LAB11 - Data dictionary

If you remember the very first lab, you were asked to design a database and your design was checked against a number of important conditions (primary keys, unique constraints and so forth).

Many databases are unfortunately badly designed (sometimes they don't look like having being designed at all). When you are asked to develop an application that uses a database that you don't, it may be wise to look for defects.

You are asked in this lab to pick one check in the following list. Your query will get information from the data dictionary to report on possible problems.

You won't be graded proper (it will be full grade or zero), but everything will be put in Sakai and available to all of you, so that you can have a collection of tools for a DBMS that is increasingly popular. So, write good stuff, your reputation is at stake...

Pick at least one:

- Isolate tables (reference no other table, referenced by no other table) - sometimes justified (parameter table, log table), but sometimes just a big mistake.
- Multiple legs relationships: unreferenced tables (many-to-many relationships) with foreign keys to more than two different tables. Rarely justified.
- Tables without a primary key (very easy) - only acceptable for log tables.
- Tables that have no other unique constraint than a system-generated numerical identifier - once again, only acceptable for log tables.
- Tables where all columns have the same varchar datatype (with the same length) - smells of sloppy design.
- Several columns have the same name in different tables but different data types (or different lengths)
- Columns that reference (in a foreign key constraint) a column of a different type (forbidden by MySQL, allowed by PostgreSQL)
- Tables where all columns can be null except perhaps a system-generated number - never justifiable.
- In the schema, all varchar columns have the same (default or "safe") length - sloppy design.
- Percentage of single-column indexes (only, or mostly, single-column indexes is often rather naive indexing).
- Redundant indexes: indexes on exactly the same columns, and in the same order, than an other index on the same table (allowed by PosgreSQL :-()

- Useless indexes: single-column indexes for which there is an index on multiple columns with the column of the single-column index in first position (the multiple-column index can be used for the same queries and will be almost as efficient - removing the other index will help inserts/deletes)
- Unreferenced tables with (useless) system-generated columns (these identifiers are only for foreign keys - that said a lot of frameworks generate this systematically, which is pretty idiotic IMHO)
- Detection of circular FKs (tough) - A references B references C references A; exclude tables directly referencing themselves.
- Views that are built on other views (careful when querying those, dreadful performance may happen)