Building and optimization of SQL - queries

Building of query:

1. Every developer must have comprehensive Data Base schema in front of them. The building of queries should be conducted on a physical model of a Data Base.
   - The volume of information in a Data Base should be equal to the minimum of the annual volume of a real working Data Base, bottom-poured by users in actual operating conditions.
- Formulate the query in your own terms and begin to build a query in terms of SQL-syntax from a basic table of query. (For example, if you need to select employees which have many different characteristics that are located in different tables then forming of sql-query will start with table - <EMPLOYEES>)
   - **ATTENTION!** If you want to select all fields of a table then **NEVER** use "*****" in clause <SELECT> it can cause mismatch in Data Base and programming errors.
3. The joining of tables is necessary to begin in order of concernment with point of logic of query. It is necessary to strongly definite the type of joining between tables (INNER, LEFT|RIGHT|CROSS|FULL OUTER JOIN) (See "Relationships between entities").
4. After the joining of tables is finished, imposed the condition of selection. The conditions of selection could be put on **JOIN** or on **result set**.
   - **ATTENTION!** The results of selection could be different when you impose conditions on joining and result set!
5. The order of synonyms on joining tables should begin with the last enumerated table of joining.
   - For example,

```
SELECT E.FIRST_NAME || '  ' || E.LAST_NAME, MF.NAME
FROM EMPLOYEE E INNER JOIN MEMBERS_FAMILY MF ON MF.EMP_NO = E.EMP_NO;
```

6. To join new tables and impose additional conditions in the query should be made in series. You should check the result set on every step!

When the query is written on SQL-language, begin to optimize it. For optimization of query you should analyze an execution plan of query. Generally, the time of query executions depends from correct usage of indexes on tables.

Optimization of query:

1. Before the testing of query, execute an operator <SET STATISTIC> for all indexes that were set into the plan of query execution.
2. Decide if you should use index in query. In case, if the table has not much data or built index would have a small selectivity then maybe the using of index could reduce the execution of query. For tripping of index you can make some fields of conditions computational.
   - Some tricks:

```
SELECT E.FIRST_NAME || ' ' || E.LAST_NAME
FROM EMPLOYEE E
WHERE E.EMP_NO + 0 = 1;
```

   or

```
SELECT MF.NAME, MF.BIRTHDAY
FROM MEMBERS_FAMILY MF
WHERE MF.NAME || '' = 'McGregor';
```

But you should understand that the tables in a real-life environment of exploitation would increase in size with time and tripping of indexes could cite to fatal consequences.

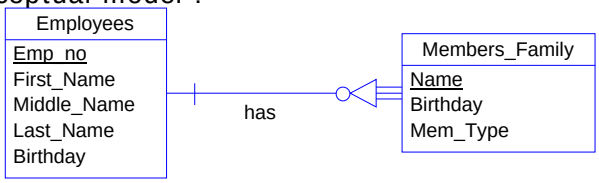
3. The subqueries in clause <SELECT> of query should be using with aggregate functions.
   - For example,

```
SELECT E.FIRST_NAME, E.LAST_NAME,
(SELECT MAX(MF.BIRTHDAY)
 FROM MEMBERS_FAMILY MF
WHERE MF.EMP_NO = E.EMP_NO) BIRTHDAY,
FROM EMPLOYEE E
WHERE A.EMP_NO < 156;
```

4. Define the functionality, which you put in every object you are going to create. Use <VIEW> and queries instead of <STORED PROCEDURES>. As opposed to <STORED PROCEDURES>, a query and <VIEW> are standard facilities of SQL.
5. As far as possible use a batch entry, fixing changes over defined set of operations of insertion, deletion and modifying with command <COMMIT WORK>.
6. Change the order of joining tables in query. Look and chose the optimal plan of query execution.
   - However, if the query is executing too long then change the logic of query if it is acceptably.
   - If slowness of query execution occurs because of scanning table, then create one or more indexes.
   - If the query is very bulky then divide it into a set of small and simple (on structure) queries and check the plan of their execution.

*All examples in this article were considered on the following models:

Conceptual model :



Physical model :