Database normalization and logical process concept paper

This short paper will explain with a simple example the process and the need of normalization In the most of the business databases. Complete proposal break down (Timeline phases, Financial phases) will be submitted per management request. Overview At first any database should be designed with the end user in mind. Logical database design, also referred to as the logical model, is the process of arranging data into logical, organized groups of objects that can easily be maintained.

The logical design f a database should reduce data repetition or go so far as to completely eliminate it. The needs of the end user should be one of the top considerations when designing a database. We should remember trough all design process that the end user is the person who ultimately uses the database. There should be ease of use through the user's front-end tool (a client program that allows a user access to a database), but this, along with optimal performance, cannot be achieved If the user's needs are not taken Into consideration.

Some user-related research and design considerations In NY business database Include the following: What data should be stored In the database? How will the user access the database? What privileges does the user require? How should the data be grouped in the database? What data is the most commonly accessed? How is all data related in the database? What measures should be taken to ensure accurate data? Ultimately Data should not be redundant, which means that the duplication of data should be kept to a minimum for several reasons.

As if data management were not difficult enough, redundancy of data could prove to be a disaster. Design Guidelines The actual guidelines of normalization, normal forms, are the three most common normal forms In the normalization research, design and process include: The first normal form, the second normal form, the third normal form. Of the three normal forms, each subsequent normal form depends on normalization steps taken In the previous normal form. For example, to normalize a database using the second normal form, the database must first be in the first normal form.

Normalization Objective The normalization process is widely used by database developers to design databases in which it is easy to organize and manage data while ensuring the accuracy of data throughout the database, reducing redundancies of data in a database, data consistency within the database, a much more flexible database design, a better handle on database security, organization is brought about by the normalization process, making everyone's Job easier, from the user who accesses tables to the database administrator (DAB) who is responsible for the overall management of every object in the database.

Also security Is provided In the sense that the DAB can grant access to limited tables to certain users. Security Is easier to control when normalization has occurred. Data Integrity Is the assurance of consistent and accurate data within a database. Lastly, normalization is an important is not normalized may include data that is contained in one or more different tables for no apparent reason. This could be bad for security reasons, disk space usage, speed of queries, efficiency of database updates, and, maybe most importantly, data integrity.

A database before normalization is one that has not been broken down logically into smaller, more manageable tables. Figure 1 illustrates the raw database used before it was normalized. Design Process with Example The objective of the First normal form is to divide the base data into logical units called tables. When each table has been designed, a primary key will assign to the most or all tables. Examine Figure 2, which illustrates how the raw database shown in the previous figure has been redeveloped using the first normal form.

To achieve the first normal form, data had to be broken into logical units of related information, each having a primary key and ensuring that there are no repeated groups in any of the tables. Instead of one large table, there are now smaller, more manageable tables: EMPLOYEE_TAB, CUSTOMER_TAB, and PRODUCTS_TAB. The primary keys are normally the first columns listed in a table, in this case: MME ID, CUTS ID, and PROD ID. The objective of the Second normal form is to take data that is only partly dependent on the primary key and enters that data into another table. Figure 3 illustrates the second normal form.

According to the figure, the second normal form is derived from the first normal form by further breaking two tables down into more specific units. EMPLOYEE_TAB split into two tables called EMPLOYEE_TAB Personal employee information is dependent on the primary key (MME_ID), so that information remained townspeople_TAB (MME_ID, LAST_NAME, ADDRESS, CITY, STATESIDE, PHONE, and PAGER). On the other hand, the information that is only partly dependent on the MME_ID (each individual employee) used to populate POSITION, POSITION_DES, and Notice that both tables contain the column MME_ID.

This is the primary key of each table and is used to match corresponding data between the two tables. CUSTOMER_TAB split into two tables called CUSTOMER_TAB endorsers_TAB. What took place is similar to what occurred in townspeople_TAB. Columns that were partly dependent on the primary key were directed to another table. The order information for a customer is dependent on each CUTS_ID, but does not directly depend on the general customer information in the original table. The third normal form's objective is to remove data in a table that is not dependent on the primary key.

Figure 4 illustrates the third normal form. Another table was created to display the use of the third normal form. EMPLOYEE_PAY_TAB is split into two tables, one table containing the actual employee pay information and the other containing the position descriptions, which really do not need to reside in The POSITION_DES column totally independent of the primary key, MME ID. Naming conventions are one of the foremost considerations when we are going to normalize a database. Names are how we will refer to objects in the database.

We should give our tables names that are descriptive of the type of information they contain so that the data we are going to business's first steps toward a successful database implementation, which need to take to the great consideration. Also referential integrity simply means that the values of one column in a table depend on the values of a column in another table. For instance, in order for a customer to have a record in the ORDERS

TAB table, there must first be a record for that customer in the CUSTOMER_TAB table.

Integrity constraints can also control values by restricting a range of values for a column. The integrity constraint should be created at the table's creation. Referential integrity is typically controlled through the use of primary and foreign keys. In a table, a foreign key, normally a single field, directly references a primary key in another table to enforce referential integrity. In the preceding paragraph, the CUTS_ID in ORDERS_TAB s a foreign key that references CUTS_ID in CUSTOMER_TAB. 3) Although most effective databases are normalized to some point and extent, there is one substantial downside of a normalized database: reduced database performance. In summary, a normalized database requires much more CAP], memory, and 1/0 to process transactions and database queries than does a demoralized database. Regardless of how deep we decide to normalize, there will most always be a trade-off, either between simple maintenance and questionable performance or complicated maintenance and better performance.

At the end, skilled individuals or team of expert designing the database must decide, and that person or team is responsible on final submission of normalized database and its right process and outcome.