KEY ('country_id')
88 |          REFERENCES 'Country' ('cid'),
89 |      CONSTRAINT 'Written_in'
90 |          FOREIGN KEY ('language_id')
91 |          REFERENCES 'Language' ('lid'),
92 |      CONSTRAINT 'Series_of_type'
93 |          FOREIGN KEY ('publication_type_id')
94 |          REFERENCES 'Series_Publication_Type' ('sptid')
95 |          ON DELETE SET NULL
96 |  );
97 |
98 |  CREATE TABLE 'Indicia_Publisher' (
99 |      'ipid' INTEGER NOT NULL,
100 |     'name' VARCHAR2(255) NOT NULL,
101 |     'publisher_id' INTEGER NOT NULL,
102 |     'country_id' INTEGER NOT NULL,
103 |     'year_began' INTEGER,
104 |     'year_ended' INTEGER,
105 |     'is_surrogate' INTEGER,
106 |     'notes' VARCHAR2(255),
107 |     'url' VARCHAR2(255),
108 |     PRIMARY KEY (ipid),
109 |     UNIQUE ('name'),
110 |     CHECK ((is_surrogate = 0) OR (is_surrogate = 1) OR (is_surrogate IS
        NULL)),
111 |     CONSTRAINT 'Owned_by'
```

```
112 |         FOREIGN KEY ('publisher_id')
113 |         REFERENCES 'Publisher' ('pid'),
114 |     CONSTRAINT 'Indicia_from'
115 |         FOREIGN KEY ('country_id')
116 |         REFERENCES 'Country' ('cid')
117 | );
118 |
119 | CREATE TABLE 'Publisher' (
120 |     'pid' INTEGER NOT NULL,
121 |     'name' VARCHAR2(255) NOT NULL,
122 |     'country_id' INTEGER NOT NULL,
123 |     'year_began' INTEGER,
124 |     'year_ended' INTEGER,
125 |     'notes' VARCHAR2(255),
126 |     'url' VARCHAR2(255),
127 |     PRIMARY KEY ('pid'),
128 |     UNIQUE ('name'),
129 |     CONSTRAINT 'Publisher_from'
130 |         FOREIGN KEY ('country_id')
131 |         REFERENCES 'Country' ('cid')
132 | );
133 |
134 | CREATE TABLE 'Brand_Group' (
135 |     'bgid' INTEGER NOT NULL,
136 |     'name' VARCHAR2(255) NOT NULL,
137 |     'year_began' INTEGER,
138 |     'year_ended' INTEGER,
139 |     'notes' VARCHAR2(255),
140 |     'url' VARCHAR2(255),
141 |     'publisher_id' INTEGER NOT NULL,
142 |     PRIMARY KEY ('bgid'),
143 |     UNIQUE ('name'),
144 |     CONSTRAINT 'Held_by'
145 |         FOREIGN KEY ('publisher_id')
146 |         REFERENCES 'Publisher' ('pid')
147 | );
148 |
149 | CREATE TABLE 'Story_Type' (
150 |     'stid' INTEGER NOT NULL,
151 |     'name' VARCHAR2(50),
152 |     PRIMARY KEY ('sptid'),
153 |     UNIQUE ('name')
154 | );
155 |
156 | CREATE TABLE 'Series_Publication_Type' (
157 |     'sptid' INTEGER NOT NULL,
158 |     'name' VARCHAR2(50) NOT NULL,
159 |     PRIMARY KEY ('sptid'),
160 |     UNIQUE ('name')
161 | );
162 |
163 | CREATE TABLE 'Language' (
```

```
164 |       'lid' INTEGER NOT NULL,
165 |       'code' VARCHAR2(4) NOT NULL,
166 |       'name' VARCHAR2(50) NOT NULL,
167 |       PRIMARY KEY ('lid'),
168 |       UNIQUE ('code')
169 |   );
170 |
171 |   CREATE TABLE 'Country' (
172 |       'cid' INTEGER NOT NULL,
173 |       'code' VARCHAR2(4) NOT NULL,
174 |       'name' VARCHAR2(50) NOT NULL,
175 |       PRIMARY KEY ('cid'),
176 |       UNIQUE ('code')
177 |   );
178 |
179 |   CREATE TABLE 'Story_reprint' (
180 |       'srid' INTEGER NOT NULL,
181 |       'origin_story_id' INTEGER NOT NULL,
182 |       'target_story_id' INTEGER NOT NULL,
183 |       PRIMARY KEY ('origin_story_id', 'target_story_id'),
184 |       FOREIGN KEY 'origin_story_id' REFERENCES 'Story' ('sid'),
185 |       FOREIGN KEY 'target_story_id' REFERENCES 'Story' ('sid')
186 |   );
187 |
188 |   CREATE TABLE 'Issue_reprint' (
189 |       'irid' INTEGER NOT NULL,
190 |       'origin_issue_id' INTEGER NOT NULL,
191 |       'target_issue_id' INTEGER NOT NULL,
192 |       PRIMARY KEY ('origin_issue_id', 'target_issue_id'),
193 |       FOREIGN KEY 'origin_issue_id' REFERENCES 'Issue' ('iid'),
194 |       FOREIGN KEY 'target_issue_id' REFERENCES 'Issue' ('iid')
195 |   );
```

1.3.1 Remarks

We thought about adding some constraints such that, for example, an indicia publisher or a brand group can not have the same url as their master publisher but did not find a way to do so.

1.4 Data Cleaning and Transformation Discussion

As we discussed in the first part, we found out that the csv file for Story was very dirty. We already suggested that every story should have an issue and a type, but still there are a lot of cleaning to do . For example, some rows have merged and as a consequence some titles have more than 1000 characters. For example, this is a title:

" Hi! Jack! Think fast! Catch!,Where Do We Stand?,237756,?,Lloyd Ostendorf (signed),Lloyd Ostendorf (signed),?,?,,drama; religious,,,,,19 1673616,Friendly Enemy...,,237756,?,Charles Raab

(signed),Charles Raab (signed),?,?,„adventure„„„,19 1673617,Crossword Puzzle„237756,?,Bill Berry ?,Bill Berry ?„?„„„„On inside front cover.,1 1673618„,,237756„?,?,?„,,adventure„„„,6 1673619,Announcing 3 Separate Editions of Our Little Messenger„237755,?,?,?„typeset„„„„On back cover.,16 1673620,Man wanted" "

It would be good to remove the lines that have merged and clean the data before putting it in the file again. We can replace these erroneous titles with a default title: "Story number " + id.

Moreover, we discussed the fact that some issues did not have an indicia publisher. We did not decide to declare that an issue must have an indicia publisher but it could actually be a good thing.

We can also assume that the first and last issue ids might not be accurate. To clean the first and last issue ids, we group the issues by series id in the series-issues relationship table, then we sort the issues by publication date and raise that the first issue id for each series should be equal to the series' first_issue_id. If there is more than one issue for the series, we raise that the last issue id should be equal to the last_issue_id. Then we can replace the erroneous ids with the correct ones.

1.5 General Comments

All group members have contributed equally to this first delivrable.

2 Deliverable

From this part the ER model and the corresponding relational schema used are the ones provided in appendix (see A.1). The relational schema of the database has been implemented using the provided DDL in appendix A.2.

2.1 Assumptions

For the data loading, we assumed that the strings in the tables were encoded in UTF-8.

2.2 Data loading and cleaning

Before loading data, the constraints are disabled (see appendix A.4). The data have been loaded using SQL LOADER and Oracle Client. When type errors have been encountered (for example, that the number should have 4 digits but has more), these data have been set manually as a null value (for example, year value larger than 9999). The constraints are finally re-enabled after loading using appendix A.3.

2.3 Query Implementation

2.3.1 Query 1

Description

Through the first query we aim at selecting the brand group names located in the United States. Then we show them in a table by descending order using the command "order by count(*) desc", which means we start with brand group names that are most recurrent in the database.

SQL Implementation

```
01 | SELECT brand.name
02 | FROM GCD_BRAND_GROUP brand,
03 |     GCD_PUBLISHER publi,
04 |     GCD_INDICIA_PUBLISHER indicia,
05 |     STDDATA_COUNTRY country
06 | WHERE brand.publisher_id = publi.id
07 |     AND indicia.publisher_id = publi.id
08 |     AND indicia.country_id = country.id
09 |     AND country.name = 'United States'
10 | GROUP BY brand.name
11 | ORDER BY COUNT(*) DESC;
```

Results

Brand Group Name
Marvel
Disney Comics
Malibu
Shadowline
Legendary
DBPRO
Marvel Knights
A Romance! Magazine
Nelson
Crossgen
2099
Ultimate
Timely Comics
Heavy Hitters
Actual Ender's Game
A Humorama Magazine
M-Tech
Pizza Hut
Spider-Man Group
Curtis Magazines

2.3.2 Query 2

Description

The second query selects IDs and names of publishers that have published albums in Italian. The Publisher names are then sorted by descending Publisher ID, and both are shown in the table.

SQL Implementation

```
01 | SELECT id, name
02 | FROM GCD_PUBLISHER publi
03 | WHERE id IN (
04 |     SELECT ser.publisher_id
05 |     FROM GCD_SERIES ser,
06 |         GCD_SERIES_PUBLICATION_TYPE typ,
07 |         STDDATA_LANGUAGE lang
08 |     WHERE SER.PUBLICATION_TYPE_ID = typ.id
09 |         AND typ.name = 'album'
10 |         AND ser.language_id = lang.id
11 |         AND lang.name = 'Italian'
```

```
12 |  )
13 |  ORDER BY id DESC;
```

Results

Publisher ID	Publisher Name
10963	Cliquot
10355	Andrea Leggeri
10349	RW Edizioni
10209	7even Age Entertainment
10056	Bel-Ami Edizioni
4881	001 Edizioni
3166	Rizzoli
2693	Panini
1177	Bonelli-Dargaud
969	Kappa Edizioni
845	Edizioni BD
442	EPC
440	Edizioni LIsola Trovata
397	Milano Libri Edizioni
324	Max Bunker Press
164	Sergio Bonelli Editore

2.3.3 Query 3

Description

In the third query, the goal is to select the name of series that have published books in Switzerland. The Series names are then shown in the table in alphabetical order using the command "order by".

SQL Implementation

```
01 |  SELECT ser.name
02 |  FROM GCD_SERIES ser,
03 |      GCD_SERIES_PUBLICATION_TYPE typ,
04 |      STDDATA_COUNTRY country
05 |  WHERE ser.publication_type_id = typ.id
06 |      AND typ.name = 'book'
07 |      AND ser.country_id = country.id
08 |      AND country.name = 'Switzerland'
09 |  ORDER BY ser.name;
```

Results

Series Name
120 Rue de la Gare
25 images de la passion d'un homme
Affentheater
Alack Sinner
Alans Kindheit
Alans Krieg
Das Geheimnis des Würgers
Den Letzten beissen die Hunde
Der Fotograf
Die Reportage
Ein Leben in China
Elender Krieg - Gesamtausgabe
Family Living
Gaza
Golem im Emmental
Jetzt kommt später
Le Soleil
Palästina
Victor Levallois
Voyages et aventures surprenantes de Robinson Crusoé

2.3.4 Query 4

Description

In the fourth query, we count the number of series that started to get published in each year between 1990 and 2017. The number of series that began in each year are then shown in the table, with the years of beginning sorted in descending order.

SQL Implementation

```
01 | SELECT COUNT(*) AS n_series, year_began
02 | FROM GCD_SERIES
03 | WHERE year_began >= 1990 AND year_began <= 2017
04 | GROUP BY year_began
05 | ORDER BY year_began DESC;
```

Results

Number of Series	Year Began
172	2017
3185	2016
3421	2015
3401	2014
3390	2013
3217	2012
3473	2011
3519	2010
3180	2009
3128	2008
3039	2007
2895	2006
2771	2005
2414	2004
2330	2003
2118	2002
2002	2001
2033	2000
1995	1999
2070	1998

2.3.5 Query 5

Description

For the fifth query, we select the country names of non-Swiss publishers that have published series in Switzerland. The country names are then shown in a table with no particular order.

SQL Implementation

```
01 | SELECT name
02 | FROM STDDATA_COUNTRY
03 | WHERE name != 'Switzerland' AND id IN (
04 |     SELECT ser.country_id
05 |     FROM GCD_SERIES ser, GCD_PUBLISHER publi
06 |     WHERE ser.publisher_id = publi.id AND publi.id IN (
07 |         SELECT publisher_id
08 |         FROM GCD_SERIES ser, STDDATA_COUNTRY country
09 |         WHERE ser.country_id = country.id AND country.name = 'Switzerland
       '
10 |     )
```

```
11 | );
```

Results

Country Name
Netherlands
Germany
France

2.3.6 Query 6

Description

In the sixth query we aim at selecting language names of series published outside of Switzerland but whose publisher is located in Switzerland. To do so we first extract all country names except Switzerland, and distinct language names thus avoiding redundancy. In a subquery, we select language IDs of series that were published outside the publisher's country, specifically in Switzerland. The results are shown in the table.

SQL Implementation

```
01 | SELECT DISTINCT lang.name
02 | FROM STDDATA_COUNTRY country, STDDATA_LANGUAGE lang
03 | WHERE country.name != 'Switzerland' AND lang.id IN (
04 |     SELECT ser.language_id
05 |     FROM GCD_SERIES ser, GCD_PUBLISHER publi, STDDATA_COUNTRY cou
06 |     WHERE ser.country_id != cou.id
07 |         AND cou.name= 'Switzerland'
08 |         AND ser.publisher_id = publi.id
09 |         AND publi.country_id = cou.id
10 | );
```

Results

Language Name
German
French
Dutch

2.3.7 Query 7

Description

Through the seventh query we select the names of publishers located in the Netherlands and that began publishing before 1995 and ended after 2000. The results are shown in reversed alphabetical order in the table.

SQL Implementation

```
01 | SELECT publi.name
02 | FROM GCD_PUBLISHER publi, STDDATA_COUNTRY country
03 | WHERE publi.year_began < 1995
04 |     AND publi.year_ended > 2000
05 |     AND  publi.country_id = country.id
06 |     AND country.name = 'Netherlands'
07 | ORDER BY publi.name DESC;
```

Results

Publisher Name
Wolters-Noordhoff
Wavery Productions
Van Holkema & Warendorf
Unie van Waterschappen
Uitgeverij M
Stivoro
Stichting Propria Cures
Nieuwsblad van het Noorden
Mondria
Maaike Hartjes
Drukwerk
De Vrijbuiter
De Stripper
De Banier
DRoodkoopren knoop
Bzztôh
Bee Dee

2.3.8 Query 8

Description

In the eighth query we select the names of publishers that have published series outside of the country they are located in. We select distinct names to avoid redundancy in case a publisher has published in many foreign countries. The results are shown in the table.

SQL Implementation

```
01 | SELECT DISTINCT publi.name
02 | FROM GCD_PUBLISHER publi, GCD_SERIES ser
03 | WHERE publi.id = ser.publisher_id AND publi.country_id != ser.country_id
04 | ORDER BY publi.name;
```

Results

Publisher Name
AK Press
APComics
Abiogenesis Press
Acme Press
Agência Portuguesa de Revistas
Aircel Publishing
Alpen Publishers
Amigo
Andina
Apocalypse
Associated Newspapers
Atomeka Press
Azeko
BSV - Williams
Beeld Beeld
Beta Publications
Bich
Bloomsbury
British Petroleum (BP)
Burnside

2.3.9 Query 9

Description

In ninth query we select countries where series have been published by foreign-based publishers, and we count how many series have been published in each of them. The results are shown in the table, in decreasing order of number of series published by foreign-based publishers.

SQL Implementation

```
01 | SELECT country.name, COUNT(*) AS n_occur
02 | FROM GCD_PUBLISHER publi, GCD_SERIES ser, STDDATA_COUNTRY country
03 | WHERE publi.id = ser.publisher_id
04 |     AND publi.country_id != ser.country_id
05 |     AND ser.country_id = country.id
06 | GROUP BY country.name
07 | ORDER BY n_occur DESC;
```

Results

Country Name	Number
Germany	184
United States	107
France	54
Netherlands	40
German Democratic Republic [former]	30
Canada	27
United Kingdom	26
Belgium	20
Italy	14
Portugal	11
Sweden	9
Argentina	5
Mexico	5
(unknown)	4
Hong Kong	3
South Africa	3
Yugoslavia	2
Colombia	2
Austria	1
Chile	1

2.3.10 Query 10

Description

In the tenth query we count the number of publishers that have only ever published series outside their country of location. To do so we select and count publishers that have never published series in their home country, and have published series in foreign countries. The results are then shown in the table.

SQL Implementation

```
01 | SELECT COUNT(*)
02 | FROM GCD_PUBLISHER publi
03 | WHERE NOT EXISTS (
04 |     SELECT id
05 |     FROM GCD_SERIES ser
06 |     WHERE ser.country_id = publi.country_id
07 |         AND ser.publisher_id = publi.id
08 | ) AND EXISTS (
09 |     SELECT id
10 |     FROM GCD_SERIES ser
11 |     WHERE ser.country_id != publi.country_id
12 |         AND ser.publisher_id = publi.id
13 | );
```

Results

Number
29

3 Deliverable 3

3.1 Query Implementation

3.1.1 Query 1

Description

Query 1 starts by selecting publisher names. In a sub-query, we count the number of distinct series languages among the first ten publishers that began publishing the earliest. That explains the sub-query where publisher IDs are selected among publishers ordered by ascending order of "year began", and where "rownum" <= 10. The publisher names are then shown in the table along with their corresponding series language count.

SQL Implementation

```
01 | SELECT P.NAME AS Publisher_Name, Series_Language_Count
02 | FROM GCD_PUBLISHER P, (
03 |     SELECT PUBLISHER_ID, COUNT(DISTINCT LANGUAGE_ID) AS
       Series_Language_Count
04 |     FROM (
05 |         SELECT PUBLISHER_ID, LANGUAGE_ID FROM GCD_SERIES
06 |         WHERE PUBLISHER_ID IN (
07 |             SELECT ID
08 |             FROM (SELECT ID FROM GCD_PUBLISHER ORDER BY YEAR_BEGAN)
09 |             WHERE ROWNUM <= 10
10 |         )
11 |     )
12 |     GROUP BY PUBLISHER_ID
13 | )
14 | WHERE P.ID=PUBLISHER_ID;
```

Results

Publisher Name	Series Language Count
Turner	1
Wilhelm Prym Werke	1
Verlag C. H. Beck	1
Editorial Ibis Lda. / Livraria Bertrand S.A.R.L.	1
Schlütersche Verlagsgesellschaft	2
Livraria Bertrand Lda.	1
Exshaw	1
Sdu Uitgevers	1
Schwabe	1
Humphrey	1

3.1.2 Query 2

Description

Query 2 selects publisher names among publishers that have a series count of more than 500. The count is made in the sub-query in the "from" clause: we count distinct publisher IDs in series to get the series count per publisher. The count is distinct to avoid redundancy in case a series was published several times by the same publisher. The publisher names are then shown in the table in decreasing order of their respective series count.

SQL Implementation

```
01 | SELECT P.NAME AS Publisher_Name, Series_Count
02 | FROM GCD_PUBLISHER P, (
03 |     SELECT PUBLISHER_ID, COUNT(DISTINCT ID) AS Series_Count
04 |     FROM GCD_SERIES
05 |     GROUP BY PUBLISHER_ID
06 | )
07 | WHERE Series_Count>500 AND P.ID=PUBLISHER_ID
08 | ORDER BY Series_Count DESC;
```

Results

Publisher Name	Series Count
Marvel	7628
DC	6993
Dark Horse	2741
Image	2080
Panini Deutschland	1811
Fantagraphics	1385
Egmont Ehapa	1364
IDW	1344
Carlsen Comics [DE]	1282
Hjemmet / Egmont	947
Planeta DeAgostini	801
Panini España	728
Viz	703
Tokyopop (de)	669
Western	653
Boom! Studios	584
Dynamite Entertainment	579
Malibu	572
Dupuis	562
Arboris	561

3.1.3 Query 3

Description

Query 3 selects brand group names that count at least 100 indicia publishers. To count the indicia publishers, a sub-query is embedded to the 'from' clause. In the sub-query we select and count distinct indicia publisher IDs that match the brand group's publisher IDs. The results are shown in the table.

SQL Implementation

```
01 | SELECT NAME AS Brand_Group_Name , Count_ID
02 | FROM GCD_BRAND_GROUP BG, (
03 |     SELECT BG.ID, COUNT(DISTINCT IP.ID) AS Count_ID
04 |     FROM GCD_BRAND_GROUP BG, GCD_INDICIA_PUBLISHER IP
05 |     WHERE BG.PUBLISHER_ID=IP.PUBLISHER_ID
06 |     GROUP BY BG.ID
07 | ) ID_COUNT
08 | WHERE BG.ID=ID_COUNT.ID AND Count_ID>100;
```

Results

Brand Group Name	Count ID
Atlas	110
2099	110
Spider-Man Group	110
A Humorama Magazine	110
A Lovers Magazine	110
A Romance! Magazine	110
Actual Ender's Game	110
Epic	110
Shadowline	110
Heavy Hitters	110
Flying Fanny The	110
Crossgen	110
Disney Comics	110
Marvel	110
Pumping Iron	110
Nelson	110
Pizza Hut	110
Marvel Knights	110
DBPRO	110
New Universe	110

3.1.4 Query 4

Description

Query 4 selects brand group names with the largest number of Belgian indicia publisher. We first select brand groups among those that have at least indicia publisher in Belgium, and count how many belgian IPs they have. We achieve that through the sub-query in the "from" clause. Then we only keep the brand group names with the maximum number of belgian indicia publishers through another sub-query, in the "where" clause. The results are shown in the table.

SQL Implementation

```
01 | SELECT BG.NAME AS Brand_Group_Name, Count_Belgian_IP AS
       Largest_Number_of_Belgian_Indicia_Publishers
02 | FROM GCD_BRAND_GROUP BG, (
03 |     SELECT BG.ID AS BG_ID, COUNT(*) AS Count_Belgian_IP
04 |     FROM GCD_INDICIA_PUBLISHER IP, STDDATA_COUNTRY C, GCD_BRAND_GROUP BG
05 |     WHERE IP.COUNTRY_ID=C.ID
06 |         AND C.NAME='Belgium'
07 |         AND BG.PUBLISHER_ID=IP.PUBLISHER_ID
```

```
08 |      GROUP BY BG.ID
09 |  )
10 |  WHERE BG.ID=BG_ID AND Count_Belgian_IP=(SELECT MAX(Count_Belgian_IP) FROM
       (
11 |      SELECT BG.ID AS BG_ID, COUNT(*) AS Count_Belgian_IP
12 |      FROM GCD_INDICIA_PUBLISHER IP, STDDATA_COUNTRY C, GCD_BRAND_GROUP BG
13 |      WHERE IP.COUNTRY_ID=C.ID
14 |          AND C.NAME='Belgium'
15 |          AND BG.PUBLISHER_ID=IP.PUBLISHER_ID
16 |      GROUP BY BG.ID
17 |      )
18 |  );
```

Results

Brand Group Name	Largest Number of Belgian Indicia Publishers
Dupuis	11
Éditions Dupuis	11
Collectie Vrolijke Vlucht	11
RepéRages Dupuis	11
Dubbel Expresso Dupuis	11
Dupuis; Pizza Hut	11
Vrije Vlucht	11
Spotlight	11
Uitgeverij Dupuis	11
Aire Libre	11
Puceron	11
Ukje	11
Expresso Dupuis	11
Graton	11
Mezzanine	11

3.1.5 Query 5

Description

Query 5 selects Indicia Publishers (IP) that have published at least 400 single-issue series. First we select IP names among those that have single-issue series, and count them, using a sub-query in the "from" clause. We identify single-issue series using "count" in a sub-sub-query in the "from" clause of the sub-query. Finally we only keep IPs that have published at least 400 single-issue series as shown in the query's "where" clause. The results are shown in descending order of single-issue series count in the table.

SQL Implementation

```
SELECT IP.NAME AS Indicia_Publisher_Name, count2
FROM GCD_INDICIA_PUBLISHER IP ,(
    SELECT IP.ID, COUNT(*) count2
    FROM GCD_INDICIA_PUBLISHER IP, (
        SELECT INDICIA_PUBLISHER_ID, SERIES_ID, COUNT(*) AS count1
        FROM GCD_ISSUE
        GROUP BY INDICIA_PUBLISHER_ID, SERIES_ID
    )
    WHERE count1 = 1 AND IP.ID = INDICIA_PUBLISHER_ID
    GROUP BY IP.ID
) Single_Issue
WHERE count2>=400 AND Single_Issue.ID=IP.ID
ORDER BY Count2 DESC;
```

Results

Indicia Publisher Name	Count
DC Comics	2759
Marvel Worldwide Inc.	1040
Marvel Comics	783
Marvel Publishing Inc.	603
Image Comics Inc.	479
Dark Horse Comics Inc.	419

3.1.6 Query 6

Description

Query 6 displays the top 5 issues that have the most reprinted stories, with their story reprint count. We selected issue IDs and their matching story IDs using the sub-query in the 'from' clause. The sub-query is used to count distinct issues for the same story, and to group these issues by their common story ID. The sub-query also orders the count by decreasing order. Then the 'where' clause selects the top 5 most reprinted stories using 'rownum <= 5' and the results are shown in the table.

SQL Implementation

```
SELECT I.ID AS Issue_ID, ORIGIN_ID AS Story_ID, Count_Reprint
FROM GCD_ISSUE I, (
    SELECT ORIGIN_ID, COUNT(DISTINCT TARGET_ID) AS Count_Reprint
    FROM GCD_STORY_REPRINT
    GROUP BY ORIGIN_ID
```

```
06 |      ORDER BY Count_Reprint DESC
07 |  ) Reprint_count
08 |  WHERE (ROWNUM <= 5) and (Reprint_count.ORIGIN_ID = I.ID);
```

Results

Issue ID	Story ID	Count
336676	1419829	90
17099	133435	52
17099	133434	43
92510	363431	36
555894	1804115	46

3.1.7 Query 7

Description

Query 7 shows heroes that are featured in stories of the three genres of humor, crime and romance. We start by selecting features among those that participating in the stories, using a sub-query in the 'from' clause. The sub-query selects features and genres where the feature participated, i.e where it "is not null". Then the query keeps the three genres humor, crime and romance and shows features that have participated in all three ('having count (distinct genre) = 3') in the table.

SQL Implementation

```
01 |  SELECT FEATURE
02 |  FROM (
03 |      SELECT S.FEATURE, SG.GENRE
04 |      FROM GCD_STORY S, GCD_STORY_TO_GENRE STG, GCD_STORY_GENRE SG
05 |      WHERE S.ID=STG.STORY_ID
06 |          AND STG.GENRE_ID=SG.ID
07 |          AND S.FEATURE IS NOT NULL
08 |  )
09 |  WHERE GENRE IN ('humor', 'crime', 'romance')
10 |  GROUP BY FEATURE
11 |  HAVING COUNT(DISTINCT GENRE) = 3;
```

Results

Feature
Spider-Man
Archie
?
Family Funnies
Harvey Comics
Wolverine
G.I. Joe
Excursions
Dick Tracy

3.1.8 Query 8

Description

Query 8 extracts the year of publication of the database's issues. We select distinct publication dates using a sub-query in the 'from' clause. In the sub-query, we extract the year from publication dates that have the 'DD/MM/YYYY' format, returning 'null' when the format does not match ('on conversion error'). The query then only keeps dates that are not null and we show the result in ascending order in the table.

SQL Implementation

```
01 | SELECT DISTINCT PUBLICATION_DATE AS Extracted_Year_Values
02 | FROM (
03 |     SELECT EXTRACT(
04 |         YEAR
05 |         FROM TO_DATE(PUBLICATION_DATE DEFAULT NULL ON CONVERSION ERROR,
06 |             'DD/MM/YYYY')
07 |     ) AS PUBLICATION_DATE
08 |     FROM GCD_ISSUE
09 | )
10 | WHERE PUBLICATION_DATE IS NOT NULL
11 | ORDER BY PUBLICATION_DATE ASC;
```

Results

Extracted Year Values
15
22
56
57
69
71
77
82
88
91
1825
1826
1830
1831
1832
1834
1835
1888
1890
1893

3.1.9 Query 9

Description

Query 9 counts the number of times each of the dat formats 'DD/MM/YYYY', 'MM/DD/YYYY', 'MONTH/YYYY' was used in the issue table. To count dates among each of the aforementioned date formats, we used three sub-queries in the query's 'from' clause, with another sub-sub-query in each sub-query's 'from' clause to select the wanted format in the same way as in query 8. The final counts are shown in the table.

SQL Implementation

```
01 | SELECT COUNT_1 AS COUNT_DD_MM_YYYY,
02 |     COUNT_2 AS COUNT_MM_DD_YYYY,
03 |     COUNT_3 AS COUNT_MONTH_YYYY
04 | FROM (
05 |     SELECT COUNT(PUBLICATION_DATE) AS COUNT_1
06 |     FROM (
07 |         SELECT TO_DATE(PUBLICATION_DATE DEFAULT NULL ON CONVERSION ERROR,
08 |             'DD/MM/YYYY')
```

```
09 |             AS PUBLICATION_DATE
10 |         FROM GCD_ISSUE
11 |     )
12 | ),
13 | (
14 |     SELECT COUNT(PUBLICATION_DATE) AS COUNT_2
15 |     FROM (
16 |         SELECT TO_DATE(PUBLICATION_DATE DEFAULT NULL ON CONVERSION ERROR,
17 |             'MM/DD/YYYY')
18 |             AS PUBLICATION_DATE
19 |         FROM GCD_ISSUE
20 |     )
21 | ),
22 | (
23 |     SELECT COUNT(PUBLICATION_DATE) AS COUNT_3
24 |     FROM (
25 |         SELECT TO_DATE(PUBLICATION_DATE DEFAULT NULL ON CONVERSION ERROR,
26 |             'MONTH YYYY')
27 |             AS PUBLICATION_DATE
28 |         FROM GCD_ISSUE
29 |     )
30 | );
```

Results

Count DD/MM/YYYY	Count MM/DD/YYYY	Count MONTH/YYYY
18851	3969	217642

3.1.10 Query 10

Description

Query 10 shows the total number of issues per year between 1965 and 1975, including both years. We counted among issues published in each year using a sub-query in the 'from' clause, where we extracted years among 'MONTH YYYY' date formats, then only kept dates between 1965 and 1975 included. The results are shown in the table in ascending year order.

SQL Implementation

```
01 | SELECT COUNT(*) AS Count_Per_year, Year_Date
02 | FROM(
03 |     SELECT EXTRACT(
04 |         YEAR
05 |         FROM TO_DATE(PUBLICATION_DATE DEFAULT NULL ON CONVERSION ERROR,
06 |             'MONTH YYYY')
07 |     ) AS Year_Date
```

```
08 |       FROM GCD_ISSUE
09 |  )
10 |  WHERE Year_Date>=1965 AND Year_Date<=1975
11 |  GROUP BY Year_Date
12 |  ORDER BY Year_Date;
```

Results

Count per Year	Year Date
1289	1965
1489	1966
1557	1967
1263	1968
1491	1969
1487	1970
1626	1971
1884	1972
2194	1973
1872	1974
2259	1975

3.1.11 Query 11

Description

Query 11 takes stories that have been reprinted at least 30 times. We select story titles among those that have been reprinted at least 30 times using a sub-query in the 'from' clause. The sub-query counts story IDs that match in story and story reprint, then we only keep stories with a count of at least 30 using "having count" and we order them in descending order of reprint count. The results are shown in the table.

SQL Implementation

```
01 |  SELECT S.TITLE AS Story_Title, Reprint_count
02 |  FROM GCD_STORY S, (
03 |      SELECT S.ID, COUNT(*) AS Reprint_count
04 |      FROM GCD_STORY S, GCD_STORY_REPRINT SR
05 |      WHERE S.ID=SR.ORIGIN_ID
06 |      GROUP BY S.ID
07 |      HAVING COUNT(*)>=30
08 |      ORDER BY Reprint_count DESC
09 |  ) ID_TO_COUNT
10 |  WHERE S.ID=ID_TO_COUNT.ID;
```

Results

Story Title	Reprint Count
Spaghetti Brothers - historien om søsknene Centobucchi	90
Spider-Man!	52
Verso l'ignoto	46
Spider Man	43
	36
The Fantastic Four!	35
Le sortilège du haricot	34
Le nain de Corneloup	34
Spider-Man	33
Flash of Two Worlds!	33
Spider-Man vs. The Chameleon!	32
Is He Man or Monster or... Is He Both!	32
Out of the Darksome Hills	31
The Chameleon Strikes!	30

3.1.12 Query 12

Description

Query 12 returns the top 10 countries with the longest publishing time between 1600 and 2020, along with the max years publishing and the average years per country. We start by selecting country names among countries where publishers have published for the longest period, using a sub-query in the 'from' clause. A sub-sub-query is made in the sub-query's 'from' clause to select publishing periods between 1600 and 2020 both included. Finally the original query only keeps the top ten countries 'rownum <= 10'. The results are shown in the table.

SQL Implementation

```
01 | SELECT C.NAME AS Country_Name,
02 |     Max_years AS Max_Years_Publishing_Per_Country,
03 |     Average_years AS Average_Years_Per_Country
04 | FROM STDDATA_COUNTRY C, (
05 |     SELECT COUNTRY_ID,
06 |         MAX(Publishing_period) AS Max_years,
07 |         ROUND(AVG(Publishing_period), 4) AS Average_years
08 |     FROM (
09 |         SELECT COUNTRY_ID, YEAR_ENDED-YEAR_BEGAN AS Publishing_period
10 |         FROM GCD_PUBLISHER
11 |         WHERE YEAR_BEGAN>=1600
12 |             AND YEAR_BEGAN<=2020
13 |             AND YEAR_ENDED>=1600
14 |             AND YEAR_ENDED<=2020
```

```
15 |       )
16 |       GROUP BY COUNTRY_ID
17 |       ORDER BY Max_years DESC
18 |  )
19 |  WHERE C.ID=COUNTRY_ID AND ROWNUM<=10;
```

Results

Country Name	Max Years Publishing Per Country	Average Years Per Country
Switzerland	206	47.2778
Germany	194	18.1682
Norway	165	22
Australia	159	15.8929
United Kingdom	156	5.3946
France	150	15.8584
United States	150	4.0622
Belgium	129	9.2059
Netherlands	114	7.2655
Spain	76	15.8696

3.1.13 Query 13

Description

Query 13 returns all Marvel heroes that have appeared in Marvel-DC crossover stories. We start by selecting distinct Marvel heroes in lower case to be able to compare strings. We first select the features among features that appear in Marvel IP issues only, using a first sub-query in the 'from' clause. The first sub-query has a sub-sub-query in its 'from' clause that selects IDs and names among Indicia Publishers that partially match 'marvel' ('where lower(name) like '%marvel%')) and not 'dc'. We also select among features that appear in 'marvel-dc' crossovers using the second sub-query, that is similar to the first but selects features from stories and issues where both 'marvel' and 'dc' have a partial match. Finally the original query keeps crossover features that have a partial match with marvel features, and the result is shown in the table.

SQL Implementation

```
01 |  SELECT DISTINCT LOWER(marvel.FEATURE)
02 |  FROM ( SELECT DISTINCT S.FEATURE
03 |      FROM GCD_STORY S, GCD_ISSUE I, (
04 |          SELECT ID, NAME
05 |          FROM GCD_INDICIA_PUBLISHER
06 |          WHERE LOWER(NAME) LIKE '%marvel%' AND LOWER(NAME) NOT LIKE '%dc%'
07 |      ) Marvel_IP
```

```
08 |      WHERE (S.ISSUE_ID = I.ID) AND (I.INDICIA_PUBLISHER_ID = Marvel_IP.ID)
09 |      AND S.FEATURE IS NOT NULL
10 |  ) marvel, (
11 |      SELECT DISTINCT S.FEATURE
12 |      FROM GCD_STORY S, GCD_ISSUE I, (
13 |          SELECT ID, NAME
14 |          FROM GCD_INDICIA_PUBLISHER
15 |          WHERE LOWER(NAME) LIKE '%marvel%' AND LOWER(NAME) LIKE '%dc%'
16 |      ) Marvel_DC_IP
17 |      WHERE (S.ISSUE_ID = I.ID) AND (I.INDICIA_PUBLISHER_ID = Marvel_DC_IP.
       ID)
18 |      AND S.FEATURE IS NOT NULL
19 |  ) dc_marvel
20 |  WHERE LOWER(dc_marvel.feature) LIKE CONCAT(CONCAT('%', LOWER(marvel.
       feature)), '%');
```

Results

Crossover Feature Heroes
batman
wizard of oz
hulk
x-men
superman
kitty pryde
doctor octopus
spider-man
wizard
superman; spider-man
titan
it
lex luthor; doctor octopus
superman and spider-man
oz
thor

3.1.14 Query 14

Description

Query 14 extracts the top 2 publishers by the number of series published for every country that has at least 200 publishers. Using a sub-query 1 (line 03) in the 'from' clause of the original query, we select country IDs and partition the data into intermediate sub-tables ordered by descending series count. Then using a sub-query 2 (line 06) we keep Publisher and Country IDs of publishers

having a series count of at least 200 (line 16). The series count is done using a sub-query 3 (line 08) in sub-query (2)'s 'from' clause. Finally in the original query, we only keep the top 2 publishing countries for each sub-table, and we show the total results in the table.

SQL Implementation

```
01 | SELECT C.NAME AS Country_Name, P.NAME AS Publisher_Name
02 | FROM STDDATA_COUNTRY C, GCD_PUBLISHER P, (
03 |     SELECT COUNTRY_ID AS C_ID, ID AS P_ID,
04 |         ROW_NUMBER() OVER(PARTITION BY COUNTRY_ID ORDER BY Series_count
       DESC) AS Row_number
05 |     FROM (
06 |         SELECT P.ID, P.COUNTRY_ID, Series_count
07 |         FROM GCD_PUBLISHER P, (
08 |             SELECT PUBLISHER_ID, COUNT(*) AS Series_count
09 |             FROM GCD_SERIES
10 |             GROUP BY PUBLISHER_ID
11 |         )
12 |         WHERE P.ID=PUBLISHER_ID AND COUNTRY_ID IN (
13 |             SELECT COUNTRY_ID
14 |             FROM GCD_PUBLISHER
15 |             GROUP BY COUNTRY_ID
16 |             HAVING COUNT(*)>=200
17 |         )
18 |     )
19 | )
20 | WHERE Row_number<=2 AND C.ID=C_ID AND P.ID=P_ID;
```

Results

Country Name	Publisher Name
Canada	Drawn & Quarterly
Canada	Bell Features
Germany	Panini Deutschland
Germany	Egmont Ehapa
Denmark	Interpresse
Denmark	Forlaget Carlsen
France	Panini France
France	Dargaud éditions
United Kingdom	IPC
United Kingdom	Titan
Italy	Arnoldo Mondadori Editore
Italy	Sergio Bonelli Editore
Netherlands	Arboris
Netherlands	Oberon
Norway	Hjemmet / Egmont
Norway	Semic
Sweden	Semic
Sweden	Egmont
United States	Marvel
United States	DC

3.2 Query Performance Analysis – Indexing

Below, we give the runtime of all the queries of deliverable 3 in seconds.

Query	Run 1	Run 2	Run 3
1	0.049	0.057	0.051
2	0.047	0.048	0.044
3	0.011	0.025	0.012
4	0.012	0.010	0.010
5	0.244	0.224	0.214
6	0.171	0.179	0.177
7	0.782	0.823	0.774
8	1.179	1.177	1.229
9	3.813	3.837	3.894
10	1.516	1.519	1.499
11	0.211	0.213	0.212
12	0.005	0.006	0.006
13	5.13	5.208	5.164
14	0.06	0.064	0.061

We can see that the queries that takes more than one second are 13, 9, 10 and 8. However, not all the queries can be optimized. For example, if we try to use indexes with queries 9 or 10 it will not improve the performance as the columns on which we operate only contain strings.

The queries that we have been able to optimize using indexes are 2, 5 and 13. You can see the results below given with the index that was created.

Query	Run 1	Run 2	Run 3	CREATE INDEX index0 ON
2	0.029	0.030	0.029	GCD_SERIES(PUBLISHER_ID);
5	0.141	0.138	0.142	GCD_ISSUE(series_id, indicia_publisher_id);
13	4.578	4.599	4.555	GCD_STORY(issue_id);

For the query 2, the query plan shows a total cost of 3648 and the cost to access GCD_SERIES is 3478. When we create the index and run the query, this cost drops to 229 and the total cost to 235. The GROUP BY and the equality condition on publisher_id are done easier as we do not need to do a full scan.

For the query 5, the total cost is given as 8974 and that corresponds mostly to the full scan on GCD_ISSUE which has a cost of 8918. When we create the index this, the cost on GCD_ISSUE drops to 987 and the total cost drops to 1043. Again, the index accelerate the runnin time on the equality conditions

For the query 13, we have a total cost of 284994, and a consequent part of this cost is due to the merged join with the equality on issue_id from GCD_STORY with a cost 207300 that can be reduced to 170272 thanks to the index.

3.3 General Comments

In query 7, one of the extracted Feature names is '?', which is due to dirty data (one of the heroes name is unknown!). Also in query 11 a NULL value is present in the results. This value has been kept as the reprint_count is obtained by counting story_ID meaning that this NULL value correspond to one single ID.

References

[1] Ramakrishnan and Gehrke. *Database Management Systems, 3rd edition.* Mc Grawhill, 2003.

A Appendix

A.1 Provided ER Model

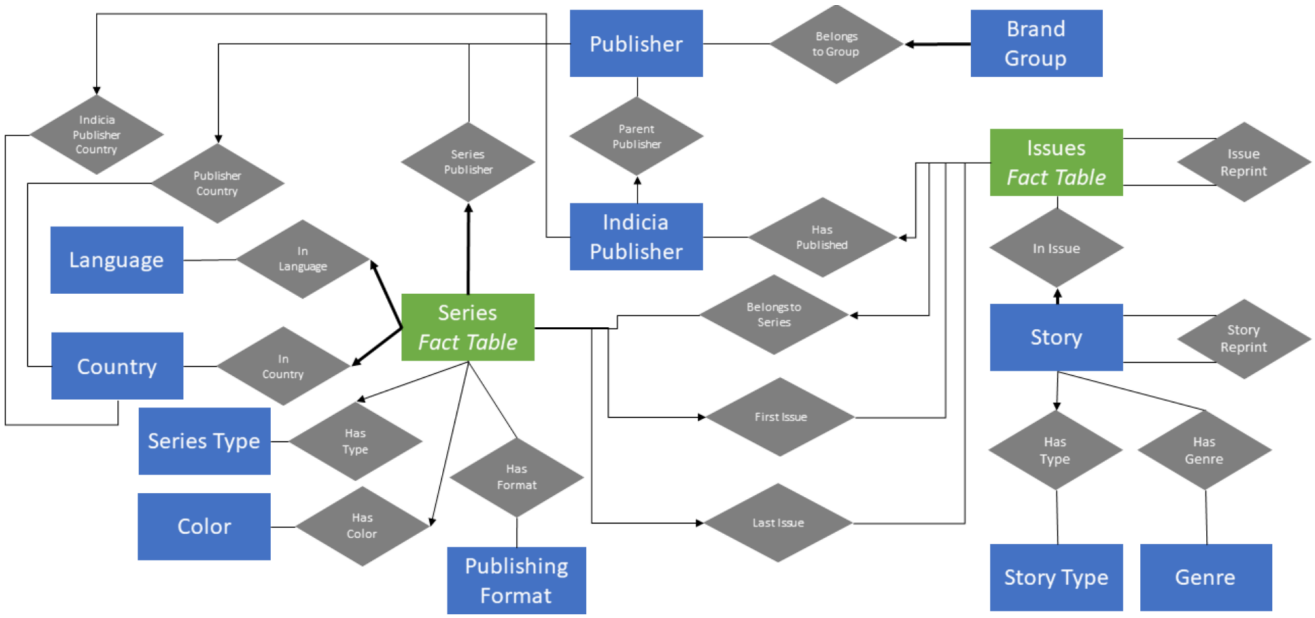

Figure 3: Provided ER Diagram

A.2 Provided DDL

```
01 | create table STDDATA_LANGUAGE
02 | (
03 |     ID   NUMBER(3) primary key,
04 |     CODE VARCHAR2(3) unique not null,
05 |     NAME VARCHAR2(64)
06 | );
07 |
08 | create table STDDATA_COUNTRY
09 | (
10 |     ID   NUMBER(3) primary key ,
11 |     CODE VARCHAR2(4) unique not null,
12 |     NAME VARCHAR2(64)
13 | );
14 |
15 | create table GCD_STORY_TYPE
16 | (
17 |     ID   NUMBER(2) primary key,
18 |     NAME VARCHAR2(64)
19 | );
20 |
```

```
21 | create table GCD_SERIES_PUBLICATION_TYPE
22 | (
23 |     ID   NUMBER(1) primary key,
24 |     NAME VARCHAR2(16)
25 | );
26 |
27 | create table GCD_SERIES_PUBLISHING_FORMAT
28 | (
29 |     ID                NUMBER(38) primary key,
30 |     PUBLISHING_FORMAT VARCHAR2(256) not null
31 | );
32 |
33 | create table GCD_PUBLISHER
34 | (
35 |     URL         VARCHAR2(256),
36 |     NOTES       VARCHAR2(4000),
37 |     YEAR_ENDED  NUMBER(4),
38 |     YEAR_BEGAN  NUMBER(4),
39 |     COUNTRY_ID  NUMBER(3),
40 |     NAME        VARCHAR2(256),
41 |     ID          INTEGER primary key,
42 |     constraint GCD_PUBLISHER_COUNTRY_FK
43 |         foreign key (COUNTRY_ID) references STDDATA_COUNTRY(ID)
44 | );
45 |
46 | create table GCD_INDICIA_PUBLISHER
47 | (
48 |     ID            INTEGER primary key,
49 |     NAME          VARCHAR2(256),
50 |     PUBLISHER_ID  INTEGER,
51 |     COUNTRY_ID    NUMBER(3),
52 |     YEAR_BEGAN    NUMBER(4),
53 |     YEAR_ENDED    NUMBER(4),
54 |     IS_SURROGATE  NUMBER(38),
55 |     NOTES         VARCHAR2(4000),
56 |     URL           VARCHAR2(256),
57 |     constraint GCD_INDICIA_PUBLISHER_COUNTRY_FK
58 |         foreign key (COUNTRY_ID) references STDDATA_COUNTRY(ID),
59 |     constraint GCD_INDICIA_PUBLISHER_PUBLISHER_FK
60 |         foreign key (PUBLISHER_ID) references GCD_PUBLISHER
61 | );
62 |
63 | create table GCD_BRAND_GROUP
64 | (
65 |     ID            NUMBER(38) primary key,
66 |     NAME          VARCHAR2(256),
67 |     YEAR_BEGAN    NUMBER(4),
68 |     YEAR_ENDED    NUMBER(4),
69 |     NOTES         VARCHAR2(4000),
70 |     URL           VARCHAR2(256),
71 |     PUBLISHER_ID  INTEGER not null,
72 |     constraint GCD_BRAND_GROUP_PUB_FK
```

```
73 |          foreign key (PUBLISHER_ID) references GCD_PUBLISHER(ID)
74 | );
75 |
76 | create table GCD_ISSUE
77 | (
78 |     ID                   NUMBER(38) primary key,
79 |     EDITION_NUMBER       VARCHAR2(64),
80 |     SERIES_ID            NUMBER(38),
81 |     INDICIA_PUBLISHER_ID INTEGER,
82 |     PUBLICATION_DATE     VARCHAR2(64),
83 |     PRICE                NUMBER(38, 2),
84 |     CURRENCY             VARCHAR2(26),
85 |     PAGE_COUNT           INTEGER,
86 |     INDICIA_FREQUENCY    VARCHAR2(256),
87 |     EDITING              VARCHAR2(2048),
88 |     NOTES                VARCHAR2(4000),
89 |     ISBN                 VARCHAR2(64),
90 |     VALID_ISBN           VARCHAR2(16),
91 |     BARCODE              VARCHAR2(64),
92 |     TITLE                VARCHAR2(128),
93 |     ON_SALE_DATE         VARCHAR2(26),
94 |     RATING               VARCHAR2(128)
95 | );
96 |
97 | create table GCD_ISSUE_REPRINT
98 | (
99 |     ID              NUMBER(20) primary key,
100|     ORIGIN_ISSUE_ID NUMBER(20),
101|     TARGET_ISSUE_ID NUMBER(20),
102|     constraint GCD_ORIGIN_ISSUE_FK
103|         foreign key (ORIGIN_ISSUE_ID) references GCD_ISSUE,
104|     constraint GCD_TARGET_ISSUE_FK
105|         foreign key (TARGET_ISSUE_ID) references GCD_ISSUE
106| );
107|
108| create table GCD_STORY_GENRE
109| (
110|     ID    NUMBER(3) primary key,
111|     GENRE VARCHAR2(128) not null
112| );
113|
114| create table GCD_SERIES
115| (
116|     ID                  NUMBER(38) primary key,
117|     NAME                VARCHAR2(256),
118|     FORMAT              VARCHAR2(256),
119|     YEAR_BEGAN          NUMBER(4),
120|     YEAR_ENDED          NUMBER(4),
121|     PUBLICATION_DATES   VARCHAR2(128),
122|     FIRST_ISSUE_ID      NUMBER(38),
123|     LAST_ISSUE_ID       NUMBER(38),
124|     PUBLISHER_ID        INTEGER,
```

```
125 |      COUNTRY_ID           NUMBER(3),
126 |      LANGUAGE_ID          NUMBER(3),
127 |      NOTES                VARCHAR2(4000),
128 |      COLOR                VARCHAR2(256),
129 |      DIMENSIONS           VARCHAR2(256),
130 |      PAPER_STOCK          VARCHAR2(256),
131 |      BINDING              VARCHAR2(256),
132 |      PUBLICATION_TYPE_ID  NUMBER(1),
133 |      constraint GCD_SERIES_COUNTRY_FK
134 |          foreign key (COUNTRY_ID) references STDDATA_COUNTRY(ID),
135 |      constraint GCD_SERIES_FIRST_ISSUE_FK
136 |          foreign key (FIRST_ISSUE_ID) references GCD_ISSUE,
137 |      constraint GCD_SERIES_LANGUAGE_FK
138 |          foreign key (LANGUAGE_ID) references STDDATA_LANGUAGE(ID),
139 |      constraint GCD_SERIES_LAST_ISSUE_FK
140 |          foreign key (LAST_ISSUE_ID) references GCD_ISSUE,
141 |      constraint GCD_SERIES_PUBLICATION_TYPE_FK
142 |          foreign key (PUBLICATION_TYPE_ID) references
        GCD_SERIES_PUBLICATION_TYPE,
143 |      constraint GCD_SERIES_PUBLISHER_FK
144 |          foreign key (PUBLISHER_ID) references GCD_PUBLISHER
145 |  );
146 |
147 |  create table GCD_SERIES_TO_PUBLISHING_FORMAT
148 |  (
149 |      SERIES_ID             NUMBER(38) references GCD_SERIES,
150 |      PUBLISHING_FORMAT_ID  NUMBER(38) references
        GCD_SERIES_PUBLISHING_FORMAT,
151 |      constraint GCD_SERIES_TO_PUBLISHING_FORMAT_PK
152 |          primary key (SERIES_ID, PUBLISHING_FORMAT_ID)
153 |  );
154 |
155 |  create table GCD_STORY
156 |  (
157 |      ID             NUMBER(38) primary key,
158 |      TITLE          VARCHAR2(512),
159 |      FEATURE        VARCHAR2(512),
160 |      ISSUE_ID       NUMBER(38) not null references GCD_ISSUE(ID),
161 |      SCRIPT         VARCHAR2(2048),
162 |      PENCILS        VARCHAR2(2048),
163 |      INKS           VARCHAR2(2048),
164 |      COLORS         VARCHAR2(2048),
165 |      LETTERS        VARCHAR2(2048),
166 |      EDITING        VARCHAR2(2048),
167 |      CHARACTERS     VARCHAR2(4000),
168 |      SYNOPSIS       VARCHAR2(4000),
169 |      REPRINT_NOTES  VARCHAR2(4000),
170 |      NOTES          VARCHAR2(4000),
171 |      TYPE_ID        NUMBER(2) references GCD_STORY_TYPE(ID)
172 |  );
173 |
174 |  create table GCD_STORY_REPRINT
```

```
175 | (
176 |     ID         NUMBER(20) primary key ,
177 |     ORIGIN_ID  NUMBER(38) references GCD_STORY,
178 |     TARGET_ID  NUMBER(38) references GCD_STORY
179 | );
180 |
181 | create table GCD_STORY_TO_GENRE
182 | (
183 |     STORY_ID NUMBER(38),
184 |     GENRE_ID NUMBER(3),
185 |     constraint GCD_STORY_TO_GENRE_PK
186 |         primary key (STORY_ID, GENRE_ID),
187 |     constraint GCD_STORY_TO_GENRE_GENRE_FK
188 |         foreign key (GENRE_ID) references GCD_STORY_GENRE,
189 |     constraint GCD_STORY_TO_GENRE_STORY_FK
190 |         foreign key (STORY_ID) references GCD_STORY
191 | );
192 |
193 | alter table GCD_ISSUE
194 | add constraint GCD_ISSUE_SERIES_ID_FK foreign key (SERIES_ID) references
        GCD_SERIES(ID);
```

A.3 Constraints enabling

```
01 | BEGIN
02 |     FOR c IN
03 |     (SELECT c.owner, c.table_name, c.constraint_name
04 |         FROM user_constraints c, user_tables t
05 |         WHERE c.table_name = t.table_name
06 |         AND c.status = 'DISABLED'
07 |         ORDER BY c.constraint_type)
08 |     LOOP
09 |         dbms_utility.exec_ddl_statement('alter table "' || c.owner || '
        "."' || c.table_name || '" enable
10 |         constraint ' || c.constraint_name);
11 |     END LOOP;
12 | END;
```

A.4 Constraints disabling

```
01 | BEGIN
02 |     FOR c IN
03 |     (SELECT c.owner, c.table_name, c.constraint_name
04 |         FROM user_constraints c, user_tables t
05 |         WHERE c.table_name = t.table_name
06 |         AND c.status = 'ENABLED'
```

```
07 |          AND NOT (t.iot_type IS NOT NULL AND c.constraint_type = 'P')
08 |          ORDER BY c.constraint_type DESC)
09 |      LOOP
10 |          dbms_utility.exec_ddl_statement('alter table "' || c.owner || '
       "."' || c.table_name || '"
11 |          disable constraint ' || c.constraint_name);
12 |      END LOOP;
13 |  END;
```