LOAD

Use the LOAD statement to insert data from an operating-system file into an existing table, synonym, or view.

Syntax

```
LOAD FROM 'filename'  
DELIMITER 'delimiter'  

INSERT INTO
  + Table Name p. 1-1044  
  + Synonym Name p. 1-1042  
  + View Name p. 1-1047

Element | Purpose | Restrictions | Syntax

| column name | The name of a column or columns that receive data values from the load file during the load operation | You must specify the columns that receive data if you are not loading data into all columns. You must also specify columns if the order of the fields in the load file does not match the default order of the columns in the table (the order established when the table was created). | Identifier, p. 1-962 |
```
<table>
<thead>
<tr>
<th>Element</th>
<th>Purpose</th>
<th>Restrictions</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>delimiter</code></td>
<td>A quoted string that identifies the character to use as the delimiter in the LOAD FROM file. The delimiter is a character that separates the data values in each line of the LOAD FROM file.</td>
<td>If you do not specify a delimiter character, the database server uses the value of the <code>DBDELIMITER</code> environment variable. If <code>DBDELIMITER</code> has not been set, the default delimiter is the vertical bar (</td>
<td>). You cannot use the following items as delimiter characters: backslash (), new-line character (=CTRL-J), and hexadecimal numbers (0 to 9, a to f, A to F).</td>
</tr>
<tr>
<td><code>filename</code></td>
<td>A quoted string that identifies the pathname and filename of the load file. The load file contains the data to be loaded into the specified table or view. The default pathname for the load file is the current directory.</td>
<td>If you do not include a list of columns in the <code>column name</code> parameter, the fields in the load file must match the columns specified for the table in number, order, and type. You must also observe restrictions about the same number of fields in each line, the relationship of field lengths to column lengths, the representation of data types in the file, the use of the backslash character () with certain special characters, and special rules for <code>VARCHAR</code> and <code>BLOB</code> data types. See &quot;The LOAD FROM File&quot; on page 1-514 for information on these restrictions.</td>
<td>Quoted String. p. 1-1010. The pathname and filename specified in the quoted string must conform to the conventions of your operating system.</td>
</tr>
</tbody>
</table>

**Usage**

The LOAD statement adds new rows to the table. It does not overwrite existing data. You cannot add a row that has the same key as an existing row.

To use the LOAD statement, you must have Insert privileges for the table where you want to insert data. For information on database-level and table-level privileges, see the GRANT statement on page 1-458.
The **LOAD FROM File**

The LOAD FROM file contains the data to add to a table. You can use the file that the UNLOAD statement creates as the LOAD FROM file.

If you do not include a list of columns in the INSERT INTO clause, the fields in the file must match the columns that are specified for the table in number, order, and data type.

Each line of the file must have the same number of fields. You must define field lengths that are less than or equal to the length that is specified for the corresponding column. Specify only values that can convert to the data type of the corresponding column. The following table indicates how your Informix product expects you to represent the data types in the LOAD file (when they use the default locale, U.S. English).

<table>
<thead>
<tr>
<th>Type of Data</th>
<th>Input Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank</td>
<td>One or more blank characters between delimiters. You can include leading blanks in fields that do not correspond to character columns.</td>
</tr>
<tr>
<td>boolean</td>
<td>A 't' or 't' indicates a TRUE value, and an 'f' or 'f' indicates a FALSE value.</td>
</tr>
<tr>
<td>collections</td>
<td>A collection must have its values surrounded by braces ({}) and a field delimiter separating each element. For more information, see &quot;Loading Complex Types&quot; on page 1-519.</td>
</tr>
<tr>
<td>date</td>
<td>A character string in the following format: mm/dd/year. You must state the month as a two-digit number. You can use a two-digit number for the year if the year is in the 20th century. (You can specify another century algorithm with the DBCENTURY environment variable.) The value must be an actual date; for example, February 30 is illegal. You can use a different date format if you indicate this format with the GL_DATE or DBDATE environment variable. See the Guide to GLS Functionality for more information about these environment variables.</td>
</tr>
<tr>
<td>Type of Data</td>
<td>Input Format</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MONEY</td>
<td>A value that can include currency notation: a leading currency symbol ($), a comma (,) as the thousands separator, and a period (.) as the decimal separator. You can use a different currency notation if you indicate this notation with the DBMONEY environment variable. For more information on this environment variable, see the Guide to GLS Functionality.</td>
</tr>
<tr>
<td>NULL</td>
<td>Nothing between the delimiters.</td>
</tr>
<tr>
<td>row types</td>
<td>A row type must have its values surrounded by parentheses and a field delimiter separating each element. For more information, see “Loading Complex Types” on page 1-519.</td>
</tr>
<tr>
<td>(named and</td>
<td></td>
</tr>
<tr>
<td>unnamed)</td>
<td></td>
</tr>
<tr>
<td>simple large</td>
<td>TEXT and BYTE columns are loaded directly from the LOAD TO file. For more information, see “Loading Simple Large Objects” on page 1-518.</td>
</tr>
<tr>
<td>objects</td>
<td></td>
</tr>
<tr>
<td>(TEXT, BYTE)</td>
<td></td>
</tr>
<tr>
<td>smart large</td>
<td>CLOB and BLOB columns are loaded from a separate operating-system file. The field for the CLOB or BLOB column in the LOAD FROM file contains the name of this separate file. For more information, see “Loading Smart Large Objects” on page 1-518.</td>
</tr>
<tr>
<td>objects</td>
<td></td>
</tr>
<tr>
<td>(CLOB, BLOB)</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>A character string in the following format: year-month-day hour:minute:second.fraction. You cannot use type specification or qualifiers for DATETIME or INTERVAL values. The year must be a four-digit number, and the month must be a two-digit number. You can specify a different date and time format with the GL_DATETIME or DBTIME environment variable. See the Guide to GLS Functionality for more information on these environment variables.</td>
</tr>
</tbody>
</table>
If you are using a nondefault locale, the formats of DATE, DATETIME, MONEY, and numeric column values in the LOAD FROM file must be compatible with the formats that the locale supports for these data types. For more information, see the Guide to GLS Functionality.

If you include any of the following special characters as part of the value of a field, you must precede the character with a backslash (\):

- Backslash
- Delimiter
- New-line character anywhere in the value of a VARCHAR or NVARCHAR column
- New-line character at end of a value for a TEXT value

Do not use the backslash character (\) as a field delimiter. It serves as an escape character to inform the LOAD statement that the next character is to be interpreted as part of the data.

The following example shows the contents of a hypothetical input file named new_custs:

```
Jeffery Padgett|Wheel Thrills|3450 El Camino|Suite 10|Palo Alto|CA|94306|
Linda Lane|Palo Alto Bicycles|2344 University||Palo Alto|CA|94301|(415)323-6440
```
This data file conveys the following information:

- Indicates a serial field by specifying a zero (0)
- Uses the vertical bar (|), the default delimiter character
- Assigns null values to the phone field for the first row and the address2 field for the second row
  The null values are shown by two delimiter characters with nothing between them.

The following statement loads the values from the new_custs file into the customer table owned by jason:

```
LOAD FROM 'new_custs' INSERT INTO jason.customer
```

For more information about the format of the input file, see the discussion of the dbload utility in the Informix Migration Guide.

**Loading Character Data**

The fields that correspond to character columns can contain more characters than the defined maximum allows for the field. The extra characters are ignored.

If you are loading columns that are the VARCHAR data type, note the following information:

- If you give the LOAD statement data in which the character fields (including VARCHAR) are longer than the column size, the excess characters are disregarded.
- Use the backslash (\) to escape embedded delimiter and backslash characters in all character fields, including VARCHAR.

These restrictions on character columns also apply to NCHAR and NVARCHAR columns. For more information on these data types, see the Guide to GLS Functionality.
Loading Simple Large Objects

The database server loads simple large objects (BYTE and TEXT columns) directly from the LOAD FROM file. Keep the following restrictions in mind when you load BYTE and TEXT data:

- You cannot have leading and trailing blanks in BYTE fields.
- Use the backslash (\) to escape embedded delimiter and backslash characters in TEXT fields.
- Data being loaded into a BYTE column must be in ASCII-hexadecimal form. BYTE columns cannot contain preceding blanks.

For TEXT columns, the database server handles any required code-set conversions for the data. For more information, see the *Guide to GLS Functionality.*

If you are unloading files that contain simple-large-object data types, objects smaller than 10 kilobytes are stored temporarily in memory. You can adjust the 10-kilobyte setting to a larger setting with the `DBBLOBBUF` environment variable. Simple large objects that are larger than the default or the setting of the `DBBLOBBUF` environment variable are stored in a temporary file. For additional information about the `DBBLOBBUF` environment variable, see the *Informix Guide to SQL: Reference.*

Loading Smart Large Objects

The database server loads smart large objects (BLOB and CLOB columns) from a separate operating-system file on the client computer. It copies all smart-large-object values into a single file. Each BLOB or CLOB value is appended to the current file. The database server might create several files if the values are extremely large or there any many values.

In a LOAD FROM file, a CLOB or BLOB column value appears as follows:

```
start_off, end_off, client_path
```
In this format, `start_off` is the starting offset of the smart-large-object value within the file, `end_off` is the length of the BLOB or CLOB value, and `client_path` is the pathname for the client file. For example, to load a CLOB value that is 2048 bytes long and stored at the beginning of the `/usr/apps/clob_val` file, the database server expects the following value in the LOAD FROM file to appear as follows:

```
|0, 2048, /usr/apps/clob_value|
```

The preceding example assumes a default field delimiter of the vertical bar.

**Loading Complex Types**

In a LOAD FROM file, complex types appear as follows:

- Collections are introduced with the appropriate constructor `SET`, `MULTISET`, `LIST`, and their elements are enclosed in braces `{}` and separated with a comma, as follows:
  
  ```
  constructor(val1, val2, ...)
  ```

  For example, to load the `SET` values `{1, 3, 4}` into a column whose data type is `SET(INTEGER NOT NULL)`, the corresponding field of the LOAD FROM file appears as:

  ```
  |SET(1, 3, 4)|
  ```

- Row types (named and unnamed) have their fields enclosed with parentheses and separated with the field separator, as follows:

  ```
  (val1 | val2 | ...)
  ```

  For example, to load the `ROW` values `(1, 'abc')`, the corresponding field of the LOAD FROM file appears as:

  ```
  |(1 | abc)|
  ```

The preceding examples use the default field separator, the vertical bar `|`.

**Loading Opaque-Type Columns**

Some opaque data types require special processing when they are inserted. For example, if an opaque data type contains spatial or multirepresentational data, it might provide a choice of how to store the data: inside the internal structure or, for very large objects, in a smart large object.
This processing is accomplished by calling a user-defined support function called \texttt{assign()}. When you execute the \texttt{LOAD} statement on a table whose rows contains one of these opaque types, the database server automatically invokes the \texttt{assign()} function for the type. The \texttt{assign()} function can make the decision of how to store the data. For more information about the \texttt{assign()} support function, see the "Extending INFORMIX-Universal Server: Data Types" manual.

\textbf{DELIMITER Clause}

Use the \texttt{DELIMITER} clause to specify the delimiter that separates the data contained in each column in a row in the \texttt{LOAD FROM} file. If you omit this clause, your Informix product checks the \texttt{DBDELIMITER} environment variable.

If the \texttt{DBDELIMITER} environment variable has not been set, the default delimiter is the vertical bar (|). See Chapter 3 in the Informix Guide to SQL: Reference for information about how to set the \texttt{DBDELIMITER} environment variable.

You can specify TAB (CTRL-i) or <blank> (= ASCII 32) as the delimiter symbol. You cannot use the following items as the delimiter symbol:

- Backslash (\)
- New-line character (= CTRL-j)
- Hexadecimal numbers (0 to 9, a to f, A to F)

The following statement identifies the semicolon (:) as the delimiter character:

\begin{verbatim}
LOAD FROM '/a/data/ord.loadfile' DELIMITER ':'
INSERT INTO orders
\end{verbatim}
**INPUT INTO Clause**

Use the INSERT INTO clause to specify the table, synonym, or view in which to load the new data. (See the discussion of Synonym Name, Table Name, and View Name that begins on page 1-1042 for details.)

You must specify the column names only if one of the following conditions is true:

- You are not loading data into all columns.
- The input file does not match the default order of the columns (determined when the table was created).

The following example identifies the **price** and **discount** columns as the only columns in which to add data:

```
LOAD FROM '/tmp/prices' DELIMITER ','
    INSERT INTO norman.worktab(price,discount)
```

**References**

See the UNLOAD and INSERT statements in this manual.

In the **Informix Migration Guide**, see the task-oriented discussion of the LOAD statement and other utilities for moving data.

In the **Guide to GLS Functionality**, see the discussion of the GLS aspects of the LOAD statement.