Referential Integrity Is Important For Databases

Michael Blaha (blaha@computer.org)
Modelsoft Consulting Corp
www.modelsoftcorp.com

Abstract

1. Introduction

Referential integrity is a database constraint that ensures that references between data are indeed valid and intact. Referential integrity is a fundamental principle of database theory and arises from the notion that a database should not only store data, but should actively seek to ensure its quality. Here are some additional definitions that we found on the Web.

- "Referential integrity in a relational database is consistency between coupled tables. Referential integrity is usually enforced by the combination of a primary key and a foreign key. For referential integrity to hold, any field in a table that is declared a foreign key can contain only values from a parent table's primary key field..." [5]
- "[Referential integrity is] a feature provided by relational database management systems (RDBMS's) that prevents users or applications from entering inconsistent data. Most RDBMS's have various referential integrity rules that you can apply when you create a relationship between two tables." [6]
- "[Referential integrity is] a database management safeguard that ensures every foreign key matches a primary key. For example, customer numbers in a customer file are the primary keys, and customer numbers in the order file are the foreign keys. If a customer record is deleted, the order records must also be deleted; otherwise they are left without a primary reference. If the DBMS does not test for this, it must be programmed into the applications." [7]

There are many benefits of defining referential integrity in a database.

- **Improved data quality**. An obvious benefit is the boost to the quality of data that is stored in a database. There can still be errors, but at least data references are genuine and intact.
- **Faster development**. Referential integrity is declared. This is much more productive (one or two orders of magnitude) than writing custom programming code.
- **Fewer bugs**. The declarations of referential integrity are more concise than the equivalent programming code. In essence, such declarations reuse the tried and tested general-purpose code in a database engine, rather than redeveloping the same logic on a case-by-case basis.
- **Consistency across applications**. Referential integrity ensures the quality of data references across the multiple application programs that may access a database.

You will note that the definitions from the Web are expressed in terms of relational databases. However, the principle of referential integrity applies more broadly. Referential integrity applies to both relational and OO databases, as well as programming languages and modeling.

2. Referential Integrity and RDBMSs

The SQL syntax for defining referential integrity looks essentially like the following. The words in capital letters denote keywords. The brackets indicate optional parameters. The foreign key columns are in *table1* and the primary key (or other unique combination of columns) is in *table2*.

```
ALTER TABLE tableName1
ADD CONSTRAINT constraintName
FOREIGN KEY (columnList)
REFERENCES tableName2 [(columnList)]
[onDeleteAction] [onUpdateAction];
```

In SQL a foreign key can refer to any unique combination of columns in the referenced table. If the referenced column list is omitted, the foreign key refers to the primary key. The SQL standard [3] provides the following referential integrity actions for deletions.

- **Cascade**. The deletion of a record may cause the deletion of corresponding foreign-key records. For example, if you delete a company, you might also want to delete the company's history of addresses.
- **No action**. Alternatively, you may forbid the deletion of a record if there are dependent foreign-key records. For example, if you have sold products to a company, you might want to prevent deletion of the company record.
- **Set null**. The deletion of a record may cause the corresponding foreign keys to be set to null. For example, if there is an aircraft substitution on a flight, you may want

to nullify some seat assignments. (These passengers must then request other seat assignments.)

- **Set default**. You may set a foreign key to a default value instead of to null upon deletion of a record.

The SQL standard also provides these actions for updates. Support for the standard varies by product.

- **SQL Server**. In SQL Server 2000 a foreign key can reference a primary key or a *unique* combination of columns. There is no support for the *set null* and *set default* options. However, *cascade* and *no action* are fully supported and often suffice in practice. We also note that the syntax for *no action* behavior differs from the SQL standard. With SQL Server, the lack of a referential integrity action implies *no action*.
- **Oracle**. In Oracle 10g a foreign key can also reference a primary key or a *unique* combination of columns. Oracle supports delete actions but omits update actions. For delete actions the *cascade*, *no action*, and *set null* options are permitted. There is no support for the *set default* option. As with SQL Server, the syntax for *no action* differs from the SQL standard. The lack of a referential integrity action implies *no action*.
- **MySQL**. MySQL 5.0 has been enhanced to support referential integrity (the InnoDB engine). MySQL more fully supports the SQL standard than SQL Server and Oracle. It supports *on delete* and *on update* with all four referential integrity actions. MySQL requires that the foreign key and the referenced key both have database indexes—this is a good practice that we would advise anyway. The referenced columns can be a primary key or a *unique* combination of columns.
- **MS-Access**. MS-Access only partially supports SQL commands for defining referential integrity, but its graphical relationship tool is more capable. The graphical interface can define referential integrity as well as the *cascade* and *no action* referential integrity actions.

3. Referential Integrity and OO-DBMSs

The notion of referential integrity also applies to OO-DBMSs. There are relationships between objects that cause them to depend on each other and maintenance of these dependencies should not be left to application code.

Some OO-DBMSs support referential integrity. For example, ObjectStore implements referential integrity with dual pointers between a pair of objects that the DBMS engine keeps mutually consistent [1]. Thus ObjectStore preserves the style of programming languages—a pointer is just another instance variable buried within an object. However, it also honors the meaning of referential integrity by maintaining the dependencies between objects and avoiding dangling references.

ObjectStore calls these pairs of mutually consistent pointers *inverse members*. ObjectStore maintains the integrity of inverse members; when one member is updated, ObjectStore transparently updates the other. Similarly, when an object is deleted, ObjectStore automatically updates any inverse members to prevent dangling pointers.

4. Referential Integrity and Languages

With few exceptions programming languages overlook referential integrity. Instead they focus on encapsulation, that implementation details should be kept private to a class. The problem is that encapsulation is often compromised by the dependencies between objects. A mere value does not suffice for a reference to an object.

For example, an *Employment* object refers to a *Person* and a *Company*. It is not possible to deal with *Employment* in isolation. If a *Person* is deleted, the related *Employment* objects must also be deleted or there will be dangling references. An *Employment* object cannot be created without a *Person* and a *Company*.

It is possible for a programming language to support referential integrity as [4] demonstrates. At GE R&D Rumbaugh implemented an object-oriented programming language (called DSM) with semantic support for dependencies between objects (called relations in the paper).

5. Referential Integrity and the UML

Referential integrity also arises with the UML via semantic support for associations. An ***association*** is a description of a group of links with common structure and common semantics. A ***link*** is a physical or conceptual connection among objects.

When constructing UML class models, it is important to represent the dependencies between objects explicitly with associations and not hide them as attributes. Conventional programming languages force you to degrade the associations, but at least you should be clear in your thinking. Then you can devise the best workaround within the limits of a language.

6. Is Referential Integrity Being Used?

Referential integrity is a prominent aspect of relational database theory. But is it really used in practice? We can answer this question because we have data on about 50 existing databases from our reverse engineering studies. [2]

In practice, referential integrity is seldom enforced. We know that it is not enforced in most programming code because programming languages lack a mechanism and make enforcement difficult. However, the problem is deeper than that. Most relational DBMS implementations also lack referential integrity. Even though the RDBMSs support the referential integrity mechanism, many developers fail to use it. From our reverse engineering studies we have found that 90% of relational database developers fail to use referential integrity in their applications. [2]

The consequences are unfortunate, yet we see them everyday—software that is buggy, performs slowly, and is difficult to extend. In a future article we will explore the implications further with the case study of the LDAP standard. LDAP did not incorporate referential integrity and there are unfortunate consequences of the omission.

7. Conclusion

Usually when someone mentions the term “referential integrity” only relational DBMSs come to mind. But the term is really broader and encompasses databases in general as well as programming languages and modeling. The connotations of “referential integrity” that arise from relational DBMSs are those of an implementation mechanism. However, the deeper meaning is that of dependencies among objects. In this broader sense, referential integrity is an integral aspect of how we think about problems and represent them via models. Consequently, we must deal with referential integrity regardless of the implementation platform—RDBMS, OO-DBMS, or programming language.

8. References

[1] Michael Blaha and William Premerlani. *Object-Oriented Modeling and Design for Database Applications*. Upper Saddle River, New Jersey: Prentice Hall, 1998.

[2] Michael Blaha. A retrospective on industrial database reverse engineering projects—part 2. *Eighth Working Conference on Reverse Engineering*, October 2001, Stuttgart, Germany, 147–153.

[3] Jim Melton and Alan R. Simon. *Understanding the New SQL: A Complete Guide*. San Francisco, California: Morgan Kaufmann, 1993.

[4] James Rumbaugh. Relations as semantic constructs in an object-oriented language. *OOPSLA’87* as *ACM SIGPLAN 22*, 12 (Dec. 1987), 466–481.

[5] en.wikipedia.org/wiki/Referential_integrity

[6] www.webopedia.com/TERM/R/referential_integrity.html

[7] computing-dictionary.thefreedictionary.com/referential+integrity