Volume 5, No. 3, March 2014
Journal of Global Research in Computer Science
ISSN-2229-371X
REVIEW ARTICLE
Available Online at www.jgrcs.info

Prime Factor Attribute on Entity to increase the performance of read-only statements in Many-to-Many relationship

M.S. Vasantharaju
Software Enginner
Email:vasa.v03@gmail.com

Abstract: It is very common to encounter many-to-many relationship where an entity will resolve into a Lookup or Reference Table while designing relational database. And using an associative entity [4] is not always a better solution as it involves a separate table and joins reduce the performance of the SQL-Data statements. This paper discusses about how adding a prime factor attribute on an entity instead of adding an associative entity improves the performance of SQL-Data statements if at least one entity in man-to-many relationship will resolve into Lookup table and number of entity-occurrence is less than 20 for best optimization

***Keywords*:** Many-to-Many, Database Design, Performance Tuning, Mathematical-based Data Modelling

INTRODUCTION

Database Design plays a critical role in developing software, as any risk caused by a flaw in Database Design persistent till the release of software involves huge cost to mitigate. A considerable amount of time and knowledge is involved in creating a Database Model. This paper will discuss about an alternate solution to specific problem that is common in relational data model thereby increasing scalability and performance of database. The solution is completely based on prime numbers and prime factors and nothing more though it needs a shift in paradigm and tries it. Again the solution is optimized for a very specific problem.

MANY-TO-MANY RELATIONSHIP

It is a common type of cardinality that refers to the relationship between two entities where two entities are children of each other. Let us take a classic example where ***students enroll for classes***, we employ ER model to better explain relationship between Student and Course

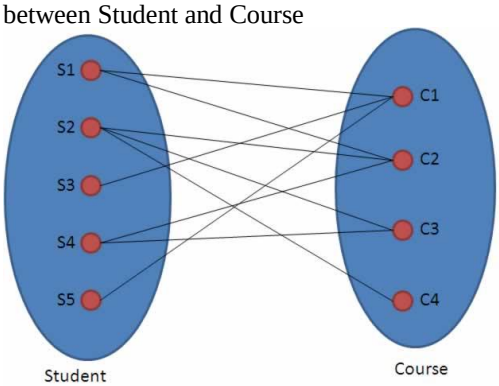

Figure 1. Many-to-Many Relation Model

COMMON APPROACH

Since it is not possible to express Many-to-Many relationship in logical data model an associative entity is introduced such a way that the associative entity will have parent-children relationship with each other entity. As in previous example a new entity called Enroll which will become a child entity for both Student and Course Entity.

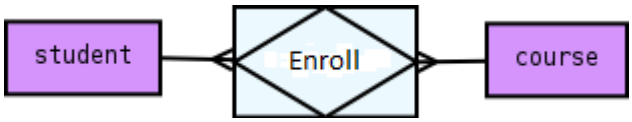

Figure 2.ER Modlel for M:N reolved with an Associative Enity[4]

PRACTICAL PROBLEM

Let us take a more practical situation for M:N relation problem. Assume that we are building a simple web application which requires authentication to login and authorization to access control resources. And focus on authorization framework where User and Permission are entities which have M:N relationship. A Junction Table called User-Permission will be used to resolve M: N and subsequent insertion of Users will make an entry in UserPermission table to map UserId and PermId.

User	
UserId	**UserName**
1	EndUser
2	Guest
3	Admin

Table 1. User Table : Set(U)

Permission	
PermId	**PermName**
1	Dashboard_View
2	Add_Or_Update_Users
3	Delete_Users
4	Admin_Screen_View
5	Manage_Configuration

Table 2. Permission Table : Set(P)

UserPermission	
UserId	**PermId**
1	1
1	2
1	3
2	1

Table 3. Junction Table : Set(UP)

Now, to determine what permissions are granted for a user after authentication we need to perform an operation, [6].

$$R=\pi_{PermName}(\ (U \bowtie_{UserId = UserId \wedge UserName=?} UP) \bowtie_{PermId = PermId} P\ )$$

It is evident that the operation involves multiple joins and also growth of UserPermission table is directly proportional to relation between User and Permission tables.

PRIME FACTOR APPROACH

The basic idea behind the approach is that the prime factors of a positive integer is always a unique set of prime numbers.
For example

$$30 = 2 \times 3 \times 5$$
$$1155 = 3 \times 5 \times 7 \times 11$$

So we leverage this idea to represent the relationship by calculating the product of prime factors. This approach is a three step process and discussed in detail further.

Adding a Surrogate key

Prime Factor approach involves adding a surrogate key which is an unique prime number greater than one to Permissions Lookup Table, best way is to start with 2 and progress along 3,5,7,11 and further for each tuple.

Permissions	
PrimeSurr	**PermName**
2	Dashboard_View
3	Add_Or_Update_Users
5	Delete_Users
7	Admin_Screen_View
11	Manage_Configuration

Table 3. Permission Table with Surrogate Key

Calculate Prime Factor

Instead of adding a Junction table a prime factor attribute can be added to User table called PermSet.
For every insertion or updating the relation of User table calculate the prime factor which is a product of surrogate keys for each Permission

User		
UserId	**UserName**	**PermSet**
1	EndUser	30
2	Guest	2
3	Admin	1155

Table 5. User Table with PermSet column

So PermSet of 30 includes Dashboard_View, Add_Or_Update_Users and Delete_Users permissions.
Now the Database Model will become

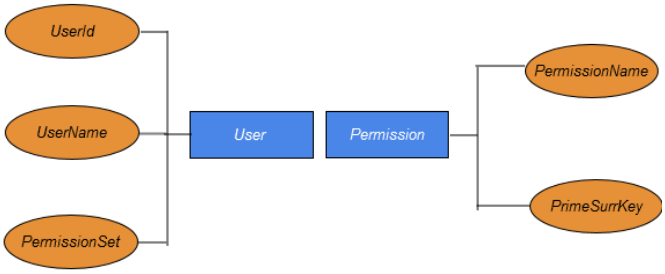

Figure 1. ER Model of M:N after Prime Factor Approach

Decode

There are many ways to decode the prime factor coposite in to entity-occurrences . Considering the previous example let us take a simple solution wherein a function will be created in database server side which accepts a composite number C and returns entity occrurences after performing an operation.

$$R_1=\pi_{PermName}(\sigma_{C\ \%\ PrimeSurr\ =0} P)$$

ANALYSIS

The operation for getting Permission for a User will now become,

$$R=\pi_{PermName}(\ U \bowtie_{UserName=? \text{ and } PermSet\ \%\ PrimeSurr\ =0} P\ )$$

A simple sub query or a sql with join like above can also be used instead of using a separate function. Now it is evident that we have reduced a joint which will result in a significant gain in performance if the volume of data is huge.

Pros of Prime Factor Method

1. Scalable solution as Junction Table is no more used. Database growth is independent on relationship between Entities
2. Can be used as a De-Normalization tool if we need to increase performance if analytical reads supersedes insertion
3. Increased performance of SELECT queries
4. Much simpler to implement.
5. Size of Data Representation is reduced, so PermSet number is enough to represent all the permissions of a User.

Cons of Prime Factor Method:

1. Optimized for very small number of entity-occurrence this is huge trade off which needs a research to arrive a max for entity-occurrence which involves features data types etc. offered by Database Management Systems.
2. Prime factor calculation overhead is involved with each insertion so this is preferable only if read supersedes insertion.
3. Performance of read statements depends on algorithm used to decode Prime Factor to entity occurrence.

CONCLUSION AND RECOMMENDATIONS

Acknowledging the fact that de-normalization is more prevalent for sql performance consideration Prime Factor Attribute technique can be used as a tool to de-normalize data. In fact the same motivation led to the development of this technique. Caching the lookup table and implementing an algorithm to decode Prime Factor to entity-occurrence in application server

side will help the performance further. Also cache the Prime Factor – Entity Occurrences map using Least Recently Used mechanism to enhance the performance of the system.

REFERENCES

[1] (2000-10-18) Mike Gahan , "An introduction to databases", Celtic Inscribed Stones Project (CISP) UCL , http://www.ucl.ac.uk/archaeology/cisp/database/manual/node1.html .

[2] "Conceptual Database Design - Entity Relationship(ER) Modeling", http://www.careerbless.com/db/rdbms/c1/design.php?sortOption=O .

[3] (2014) Many-to-many (data model), Creative Commons Attribution/Share-Alike License , http://en.wikipedia.org/wiki/Many-to-many_%28data_model%29 **(Article in Wikipedia**).

[4] (2013) Associative entity, Creative Commons Attribution/Share-Alike License , http://en.wikipedia.org/wiki/Associative_Entities **(Article in Wikipedia)**.

[5] (2012) Junction table, Creative Commons Attribution/Share-Alike License , http://en.wikipedia.org/wiki/Junction_table **(Article in Wikipedia)**.

[6] Thomas Padron-McCarthy,"Lecture Notes: Relational Algebra",**Section of a web course on databases**, June 2008, http://www.databasteknik.se/webbkursen/relalg-lecture.

SHORT BIODATA OF THE AUTHOR

M.S.Vasantharaju graduated from National Institue of Technology in the year 2007 under *Metallurgy and Materials Engineering* discipline and currently employed as Software Development Engineer in a security technology company and posses expertise in the area of software development , defensive programming and web appplication vulnerabilities and risks.

