CREATE TABLE

Using the FOREIGN KEY Keywords

A foreign key joins and establishes dependencies between tables. A foreign key references a unique or primary key in a table. For every entry in the foreign-key columns, a matching entry must exist in the unique or primary-key columns if all foreign-key columns contain non-null values. You cannot make BYTE or TEXT columns foreign keys.

When you use FOREIGN KEY keywords, you must use the REFERENCES clause, page 1-232, to complete the foreign key dependencies.

CHECK Clause

Check constraints allow you to designate conditions that must be met before data can be assigned to a column during an INSERT or UPDATE statement. If a row evaluates to false for any check constraint defined on a table during an insert or update, the database server returns an error.

Check constraints are defined using search conditions. The search condition cannot contain subqueries; aggregates; host variables; rowids; the CURRENT, USER, SITENAME, DBSERVERNAME, or TODAY functions; or stored procedure calls.

Warning: When you specify a date value in a search condition, make sure to specify 4 digits instead of 2 digits for the year. When you specify a 4-digit year, the DBCENTURY environment variable has no effect on how the database server interprets the search condition. When you specify a 2-digit year, the DBCENTURY environment variable can affect how the database server interprets the search condition, so the check constraint might not work as you intended. See the “Informix Guide to SQL: Reference” for more information on the DBCENTURY environment variable.

The Column-Level Constraint definition on page 1-227 and the Table-Level Constraint definition on page 1-228 refer to the CHECK clause.
Defining Check Constraints at the Column Level

If you define a check constraint at the column level, the only column that the check constraint can check against is the column itself. In other words, the check constraint cannot depend upon values in other columns of the table. For example, as the following statement shows, the table `my_accounts` has two columns with check constraints:

```
CREATE TABLE my_accounts (
    chk_id SERIAL PRIMARY KEY,
    acct1 MONEY CHECK (acct1 BETWEEN 0 AND 99999),
    acct2 MONEY CHECK (acct2 BETWEEN 0 AND 99999))
```

Both `acct1` and `acct2` are columns of MONEY data type whose values must be between 0 and 99999. If, however, you wanted to test that `acct1` had a larger balance than `acct2`, you would not be able to create the check constraint at the column level. To create a constraint that checks values in more than one column, you must define the constraint at the table level.

Defining Check Constraints at the Table Level

When you defined a check constraint at the table level, each column in the search condition must be a column in that table. You cannot create a check constraint for columns across tables. The next example builds the same table and columns as the previous example. However, the check constraint now spans two columns in the table.

```
CREATE TABLE my_accounts (
    chk_id SERIAL PRIMARY KEY,
    acct1 MONEY,
    acct2 MONEY,
    CHECK (acct1 > acct2))
```

In this example, the `acct1` column must be greater than the `acct2` column, or the insert or update fails.