CREATE VIEW

Use the CREATE VIEW statement to create a new view that is based upon existing tables and views in the database.

Syntax

<table>
<thead>
<tr>
<th>Element</th>
<th>Purpose</th>
<th>Restrictions</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>row type name</td>
<td>The name of a named row type that you use to specify the type of a typed view</td>
<td>You must have USAGE privileges on the named row type or be its owner or the DBA. The named row type must exist before you can assign it to a view.</td>
<td>Data Type, p. 1-855</td>
</tr>
<tr>
<td>column name</td>
<td>The name of a column in the view being created</td>
<td>See “Naming View Columns” on page 1-288.</td>
<td>Identifier, p. 1-962</td>
</tr>
</tbody>
</table>
Usage

You can create typed or untyped views. If you omit the OF TYPE clause, the rows in the view are considered to be untyped and default to an unnamed row type.

Typed views, like typed tables, are based on a named row type. Each column in the view corresponds to a field in the named row type.

You can use a view in any SQL statement where you can use a table, except the following.

<table>
<thead>
<tr>
<th>SQL Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER FRAGMENT</td>
<td>DROP TABLE</td>
</tr>
<tr>
<td>ALTER INDEX</td>
<td>DROP TRIGGER</td>
</tr>
<tr>
<td>ALTER TABLE</td>
<td>LOCK TABLE</td>
</tr>
<tr>
<td>CREATE INDEX</td>
<td>RECOVER TABLE</td>
</tr>
<tr>
<td>CREATE TABLE</td>
<td>RENAME TABLE</td>
</tr>
<tr>
<td>CREATE TRIGGER</td>
<td>UNLOCK TABLE</td>
</tr>
<tr>
<td>DROP INDEX</td>
<td></td>
</tr>
</tbody>
</table>

The view behaves like a table that is called view name. It consists of the set of rows and columns that the SELECT statement returns each time the SELECT statement is executed by using the view. The view reflects changes to the underlying tables with one exception. If a SELECT * clause defines the view, the view has only the columns in the underlying tables at the time the view is created. New columns that are subsequently added to the underlying tables with the ALTER TABLE statement do not appear in the view.

The view name must be unique; that is, a view name cannot have the same name as another database object, such as a table, synonym, or temporary table.

The view inherits the data types of the columns from the tables from which they come. Data types of virtual columns are determined from the nature of the expression.

To create a view, you must have the Select privilege on all columns from which the view is derived.
CREATE VIEW

The SELECT statement is stored in the sysviews system catalog table. When you subsequently refer to a view in another statement, the database server performs the defining SELECT statement while it executes the new statement.

You cannot create a view on a temporary table.

If you create a view outside the CREATE SCHEMA statement, you receive warnings if you use the -ansi flag or set DBANSIWARN.

Subset of a SELECT Allowed in CREATE VIEW

The SELECT statement has the form that is described on page 1-593, but in CREATE VIEW, it cannot have an ORDER BY clause, INTO TEMP clause, or UNION operator. Do not use display labels in the select list; display labels are interpreted as column names.

Naming View Columns

The number of columns that you specify in the column name parameter must match the number of columns returned by the SELECT statement that defines the view.

If you do not specify a list of columns, the view inherits the column names of the underlying tables. In the following example, the view herostock has the same column names as the ones in the SELECT statement:

```
CREATE VIEW herostock AS
  SELECT stock_num, description, unit_price, unit, unit_descr
  FROM stock WHERE manu_code = 'HRO'
```

If the SELECT statement returns an expression, the corresponding column in the view is called a virtual column. You must provide a name for virtual columns. You must also provide a column name in cases where the selected columns have duplicate column names when the table prefixes are stripped. For example, when both orders.order_num and items.order_num appear in the SELECT statement, you must provide two separate column names to label them in the CREATE VIEW statement, as the following example shows:

```
CREATE VIEW someorders (custnum,ocustnum,newprice) AS
  SELECT orders.order_num,items.order_num,
        items.total_price*1.5
  FROM orders, items
  WHERE orders.order_num = items.order_num
  AND items.total_price > 100.00
```
If you must provide names for some of the columns in a view, then you must provide names for all the columns; that is, the column list must contain an entry for every column that appears in the view.

### Using a View in the SELECT Statement

You can define a view in terms of other views, but you must abide by the restrictions on creating views that are listed in Chapter 11 of the *Informix Guide to SQL: Tutorial*. See that manual for further information.

### WITH CHECK OPTION Keywords

The WITH CHECK OPTION keywords instruct the database server to ensure that all modifications that are made through the view to the underlying tables satisfy the definition of the view.

The following example creates a view that is named `palo_alto`, which uses all the information in the `customer` table for customers in the city of Palo Alto. The database server checks any modifications made to the `customer` table through `palo_alto` because the WITH CHECK OPTION is specified.

```sql
CREATE VIEW palo_alto AS
  SELECT * FROM customer
  WHERE city = 'Palo Alto'
  WITH CHECK OPTION
```

What do the WITH CHECK OPTION keywords really check and prevent? It is possible to insert into a view a row that does not satisfy the conditions of the view (that is, a row that is not visible through the view). It is also possible to update a row of a view so that it no longer satisfies the conditions of the view. For example, if the view was created without the WITH CHECK OPTION keywords, you could insert a row through the view where the city is Los Altos, or you could update a row through the view by changing the city from Palo Alto to Los Altos.

To prevent such inserts and updates, you can add the WITH CHECK OPTION keywords when you create the view. These keywords ask the database server to test every inserted or updated row to ensure that it meets the conditions that are set by the WHERE clause of the view. The database server rejects the operation with an error if the row does not meet the conditions.
However, even if the view was created with the WITH CHECK OPTION keywords, you can perform inserts and updates through the view to change columns that are not part of the view definition. A column is not part of the view definition if it does not appear in the WHERE clause of the SELECT statement that defines the view.

**Updating Through Views**

If a view is built on a single table, the view is *updatable* if the SELECT statement that defined it did not contain any of the following items:

- Columns in the select list that are aggregate values
- Columns in the select list that use the UNIQUE or DISTINCT keyword
- A GROUP BY clause
- A derived value for a column, which was created using an arithmetical expression

In an updatable view, you can update the values in the underlying table by inserting values into the view.

**Important:** You cannot update or insert rows in a remote table through views with check options.

**Examples**

The following statement creates a view that is based on the `person` table. When you create a view without an OF TYPE clause, the view is referred to as an *untyped view*.

```
CREATE VIEW v1 AS SELECT * 
FROM person
```

The following statement creates a typed view that is based on the table `person`. To create a typed view, you must include an OF TYPE clause. When you create a typed view, the named row type that you specify immediately after the OF TYPE keywords must already exist.

```
CREATE VIEW v2 OF TYPE person_t AS SELECT * 
FROM person
```

For more information about how to create and use typed views, see Chapter 11 of the *Informix Guide to SQL: Tutorial*.
References

See the CREATE TABLE, DROP VIEW, GRANT, SELECT, and SET SESSION AUTHORIZATION statements in this manual.

In the Informix Guide to SQL: Tutorial, see the discussions of views and security in Chapter 11. Also, see the discussion of named row types in Chapter 10.