



COMPLIANCE COMPONENT

DEFINITION	
<i>Name</i>	Database Management Systems – Integrity
<i>Description</i>	<p>Data Integrity is an umbrella term that refers to the consistency, accuracy, and correctness of data stored in a database. Data integrity is not about physical security, fault tolerance, or data preservation (backups). Think of data integrity in terms of the old adage: 'garbage in, garbage out'. Data integrity is about keeping the garbage out. There are four primary types of data integrity: entity, domain, referential, and user-defined. In general terms; entity integrity applies at the row level; domain integrity applies at the column level; and referential integrity applies at the table level.</p> <ol style="list-style-type: none"> 1. Entity integrity ensures that each row in the table is uniquely identified. In other words, entity integrity ensures a table does not have any duplicate rows. 2. Domain integrity requires that a set of data values fall within a specific range (domain) in order to be valid. In other words, domain integrity defines the permissible entries for a given column by restricting the data type, format, or range of possible values. 3. Referential integrity is concerned with keeping the relationships between tables synchronized. 4. User-defined integrity refers to specific business rules not covered by the other integrity categories. It is typically implemented through triggers and stored procedures. <p>Data integrity is enforced by features such as check constraints, triggers, views, stored procedures, user defined functions, and/or referential constraints.</p>
<i>Rationale</i>	Maintaining Integrity of the data is a cornerstone function of the DBMS. Integrity constraints that the DBMS maintains are the business rules that are defined by the enterprise. The DBMS should have the capability to enforce data integrity for all the applications that use the data.
<i>Benefits</i>	Data integrity rules defined centrally in the database are independent from the applications that use the database. When data integrity is implemented directly on the database, the application developer does not need to worry about coding all of the data integrity features directly into the application. Potential problems associated with coding data integrity into the application are 1) additional programming required; 2) inconsistencies and potential errors; 3) it is difficult to make changes; and, 4) weaker security (i.e., it is easier to bypass).
ASSOCIATED ARCHITECTURE LEVELS	
<i>Specify the Domain Name</i>	Information
<i>Specify the Discipline Name</i>	Database Management
<i>Specify the Technology Area Name</i>	Database Management Systems
<i>Specify the Product Component Name</i>	Relational Database Management Systems
COMPLIANCE COMPONENT TYPE	
<i>Document the Compliance Component Type</i>	Standard
<i>Component Sub-type</i>	

COMPLIANCE DETAIL

Entity Integrity

Entity integrity ensures that each row in a table has a unique identifier that allows one row to be distinguished from another. Placing a primary key (PK) constraint on a specific column (although it can also be enforced with a UNIQUE constraint, a unique index, or the IDENTITY property) most often enforces entity integrity. The PK constraint forces each value inserted into a column (or combination of columns) to be unique; if a user attempts to insert a duplicate value into the column(s), the PK constraint will cause the insert to fail. A PK will not allow any Nulls to be inserted into the column(s) (A NULL entry would be disallowed even if it would be the only NULL in the column and therefore unique.). A PK is referred to as a 'surrogate key' if the column contains no real data other than a uniqueness identifier. If 'real' data can be used as a PK (e.g., a social security number), then it is referred to as an 'intelligent key'. There can be only one PK per table. A composite PK is a PK that consists of more than one column; it is used when none of the columns in the composite key is unique by itself. Thus, there can be only one PK in a table but the PK can consist of more than one column. If you need to enforce uniqueness on more than one column, use a PK constraint on one column and a UNIQUE constraint or IDENTITY property on any other columns that must not contain duplicates. Non-PK columns on which uniqueness is enforced are referred to as alternative keys or AKs; they get their name from the fact that they are 'alternatives' to the PK and as such, make good candidates for indexing or 'joining' on.

Domain Integrity

A domain in database terminology refers to a set of permissible values for a column (it should not be confused with an Internet or DNS 'domain' or a Windows NT 'domain'). Examples of domain integrity: correct data type; values that fall within the range supported by the system; null status; permitted size values. Domain integrity is sometimes referred to as 'attribute' integrity. Domain Integrity can be enforced with a DEFAULT constraint, FOREIGN KEY, CHECK constraint and data types. Data types limit fields to broad categories (e.g., integers). A default is a definition of a value that can be inserted into a column; a rule is a definition of acceptable values that can be inserted into a column. Rules and defaults are similar to constraints but are not ANSI standard; their continued use is not encouraged.

Referential Integrity

Referential integrity is typically enforced with a Primary Key (PK) and Foreign Key (FK) combination. A Foreign Key (FK) is a column or combination of columns in one table (referred to as the 'child table') that takes its values from the PK in another table (referred to as the 'parent table').

Note that while PK-FK combinations represent logical relationships among data, they do not necessarily limit the possible access paths through the data.

In order for referential integrity to be maintained, the FK in the 'child' table can only accept values that exist in the PK of 'parent' table. The primary objective of referential integrity is to prevent 'orphans;' i.e., records in the child table that cannot be related to a record in the Parent table. Enforcing referential integrity means the relationship between the tables must be preserved when records are added (INSERT), changed (UPDATE), or deleted (DELETE).

Although referential Integrity is often implemented with a PK-FK combination, database developers can also use triggers or stored procedures as well. There are three fundamental approaches to implementing referential integrity: 1) restrict

State the Guideline, Standard or Legislation

	<p>(disallow the data modification); 2) cascade (extend the data modification to related tables); or 3) nullify (set the values of matching FKs to NULL).</p> <p><i>Threats to Referential Integrity</i></p> <p>The UPDATE Threat to Referential Integrity</p> <p>UPDATES can produce orphans when either the PK of the parent is changed or the FK of child is changed. In order to preserve referential integrity, the offending UPDATE can be disallowed; this happens automatically when a FK references a PK. Alternatively, the UPDATE can be 'cascaded' from the parent table to the child table. A third option for dealing the UPDATE threat is to set the FK values to NULL when the PK is changed; this is generally not a good solution.</p> <p>The INSERT Threat to Referential Integrity</p> <p>The INSERT threat only applies to data modifications to the child table. The INSERT threat involves adding records to the child table with no associated record in the parent table; again, the result is orphaned records. There are two ways to preserve referential integrity in the case of an INSERT: The INSERT can be disallowed; this is what happens automatically when a FK references a PK. Alternatively, the FK can be set to null (but, as with the UPDATE threat, this option is generally not a good idea). Note that unlike UPDATES and DELETES, INSERTS cannot be cascaded.</p> <p>The DELETE Threat to Referential Integrity</p> <p>The DELETE threat applies only to data modifications to the parent table. The DELETE threat involves deleting records in the parent table when there are matching records in the child table; as always, the result is orphaned records. Like UPDATES, there are 3 ways to preserve referential integrity with a DELETE the offending DELETE can be disallowed; this happens automatically when a FK references a PK. Alternatively, the DELETE can be 'cascaded' from the parent table to the child table. The third (and bad) option for dealing the DELETE threat is to set the FK values to NULL when the PK is changed.</p> <p>User-Defined Integrity</p> <p>User-defined integrity refers to specific business rules not covered by the other integrity categories. It is typically implemented through triggers and stored procedures.</p>
--	---

<i>Document Source Reference #</i>	N/A
------------------------------------	-----

Compliance Sources			
<i>Name</i>	David R. Frick & Co. , CPA	<i>Website</i>	http://www.frick-cpa.com
<i>Contact Information</i>			
<i>Name</i>	Microsoft Developer Network	<i>Website</i>	http://msdn.microsoft.com
<i>Contact Information</i>			
<i>Name</i>	University of Texas	<i>Website</i>	http://www.utexas.edu/its/windows/database/datamodeling/dm/integrity.html
<i>Contact Information</i>			

KEYWORDS			
<i>List Keywords</i>	Referential integrity, user-defined integrity, domain integrity, entity integrity, data integrity, DB2, SQL Server, Oracle		
COMPONENT CLASSIFICATION			
<i>Provide the Classification</i>	<input type="checkbox"/> <i>Emerging</i>	<input checked="" type="checkbox"/> <i>Current</i>	<input type="checkbox"/> <i>Twilight</i> <input type="checkbox"/> <i>Sunset</i>
<i>Sunset Date</i>			
COMPONENT SUB-CLASSIFICATION			
<i>Sub-Classification</i>	<i>Date</i>	<i>Additional Sub-Classification Information</i>	
<input type="checkbox"/> <i>Technology Watch</i>			
<input type="checkbox"/> <i>Variance</i>			
<input type="checkbox"/> <i>Conditional Use</i>			
Rationale for Component Classification			
<i>Document the Rationale for Component Classification</i>			
Migration Strategy			
<i>Document the Migration Strategy</i>			
Impact Position Statement			
<i>Document the Position Statement on Impact</i>			
CURRENT STATUS			
<i>Provide the Current Status</i>	<input type="checkbox"/> <i>In Development</i>	<input type="checkbox"/> <i>Under Review</i>	<input checked="" type="checkbox"/> <i>Approved</i> <input type="checkbox"/> <i>Rejected</i>
AUDIT TRAIL			
<i>Creation Date</i>	11-29-2004	<i>Date Approved / Rejected</i>	2-8-05
<i>Reason for Rejection</i>			
<i>Last Date Reviewed</i>		<i>Last Date Updated</i>	
<i>Reason for Update</i>			