1. Тарасов Д.О., Основні задачі захисту баз даних // Вісник НУ "Львівська політехніка".— 2000 р., № 406.— с. 216-221. 2. Тарасов Д.О. Аудит баз даних. - Защита информации: Сборник научных трудов.— К.: КМУГА, 2000.— С.136-140. 3. Жежнич П.І., Кравець Р.Б., Пасічник В.В., Пелещишин А.М. Семантично відкриті інформаційні системи // Вісник НУ "Львівська політехніка", 1999р., № 383.— С. 73-84. 4. Жежнич П.І., Кравець Р.Б., Пасічник В.В., Пелещишин А.М. Основні правила побудови семантично відкритих інформаційних систем// Вісник НУ "Львівська політехніка", 1999р., № 383.— С. 84-95 5. Ahmed R., Navathe S.B. Version management of composite objects in CAD databases", Proc. ACM SIGMOD Conf. On the Management of Data, 1991, Pp. 218—227.

УДК 681.3

Н. Б. Шаховська
НУ "Львівська політехніка",
кафедра "Інформаційні системи та мережі"

ЗАСТОСУВАННЯ АПАРАТУ БАГАТОЗНАЧНОЇ ЛОГІКИ У СИСТЕМАХ БАЗ ДАНИХ

Indeterminate and fuzzy data stored in relational database are described in this paper.

Розглянуто способи подання нечіткої та неповної інформації у базах даних за допомогою багатозначної логіки.

Для опису даних та роботи з ними в умовах невизначеності та слабкої структурованості використовують:

- теорію ймовірностей,
- нечітку логіку (нечіткі множини, лінгвістичні змінні, інтервальні оцінки, нечіткі числа, емпіричні оцінки) [18],
- k-значну логіку [18, 29].

Неповнота інформації може зустрічатися на різних рівнях [28]:

- невідомо, чи властивість притаманна ПО – невизначеність на рівні відношення;
- відомо, що властивість притаманна ПО, але невідомо, чи вона притаманна даному об'єктові – невизначеність на рівні кортежів;
- відомо, що властивість притаманна ПО і даному об'єктові, але невідомо, як вона на об'єкті проявляється – невизначеність на рівні значень атрибутів.

Найбільші можливості для подання неповноти даних мають неоднорідні бази даних, коли невизначеність вводиться на рівні значень у відношеннях. Такий підхід дає можливість відображати у базах такі випадки неповноти даних:

- значення знаходяться в інтервалі або є одним із дискретної множини значень, зокрема сюди належить невідоме значення;
- значення не існує;
- є неповна чи часткова інформація про значення, яка подається за допомогою нечіткого поняття.

Класифікацію підходів подання неповноти можна зобразити так:

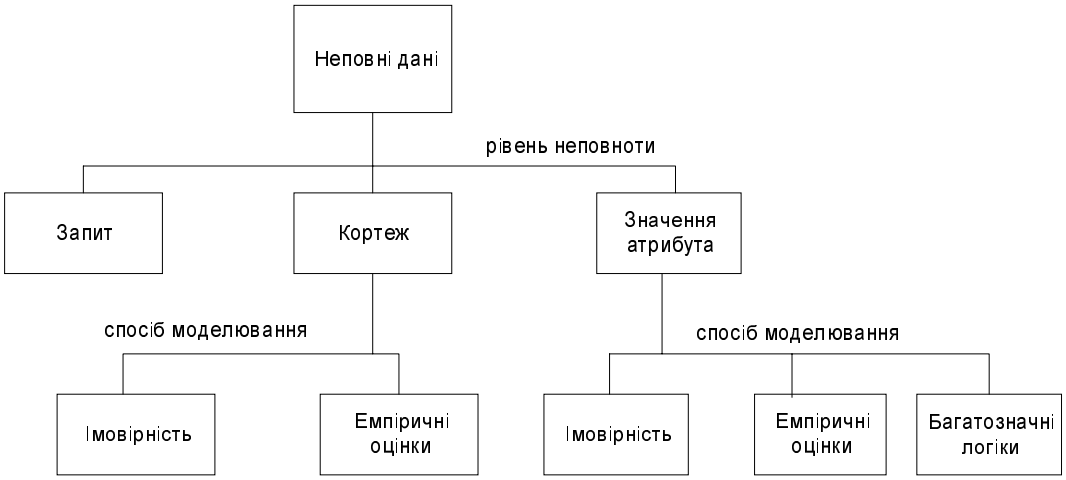


Рис. 1. *Класифікація підходів подання неповноти у базах даних*

Найменше досліджень у моделюванні нечіткостей у базах даних та знань проведено в області використання багатозначної логіки (MVL). Хоча сама MVL бурхливо розвивається ще з 50-х років і має величезні можливості опрацьовування неповних або неточних даних, на сьогодні в існуючих інформаційних системах використано лише частково деякі основні досягнення. Максимальна кількість ступенів істинності становить 4.

1. Системи багатозначної логіки та їх аплікації

Для моделювання невизначеної та нечіткої інформації у базах даних та знань використовують такі системи багатозначних логік [1, 2, 10 - 16]:

1) *скінченнозначна логіка Лукасевича* з одиницею як єдиним визначеним ступенем істинності [22] – уведена у стандарт ANSI/ISO X3.135-1992 і реалізована у всіх сучасних СУБД:

$$u \ \& \ v = \max\{0,\ u + v - 1\}, \qquad (1)$$

$$u \wedge v = \min\{u,\ v\},$$

$$\neg u = 1 - u,$$

$$u \rightarrow v = \min\{1,\ 1 - u + v\};$$

2) *скінченнозначна логіка Ґьодела* з одиницею як єдиним визначеним ступенем істинності:

$$u \wedge v = \min(u, v), \qquad (2)$$

$$u \vee v = \max(u, v)$$

$$u \rightarrow v = \begin{cases} 1, \text{if } u \le v \\ v, \text{ if } u > v \end{cases},$$

$$\neg u = \begin{cases} 1, \text{if } u = 0 \\ 0, \text{if } u \ne 0 \end{cases}.$$

Використовується у дедуктивних базах даних, зокрема, для визначення важливості правил у методі дослідження джерела інформації (IST), розробленого для моделювання і маніпуляції непевними й неточними даними у відносних базах даних [10, 11, 25];

3) *T-norm зв'язані системи*

$$u \wedge v = \min\{u, v\},$$
$$u \vee v = \max\{u, v\}. \quad (3)$$

Частковий випадки такої t-norm пов'язаних систем – нескінченнозначні системи Лукасевича і Ґьодела і також продукційна логіка, що, звичайно, використовує результат алгебраїчних обчислень як базисну t-норму.

Один із прикладів застосування – пошук та побудова гранул знань у базах даних та знань [16];

4) *3-значні системи* – найпростіші випадки, що пропонують інтуїтивні інтерпретації ступенів істинності; ці системи включають тільки один додатковий ступінь крім класичних значень істинності.

У сучасних СУБД найчастіше використовується логіка Лукасевича з 3-ма ступенями істинності. Дослідження у цій області проводилися також Кліні, Постом та Блау. Основна відмінність у цих підходах полягає у інтерпретації ступенів істинності.

Таблиця. 1

Таблиці функцій істинності для деяких зв'язок у 3-значній системі

$\wedge_+$	0	½	1
0	0	½	0
½	½	½	½
1	0	½	1

$\rightarrow_+$	0	½	1
0	1	1	1
½	½	½	½
1	0	½	1

Тут ½ позначає третій "невизначений" ступінь істинності.

5) *4-значна система Дуна-Белнапа* [9, 10] є результатом дослідження у логіці доречності (relevance logic). Множина ступенів істинності визначена як

$$W^* = \{Ø, \{\bot\}, \{T\}, \{\bot, T\},$$

і ступені істинності, інтерпретовані як індикація (наприклад, у запитах бази даних) випадків:

- Ø – нема інформації,
- {⊥} – помилкова інформація,
- {T} – є інформація,
- {⊥, T} – суперечлива інформація, яка говорить як про наявність інформація, так і про її помилковість.

Ця множина ступенів істинності має два природні (гратчастих) упорядкування:

- упорядкування за значенням істинності {T} у вершині непорівнянних ступенів Ø , {⊥ , T}, і {⊥} в основі;
- інформація (чи знання), упорядкована так, що {⊥ , T} є у вершині непорівнянних ступенів {⊥ }, {T}, і Ø – в основі.

Заперечення визначене функцією ступеня істини, що заміняє ступені {⊥} і {T} і залишає ступені {⊥ , T} з установленим Ø.

Дана система частково використовується для гранулювання відношень [12];

6) *продукційні системи [9]*

Загальна проблема виявлення інтуїтивного розуміння ступенів істинності іноді має гарне рішення: можна розглядати їх як включення різних видів оцінки речень. У такому

випадку *k* різні аспекти ступенів істинності можуть бути обрані як *k*-кортежі значень, що оцінюють одиничні аспекти. Вони можуть позначатись стандартними значеннями істинності.

Функції ступеня істинності по таких *k*-кортежах додатково можуть бути визначені "покомпонентно" від функції ступеня істинності (чи значення істинності) для значень одиничних компонентів. У такому випадку, *k*- логічні системи можуть бути об'єднані в одну багатозначну продуктивну систему.

Отже, ступені істинності 4-значної системи Дунна-Белнапа можуть розглядатися як оцінка двох видів даних у базах даних [9]:

1) присутня чи відсутня достовірна інформація;

2) присутня чи відсутня недостовірна інформація.

Обидва види можуть використовувати стандартні значення істинності для цієї оцінки. У цьому випадку, кон'юнкція, диз'юнкція, і заперечення 4-значної системи Дунна-Белнапа покомпонентно визначається кон'юнкцією, диз'юнкцією чи запереченням, відповідно, класичної логіки, тобто 4-значна система – продукт двох копій класичної двозначної логіки.

Продукційні системи використовуються у ряді підходів до проектування сучасних баз знань, зокрема, у згаданому вже IST.

Розглянемо застосування систем MVL у сучасних СУБД.

2. MVL у сучасних СУБД

2.1. NULL-маркери

Для реляційних СУБД Ф. Е. Кодом [3] був запропоновано підхід, у якому для представлення відсутньої інформації використовується спеціальний NULL-маркер (застосування тризначної логіки). Усі предикати, за винятком EXISTS, NULL, UNIQUE, MATCH, в реляційній алгебрі можуть набувати значення TRUE, FALSE, UNKNOWN.

Таблиця 2

Таблиця істинності для реляційної алгебри

Вираз	**Значення істинності**		
NOT true	false		
NOT false	true		
NOT unknown	unknown		
OR	TRUE	FALSE	UNKNOWN
TRUE	true	true	true
FALSE	true	false	unknown
UNKNOWN	true	unknown	unknown
AND	TRUE	FALSE	UNKNOWN
TRUE	true	false	unknown
FALSE	false	false	false
UNKNOWN	unknown	false	unknown
IS	TRUE	FALSE	UNKNOWN
TRUE	true	false	false
FALSE	false	true	false
UNKNOWN	false	false	true

NULL-маркер введений у стандарт SQL92.

У сучасних СУБД прийняті деякі правила декларації атрибутів, пов'язані з NULL-маркерами:

- за замовчуванням NULL припустимі, заборона невизначених значень NULL для даного атрибута вказується декларацією **NOT NULL**;
- якщо явно не задане значення за замовчуванням для даного атрибута (відсутня декларація **DEFAULT**), то це еквівалентно декларації **DEFAULT NULL**;
- декларація **NOT NULL** без явної декларації **DEFAULT**, відмінного від NULL, потенційно суперечлива – спроба змінити значення атрибута на значення за замовчуванням буде породжувати виключення "NULL не припустимі для даного атрибута". У випадку використання при проектуванні бази даних декларацій атрибутів такого типу варто відслідковувати випадки і виключати, коли значення атрибута не вказується явно, тому що у всіх таких випадках сервер поверне помилку. Це означає, що такі пропозиції будуть просто створювати наперед відоме "сміття" при обміні застосування клієнта й сервера, а виконані ніколи не будуть.

При проектуванні бази даних завжди доводиться враховувати можливість появи NULL у вибірці.

У той час як у логічних операціях невідоме значення не дорівнює навіть саме собі, при виконанні реляційних операцій з видаленням дублікатів невідомі значення вважаються рівними один одному.

З поняттям NULL нерозривно зв'язане і введення таких операцій, як зовнішні з'єднання (outer joins). На практиці зовнішні з'єднання виявилися корисні при виконанні звітів і в аналітичних запитах. У сучасних візуальних інструментах по побудові звітів простіше мати справу із завжди присутніми полями, що позначаються NULL-маркерами при відсутності даних, ніж із полями таблиць, асоційованими з даною таблицею, рядки якої відсутні при відсутності даних.

Слід зазначити, що в більшості реалізацій NULL для числових атрибутів зберігається, як правило, у вигляді 0 із спеціальним маркером NULL, а для рядків – у вигляді порожнього рядка зі спеціальним маркером NULL. Так, значення атрибутів зберігаються як у файлах даних, так і у файлах індексів. Це означає, що NULL не дорівнює ні 0 для числових атрибутів, ні “” для рядків. Таке виконання операції IS регламентовано стандартом ANSI X3.135-1992. Проте існують реалізації, де це не так. Зокрема, для Oracle версій 7, 8 (а також молодших версій) для стрічкових атрибутів змінної довжини “” і NULL еквівалентні. Це також стосується типів char, varchar, varchar2, raw, long raw, long, date, rowid, blob, clob, MLSLABEL.

У своїх нових роботах Код запропонував використовувати не один, а два маркери – маркер незастосовності й маркер невідомості, для відображення концептуальної різниці цих понять. Відповідно їм пропонується ввести чотиризначну логіку.

У стандарті ANSI/ISO 92 SQL дана ідея знайшла відображення у введені поняття NULL, логічного значення unknown, і відповідних операцій. Зокрема, перевірку на наявність і відсутність маркера NULL можна провести за допомогою операцій operand IS NULL і operand IS NOT NULL. Пізніші ідеї Ф. Е. Кода в стандарті ANSI/ISO SQL відображення не знайшли [27].

Похідні відношення (у SQL відомі як Views – представлення), на відміну від базових відношень (у SQL відомі як Tables – таблиці), можуть мати кортежі з первинним ключем NULL.

При декларації зовнішніх ключів так само припустимі NULL. Відповідно розширені правила цілісності зв’язків між таблицями. Якщо при визначенні зовнішнього ключа допускаються NULL (хоча б для одного атрибута, що входить у ключ, відсутня декларація

NOT NULL), то такий ключ повинний або посилатися на реально існуючий запис у цільовій таблиці, або бути NULL, що означає, що дане посилання відсутнє [17].

2.2. Агреговані дані

Поряд з уведенням у стандарт SQL поняття NULL була розширена функціональність всіх агрегатних операцій

Єдина агрегатна функція, що завжди враховує NULL – це COUNT(*) [27, 32]. NULL для цієї агрегатної функції сприймається як спеціальне значення і ніколи не ігнорується. Для інших агрегатних функцій результатом буде наступне:

Таблиця 3

Результати виконання агрегатної функції для атрибутів з NULL

Дані в таблиці для атрибута A	**Результат виконання агрегатної функції**
порожня таблиця	NULL
тільки NULL	NULL
NULL і визначені значення	NULL ігноруються

Нехай таблиця T визначена так.

```
create table t (i int);
insert into t values (1);
insert into t values (NULL);
```

Тоді в результаті виконання запитів одержимо

```
select count(*) from t;
|2
select count(i) from t;
|1
select max(i) from t;
|1
select min(i) from t;
|1
select avg(i) from t;
|1
```

Такий результат регламентується стандартом ANSI X3.135-1992.

Опишемо приклад роботи з агрегованим полем Out:

```
CREATE DOMAIN Identifier INTEGER NOT NULL;
CREATE TABLE Enterprise
(
   EnterpriseId Identifier,
  Out NUMERIC(12, 2) NULL CHECK ((VALUE >= 0.0) OR (VALUE IS NULL)),
   PRIMARY KEY(EnterpriseId)
);
UPDATE Enterprise
SET Enterprise.Out = 60000.00
WHERE Enterprise.EnterpriseId = :AnEnterprise;
UPDATE Enterprise
SET Enterprise.Out = NULL
WHERE Enterprise.EnterpriseId = :AnotherEnterprise;
UPDATE Enterprise
SET Enterprise Out = 0.00
WHERE Enterprise.EnterpriseId = :YetAnotherEnterprise;
SELECT Avg(EnterpriseOut) FROM Enterprise; [28]
```

Остання вибірка поверне середнє значення прибутку для двох підприємств, яка дорівнює 30000.00 (три кортежі, але один не враховується (розрахунок агрегатних функцій за стандартом ANSI X3.135-1992), тому що NULL для атрибута Out у даній предметній області означає, що підприємство безприбуткове).

2.3. Спеціальні значення

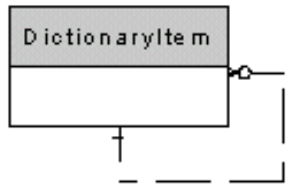
Є також деякі загальні методики при роботі з невизначеною інформацією, що дозволяють не використовувати NULL. Фактично здійснюється підміна NULL деяким спеціальним значенням [30].

Перша з цих методик полягає у використанні атрибутів із визначеним можливим значенням, що означає відсутність інформації. Часто це спеціальне значення також є і значенням за замовчуванням, підтримка якого (DEFAULT) також уведена в стандарт SQL. Зовнішні ключі можуть бути оголошені на наборі атрибутів, для яких визначені значення за замовчуванням. Зовнішній ключ завжди повинний посилатися на реально існуючий запис у цільовій таблиці, нехай навіть він і має значення за замовчуванням. Розширено правила декларативної цілісності зв'язків між таблицями, що дають змогу для таких випадків, як втрата запису власника, встановлювати в якості цільового значення за замовчуванням (правило SET DEFAULT).

Найчастіше ця методика використовується для символьних атрибутів, де, наприклад, рядок нульової довжини чи рядок, забитий пробілами, означає відсутність інформації. Для атрибутів із позитивними значеннями таким визначеним значенням може стати будь-яка негативна величина. Цю методику рекомендується застосовувати тільки для незліченних атрибутів (значення яких не використовуються при виконанні різних агрегатних операцій типу підсумовування), для випадків поки не відомих значень. Слід зазначити відхилення від стандарту деяких СУБД, зокрема, дану методику не рекомендується застосовувати для Oracle 7, 8 (і молодших версій), тому що для типів char, varchar, varchar2, raw, long raw, long, date, rowid, blob, clob, MLSLABEL значення "" і NULL еквівалентні. У документації Oracle для версій 8 не рекомендується користатися даною особливістю Oracle, тому що в наступних версіях планується привести обробку NULL у повну відповідність стандарту ANSI X3.135-1992. [27]

На жаль, при операціях над такими атрибутами також явно доводиться враховувати факт можливої появи спеціального значення у вибірці. До того ж у мові SQL немає однозначного декларативного способу опису поведінки таких атрибутів (є декларативний спосіб опису значень за замовчуванням, але останні можуть використовуватися не тільки як спеціальні значення), що приводить до того, що база даних погано документує свою структуру. Доводиться вчитуватися в супутню документацію і вихідні тексти тригерів і вибірок.

Деревоподібна структура з використанням спеціальних значень подається так [28]




```
CREATE DOMAIN Identifier INTEGER NOT NULL;
CREATE TABLE DictionaryItem
(
   DictItemId Identifier, /*ПК*/
   ParentDictItemId Identifier, /* власник */
   Name VARCHAR( 80) NOT NULL, /* інші дані */
   PRIMARY KEY( DictItemId ),
   FOREIGN KEY( ParentDictItemId ) REFERENCES DictionaryItem );
```

Кореневі елементи довідника заповнюють поле "власник" значенням свого первинного ключа (створюючи "петлі"):

```
CREATE VIEW RootDictionaryItem( DictItemId, Name ) AS
SELECT DI.DictItemId, DI.Name FROM DictionaryItem DI WHERE DI.ParentDictItemId = DI.DictItemId;
```

Вибірка елементів, підлеглих даному елементові (заданому параметрами OwnerId1 і OwnerId2), у загальному випадку проводиться так:

```
SELECT
  DI.DictItemId, DI.Name
FROM
  DictionaryItem DI
WHERE
  (DI.ParentDictItemId = :OwnerId1) AND (DI.DictItemId <> :OwnerId2);
```

Наведемо дещо змінений приклад. Як бачимо, виконувати агрегатні операції над даними, представленими з використанням спеціальних значень, стає важко.

```
CREATE DOMAIN Identifier INTEGER NOT NULL;
CREATE TABLE Enterprise
(
   EnterpriseId Identifier,
   Income NUMERIC(12, 2) NOT NULL DEFAULT -1.0 CHECK ((VALUE = -1.0) OR (VALUE >= 0.0)),
   PRIMARY KEY(EnterpriseId)
);
UPDATE Enterprise
SET Enterprise.Income = 120000.00
WHERE Enterprise.EnterpriseId = :AnEnterprise;
UPDATE Enterprise
SET Enterprise.Income = -1.0
WHERE Enterprise.EnterpriseId = :AnotherEnterprise;
UPDATE Enterprise
SET Enterprise.Income = 0.00
WHERE Enterprise.EnterpriseId = :YetAnotherEnterprise;
SELECT Avg(Enterprise.Income) FROM Enterprise WHERE (Enterprise.Income <> -1.0); [28]
```

2.4. Розширення реляційної алгебри

У статті [3] і роботі [29] визначаються наступні додаткові операції для роботи з невизначеностями у реляційній алгебрі: MAYBE O-JOIN, MAYBE DIVIDE, OUTER O-JOIN, NATURAL O-JOIN, OUTER NATURAL JOIN і OUTER UNION. Використовується тризначна система Лукасевича.

2.5. Побудова логічних виразів, які відображають обмеження на залежності даних

У [30] дається означення К-значної залежності та К-значної функції як її логічного еквівалента. Для отримання логічного виразу, який описує обмеження на дані функціональною залежністю, використовується апарат К-значної логіки із системою Ґьодела елементарних функцій.

Для заданого інформаційного відношення з невизначеними значеннями розглядається К-матриця всіх можливих наборів, які з погляду підтримки заданої функціональної залежності поділяються на заперечувальні та істинні [30]. На основі цих наборів будується результуючий логічний вираз. По К-матриці можна зробити висновок про можливе розкриття невизначених значень на основі отримуваних знань про обмеження, що

накладаються на дані в результаті порівняння кортежів за умови підтримки конкретної функціональної залежності.

Практична цінність К-значних еквівалентів системи обмежень цілісності, заданих у відношенні, полягає в тому, що вони дають змогу повністю абстрагуватись від семантики вказаних обмежень.

2.6. Розширення домена

У серії статей "Faults and Defaults" [4 - 8] Дейт запропонував схему "спеціальних значень" UNK, яка не підтримується існуючими СУБД і потребує відмови від використання MVL у SQL і впровадження в мову підтримки альтернативного підходу. Дейт вважає, що по можливості варто усунути у своїх базах даних атрибути, що допускають невизначені значення, а для їх опрацювання він пропонує розщепити тип сутності з подібним атрибутом на два підтипи. У першому підтипі атрибут не зможе містити невизначені значення, а в другому не буде самого цього атрибута [20].

Як вважає Дейт у [8], єдиним варіантом повного вирішення цієї проблеми на основі багатозначної логіки є введення в мову трьох спеціальних значень із різним змістом (наприклад, NULL, NULL1 і NULL2), тобто використання не тризначної, а п'ятизначної логіки. Якби в майбутньому виявилося, що не враховано ще один спеціальний випадок неповноти інформації, то набір спеціальних значень і число логічних значень розширювалися б відповідним чином. З огляду на те, що в користувачів часто виникають утруднення при роботі навіть і з тризначною логікою, можна з достатньою впевненістю заявити про практичну незастосовність такого підходу.

Висновки

1. На сьогодні існує мало застосувань багатозначної логіки у базах даних.

2. Найбільша кількість ступенів істинності – 4 – неявно реалізована у сучасних стандартах SQL.

3. За допомогою апарату К-значних логік можна не тільки подавати у базах даних неточні дані з різними ступенями істинності, але й аналізувати та розкривати невизначеності.

4. Водночас з очевидними перевагами застосування апарату К-значних логік до баз даних та знань необхідно відмітити складність сприйняття, представлення та опрацювання навіть невеликого числа ступенів істинності.

1. Any C. Brualdi ERIC/AE. Multiple Intelligences: Gardner's Theory. +ERIC/AE Digest Series EDO-TM-96-01, September 1996, http://www.ericae.net/digests/tm9601.htm 2. Cignoli, R., d'Ottaviano, I. and Mundici, D. (2000): Algebraic Foundations of Many-Valued Reasoning. Kluwer Acad. Publ., Dordrecht. 3. Codd, E.F. The Relational Model for Database Management Version 2. Reading, Mass.: Addison-Wesley, 1990. 4. Date, C.J. "Don't Mix Pointers and Relations!" and "Don't Mix Pointers and Relations - Please!". In C.J.Date, Hugh Darwen, and David McGoveran: Relational Database Writings 1994 – 1997. Reading, Mass.: Addison-Wesley, 1998. 5. Date, C.J. "Don't Mix Pointers and Relations!" and "Don't Mix Pointers and Relations - Please!". In C.J.Date, Hugh Darwen, and David McGoveran: Relational Database Writings 1994 – 1997. Reading, Mass.: Addison-Wesley, 1998. 6. Date, C.J. "The Extended Relational Model RM/T." In C.J.Date, Relational Database Writings 1991 – 1994. Reading, Mass.: Addison-Wesley, 1995. 7. Date, C.J. and Hugh Darwen. A Guide to the SQL Standard (4th edition). Reading, Mass.: Addison-Wesley, 1997. 8. Date, C.J. and Hugh Darwen. Foundation for Object/Relational Databases: The Third Manifesto. Reading, Mass.: Addison-Wesley, 1998. 9. Date, C.J. and Hugh Darwen. Foundation for Object/Relational Databases: The Third Manifesto. Reading, Mass.: Addison-Wesley, 1998. 10. Hähnle, R. (1993): Automated Deduction in Multiple-Valued Logics. Clarendon Press, Oxford. 11. Hähnle, R. (1999) Deduction in Many-Valued Logics: Surway,*

http://i12www.ira.uk.de 12. Hähnle, R. (1999): Tableaux for many-valued logics, in: M. d'Agostino et al. (eds.) Handbook of Tableau Methods. Kluwer, Dordrecht, 529-580. 13. K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient implementation of a BDD package. In Proc. 27-th ACM/IEEE Design Automation Conference. P. 40 – 45, IEEE Press, Los Alamitis, 1998 14. L. Godo, W. van der Hoek, J.J. Ch. Meyer and C. Sierra (1995); Many-valued Epistemic States. Application to a Reflective Architecture: Milord-II. Lecture Notes in Computer Science. Springer-Verlag. # 945. P. 440 – 452. 15. M. Takahashi. Many-valued logics of extended Gentzen style I. Science Reports of the Tokyo Kyoiku Daigaku. – Section A.. – 9(231). – 1997 16. Panti, G. (1998): Multi-valued logics, in: D. Gabbay, P. Smets (eds.) Handbook of Defeasible Reasoning and Uncertainty Management Systems. Vol. 1: P. Smets (ed.) Quantified Representation of Uncertainty and Imprecision. Kluwer Acad. Publ., Dordrecht, 25 – 74 . 17. Pin-Shan Chen. :The Entity-Relationship Model - Toward a Unified View of Data". ACM Transactions on Database Systems 1(1), March 1976. Republished in Michhael Stonebraker (ed.): Readings in Database Systems (2nd edition), San Mateo, Calif.: Morgan Kaufmann, 1994. 18. R. Barbuti and Paolo Mancarella. A Multiple-Valued Logical Semantics for Prolog, http://www.di.unipi.it/~paolo/abstracts/esop96.abstract.html 19. Stanford Encyclopedia of Philosophy, http://plato.stanford.edu/entries/logic-manyvalued/ 20. The Fault with Defaults Tom Johnson, Database Programming & Design On-Line. February 1998. Vol.11. N 2, , http://www.dbpd.com/9802xtra.htm 21. Höhle U.R.., Hähnle S.E., (2000): Advanced many-valued logics, in: D.M. Gabbay (ed.) Handbook of Philosophical Logic. 2nd ed. (to appear). 22. Vienna Group for Multiple Valued Logic. Multlog 1.0: Towards an expert system for many-valued logics. In M. McRobbie and J. Slaney, editors, Proc. 13th Conference on Automated Deduction, New Brunswick/NJ, USA, 1996 23. Wojcicki, R. and Malinowski, G. (eds.) (1977): Selected Papers on Lukasiewicz Sentential Calculi. Ossolineum, Wroclaw. 24. Карпенко А.С. Логика на пороге нового тысячелетия, 2001, http://kiev.philosophy.ru/library/logic/karpenko/01.html 25. Автоматическое построение программ по знаниям, Компьютерная неделя N15 (89) от 22/4/1997 Москва, http://ns.edison.ru/magazins/www.pcweek.ru/97_15/win/re2.htm 26. Батыршин И.З. Методы представления и обработки нечеткой информации в интеллектуальных системах. // Новости искусственного интеллекта. –1996, 2, 9 – 65. 27. Гетманова А. Взаимосвязь информатики с классической и неклассическими логиками. М.: 1998, http://bitpro.aha.ru/ITO/ITO98/1/GETMAN.html 28. Образцы проектирования баз данных. Представление отсутствующей информации. А. Абдулин, Л. Козленко. 2000, версия 1.30, http://akzhan.midi.ru/devcorner/articles/DDP-Representation-of-an-absent-information-rus.html#Note1 29. Пасичник В.В., Тавпаш Ю. А. Методические указания по изучению отдельных глав курса "Базы и банки данных" по теме "К-значная логика". – Львов: ЛПИ, 1991 30. Пасичник В.В., Малюта Т. А. Методические указания по изучению отдельных глав курса "Базы и банки данны". – Львов: ЛПИ, 1990. – С. 23 – 40 31. Кузнецов С. Д. Неопределенная информация и трехзначная логика // СУБД #05-06/97, http://www.osp.ru./dbms/1997/05_06/65.html 32. Труды международного семинара "Мягкие вычисления - 96"/ Под ред. И.З. Батыршина, Д.А. Поспелова. – Казань, 1996. – 222 с. 33. Черноруцкий И. Г., Смолко Д. С. Система поддержки принятия решения с нечеткой базой данных для портфеля ценных бумаг. СПБ, 1999, http://inftech.webservis.ru/it/conference/scm/1999/session11/smolko.html