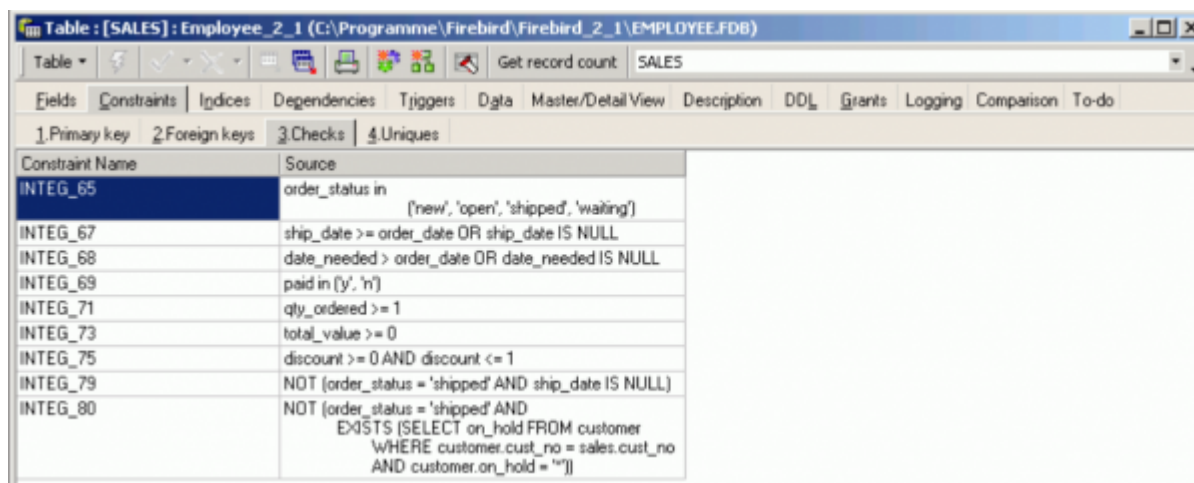


Check constraint

A check is a database examination, which ensures data consistency in the tables among each other. It can be executed automatically and so ensures that data contents are kept consistent by testing them before they are stored in the database.

The check constraint option enables each [data set](#) to be examined for validation of the expression in brackets following the check constraint. Check constraints in [tables](#) are identical to check constraints in [domains](#).

A check constraint can be specified for each [column](#) in a table, to guarantee the mechanism described above. It includes an expression that must be true, so that the data set following an insert or update can be written. The field contents must be included in the permissible values, which can be specified in a list. It is also possible to test the value for a minimum and maximum value. Furthermore the value can be compared to values in other columns, in order to test dependencies.



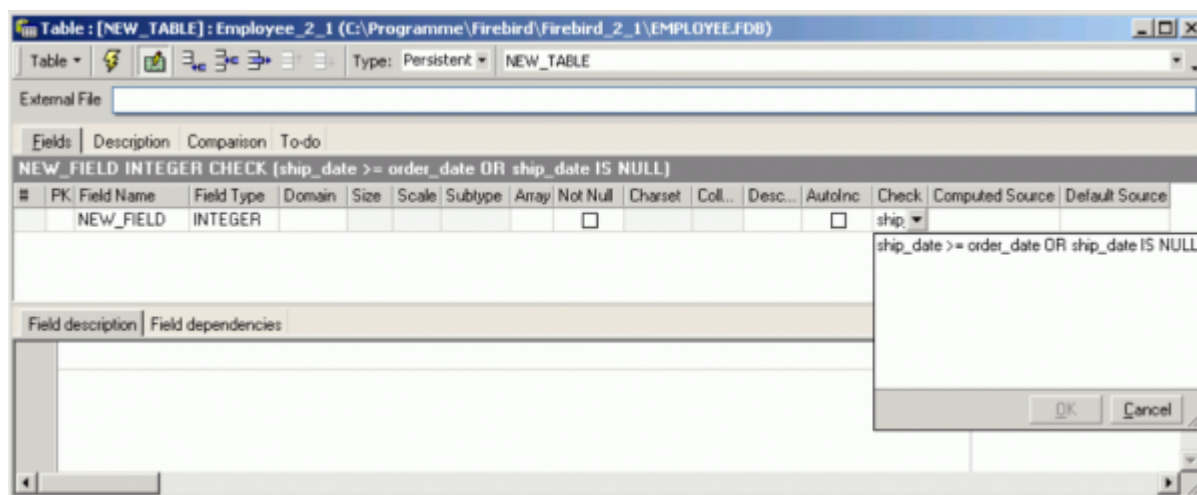
Constraint Name	Source
INTEG_65	order_status in ('new', 'open', 'shipped', 'waiting')
INTEG_67	ship_date >= order_date OR ship_date IS NULL
INTEG_68	date_needed > order_date OR date_needed IS NULL
INTEG_69	paid in ('y', 'n')
INTEG_71	qty_ordered >= 1
INTEG_73	total_value >= 0
INTEG_75	discount >= 0 AND discount <= 1
INTEG_79	NOT (order_status = 'shipped' AND ship_date IS NULL)
INTEG_80	NOT (order_status = 'shipped' AND EXISTS (SELECT on_hold FROM customer WHERE customer.cust_no = sales.cust_no AND customer.on_hold = "1"))

A check constraint can only examine the values in the current data set. When simultaneously inserting or altering multiple data sets, a check constraint can only guarantee one data integrity at a time at data set level.

If other data sets are referenced in the check, these could have been modified by another user at the time of entry, and therefore possibly have become invalid, even though the check constraint's test approved the data set. At the time of a check constraint validation, other data is only read for the check. For this reason, the values for the current operating sequence remain constant, even if another user has modified one of the values already referenced for validation.

A check constraint can be created directly when creating a table. When creating a check constraint, the following criteria should be taken into consideration:

- A check constraint cannot reference a domain.
 - A table column can only contain one check constraint.
 - A check constraint defined by a domain cannot be overridden by a local check constraint.
- However additional constraints can be specified.



In a check definition the **VALUE** keyword represents the value of the respective table column. The value examination is generally performed when inserting or updating this table column. The **Check Value** options permit diverse operations (please refer to [Comparison Operators](#) for a full list of possible operators).

[Referential integrity](#) declarations and [primary key](#) definitions are special check constraint compositions.

Only one constraint is permitted per column. If the [column](#) is based on a domain containing a constraint, both check constraints are active.

The specification of the keyword **CONSTRAINT** and the name are optional for all constraints. If no name is specified, Firebird/InterBase® generates a name automatically. All constraint names are stored in a system table called **DB\$RELATION_CONSTRAINTS**.

It is only necessary to name constraints, if they are to be deactivated at a later date using the **ALTER TABLE DROP** statement.

Please note that if you want to change the **CHECK constraint** for a domain that already has a constraint defined, the existing constraint must first be dropped and then the new one added. **ADD CHECK** does not replace the current constraint with the new one. It is also important to realize that altering a **CHECK constraint** does not cause existing database rows to be revalidated; **CHECK constraints** are only validated when an **INSERT** or **UPDATE** is performed. One way of overcoming this limitation is to perform an **UPDATE** query using a dummy operation. If existing rows violate the new **CHECK constraint**, the query fails. These rows can then be extracted by performing a **SELECT**.

Check constraints and NULLs

If a **CHECK** constraint resolves to **NULL**, Firebird versions before 2.0 reject the input. Following the SQL standard to the letter, Firebird 2.0 and above let **NULLs** pass and only consider the check failed if the outcome is false.

This change may cause existing databases to behave differently when migrated to Firebird 2.0+. Carefully examine your **CREATE/ALTER TABLE** statements and add and **XXX is not null** predicates to your **CHECKs** if they should continue to reject **NULL** input.

Please refer to the *Firebird 2.0 Language Reference Update* chapter, [CHECK accepts NULL outcome](#) for further information.

From:

<http://ibexpert.com/docu/> - **IBExpert**

Permanent link:

<http://ibexpert.com/docu/doku.php?id=01-documentation:01-13-miscellaneous:glossary:check-constraint>

Last update: **2023/08/14 09:40**

