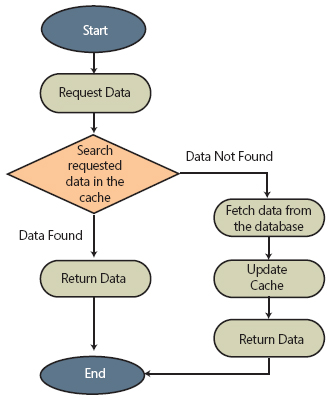
Кеширането на информацията която се взема от базата данни е много важно във всеки проект. SQL-a е ограничен по брои конекции които могат да се правят. От друга страна няма смисъл едни и същи данни да хвърчат по мрежата постоянно на пред на зад. Кеша помага да се намалят броя конекции към базата и да се забърза програмата ни. Винаги задължително трябва да се мисли за това подобряване. Хардуера е физически ограничен и един реален сървър, който хоства стотици сайтове много бързо ще се задръсти ако те не са оптимизирани. Неосъществените връзки към базата се изчакват в опашка. Става забавяне и после се чудим, защо ни работи бавно сайта. Хайде представете си сега 60% от сайтовете на този сървър да нямат кеш и едно и също нещо да се вика отново и отново от всеки един потребител минал за няколко секунди през него.

Обикновенно кеша се прави със статична променлива като всеки път се проверява – има ли вече данни в нея: ако да се вземат от там, ако не се прави връзка с базата.



Да вземем за пример един dropdown контрол които показва списъл от градове. Градовете не са информация която се сменя често. Това са по-скоро статични данни. В този случай се прави кеширане. За първи път когато на програмата и трябва този списък се прави връзка до базата, изтегля се и се пази в статична променлива. Може да се ползва в комбинация с **Lazy design pattern.** Вземат се данните със забавяне. Ако са нужни на програмата. Ако не са нужни – никои не е отворил формата която листва градовете, защо трябва те да се теглят и държът в кеша? Едно е 1000 потребителя едновременно да направят връзка с базата за да си изтеглят листа с градове друго е само първия да го дръпне и после кеша да си го раздава на всички останали. Но може и без Lazy ☺ Част от данните да се заредят още при старта на програмата а не да чакат първия потребител. Нормално би било това да е информация която ще е постоянно показвана на пример някакви менюта на първата страница на един сайт. Всичко зависи от това каква ни е целта и какво искаме да направим.

Кеша трябва да се защити да не може да се проемня от всякъде. Най-лесно това може да се направи със **Singleton design pattern.** Не би трябвало логиката на клиента да променя кеша както му хрумне. Само load и change( save, edint, delete ) операциите към базата данни трябва да променят кеша. Тук много голямо внимание да се обрърне на синхронизацията между тях двете. Ако не са еднакви данните в базата и в кеша може да стане голям проблем за работата на програмата. Ние виждане на екрана това което се намира в кеша, но ако някак си се е записала информация в базата и не е била дописана или променена в кеша става проблем. При web сайтове е по-лесна синхронизацията понеже статичните променливи се пазят на сървъра и са общи за всички клиенти. При desktop/mobile програми или сайтове работещи с данни от webservices трябва много да се внимава. Там статичните променливи може по подразбиране да са си валидни за цялата програма но на нея имаме само един потребител – този на локалната машина или на мобилния телефон. Ако той промени информация към базата тя трябва да се разпространи към останалите потребители. Тук на помощ идва **Оbserver design pattern.** Чрез него можем да изпращаме съобщения при промяна на информацията до останалите програми които ползват нашия webservice.