Abstract

The main theme of Online Grocery Shop Management application is to maintain all kinds of grocery products accordingly with the correct information so that a user can buy his desired products and know when it will be delivered to him and by whom. All of these are managed through a database. The database is divided based on different conditions which are known as fragments and these fragments are kept at different locations which has Database Management System to deal with the data. The idea of dividing/fragmenting the data makes the system reliable, fast with better response. In case of database failures, the system remains functional though it may reduce performance.

Motivation

The idea of the project comes from one of the biggest online market place amazon.com. Amazon the biggest online market place uses their own Distributed Database Management System. With 0.5 billion products selling all over the world, Amazon needs their system to be very fast in case of data insertion and retrieval, reliable, scalable enough to handle increased data volume. Seeing Amazon in real life we tried to implement a smaller conceptual version of Amazon through our project. Though amazon doesn’t share their methods of creating a distributed system, we have made some assumptions on how their system may work in a distributed scenario. Based on our assumption, we try to implement our grocery management system.

ERD

Transparency

There are three levels of transparency in distributed database.

* Level 1: Location Transparency
* Level 2: Fragmentation Transparency
* Level 3: Local Mapping Transparency

In Level 3 transparency, location and fragmentation details are available to application user. In our project we divided the data into different fragments and kept them in different locations. Most of the fragmentations were Horizontal fragmentation based on the location of the buyer and seller. These fragments were kept in different sites. For example:

Product1: Select \* from Product where Location = ‘Dhaka’;

Site Link

Function

* CountCartItems()
  + Usage: Counts the total items in a given cart.
  + Parameters: Cart\_ID
  + Return Value: Number of items in the cart.

\*\*Operator tree goes here

* GetDateReceivable()
  + Usage: Finds the receivable date or delivery date of a given cart.
  + Parameters: Cart\_ID
  + Return Value: Expected delivery date of the cart.

\*\*Operator tree goes here

Procedure

* GetProductsOfCategory()
  + Usage: Gets all the products of a given category.
  + Parameters: Category\_Id
  + Output: All the products with details of that category.

\*\*Operator tree goes here

* ShowDeliveryDetail()
  + Usage: Show the delivery details of a given Cart
  + Parameters: Cart\_ID
  + Output: Delivery details of the cart.

\*\*Operator tree goes here

* ShowProductDetails()
  + Usage: Show the details of a given product
  + Parameters: Product\_Id
  + Output: All the details about that specific product such as name, details, price, availability etc.

\*\*Operator tree goes here

Trigger

* ProductOutOfStock
  + Usage: Gives a warning if quantity of product goes less than 2.
* ProductAdded
  + Usage: Triggered if product is added into the product table.

Package

Effect of Update