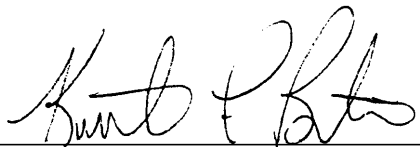


FOREWARD

This report is the second of five companion documents to the *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*. The companion documents address topics that are important to the design and development of secure database management systems, and are written for database vendors, system designers, evaluators, and researchers. This report addresses entity and referential integrity issues in multilevel secure database management systems.



Keith F. Brewster
Acting Chief, Partnerships and Processes

May 1996

ACKNOWLEDGMENTS

The National Computer Security Center extends special recognition to the authors of this document. The initial version was written by Vinti Doshi and Sushil Jajodia of the MITRE Corporation. The final version was written by Bill Wilson and Stan Wisseman of Arca Systems, Inc.

The documents in this series were produced under the guidance of Shawn P. O'Brien of the National Security Agency, LouAnna Notargiacomo and Barbara Blaustein of the MITRE Corporation, and David Wichers of Arca Systems, Inc.

We wish to thank the members of the information security community who enthusiastically gave of their time and technical expertise in reviewing these documents and in providing valuable comments and suggestions.

TABLE OF CONTENTS

| SECTION | PAGE |
|---|------|
| 1.0 INTRODUCTION..... | 1 |
| 1.1 BACKGROUND AND PURPOSE | 1 |
| 1.2 SCOPE | 1 |
| 1.3 INTRODUCTION TO ENTITY AND REFERENTIAL INTEGRITY..... | 2 |
| 1.4 AUDIENCES OF THIS DOCUMENT..... | 3 |
| 1.5 ORGANIZATION OF THIS DOCUMENT..... | 4 |
| 2.0 BACKGROUND..... | 5 |
| 2.1 BASIC CONCEPTS OF ENTITY AND REFERENTIAL INTEGRITY | 6 |
| 2.2 REFERENTIAL INTEGRITY RULES | 10 |
| 2.2.1 DELETE | 10 |
| 2.2.2 UPDATE of Referenced Relation..... | 13 |
| 2.2.3 NSERT or UPDATE of Referencing Relation..... | 17 |
| 2.2.4 Additional Rules | 17 |
| 2.3 PROBLEMS WITH REFERENTIAL INTEGRITY CONSTRAINTS | 17 |
| 3.0 ENTITY AND REFERENTIAL INTEGRITY IN MLS DATABASES | 18 |
| 3.1 RELATED TCSEC REQUIREMENTS | 18 |
| 3.2 BASIC CONCEPTS IN MULTILEVEL RELATIONS..... | 19 |
| 3.3 THE CONCEPT OF PRIMARY KEY IN MULTILEVEL RELATIONS..... | 21 |
| 3.4 ENTITY INTEGRITY IN MULTILEVEL RELATIONS..... | 21 |
| 3.5 REFERENTIAL INTEGRITY IN MULTILEVEL RELATIONS..... | 22 |
| 3.6 ENFORCEMENT OF INTEGRITY CONSTRAINTS | 25 |
| 4.0 COVERT CHANNELS..... | 27 |
| 4.1 PRIMARY KEY UNIQUENESS | 27 |
| 4.2 ENFORCEMENT OF REFERENTIAL INTEGRITY RULES | 28 |
| 4.2.1 Enforcement of Integrity Rules When $C[FK] = C[PK]$ | 29 |
| 4.2.2 Enforcement of Integrity Rules When $C[FK] > C[PK]$ | 30 |
| 4.3 RELATION TO DBMS ARCHITECTURE..... | 36 |
| 5.0 SUMMARY | 38 |
| REFERENCES | 39 |

LIST OF FIGURES

| FIGURE | PAGE |
|--|------|
| 2.1: RELATION SOD | 6 |
| 2.2: RELATION PS | 8 |
| 2.3: REFERENTIAL CONSTRAINTS IN RELATIONS PR, PS, AND SOD. | 9 |
| 2.4: RELATIONS SOD AND PS AFTER STARSHIP = "APOLLO" IS DELETED UNDER THE CASCADES-DELETE RULE | 11 |
| 2.5: RELATIONS SOD AND PS AFTER STARSHIP = "APOLLO" IS DELETED UNDER THE NULLIFIES-DELETE RULE | 12 |
| 2.6: RELATIONS SOD AND PS AFTER STARSHIP = "ENTERPRISE" IS UPDATED TO THE VALUE "USS ENTERPRISE" UNDER THE CASCADES-UPDATE RULE | 15 |
| 2.7: RELATIONS SOD AND PS AFTER STARSHIP = "ENTERPRISE" IS UPDATED TO THE VALUE "USS ENTERPRISE" UNDER THE NULLIFIES-UPDATE RULE | 16 |
| 3.1: DIFFERENT SOD VIEWS BASED ON ACCESS CLASS | 21 |
| 3.2: TYPICAL RELATION INSTANCES FOR SOD AND PS. | 23 |
| 3.3: UNCLASSIFIED INSTANCES OF PS AND SOD RELATIONS | 24 |
| 4.1: SOD AFTER INSERTION OF TUPLE CORRESPONDING TO STARSHIP = "VOYAGER" BY UNCLASSIFIED USER. | 28 |
| 4.2: RELATION PS AND SOD WHEN $C[FK] = C[PK]$ | 30 |
| 4.3: RELATIONS SOD AND PS WITHOUT POLYINSTANTIATION | 32 |
| 4.4: RELATIONS WITH TUPLE-LEVEL GRANULARITY | 34 |
| 4.5: RELATIONS WITH ATTRIBUTE-LEVEL GRANULARITY | 35 |

LIST OF TABLES

| TABLE | PAGE |
|---|------|
| 4.1: SUMMARY OF COVERT CHANNELS IN REFERENTIAL INTEGRITY RULES | 36 |

SECTION 1

INTRODUCTION

This document is the second volume in the series of companion documents to the *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria* [TDI 91; DoD 85]. This document examines entity and referential integrity issues in multilevel secure (MLS) database management systems and summarizes the research to date in these areas.

1.1 BACKGROUND AND PURPOSE

In 1991 the National Computer Security Center published the *Trusted Database Management System Interpretation (TDI) of the Trusted Computer System Evaluation Criteria* (TCSEC). The TDI, however, does not address many topics that are important to the design and development of secure database management systems (DBMSs). These topics (such as inference, aggregation, and database integrity) are being addressed by ongoing research and development. Since specific techniques in these topic areas had not yet gained broad acceptance, the topics were considered inappropriate for inclusion in the TDI.

The TDI is being supplemented by a series of companion documents to address these issues specific to secure DBMSs. Each companion document focuses on one topic by describing the problem, discussing the issues, and summarizing the research that has been done to date. The intent of the series is to make it clear to DBMS vendors, system designers, evaluators, and researchers what the issues are, the current approaches, their pros and cons, how they relate to a TCSEC/TDI evaluation, and what specific areas require additional research. Although some guidance may be presented, nothing contained within these documents should be interpreted as criteria.

These documents assume the reader understands basic DBMS concepts and relational database terminology. A security background sufficient to use the TDI and TCSEC is also assumed; however, fundamentals are discussed whenever a common understanding is important to the discussion.

1.2 SCOPE

This document addresses entity and referential integrity issues in multilevel secure DBMSs. It is the second of five volumes in the series of TDI companion documents, which includes the following documents:

- *Inference and Aggregation Issues in Secure Database Management Systems* [Inference 96]
- *Entity and Referential Integrity Issues in Multilevel Secure Database Management Systems*
- *Polyinstantiation Issues in Multilevel Secure Database Management Systems* [Poly 96]
- *Auditing Issues in Secure Database Management Systems* [Audit 96]
- *Discretionary Access Control Issues in High Assurance Secure Database Management Systems* [DAC 96]

This series of documents uses terminology from the relational model to provide a common basis for understanding the concepts presented. For most of the topics covered in this series the concepts presented should apply to most database modeling paradigms, depending on the specifics of each model. For the entity and referential integrity constraints discussed in this document, however, there are important differences between different models. This is because the primary keys in a relational DBMS represent both unique identifiers for tuples and values of attributes in the tuple. They must be protected in accordance with the sensitivity of the data they represent and this can interfere with their role as a unique identifier of tuples. By contrast, the object identifier (OID) in an Object-Oriented DBMS is not a property or attribute of the object it identifies, and does not represent a user-visible data value. Khoshafian and Copeland provide a useful taxonomy of different ways of specifying object identity including the techniques used in relational and Object-Oriented DBMSs [Khoshafian 86]. Similarly, in a relational DBMS, foreign keys represent data values and serve as pointers to other tuples. In an Object-Oriented DBMS, pointers to other objects are not used to store data values, but rather to represent the inclusion of one object within another. As in relational databases, problems may occur if a reference is deleted and some references remain in objects. The implications of these distinctions for enforcement of secrecy in an Object-Oriented DBMS require further research. This document specifically addresses relational DBMSs.

This document is related to the companion documents *Inference and Aggregation Issues in Secure Database Management Systems* and *Polyinstantiation Issues in Multilevel Secure Database Management Systems*. Much of the discussion of the relationship between enforcement of integrity constraints and multilevel security centers on the potential inference channels which integrity constraints can introduce. One way to avoid these channels for entity integrity is to use polyinstantiation. This is discussed later in more detail.

1.3 INTRODUCTION TO ENTITY AND REFERENTIAL INTEGRITY

Secrecy and integrity are two key concepts in the community of database security researchers and users. Secrecy refers to the protection or safety of the data against unauthorized disclosure, and integrity refers to the unauthorized alteration or destruction of data. In the wider database research community, integrity refers more generally to the correctness, accuracy, and internal consistency of data.

Preserving the accuracy of information is extremely important in any database. In the relational model, one way in which accuracy of information is preserved is by preventing updates that violate integrity constraints. Entity integrity and referential integrity are two of the most important integrity constraints. They are defined on relations and are enforced by the DBMS.

Entity integrity states that no primary key attribute of a relation is allowed to have null values. In addition, the primary key must have a unique value. These properties of a primary key correspond to the fact that entities in the real world are distinguishable (i.e., they have a unique identification of some kind). In the relational model, the primary key performs the function of unique identification. Referential integrity is an interrelation integrity constraint. In general, referential integrity defines existence relationships between entities in a database that need to be maintained by the DBMS. This document defines these integrity constraints formally and presents their importance in the context of security in detail.

In multilevel applications, data are classified to control disclosure. It is important to realize the necessary trade-off to be made between secrecy and integrity because of the challenges in enforcing integrity constraints across multiple secrecy levels without disclosure of information. This document defines entity and referential integrity for a standard relational DBMS and then presents methods for resolving the inherent integrity/secrecy conflict in an MLS environment.

1.4 AUDIENCES OF THIS DOCUMENT

This document is targeted at four primary audiences: the security research community, database application developers/system integrators, trusted product vendors, and product evaluators. In general, this document is intended to present a framework for understanding the nature of entity and referential integrity policies or requirements and how they conflict with the need to enforce an access control policy. This framework is used to describe the research that has been done to date and the approaches they present for providing solutions. Each of the specific audiences should expect to get the following from this document:

Researcher

This document describes the basic problems in simultaneously enforcing integrity constraints and MAC. Some important research contributions are discussed as various topics are covered. Related research on other aspects of integrity and the relationship to secrecy is discussed in Section 3.6. For additional relevant work, the researcher should consult two other companion documents, *Inference and Aggregation Issues in Secure Database Management Systems* [Inference 96] and *Polyinstantiation Issues in Multilevel Secure Database Management Systems* [Poly 96].

Database Application Developer/System Integrator

This document describes various entity and referential integrity facilities which could be provided by an MLS DBMS. It discusses the implications of these facilities relative to the tradeoffs involved in meeting a system's requirements for secrecy and integrity.

Trusted Product Vendor

This document describes the conflicts between Mandatory Access Controls (MAC) and multilevel entity and referential integrity constraints. It then discusses approaches to entity and referential integrity enforcement in an MLS database and the benefits and drawbacks of these approaches. In particular, it considers how the issues with enforcement of entity and referential integrity constraints in an MLS database relate to the TCSEC access control and covert channel requirements. This is discussed in the context of tuple level as well as element level labeling. This document will provide a framework for understanding specific requirements interpretations as they are developed by the National Computer Security Center.

Evaluator

This document describes access control and covert channel issues related to entity and referential integrity. It provides a framework for analyzing specific vendor mechanisms and understanding Technical Review Board/Interpretations Working Group decisions.

1.5 ORGANIZATION OF THIS DOCUMENT

The organization of the remainder of this document is as follows:

- Section 2 gives the definitions of entity and referential integrity with respect to single-level DBMSs and describes various important concepts related to referential integrity.
- Section 3 defines the basic concepts of entity and referential integrity with respect to multilevel relations.
- Section 4 discusses the issues associated with covert channels related to enforcing entity and referential integrity constraints.
- Section 5 presents conclusions.

SECTION 2

BACKGROUND

Entity integrity and referential integrity are two basic integrity requirements in a relational DBMS (RDBMS). They help prevent incorrect data from being entered into the database. Entity integrity guarantees a unique representation of each entity in the database through specification of primary key attributes for each relation. Referential integrity assures that if any references exist between two or more entities, then the related entities do exist in the database. Referential integrity places restrictions on foreign key attributes.

Referential integrity is one of the most important interrelation integrity constraints for the relational data model. Although its basic concepts have been discussed over the past ten years, a successful referential integrity capability for a RDBMS is still a challenging implementation problem. While referential integrity appears to be a simple concept, the operational definitions that appear in the literature are imprecise and lead to a host of anomalies.

A standard relational database can be perceived by its users as a collection of relations. A *relation* has well-defined mathematical properties and is used to store data. Each relation has two parts:

1. A state-invariant relation schema \mathbf{R} (A_1, A_2, \dots, A_n), where each A_i is an attribute over some domain D_i , $1 \leq i \leq n$, which is a set of acceptable values for A_i .
2. A state-dependent relation R over \mathbf{R} , which is a set of distinct tuples of the form (a_1, a_2, \dots, a_n) , where each element a_i is a value in domain d_i , $1 \leq i \leq n$.

The following notation and conventions are used throughout this document. In definitions, relation schemes are denoted in bold fonts (such as \mathbf{R} , $\mathbf{R1}$, $\mathbf{R2}$), and the corresponding relations are denoted using plain fonts (such as R , $R1$, $R2$). However, when clear in specific examples, the same symbol is used to denote the relation schema, as well as the corresponding relation. Let X denote one or more attributes of a relation R . If t is a tuple in a relation R , $t[X]$ denotes the projection of t onto X .

This section first defines entity and referential integrity, followed by terminology related to these concepts. The examples make use of Structured Query Language (SQL) syntax [ANSI 92, 94]. The section then gives various rules for enforcing referential integrity and discusses the problems associated with referential integrity in single-level relational databases. From this understanding of integrity, we can then better discuss the conflict with enforcement of a MAC policy in Section 3.

2.1 BASIC CONCEPTS OF ENTITY AND REFERENTIAL INTEGRITY

Before defining entity integrity, it is necessary to define the concept of a primary key.

In simple terms, the primary key of a relation is just a unique identifier of each tuple in the relation. More precisely, a primary key for a relation \mathbf{R} is a set of attributes PK with the following properties:

1. The value of PK uniquely identifies each tuple in the relation R. That is, for any state of R, and any tuples t_1 and t_2 , if $t_1[PK] = t_2[PK]$ then $t_1 = t_2$.
2. PK is minimal. That is, there is no proper subset of PK satisfying property 1.

It is possible there is more than one set of attributes satisfying these properties. Any such set is called a candidate key. Precisely one candidate must be chosen and designated as the primary key of R.

Example 1. Consider the relation schema SOD (Starship, Objective, Destination), which contains for each starship its name (Starship), the purpose of the flight (Objective), and its destination (Destination). An example relation for relation schema SOD is given in Figure 2.1.

| STARSHIP | OBJECTIVE | DESTINATION |
|------------|-------------|-------------|
| Enterprise | Exploration | Talos |
| Voyager | Spying | Mars |
| Apollo | Exploration | Moon |
| Saratoga | Mining | Moon |

Figure 2.1: Relation SOD

Since the attribute Starship uniquely identifies each tuple of the relation SOD, it is a candidate key of the relation SOD. As Starship is the only candidate key of the relation SOD, it is also the primary key of relation SOD.

In SQL syntax, the attribute(s) constituting the primary key can be specified when a relation schema is created, as the following shows:

```
CREATE TABLE SOD
  (Starship CHAR(20) NOT NULL,
   Objective CHAR(15),
   Destination CHAR(10),
   PRIMARY KEY (Starship))
```

Notice that the ANSI syntax uses the term “TABLE” instead of the term “relation” used in this document.

The entity integrity constraint requires that all attribute values of the primary key be defined for each tuple.

Entity Integrity: If PK is the primary key for relation **R**, then for all states of R, for every tuple t in R and every attribute A in PK, t[A] is not null.

A foreign key provides a basis for a tuple to refer to another tuple. Let **R1** and **R2** be two relation schemas, and let R1 and R2 denote the respective relations corresponding to these schemas.

Referential Integrity: Let PK denote the primary key of **R2**, and let FK denote one or more attributes of the relational schema R1. FK is said to be a *foreign key* of **R1** referencing **R2** if given any tuple t_1 in **R1**, the following two requirements are met:

1. t_1 [FK] is either wholly null or wholly non-null, and
2. Whenever t_1 [FK] is non-null, there is a tuple t_2 in R2 such that t_1 [FK] = t_2 [PK].

Here R1 is the *referencing* relation and R2 is the *referenced* relation. Sometimes PK is referred to as the target value of the foreign key FK. Notice that **R1** and **R2** need not be distinct.

Example 2. In addition to the relation schema SOD given in Example 1, consider the relation schema PS (Person_Name, Starship), which contains the name of the employee (Person_Name) and the name of the ship assigned to the employee (Starship). An example relation for PS is given in Figure 2.2.

Suppose Person_Name uniquely identifies each employee in the PS relation, then Person_Name becomes the primary key for the relation schema PS. If Starship is declared a foreign key for PS with the relation schema SOD as the referenced relation, a DBMS that supports referential integrity will automatically ensure that whenever an employee is assigned a (non-null) ship, there is a matching tuple in the relation schema SOD with an identical value for the attribute Starship.

Notice that employee Mr. Spock, given in Figure 2.2, has not been assigned any starship. It seems from the definition of the PS relation that nulls are allowed. In some applications it may be desirable to prohibit this situation (i.e., every employee must be assigned a starship). The referencing relation's schema definition can specify whether or not the foreign key attribute can assume a null value.

Using SQL syntax, foreign keys can be specified as follows:

```
CREATE TABLE PS
    Person_Name      CHAR(15) NOT NULL,
    Starship         CHAR(20),
    PRIMARY KEY     (Person_Name),
    FOREIGN KEY     (Starship) REFERENCES SOD (Starship)).
```

| PERSON_NAME | STARSHIP |
|--------------|------------|
| James Kirk | Enterprise |
| Mr. Spock | <null> |
| Tim McKelley | Apollo |

Figure 2.2: Relation PS

It follows from the definition of the foreign key that there is an identical matching primary key value in the referenced relation for every foreign key value in the referencing relation. It is important to maintain the integrity between the referencing values (foreign key values) and the referenced values (primary key values).

Example 3. Consider the relation schema PR (Person_Name, Rank), which contains the employee name (Person_Name) and the rank of the employee (Rank). Person_Name is the primary key for the relation schema. Person_Name is also the foreign key referring to relation schema PS.

The relation schema PR can be expressed in SQL as follows:

```
CREATE TABLE PR
    (Person_Name      CHAR(20) NOT NULL,
     Rank             CHAR(15),
     PRIMARY KEY      (Person_Name),
     FOREIGN KEY      (Person_Name) REFERENCES PS (Person Name))
```

Here, the relation schema PR references the relation schema PS given in Example 2. In addition, the relation schema PS in Example 2 references the relation schema SOD. Since there is a referential constraint *from* PR *to* PS, and also *from* PS *to* SOD, relation schema PS becomes both a referencing relation and a referenced relation, see Figure 2.3.

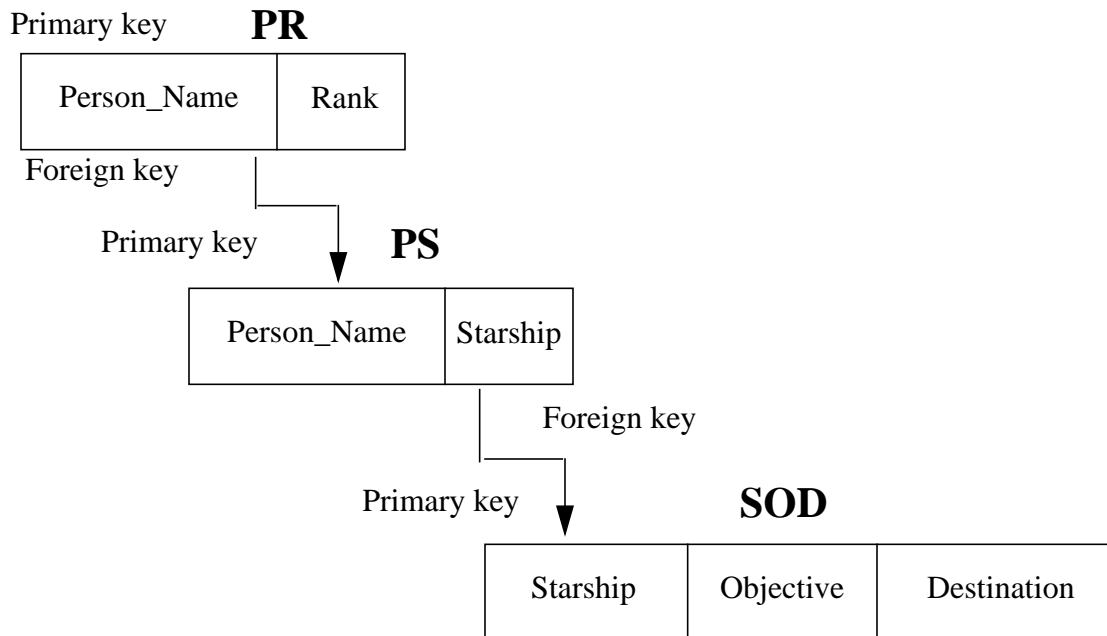


Figure 2.3: Referential Constraints in Relations PR, PS, and SOD

This can be depicted as a referential path from PR to SOD as follows:

$$\text{PR} \rightarrow \text{PS} \rightarrow \text{SOD}$$

Referential Cycles: Closed referential paths, or referential paths that point back to the starting relation, are referred to as *referential cycles*. In general, the referential cycle including n relations (where $n > 1$) can be represented as follows [Date 86, 90]:

$$R_n \rightarrow R_{n-1} \rightarrow R_{n-2} \rightarrow \dots \rightarrow R_2 \rightarrow R_1 \rightarrow R_n.$$

Referential cycles are of interest since they can cause insertion problems. In the presence of referential cycles, it is necessary that one of the following constraints be true; otherwise, it is impossible to insert the first tuple in any or each of the relations:

1. One of the foreign keys in the referential cycle must be allowed to have a null value, or
2. The constraint check should be done only at the end-of-transaction (i.e., when the transaction commits).

A *self-referencing* relation is a special case of a referential cycle that includes exactly one relation: a relation can have multiple foreign keys corresponding to different relations.

Some important concepts related to the terms defined in this section can be summarized as follows:

- A primary key of a relation cannot have a null value in any tuple within that relation (entity integrity) and each primary key must be unique (property of a primary key).
- Foreign keys can have null values. For example, in Figure 2.2 the employee Mr. Spock has not been assigned any starship as yet; therefore, the Starship value for Mr. Spock is null. (Note: If in a specific instance it is desired to prohibit null values for foreign keys, this can be done by using a NOT NULL clause for the attributes in the foreign key when the relation is created.)
- For referential integrity, every non-null value of a foreign key must be matched by an identical primary key value in the referenced relation. But the converse need not be true; for example, starship Saratoga in Figure 2.1 has no employee assigned to it in Figure 2.2.
- A relation can be a referenced relation, as well as a referencing relation.

2.2 REFERENTIAL INTEGRITY RULES

Whenever two or more relations are related through referential constraints, it is necessary that references be kept consistent in the face of insertions, deletions, and updates to these relations. Date identifies several actions which can be taken to maintain consistency of references [Date 90]. Exactly which action is chosen for a particular relation depends on the behavior desired by the underlying application. These possible actions are discussed in the following subsections.

2.2.1 DELETE

When the target of a foreign key is deleted it could result in a dangling reference. Some action is needed to make sure this does not happen. Four options that may be specified for the referenced relation are RESTRICTED-delete, CASCADES-delete, NULLIFIES-delete, and SET DEFALILT-delete. Each of these is explained in turn.

RESTRICTED-delete

If the *RESTRICTED-delete* option is specified, the primary key value in the referenced relation cannot be deleted as long as there exists at least one matching foreign key value in the referencing relation. The deletion of the referenced value is restricted to the case where there is no corresponding referencing value in the referencing relation.

For instance, suppose an attempt is made to delete the starship Apollo from the relation SOD. Since the referencing relation PS refers to the relation SOD through the attribute Starship, under the RESTRICTED-delete rule, this deletion cannot be performed as there is a tuple in PS referencing starship Apollo. The delete is denied and the two relations would remain unchanged.

The syntax for specifying the RESTRICTED-delete rule may be given as follows:

```
CREATE TABLE PS
    (Person_Name      CHAR(20) NOT NULL,
     Starship         CHAR(15),
     PRIMARY KEY     (Person_Name),
     FOREIGN KEY     (Starship) REFERENCES SOD (Starship),
     ON DELETE       RESTRICT)
```

CASCADES-delete

Under the *CASCADES-delete* option, whenever the target tuple of a foreign key is deleted, all referencing foreign key tuples are also deleted. In the previous example, if an attempt is made to delete the starship Apollo from the relation SOD, the delete would be cascaded to the relation PS and the employee tuples referencing the starship Apollo would also be deleted. After such a CASCADES-delete operation, the relations SOD and PD would change to the form given in Figure 2.4.

SOD

| STARSHIP | OBJECTIVE | DESTINATION |
|------------|-------------|-------------|
| Enterprise | Exploration | Talos |
| Voyager | Spying | Mars |
| Saratoga | Mining | Rigel |

PS

| PERSON_NAME | STARSHIP |
|-------------|------------|
| James Kirk | Enterprise |
| Mr. Spock | <null> |

Figure 2.4: Relations SOD and PS after Starship = “Apollo” is Deleted under the CASCADES-Delete Rule

In SQL, the CASCADES-delete rule can be specified as follows:

```
CREATE TABLE PS
```

```
(Person_Name      CHAR(20) NOT NULL,  
Starship          CHAR(15),  
PRIMARY KEY      (Person_Name),  
FOREIGN KEY      (Starship) REFERENCES SOD (Starship),  
ON DELETE        CASCADE)
```

NULLIFIES-delete

Whenever the *NULLIFIES-delete* option is specified for a foreign key, the referencing foreign key attribute values will be set to a null value when the tuple containing the referenced value is deleted. In the example given earlier, when starship Apollo is deleted, the foreign key value, for all tuples referencing Apollo in relation PS, is set to null. The new state of the relations is as shown in Figure 2.5.

SOD

| STARSHIP | OBJECTIVE | DESTINATION |
|------------|-------------|-------------|
| Enterprise | Exploration | Talos |
| Voyager | Spying | Mars |
| Saratoga | Mining | Rigel |

PS

| PERSON_NAME | STARSHIP |
|--------------|------------|
| James Kirk | Enterprise |
| Mr. Spock | <null> |
| Tim McKelley | <null> |

Figure 2.5: Relations SOD and PS after Starship = “Apollo” is Deleted under the NULLIFIES-Delete Rule

Before setting the referencing value to null, it is important to make sure that the relation schema allows a null value for the foreign key attribute. Otherwise, there will be a conflict.

The NULLIFIES-delete rule can be expressed in SQL as follows:

```
CREATE TABLE PS
```

```
(Person_Name      CHAR(20) NOT NULL,  
Starship          CHAR(15),  
PRIMARY KEY      (Person_Name),  
FOREIGN KEY      (Starship) REFERENCES SOD (Starship),  
ON DELETE        SET NULL)
```

SET DEFAULT-delete

The SQL2 [ANSI 92] and SQL3 [ANSI 94] standards provide yet another referential action for the delete operation, called SET DEFAULT. This option is similar to the NULLIFIES delete rule given above. The only difference is that the SET DEFAULT option contains a default clause that specifies the default value for the attribute. The default value may be a null or a non-null value. If the referential action specifies the SET DEFAULT option, then each referencing attribute is defined by an implicit or explicit default clause. If the default value is null, then SET DEFAULT is equivalent to a SET NULL.

The SQL syntax for the SET DEFAULT-delete rule is as follows:

```
CREATE TABLE PS
    (Person_Name      CHAR(20) NOT NULL,
     Starship         CHAR(15),
     PRIMARY KEY      (Person_Name),
     FOREIGN KEY       (Starship) REFERENCES SOD (Starship),
     ON DELETE        SET DEFAULT < option>)
```

The default option is specified in the <option> field.

2.2.2 UPDATE of Referenced Relation

Changing the primary key values in a referenced relation could violate referential integrity unless appropriate action is taken. There are different options which can be specified for the referencing relation whenever an update is made to the target primary key values of a referenced relation. The possible options are RESTRICTED-update, CASCADES-update, NULLIFIES-update, or SET DEFAULT-update.

RESTRICTED-update

Under the *RESTRICTED-update* option, an update to the target primary key values of a referenced relation cannot be made as long as there exists at least one matching foreign key value in the referencing relation. The update to the referenced tuple is restricted to the case where there are no matching referencing tuples.

As an example, suppose an attempt is made to update the name of the starship from Enterprise to USS Enterprise in relation SOD shown in Figure 2.1. As there is an employee assigned to the ship, the update is rejected. The relations SOD and PS remain unchanged, and look like Figures 2.1 and 2.2.

The syntax for specifying the RESTRICTED-update option may be specified as follows:

```
CREATE TABLE PS
    (Person_Name      CHAR(20) NOT NULL,
     Starship         CHAR(15),
     PRIMARY KEY      (Person_Name),
     FOREIGN KEY       (Starship) REFERENCES SOD (Starship),
     ON UPDATE        RESTRICT)
```

CASCADES-update

If the *CASCADES-update* option is specified for a referencing relation, whenever an update is made to the referenced attribute, all changes are cascaded to the matching referencing attributes.

This means that whenever the target value of a foreign key is modified, all referencing foreign key values are also similarly modified. Consider the example where an attempt is made to update the name of the starship from Enterprise to USS Enterprise in the relation schema SOD. Under CASCADE-update, the update is cascaded to the matching referencing tuples in the relation schema PS. For the employee assigned to starship Enterprise, the name is updated to USS Enterprise. The new state of the two relations SOD and PS is shown in Figure 2.6.

In SQL, the CASCADE-update option can be expressed as follows:

CREATE TABLE PS

```
(Person_Name      CHAR(20) NOT NULL,
 Starship         CHAR(15),
 PRIMARY KEY     (Person_Name),
 FOREIGN KEY     (Starship) REFERENCES SOD (Starship),
 ON UPDATE      CASCADE)
```

SOD

| STARSHIP | OBJECTIVE | DESTINATION |
|----------------|-------------|-------------|
| USS Enterprise | Exploration | Talos |
| Voyager | Spying | Mars |
| Apollo | Exploration | Moon |
| Saratoga | Mining | Rigel |

PS

| PERSON_NAME | STARSHIP |
|--------------|----------------|
| James Kirk | USS Enterprise |
| Mr. Spock | Null |
| Tim McKelley | Apollo |

Figure 2.6: Relations SOD and PS after Starship = “Enterprise” is Updated to the Value “USS Enterprise” under the CASCADES-Update Rule

NULLIFIES-update

Under the *NULLIFIES-update* option, the referencing foreign key attribute values are set to null whenever the target primary key values in the referenced relation are updated. In the example given earlier, when the name of the starship is updated from Enterprise to USS Enterprise, the Starship for the employee assigned to Enterprise is set to null. The new state of the relations is shown in Figure 2.7.

Before setting the referencing value to null, it is important to make sure that the schema definition allows a null value for the foreign key attribute. If the nulls rule does not allow a null value, then there will be a conflict between the two integrity rules.

The NULLIFIES-update rule is expressed in SQL syntax as follows:

```
CREATE TABLE PS
```

```
(Person_Name      CHAR(20) NOT NULL,  
Starship          CHAR(15),  
PRIMARY KEY      (Person_Name),  
FOREIGN KEY      (Starship) REFERENCES SOD (Starship),  
ON UPDATE        SET NULL)
```

SOD

| STARSHIP | OBJECTIVE | DESTINATION |
|----------------|-------------|-------------|
| USS Enterprise | Exploration | Talos |
| Voyager | Spying | Mars |
| Apollo | Exploration | Moon |
| Saratoga | Mining | Rigel |

PS

| PERSON_NAME | STARSHIP |
|--------------|----------|
| James Kirk | < null > |
| Mr. Spock | < null > |
| Tim McKelley | Apollo |

Figure 2.7: Relations SOD and PS after Starship = “Enterprise” is Updated to the Value “USS Enterprise” under the NULLIFIES Update Rule

SET DEFAULT-update

SQL2 and SQL3 define another referential constraint for update, which is SET DEFAULT. If the referential constraint specifies the SET DEFAULT option, then each referencing attribute is defined by an implicit or explicit default clause. The default clause specifies the default value for the attribute, and the default value may be a null or a non-null value. If the default value is null, then the SET DEFAULT option can be considered equivalent to the NULLIFIES update rule.

The SQL syntax for the SET DEFAULT-update rule is as follows:

```
CREATE TABLE PS
    (Person_Name      CHAR(20) NOT NULL,
     Starship         CHAR(15),
     PRIMARY KEY     (Person_Name),
     FOREIGN KEY     (Starship) REFERENCES SOD (Starship),
     ON UPDATE       SET DEFAULT <option>)
```

The default option is specified in the <option> field.

2.2.3 INSERT or UPDATE of Referencing Relation

When a tuple is inserted into a relation with a foreign key or a tuple in the relation is updated, the insertion (or update) of a foreign key value should be in accordance with referential integrity. Each foreign key value must be NULL or have an identical matching primary key value in the referenced relation. This requirement can be enforced by disallowing the insert (or update) unless the condition is met or by setting the foreign key to a default value or null (if nulls are allowed) when the condition is not met.

2.2.4 Additional Rules

In spite of the many integrity rules discussed in this section, the rules above are not exhaustive. Additional options include conversation with the end user and transferring information to some other files. For instance, an additional option in SQL3 is the PENDANT specification, which implies that each primary key value in the referenced relation is the same as some foreign key value in the referencing relation. For example, if the referential constraint for the relation EMP is as shown below, then as soon as the last employee in a particular department is deleted, the department will also be deleted.

```
FOREIGN KEY (Starship) REFERENCES [PENDANT] SOD(Starship)
```

2.3 PROBLEMS WITH REFERENTIAL INTEGRITY CONSTRAINTS

Referential integrity plays a major role in RDBMSs where it is employed to express existence dependencies. The concept of referential integrity seems to be simple, but it has been a topic of discussion for the past several years [Markowitz 90; Horowitz 92]. The operational definitions are not precise and, therefore, lead to some anomalies. Confusion surrounds both the concept itself and

its implementation in RDBMSs. There is a need to carefully examine the data manipulation problems caused by certain referential integrity structures and to find ways to avoid these problems.

In a RDBMS, a single update or delete can affect multiple tuples through the application of referential integrity rules (e.g., cascading deletes or updates). It is necessary for these processes to be deterministic. In other words, the final state of the database should be a function of the initial state of the database, and the specific statement to be executed. At times, certain referential constraints associated with relations may block update or deletion of some tuples because of conflicting referential constraint specifications, thus making the outcome of the update or delete statement unpredictable. Some of the problems that have been identified are dependency on the order in which referential integrity constraints are applied, referential cycles, and dependency on the order in which the tuples are processed in response to a query. Discussion of these problems in detail is outside the scope of this document.

SECTION 3

ENTITY AND REFERENTIAL INTEGRITY IN MLS DATABASES

Before entity and referential integrity can be defined for multilevel relations, it is necessary to extend the basic concepts of classical (single-level) relations to the multilevel context. In MLS databases, data are assigned different security levels and each user is assigned a clearance level. Each user can only access data they are cleared for based on the user's security clearance. As a consequence the meaning of relations becomes more complicated in the multilevel world, and there is no clear consensus regarding the exact definition of a multilevel relational model.

To develop an MLS database, it is necessary to (1) define a *security policy*, (2) develop mechanisms that support the policy, and (3) get the necessary assurance that the system developed is secure. This section first reviews the basic concepts for the multilevel relational data model, then defines entity and referential integrity constraints for multilevel relations.

3.1 RELATED TCSEC REQUIREMENTS

The TCSEC imposes two types of requirements which affect how entity and referential integrity can be enforced in a DBMS targeted to meet the TCSEC requirements at B1 and above. A DBMS targeted for Class B1 or above must enforce a MAC policy over all subjects and storage objects it controls. This places requirements on the sensitivity labels of elements of primary and foreign keys and on the relationships between sensitivity levels of foreign keys and the primary keys they reference. For a DBMS targeted at Class B2 or above, the TCSEC requires a thorough search for covert channels (storage channels only at Class B2). In the multilevel context, enforcement of the uniqueness of primary keys and enforcement of referential integrity can create significant covert channels. Guidelines on acceptable bandwidths for covert channels are given in Section 8 of the TCSEC. This section specifically addresses the implications of direct enforcement of MAC controls (as required at B1 and above). The implications of integrity enforcement on covert channels is covered in Section 4. A determination of precisely which approaches to enforcing entity and referential integrity will be considered compliant with the TCSEC requirements will evolve over time as the Evaluation Community publishes official interpretations on this subject. However, with the successful evaluation of Trusted Oracle7 DBMS MAC Mode with an unmodified integrity mechanism, precedence has been set for Class B1 DBMSs [Oracle 94].

3.2 BASIC CONCEPTS IN MULTILEVEL RELATIONS

The basic model for multilevel relations needs to be defined with a MAC policy in mind. The MAC policy for MLS databases is often based on the Bell-LaPadula model [Bell 76], which is stated in terms of subjects and objects. A subject is an active entity, such as a process that can request access to objects, whereas an object¹ is a passive entity, such as a record, a file, or a field within a record. Every subject is assigned a clearance level and every object a classification level. Classification levels and clearances are collectively referred to as *security levels*, and form a lattice. Each security level has two components: a hierarchical component and a set (possibly empty) of unordered

1. "object", as used in Bell and LaPadula model, is not the same as an object-oriented DBMS (OODBMS) where objects are active containers of information.

categories. A security level c_1 is said to *dominate* security level c_2 , in the partial order, if (1) the hierarchical component of c_1 is greater than or equal to that of c_2 , and (2) all categories in c_2 are included in those of c_1 . A security level c_1 *strictly dominates* security level c_2 , in the partial order, if (1) c_1 dominates c_2 , and (2) c_1 does not equal c_2 . The following are two necessary conditions in the Bell-LaPadula model [Bell 76; DoD 85]:

1. *The Simple Security Condition or “No Read Ups”*: A subject can only read objects at a security level dominated by the subject’s level, and
2. *The *-Property (Star Property) or “No Write Downs”*: A subject can only write objects at a security level that dominates the subject’s level.

To apply these concepts to a DBMS it is necessary to determine the granularity of the objects protected by MAC, i.e., the storage objects. Security levels are then associated with these storage objects. Work on MLS databases has focused on four choices for assigning security levels to data stored in a relation. One can assign security levels to entire relations, to individual tuples (rows) of a relation, to individual attributes (columns) of a relation, or to individual elements of a relation. In the definitions below, the most general case is considered, in which security levels are assigned to individual data elements stored in relations. However, since tuple level labeling is used in all DBMS products evaluated to date, the concepts are also considered in the (simplified) context of tuple level labeling.

A relation schema in the multilevel context not only contains all its data attributes, but also includes the corresponding security level for each data attribute. Formal definitions of a multilevel relation schema and multilevel relations are as follows [Denning 87]:

Relation Schema: A multilevel relation schema has the following form:

$$\mathbf{R}(A_1, C_1, A_2, C_2, \dots, A_n, C_n).$$

Each A_i is a data attribute over some domain D_i and C_i is the classification attribute of A_i . The domain of C_i , $i = 1, \dots, n$ is some set consisting of security levels.

If tuple level labeling is used, there will only be one classification attribute for the entire tuple:

$$\mathbf{R}(A_1, A_2, \dots, A_n, C).$$

For a multilevel logical relation schema \mathbf{R} , there is a corresponding relation R_c per security level in the security lattice. A user having a clearance at a security level c sees the relation R_c , which contains data at security level c or below. More formally,

Relations: A relation at a security level c has the following form:

$$R_c(A_1, C_1, A_2, C_2, \dots, A_n, C_n)$$

R_c consists of a set of tuples of the form $(a_1, c_1, a_2, \dots, a_n, c_n)$, where each a_i lies in the domain D_i or a_i is null, $c \geq c_i$, c_i , $i = 1 \dots, n$. Even if a_i is null, its classification attribute c_i is never null.

When an attribute value a_i is null, there are two reasonable possibilities for the corresponding security level value: c_i could be assigned a value that is identical to c , or the security level of the attributes constituting the primary key. In this document, c_i would be assigned a value consistent with the second possibility.

If the security level of the primary key value is not dominated by c in a tuple, then that tuple is not shown in R_c at all. If the security level of the primary key value is dominated by c , then that tuple is shown in R_c ; however, any attribute value with a security level not dominated by c , is shown having a null value with the corresponding security level the same as that of the primary key value.

For tuple level labeling, R_c consists of those tuples whose security level is dominated by c .

These concepts can be better explained with the help of an example taken from [Jajodia 91].

Example 4. Consider the relation schema SOD given in Example 1. Starship is the primary key of the relation schema SOD, and the security levels are assigned at the granularity of individual data elements. The hierarchical order usually followed is Top Secret (TS), Secret (S), Confidential (C), and Unclassified (U). A user with a Secret clearance will see the entire multilevel relation SOD_S , while a user with an Unclassified clearance will see the Unclassified instance SOD_U as shown in Figure 3.1.

SOD_S

| STARSHIP | OBJECTIVE | DESTINATION |
|--------------|---------------|-------------|
| Enterprise U | Exploration U | Talos U |
| Voyager U | Spying S | Mars S |

SOD_U

| STARSHIP | OBJECTIVE | DESTINATION |
|--------------|-------------|-------------|
| Enterprise U | Exploration | Talos U |
| Voyager U | <null> | <nulls> U |

Figure 3.1: Different SOD Views Based on Access Class

3.3 THE CONCEPT OF PRIMARY KEY IN MULTILEVEL RELATIONS

The notion of a primary key is a fundamental concept in the world of single-level relational databases. The primary key is used to maintain integrity of relations, it is used during the database design, and it is also used for storage and retrieval purposes. The primary key for a single-level relation must satisfy the *uniqueness property* and the *minimality property*, as discussed in Section 2.1. Entity integrity requires that the primary key serves as a unique identifier of each tuple in the relation and that it does not contain a null value. The uniqueness property can create covert channels as discussed in Section 4. One approach to avoiding these channels involves augmenting the user-defined primary key with security level attributes associated with the primary key attributes. However, in the remainder of this document it is assumed that there is a user-specified primary key consisting of a subset of the data attributes and not including the security level [Denning 87]. It is called the apparent primary key of the multilevel relation schema.

3.4 ENTITY INTEGRITY IN MULTILEVEL RELATIONS

A relation schema R will be assumed to have a user-defined apparent primary key consisting of one or more data attributes of R . It will be denoted by PK . The entity integrity property of the standard relational model can be extended to the multilevel environment by defining the three requirements given below, which are applicable only to element level classification [Denning 87]. The three requirements are automatically met for tuple level classification.

The first requirement is the same as for the standard relational model, that no tuple in an R_c have a null value for any attribute in PK . The second requirement states that the PK values should be uniformly classified; otherwise, the PK value would not be wholly non-null or wholly null for some security level. The third requirement states that the security level of values for the attributes not in PK must dominate the security level of the PK attributes; otherwise, there could be instances where non-null attributes would be associated with a null primary key when viewed below the level of the primary key. These three requirements can be more formally defined as follows:

A multilevel relation schema R is said to satisfy *entity integrity*, if for all relation instances R_c of R and, tuple $t \in R_c$,

1. If $A_i \in PK$, then $t[A_i] \neq \text{null}$,
2. If $A_i, A_j \in PK$, then $t[C_i] = t[C_j]$, i.e., PK is said to be uniformly classified, and
3. If $A_i \notin PK$, then $t[C_i] \geq t[C[PK]]$ (where $C[PK]$ is the classification of the apparent primary key PK).

3.5 REFERENTIAL INTEGRITY IN MULTILEVEL RELATIONS

As discussed in the previous section, a non-null foreign key value in a referencing relation should always have a matching primary key value in the referenced relation. In an MLS database, this requirement has implications for the relationship among the security levels of the attributes of the foreign key and the security level of the referenced primary key. In addition, the possible actions which could be taken to enforce referential integrity on update and deletion can introduce covert

channels as discussed in Section 4. Relation schemas, in the following example, will be used to illustrate the different situations in which a security conflict may occur.

Example 5. Consider the multilevel relation schema PS given in Example 2. Person_Name is the apparent primary key of the relation schema PS, and Starship is the foreign key of PS, which refers to the relation schema SOD. Typical relation instances for SOD and PS are given in Figure 3.2. Note that in the example schema only the attribute names are part of the primary key and the foreign key. The respective security levels are not a part of the key. Therefore, there are some tuples with the security level of the foreign key in relation PS higher or lower than the security level of the corresponding primary key in relation SOD. The example includes such tuples to explain the conflicts arising when the security levels are at different levels.

SOD

| STARSHIP | OBJECTIVE | DESTINATION |
|--------------|---------------|-------------|
| Enterprise U | Exploration U | Talos U |
| Voyager U | Spying S | Mars S |
| Apollo S | Spying S | Moon S |

PS

| PERSON_NAME | STARSHIP |
|----------------|--------------|
| James Kirk U | Enterprise U |
| Mr. Spock U | Voyager S |
| Tim McKelley U | Apollo U |

Figure 3.2: Typical Relation Instances for SOD and PS

Recall the definition of referential integrity for standard relational databases:

Referential Integrity: If FK is a foreign key of **R1** referencing **R2**, PK is the primary key of **R2**, t_1 is a tuple of an instance R1 of **R1**, and t_2 is a tuple of an instance R2 of **R2**, then

1. t_1 [FK] is either wholly null or wholly non-null, and
2. Whenever t_1 [FK] is non-null, there is a tuple t_2 in R2 such that t_1 [FK] = t_2 [PK].

For MLS databases we need to have this property be true for the database as seen from any security level. Assume t_1 [FK] is not wholly null as viewed from some security level. The first requirement above says that it must then be wholly non-null. If the attribute values making up FK did not all

have the same security level, there would be some security level for which some of the attribute values in FK would be null and others non-null. Since this is not allowed, all attribute values in FK must have the same security level. If all attribute values of FK are null, then by convention they are assigned the same security level. This argument gives us the first rule for referential integrity.

Rule 1. The foreign key of the referencing relation must be uniformly classified (i.e., all attribute values that make up the foreign key must be assigned an identical security level).

This rule is automatically satisfied for tuple level labeling.

The second part of referential integrity requires that if the foreign key FK is not null then there is a matching primary key PK in the referenced relation R2. For this to be true for the database as seen from the security level of FK, the matching PK must be visible at the security level of FK. Hence, the security level of FK must dominate the security level of the matching primary key PK. This gives us the second rule of referential integrity.

Rule 2. The security level of the foreign key must dominate the security level of the primary key of the referenced tuple (i.e., $C[FK] \geq C[PK]$).

For tuple level labeling, this rule reduces to the requirement that the security level of the referenced tuple must be dominated by the security level of the referencing tuple.

PS

| PERSON_NAME | STARSHIP |
|----------------|--------------|
| James Kirk U | Enterprise U |
| Mr. Spock U | Null U |
| Tim McKelley U | Apollo U |

SOD

| STARSHIP | OBJECTIVE | DESTINATION |
|--------------|---------------|-------------|
| Enterprise U | Exploration U | Talos U |
| Voyager U | Null U | Null U |

Figure 3.3: Unclassified Instances of PS and SOD Relations

Example 6. Consider the relations SOD and PS as shown in Figure 3.2. An Unclassified user sees the two relations as given in Figure 3.3. For the Unclassified user, the referential integrity property has been violated because there is no Starship “Apollo” visible to the Unclassified user. From the Unclassified user’s point of view either Starship “Apollo” does not exist, which is a referential integrity error, or it references a classified Starship that is not displayed to the user, which is a security error.

3.6 ENFORCEMENT OF INTEGRITY CONSTRAINTS

Most of the research work on the topics covered in this document has either focused on polyinstantiation as a mechanism to deal with primary key requirements and entity integrity or has dealt with the broader question of inference. Meadows did some early work on the conflicts of integrity and security that mainly considered approaches for maximizing integrity while eliminating inference channels [Meadows 88]. Polyinstantiation and inference research are covered in the appropriate TDI companion documents. However, there are several works dealing with aspects of database integrity which go beyond just entity and referential integrity and do not fit into those areas.

Notargiacomo and Wiseman have explored separation of duty as an approach to protect data from unauthorized modification within MLS databases [Notargiacomo 91; Wiseman 90]. The concept of applying separation of duty to integrity enforcement is to ensure that, before a modification to the state is made, sufficient people agree that it properly reflects the real world requirement. The fact that certain users are unlikely to collude with a particular change, an underlying assumption, is indicated by the way roles are assigned to users. Notargiacomo explores the issues of integrating an interpretation of the Clark-Wilson integrity model [Clark 87, 88] into a multilevel DBMS security policy. Her preliminary investigation raised issues on object granularity; interaction of MAC, DAC, and integrity policies; application of Clark-Wilson type controls to classes of users; and the need for nested transactions. Wiseman sees the high level abstract view of separation of duty as several people agreeing to change state. Integrity can be encouraged by enforcing constraints which ensure the state is always valid and by subjecting inputs to separation of duty.

Reind van de Reit and Beukering describe how constraints involving integrity and security can be specified in MOXUM, their active object-oriented knowledge-base system [van de Reit 94]. Their premise is that changes to the database should be governed by rules in the form of integrity constraints and security checks. In this case, integrity constraints refer to the quality of the data, while security constraints refer to protection and access rights. Both kinds of constraints are specified in MOKUM as Prolog predicates and are not separated. Each time some manipulation on an object or a collection mentioned in a constraint is performed (create/change/inspect/destroy), some check has to be performed. The Prolog scripts regulate access to object collections and enforce integrity constraints through triggers. This implies a need to extend the Trusted Computing Base to include the Prolog compiler which is problematic with respect to assurance.

In addition to considering the entity and referential integrity conflicts with security, Maimone and Allen provide alternatives to resolving problems that arise in maintaining transaction integrity and value constraints [Maimone 91]. A security conflict with transaction integrity may arise if a single transaction is designed so that some steps must perform write operations at different security

levels. Value constraints are defined integrity rules that may restrict the valid values for a data element. A conflict arises when some existing data classified above the label of the user activating the constraint does not meet constraints defined after object creation. They make the distinction between the use of triggers and constraints for enforcing integrity rules. The utility of declarative constraints, including primary and foreign key and value or domain constraints, is that the constraint declares an invariant property, detailing the meaning of a consistent state.

Marks and Jajodia [Jajodia 94] have developed a technique to enforce certain integrity constraints and minimize the amount of sensitive information disclosed. They transform multilevel constraints into constraints that can be enforced at individual levels and approximate enforcement of the multilevel constraint.

SECTION 4

COVERT CHANNELS

The B2 or higher MAC requirements of the TCSEC specify that an MLS system must additionally guard against *indirect accesses* through information flows. It is recognized that some covert channels will exist in any system. The TCSEC provides guidelines on what actions must be taken for channels depending on the bandwidth of the channel. Enforcement of integrity constraints can create covert channels. These channels are discussed in this section. Interpretations indicating precisely what channels will be deemed acceptable for systems evaluated against various TCSEC levels will be developed as the Evaluation Community deals with specific vendor implementations.

4.1 PRIMARY KEY UNIQUENESS

Problems arise in multilevel relations when a user at a lower security level tries to enter a tuple with the same primary key value as that of an existing tuple at a higher security level or when a user tries to update the primary key attribute to a value that already exists at a higher level. This insert/update cannot be allowed if primary key uniqueness is to be preserved. On the other hand, if the user is not allowed to insert the primary key value, then there exists a covert channel. The following example provides a concrete illustration.

Example 7. Consider the multilevel relation schema SOD of Example 4 in Section 3.2; the actual relation is the same as the Secret instance shown in Figure 3.3. Suppose the Unclassified user, who sees the Unclassified instance in Figure 3.3, attempts to update the missing values in the second tuple corresponding to the starship “Voyager.” If the user is allowed to do this update, then there will be two tuples corresponding to the primary key value of Starship “Voyager” in the Secret instance, leading to a violation of the primary key constraint, as shown in Figure 4.1. But if the Unclassified user’s attempt is rejected, then there is a channel. Since this is unacceptable, a way must be found to deal with the existence of both tuples in the Secret instance.

| STARSHIP | OBJECTIVE | DESTINATION |
|--------------|---------------|-------------|
| Enterprise U | Exploration U | Talos U |
| Voyager U | Spying S | Mars S |
| Voyager U | Exploration U | Moon U |

Figure 4.1: SOD After Insertion of Tuple Corresponding to Starship = “Voyager” by Unclassified User

Since the user can infer that a Secret tuple exists with the same primary key the user is attempting to insert, this channel can be viewed as an example of inference. However, it also represents a covert channel because a Secret process can insert or delete tuples with a given primary key and this action could be detected by the Unclassified user. To avoid this channel, both the Secret tuple

and the Unclassified tuple for the starship “Voyager” must somehow co-exist in the instance at the Secret level.

These security considerations have led to the notion of polyinstantiation [Denning 87]. Polyinstantiation forces a relation to contain multiple tuples with the same primary key but distinguishable by their classification levels or by the non-primary key attributes of the relation [Lunt 91]. It is because the user designated primary key must be augmented with classification levels to maintain primary key uniqueness in a polyinstantiated database that we have used Denning’s term apparent primary key for the user-specified primary key. Jajodia contends that the use of polyinstantiation for cover stories needs to be confined to those cases where it is intended since uncontrolled use of polyinstantiation leads to further loss of integrity and confusion with the semantics of the relational data model [Jajodia 90].

The debate continues as to whether polyinstantiation is needed in multilevel relations or not [Sandhu 91]. If polyinstantiation is not required in multilevel relations, then there must be a solution to close channels. If polyinstantiation is used, then the question is how polyinstantiation should be managed. Burns maintains that as a result of the fundamental conflict with enforcing confidentiality and secrecy of information in an MLS database with integrity, DBMS products will have to be expanded to include the ability to correct the errors and inconsistencies introduced by polyinstantiation [Burns 90a]. A thorough discussion of all the issues associated with polyinstantiation is beyond the scope of this document, but is covered in another volume in this series: *Polyinstantiation Issues in Multilevel Secure Database Management Systems* [Poly 96]. For the purpose of this document, the discussion of polyinstantiation is confined to the impact it has on referential integrity.

4.2 ENFORCEMENT OF REFERENTIAL INTEGRITY RULES

Section 3 showed that for referential integrity in an MLS database, the security level of the foreign key must dominate the security level of the primary key of the referenced tuple ($C[FK] \geq C[PK]$). This requirement guarantees that the referenced primary key will be visible to any user who sees the foreign key as non-null. However, the situation is somewhat more complicated if the referenced relation is polyinstantiated. In this event, there may be more than one tuple with an apparent primary key matching FK. Some method must be used to uniquely and consistently determine which primary key will be considered to be the referenced primary key. One approach is to augment FK with the security level of its attributes (the tuple security level if tuple level labeling is used). If that approach is used, a matching PK will have the same security level as FK and the requirement on the relationship between security levels of PK and FK becomes $C[FK] = C[PK]$.

Enforcement of referential integrity through the rules described in Section 2.2 can result in covert channels. These channels are discussed below. The discussion is divided into the two permitted cases: $C[FK] = C[PK]$ and $C[FK] > C[PK]$.

4.2.1 Enforcement of Integrity Rules When $C[FK] = C[PK]$

To study the enforcement of integrity rules when the security level of the referencing tuple and the security level of the referenced tuple are the same, we need to look at the cases of the four different levels of granularity in multilevel relations. For relations with relation level labeling or tuple level

labeling, all the integrity rules are usable without modification because this case can be considered to be equivalent to relationships occurring in a single-level database. For attribute and element level labeling there is a potential channel for the CASCADFS-delete rule. All other rules can be enforced without any problem. This channel can be better explained with the help of an example.

Example 8. Consider the two relations SOD and PS as given in Figure 4.2. The security level of the foreign key in relation PS in this example is the same as the security level of the referenced primary key in relation SOD. For the second tuple in relation PS, an Unclassified user only sees the attribute Person_Name. The user is not cleared to see the value of Starship for this tuple and also the user cannot see the corresponding referenced tuple in relation SOD. Suppose a Confidential user comes in and attempts to delete the second tuple in relation SOD where Starship = "Saratoga." If the delete rule specified is a CASCADES-delete, then the referencing tuple in PS is also deleted. An Unclassified user now will not see the attribute value Person_Name = "Mr. Spock" any more, and this causes a channel.

PS

| PERSON_NAME | STARSHIP |
|--------------|--------------|
| James Kirk U | Enterprise U |
| Mr. Spock U | Saratoga C |

SOD

| STARSHIP | OBJECTIVE | DESTINATION |
|--------------|---------------|-------------|
| Enterprise U | Exploration C | Talos C |
| Saratoga C | Mining C | Rigel C |

Figure 4.2: Relation PS and SOD when C[FK] = C[PK]

The channel in this case can be avoided if the equivalence of the security levels of the foreign key and the referenced primary key is extended. If the security level of the entire referencing tuple is the same as the security level of the entire referenced tuple, then no violations occur when enforcing the insert, delete, and update rules. This fact has been acknowledged in the SeaView model [Lunt 90] and the referential secrecy model given by Burns [Burns 90b]. Burns also suggests selective enforcement of referential integrity in those cases where the security level of both the referencing relation and the referenced relation are not the same, assuming that the channel either can be monitored or is deemed not to cause a serious threat.

4.2.2 Enforcement of Integrity Rules When $C[FK] > C[PK]$

Now the final case will be considered where the security level of the referencing tuple dominates the security level of the referenced tuple. Each of the integrity rules for insert, delete, and update will be considered individually, and the effect of the dominance of the referencing tuple's security level over that of the referenced tuple will be investigated. The effect of different levels of labeling granularity in multilevel relations upon enforcement of integrity will also be considered. Throughout this discussion, we will assume that either the relations are not polyinstantiated or a consistent method is employed to determine the unique primary key referred to by a foreign key.

1. *Element-Level Granularity* - Each of the integrity rules for insert, delete, and update is examined individually for relations with element-level granularity, and conditions violating the security constraints are investigated.
 - a. Delete Rule - As discussed in Section 2.2.1, the four delete rules for referential integrity in single-level relations are as follows:
 - i. RESTRICTED-delete Rule
 - ii. CASCADES-delete Rule
 - iii. NULLIFIES-delete Rule
 - iv. SET-DEFAULT-delete Rule

Each of these rules is considered in turn for applicability to multilevel relations labeled at the element level.

- i. RESTRICTED-delete Rule - The RESTRICTED-delete rule states that the referenced primary key tuple cannot be deleted as long as there is a corresponding referencing tuple somewhere in the database. For instance, consider the relations SOD and PS given in Figure 4.3.

In relation PS, the security level of the foreign key value Starship = "Voyager" dominates the security level of the primary key value in the relation SOD. According to the RESTRICTED-delete rule, as long as there is a tuple having foreign key value "Voyager" in PS, the tuple with primary key value "Voyager" cannot be deleted from the relation SOD. This gives rise to a channel, as there is a downward flow of information from the high-level referencing foreign key to the low-level subject attempting to delete the tuple containing the primary key. Therefore, the RESTRICTED-delete Rule violates secrecy when the security level of the foreign key dominates the security level of the primary key of the referenced tuple.

- ii. CASCADES-delete Rule - If the delete rule specified is the CASCADES-delete rule, then if the tuple in SOD with Starship = "Voyager" is deleted at the Unclassified level, then the tuple in PS with Starship = "Voyager" at the Secret level should also be deleted. In the relation instance of PS, given in Figure 4.3, Person_Name = "Mr. Spock" is at the Unclassified level in the tuple containing Starship = "Voyager" at the Secret level. Although the Unclassified user would not see any value for Starship in

the PS tuple with Person_Name = “Mr. Spock,” this tuple will be deleted as a result of the CASCADES-delete action. Therefore, the user will be able to infer that the tuple contained Starship = “Voyager.” This is a channel. If the attribute value “Mr. Spock” were classified as Secret, there would have been no channel. Hence, the CASCADES-delete rule fails in relations with element-level labeling and $C[FK] > C[PK]$, if any non-foreign key attribute value in the referencing tuple is at a lower security level than the foreign key attribute value. Even though the CASCADES-delete rule is not invalid for all the cases, channels exist that cannot be covered or be removed. Hence, CASCADES-delete cannot be used for defining an integrity constraint in relations with element-level labeling which allow a foreign key to strictly dominate its corresponding primary key.

SOD

| STARSHIP | OBJECTIVE | DESTINATION |
|--------------|---------------|-------------|
| Enterprise U | Exploration U | Talos U |
| Voyager U | Spying S | Mars S |
| Apollo S | Spying S | Moon S |

PS

| PERSON_NAME | STARSHIP |
|--------------|--------------|
| James Kirk U | Enterprise U |
| Mr. Spock U | Voyager S |

Figure 4.3: Relations SOD and PS without Polyinstantiation

- iii. NULLIFIES-delete Rule - If the delete rule specified is NULLIFIES-delete, then the value of Starship = “Voyager” in relation PS is set to null when the tuple in SOD for Starship = “Voyager” is deleted. This does not conflict with the security constraints, as long as *write-ups* are allowed, since there would not be a downward flow of information.
- iv. SET DEFAULT-delete Rule - The argument for the SET DEFAULT-delete rule is the same as for the NULLIFIES-delete rule. As previously noted, the SET DEFAULT rule is equivalent to the NULLIFIES rule when null is specified as the default.

In summary, when the foreign key value is higher than the security level of the primary key value, then the NULLIFIES-delete rule and the SET DEFAULT-delete do not create downward information flows. However, the RESTRICTED-delete and CASCADES-delete rules can result in covert channels.

- b. UPDATE Rule - As discussed in Section 2.2.2, the four update rules for referential integrity in single-level relations are as follows:
 - i. RESTRICTED-update Rule
 - ii. CASCADES-update Rule
 - iii. NULLIFIES-update Rule
 - iv. SET DEFAULT-update Rule

The following paragraphs discuss the application of the update rules to multilevel relations labeled at the element level.

- i. RESTRICTED-update Rule - Consider the relations SOD and PS given in Figure 4.2. In relation PS, the security level of the foreign key value Starship = "Voyager" dominates that of the primary key value in the relation SOD. According to the RESTRICTED-update rule, as long as there is a tuple having foreign key value "Voyager" in PS, the primary key value "Voyager" cannot be updated in the relation SOD. This gives rise to a channel because there is flow of information from the high-level referencing foreign key to the low-level subject attempting to delete the tuple containing the primary key. Therefore, the RESTRICTED-update Rule will not work when updating a referenced tuple if the security level of the referencing tuple dominates the security level of the referenced tuple.
- ii. CASCADES-update Rule - The CASCADES-update Rule states that if the primary key value Starship = "Voyager" in SOD is updated, then the foreign key value Starship = "Voyager" in PS will also be updated. This does not conflict with the security constraints, as long as write-ups are allowed, since there would not be a downward flow of information.
- iii. NULLIFIES-update Rule - The NULLIFIES-update Rule specifies that the value of Starship = "Voyager" in relation PS be set to null if Starship = "Voyager" in SOD is updated. Again, this does not conflict with the security constraints, as long as write-ups are allowed, since there would not be a downward flow of information.
- iv. SET DEFAULT-update Rule - The SET DEFAULT-update Rule does not violate any security or integrity constraints when the security level of the foreign key dominates the security level of the primary key. The explanation is the same as for the NULLIFIES-update rule.

In summary, CASCADES-update, NULLIFIES-update, and SET DEFAULT-update do not result in downward information flow. The RESTRICTED-update rule can result in a covert channel.

- c. INSERT Rule - The insert rule for integrity states that the insertion of a foreign key value should comply with the referential integrity rules; that is, each foreign key value in the referencing relation should have an identical primary key value in the referenced relation. In MLS databases, it is important to ensure that there is no downward flow of

information when the insertion is made. If the security level of the referencing tuple (foreign key value) dominates the security level of the referenced tuple (primary key value), then there is no such possibility of a channel. If a higher user attempts to insert a foreign key value, the user's insertion is accepted or rejected based on the presence or absence of the referenced value at the lower security level. There will be only an upward flow of information, which means that both integrity and security rules are satisfied. It should be noted that, unlike with polyinstantiated relations, there is no confusion while inserting the tuples in the two relations.

SOD

| No. | STARSHIP | OBJECTIVE | DESTINATION | TC |
|-----|------------|-------------|-------------|----|
| 1 | Enterprise | Exploration | Talos | U |
| 2 | Enterprise | Spying | Mars | S |

PS

| No. | PERSON_NAME | STARSHIP | TC |
|-----|-------------|------------|----|
| 3 | James Kirk | Enterprise | U |
| 4 | Mr. Spock | Enterprise | S |

Figure 4.4: Relations with Tuple-Level Granularity

2. *Tuple-Level Granularity* - The argument for referential integrity control in relations with tuple-level granularity is the same as the discussion above for element-level granularity. The only difference is that instead of each data element having an individual security level, the entire tuple in the relation is assigned a security level as shown in Figure 4.4. As is the case for element-level granularity, a security conflict occurs when the RESTRICTED-update/delete option is enforced. There is a downward flow of information when a low user tries to update/delete a target value, and is allowed to do so depending on the absence of a matching referencing tuple at a higher level. All other integrity rules are valid for relations with tuple-level granularity. The problem with CASCADES-delete that occurs for element-level granularity does not occur with tuple-level granularity because, for tuple-level granularity, each element in the tuple is explicitly assumed to have the same security level as defined in the tuple class. As such, a user will never be in the situation where he can view some attributes of a tuple but not others and then witness the tuple vanish due to a cascading delete on attributes that the user is not cleared to see, which would be a covert channel due to the inference of the high attribute's value.

3. *Attribute-Level Granularity* - As described earlier, attribute-level granularity means that each attribute in a relation is assigned a single security level. All values of an attribute are classified at the same level of security across all tuples in the relation. From the discussion in Section 3.5, to maintain referential integrity, the security level of the foreign key attribute in the referencing relation must dominate the security level of the primary key attribute in the referenced relation.

The case to be considered is where the security level of the foreign key attribute strictly dominates the security level of the apparent primary key attribute. In this case, the relations SOD and PS are as given in Figure 4.5.

SOD

| No. | STARSHIP U | OBJECTIVE S | DESTINATION U |
|-----|------------|-------------|---------------|
| 1 | Enterprise | Exploration | Talos |
| 2 | Voyager | Spying | Mars |

PS

| No. | PERSON_NAME U | STARSHIP S |
|-----|---------------|------------|
| 3 | James Kirk | Enterprise |
| 4 | Mr. Spock | Enterprise |

Figure 4.5: Relations with Attribute-Level Granularity

Following the discussion earlier, it can be observed that all the integrity rules can be applied to the case above except for the RESTRICTED-delete or the RESTRICTED-update rule. Under RESTRICTED-update or delete, whenever a lower subject tries to update or delete the tuple containing a particular primary key attribute value, the attempt is accepted or rejected based on the presence of a matching value in the foreign key attribute at the higher security level. This gives rise to a channel.

4. *Relation-Level Granularity* - As defined earlier, a relation with relation-level granularity has a single classification level defined for the entire relation, or in other words, all data elements in the relation are at the same security level.

For instance, consider the relation schema SOD, given in Example 4, and the relation schema PS, given in Example 6. Relation PS references relation SOD, with Starship being the foreign key attribute. The security level of the primary key “Starship” (i.e., C[PK]) is the same as the security level of the relation SOD, and the security level of the foreign key “Starship” (i.e., C[FK]) is the same as the security level of the relation PS. Based on the restrictions derived in rule 2 in Section 3.5, the security level of the foreign key should

always dominate the security level of the primary key. Therefore, for relation-level granularity, the security level of the relation PS should dominate the security level of the relation SOD.

Let the security level of relation SOD be Unclassified and of PS be Secret. Then $C[FK] = \text{Secret}$ and $C[PK] = \text{Unclassified}$ (i.e., $C[FK] > C[PK]$). From the discussion in Section 3.5, it can be seen that, if $C[FK] > C[PK]$, then all integrity rules except RESTRICTED-delete and RESTRICTED-update can be used, without any covert channels.

The table below gives a summary of the integrity rules that apply to different levels of granularity without compromising secrecy by permitting downward information flows. In the case of polyinstantiated relations it is necessary in all cases that there be a method to unambiguously determine which instance of the apparent key is referenced.

| Level of Granularity | Element | Tuple | Attribute | Relation |
|----------------------|---------|---------|-----------|----------|
| Insert Rule | OK | OK | OK | OK |
| Delete Rules: | | | | |
| RESTRICTED-delete | Channel | Channel | Channel | Channel |
| CASCADES-delete | Channel | OK | OK | OK |
| NULLIFIES-delete | OK | OK | OK | OK |
| SET-DEFAULT-delete | OK | OK | OK | OK |
| Update Rules: | | | | |
| RESTRICTED-update | Channel | Channel | Channel | Channel |
| CASCADES-update | OK | OK | OK | OK |
| NULLIFIES-update | OK | OK | OK | OK |
| SET-DEFAULT-update | OK | OK | OK | OK |

Table 4.1: Summary of Covert Channels in Referential Integrity Rules for $C[FK] > C[PK]$

4.3 RELATION TO DBMS ARCHITECTURE

A Trusted Computing Base (TCB) consists of one or more sub-elements that together enforce a security policy over a system. Complex systems distribute policy enforcement to various sub-elements. While determining the trust characteristics of a complex system on the basis of a collection of subparts is not well understood, there are approaches for arguing about composition of systems.

The approach used in the TDI is to view the system as being composed of hierarchically-ordered systems. There is a “privilege” hierarchy characterized by dependency. The construct developed for dealing with layered systems is TCB subsets. A TCB subset enforces a security policy over a set of resources for that layer, and may or may not include hardware. If a TCB subset can be shown to be constrained and unprivileged relative to the more primitive TCB subset from which it obtains resources, then the scope of the covert channel analysis can be limited. An example of this approach is the SeaView prototype.

SRI International developed the SeaView prototype multilevel RDBMS to validate the SeaView theoretical model and to demonstrate that the prototype is suitable for engineering development [Hsieh 93]. The design approach provides users with element-level labeling. SeaView constructs multilevel relations as views over stored single-level relations. The single-level relations are stored in files managed by an underlying multilevel operating system. Thus, individually labeled data elements need not be stored in individually labeled storage objects. The SeaView approach allows multilevel database operations to be decomposed into corresponding single-level operations on the single-level relations. The decomposition is transparent to the user, who considers the multilevel relations to be stored relations. Thus, the SeaView model extends entity and referential integrity to multilevel relations. It also allows application dependent integrity rules to be defined on multilevel relations; and it ensures that updates of multilevel relations are well defined. In addition, the SeaView model constrains multilevel relations with polyinstantiation integrity, which specifies consistency for polyinstantiated tuples and elements.

In the SeaView approach, the underlying architecture guards against covert channels during enforcement of entity and referential integrity. This is because the DBMS instance operating on behalf of a subject is untrusted with respect to the underlying operating system MAC policy (i.e., it can only read data with a security level dominated by the subject’s level and can only write data with a security level dominating the subject’s level). SRI utilized Trusted Oracle OS MAC Mode, which has been evaluated as a constrained TCB subset architecture [Oracle 94].

On the other hand, a secure DBMS can be built with MAC enforcement done in the DBMS, not just the underlying operating system. In this approach, the DBMS will have the privilege to violate the operating system’s MAC controls. With this architecture, called a trusted subject architecture, the DBMS designer has to make a tradeoff between channel free enforcement of entity and referential integrity constraints, or enforcement of the constraints in a manner which allows selected channels. In the latter case, the constraint enforcement mechanism must be carefully analyzed to make sure that the downward information flows are thoroughly understood and that they are acceptable for the target environment or evaluation class. For example, Trusted Oracle7 DBMS MAC Mode (which uses the trusted subject architecture) was successfully evaluated at Class B1 [Oracle 94]. In DBMS MAC Mode, entity and referential integrity can be enforced. An audit event was added to Trusted Oracle7 to detect use of potential covert channels.

SECTION 5

SUMMARY

Entity integrity and referential integrity are two important integrity constraints that should be enforced by the DBMS. While entity integrity is necessary for preserving correctness of data within a relation, referential integrity is required for maintaining interrelation integrity in the relational data model. In this document, these concepts have been extended to multilevel DBMSs.

The extension of the concept of referential integrity from single-level relations to multilevel relations is not straightforward. This is because restrictions are needed to provide referential integrity control in multilevel DBMSs without compromising secrecy. The basic requirement for referential integrity is that each referencing foreign key value must have an identical target primary key value in the referenced relation. An additional requirement for multilevel relations is that each foreign key and primary key should be uniformly classified (i.e., all attributes included in a key should have the same security level) and each foreign key dominates its referenced primary key.

From the discussion in the document, it can be concluded that enforcing referential integrity, when the security level of the foreign key is equal to the security level of the referenced primary key, is simple and without any ambiguity. All integrity rules apply in this case, whether the relations allow polyinstantiation or not. When polyinstantiation is allowed, some method such as including the security level of the primary and foreign key values as part of the key must be used to differentiate references, and allow the referential integrity rules to be enforced.

Referential integrity completely fails when the security level of the foreign key does not dominate the security level of the primary key. When the security level of the foreign key strictly dominates the security level of the referenced primary key, some of the integrity rules for actions to be taken upon deletion or modification of a key value can be applied without channels, with some differences based on labeling granularity.

Inclusion of a mechanism which enforces integrity across security levels in Class B2 and above DBMSs may not be feasible due to conflicts with the covert channel requirements. However, with the successful evaluation of Trusted Oracle7 DBMS MAC Mode with an unmodified integrity mechanism, precedence has been set for Class B1 DBMSs.

REFERENCES

- [ANSI 92] ANSI. Database Language SQL2. In ANSI X3.135-1992, American National Standards Institute, New York. December 1992.
- [ANSI 94] ANSI. SQL3 - Working Draft. Draft Version, Available From X3 Secretariat, Computer and Business Equipment Manufacturers Association (CBEMA), 1250 Eye St. N.W., Suite 200 Washington, D.C. 20005-3992. September 1994.
- [Audit 96] National Computer Security Center, *Auditing Issues in Secure Database Management Systems*, NCSC Technical Report-005, Volume 4/5, May 1996.
- [Bell 76] D. E. Bell and L. J. LaPadula. *Secure Computer Systems: Unified Exposition and Multics Interpretation*, The MITRE Corporation. March 1976.
- [Burns 90a] R. K. Burns. Integrity and Secrecy: Fundamental Conflicts in the Database Environment. In *Proceedings of the Third RADC Database Security Workshop*, Castille, NY. 1990.
- [Burns 90b] R. K. Burns. Referential Secrecy. In *Proceedings of the IEEE Symposium on Security and Privacy*, IEEE Computer Society Press. May 1990.
- [Clark 87] D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA. May 1987.
- [Clark 88] D. D. Clark and D. R. Wilson. Evolution of a Model for Computer Integrity. In *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD. October 1988.
- [DAC 96] National Computer Security Center, *Discretionary Access Control Issues in High Assurance Secure Database Management Systems*, NCSC Technical Report-005, Volume 5/5, May 1996.
- [Date 86] C. J. Date. *An Introduction to Database Systems*, Vol. I, 4th Edition, Addison-Wesley, Reading, MA. 1986.
- [Date 90] C. J. Date. Referential Integrity and Foreign Keys: Further Considerations. In *Relational Database Writings 1985-1989*, Addison-Wesley, pp. 99-184. 1990.
- [Denning 87] D. E. Denning, T. F. Lunt, R. R. Schell, M. Heckman, and W. R. Shockley. "A Multilevel Relational Data Model," *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, pp. 220-234. 1987.
- [DoD 85] Department of Defense *Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, Washington, DC. December 1985.

- [Hsieh 93] D. Hsieh, T. Lunt, and P. Boucher. *The SeaView Prototype*. RL-TR-93-216 Final Technical Report, SRI International. November 1993.
- [Horowitz 92] B. M. Horowitz. A Run-Time Execution Model for Referential Integrity Maintenance. In *Proceedings of the 8th International Conference on Data Engineering*, Tempe, AZ, pp. 548-556. February 1992.
- [Inference 96] National Computer Security Center, *Inference and Aggregation Issues in Secure Database Management Systems*, NCSC Technical Report-005, Volume 1/5, May 1996.
- [Jajodia 90] S. Jajodia, and R. Sandhu. Polyinstantiation Integrity in Multilevel Relations. In *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, Oakland, CA. May 1990.
- [Jajodia 91] S. Jajodia, and R. Sandhu. Toward a Multilevel Secure Relational Data Model. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Denver, CO, pp. 50-59. May 1991.
- [Jajodia 94] S. Jajodia and D. G. Marks. Maintaining Secrecy and Integrity in Multilevel Databases: A Practical Approach. In *Proceedings of the 18th National Information Systems Security Conference*, Baltimore, MD. Oct. 10-13 1995.
- [Khoshafian 86] S. N. Khoshafian and G. P. Copeland. Object Identity. In *Proceedings for OOPSLA 1986*. pp. 406-416. September 1986.
- [Lunt 90] T. F. Lunt, D. E. Denning, R. R. Schell, M. Heckman, and W. R. Shockley. The SeaView Security Model. In *IEEE Transactions on Software Engineering*, Vol. 16, No. 6, pp. 593-607. June 1990.
- [Lunt 91] T. F. Lunt, and D. Hsieh. Update Semantics for a Multilevel Relational Database System. In *Database Security, IV: Status and Prospects*, S. Jajodia and C. E. Landwehr (editors), Elsevier Science Publishers B. V. (North Holland), IFIP, pp. 281-296.1991.
- [Maimone 91] B. Maimone and R. Allen. Methods for Resolving the Security vs. Integrity Conflict. In *Research Directions in Database Security, IV: Proceedings of the Fourth RADC Workshop*, Rae Burns (editor). April 1991.
- [Markowitz 90] V. M. Markowitz. Referential Integrity in Relational Database Management Systems: A Comparative Study. Preprint. 1990.
- [Meadows 88] C. Meadows and S. Jajodia. Integrity Versus Security in Multi-Level Secure Databases. In *Database Security Status and Prospects*, ed. C. Landwehr, North Holland. 1988.

- [Notargiacomo 91] L. Notargiacomo. Database Integrity: Based on the Clark-Wilson Integrity Model. In *Proceedings of the 3rd RADC Database Security Workshop*. 1991.
- [Oracle 94] *Oracle7 and Trusted Oracle7 Final Evaluation Report*, NCSC-EPL-94/004, C-Evaluation Report No. 07-95, Library No. S242,198, National Computer Security Center, 5 April 1994.
- [Poly 96] National Computer Security Center, *Polyinstantiation Issues in Multilevel Secure Database Management Systems*, NCSC Technical Report-005, Volume 3/5, May 1996.
- [Sandhu 91] R. Sandhu and S. Jajodia. Honest Databases That can Keep Secrets. In *Proceedings 14th National Computer Security Conference*, Washington, D.C. Oct. 1991.
- [TDI 91] *Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-021, National Computer Security Center. April 1991.
- [van de Reit 94] R. van de Reit and J. Beukering. The Integration of Security and Integrity Constraints in MOKUM. In *Proceedings of the 8th IFIP 11.3 WG on Database Security*. Bad Sdzdetfurth, Germany, 23-26 August 1994.
- [Wiseman 90] S. Wiseman. The Control of Integrity in Databases. In *Proceedings of the 4th IFIP 11.3 WG on Database Security*, Halifax, England, 18-21 September 1990.

| REPORT DOCUMENTATION PAGE | | | Form Approved OMB No. 0704-0188 | |
|--|--|---|--|----------------|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503. | | | | |
| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE May 1996 | 3. REPORT TYPE AND DATES COVERED Final | | |
| 4. TITLE AND SUBTITLE Entity and Referential Integrity Issues in Multilevel Secure Database Management Systems | | | 5. FUNDING NUMBERS | |
| 6. AUTHOR(S) | | | | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Security Agency Attn: V21, Partnerships and Processes Fort George G. Meade, MD 20755-6000 | | | 8. PERFORMING ORGANIZATION REPORT NUMBER NCSC Technical Report - 005 Volume 2/5 Library No. S-243,039 | |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | | | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER | |
| 11. SUPPLEMENTARY NOTES | | | | |
| 12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release Distribution Unlimited | | | 12b. DISTRIBUTION CODE | |
| 13. ABSTRACT (<i>Maximum 200 words</i>) This report is the second of five companion documents to the <i>Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria</i> . The companion documents address topics that are important to the design and development of secure database management systems, and are written for database vendors, system designers, evaluators, and researchers. This report addresses entity and referential integrity issues in multilevel secure database management systems. | | | | |
| 14. SUBJECT TERMS Entity Integrity; Referential Integrity; Multilevel Secure Database Management Systems | | | 15. NUMBER OF PAGES 45 | 16. PRICE CODE |
| 17. SECURITY CLASSIFICATION OF REPORT Unclassified | 18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified | 19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified | 20. LIMITATION OF ABSTRACT | |