Index Root Page - type 0x06

Every table in the database has an Index Root Page which holds data that describes the indexes for that table. Even tables that have no indices defined have an index root page.

The C code representation of an index root page is:

```
struct index_root_page
{
    pag irt_header;
    USHORT irt_relation;
    USHORT irt_count;
    struct irt_repeat {
        SLONG irt_root;
        union {
            float irt_selectivity;
            SLONG irt_transaction;
        } irt_stuff;
        USHORT irt_desc;
       UCHAR irt_keys;
        UCHAR irt_flags;
    } irt_rpt[1];
};
```

Irt_header: The page starts with a standard page header. The flags byte - pag_flags - is not used on this page type.

Irt_relation: Two bytes, unsigned. Offset 0x10 on the page. The relation ID. This is the value of RDB$RELATIONS.RDB$RELATION_ID.

Irt_count: Two bytes, unsigned. Offset 0x12 on the page. The number of indices defined for this table. If there are no indices defined this counter will show the value zero. (Every table in the database has an Index Root Page regardless of whether or not it has any indices defined.)

Irt_rpt: This is an array of index descriptors. The array begins at offset 0x14 on the page with the descriptor for the first index defined for the table. Descriptors are added to the 'top' of the array so the next index defined will have its descriptor at a higher page address than the previous descriptor. The descriptor entries consist of the following 6 fields (irt_root through irt_flags). Each descriptor is 0x0b bytes long.

Irt_root: Four bytes, signed. Offset 0x00 in each descriptor array entry. This field is the page number where the root page for the individual index (page type 0x07) is located. Experimenting has shown that if this value is zero, then you are most likely looking at the index root page for a deleted index.

Irt_selectivity: Four bytes, signed floating point. Offset 0x04 in each descriptor array entry. This is the same offset as for irt_transaction below. In ODS versions previous to 11.0 this field holds the index selectivity in floating point format.

Note: From ODS version 11.0, this field is no longer used as selectivity has been moved to the index field descriptors (see below).

Irt_transaction: Four bytes, signed. Offset 0x04 in each descriptor array entry - the same offset as irt_selectivity above. Normally this field will be zero but if an index is in the process of being created, the transaction ID will be found here.

Irt_desc: Two bytes, unsigned. Offset 0x08 in each descriptor array entry. This field holds the offset, from the start of the page, to the index field descriptors which are located at the bottom end (ie, highest addresses) of the page. To calculate the starting address, add the value in this field to the address of the start of the page.

Irt_keys: One byte, unsigned. Offset 0x0a in each descriptor array entry. This defines the number of keys (columns) in this index.

Irt_flags: One byte, unsigned. Offset 0x0b in each descriptor array entry. The flags define various attributes for this index, these are encoded into various bits in the field, as follows:

- *Bit 0*: Index is unique (set) or not (unset).
- *Bit 1*: Index is descending (set) or ascending (unset).
- *Bit 2*: Index [creation?] is in progress (set) or not (unset).
- *Bit 3*: Index is a foreign key index (set) or not (unset).
- *Bit 4*: Index is a primary key index (set) or not (unset).
- *Bit 5*: Index is expression based (set) or not (unset).

Each descriptor entry in the array holds an offset to a list of key descriptors.

These start at the highest address on the page and extend towards the lowest address. (The array of index descriptors (irt_rpt) starts at a low address on the page and increases upwards. At some point, they will meet and the page will be full.

The index field descriptors are defined as follows:

Irtd_field: Two bytes, unsigned. Offset 0x00 in each field descriptor. This field defines the field number of the table that makes up *this* field in the index. This number is equivalent to RDB$RELATION_FIELDS.RDB$FIELD_ID.

Irtd_itype: Two bytes, unsigned. Offset 0x02 in each field descriptor. This determines the data type of the appropriate field in the index. The allowed values in this field are:

- 0: field is numeric, but is not a 64-bit integer.
- 1: field is string data.
- 3: Field is a byte array.
- 4: Field is metadata.
- 5: Field is a date.
- 6: Field is a time.
- 7: Field is a timestamp.
- 8: field is numeric - and is a 64-bit integer.

You may note from the above that an irtd_itype with value 2 is not permitted.

Irtd_selectivity: Four bytes, floating point format. Offset 0x04 in each field descriptor. This field holds the selectivity of this particular column in the index. This applies to ODS 11.0 onwards. In pre ODS 11.0 databases, this field is not part of the index field descriptors and selectivity is applied to the index as a whole. See irt_selectivity above.

back to top of page

The following commands have been executed to create a parent child set of two tables and a selection of indices:

```
SQL> CREATE TABLE PARENT (
CON>    ID INTEGER NOT NULL,
CON>    EMAIL VARCHAR(150)
CON> );

SQL> ALTER TABLE PARENT
CON>    ADD CONSTRAINT PK_PARENT
CON>    PRIMARY KEY (ID);

SQL> ALTER TABLE PARENT
CON>    ADD CONSTRAINT UQ_EMAIL
CON>    UNIQUE (EMAIL);

SQL> COMMIT;

SQL> CREATE TABLE CHILD (
CON>    ID INTEGER NOT NULL,
CON>    PARENT_ID INTEGER,
CON>    STUFF VARCHAR(200)
CON> );

SQL> ALTER TABLE CHILD
CON>    ADD CONSTRAINT FK_CHILD
CON>    FOREIGN KEY (PARENT_ID)
CON>    REFERENCES PARENT (ID);

SQL> COMMIT;
```

The Following command was then executed to extract the index root pages for both of these tables:

```
SQL> SELECT R.RDB$RELATION_NAME,
CON>    R.RDB$RELATION_ID,
CON>    P.RDB$PAGE_TYPE,
CON>    P.RDB$PAGE_NUMBER
CON> FROM RDB$RELATIONS R
CON> JOIN RDB$PAGES P ON (P.RDB$RELATION_ID = R.RDB$RELATION_ID)
CON> WHERE R.RDB$RELATION_NAME IN ('PARENT','CHILD')
CON>    AND P.RDB$PAGE_TYPE = 6;

RDB$RELATION_NAME RDB$RELATION_ID RDB$PAGE_TYPE RDB$PAGE_NUMBER
================= =============== ============= ===============
PARENT                        139             6             173
CHILD                         140             6             178
```

back to top of page

Now that the root pages are known, we can take a look at the layout of these two pages and see how the details of the various indices are stored internally:

tux> ./fbdump ../blank.fdb -p 173,178

```
FBDUMP 1.00 - Firebird Page Dump Utility

Parameters : -p 173,178 -v
Database: ../blank.fdb

DATABASE PAGE DETAILS - Page 173
       Page Type: 6
       Flags: 0
       Checksum: 12345
       Generation: 5
       SCN: 0
       Reserved: 0
PAGE DATA
       Relation: 139
       Index Count: 2

       Root Page[0000]: 174
       Transaction[0000]: 0
       Descriptor[0000]: 4088 (0x0ff8)
       Keys[0000]: 1
       Flags[0000]: 17 :Unique:Ascending:Primary Key:
       Descriptor[0000].Field: 0
       Descriptor[0000].Itype: 0 :Numeric (Not BigInt)
       Descriptor[0000].Selectivity: 0.000000

       Root Page[0001]: 176
       Transaction[0001]: 0
       Descriptor[0001]: 4080 (0x0ff0)
       Keys[0001]: 1
       Flags[0001]: 1 :Unique:Ascending:
       Descriptor[0001].Field: 1
       Descriptor[0001].Itype: 1 :String
       Descriptor[0001].Selectivity: 0.000000

DATABASE PAGE DETAILS - Page 178
PAGE HEADER
       Page Type: 6
       Flags: 0
       Checksum: 12345
       Generation: 3
       SCN: 0
       Reserved: 0
PAGE DATA
       Relation: 140
       Index Count: 1
```

```
        Root Page[0000]: 180
        Transaction[0000]: 0
        Descriptor[0000]: 4088 (0x0ff8)
        Keys[0000]: 1
        Flags[0000]: 8 :NonUnique:Ascending:Foreign Key:
        Descriptor[0000].Field: 1
        Descriptor[0000].Itype: 0 :Numeric (Not BigInt)
        Descriptor[0000].Selectivity: 0.000000
```

We can see that the PARENT table (relation 139) has two defined indices while the CHILD table (relation 140) has one.

If we examine the above output we can see that the indices do match up to those that were created above. We can also see that in the event of an index being created without a sort order (ascending or descending) that the default is ascending.