Primary key

A primary key is a column (= simple key) or group of columns (= composite key/compound key) used to uniquely define a data set/row in the table. A primary key should always be defined at the time of defining a new table for each table. If you have a database that does not contain primary keys in all tables, and need to add these subsequently, please refer to Adding primary keys to existing tables below.

Relational theory states that a primary key should be designated for every table. It must be unique, and therefore cannot be NULL. It provides automatic protection against storing multiple values. In fact, without a primary key it is impossible to delete just one of two identical data sets. Each table can have only one designated primary key, although it can have other columns that are defined as UNIQUE and NOT NULL.

A primary key column is nothing other than a unique constraint complemented by a system index and the check constraint NOT NULL. Primary keys are always the preferred index of the Firebird/InterBase® Optimizer.

When a data set is created or changed, Firebird/InterBase® immediately checks the validity of the primary key. If the number already exists, a key violation results, and the storage process is immediately cancelled. Unfortunately Firebird/InterBase® allows tables to be created without a primary key, which is a mistake. Data tables should always be keyed.

Existing primary keys and their system names can be viewed on the IBExpert Table Editor / Constraints page.

It is wise to keep the primary key as short as possible to minimize the amount of disk space required, and to improve performance. IBExpert recommends the use of an autoincrement generator ID number used as an internal primary key for all tables. For example, a simple BIGINT data type generator not influenced in any way by any actual data. They do not need to be visible to the user as they are merely a tool to help the database work more efficiently and increase database integrity. One generator can be used as a source for all primary keys in a database, as the numbers do not need to be consecutive but merely unique. Each time a new data set is inserted, the generator automatically generates an ID number, regardless of the table name, for example, new customer_id = 1, new order_id = 2, new orderline_id = 3, new orderline_id = 4, new customer_id = 5, etc. A further advantage of such a single autoincrement generator primary key is that the database is perfectly prepared for replication; two or more servers can be connected and their data easily swapped, as the primary keys can be simply defined on both servers, e.g. server 1's generator should start at the value 1000000000 and server 2's at 2000000000 thus avoiding any conflict.

Although this method is unfortunately seldom used in the real world, it should be. Each primary key will only ever appear once in the database, which can be quite important in an OO (object-oriented) framework where there are so many objects floating around. They and you both need some unique identifier for the system to tell you what is behind the number, product, order etc.

Since version 1.5 Firebird allows a USING INDEX subclause to be placed at the end of a primary, unique or foreign key definition. Please refer to the *Firebird 2.0 Language Reference Update* chapter, USING INDEX subclause for further information.

Composite keys are not recommended, as these always slow performance and the sequence of the

fields concerned must be identical in all referenced tables.