École polytechnique fédérale de Lausanne

Introduction to Database Systems
CS-322

Database Project

Spring 2022 — Grand Comics Database

Team 1
François Dumoncel, Souleyman Boudouh, Fouad Mahmoud

23 March 2022

Contents

Chapter 1

ER model and DDL

1.1 Entity Relationship Schema

We will build the ER model step by step, first starting by what seems to be the central data, then expanding until we consider the smaller datasets. Finally we will discuss the possibility of transforming some of the attributes of our entities as new dimensions in the ER model.

1.1.1 Story, Series and Issues

We can see in the Story dataset that multiple stories can be featured in one issue, following with our previous example: the saga 'Trick' has all its 7 stories featured in issue 7. But since there are no stories spanning over multiple issues, and the Issue dataset contains no information about stories, we can only assess the following one-to-many relationship: **Featured**.
Stories are sometimes reprinted, all the reprints form a relation of two stories with different IDs that is described in the **Story Reprint dataset**. For every tuple, there is the reprint ID, which associates two different stories IDs. This dataset hence forms an unary relationship of Story with itself.

Secondly, we can see in the Issues dataset that the same series can be published in multiple issues, (e.g. serie ID 87 is associated to issue ID 258 and 267), but each issue can only be used to publish one part of one series. So this connection between Series and Issues will be represented by the one-to-many relationship **Publishes**.
Also, issues can sometimes be reprinted. In the **Issue Reprint dataset**, every tuple is composed of the id of one issue, and the id of the reprinted issue. Hence this dataset obviously forms an unary relationship of Issues with itself.

Finally, every series in the Series dataset has two connections with the Issues datasets, namely the first issue, and the last issue the series was published in. Some series are still on-going and thus do not have a last issue, we will represent these two different connections using two distinct relationships: **First publication** and **Last publication**.

1.1.2 Publisher, Indicia Publisher and Brand Group

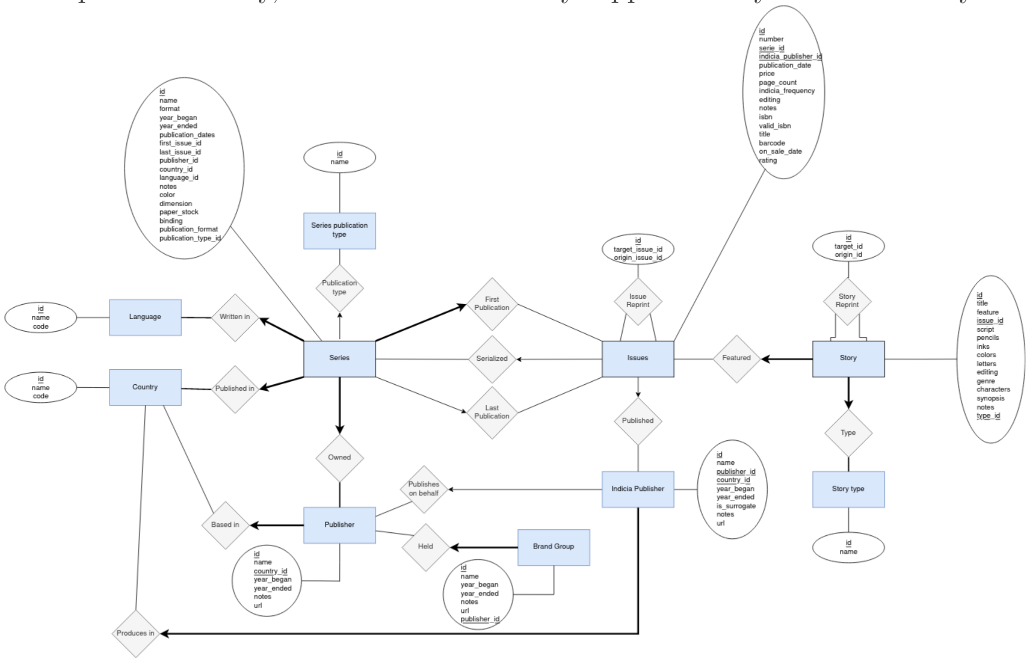
First we will see how these three datasets are related to eachothers, then we will expand their connections with the Story, Series and Issues datasets.

The Indicia Publisher dataset has only one connection to the Publisher dataset, namely the master publisher the company publishes on behalf of. After looking online, we saw that there may exist independant indicia publishers, however after looking at the dataset, we can be sure that every indicia publisher may **publish on behalf of at most one** master publisher. When it comes to the Brand Group dataset, we see that every brand group is **held by one publisher**, note that one publisher may hold multiple groups.

Trivially, an issue is published by at most one indicia publisher, however, we can see in the Issues dataset that some issues are not assigned to any indicia publisher ID, for example issue with ID 265 has no Indicia Publisher ID attribute. Hence we can only tell that an issue may be **published by at most one** indicia publisher.
Finally, according to the Series dataset, a single publisher may own multiple series (e.g. publisher ID 78 owns series ID 409-420), however, we have not found any instance of a series being owned by multiple publishers, especially since only the series dataset holds information relevant to the relationship between the two entities, hence we deduce that series can be **owned by at most one** publisher.

1.1.3 ER diagram

Using all our previous information, and linking the remaining datasets, i.e. Publication Type, Story Type, Language, Country, we add constraints whenever we see that the relationships formed in the datasets have no blank/null values. For example, all publishers have exactly one country assigned to them so we can use an exactly-one constraint. We obtain the following diagram. Note that we kept the same attributes as given in the description for clarity, and we did not add any supplementary attribute to any relationship:



id name format year_began year_ended publication_dates first_issue_id last_issue_id publisher_id country_id language_id notes color dimension paper_stock binding publication_format publication_type_id
id name
Series publication type
Publication type
id name code
Language
Written in
Series
First Publication
Serialized
Last Publication
id name code
Country
Published in
Owned
Publishes on behalf
Based in
Publisher
Held
Brand Group
id name country_id year_began year_ended notes url
id name year_began year_ended notes url publisher_id
Produces in
id number serie_id indicia_publisher_id publication_date price page_count indicia_frequency editing notes isbn valid_isbn title barcode on_sale_date rating
id target_issue_id origin_issue_id
Issue Reprint
Issues
Featured
id target_id origin_id
Story Reprint
Story
id title feature issue_id script pencils inks colors letters editing genre characters synopsis notes type_id
Published
Indicia Publisher
id name publisher_id country_id year_began year_ended is_surrogate notes url
Type
Story type
id name

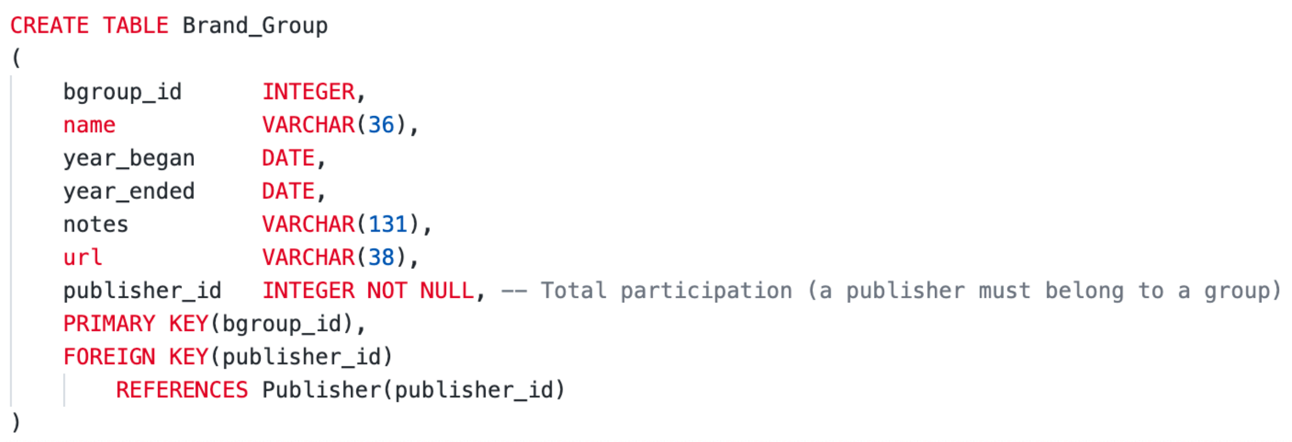
1.2 Relational Model: DDL

In the dataset, many fields are strings of variable length, we have used a python script that scan the whole column to find the maximum size that a string can have.

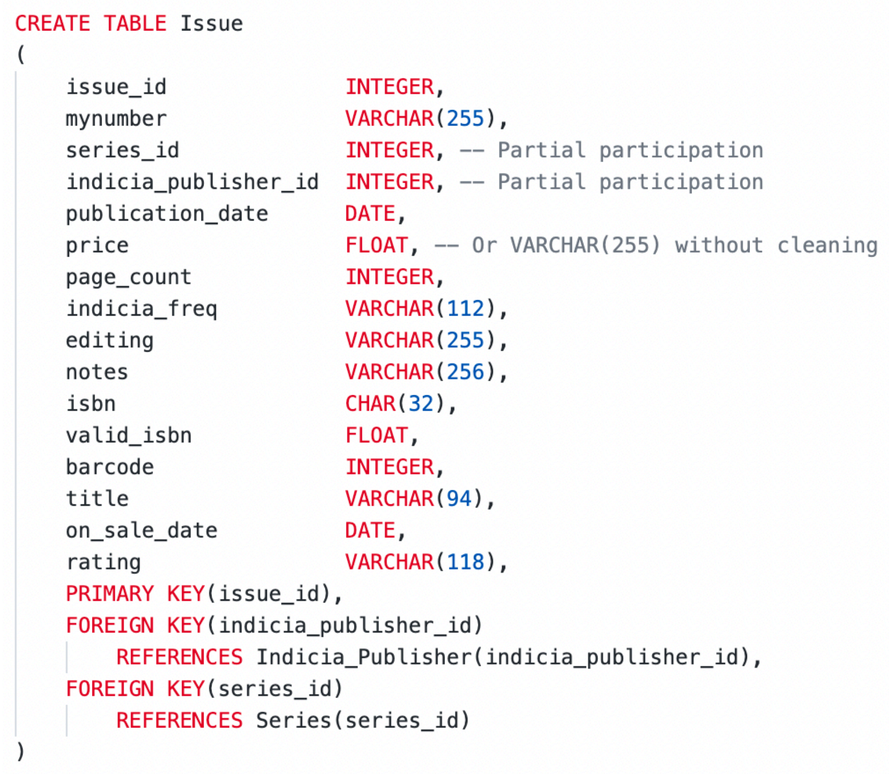
```python
from pandas import *

# Use only if the field consist of string

data = read_csv("GCD_2022/name_data_file.csv", sep=',')

col = data['column_name'].tolist()

str_col = [str(x) for x in col]

curr_max = 0
for elem in name:
    if (elem != "nan") and (len(elem) > curr_max):
        curr_max = len(elem)

print(curr_max)
```

```sql
CREATE TABLE Country
(
    country_id INTEGER,
    country    VARCHAR(60),
    code       VARCHAR(10),
    PRIMARY KEY(country_id)
)

CREATE TABLE Language
(
    language_id  INTEGER,
    code         CHAR(3),
    name         VARCHAR(27),
    PRIMARY KEY(language_id)
)
```

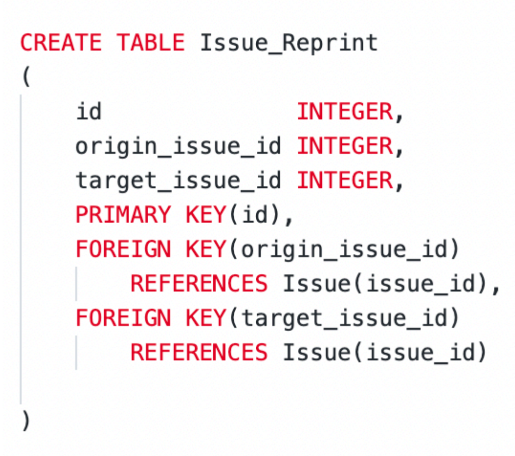
```sql
CREATE TABLE Series_Publication_Type
(
    id       INTEGER,
    name     VARCHAR(8),
    PRIMARY KEY(id)
)

CREATE TABLE Story_Type
(
    id      INTEGER,
    name    VARCHAR(50),
    PRIMARY KEY(id)
)

CREATE TABLE Publisher
(
    publisher_id    INTEGER,
    name            VARCHAR(124),
    country_id      INTEGER NOT NULL, -- Total participation
    year_began      DATE,
    year_ended      DATE,
    notes           VARCHAR(255),
    url             VARCHAR(82),
    PRIMARY KEY(publisher_id),
    FOREIGN KEY(country_id)
        REFERENCES Country(country_id)
)

CREATE TABLE Indicia_Publisher
(
    indicia_publisher_id INTEGER,
    name                 VARCHAR(25),
    publisher_id         INTEGER, -- Partial participation
    country_id           INTEGER NOT NULL, -- Total participation
    year_began           DATE,
    year_ended           DATE,
    is_surrogate         BOOLEAN,
    notes                VARCHAR(226),
    url                  VARCHAR(115),
    PRIMARY KEY(indicia_publisher_id),
    FOREIGN KEY(publisher_id)
        REFERENCES Publisher(publisher_id),
    FOREIGN KEY(country_id)
        REFERENCES Country(country_id)
)
```



```
CREATE TABLE Brand_Group
(
    bgroup_id       INTEGER,
    name            VARCHAR(36),
    year_began      DATE,
    year_ended      DATE,
    notes           VARCHAR(131),
    url             VARCHAR(38),
    publisher_id    INTEGER NOT NULL, -- Total participation (a publisher must belong to a group)
    PRIMARY KEY(bgroup_id),
    FOREIGN KEY(publisher_id)
        REFERENCES Publisher(publisher_id)
)
```



```
CREATE TABLE Issue
(
    issue_id                INTEGER,
    mynumber                VARCHAR(255),
    series_id               INTEGER, -- Partial participation
    indicia_publisher_id    INTEGER, -- Partial participation
    publication_date        DATE,
    price                   FLOAT, -- Or VARCHAR(255) without cleaning
    page_count              INTEGER,
    indicia_freq            VARCHAR(112),
    editing                 VARCHAR(255),
    notes                   VARCHAR(256),
    isbn                    CHAR(32),
    valid_isbn              FLOAT,
    barcode                 INTEGER,
    title                   VARCHAR(94),
    on_sale_date            DATE,
    rating                  VARCHAR(118),
    PRIMARY KEY(issue_id),
    FOREIGN KEY(indicia_publisher_id)
        REFERENCES Indicia_Publisher(indicia_publisher_id),
    FOREIGN KEY(series_id)
        REFERENCES Series(series_id)
)
```



```
CREATE TABLE Issue_Reprint
(
    id               INTEGER,
    origin_issue_id  INTEGER,
    target_issue_id  INTEGER,
    PRIMARY KEY(id),
    FOREIGN KEY(origin_issue_id)
        REFERENCES Issue(issue_id),
    FOREIGN KEY(target_issue_id)
        REFERENCES Issue(issue_id)

)
```

```
-- In the dataset :
--     line 8230    has 29 fields instead of 16
--     line 16162   has 29 fields instead of 16
--     line 166744  has 18 fields instead of 16
--     line 395218  has 17 fields instead of 16
--     line 426091  has 18 fields instead of 16
--     line 431008  has 29 fields instead of 16
--     line 524250  has 27 fields instead of 16
--     line 743795  has 27 fields instead of 16
--     line 760431  has 27 fields instead of 16
--     line 868086  has 29 fields instead of 16
--     line 1143326 has 29 fields instead of 16
--     line 1147743 has 29 fields instead of 16
--     line 1271485 has 29 fields instead of 16
--     line 1468794 has 22 fields instead of 16
--     line 1602946 has 22 fields instead of 16
--     line 1657428 has 22 fields instead of 16
CREATE TABLE Story
(
    id              INTEGER,
    title           VARCHAR(32999),
    feature         VARCHAR(589),
    issue_id        INTEGER NOT NULL, -- Total participation
    script          VARCHAR(1166),
    pencils         VARCHAR(1702),
    inks            VARCHAR(1610),
    colors          VARCHAR(1468),
    letters         VARCHAR(776),
    editing         VARCHAR(347),
    genre           VARCHAR(125),
    characters      VARCHAR(),
    synopsis        VARCHAR(6276),
    reprint_notes   VARCHAR(3562),
    notes           VARCHAR(10374),
    type_id         INTEGER NOT NULL, -- Total participation
    PRIMARY KEY(id),
    FOREIGN KEY(issue_id)
        REFERENCES Issue(issue_id),
    FOREIGN KEY(type_id)
        REFERENCES Story_Type(id)
)
```

```sql
CREATE TABLE Story_Reprint
(
    id          INTEGER,
    origin_id   INTEGER,
    target_id   INTEGER,
    PRIMARY KEY(id),
    FOREIGN KEY(origin_id)
        REFERENCES Story(id),
    FOREIGN KEY(target_id)
        REFERENCES Story(id)

)
-- In the dataset, line 53073 has 20 fields instead of 18
CREATE TABLE Series
(
    series_id           INTEGER,
    name                VARCHAR(239),
    format              VARCHAR(126),
    year_began          DATE,
    year_ended          DATE,
    publication_dates   DATE, -- After cleaning
    first_issue_id      INTEGER NOT NULL, -- Total participation
    last_issue_id       INTEGER, -- Partial participation
    publisher_id        INTEGER NOT NULL, -- Total participation
    country_id          INTEGER NOT NULL, -- Total participation
    language_id         INTEGER NOT NULL, -- Total participation
    notes               VARCHAR(256),
    color               VARCHAR(137),
    dimensions          VARCHAR(171),
    paper_stock         VARCHAR(117),
    binding             VARCHAR(90),
    publishing_format   VARCHAR(92),
    publication_type_id INTEGER,
    PRIMARY KEY(series_id),
    FOREIGN KEY(first_issue_id)
        REFERENCES Issue(issue_id) -- issue_id (changed)
        REFERENCES Issue(issue_id), -- issue_id (changed)
    FOREIGN KEY(publisher_id)
        REFERENCES Publisher(publisher_id),
    FOREIGN KEY(country_id)
        REFERENCES Country(country_id),
    FOREIGN KEY(language_id)
        REFERENCES Language(language_id),
    FOREIGN KEY(publication_type_id)
        REFERENCES Series_Publication_Type(id)
)
```

1.3 Data cleaning : how and why

The first remark we can make concerning the Series dataset is that some attributes, namely Color, Dimension, Paper-Stock, Binding and Publishing Format all have very few values that are repeated a lot throughout the dataset, but aren't necessarily easily attainable due to the values not being homogeneous.
For example, in the color dataset, 29 out of the first 35 tuples have "color" as a Color attribute, with many variations including combinations of uppercase and lowercase letters. Also, some series have multiple colors, but only one attribute to describe it, therefore when querying all series that have colors red and white, we need to know about all possible combinations of key words, e.g. "red white", "Red and White", "red White" etc. This observation can also be made for the Dimension, Paper-Stock, Binding and Publishing Format attributes. We hence propose to fix these problems the following ways:

1. Put all attributes in lowercase and set a standard of acceptable symbols (e.g. hyphens are used for attributes like "saddle-stitched"), replace all instances of unwanted symbols with one of the acceptable symbols.

2. Go over the values of every cleaned attributes to make-up a new dimension.

3. Assign a unique identifier to each element of the new entity we just formed, which will be used as primary keys.

4. Use these identifiers as foreign keys in the entities from where the attribute comes from.

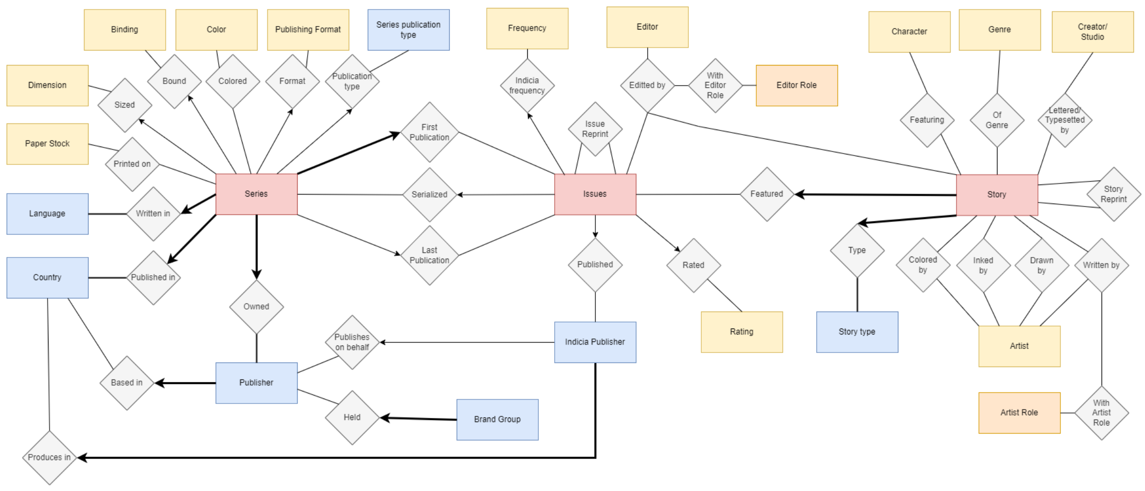
Using the new dimensions and their foreign keys, we can now create relationships between facts and their dimensions which makes queries more robust.

We would apply the same method for the Issues dataset which also features many repeated values with unclean variations, e.g. "Monthly" and "monthly" for the Indicia Frequency, or "Malcolm Wheeler-Nicholson" which is a very frequent name in the Editing attribute. Moreover, there are often multiple editors with different roles (typed between brackets) in the Editing attribute. One way to clean this attribute's dataset is to have an Editor entity with a relationship to the Issue entity, aggregated with an Editor Role entity, since one editor can have different roles. The Role entity would be created using the method above. The same goes for the attributes of the Story entity, sometimes the script is written by the same person who drew and wrote the comic, so instead of having many repetitions of the same name, we introduce a new dimension called Artist, with multiple relationships like **Written By**, **Drawn By** or **Inked By**. Also, sometimes the artists writing the script have different roles that are also repeated throughout the dataset, hence we introduce another aggregation with an Artist Role dimension.

Other attributes needing to be cleaned are any date attributes which are not always written using the same format. For example, the Publication Date attribute contains values such as "Jan-38", "March-April 1938", "September 9 1938" or even "[circa 1910?]" which have very different formats. The same goes for the On Sale attribute. One way of homogenizing these datasets is to enforce a certain date format. However applying an algorithm to transform the date values into a specific format may not be feasible with the given data and prone to errors as it would be difficult to for example know if a parsed number less than or equal to 12 is a day or a month. Therefore it would be best to leave this dataset as is and represent it as a VARCHAR type in the relational model.

Other attributes that could be reformatted are My Numbers, which could be constrained to only integer values or left as a VARCHAR type. The ISBN and the Valid ISBN attributes can be reduces into one attribute with a valid ISBN format. Finally the Price attribute is sometimes displayed using multiple currencies. One way of homogenizing its dataset is by having a standard currency to be used for the entire dataset and for each Price value read, read the value before the first ISO code (Currency type) and transform it into the standard currency, for instance in USD, before storing it. We should also remove any outlier values for some attributes such as a ”0” for the Year Began or Year Ended attributes in the Publisher dataset to avoid working with erroneous data.

Granted all the cleaning done, we would eventually obtain the following diagram. We mark dimensions in yellow, facts in red and the extra entities using aggregation in orange:



Chapter 2

Import the Data, and first SQL queries

2.1 Importing the Data

Using the provided schema and DDL commands, it was quite simple to load the data into the database using DataGrip.

We just ran into a problem, some of the values provided in the text files were incompatible with some domain in the relational model, for example some years were not of the type `NUMBER(4)` as specified by the model, so we made the decision to ignore them and set `null` values instead. This is of little consequence, as the proportion of values that were problematic was minimal.

2.2 First SQL queries

Query 1. *Display brand group names with the highest number of indicia publishers from the United States.*

Description of the logic. For this query we used 3 different views. The first one fetches the number of US Indicia Publishers per Publisher by querying all the Indicia Publishers that have a country ID being that of the United States and have a Publisher ID the same as that of the paired Publisher, then grouping the result by the Publisher ID and obtaining the pairs of Publisher ID ad their corresponding US Indicia Publisher count. The second view simply fetches the Publisher ID(s) with the highest number of US Indicia Publishers by seeing if their crresponding count in the previous view is equal to the maximum count in that view. Lastly, the final view simply outputs the Brand Group names corresponding to that Publisher ID.

```
CREATE VIEW US_IP_COUNT_PER_PUBLISHER AS
```

```
SELECT  P.ID, COUNT(IP.ID) AS IPS
FROM    GCD_PUBLISHER P, GCD_INDICIA_PUBLISHER IP
WHERE   IP.COUNTRY_ID =
            (SELECT C.ID
             FROM   STDDATA_COUNTRY C
             WHERE  C.NAME = 'United States')
        AND
        P.ID = IP.PUBLISHER_ID
GROUP BY P.ID;

CREATE VIEW MAX_US_IP_PUBLISHER_ID AS
SELECT   USP.ID AS PUBLISHER_ID
FROM     US_IP_COUNT_PER_PUBLISHER USP
WHERE USP.IPS=(SELECT MAX(USP2.IPS) FROM US_IP_COUNT_PER_PUBLISHER
    USP2);

SELECT DISTINCT BG.NAME
FROM GCD_BRAND_GROUP BG, MAX_US_IP_PUBLISHER_ID MPID
WHERE BG.PUBLISHER_ID = MPID.PUBLISHER_ID;
```

Result (20 first rows out of 36 rows)

	Brand group name		Brand group name
1	Actual Ender's Game	11	A Romance! Magazine
2	Heavy Hitters	12	Curtis Magazines
3	Malibu	13	DBPRO
4	Marvel	14	Marvel Knights
5	Spider-Man Group	15	A Humorama Magazine
6	Timely Comics	16	M-Tech
7	Ultimate	17	Shadowline
8	2099	18	Marvel; New Universe [white box]
9	Crossgen	19	Midnight Sons
10	Nelson	20	Pizza Hut

Query 2. *Find the IDs and names of publishers who published at least one album-type series in the Italian language. Order the results by the ID, descending (highest first).*

Description of the logic. We select series and publisher that matched and we then select *album* series that have been published in *italian* using two subqueries to find appropriate IDs.

```
SELECT DISTINCT S.PUBLISHER_ID, P.NAME AS PUBLISHER_NAME
FROM   GCD_SERIES S, GCD_PUBLISHER P
WHERE  S.PUBLICATION_TYPE_ID =
          (SELECT P.ID
           FROM   GCD_SERIES_PUBLICATION_TYPE P
           WHERE  P.NAME = 'album')
```

```
        AND
        S.LANGUAGE_ID =
           (SELECT L.ID
            FROM   STDDATA_LANGUAGE L
            WHERE  L.NAME = 'Italian')
        AND
        S.PUBLISHER_ID = P.ID

ORDER BY S.PUBLISHER_ID DESC;
```

Result (16 first rows out of 16 rows)

	Publisher ID	Publisher name		Publisher ID	Publisher name
1	10963	Cliquot	11	845	Edizioni BD
2	10355	Andrea Leggeri	12	442	EPC
2	10349	RW Edizioni	13	440	Edizioni LIsola Trovata
4	10209	7even Age Entertainment	14	397	Milano Libri Edizioni
5	10056	Bel-Ami Edizioni	15	324	Max Bunker Press
6	4881	001 Edizioni	16	164	Sergio Bonelli Editore
7	3166	Rizzoli	17	—	—
8	2693	Panini	18	—	—
9	1177	Bonelli-Dargaud	19	—	—
10	969	Kappa Edizioni	20	—	—

Query 3. *Find the names of all series of publication type book published in Switzerland. Order the resulting names alphabetically (ascending, A to Z).*

Description of the logic. We select series that match publication type id of 'book' and country id of 'Switzerland' using two subqueries.

```
SELECT DISTINCT S.NAME
FROM   GCD_SERIES S
WHERE  S.PUBLICATION_TYPE_ID =
                (SELECT P.ID
                 FROM   GCD_SERIES_PUBLICATION_TYPE P
                 WHERE  P.NAME = 'book')
       AND
       S.COUNTRY_ID = (SELECT C.ID
                       FROM   STDDATA_COUNTRY C
                       WHERE  C.NAME = 'Switzerland')
ORDER BY S.NAME;
```

Result (20 first rows out of 21 rows)

	Serie name		Serie name
1	Affentheater	11	Family Living
2	Alack Sinner	12	Gaza
2	Alans Kindheit	13	Golem im Emmental
4	Alans Krieg	14	Jetzt kommt später
5	Das Geheimnis des Würgers	15	Le Soleil
6	Den Letzten beissen die Hunde	16	Palästina
7	Der Fotograf	17	Victor Levallois
8	Die Reportage	18	* : see below
9	Ein Leben in China	19	Who killed Professor X?
10	Elender Krieg - Gesamtausgabe	20	120 Rue de la Gare

* : Voyages et aventures surprenantes de Robinson Crusoé

Query 4. *We are interested in the number of series per year they started being published, for the year range [1990-2017] (including 1990 and 2017). Sort the output based on the year, descending.*

Description of the logic. We simply keep the series that start between 1990 and 2017, then we group and order by the launch year, using `count` for grouping.

```sql
SELECT    COUNT(*) AS TOTAL, S.YEAR_BEGAN
FROM      GCD_SERIES S
-- Between operator is inclusive
WHERE     S.YEAR_BEGAN BETWEEN 1990 AND 2017
GROUP BY S.YEAR_BEGAN
ORDER BY S.YEAR_BEGAN DESC;
```

Result (20 first rows out of 28 rows)

	Total	Year began		Total	Year began
1	172	2017	11	3039	2007
2	3185	2016	12	2895	2006
3	3421	2015	13	2771	2005
4	3401	2014	14	2414	2004
5	3390	2013	15	2330	2003
6	3217	2012	16	2118	2002
7	3473	2011	17	2002	2001
8	3519	2010	18	2033	2000
9	3180	2009	19	1995	1999
10	3128	2008	20	2070	1998

Query 5. *What are the names of countries (excluding Switzerland!) where the publishers*

have published series, for those publishers who published at least one series in Switzerland.

Description of the logic. We select from tuples of (Country 1, Series 1, Country 2, Series 2) the distinct Countries from Country 1 satisfying the condition of Series 1 and 2 having the same Publisher ID, Series 1 being published in a country other than Switzerland ans Series 2 being Published in Switzerland to get the result.

```sql
SELECT DISTINCT C1.NAME
FROM STDDATA_COUNTRY C1, STDDATA_COUNTRY C2, GCD_SERIES S1,
    GCD_SERIES S2
WHERE C1.ID = S1.COUNTRY_ID
AND C1.NAME != 'Switzerland'
AND C2.ID = S2.COUNTRY_ID
AND C2.NAME = 'Switzerland'
AND S1.PUBLISHER_ID = S2.PUBLISHER_ID
```

Result (3 rows out of 3 rows)

	Country name
1	Netherlands
2	Germany
3	France

Query 6. *In which languages (names of languages, without duplicates) have the publishers located in Switzerland published series located in countries other than, different from Switzerland.*

Description of the logic. First, we create a view that contains the country id of 'Switzerland', this will be useful later. Then, we apply several simple conditions to find the corresonding languages.

```sql
-- Get the ID of Switzerland
CREATE VIEW SWISS_ID AS
SELECT C.ID
FROM   STDDATA_COUNTRY C
WHERE  C.NAME = 'Switzerland';

SELECT DISTINCT L.NAME
FROM GCD_PUBLISHER P,
     GCD_SERIES    S,
     STDDATA_COUNTRY  C,
     STDDATA_LANGUAGE L,
     SWISS_ID
WHERE P.COUNTRY_ID = SWISS_ID.ID
```

```
        AND S.PUBLISHER_ID = P.ID
        AND S.COUNTRY_ID  != SWISS_ID.ID
        AND S.LANGUAGE_ID = L.ID;
```

Result (3 rows out of 3 rows)

	Country name
1	German
2	French
3	Dutch

Query 7. *What are the names of publishers from the Netherlands who are known to have operated/existed at least between 1995 and 2000 (both years are included in the range). Sort the publisher names in descending order (Z to A).*

Description of the logic. Firstly, the ambiguity between *operated/existed* must be removed. We will simply consider as valid those publishers

- who started publishing before 1995 but finished after 2000,
- who started publishing between 1995 and 2000.

That is, we can be sure that we count the publishers who did not publish between 1995 and 2000.

```
CREATE VIEW PUBLISHER_ID AS
SELECT DISTINCT P.ID
FROM GCD_PUBLISHER P
WHERE P.YEAR_BEGAN <= 1995 AND P.YEAR_ENDED >= 2000
   OR P.YEAR_BEGAN BETWEEN 1995 AND 2000;

SELECT P.NAME
FROM PUBLISHER_ID PUB,
     GCD_PUBLISHER P
WHERE P.ID = PUB.ID
  AND P.COUNTRY_ID =
      (SELECT C.ID
       FROM STDDATA_COUNTRY C
       WHERE C.NAME = 'Netherlands')
ORDER BY P.NAME DESC;
```

Result (20 first rows out of 77 rows)

	Total		Year began
1	Wolters-Noordhoff	11	Teken Mijn Verhaal
2	Wiebe J. Berkhuizen	12	Strip2000
3	Wavery Productions	13	Stivoro
4	VNU Tijdschriften	14	Stichting Zone 5300
5	van Speijk	15	Stichting Zeeschuim
6	Van Holkema & Warendorf	16	Stichting Propria Cures
7	Van de Berg	17	Stichting Aniway
8	Unie van Waterschappen	18	Stalactiet
9	Uitgeverij M	19	Safe Comics
10	Tekenstudio van Muylwijck	20	Rob Derks

Query 8. *What are the names of the publishers who published at least one series in a country different from where they are based? Sort the names in ascending order.*

Description of the logic. We simply match publishers and series by id and make sure that the series' country ID is different than that of the publisher's.

```
SELECT DISTINCT P.NAME
FROM     GCD_PUBLISHER P, GCD_SERIES s
WHERE     P.ID = S.PUBLISHER_ID
AND       P.COUNTRY_ID != S.COUNTRY_ID
ORDER BY P.NAME;
```

Result (20 rows out of 138 rows)

	Publisher name		Publisher name
1	(arabic text) [Teshkeel]	11	APComics
2	(asiatic text) [Square Enix]	12	Apocalypse
3	Abiogenesis Press	13	Associated Newspapers
4	Acme Press	14	Atomeka Press
5	Agência Portuguesa de Revistas	15	Azeko
6	Aircel Publishing	16	Bédé Adult
7	AK Press	17	Beeld Beeld
8	Alpen Publishers	18	Beta Publications
9	Amigo	19	Bich
10	Andina	20	Bloomsbury

Query 9. *Related to question 8: we are interested in the breakdown of the number of such occurrences of publishers/series per country. Order the output by the number of occurrences descending.*

Description of the logic. We find the result using the table from query 8 as a view.

We keep only tuples (publishers, country, publisher.name) that match the country id as well as the publisher name. Then, we group country name and we aggregate using `COUNT` on countries id.

```
CREATE VIEW QUERY_8 AS
SELECT DISTINCT P.NAME
FROM     GCD_PUBLISHER P, GCD_SERIES s
WHERE     P.ID = S.PUBLISHER_ID
AND       P.COUNTRY_ID != S.COUNTRY_ID
ORDER BY P.NAME;


SELECT C.NAME, COUNT(C.ID) AS Occurences
FROM GCD_PUBLISHER P, STDDATA_COUNTRY C, QUERY_8 Q
WHERE P.COUNTRY_ID = C.ID AND Q.NAME = P.NAME
GROUP BY C.NAME
ORDER BY COUNT(C.ID) DESC;
```

Result (20 rows out of 28 rows)

	Country name	Occurrences
1	United States	41
2	Canada	16
3	United Kingdom	14
4	Belgium	9
5	France	9
6	Switzerland	6
7	Netherlands	5
8	Australia	4
9	Spain	4
10	Germany	4
11	Denmark	4
12	Sweden	4
13	Italy	3
14	Chile	3
15	Japan	2
16	Norway	2
17	Brazil	2
18	Luxembourg	1
19	Portugal	1
20	Mexico	1

Query 10. *How many publishers have published series only in a country/countries different from the country they are in ?*

Description of the logic. We query on each Publisher and verify that they published at least one series in a country different than the one they are in by the second condition, and that there are no series published by this publisher in their country of origin by the first condition. We then aggregate the result using COUNT to obtain the number of such publishers.

```sql
SELECT COUNT(*) AS NUMBER_OF_PUBLISHERS
FROM GCD_PUBLISHER P
WHERE NOT EXISTS (SELECT P.ID FROM GCD_SERIES S1 WHERE S1.
    COUNTRY_ID=P.COUNTRY_ID AND S1.PUBLISHER_ID=P.ID)
AND EXISTS (SELECT P.ID FROM GCD_SERIES S2 WHERE S2.COUNTRY_ID!=P.
    COUNTRY_ID AND S2.PUBLISHER_ID=P.ID)
```

Result

	Number of publishers
1	29

Chapter 3

More and more queries

3.1 Query implementation

Query 1. *For each of the 10 publishers that began publishing first (the first 10 based on the year – limit the top 10 in the query even if there are ties), display in how many languages each of them published series in.*

Description of logic. *First, we create a view that contains the 10 most publishers who started publishing first. Then we just match id between oldest publishers and series, we group by publisher name and we aggregate using `COUNT(S.LANGUAGE-ID)`.*

```
CREATE VIEW OLDEST_PUBLISHERS AS
SELECT DISTINCT P.ID, P.NAME
FROM GCD_PUBLISHER P
ORDER BY P.YEAR_BEGAN
FETCH FIRST 10 ROWS ONLY;

SELECT OLD_P.NAME, COUNT(S.LANGUAGE_ID) AS SERIE_LANG_COUNT
FROM  OLDEST_PUBLISHERS OLD_P, GCD_SERIES S
WHERE S.PUBLISHER_ID = OLD_P.ID
GROUP BY OLD_P.NAME;
```

Result (10 rows out of 10 rows)

	Publisher name	Series Language Count
1	Sdu Uitgevers	1
2	Exshaw	1
3	Turner	1
4	Schlütersche Verlagsgesellschaft	2
5	Editorial Ibis Lda. / Livraria Bertrand S.A.R.L.	3
6	Schwabe	1

7	Humphrey	3
8	Verlag C. H. Beck	1
9	Livraria Bertrand Lda.	5
10	Wilhelm Prym Werke	1

Query 2. *Display the publisher names that have published more than 500 series, along with the number of series. Show the result in descending order of the count of published series.*

Description of logic. This query is pretty straightforward, we simply match the publisher id with the serie publisher id and then group by publisher name with an aggregation with `COUNT(S.ID)`. We select only those who have published more than 500 thanks to the `HAVING` clause.

```
SELECT P.NAME, COUNT(P.NAME)
FROM   GCD_SERIES S, GCD_PUBLISHER P
WHERE  S.PUBLISHER_ID = P.ID
GROUP BY P.NAME
HAVING   COUNT(S.ID) > 500
ORDER BY COUNT(P.NAME) DESC;
```

Result (20 rows out of 24 rows)

	Publisher name	Series Count
1	Marvel	7628
2	DC	6993
3	Dark Horse	2741
4	Image	2080
5	Panini Deutschland	1811
6	Fantagraphics	1385
7	Egmont Ehapa	1364
8	IDW	1344
9	Carlsen Comics [DE]	1282
10	Hjemmet / Egmont	947
11	Semic	899
12	Planeta DeAgostini	801
13	Panini España	728
14	Viz	703
15	Tokyopop (de)	669
16	Western	653
17	Boom! Studios	584
18	Dynamite Entertainment	579
19	Malibu	572
20	Dupuis	562

Query 3. *We are interested in the brand group names with more than 100 indicia publishers under the brand group. Display the brand group name and the corresponding count.*

Description of logic. *For this query, we simply a do a full matching on id's and then we group by the brand group name using COUNT(G.NAME) as an aggregation operation. We also make sure that the count is greater than 100 to achieve the requirement of the query.*

```
SELECT G.NAME, COUNT(G.NAME)
FROM   GCD_BRAND_GROUP G, GCD_INDICIA_PUBLISHER IP, GCD_PUBLISHER
     P
WHERE  G.PUBLISHER_ID = P.ID AND IP.PUBLISHER_ID = P.ID
GROUP BY G.NAME
HAVING   COUNT(G.NAME) > 100
ORDER BY COUNT(G.NAME);
```

Result (20 rows out of 36 rows)

	Publisher name	Series Count
1	Actual Ender's Game	110
2	Spider-Man Group	110
3	Razorline	110
4	MAX Comics	110
5	Pumping Iron	110
6	Marvel Universe Fantastic Four Group	110
7	Thumbtack	110
8	Tsunami	110
9	New Universe	110
10	Heavy Hitters	110
11	Timely Comics	110
12	Ultimate	110
13	2099	110
14	Crossgen	110
15	Nelson	110
16	A Romance! Magazine	110
17	Marvel Knights	110
18	DBPRO	110
19	Curtis Magazines	110
20	Dupuis	110

Query 4. *Show the brand group names with the largest number of Belgian Indicia Publishers (Indicia Publisher's country at least partially matches "Belgium"). Show the Brand*

Group Names and the resulting largest number.

We had two interpretations for this query:

Interpretation 1:

Description of logic. *We simply match id's and for the country id we get the id of country name that partially match "Belgium" using a subquery and the* `LIKE` *operator.*

```
SELECT G.NAME, COUNT(G.NAME)
FROM   GCD_BRAND_GROUP G, GCD_INDICIA_PUBLISHER IP, GCD_PUBLISHER
     P
WHERE  G.PUBLISHER_ID = P.ID AND IP.PUBLISHER_ID = P.ID
       AND IP.COUNTRY_ID = (SELECT C.ID
                            FROM  STDDATA_COUNTRY C
                            WHERE C.NAME LIKE '%Belgium%')
GROUP BY G.NAME, G.ID
ORDER BY COUNT(G.NAME) DESC;
```

Result (20 rows out of 133 rows)

	Brand Group Name	Largest Belgian Indicia Publisher Count
1	Vrije Vlucht	11
2	Uitgeverij Dupuis	11
3	Mezzanine	11
4	Aire Libre	11
5	Dubbel Expresso Dupuis	11
6	Éditions Dupuis	11
7	Spotlight	11
8	Puceron	11
9	Collectie Vrolijke Vlucht	11
10	Ukje	11
11	Dupuis; Pizza Hut	11
12	Expresso Dupuis	11
13	Graton	11
14	RepéRages Dupuis	11
15	Dupuis	9
16	Le Lombard	9
17	Talent Uitgeverij	9
18	Polyptyque	9
19	Uitgeverij Oranje	9
20	verhalen en legenden	9

Interpretation 2:

Description of logic. *We first create a view of the number of Belgian Indicia Publishers per Publisher by matching the Publishers and their Indicia Publishers from countries in the database containing the keyword 'Belgium', grouping by the Publisher IDs and obtaining the COUNT per publisher ID. Then, we select all Brand Groups belonging to the Publisher(s) with the highest number of Belgian Indicia Publishers by verifying that their COUNT in the previous view is equal to the maximal COUNT in that view.*

```sql
CREATE VIEW BELGIAN_IPS_PER_PUBLISHER AS
SELECT P.ID, COUNT(IP.ID) AS BG_IPS
FROM GCD_PUBLISHER P,
     GCD_INDICIA_PUBLISHER IP
WHERE P.ID = IP.PUBLISHER_ID AND IP.COUNTRY_ID = (SELECT C.ID
                                FROM  STDDATA_COUNTRY C
                                WHERE C.NAME LIKE '%Belgium%')
GROUP BY P.ID;

SELECT BG.NAME, BGIP.BG_IPS FROM GCD_BRAND_GROUP BG,
    BELGIAN_IPS_PER_PUBLISHER BGIP
WHERE BG.PUBLISHER_ID = BGIP.ID AND BGIP.BG_IPS = (SELECT MAX(
    BGIP2.BG_IPS) FROM BELGIAN_IPS_PER_PUBLISHER BGIP2);
```

Result (15 rows out of 15 rows)

	Brand Group Name	Largest Belgian Indicia Publisher Count
1	Aire Libre	11
2	Collectie Vrolijke Vlucht	11
3	Dubbel Expresso Dupuis	11
4	Dupuis	11
5	Dupuis; Pizza Hut	11
6	Expresso Dupuis	11
7	Graton	11
8	Mezzanine	11
9	Puceron	11
10	RepéRages Dupuis	11
11	Spotlight	11
12	Uitgeverij Dupuis	11
13	Ukje	11
14	Vrije Vlucht	11
15	Éditions Dupuis	11

Query 5. *Find Indicia Publishers that have published at least 400 single-issue series. Display their names and the count (of single-issue series), and order the result by the count descending. Remember that issues may have multiple instances with the same indicia*

publisher and series, and single-issue series are the ones that have exactly one such pair-value (indicia-publisher, series) occurrence in the issues.

Description of logic. We first create a view retrieving all pairs of single-issues' IDs and their corresponding Indicia Publisher ID by verifying that there isn't another different Issue ID with the same Series and Indicia Publisher IDs. Then, from the list of all Indicia Publishers, we pair them with their issues from the SINGLE-ISSUE-ID view and group them by the Indicia Publisher ID to get the number of single issues per Indicia Publisher, we then put the constraint of displaying only the ones with a count higher than 400 by the HAVING clause and set the ordering to be descending based on the Single-Issue ID counts.

```sql
CREATE VIEW SINGLE_ISSUE_ID AS
SELECT I1.ID, I1.INDICIA_PUBLISHER_ID FROM GCD_ISSUE I1
WHERE NOT EXISTS (SELECT I2.ID FROM GCD_ISSUE I2 WHERE I1.ID !=
    I2.ID AND I1.SERIES_ID=I2.SERIES_ID AND I1.
    INDICIA_PUBLISHER_ID= I2.INDICIA_PUBLISHER_ID);

SELECT IP.NAME AS INDICIA_PUBLISHER_NAME, COUNT(SIID.ID) AS
    SINGLE_ISSUE_SERIES_COUNT FROM GCD_INDICIA_PUBLISHER IP,
    SINGLE_ISSUE_ID SIID
WHERE SIID.INDICIA_PUBLISHER_ID = IP.ID
GROUP BY IP.NAME
HAVING COUNT(SIID.ID)>400
ORDER BY COUNT(SIID.ID) DESC;
```

Result (6 rows out of 6 rows)

	Indicia Publisher Name	Single Issue Series Count
1	DC Comics	2769
2	Marvel Worldwide Inc.	1040
3	Marvel Comics	798
4	Marvel Publishing Inc.	604
5	Image Comics Inc.	483
6	Dark Horse Comics Inc.	421

Query 6. *What is the most reprinted story for an issue? Display the issue ID, the story ID with the most reprints, and the corresponding reprint count. Sort the output based on the reprint count, and display the top 5 results. Count only the immediate reprints (the stories that match the origin ID)*

Description of logic. We simply compare id and we make sure to use `R.ORIGIN-ID = S.ID` to count only immediate reprints (as specified). Once again, we use the `GROUP BY` and the `COUNT` operator to aggregate the results.

EPFL

```sql
SELECT I.ID AS Issue_ID, S.ID AS Story_ID, COUNT(S.ID) AS
    Number_of_reprints
FROM   GCD_STORY S, GCD_ISSUE I, GCD_STORY_REPRINT R
WHERE  R.ORIGIN_ID = S.ID AND S.ISSUE_ID = I.ID
GROUP BY I.ID, S.ID
ORDER BY COUNT(S.ID) DESC
FETCH FIRST 5 ROWS ONLY;
```

Result (5 rows out of 5 rows)

	Issue ID	Serie ID	Number of reprints
1	336676	1419829	90
2	17099	133435	52
3	555894	1804115	46
4	17099	133434	43
5	92510	363431	36

Query 7. *Which heroes are featured in stories (feature attribute) that have all three genres: humor, crime, and romance.*

Description of logic. *For a story to be categorized as all 3 genres, there needs to be, for each of these genres, one entry in the STORY_TO_GENRE relationship linking said genre to the story. Once we find these stories, we project the tuples to their FEATURE attribute.*

```sql
select distinct S.FEATURE
from GCD_STORY S,
     GCD_STORY_TO_GENRE S2G1, GCD_STORY_GENRE G1,
     GCD_STORY_TO_GENRE S2G2, GCD_STORY_GENRE G2,
     GCD_STORY_TO_GENRE S2G3, GCD_STORY_GENRE G3
where S.ID = S2G1.STORY_ID and S2G1.GENRE_ID = G1.ID and G1.GENRE
     like '%humor%'
and S.ID = S2G2.STORY_ID and S2G2.GENRE_ID = G2.ID and G2.GENRE
    like '%crime%'
and S.ID = S2G3.STORY_ID and S2G3.GENRE_ID = G3.ID and G3.GENRE
    like '%romance%'
```

Result

	Feature
1	Family Funnies
2	Dick Tracy

Query 8. *Considering the PUBLICATION-DATE column in GCD-ISSUE table – it was kept as a string due to having a variety of date formats. Write a SQL query to extract the years from the PUBLICATION-DATE column that has a 'DD/MM/YYYY' date format. Display the distinct years only once, in ascending order. Hint: You will find useful the Oracle documentation on TO-DATE function, extract function to get the particular part/value from the date, as well as using the default null on conversion error in to-date function to handle formatting issues that happen since other date formats exist in the data and for some the matching will fail therefore instead of failing to return a null value.*

Description of logic. We apply the hint, and we use a view to store all converted dates, that is the view can contains NULL values resulting of conversion error. Then we just filter converted dates to get years with the `EXTRACT` function.

```sql
CREATE VIEW DATE_CONVERSION AS
SELECT TO_DATE(I.PUBLICATION_DATE DEFAULT NULL ON CONVERSION
    ERROR, 'DD/MM/YYYY') AS CONVERTED
FROM   GCD_ISSUE I;

SELECT DISTINCT EXTRACT(YEAR FROM D.CONVERTED)
FROM   DATE_CONVERSION D
WHERE  D.CONVERTED IS NOT NULL
ORDER BY EXTRACT(YEAR FROM D.CONVERTED);
```

Result (20 rows out of 129 rows)

	Year		Year
1	15	11	1832
2	22	12	1833
3	56	13	1834
4	57	14	1835
5	71	15	1838
6	82	16	1839
7	88	17	1840
8	91	18	1841
9	1830	19	1842
10	1831	20	1843

Query 9. *Related to question 8: we are interested in the counts of years of the publication date field of GCD-ISSUE table in different formats. To limit the scope of this question, write a SQL query that returns the count of years in given formats, in this order: 'DD/MM/YYYY', 'MM/DD/YYYY', 'MONTH YYYY'. Return the result in a single line, with 3 columns (one for each format), and a single row (only one value, the count, per column). Hint: You will find useful the Oracle documentation on TO-DATE function, as well as using the default null on conversion error in to-date function to handle*

the formatting errors. Note that COUNT(*) returns all the rows, including null values, while COUNT(with-specified-column) does not count null values - so you can omit the null check in the where clause in that case, if sufficient as a check. Recall that you can use subqueries to simulate multiple input tables to achieve the required output.

Description of logic. We follow the same logic that in query 8. We convert and apply the `to-date` function and then use `extract` and `count` to obtain the count of years.

```
SELECT * FROM
    (SELECT DISTINCT COUNT(FORMAT_1) FROM
        (SELECT EXTRACT(YEAR FROM
                TO_DATE(I.PUBLICATION_DATE DEFAULT NULL ON
    CONVERSION ERROR,'dd/mm/yyyy'))
                AS FORMAT_1 FROM GCD_ISSUE I)),
    (SELECT DISTINCT COUNT(FORMAT_2) FROM
        (SELECT EXTRACT(YEAR FROM
                TO_DATE(I.PUBLICATION_DATE DEFAULT NULL ON
    CONVERSION ERROR,'mm/dd/yyyy'))
                AS FORMAT_2 FROM GCD_ISSUE I)),
    (SELECT DISTINCT COUNT(FORMAT_3) FROM
        (SELECT EXTRACT(YEAR FROM
                TO_DATE(I.PUBLICATION_DATE DEFAULT NULL ON
    CONVERSION ERROR,'month yyyy'))
                AS FORMAT_3 FROM GCD_ISSUE I));
```

Result

	Count(dd/mm/yyyy)	Count(mm/dd/yyyy)	Count(month yyyy)
1	3146	78	5413

Query 10. *Finally, related to the date format and extraction, display the total number of issues per year between 1965 and 1975, including both years in the result. The year should be taken from PUBLICATION-DATE, with MONTH YYYY format. Display the results by years ascending (1965 to 1975).*

Description of logic. *We follow more or less the same logic as for the two previous queries. We start by converting and extracting the years from the table of issues. At the same time, we retrieve the IDs of the issues. Finally, we sort by year to keep only the issues that were published between 1965 and 1975 and aggregate using the* `COUNT` *function*

```
CREATE VIEW DATE_CONVERSION_10 AS
SELECT TO_DATE(I.PUBLICATION_DATE DEFAULT NULL ON CONVERSION
    ERROR, 'MONTH YYYY') AS CONVERTED, I.ID
FROM   GCD_ISSUE I;

```

```sql
CREATE VIEW YEARS AS
SELECT DISTINCT EXTRACT(YEAR FROM D.CONVERTED) AS YEAR, D.ID
FROM   DATE_CONVERSION_10 D
WHERE  D.CONVERTED IS NOT NULL
ORDER BY EXTRACT(YEAR FROM D.CONVERTED);

SELECT COUNT(Y.ID), Y.YEAR
FROM YEARS Y
WHERE Y.YEAR BETWEEN 1965 AND 1975
GROUP BY Y.YEAR
ORDER BY Y.YEAR;
```

Result (11 rows out of 11 rows)

	Count per year	Year
1	21	1965
2	29	1966
3	36	1967
4	38	1968
5	88	1969
6	49	1970
7	39	1971
8	44	1972
9	59	1973
10	50	1974
11	68	1975

Query 11. *What are the titles of the stories that have been reprinted at least 30 times? Print the reprint count along with the title. Order the output by the reprint count descending (higher to lower).*

Description of logic. Similar to query 6. We match id correctly to count only immediate reprints. Then we use a `GROUP BY` and `HAVING` as well as `COUNT` to only keep stories that have been reprinted at least 30 times. Finally, we order the result descending.

```sql
SELECT S.TITLE, COUNT(R.ORIGIN_ID)
FROM GCD_STORY_REPRINT R, GCD_STORY S
WHERE R.ORIGIN_ID = S.ID
GROUP BY R.ORIGIN_ID, S.TITLE
HAVING COUNT(R.ORIGIN_ID) >= 30
ORDER BY COUNT(R.ORIGIN_ID) DESC;
```

Result (14 rows out of 14 rows)

	Story title	Reprint count
1	Spaghetti Brothers - historien om søsknene Centobucchi	90
2	Spider-Man!	52
3	Verso l'ignoto	46
4	Spider Man	43
5	NULL	36
6	The Fantastic Four!	35
7	Le nain de Corneloup	34
8	Le sortilège du haricot	34
9	Flash of Two Worlds!	33
10	Spider-Man	33
11	Spider-Man vs. The Chameleon!	32
12	Is He Man or Monster or... Is He Both!	32
13	Out of the Darksome Hills	31
14	The Chameleon Strikes!	30

Query 12. *We are interested in the top 10 countries based on the publisher(s) with the longest time publishing, meaning the longest duration between the year they began and ended publishing. Take into consideration the year range between and including 1600 for the year began and 2020 for the year ended (both included). Display for those countries the longest duration of publisher publishing, as well as the average duration of publishing. Order the top 10 countries by the maximum duration descending (highest first)*

Description of the logic. We solve this query but using views. We start by retrieving all publishers who started and finished publishing between 1600 and 2020. Then, among all these publishers, we retrieve their country of publication and the publisher's length of service (`P.YEARENDED - P.YEARBEGAN`). Finally, we group the countries by taking the maximum duration of exercise thanks to the `MAX` function and the average duration of exercise thanks to the `AVG` function

```sql
SELECT *
FROM GCD_PUBLISHER P
WHERE P.YEAR_BEGAN BETWEEN 1600 AND 2020
AND   P.YEAR_ENDED BETWEEN 1600 AND 2020;

CREATE VIEW COUNTRY_AND_DIFFERENCE AS
SELECT P.ID, C.NAME, (P.YEAR_ENDED - P.YEAR_BEGAN) AS
    PUBLISHING_TIME
FROM PUBLISHER_IN_YEAR_RANGE P, STDDATA_COUNTRY C
WHERE C.ID = P.COUNTRY_ID;

SELECT CAD.NAME, MAX(PUBLISHING_TIME),
                 ROUND(AVG(PUBLISHING_TIME), 1)
FROM COUNTRY_AND_DIFFERENCE CAD
GROUP BY CAD.NAME
ORDER BY MAX(PUBLISHING_TIME) DESC
```

```
FETCH FIRST 10 ROWS ONLY;
```

Result (10 rows out of 10 rows)

	Country name	Publishing time	Average publishing time
1	Switzerland	206	47.3
2	Germany	194	18.2
3	Norway	165	22
4	Australia	159	15.9
5	United Kingdom	156	5.4
6	United States	150	4.1
7	France	150	15.9
8	Belgium	129	9.2
9	Netherlands	114	7.3
10	Spain	76	15.9

Query 13. *List all Marvel heroes that appear in Marvel-DC story crossovers. Marvel and DC are considered to be (parts of) Indicia Publisher names, and the heroes are described in the story feature attribute. Therefore, we are interested in heroes that appear in purely Marvel stories (without DC) AND in the ones that have both Marvel and DC in the corresponding Indicia publisher name. When comparing strings, you must transform all the strings to lowercase (there is a corresponding function to use – check the documentation) and use partial string matches – so for Marvel and DC you need to find Indicia Publishers that partially match 'marvel' and 'dc' anywhere in the string. Display distinct, lowercase hero/feature strings – but do not manually deduplicate them further. Make sure that in the final result once you combine purely Marvel heroes with crossover heroes, you match their names partially as well!*

Description of the logic. First we look for all heroes that appear in Marvel only stories, for that we retrieve all issues published by an indicia publisher whose name partially matches 'marvel' but not 'dc', these are the marvel-only published issues. We perform a join over the Marvel-only issues and the Story relation, with respect to the Issue ID attribute, and we retrieve the Feature attribute. Then we look for all heroes that appear in Marvel and DC crossovers, for that we retrieve all issues that were published by an indicia publisher whose name partially matches both 'marvel' and 'dc', once that's done, we perform a join over these issues and the Story table, and we retrieve the Feature attribute. Finally we intersect the Marvel-only heroes with the DC-Marvel crossover heroes. To do that, we retrieve all Marvel-only heroes which appear in the list of crossover Heroes, i.e. all Marvel-only Feature attributes that are partially matched by Crossover Feature attributes.

```
create view MARVEL as
```

```sql
select distinct LOWER(S.FEATURE) as FEATURE
from GCD_STORY S, GCD_ISSUE I, GCD_INDICIA_PUBLISHER P
where S.ISSUE_ID = I.ID
and I.INDICIA_PUBLISHER_ID = P.ID
and LOWER(P.NAME) like '%marvel%'
and LOWER(P.NAME) not like '%dc%'
and LOWER(S.FEATURE) is not null;

create view CROSSOVER as
select distinct LOWER(S.FEATURE) as FEATURE
from GCD_STORY S, GCD_ISSUE I, GCD_INDICIA_PUBLISHER P
where S.ISSUE_ID = I.ID
and I.INDICIA_PUBLISHER_ID = P.ID
and LOWER(P.NAME) like '%marvel%'
and LOWER(P.NAME) like '%dc%';

select distinct MARVEL.FEATURE
from MARVEL, CROSSOVER
where CROSSOVER.FEATURE like concat(concat('%', MARVEL.FEATURE),
    '%');
```

Result (16 rows out of 16 rows)

	Crossover feature heroes
1	batman
2	wizard of oz
3	hulk
4	x-men
5	kitty pryde
6	superman
7	doctor octopus
8	spider-man
9	wizard
10	superman; spider-man
11	titan
12	it
13	lex luthor; doctor octopus
14	superman and spider-man
15	oz
16	thor

Query 14. *For every country that has at least 200 publishers (based on Publisher Country location, they don't need to have published any series, and the series they published can be in other countries than the Publisher Country), print the top 2 publishers by the number of series published (in any country, do not enforce that the series country is the same as publisher country).*

Description of logic. First, we partition the publishers by their country ID attribute, and for each subset of the partition, we rank the publishers with respect to the number of series they published. This ranking is done on line 04: for each subset of the partition, we order the tuples in descending order with respect to the count of series published, and then assign a row number. We then intersect this partition with countries that have more than 200 publishers.

```
01 |  create view RANKED_PUBLISHERS as
02 |  select P.COUNTRY_ID as COUNTRY_ID,
03 |         P.NAME as NAME,
04 |         ROW_NUMBER() over (partition by P.COUNTRY_ID order by
         COUNT(S.ID) desc) as ROW_NUMBER
05 |  from GCD_PUBLISHER P, GCD_SERIES S
06 |  where S.PUBLISHER_ID = P.ID
07 |  group by P.COUNTRY_ID, P.NAME;
08 |
09 |  create view RANKED_COUNTRIES as
10 |  select C.ID, C.NAME
11 |  from STDDATA_COUNTRY C, GCD_PUBLISHER P
12 |  where C.ID = P.COUNTRY_ID
13 |  group by C.ID, C.NAME
14 |  having count(P.ID) > 200;
15 |
16 |  select C.NAME, P.NAME
17 |  from RANKED_PUBLISHERS P, RANKED_COUNTRIES C
18 |  where P.COUNTRY_ID = C.ID
19 |  and P.ROW_NUMBER < 3;
```

Result (20 rows out of 20 rows)

	Country Name	Publisher Name
1	Canada	Drawn & Quarterly
2	Canada	Bell Features
3	Germany	Panini Deutschland
4	Germany	Egmont Ehapa
5	Denmark	Interpresse
6	Denmark	Forlaget Carlsen
7	France	Panini France
8	France	Dargaud éditions
9	United Kingdom	IPC
10	United Kingdom	Titan
11	Italy	Arnoldo Mondadori Editore
12	Italy	Sergio Bonelli Editore
13	Netherlands	Arboris
14	Netherlands	Oberon
15	Norway	Hjemmet / Egmont
16	Norway	Semic
17	Sweden	Semic

18	Sweden	Egmont
19	United States	Marvel
20	United States	DC

3.2 Query optimization

We present in this section 3 query optimizations that we find interesting.

3.2.1 Optimization of query 3

- Initial running time (I/O) : 62
- Improved running time (I/O) : 24
- Explain the improvement : We added three indexes that allow us to decrease the I/O cost by 61% because they allow us to perform fast full scans which is less costly.

```sql
CREATE INDEX gcd_brand_group_idx_publisher_name_id ON
    GCD_BRAND_GROUP (PUBLISHER_ID,NAME,ID);
CREATE INDEX gcd_indicia_pub_idx_publisher_id ON
    GCD_INDICIA_PUBLISHER (PUBLISHER_ID);
CREATE INDEX gcd_publisher_idx_id ON GCD_PUBLISHER (ID);
```

- Initial plan : please see appendix A.
- Improved plan : please see appendix A.

3.2.2 Optimization of query 4

- Initial running time (I/O) : 64
- Improved running time (I/O) : 36
- Explain the improvement : Using the same three indexes that we used for query 3 we have decreased the I/O cost by 43.75% because they also allow us to perform fast full scans which is less costly.

```sql
CREATE INDEX gcd_brand_group_idx_publisher_name_id ON
    GCD_BRAND_GROUP (PUBLISHER_ID,NAME,ID);
CREATE INDEX gcd_indicia_publis_idx_publisher_id ON
    GCD_INDICIA_PUBLISHER (PUBLISHER_ID);
CREATE INDEX gcd_publisher_idx_id ON GCD_PUBLISHER (ID);
```

- Initial plan : please see appendix B.
- Improved plan : please see appendix B.

3.2.3 Optimization of query 6

- Initial running time (I/O) : 25995
- Improved running time (I/O) : 12096
- Explain the improvement : We added three indexes that allow us to decrease the I/O cost by 43.75% because they also allow us to perform fast full scans which is less costly, specially during the hash join in the groupy by operator, where we have roughly decreased this cost by 4.

```sql
CREATE INDEX gcd_issue_idx_id ON GCD_ISSUE(ID);
CREATE INDEX gcd_story_idx_id_issue_id ON GCD_STORY (ID,
    ISSUE_ID);
CREATE INDEX gcd_story_reprint_idx_origin_id ON
    GCD_STORY_REPRINT (ORIGIN_ID);
```

- Initial plan : please see appendix C.
- Improved plan : please see appendix C.

3.3 General comments

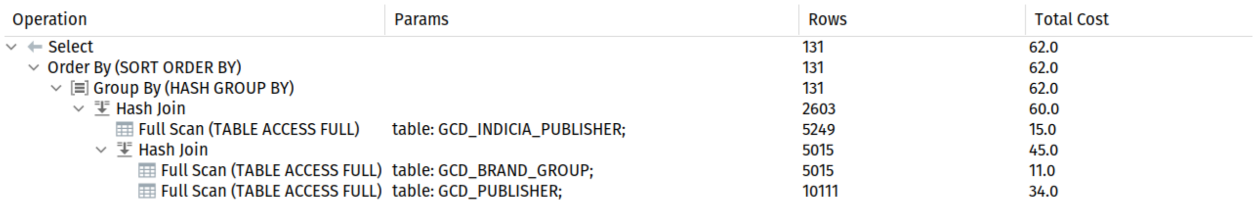
3.3.1 Work allocation between team members

Francois, Souleyman and Fouad all wrote the 24 queries. Francois did the optimizations. Souleyman redid the ER model (we lost points on this in the first deliverable) and Fouad did a complete re-reading of the report.

Appendix A

Plans for query 3

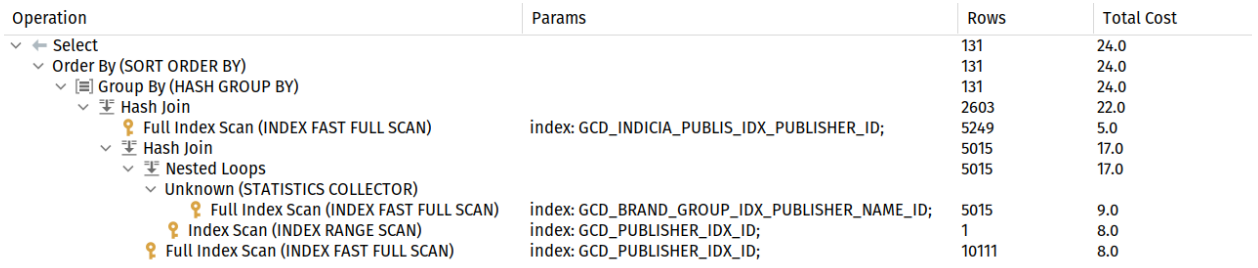
Initial plan :



Operation	Params	Rows	Total Cost
Select		131	62.0
Order By (SORT ORDER BY)		131	62.0
Group By (HASH GROUP BY)		131	62.0
Hash Join		2603	60.0
Full Scan (TABLE ACCESS FULL)	table: GCD_INDICIA_PUBLISHER;	5249	15.0
Hash Join		5015	45.0
Full Scan (TABLE ACCESS FULL)	table: GCD_BRAND_GROUP;	5015	11.0
Full Scan (TABLE ACCESS FULL)	table: GCD_PUBLISHER;	10111	34.0

Figure A.1: Initial plan of part3-query4

Improved plan :



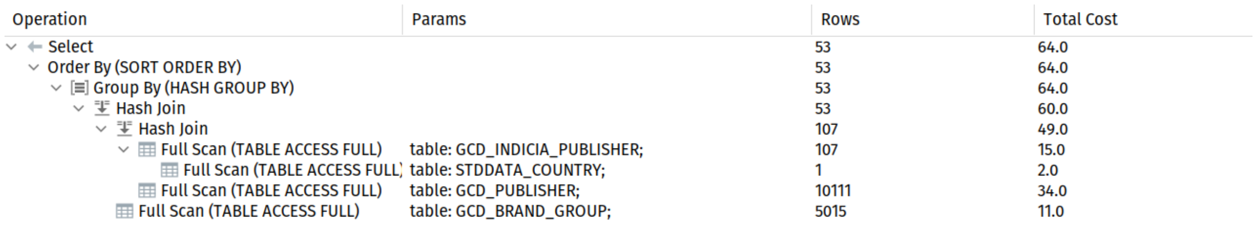
Operation	Params	Rows	Total Cost
Select		131	24.0
Order By (SORT ORDER BY)		131	24.0
Group By (HASH GROUP BY)		131	24.0
Hash Join		2603	22.0
Full Index Scan (INDEX FAST FULL SCAN)	index: GCD_INDICIA_PUBLIS_IDX_PUBLISHER_ID;	5249	5.0
Hash Join		5015	17.0
Nested Loops		5015	17.0
Unknown (STATISTICS COLLECTOR)			
Full Index Scan (INDEX FAST FULL SCAN)	index: GCD_BRAND_GROUP_IDX_PUBLISHER_NAME_ID;	5015	9.0
Index Scan (INDEX RANGE SCAN)	index: GCD_PUBLISHER_IDX_ID;	1	8.0
Full Index Scan (INDEX FAST FULL SCAN)	index: GCD_PUBLISHER_IDX_ID;	10111	8.0

Figure A.2: Initial plan of part3-query4

Appendix B

Plans for query 4

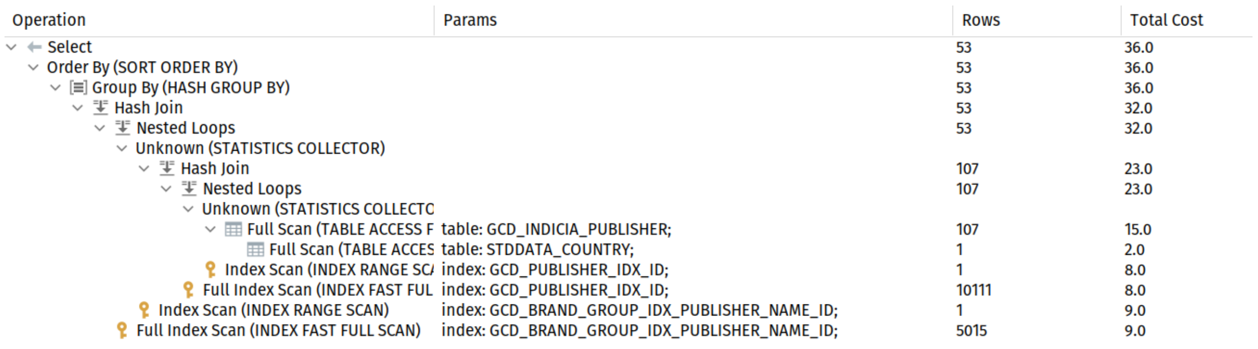
Initial plan :



Operation	Params	Rows	Total Cost
Select		53	64.0
Order By (SORT ORDER BY)		53	64.0
Group By (HASH GROUP BY)		53	64.0
Hash Join		53	60.0
Hash Join		107	49.0
Full Scan (TABLE ACCESS FULL)	table: GCD_INDICIA_PUBLISHER;	107	15.0
Full Scan (TABLE ACCESS FULL)	table: STDDATA_COUNTRY;	1	2.0
Full Scan (TABLE ACCESS FULL)	table: GCD_PUBLISHER;	10111	34.0
Full Scan (TABLE ACCESS FULL)	table: GCD_BRAND_GROUP;	5015	11.0

Figure B.1: Initial plan of part3-query4

Improved plan :



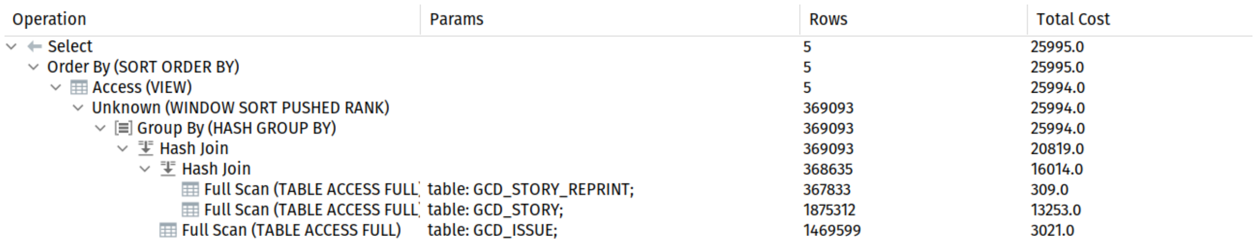
Operation	Params	Rows	Total Cost
Select		53	36.0
Order By (SORT ORDER BY)		53	36.0
Group By (HASH GROUP BY)		53	36.0
Hash Join		53	32.0
Nested Loops		53	32.0
Unknown (STATISTICS COLLECTOR)			
Hash Join		107	23.0
Nested Loops		107	23.0
Unknown (STATISTICS COLLECTC			
Full Scan (TABLE ACCESS F	table: GCD_INDICIA_PUBLISHER;	107	15.0
Full Scan (TABLE ACCES	table: STDDATA_COUNTRY;	1	2.0
Index Scan (INDEX RANGE SC/	index: GCD_PUBLISHER_IDX_ID;	1	8.0
Full Index Scan (INDEX FAST FUL	index: GCD_PUBLISHER_IDX_ID;	10111	8.0
Index Scan (INDEX RANGE SCAN)	index: GCD_BRAND_GROUP_IDX_PUBLISHER_NAME_ID;	1	9.0
Full Index Scan (INDEX FAST FULL SCAN)	index: GCD_BRAND_GROUP_IDX_PUBLISHER_NAME_ID;	5015	9.0

Figure B.2: Initial plan of part3-query4

Appendix C

Plans for query 6

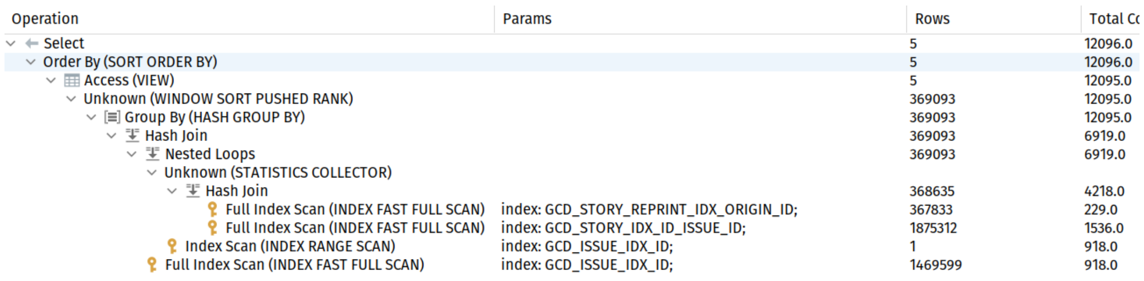
Initial plan :



Operation	Params	Rows	Total Cost
Select		5	25995.0
Order By (SORT ORDER BY)		5	25995.0
Access (VIEW)		5	25994.0
Unknown (WINDOW SORT PUSHED RANK)		369093	25994.0
Group By (HASH GROUP BY)		369093	25994.0
Hash Join		369093	20819.0
Hash Join		368635	16014.0
Full Scan (TABLE ACCESS FULL	table: GCD_STORY_REPRINT;	367833	309.0
Full Scan (TABLE ACCESS FULL	table: GCD_STORY;	1875312	13253.0
Full Scan (TABLE ACCESS FULL)	table: GCD_ISSUE;	1469599	3021.0

Figure C.1: Initial plan of part3-query6

Improved plan :



Operation	Params	Rows	Total C
Select		5	12096.0
Order By (SORT ORDER BY)		5	12096.0
Access (VIEW)		5	12095.0
Unknown (WINDOW SORT PUSHED RANK)		369093	12095.0
Group By (HASH GROUP BY)		369093	12095.0
Hash Join		369093	6919.0
Nested Loops		369093	6919.0
Unknown (STATISTICS COLLECTOR)			
Hash Join		368635	4218.0
Full Index Scan (INDEX FAST FULL SCAN)	index: GCD_STORY_REPRINT_IDX_ORIGIN_ID;	367833	229.0
Full Index Scan (INDEX FAST FULL SCAN)	index: GCD_STORY_IDX_ID_ISSUE_ID;	1875312	1536.0
Index Scan (INDEX RANGE SCAN)	index: GCD_ISSUE_IDX_ID;	1	918.0
Full Index Scan (INDEX FAST FULL SCAN)	index: GCD_ISSUE_IDX_ID;	1469599	918.0

Figure C.2: Initial plan of part3-query6