TVORBA PROGRAMOV V DATABANKOVOM PROSTREDÍ I D M S

Ing. Jozef Kováč,
EVÚCHP, Bratislava

Úvod

Rozvoj zavádzania databankových systémov a tým aj novej technológie spracovania dát je za posledné roky u nás očividný. Výkonné počítače radu JSEP I a II s dostatočnou operačnou pamäťou s 29 a 100 MB diskami umožňujú pristúpiť k implementácii systémov riadenia bázy dát pri riešení projektov ASR.

Nesporne najviac rozšíreným SRBD je u nás systém IDMS, ktorý koncom roku 1982 mal už vyše 90 inštalácií. Treba ovšem priznať, že veľká časť užívateľov sa od zavedenia a skúšania tohoto systému ďalej nedostala. Jednou z príčin oneskorenej realizácie databankových projektov je aj nedostatok skúseností,ktorý vedie k určitej opatrnosti až obavám, ako aj nutnosti odskúšavanie jeho funkcií.

Jednou z oblastí riešenia projektov, kde si SRBD vyžaduje nové vedomosti a prístup k práci je aj programové riešenie projektu.

V našej organizácii sme pristúpili k riešeniu prvého projektu v roku 1979 v rámci riešenia odborovej úlohy "Budovanie ASR VHJ Slovchémia". V dnešnej dobe je v stave rutinnej prevádzky päť projektov a ďalšie dva sú v štádiu technického a vykonávacieho projektu. V súčasnosti je vytvorených vyše 500 programov, ktoré pracujú s bázou dát v režime aktualizácie a výběru dát. Programy sú písané v programovacom jazyku PL/I optimalizujúca verzia a časť výběrových programov je vytvorená výstupným generátorom CULPRIT.

Báza dát je spracovávaná v dávkovom aj interaktívnom režime. Pre interaktívne spracovanie používame systém CICS, ktorý komunikuje s terminálmi v lokálnej aj diaľkovej sieti. Ako vzdialené terminály na podnikoch VHJ používame ďalekopisy T 100 a v lokálnej sieti poľské obrazovkové terminály MERA 7900.

1. Priebeh riešenia projektu s databankovým systémom

Databanková technológia prináša do zvyčajného postupu riešenia technického a vykonávacieho projektu určité zmeny. Princíp integrovanosti, ktorý so sebou banka dát prináša sa netýka iba údajov základne-bázy dát, ale aj procesu riešenia projektu a spolupráce pracovníkov jednotlivých profesií - analytik, programátor, správca banky dát.

Návaznosť jednotlivých fáz riešenia projektov a účasť pracovníkov, tak ako ju máme organizačne zabezpečenú, je znázornená na obr.1.

V priebehu riešenia technického projektu je rozhodujúcou otázka návrhu údajovej základne. Oproti klasickému prístupu sa líši nielen rozsahom navrhovaných typov dát, ale aj vytváraním štruktúry medzi nimi. Preto v prostredí banky dát hovoríme o vytváraní štruktúry bázy dát.

Existencia štruktúry dát má veľký význam pre technológiu programovania. Úlohou programu nie je teraz realizovať vzťahy medzi dátami v algoritme programu, ale rešpektovať tieto vzťahy, ktoré sú v tejto štruktúre zachytené. Programátor nemože získať všetky potrebné položky prečítaním jednej vety súboru, ale prečítaním viacerých viet, pričom ich získava pohybom v báze dát-navigáciou. Preto sa pri návrhu štruktúry bázy dát vytvára v značnej miere aj návrh algoritmu programov. To si vyžaduje samozrejmú účasť hlavného programátora úlohy ako aj správcu banky dát.

Vo fáze programového riešenia úlohy je vplyv použitia SRBD (systém riadenia bázy dát) najvýraznejší. Znalosť jazyka pre manipuláciu dát (DML) je takisto potrebná ako znalosť hostiteľského programovacieho jazyka. Potrebná je aj znalosť jazyka pre definovanie dát (DDL), ktorá je nutná k tomu, aby programátor dokonale pochopil štruktúru bázy dát. Znalosť štruktúry mu je vodítkom pre spracovanie údajov v báze dát.

Vo fáze skúšobnej prevádzky dochádza k preverení spoľahlivosti programov na reálnych dátach v prostredí produktívnej bázy dát.

Je nutné rozšíriť pojem spoľahlivosti programu o prvok jeho efektívnosti. Program nie je spoľahlivý len keď dokáže správne spracovať dáta, ale až vtedy keď ich spracováva efektívne, s minimálnym počtom prístupov na disk.

Vo fáze rutinnej prevádzky môžu nastať situácie, ktoré programátor nedokázal pri logickom ladení predvídať a preto môže dojsť k abendom programov, ktoré už dlhšiu dobu spoľahlivo pracovali. Môžeme uviesť prípad, keď došlo k abendu programu, ktorý už vyše roka s frekvenciou niekoľkokrát mesačne spracovával väčšie objemy dát. Príčina, na ktorej zlyhal, bola vytvorená výnimočnou situáciou v štruktúre dát v báze dát v kombinácii so situáciou v poradí vstupných dát.

Ďalšou starostlivosťou o programy v rutinnej prevádzke je sledovanie ich efektívnosti, a to najmä po väčšom zaplnení bázy dát, kedy sa môže prejaviť dovtedy skrytý nedostatok v programe. Úlohou správy banky dát je uskutočniť optimalizáciu takýchto programov.

2. Zabezpečenie programového riešenia projektu

Pred zahájením programových prác je nutné, aby predchádzajúca fáza návrhu štruktúry bázy dát bola definitívne vyriešená. Neukončenosť návrhu, resp. chyby v návrhu, ktoré sa odhalia až pri programovom riešení zvyšujú časovú a pracovnú náročnosť programovania. Dá sa povedať, že zmena logickej štruktúry na jednom mieste môže ovplyvniť všetky programy, ktoré cez túto časť bázy dát navigujú. Väčšina parametrov fyzickej štruktúry nemá vplyv na stavbu programu, preto je aj možné optimalizovať fyzickú štruktúru produktívnej bázy dát v priebehu jej rutinného využívania, bez dopadu na programy.

Navigácia v báze dát

Pojem navigácie v báze dát sme už uviedli. V klasickom súbore spracovanie dát znamenalo sekvenčné čítanie alebo zápis radu viet. V báze programátor spracúva, alebo prechádza postupnosťou rozných typov viet, ku ktorým sa dostáva po určitých hierarchických cestách. Táto hierarchia predstavuje logické vzťahy medzi vetami, za ktorými

treba vidieť vecnú stránku - vecné hľadisko nadriadenosti a podriadenosti dát. Navigácia v báze dát sa dá prirovnať k pohybu v trojrozmernom priestore. Táto predstavivosť je pre programátora veľmi užitočná, lebo mu umožní ľahšie pochopenie možných situácií v báze dát, ktoré musí v programe ošetriť.

Poznanie vecného hľadiska spracovania je takisto dôležité, lebo napomáha pochopiť spôsob navigácie v programu. Preto je potrebné poznať nielen časť bázy dát, ktorú má program spracovať, ale aj tú, cez ktorú bude navigovať. V zásade sa dá povedať, že treba poznať štruktúru bázy dát v rozsahu subschémy, hoci subschéma môže obsahovať väčšiu časť bázy dát než akú potrebuje daný program. Schémadiagram bázy dát v IDMS je dostatočným prostriedkom pre pochopenie obsahu bázy dát a jej spracovania. Možno s ňou dokonca komunikovať aj s užívateľom, ktorý o princípoch automatizovaného spracovania nemá predstavy a ovláda iba vecné hľadisko.

Efektívnosť programu

Pod pojmom efektívnosti chápeme nároky programu na počet prístupov na disk a z toho vyplývajúcu dobu spracovania. Efektívnosť spracovania je v prvom rade daná efektívnosťou fyzickej štruktúry dát. Nás však zaujíma teraz iba otázka tvorby efektívneho programu.

Navigácia v báze dát poskytuje viaceré možnosti pohybu v báze a tým aj rozne spôsoby ako program bude dáta spracovávať. Možno povedať, že existuje riešenie s najmenším nutným počtom prístupov na disk a toto by mal programátor hľadať. Pri jednoduchej štruktúre väčšinou nie je problém takto program vyriešiť. Náročnejšie je riešenie pri zložitejších štruktúrach a pri spracovaní viacerých typov viet.

Veľmi dôležité sú vedomosti a určité praktické skúsenosti programátora, preto u začínajúcich programátorov je potrebná spolupráca a kontrola pracovníka správy banky dát.

2.1. Přístup k tvorbe programov

Chybový kód

Jednou z nových čŕt databankového programovania je ošetrovanie

chybového kódu, ktorý SRBD odovzdáva programu po vykonaní každého DML príkazu. Niektorí programátori si zo začiatku sťažujú na túto povinnosť a posudzujú túto skutočnosť ako záťaž pre programátora, od ktorej boli v klasickom prostredí odbremenení. Treba si uvedomiť, že chybový kód je dôležitá informácia o tom ako sa náš príkaz vykonal a je výsledkom kontroly, ktorú systém vzhľadom na existujúcu situáciu v báze dát vykonal. Systém takto veľmi účinne predchádza chybám v spracovaní bázy dát. Čím bohatšia je paleta oznamov,tým väčšia je kontrolná činnosť systému a tým väčšia je aj jeho spoľahlivosť.

Programátor musí predvídať vznik niektorých situácií pri spracovaní bázy dát a tieto situácie ošetriť. To znamená, že musí rozhodnúť, či program môže pokračovať, alebo musí svoju prácu ukončiť. Ďalej musí poskytnúť užívateľovi dostatočný oznam a informovať ho tak o príčine neúspešného spracovania. Táto môže byť často zavinená chybnými dátami užívateľa a preto mu treba poskytnúť informáciu,aby mohol chybu odstrániť a dáta znova zadať na spracovanie.

V interaktívnom režime dochádza k dialógu s užívateľom v priebehu ktorého mu treba dať možnosť bez prerušenia dáta opraviť, alebo pokračovať v ďalšej činnosti. IDMS vlastní príkaz (ROLLBACK CONTINUE), pomocou ktorého možno odstrániť účinky programu,vrátiť ho naspäť po určitý kontrolný bod a potom pokračovať ďalej. Tento kontrolný bod si musí vytvárať program.

Protokol zo spracovania

V dávkovom režime je potrebné, aby program vytváral zo spracovania protokol pre užívateľa, z ktorého tento môže spoľahlivo zistiť ako mu prebehlo zadané spracovanie. Tento protokol je dôležitý najmä u aktualizačných programov.

Protokol by mal obsahovať dátum a čas spracovania, meno programu, opis vstupných dát, diagnostické oznamy a záverečnú štatistiku o práci systému, ktorú poskytuje SRBD. Diagnostické oznamy musia byť zrozumiteľné užívateľovi a musia sa vyvarovať technickým pojmom. Forma protokolu musí zabezpečovať prehľadnosť a ľahkú orientáciu, v opačnom prípade odradíme užívateľa od jeho kontroly a potrebnej

opravy spracovaných dát.

Treba si uvedomiť, že užívateľ takto vykonáva funkciu gestora dát a zabezpečuje popri správe banky dát vecnú integritu a aktuálnosť údajov v báze dát.

Interný výkon DML príkazov

Spôsob, ako SRBD vykonáva jednotlivé príkazy, je popísaný v príručkách systému. Znalosť týchto pravidiel programátorom je potrebná, pretože v niektorých prípadoch existuje viacero spôsobov, ako sa tieto príkazy vykonávajú. A to v závislosti od parametrov štruktúry bázy dát ako aj od spôsobu navigácie programu v báze dát.

SRBD udržuje počas práce programu tzv. mechanizmus aktuálnych viet (currency), ktorý predstavuje zapamätávanie si určitých pozícií v báze dát, cez ktoré program navigoval. Výkon každého príkazu sa viaže na momentálnu situáciu v tomto mechanizme, preto je jeho znalosť podmienkou programovania. V opačnom prípade dochádza často k situáciám, že program prešiel navonok správne, ale z vecného hľadiska bolo spracovanie chybné.

Rešpektovanie systému aktuálnych viet a znalosť interného výkonu DML príkazov poskytuje možnosti optimalizácie programu z hľadiska jeho časovej efektívnosti. Informácie v mechanizme aktuálnych viet sa po každom DML príkaze menia. V programe však môže vzniknúť situácia,kedy je potrebné vrátiť sa na pozíciu, ktorú si systém z uvedeného dôvodu už nepamätá. V takom prípade je možné s výhodou využiť možnosť odpamätania si adresy týchto pozícií a potenciálnym návratom, do vlastnej položky v programe.

Odkladanie informácie o takýchto pozíciách, v podstate databázových kľúčov viet, je niekedy veľmi výhodné uskutočniť aj pre ďalšie programy v budúcich spracovaniach bázy dát. Pre tieto programy zabezpečíme takto možnosť rýchleho vstupu do bázy dát na potrebné miesta, bez toho aby museli absolvovať navigačnú cestu, ktorá by bola časove náročnejšia. Tento mechanizmus je veľmi účinný v prípade setov s dlhými reťazcami viet, kedy je potrebný vstup na pozície vnútri týchto setov.

Sposob prístupu do bázy dát

Pri návrhu jednotlivých programov je potrebné zvážiť sposob, akým bude program do bázy dát vstupovať: súbežný (SHARED), chránený (PROTECTED) alebo jedinečný (EXCLUSIVE). Toto rozhodnutie sa robí vo väzbe ku sposobu, akým program bude s bázou dát pracovať: aktualizácia (UPDATE) alebo výber (RETRIEVAL). Sposob prístupu nám zabezpečuje ochranu nášho programu pred súbežnou prácou iných programov na danej oblasti bázy dát.

Túto ochranu môže realizovať IDMS na dvoch úrovniach: úroveň vety alebo úroveň oblasti.

Pri potrebe ochrany na úrovni celej oblasti použijeme sposob PROTECTED (zabránenie súbežnej práci iných aktualizačných programov) alebo EXCLUSIVE (zabránenie súbežnej práci akéhokoľvek iného programu). Tento sposob ochrany je najjednoduchší, je menej náročný na čas CPU (centrálnej jednotky) a vhodný je pri dávkovom spracovaní bázy dát.

V prípade interaktívneho spracovania bázy dát, by sme však uvedeným postupom znížili priepustnosť bázy dát, ktorá by bola takto v jednotlivých časových úsekoch prístupná iba jednému užívateľovi a ostatní by museli čakať. IDMS umožňuje od verzie 5.0 použiť SHARED prístup do bázy dát a vykonávať pritom ochranu na úrovni viet.

V takom prípade pracujú s oblasťou viacerí užívatelia so sposobom UPDATE, pričom každá aktualizovaná veta zostáva vyhradená len programu, ktorý ju menil , a to až do okamihu ukončenia tohoto programu.

Tento režim práce je však značne náročný na spotrebu CPU času. Preto ho volíme iba pri práci s oblasťami v interaktívnom režime a to pre interaktívne programy, ale aj dávkové programy, ktoré môžu pracovať v priebehu dňa počas interaktívnej prevádzky. Programy ktoré pracujú až po skončení interaktívnej prevádzky, alebo pracujú s oblasťami,kde je len dávkový režim, používajú zásadne PROTECTED sposob prístupu do bázy dát.

Štatistika o práci programu

IDMS počas práce každého programu vytvára priebežne o jeho činnosti štatistiku, ktorá charakterizuje jeho činnosť a stupeň aktivity v báze dát. Tieto údaje sú veľmi cenné pre nasledovné použitie: ladenie algoritmu programu, ladenie efektívnosti programu, posudzovanie priepustnosti bázy dát a posudzovanie efektívnosti programov v obdobi rutinnej prevádzky bázy dát.

Preto je potrebné zabezpečiť, aby programy pred svojim ukončením práce s bázou dát túto štatistiku vytlačili do protokolu o spracovaní.

Ošetrenie spracovania veľkých dávok dát

Spracovanie veľkých dávok údajov je charakteristické pre obdobie napĺňania bázy dát. Napĺňanie bázy dát však môže byť dlhodobým javom-báza dát sa už rutinne využíva, ale nahrávanie väčších objemov dát pokračuje postupne s tým, ako užívateľ rozširuje spracovanie na ďalšie objekty modelovanej dátovej reality.

Pri spracovaní takýchto objemov dát vzniká problém možnosti výskytu chyby, pri ktorej dojde k prerušeniu práce programu a k jeho ahendu. V takom prípade sú účinky programu automatický odstránené (pro prevádzke v režime centrálnej verzie), alebo sa musia odstrániť manuálne (režim lokálnej verzie). V prípade, že tá časť spracovania do okamihu výskytu chyby bola správna,dochádza k jej strate a potrebe jej zopakovania. Ak k tejto chybe došlo na konci veľkej dávky dát toto zbytočné opakovanie spracovanie je citeľné. K tejto situácii môže dojsť aj z dovodov výpadku prúdu, zlyhaniu operačného syatému apod.

Systém IDMS umožňuje priebežne vytvárať kontrolné body (príkazom COMMIT), ktoré zabezpečia, že uvedená obnova bázy dát sa vykoná len po posledný takýto bod. Správny moment vytvárania týchto kontrolných bodov musí zvážiť programátor.

Špecifiká interaktívnych programov

Tvorba programov pre interaktívny režim si vyžaduje rešpektovanie ďalších požiadaviek. Krátka doba odozvy potrebuje,aby program pracoval čo najrýchlejšie. Z tohoto hľadiska je potrebné brať do úvahy aj vlastnosti programovacieho jazyka, ale hlavne spôsob navigácie v báze dát. Efektívna navigácia v báze dát má u tohoto typu programov väčší účinok ako u dávkových.

Predpokladom pre takúto navigáciu je optimálnosť štruktúry bázy dát, ktorá by mala predovšetkým vyhovovať potrebám interaktívnych programov. Niekedy však nie je možné zohľadniť pri tvorbe štruktúry bázy dát všetky požiadavky interaktívnych programov, prípadne tieto sa navrhujú a tvoria neskoršie, keď báza dát už existuje. V takých situáciách je vhodným riešením vytváranie špeciálnych oblastí údajov, ktoré slúžia iba pre potreby interaktívneho prístupu v režime dotazov. Aktualisácia takejto oblasti sa robí spravidla po skončení aktualizácie primárnej oblasti. Dochádza tu k posunu aktuálnosti takejto oblasti pre užívateľa, ktorá odráža skutočnú situáciu v realite spravidla s jednodenným oneskorením. Ak pre užívateľa tento posun nie je podstatný je možné toto riešenie realisovať.

Ďalšie urýchlenie interaktívnych programov pomocou vhodnej štruktúry bázy dát sa dá dosiahnuť zabezpečením vstupných bodov do bázy slúžiacich len týmto spracovaniam. Tieto vstupné body je vhodné vytvárať aj za cenu redundancie kľúčových položiek viet a obsadzovania ďalšieho priestoru na disku.

2.2 Organizácia ladenia programov

Logické ladenie programu musí zabezpečiť nielen správnu prácu algoritmu programu, ale ako sme už uviedli aj dosiahnutie jeho efektívnej práce. Predpokladom pre odladenie programu je vytvorenie testovacej bázy dát, ktorá musí vytvárať všetky situácie v štruktúre dát, aké v nej môžu nastať.

Testovacia báza dát, ktorá je rozmerom podstatne menšia, ako budúca produktívna báza dát, musí byť jej reprezentatívnou vzorkou. Preto údaje pre jej vytvorenie by mal pripravit analytik aj programátor.

Predvídavosť pri príprave týchto dát zabezpečí, že program sa v rutinnej prevádzke nestretne so situáciami, ktoré sa pri jeho ladení nevyskytli. Vytvorenie testovacích dát pre bázu dát je náročnejšie ako pre ladenie programu s klasickým súborom.

Pri tvorbe aktualizačných programov je nutné postupovať v poradí podľa hierarchickej štruktúry bázy dát. Ladiť program pre určitý typ vety možno až vtedy, keď sú jej vlastnícke vety v báze dát už nahrané. Preto sa práca v týme musí rozdeliť tak, aby bola táto návaznosť vytvárania programov zabezpečená.

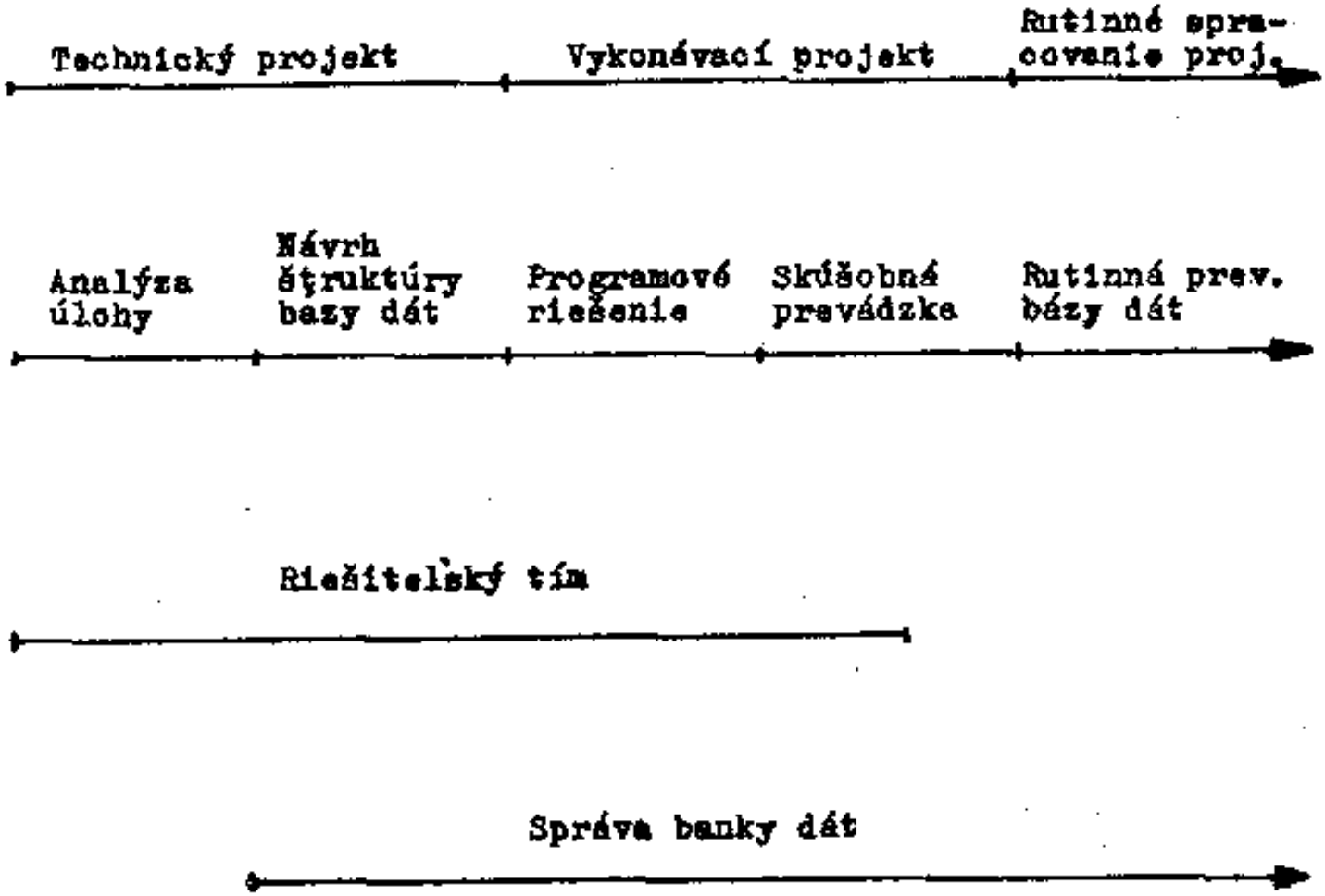
Testovaciu bázu je potrebné udržovať permanentne, aj počas rutinnej prevádzky bázy dát. V prípade, že zlyhá program, ktorý je v rutine, alebo je potrebná jeho úprava podľa zmenenej požiadavky užívateľa, je nutné ho odskúšať na testovacej báze dát.

Pri logickom ladení aktualizačných programov sa nezaobídeme bez výpisov z bázy dát, ktoré musíme uskutočniť pred a po práci testovaného programu. Iba takýmto spôsobom sa môžeme s istotou presvedčiť, či program pracuje správne. Je preto potrebné mať k dispozícii vhodný vypisovací nástroj. Poslúžiť môže aj generátor CULPRIT, alebo vlastné vypisovacie programy.

Po prvých skúsenostiach, kedy správa banky dát zhotovovala pre každý projekt programátorom špeciálne vypisovacie programy, sme zhotovili univerzálny vypisovací program. Tento program môže pracovať po malom rozšírení tabuľkovej časti s každou bázou dát, pričom poskytuje niekoľko typov výpisov podľa požiadavky jeho používateľa.

Záver

Programovanie v databankovom prostredí prináša nové požiadavky na programátora ako aj požiadavky, ktoré môže zabezpečiť len pracovník správy banky dát. Poznanie týchto požiadaviek a zabezpečenie ich riešenia sú nutné pre úspešný priebeh riešenia projektu a jeho rutinnú prevádzku.



Obr. 1: Priebeh riešenia projektu s použitím databankovej technológie.