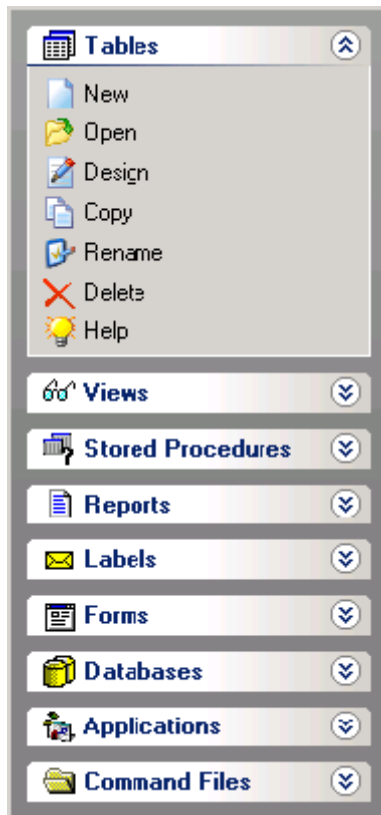
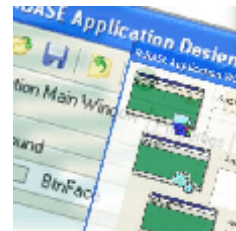
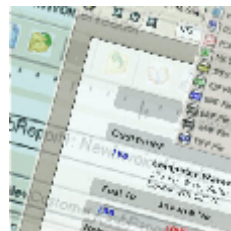
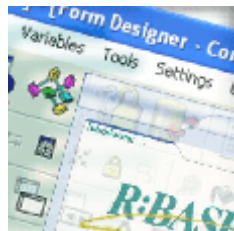
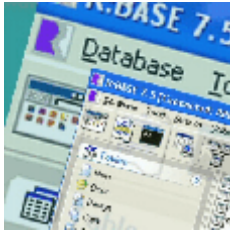


From The Edge:

All About Constraints in R:BASE

(R:BASE 7.5 and V-8 Turbo for Windows)



This is the database you have always wanted!

A. Razzak Memon

August 21, 2006

From the Edge: All About Constraints in R:BASE
Section: Data Integrity
Chapter: Running R:BASE Your Way!
Platform: R:BASE 7.5 and V-8 Turbo for Windows
Builds: R:BASE 7.5 (7.5.25.30814 or higher ...)
R:BASE V-8 Turbo (8.0.12.30814 or higher ...)

Topics:

- Introduction
- Types and Definitions
 - Primary Key
 - Foreign Key
 - Not Null
 - Unique Key
- Benefits of Constraints
- Adding CASCADE to Maintain Primary/Foreign Key Relationship
- Defining Constraints
- Listing Constraints
 - LIST CONSTRAINTS
 - LIST PKEYS
 - LIST PKEYS for tablename
 - LIST FKEYS
 - LIST FKEYS for tablename
 - LIST UKEYS
 - LIST UKEYS for tablename
- Removing Constraints
- Customizing Constraint Violation Messages

Introduction:

One of the strengths of R:BASE is to provide automatic data integrity and referential integrity using constraints. In legacy versions of R:BASE, data integrity and referential integrity had to be enforced through rules and programming. Constraints are a part of the database structure and are automatically enforced.

Types and Definitions:

The constraints that can be defined are *Primary Key*, *Foreign Key*, *Not Null* and *Unique Key*.

Primary Key:

A primary key is the column, or set of columns, that uniquely identifies a row in a table. In other words, it is a set of values that distinguishes one row from another. This can be something like an Employee ID column in an Employee table or it could be a combination of the Customer's ID and the Customer's Phone number. Every table will have a column or set of columns that identify a row, but not every table will have a defined primary key.

- A primary key cannot be defined if any one of the columns included in the desired key already have null or duplicate values.
- Defining a primary key automatically enforces NOT NULL and UNIQUE constraints on the column.
- A primary key definition can be used instead of a rule to prevent duplicates, and will be faster.
- R:BASE automatically builds an index on the specified column(s) when a primary key is defined.

Foreign Key:

Like a primary key, a foreign key is a column or a group of columns. A foreign key matches a defined primary key, i.e. the values in a foreign key reference values in the primary key.

- A primary key can exist without a foreign key, but a foreign key cannot exist without a primary key.
- A foreign key is always defined to reference a primary key.
- A foreign key automatically checks that the values in the foreign key exist in the referenced primary key.
- A foreign key replaces the verification of a value rule.
- An index is automatically built when a foreign key constraint is defined.
- Primary and foreign keys must match in terms of the specified columns. If you have a multi-column primary key, you cannot have single column foreign keys reference it. If you have a multi-column foreign key, it cannot reference a single column primary key.
- When a primary key is defined as more than one column, those columns are treated as a whole. Primary and foreign keys must match exactly. Thus if your primary key is defined as full-text, then the corresponding foreign keys referencing it must also be full-text.
- Primary and foreign keys automatically preserve referential integrity. You cannot delete a row from a table with a defined primary key if there are referenced foreign keys, thus you can never have detail records without a matching master record.
- You can delete a row from a table with a foreign key.
- You cannot add a row to table with a foreign key defined unless the value entered matches a value in the referenced primary key.
- You cannot update a primary key value if there are references, thus you are ensured that linking columns always match, unless a CASCADE option is added for the table with primary key (see *About CASCADE* on page 5).
- You can update primary key values, if there are not matching values in the referenced foreign key.

Not Null:

Placing a not null constraint on a column requires that the data in the column must contain a value; it cannot be null. This prevents users from adding a "blank" value to a table.

- A not null constraint cannot be added if the column already contains null values. The null values must first be edited to actual data values, then the not null constraint can be added. A not null constraint can replace a "Require a value" rule.
- R:BASE does not build an index for a not null constraint, but since it stores the not null as part of the column definition, it is able to check the constraint faster than it could check the rule.

Unique Key:

A unique constraint requires that the data values in the column be unique, i.e. the column cannot contain duplicate values. By definition, the column must also be defined as NOT NULL.

- A unique constraint can replace a "Require a unique value" rule. A unique constraint also automatically builds an index.
- The unique constraint can be defined through the Data Designer (RBDefine) or the R> prompt using either the CREATE TABLE or ALTER TABLE command. For example, ALTER TABLE tblname ALTER COLUMN colname data type NOT NULL UNIQUE

Benefits of Constraints:

- Constraints provide automatic, database-wide data integrity and referential integrity.
- Constraints are quicker and easier than rules.
- The not null and unique constraints restrict data entry. No matter what command is used to enter data, the constraint verifies that the specified column has a value and the value is unique.
- The primary, foreign key constraints provide both data integrity and referential integrity.
- The primary key column is automatically not null and unique.
- Deletions to a table with a defined primary key are automatically restricted if a referenced foreign key is defined.
- A value cannot be entered into a table with a foreign key unless that value exists in the referenced primary key table. In legacy versions of R:BASE, rules could be used to force these constraints. Using rules to enforce these same data constraints required several different rules. You would need a rule to prevent duplicates (replaced by the primary key or unique constraint), verify a value rule (foreign key), and require a value rule (not null). In addition, you would need to define delete rules to prevent rows being deleted from the primary key table if there were matching rows in the foreign key table. If you wanted to prevent users from changing a primary key value that exists in the foreign key table, you would need to define an additional verify a value rule.
- Unlike rules, constraints cannot be turned off, they are always checked for. Constraints also provide much faster performance than rules, but can use more disk space since constraints are enforced by using indexes. Many rules, however, also required the rule column to be indexed for performance.
- It is recommended where possible to use constraints over rules for faster performance and ease of use. Rules are still needed to check other conditions, such as checking for specific values or a range of values.

Adding CASCADE to Maintain Primary/Foreign Key Relationship:

ADD CASCADE maintains the Primary/Foreign key relationships automatically. For example, if you update or delete a Primary Key value in the table, the corresponding Foreign Key values are also updated or deleted automatically. This option can only be added to tables with Primary Keys.

The technique of global updates using constraints assumes you are updating a PRIMARY KEY value. Using the CASCADE option now you can cascade the change from the PRIMARY KEY to the referenced FOREIGN KEY columns.

The CASCADE option must be added from the R> prompt using the ALTER TABLE command.

Syntax:

```
ALTER TABLE PrimaryKeyTable ADD CASCADE
```

Example:

```
CONNECT ConComp
```

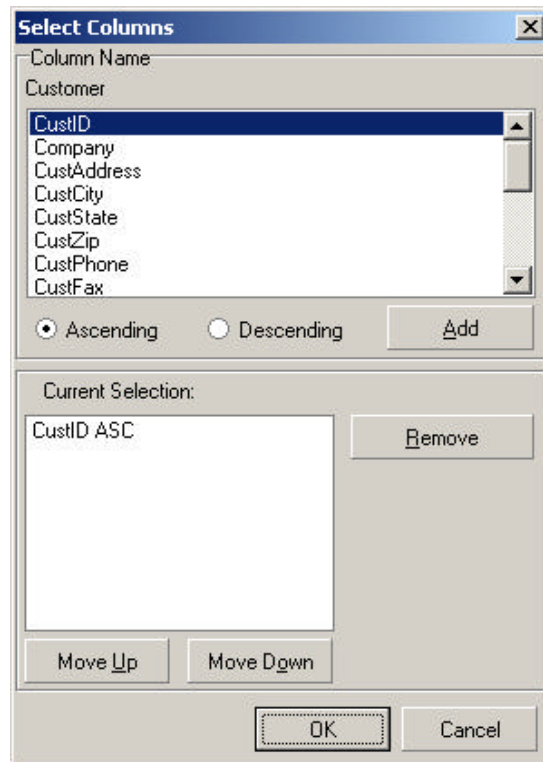
```
ALTER TABLE Customer ADD CASCADE
```

The CASCADE option makes sure that with any change to a PRIMARY KEY value, all the corresponding FOREIGN KEY values are updated. This update happens when the data is changed using a form, using the Data Browser/Editor, or using the UPDATE command. This will cascade the changes to all Foreign Keys referencing that table that contain the changed value.

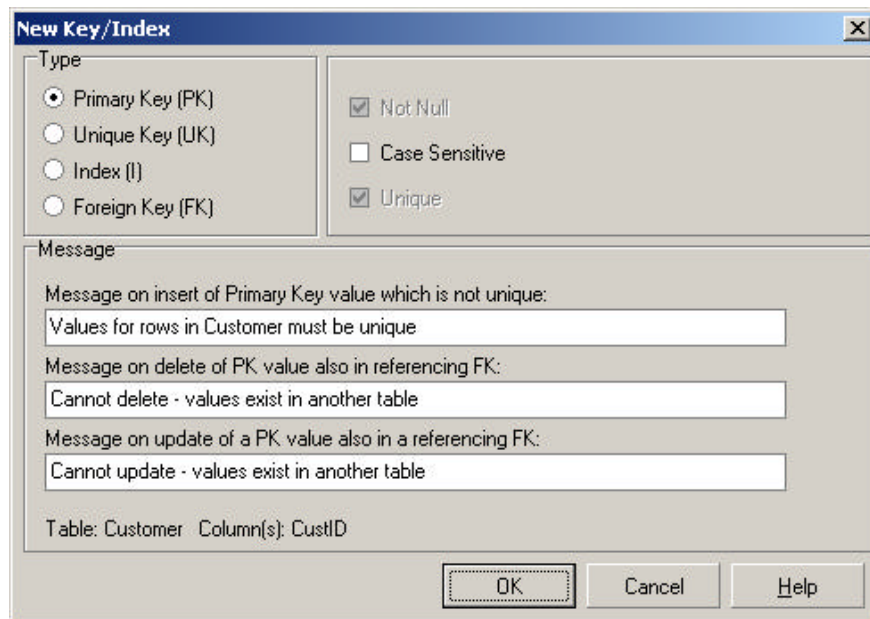
DROP CASCADE disables the CASCADE feature so that Primary/Foreign key relationships are not maintained automatically.

Example:

```
ALTER TABLE Customer DROP CASCADE
```

(Figure 3)



(Figure 4)

- Defining a Primary Key or Unique constraint from the R> prompt using CREATE TABLE or ALTER TABLE commands requires the column be explicitly defined as NOT NULL as well as the Unique or Primary Key designation. However, defining a Primary Key from the Database Explorer > Data Designer does not require the explicit NOT NULL, the data designer do it for you automatically. For example, CREATE TABLE tblname (colname data type NOT NULL PRIMARY KEY, ...)

- Before defining Primary and Foreign key constraints, decide which column or columns and which tables are best suited for the Primary Key. The column or columns selected should be the ones that uniquely identify the row from other rows in the table, and the table should be the one where that value is first entered into the database and is unique. Linking columns are generally good candidates for primary, foreign key constraints. For example, in the classic ConComp sample database, the EmpID column is used to link data in the Employee table with data in the TransMaster table and data in the SalesBonus table. Since a record is first entered into the Employee table, and when data is entered into the TransMaster or SalesBonus tables a matching record must already exist in the Employee table, the EmpID column in Employee becomes the Primary Key. The EmpID columns in TransMaster and SalesBonus become foreign keys referencing the Employee table Primary Key. What benefit is gained from designating EmpID as a Primary Key in the Employee table and Foreign Keys in the TransMaster and SalesBonus tables? By this designation you are protecting your data from inadvertent changes. The EmpID value in Employee cannot be changed when there is a matching row in the TransMaster or SalesBonus table. Rows cannot be deleted from either table. When adding data to TransMaster or SalesBonus, you automatically require a matching value in Employee without having to define a rule.

Listing Constraints:

LIST CONSTRAINTS at the R> prompt shows Primary Key, Foreign Key and Unique Key constraints. The constraint ID, the type of constraint, the table name, and table references if the key was a foreign key are displayed. NOT NULL constraints are also displayed.

R>LIST CONSTRAINTS

Id	Type	Table Name	References
#31	PRIMARY KEY REFERENCED	Component	
#37	FOREIGN KEY	SalesBonus	Employee
#32	PRIMARY KEY REFERENCED	Titles	
#39	FOREIGN KEY	TransMaster	Customer
#38	FOREIGN KEY	TransMaster	Employee
#33	PRIMARY KEY REFERENCED	TransMaster	
#40	FOREIGN KEY	Contact	Customer
#42	FOREIGN KEY	CompUsed	Product
#41	FOREIGN KEY	CompUsed	Component
#44	FOREIGN KEY	TransDetail	TransMaster
#43	FOREIGN KEY	TransDetail	Product
#45	FOREIGN KEY	ProdLocation	Product
#46	FOREIGN KEY	Levels	Product
#34	PRIMARY KEY	Levels	
#35	PRIMARY KEY REFERENCED	Product	
#47	FOREIGN KEY	Employee	Titles
#36	PRIMARY KEY REFERENCED	Employee	
#51	PRIMARY KEY REFERENCED	Customer	

NOT NULL Constraints:

Table Name	Column Name
Component	CompID
Titles	EmpTID
TransMaster	TransID
Contact	ContFName
	ContLName
ProdLocation	OnHand
Levels	ModLevel
	Model
Product	Model
Employee	EmpID
Customer	CustID

LIST PKEYS at the R> prompt will return a list of all PRIMARY KEYS and PRIMARY KEYS REFERENCED. The constraint ID, the type of constraint, and the table name are displayed.

R>LIST PKEYS

Id	Type	Table Name
#31	PRIMARY KEY REFERENCED	Component
#32	PRIMARY KEY REFERENCED	Titles
#33	PRIMARY KEY REFERENCED	TransMaster
#34	PRIMARY KEY	Levels
#35	PRIMARY KEY REFERENCED	Product
#36	PRIMARY KEY REFERENCED	Employee
#51	PRIMARY KEY REFERENCED	Customer

LIST PKEYS FOR tablename will return the name of a PRIMARY KEY column for a given table name and associated REFERENCED keys, if defined. The constraint ID, the type of constraint, primary key column name, referenced table names, and the referenced column names are displayed.

R>LIST PKEYS FOR Customer

Table Name: Customer

Id	Type	Column Name(s)	Ref Table Name	Ref Column Name(s)
#51	PRIMARY KEY	CustID	TransMaster Contact	CustID CustID

R>LIST PKEYS FOR TransMaster

Table Name: TransMaster

Id	Type	Column Name(s)	Ref Table Name	Ref Column Name(s)
#33	PRIMARY KEY	TransID	TransDetail	TransID

R>LIST PKEYS FOR Employee

Table Name: Employee

Id	Type	Column Name(s)	Ref Table Name	Ref Column Name(s)
#36	PRIMARY KEY	EmpID	SalesBonus TransMaster	EmpID EmpID

R>LIST PKEYS FOR Product

Table Name: Product

Id	Type	Column Name(s)	Ref Table Name	Ref Column Name(s)
#35	PRIMARY KEY	Model	CompUsed TransDetail ProdLocation Levels	Model Model Model Model

LIST FKEYS command will return a list of all FOREIGN KEYS. The constraint ID, the type of constraint, the table name, and referenced tables are displayed.

R>LIST FKEYS

Id	Type	Table Name	References
#37	FOREIGN KEY	SalesBonus	Employee
#39	FOREIGN KEY	TransMaster	Customer
#38	FOREIGN KEY	TransMaster	Employee
#40	FOREIGN KEY	Contact	Customer
#42	FOREIGN KEY	CompUsed	Product
#41	FOREIGN KEY	CompUsed	Component
#44	FOREIGN KEY	TransDetail	TransMaster
#43	FOREIGN KEY	TransDetail	Product
#45	FOREIGN KEY	ProdLocation	Product
#46	FOREIGN KEY	Levels	Product
#47	FOREIGN KEY	Employee	Titles

LIST FKEYS FOR tablename will return the name of a FOREIGN KEY column for a given table name and associated REFERENCED keys, if defined. The constraint ID, the type of constraint, primary key column name, referenced table names, and the referenced column names are displayed.

R>LIST FKEYS FOR TransMaster

Table Name: TransMaster

Id	Type	Column Name(s)	Ref Table Name	Ref Column Name(s)
#39	FOREIGN KEY	CustID	Customer	CustID
#38	FOREIGN KEY	EmpID	Employee	EmpID

R>LIST FKEYS FOR TransDetail

Table Name: TransDetail

Id	Type	Column Name(s)	Ref Table Name	Ref Column Name(s)
#44	FOREIGN KEY	TransID	TransMaster	TransID
#43	FOREIGN KEY	Model	Product	Model

R>LIST FKEYS FOR Contact

Table Name: Contact

Id	Type	Column Name(s)	Ref Table Name	Ref Column Name(s)
#40	FOREIGN KEY	CustID	Customer	CustID

R>LIST FKEYS FOR Employee

Table Name: Employee

Id	Type	Column Name(s)	Ref Table Name	Ref Column Name(s)
#47	FOREIGN KEY	EmpTID	Titles	EmpTID

LIST UKEYS command will return a list of all UNIQUE KEYs in the connected database.

LIST UKEYS FOR tablename command will return a list of all UNIQUE KEY columns for a given table, if defined.

Removing Constraints:

A primary key constraint that is referenced by a foreign key cannot be removed until the foreign key constraint has first been deleted. If a column was first defined as not null, and then as a primary key, the not null constraint on the column cannot be removed until the primary key constraint has been removed. Removing a primary key constraint does not remove the NOT NULL part of the constraint. That must be removed separately.

Customizing Constraint Violation Messages:

When primary key, foreign key and not null constraints are defined, custom violation messages can be entered. The messages cannot be added or modified after the constraint is defined. The constraint must be deleted and re-defined to add or modify custom messages. UNLOADing the database, editing the structure file and then re-building the database technique can also be used to customize the constraint violation messages.