SPECIFIKACE VSTUPŮ A VÝSTUPŮ JAKO VÝCHODISKO NÁVRHU DATOVÉ ZÁKLADNY

R. Zuczek

V článku je popsán určitý přístup k návrhu datové základny. Východiskem návrhu je specifikace vstupů a nejdůležitějších výstupů, popřípadě seznam dotazů.

1. Úvod

Bezprostředním popudem k napsání tohoto příspěvku je diskuse kolem metodologie návrhu datových základen, vedená v nejbližším okolí autora. Metodologie, o které se diskutuje říká, že je třeba provést analýzu daného výseku reality, najít relevantní entity, jejich atributy, vztahy mezi entitami a taky atributy těchto vztahů /1/. Entity se pak zobrazí do vět, atributy do položek vět, entitní vztahy se realizují sety, a ve složitějších případech je tato transformace složitější. Načrtnutá metodologie je zde příkladově spojená s použitím databázového systému typu DBTG. Volba databázového systému se nicméně projevuje až u zmíněné transformace, ne v první části návrhu.

V praxi se setkáváme s různými obměnami této metodologie, viz např. /5,6/. Jejich společným znakem je používání pojmů entita, atribut, vztah. Tyto metodologie jsou poměrně abstraktní, takže jim rozumí málokterý analytik, o konzultujícím uživateli nemluvě. Naše hlavní námitka je však jiného druhu. Totiž o obsahu datové základny se rozhoduje, aniž by se zjišťovalo, zda a jak je vůbec možné potřebná data získat. Mnohdy se až dodatečně ukáže, že uživatel, pro nějž celý tento systém děláme, není ochoten nám potřebné vstupy dávat. Nebo se stane, že uživatel není spokojen s výstupy, které lze z datové základny odvodit. Vedlejší potíž (méně důležitá a odstranitelná dodatečně) spočívá v tom, že když datová základna nezachytí správně (optimálně) požadavky uživatele na výstupy, odnese to programátor výstupních sestav, resp. doba výstupu.

V tomto článku popíšeme alternativní způsob návrhu datové základny, resp. celého uživatelského systému. Proces návrhu začíná

u analytika, který s pomocí uživatele vyspecifikuje v první řadě vstupy. Dnešní uživatelé jsou již natolik vyspělí, že vstupním formulářům rozumí a taky chápou, co se z nich dá odvodit a co ne. Po odsouhlasení vstupů a vyjasnění otázky výstupů může být uzavřena dohoda mezi uživatelem a řešiteli systému. Její formulace z hlediska uživatele zní "zde máte vstupy, dělejte si s nimi co chcete, ale chceme z toho mít takové a takové výstupy". Načež následují specifikace vstupů a výstupů, resp. dotazů, a několik dalších důležitých parametrů pro návrh. Teprve nyní se s takto postaveným problémem seznamuje návrhář datové základny. Poskytnutá informace je naprosto postačující na odvození struktury datové základny. Nakonec se se specifikacemi vstupů a výstupů a datovou základnou seznamuje programátor, který završí návrh systému příslušnými programy.

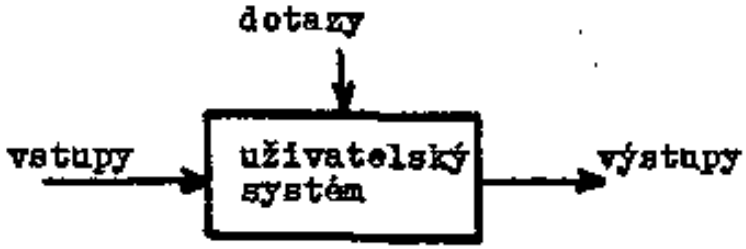
Uvedeme některé výhody popsaného přístupu. Především je větší naděje, že se uživatel s analytikem dohodnou. Analytik nemusí být příliš teoreticky fundovaný, nemusí mít ani zkušenosti s databázovým systémem. Musí však dobře porozumět problému uživatele, musí dobře navrhnout vstupy (mimořádně důležité!), a musí znát pravidla komunikace s návrhářem datové základny a programátorem. Často je práce analytika usnadněná tím, že je třeba převést zpracování z klasického na databázové. Pak jsou vstupy vlastně hotové, i když možná nepříliš vhodně navržené. Samotný problém návrhu datové základny je zadán velmi konkrétně, a jeho vyřešení je záležitost čistě technická.

Nepochybně libovolný praktický postup návrhu uživatelského systému nějakým způsobem přihlíží k možným vstupům a výstupům. Zde předkládaná metodologie se vyznačuje tím, že toto "přihlížení" povyšuje na hlavní princip. V tomto článku se pokusíme tento princip rozvinout a dát mu solidnější základ. Poznamenejme, že náš přístup se označuje jako funkční, na rozdíl od přístupu /1/, který lze nazvat modelový (snaží se "modelovat" realitu). Jedna metoda návrhu patřící do kategorie funkčních je popsána v /4/. Za asistence počítače se struktura datové základny odvozuje na základě dotazů.

2. Základní pojmy

Pod pojmem (databázově orientovaný) uživatelský systém rozumíme systém složený z datové základny a z příslušných aplikačních programů. Příklady uživatelských systémů: systém pro investice, systém základní prostředky, systém MTZ, a z méně typických: systém evidence souborů na magnetických páskách, knihovna zdrojových programů.

Aplikační programy uživatelského systému dělíme na vstupní a výstupní. Vstupní programy jsou takové, které zavádějí vstupní data do datové základny, resp. ji těmito daty aktualizují. Výstupní programy jsou takové, které na základě dotazů kladených těmto programům vyvedou příslušné informace z datové základny ven.



obr.1

Na obr.1 je znázorněn uživatelský systém jako černá schránka, do níž jdou vstupy a z níž vycházejí výstupy jako odezva na dotazy. Naším úkolem je určit vnitřek této schránky, když máme k dispozici popis všech druhů vstupů a charakteristiku nejdůležitějších výstupů (resp. seznam dotazů). Zde nás bude zajímat především otázka návrhu datové základny. U programů si všimneme hlavně souvislosti mezi složitostí vstupních a výstupních programů.

Na jednotlivé vstupy vstupující do černé schránky z obr.1 lze pohlížet jako na záznamy o některých událostech ve vnějším světě. Například může jít o registraci výdeje materiálu ze skladu, nebo o registraci faktu, že bylo vytvořeno nové pracoviště.

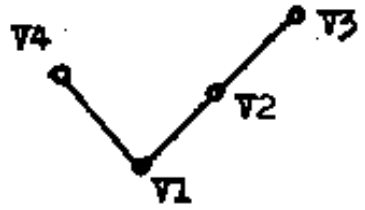
Druhy vstupů budeme označovat V1,V2,... . Budeme předpokládat, že každý vstup je možno charakterizovat jednoduchým seznamem atributů, resp. položek, Vi = Vi(A1,A2,...,An). Symbol Vi(a1,a2,..

...,An) budeme nazývat schématem vstupu. Příklady:

V1(SKLAD,DATUM,RAD,PRAC,CMAT,VYD)
V2(CMAT,SKUP,JCENA,NAZMAT)
V3(SKUP,NAZSK)
V4(PRAC)

Ve vstupu V2 označuje CMAT číslo materiálu, SKUP skupinu materiálu, JCENA jednotkovou cenu, NAZMAT název materiálu. Vstup V2 je registrací faktu, že "od nyní" existuje v některém skladu materiál s daným číslem, názvem, jednotkovou cenou, a patřící k dané skupině materiálů. Vstup V1 je registrací výdeje materiálu CMAT, dne DATUM, ze skladu SKLAD, na pracoviště PRAC, v množství VYD. RAD je pořadové číslo výdeje v rámci skladu a dne.

V2 je příklad kmenového vstupu, protože se takové vstupy zavedou najednou v dávce a později vstupují sporadicky. Vstup V3, ve kterém SKUP je číslo skupiny a NAZSK je název skupiny materiálů, je "kmenovější" než V2, protože musí být zaveden dříve než V2. Máme tedy omezení: nesmí vstupovat V2 s hodnotou SKUP=s pokud dříve nevstoupil vstup V3 s hodnotou SKUP=s. Vstup V1 je "pohyblivější" než V2 a V3 a bude zřejmě vstupovat rovnoměrněji. Opět můžeme formulovat přirozená omezení, že nesmí vstoupit V1 s hodnotou CMAT=c dříve než vstup V2 s hodnotou CMAT=c. Tuto závislost mezi vstupy můžeme znázornit graficky. Na obr.2 je pomocí hran vyznačeno, že V3 je kmenovější než V2, a že V2 je kmenovější než V1. Vstup V4, ve kterém PRAC je číslo pracoviště, je kmenovější než V1. Nesmí vstoupit výdejka s hodnotou PRAC=p dříve než V4 s hodnotou PRAC=p.



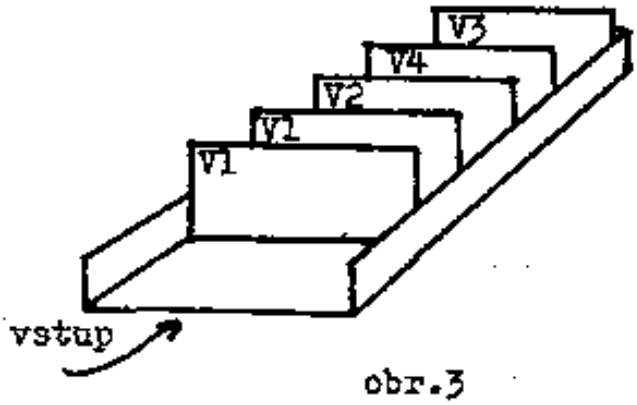
obr.2

Graf z obr.2 vyjadřuje pouze část omezení, které bude muset návrhář datové základny a programátor respektovat, a které patří ke specifikaci vstupů. O funkční závislosti mezi položkami vstupů, resp. o klíčích, které taky představují jistý druh omezení, budeme mluvit později.

Vstupy se běžně realizují pomocí štítků, přitom rozvržení položek na štítku je určeno schématem vstupu. Původním zdrojem štítků jsou formuláře, a přesněji řečeno řádky formulářů. Jinou formou vstupů mohou být zprávy (transakce) z terminálů. Papírové formuláře jsou pak vystřídány formuláři zobrazenými na obrazovce.

3. Vstupní část datové základny

Datovou základnu dělíme na vstupní část a na redundantní část. Pro návrh datové základny je rozhodující návrh její vstupní části. Modelem vstupní části je "vstupní kartotéka" (obr.3). Je to vlastně soubor, do kterého se ukládají vstupy (vstupní štítky) v pořadí jak jsou čteny na vstupu. Vstup uložený do vstupní kartotéky budeme nazývat záznamem.



obr.3

Představme si, že datová základna uživatelského systému má formu takovéto vstupní kartotéky. Vstupní programy to mají zřejmě velice jednoduché (odhlédneme-li od vstupních kontrol), protože nedělají nic než zařazují vstupní štítky na začátek kartotéky. Horší to mají výstupní programy. Jednoduchý dotaz "jaký je součet výdejů materiálu CMAT= pro období mezi DATUM= a DATUM=" vyžaduje průchod přes celou kartotéku, resp. přes všechny záznamy typu V1. Ještě horší by to bylo s dotazem "najít součet výdejů v korunách za pracoviště PRAC=, rozdělený podle jednotlivých skupin materiálů".

Specifickým rysem vstupní kartotéky je to, že se v ní zobrazuje kompletní časový vývoj událostí ve vnějším světě. Vstupní kartotéka je tedy úplná, obsahuje všechno, umožňuje odpovědět na jakýkoliv dotaz. (Aniž bychom přesně definovali pojem dotaz, je

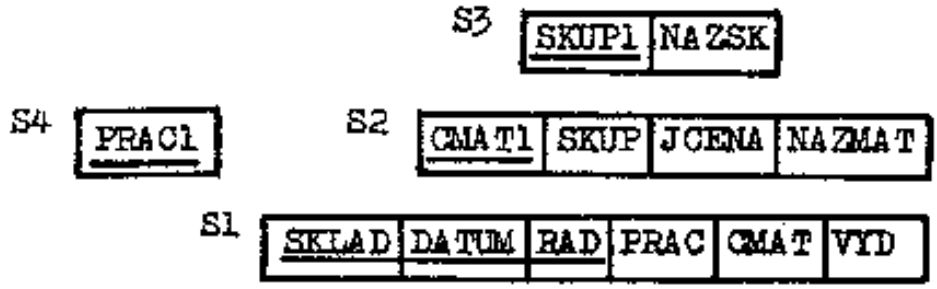
jasné, že veškeré dotazy se formulují v termínech vstupů a jejich položek, a snad taky s pomocí funkcí "součet","maximum", apod., a s použitím určitých "organizačních" klauzulí.) Poznamenejme, že za normální vstup bychom mohli považovat i informaci "pracoviště PRAC= bylo zrušeno", která by se vstupním programem normálně zařadila na začátek kartotéky. Mohli bychom se pak ptát "jaký je součet výdejů materiálu od DATUM= za všechna pracoviště, která existovala kdysi a teď již ne". Teoreticky neexistuje omezení na charakter vstupu. Nicméně v praxi uvažujeme pouze vstupy, při kterých datová základna skutečně "roste", spoléhaje na prostředky databázového systému, které umožňují rušit dříve vložené záznamy. Toto zjednodušení si můžeme samozřejmě dovolit, jen když máme od uživatele zajištěno, že podobné dotazy, jako výše uvedený, nebude vyžadovat.

Než se pustíme do realizace vstupní kartotéky prostředky daného databázového systému, srovnejme naši vstupní kartotéku s relačním modelem dat /2,3/. Není třeba dlouho vysvětlovat, že záznamy v kartotéce můžeme považovat za entice relací, pokud k nim připojíme atribut "čítač", který bude mít hodnotu pořadového čísla umístění záznamu v kartotéce. Když tedy ke schématům vstupů V1,V2, V3,... přidáme položky "čítač", dostaneme schémata bázových relací V1',V2',... relační datové základny. Znění naší původní úlohy lze nyní parafrazovat takto: máme navrhnout fyzickou realizaci relační datové základny V1',V2',... .

Začneme od výběru primárních klíčů mezi atributy relací V1', V2',... . Mohli bychom pro tento účel vybrat "čítač" a ostatní atributy (resp. některé z nich) indexovat, ale to nevede k úspornému řešení. Proto vybíráme klíče mezi položkami vstupů V1,V2,... . Informaci o možných klíčích vstupů nám poskytne analytik v rámci specifikace vstupů. Klíč má jednoznačně identifikovat vstup. Formuláře vstupů musí být tedy navrženy tak, aby každý řádek měl svůj jednoznačný identifikátor. Například pro vstup V1 z našeho příkladu by mohla dvojice DATUM,SKLAD identifikovat formulář, a řádky formuláře by se číslovaly 1,2,3,... = RAD . Z důvodů známých z relačního modelu dat, vstupy by měly být vzhledem ke klíči ve třetí normální formě/3/. Tuto podmínku analytici dost běžně respektují aniž si to uvědomují. Například nikdo nenavrhne výdejku materiálu, z níž je třeba děrovat kromě čísla materiálu taky název materiálu

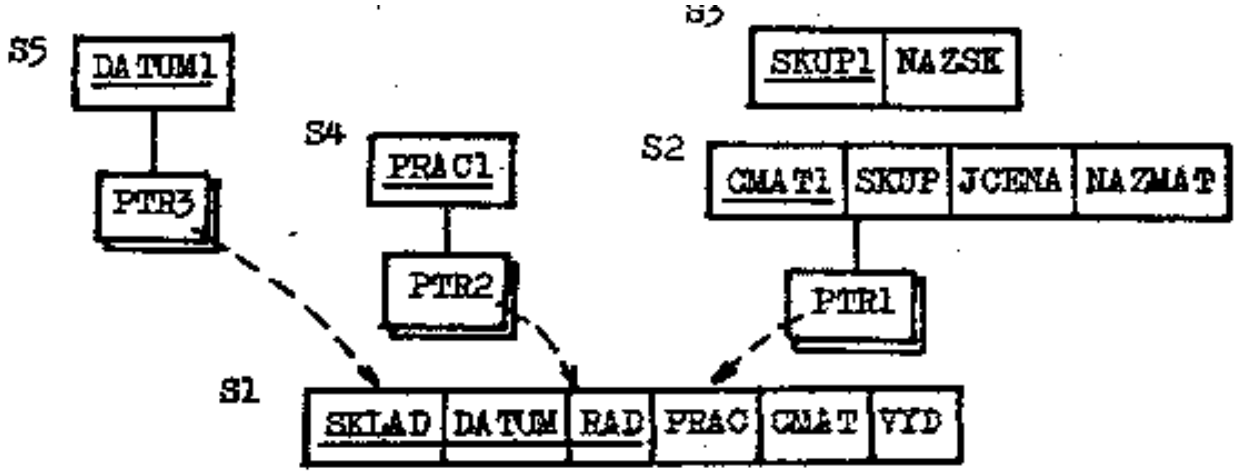
nebo skupinu materiálu. Třetí normální forma představuje poměrně dobrý test na správnou dekompozici vstupů.

Na obr.4 je nakreslena první varianta realizace vstupní kartotéky pro náš příklad s výdejem materiálu. Datová základna je tvořená čtyřmi kořenovými segmenty (budeme používat databázový systém IMS /3,6/, resp. EX /8/), které jsou bezprostředním obrazem našich čtyř vstupů V1,V2,V3,V4. Rozhodli jsme se, že nebudeme realizovat přesně kartotéční model, protože máme bezpečně zjištěno, že detailní informace o vývoji událostí ve vnějším světě uživatele nebudou zajímat. Proto taky zde nemáme zastoupeny položky "čítač". Kvůli rychlému přístupu podle klíčů, zvolíme nejspíš rozptylovou, případně indexsekvenční přístupovou metodu. Vstupní program pro vstup V1 bude pracovat tak, že nejprve přečte S2 a S4 aby zjistil že existuje CMAT a PRAC, a pak teprve zařadí V1 do S1, přičemž duplikát nelze zařadit.



obr.4

V další fázi návrhu přihlížíme k dotazům, které formuloval uživatel a které bude realizovat programátor. Zjišťujeme například, že se nám bude hodit indexování atributů DATUM,PRAC a CMAT v segmentu S1 . Využijeme již existující kořenové segmenty S2 a S4 a definujeme pod nimi podřízené segmenty s pointry na S1 . Index pro DATUM realizujeme celý zvlášť pomocí S5. Výsledek je uveden na obr.5. Poznamenejme, že například u IDMS bychom vytvořené přístupové cesty mohli znázornit méně explicitně nakreslením spojovacích čar z S4,S2 a S5 na S1. Nicméně ve fyzickém popisu databáze by bylo nutno pointry skutečně definovat. Poznamenejme, že položky PRAC a CMAT v segmentu S1 by bylo možné nahradit pointry na S4 resp. S2. Všiměme si taky, že vstupní program má nyní více práce, protože musí zařazovat pointry PTR1,PTR2 a PTR3, a občas taky kořen S5.



obr.5

Rozhodnout které atributy se vyplatí indexovat bývá někdy dost obtížné. V každém případě pro rozhodování je minimálně zapotřebí, aby analytik zjistil údaje o předpokládaných počtech vstupů. Při návrhu struktury datové základny i výst- ních programů nám může pomoci literatura týkající se realizace dotazů v relačních databázových systémech. Pro ucelený přehled a odkazy viz /9/.

4. Redundantní část datové základny

Omezený prostor pro ukládání dat nás bohužel nutí některé části vstupní kartotéky v pravidelných intervalech likvidovat. Téměř bez výjimky jde o ty nejpohyblivější vstupy. S nimi pochopitelně rušíme i příslušné indexy (pointry). Cílem této malé reorganizace je získat zpět prostor pro další vstupy z následujícího období, a taky udržet rychlost odezvy systému v přijatelných mezích. Předtím musíme zabezpečit, aby uživateli zůstala možnost se dozvídat o detailech obsažených ve vstupech, které rušíme.

V našem příkladě bychom mohli koncem každého měsíce zrušit všechny záznamy V1 a nový měsíc začít od začátku. Předtím bychom však tyto záznamy vytiskli (nebo zapsali na magnetickou pásku) setříděně podle různých hledisek, aby uživatel mohl v případě potřeby v takovémto výtisku snadno hledat (resp. tuto informaci dál zpracovávat). Současně bychom mohli zavést do datové základny (z globálního hlediska) redundantní záznamy, obsahující některé součty, když už ne detaily rušené informace. Například by bylo

možné pod kořenem S2 definovat podřízený segment (MES,VYDM), kde MES je měsíc a VYDM je množství příslušného materiálu vydaného v tomto měsíci. Opět argumentem pro zavedení takového či jiného součtového segmentu je konkrétní požadavek na výstup.

Někdy potřebujeme udržovat v datové základně nejen součty hodnot položek, ale i "součty" nebo spíš "extrakty" některých vztahů. Například mezi PRAC1 a CMAT1 existuje vztah, realizovaný prostřednictvím segmentu S1, že pracoviště odebírá několik materiálů, a materiál je odebírán několika pracovišti. Tento vztah můžeme udržovat samostatně, nezávisle na S1. Vždy je třeba se však dívat, zda to je z hlediska požadavků uživatele skutečně potřebné. Čím víc je druhů vstupů (zvláště těch nejpohyblivějších), tím víc je možných vztahů a součtů, a tím větší je nejistota při výběru správného řešení struktury datové základny. Spolehlivým vodítkem je pak pravidlo: držet se, pokud možno, co nejtěsněji vstupní části datové základny. Vedlejším efektem této strategie je pak to, že uživatel má bezprostřední možnost zpětně kontrolovat a opravovat své vstupy.

Nutno poznamenat, že redundantní záznamy zavádíme nejen jako náhradu za rušené nejpohyblivější vstupy. Pro tento prostředek sáhneme také tehdy, když jsou specifikovány složité dotazy, které pomocí pouhé vstupní části datové základny nelze odpovědět dostatečně rychle. Vezměme například již dříve uvažovaný dotaz "najít součet výdejů v korunách za pracoviště PRAC=, rozdělený podle jednotlivých skupin materiálů". Normálně bychom museli projít všechny záznamy S1, na které ukazují pointry PTR2 pod daným pracovištěm, zjišťovat jednotkové ceny čtením S2, součiny VÝD*JCENA přičítat pod příslušnou skupinou v pomocné datové struktuře vytvořené v programu (v hlavní paměti), vnitřně setřídit podle skupin, a až pak zobrazit na výstup. Dejme tomu, že jsme vypočítali, že by to trvalo příliš dlouho nebo že by příslušný program byl nevhodně velký. Řešením problému je zavedení pod kořenem S4 redundantního segmentu (skupina, výdej v korunách) a doplnění vstupního programu pro V1 o část aktualizující tento segment. Zpracování vstupu V1 se tím sice prodlouží, ale o tolik se ušetří ve výstupním programu, kterému stačí tisknout, co již je připraveno v redundantním segmentu. Prodloužení doby vstupu je přece jen snesitelnější, zvláště když tento vstup bude zpracováván dávkově.

5. Závěr

Nejpodstatnějším rysem zde předložené metodologie je představa o vstupu jako o záznamu o nějaké události ve vnějším světě. Tyto záznamy se ukládají do vstupní kartotéky, která představuje model vstupní části datové základny. Jde teď o to, realizovat vstupní kartotéku tak, aby bylo možné efektivně odvozovat vstupy. Existují dva způsoby jak toho dosáhnout: indexováním a zaváděním redundantních záznamů. Tato idea je snadno pochopitelná a umožňuje i méně zkušeným lidem uspokojivým způsobem navrhovat datové základny.

Poznamenejme, že metoda ER-diagramů /1/, od které jsme začali v úvodu, se dá použít i na odvození relační datové základny a tedy i vstupů. Pěkný příklad lze nalézt v /7/ na str. 21-24.

Literatura

1. Chen,P.P.: The Entity-Relationship Model: Toward a Unified View of Data, ACM Trans.Database Syst., Vol.1, March 1976, 9-36
2. Codd,E.F.: A Relational Model of Data for Large Shared Data Banks, Comm.ACM, Vol.13, June 1970, 377-387
3. Date,C.J.: An Introduction to Database Systems, Addison-Wesley, 1976
4. Goerritsen,R.: A preliminary System for the Design of DBTG Data Structures, Comm.ACM, Vol.18, October 1975, 551-556
5. Krejčí,P.: Navrhování datových struktur a jejich implementace, Databanka'81, 1981, Ústí n. Labem, 7-19
6. Martin,J.: Computer Data-Base Organization, Prentice-Hall, 1977
7. Pokorný,J.: Dotazy a odpovědi v databázových systémech, Ústav výpočetní techniky ČVUT, Praha, 1980
8. System EK, Základní popis, OKR Automatizace řízení, Ostrava, 1980
9. Yao,S.B.: Optimization of Query Evaluation Algorithms, ACM Trans.Database Syst., Vol.4, June 1979, 133-155