

## Data fusion survey阅读报告

第一部分：可以提升的方向：

其实整篇论文的核心要点就是阐述数据融合的步骤，介绍数据融合的技术以及对介绍数据融合的评估方式。（最后也介绍了不同的信息系统，但那个只是展示当时技术的应用情况）

通过阅读我们也不难明确：各种连接、聚合、分组、vote、“信任你的朋友”、用户自定义函数都在某一些信息系统进行了应用，并且这些方法的应用场景各不相同，那么：可以利用机器学习分别对这些数据融合方法进行学习，最后通过一个权重投票来判断每一次的数据冲突以及数据融合的处理方法。

或者建立概率统计模型，通过计算概率，设定阈值的方式判定不同修复的可能性并给出综合修复后的元组

第二部分：文章具体内容

### 摘要：

---

核心：基于数据整合（data integration）的大背景下的异构信息系统的融合（data fusion）研究

数据融合的定义：数据融合是将代表同一现实世界对象的多条记录融合成一个单一的、一致的、干净的表述的过程

数据融合的挑战：不确定和冲突值

文章介绍了多种数据融合技术，并介绍了他们在不同领域的应用

### 第一节 Introduction：

---

目标：通过一个一致的界面访问所有这些信息源

效果：完整是因为没有对象被遗忘在结果中；简明是因为没有对象被代表两次，呈现给用户的数据没有冲突（后者比前者更难，因为同一个数据可能来源于多个数据源）

挑战：连接不同机器上的不同数据源&&语义异质性（同一信息的不同表达方式）

解决方式：检查不同来源中的等价模式元素（模式匹配schema matching）和检测不同来源中的等价对象描述（重复检测duplicate detection）----》产生单一的、一致的表示，但是目前缺乏一致性的检测

挑战：冲突值的解决

解决方式：经常被忽略，通常只解决缺失值的不确定性来避免数据冲突

本论文的贡献：

介绍信息整合的大背景下的数据融合过程。介绍并比较了现有的实现这种数据融合步骤的方法

这个过程在文献中也被称为数据合并（data merging）、数据整合（data consolidation）、实体解析（entity resolution）或寻找代表/幸存者（finding representations/ survivors）

论文结构：

第一节：介绍

第二节：简要介绍了信息整合和需要执行的不同任务

第三节：对数据融合的关系技术进行了介绍、描述和分类

第四节：对综合信息系统及其数据融合能力进行了详细概述和分类

## 第二节：Data Fusion

### 2.0数据融合概览

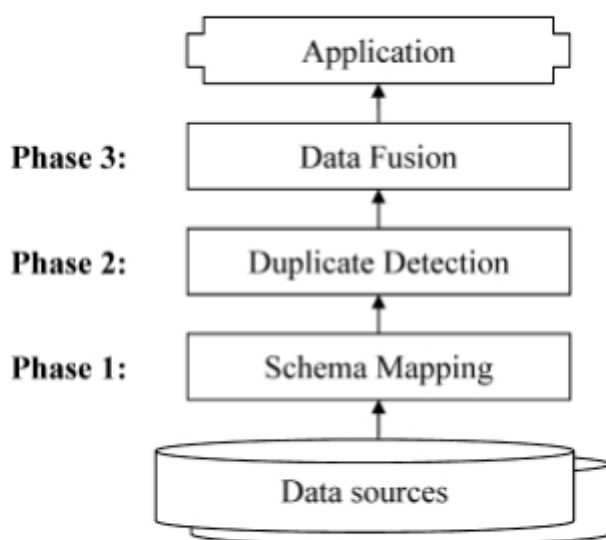


Fig. 1. A data integration process.

集成信息系统：集合多个数据源的信息（查询底层数据源，合并结果）

第一步：模式映射，确定用来描述信息的属性值，将源中存在的数据转换为共同的表示形式。

第二步：识别和对齐不同对象，找到同一事件的不一致表达

第三步：重复表述被融合成单一表述&&解决数据不一致的问题---is called 数据融合（本文聚焦的焦点）

本节内容结构：介绍了前两个步骤的常见解决方案-----》然后在数据融合领域进行了更详细的讨论-----》展示技术和系统之间的差异

### 2.1数据转换

弥合异质性并因此指定数据转换的方法有两种：模式整合（schema integration）和模式映射（schema mapping），可以离线执行

模式整合：整合一组已知的数据源的愿望驱动的，**模式整合考虑的是各个模式，并试图生成一个新的模式**，这个模式对于源模式来说是完整和正确的，是最小的，而且是可以理解的。**换句话说，就是我也不知道能生成出一个什么样的表，但是我会尽可能的避免掉相同语义的属性并且尽可能的包含所有不同语义的属性列。**

模式映射：**假定有一个给定的目标模式**；也就是说，它是由在一个给定的综合信息系统中**包括一组来源的需要**所驱动的。**换句话说，就是要生成什么样的数据模式我是知道的，我就照着那个方向努力就好了**

模式匹配技术（模式映射的补充技术），它半自动地寻找两个模式之间的对应关系

模式集成和模式映射这两种方法的目标是相同的：**转换源的数据，使其符合一个共同的全局模式。**

在数据整合之后，所有某种类型的对象都被同质化地表示。

## 2.2冗余检测

别名：记录链接（record linkage）、对象识别（object identification）、参照物调和等（reference reconciliation）

本阶段的处理目标：**识别同一现实世界对象的多种表现形式：数据融合（第三步）的基本输入。**

一般而言的冗余检测流程：使用**相似度测量法**比较每一对对象，并应用一个阈值。如果一对物体的**相似度高于给定的阈值，则宣布为重复**，一般而言有效性和效率是亟需关注的两个方面

有效性的讨论：相似性度量的质量&&相似性阈值的影响。

相似性度量：利用相似性函数，特定领域的度量&&一般的度量（字符的编辑距离）

相似性阈值：精准度和召回率的平衡，需要针对领域和数据集调整阈值

效率问题的讨论：1、存储所有数据集不可能---解决方法：智能分区，仅在分区内比对--排序邻域法

2、列文斯坦距离&&编辑距离的算法时间复杂度--解决方法：计算相似度上界，只计算上界高于相似度上界的pairs

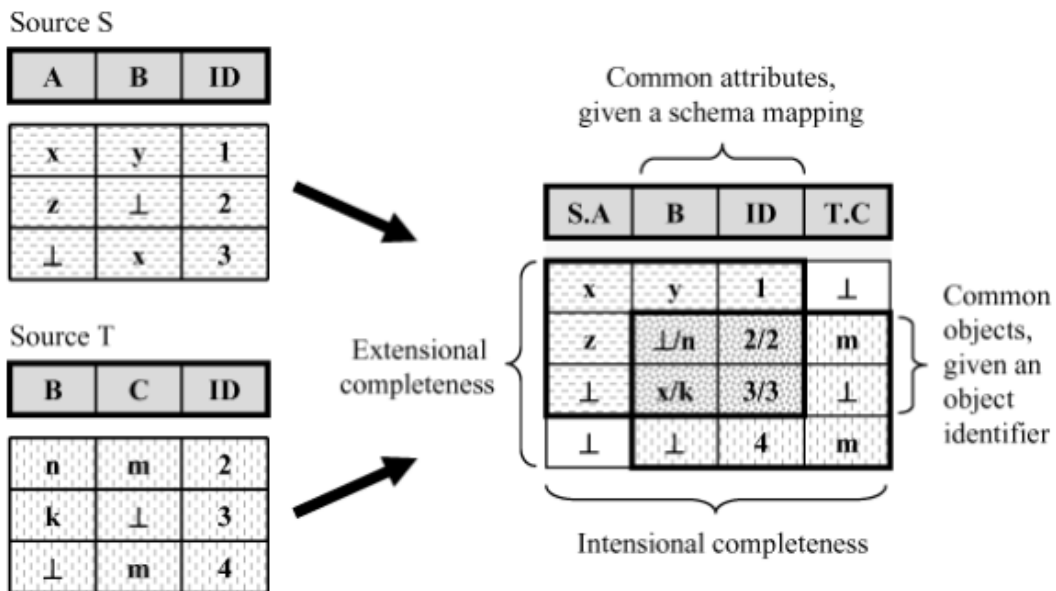
冗余检测的结果：为每一种表示分配一个对象的ID，拥有相同id的表示重复。数据融合就是将一个对象的多种表征融合为一个表征

## 2.3完整而简明的数据整合

完整性的提高：在系统中增加更多的数据源（更多的对象，更多的描述对象的属性）

简明性的提高：删除多余的数据，融合重复的条目和将共同的属性合并成一个

我们可以自定义关于完整性以及简明性的客观公式定义：



**扩展完备性** (Extensional completeness)：一个数据集中唯一的对象表述的数量与现实世界中唯一对象的总体数量的关系，该数据集所涵盖的现实世界物体的百分比。**扩展完备性的提高是通过增加更多的独特对象来实现的**

公式：

$$extensional\ completeness = \frac{||unique\ objects\ in\ dataset||}{||all\ unique\ objects\ in\ universe||} = \frac{a}{a + c}$$

S&T的扩展完备性：3/4

整合后的完备性：4/4

**内在完整性** (Intensional completeness)：在一个数据集中的唯一属性的数量与可用的唯一属性的总数量的关系。**内在完整的提高可以通过整合那些可以提供新属性的关系（也就是数据库中的表）来提高**

S&T的内在完备性：3/4

整合后的内在完备性：4/4

**扩展简洁性** (extensional conciseness)：数据集中唯一对象的数量与数据集中对象表现的总体数量的关系

$$extensional\ conciseness = \frac{||unique\ objects\ in\ dataset||}{||all\ objects\ in\ dataset||} = \frac{a}{a + b}$$

整合后的内在完备性：4/4（四个对象都是不同的）

**内在简洁性** (intensional conciseness)：是一个数据集的独特属性的数量与总体属性数量的关系

整合后的内在完备性：4/4（四个属性都是不同的）

**为了使这些衡量措施起到作用，全局中的对象和所考虑的数据集的定义需要是相同的。**

四种不同程度的数据整合：

S.A	S.B	S.ID	T.ID	T.B	T.C
x	y	1	⊥	⊥	⊥
z	⊥	2	⊥	⊥	⊥
⊥	x	3	⊥	⊥	⊥
⊥	⊥	⊥	2	n	m
⊥	⊥	⊥	3	k	⊥
⊥	⊥	⊥	4	⊥	m

(a) No mapping and no object identifier used

S.A	B	ID	T.C
x	y	1	⊥
z	⊥	2	⊥
⊥	x	3	⊥
⊥	n	2	m
⊥	k	3	⊥
⊥	⊥	4	m

(b) Mapping (S.B↔T.B, S.ID↔T.ID) used, but no object identifier used

S.A	S.B	ID	T.B	T.C
x	y	1	⊥	⊥
z	⊥	2	n	m
⊥	x	3	k	⊥
⊥	⊥	4	⊥	m

(c) Partial mapping (S.ID↔T.ID) used, object identifier (S.ID, T.ID) used

S.A	B	ID	T.C
x	y	1	⊥
z	n	2	m
⊥	x	3	⊥
⊥	⊥	4	m

(d) Full mapping (S.B↔T.B, S.ID↔T.ID) and object identifier (S.ID, T.ID) used

(1) 如果没有模式映射信息和对象标识符的知识，集成系统能做的最好的事情就是产生一个如图4(a)所示的结果。虽然这个结果**有很高的完整性，但它并不简明**

(2) 如果能够由模式映射提供关于共同属性的结果，我们就可以得到 (b) 这个形状是对两个源关系（具有相同名称的映射属性）进行**外联合操作**的结果。我们称这样的结果为**内在简明**（intensionally concise）。没有一个真实世界的**属性**被一个以上的**属性**所代表。

(3) 如果我们能够知道依据哪个/哪几个属性可以区分不同的对象，那么我们就可以根据这些属性进行全外连接，我们称这样的结果为**扩展简洁**（extensionally concise）。没有一个现实世界的对象是由一个以上的元组表示的

(4) 识别共同属性（使用模式映射）和共同对象（使用对象标识符）后的结果：每个对象只有一行代表，每个属性只有一列代表。这样的结果（从内涵和外延上看都是简洁的）是数据融合的最终目标

问题：处理两个来源重叠区域的这些数据冲突（id=3时属性B的取值）（由结果中的检查模式标记）是数据融合的一个组成部分，将在接下来的章节中考虑。

## 2.4冲突分类

不同的冲突可以分为三类：

- (1) 有模式上的冲突，例如，不同的属性名称或不同结构的数据源---在前面通过模式映射都方式解决
- (2) 身份冲突，因为数据源中识别现实世界对象的方式可能不同。---通过寻找全局ID进行解决
- (3) 数据冲突--对同一个对象同一属性的值描述不同----仍然没有解决

对数据冲突的分类：

(1) 属性值的不确定性，由信息缺失引起

(2) 冲突，由不同属性值引起

不确定性：是一个非空值和一个或多个空值之间的冲突，这些空值都被用来描述一个对象的相同属性；不确定性比冲突更容易处理；假设控制是未知值。

冲突：两个或多个不同的非空值之间的冲突，这些值都被用来描述同一对象的相同属性。

## 2.5数据融合的策略和答案

1、数据融合策略：利用不同的策略将来源不同的数据融合为一个一致表示。

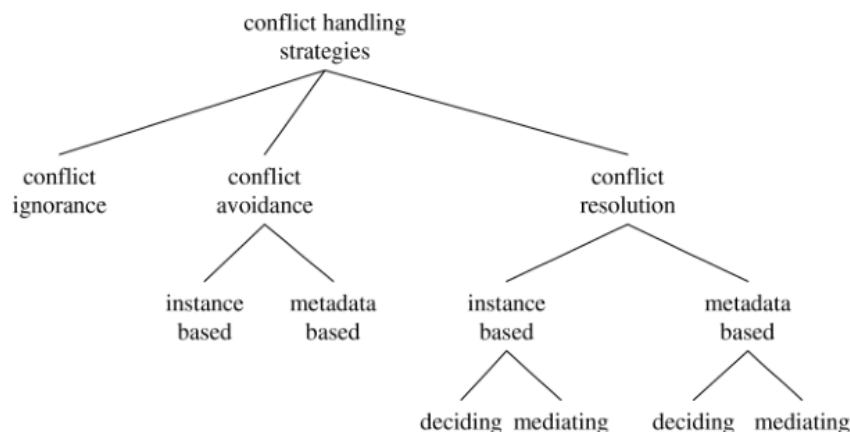


Fig. 5. A classification of strategies to handle inconsistent data [Bleiholder and Naumann 2006].

Table I. Possible Conflict Handling Strategies from Bleiholder and Naumann [2006]

Strategy	Classification	Short Description
PASS IT ON	ignoring	escalates conflicts to user or application
CONSIDER ALL POSSIBILITIES	ignoring	creates all possible value combinations
TAKE THE INFORMATION	avoiding, instance based	prefers values over null values
NO GOSSIPING	avoiding, instance based	returns only consistent tuples
TRUST YOUR FRIENDS	avoiding, metadata based	takes the value of a preferred source
CRY WITH THE WOLVES	resolution, instance based, deciding	takes the most often occurring value
ROLL THE DICE	resolution, instance based, deciding	takes a random value
MEET IN THE MIDDLE	resolution, instance based, mediating	takes an average value
KEEP UP TO DATE	resolution, metadata based, deciding	takes the most recent value

上面的两个表阐释了不同得数据融合策略：

**忽略冲突**的策略并不决定如何处理冲突的数据，有时甚至不知道数据冲突。他把冲突解决的工作延迟到了用户身上。

**避免冲突**的策略承认一般情况下可能存在的冲突，但不检测和解决单一的现有冲突。

常用思想：更偏爱某一个数据源的数据；用确定值代替空值；只返回一致的元组

**冲突解决策略**在决定如何解决冲突之前确实考虑了所有的数据和元数据

常用思想：决定性策略（从所有可能值中选一个）；调和性策略：（所选的值可能不在候选值中，例如取平均值）

2、数据融合答案：

对综合信息系统的查询结果就是数据融合答案。

答案的分类：

**完整的答案：**包含了所有的对象（外延完整）和所有的属性（内涵完整），这些都是来源中出现的。完整的答案不一定简洁

**简洁的答案** 现实中的对象（扩展简洁性）和语义等价的属性（内在简洁性）都只被描述一次

**一致的答案：**包含了所有满足完整性约束的元组，一致性答案不一定是完整的。一个一致性答案对于包含的对象来讲都是扩展简洁的（完整性约束有对key的约束）

**完整的答案和一致的答案：**包含所有现实中对象及其描述属性（不重复）的一致答案；这也是数据融合的目标

## 2.6激励性的例子

**Table II.** Example Data Source, Representing Students from the First University ( $U_1$ ) Showing Extensional Completeness of 6/7 and Extensional Conciseness of 6/7 With Regard to the Considered Universe, all Seven Students in the Two Example Tables

Name	Age	Status	Address	Field	Library
Peter	⊥	0	TUB	Computer Science	P201
Alice	22	1	Berlin	Mechanical Engineering	A709
Bob	⊥	1	Boston	⊥	B321
Charly	25	1	⊥	Psychology	⊥
Paul	⊥	1	TUB	Architecture	⊥
Paul	26	1	Berlin	Arch.	P233
Frank	23	0	TUB	Mech. Engineering	F205

Data from data source  $U_1$ .

**Table III.** Example Data Source, Representing Students from a Second University ( $U_2$ ), Showing Extensional Completeness of 5/7 and Extensional Conciseness of 5/7 With Regard to the Considered Universe, All Seven Students in the Two Example Tables

Name	Age	Status	Address	Field	Phone
Alice	⊥	0	⊥	ME	030/12345
Bob	27	⊥	HUB	CS	030/54321
Charly	25	1	⊥	⊥	⊥
Alice	21	1	HUB	Mech. Eng.	030/98765
Eve	24	1	Berlin	CS/EE	030/55544
Eve	24	1	Berlin	CS/EE	030/55544
Frank	23	0	TUB	Mech. Engineering	⊥

Data from data source  $U_2$ .

上图所显示的就是在本论文中使用的例子。

内容解读：

- 1、大学提供的在校学生的数据。
- 2、由于这些大学都位于同一个城市，因此有可能在多所大学学习。
- 3、有可能将几个学习领域结合起来。
- 4、有些领域在不止一所大学提供，所以在主题以及学生方面有一定的重叠。
- 5、我们在例子中假设模式已经匹配，并且语义上相等的属性在例子中具有相同的名称。
- 6、所有的学生都可以通过他们的名字进行全局唯一的识别（真实世界的标识符）
- 7、真实世界的标识符不应该与数据库的主键相混淆
- 8、在本例中没有假设数据库主键

## 第三节 数据融合的关系运算符和技术

### 3.0简介

本节介绍了标准和高级关系运算符，并研究了它们在融合不同数据源的数据方面的能力。

标准的运算符：联合（union）和连接（join）

连接的技术：将几个表的元组结合起来同时对一些列进行谓词评估

结合的技术：先建立一个共同模式，再加入来自于源表的不同元组

更高级的运算符：结合标准运算符，发明更高级的运算符

约束：只对两个表进行运算符的操作

### 3.1操作符和技术的属性和特点（operators运算符）

我们使用以下一个或多个特征来描述以下操作符和技术。

#### 1、数据保存（value preservation）

当为了提高扩展完整性而组合表（combine tables）时，不一定所有的源值都包括在结果中，比如说两个属性描述的同一个事情。（下图为结合表的例子，注意区分结合表和数据库查询时做表连接时的区别）

S.A	S.B	S.ID	T.ID	T.B	T.C
x	y	1	⊥	⊥	⊥
z	⊥	2	⊥	⊥	⊥
⊥	x	3	⊥	⊥	⊥
⊥	⊥	⊥	2	n	m
⊥	⊥	⊥	3	k	⊥
⊥	⊥	⊥	4	⊥	m

(a) No mapping and no object identifier used

然而，为了使运算符完全达到数据保存的目的，所有描述对象的所有属性的所有值都需要保留在结果中。

我们表示一个不会丢失任何值或创建或产生重复值的运算符为可以达到数据保存的运算符

一个值的唯一存在是不够的，我们允许重复的值（详见下面关于bag union的定义）

因此，bag union是保值算子的例子，set union和Cartesian product是不保值算子的例子。



参考网址: [http://www.turingmachine.org/courses/2007/saved.csc370S07/lectures/04\\_rel-algebra2.pdf](http://www.turingmachine.org/courses/2007/saved.csc370S07/lectures/04_rel-algebra2.pdf)

Cartesian product: 笛卡尔积---会产生大量重复值

$$r \times s = \{t \ q \mid t \in r \text{ and } q \in s\}$$

bag union: 在包上的并集计算, 只是将值拿出来, 并不是产生重复值

## Bag Union

- **Union, intersection, and difference** need new definitions for bags.
- An element appears in the **union** of two bags the **sum** of the number of times it appears in each bag.
- Example:

$$\begin{aligned} &\{1,2,1\} \cup \{1,1,2,3,1\} \\ &= \{1,1,1,1,1,2,2,3\} \end{aligned}$$

set union: set只能存储唯一的对象 (一个对象只能出现一次, 所以会丢失值)

价值保存不应该被误认为是可恢复性的属性 (这里不太懂是什么意思), 也就是说, 我们可以反转操作, 并从结果中推断出来源中的价值 (对象的属性值可能会丢失, 但所有在源中被描述的对象夜莺包括在结果中)

可恢复性的一些可能的参考网页: <https://blog.csdn.net/u010486124/article/details/42426127> 大部分都是事务之间的调度

在价值保存中不考虑扩展完整性

## 2、唯一性

运算符的唯一性保护属性: 在数据源中包含唯一属性的那些值在结果中同样保持唯一属性。举例: 对于属性的等值连接为真, 但对union操作为假 (这里的union可以理解为表的union或理解为对查询结果的union)

一个创建包含唯一属性的结果 (不管源中是否有唯一属性) 的操作被称为具有唯一性强化属性 (uniqueness-enforcing property)

一个操作符需要在持有现实世界标识符的属性上具有唯一性

### 3、对操作符两种属性的总结：

理想的数据融合操作者，创建完整和一致的答案，应该尽可能多地保留数值和对象，同时强制或保留唯一性并解决冲突。

## 3.2连接操作符 (Join Approaches)

### 3.2.0 连接方法的总结

连接方法可以增加内在完整性（增加属性值），但不能保证扩展完整性（除非使用完全外连接full outer join）。

连接在扩展简洁性方面做的很好，但他依赖于全局唯一标识符并且源内不重复

#### 3.2.1标准连接 (stander join)

(equi\_join)

如果连接条件由列之间的相等条件组成

在一些属性上使用等价连接建立一个真实世界的标识符，但在真实场景中有可能不使用全局标识符进行连接

键连接是唯一性的，但不一定是保值/保对象的

(natrual join)

把所用属性名相同的列中所有属性值相同的行连接起来

自然连接具有唯一性，但不具有数据保存性质

(full out join) (left out join) (right out join)

对不满足条件的行的处理区分了内连接和外连接

详细的教程：[带你了解数据库中JOIN的用法 - Coder编程 - 博客园\(cnblogs.com\)](http://cnblogs.com/)

该操作符仅在其基于键连接的完整变体中是保值/对象的。但是它在所有的变体中都是唯一性保存的

没太懂的地方：存在一个基于键连接or左连接or又连接的实体，他们只包含来自（第一or第二个）关系的元组

#### 3.2.2 Full Disjunction.

外连接的缺陷：通过外连接将两个以上的表结合起来可能会产生不同的结果表，这取决于表的连接顺序。

完全分离运算符：两个或多个表的组合，其中所有匹配的元组被合并为一个单一的元组。

完全分离运算符是具有唯一性的，并且是保值/保对象的，就像完全外部连接一样。

google讲议：[https://www.slideshare.net/atul\\_shridhar/full-disjunction-56682](https://www.slideshare.net/atul_shridhar/full-disjunction-56682)

# ?Why and What is Full Disjunctions

The *full-disjunction* operation is a variation of the join operator that *maximally* combines *join consistent* tuples from *connected* relations, while *preserving* all information in the relations.

## An Example of a Full Disjunction

Climates		Accommodations			
Country	Climate	Country	City	Hotel	Stars
Canada	diverse	Canada	Toronto	Plaza	4
UK	temperate	Canada	London	Ramada	3

Sites			$\mathcal{R}$
Country	City	Site	
Canada	London	Air Show	
Canada		Mount Logan	
UK	London	Buckingham	

FD( $\mathcal{R}$ )					
Country	Climate	City	Hotel	Stars	Site
Canada	diverse	Toronto	Plaza	4	
Canada	diverse	London	Ramada	3	Air Show
Canada	diverse				Mount Logan
UK	temperate	London			Buckingham

### 3.2.3 匹配连接 (match join) 和容忍冲突的查询

第一步：来自所有来源的相应属性值被单独投射出来，并通过联合进行组合

第二步：然后，现实世界的标识符被用来重新连接，从而形成一个单一的大表。

参考实现代码：

```

WITH OU AS (
  ( SELECT Name, Age, Status, Address, Field, Library, NULL as Phone
    FROM U1 )
  UNION
  ( SELECT Name, Age, Status, Address, Field, NULL as Library, Phone
    FROM U2 )
),
AGE_V (Name, Age) AS ( SELECT DISTINCT Name, Age FROM OU ),
STATUS_V (Name, Status) AS ( SELECT DISTINCT Name, Status FROM OU ),
...
PHONE_V (Name, Phone) AS ( SELECT DISTINCT Name, Phone FROM OU ),
SELECT Name, Age, Status, Address, Field, Library, Phone
FROM AGE_V FULL OUTER JOIN STATUS_V ... FULL OUTER JOIN PHONE_V
ON AGE_V.Name=STATUS_V.Name AND STATUS_V.Name= ... =PHONE_V.Name
AS MATCH_JOIN_TABLE

```

第三步：在产生的大表的基础上进行选择，根据附带的冲突容忍查询模型，从这个表中选择元组。在选择元组时增加参数传递。这个参数要么是高置信度，要么是随机证据，要么是根本不可能，并决定哪些图元被保留在结果中。

如果它被设置为高置信度，选择谓词需要对每个真实世界标识符的所有图元都为真。

如果它被设置为可能-所有，则选择谓词需要对每个真实世界标识符的至少一个图元为真。

随机证据 (Random-evidence) 对每个真实世界标识符的一个随机元组验证该谓词。

冲突解决：冲突是通过应用解决函数来解决的，比如sum、max、min、any等等

他的查询结果可能包含具有属性值组合的图元，这些属性值组合作为一个整体没有出现在任何一个源中。

### 3.2.4 提高连接结果的简洁性

普通的连接结果不包含对于相同语义属性的合并，生成的表完整但不一定简洁，有可能是矛盾的。

思路：增加额外的操作将属性进行合并。

```

SELECT Name, max(U1.Age, U2.Age), vote(U1.Status, U2.Status),
       U1.Address as Address, concat(U1.Field,U2.Field) Library, Phone
FROM fulldisjunction(U1,U2)

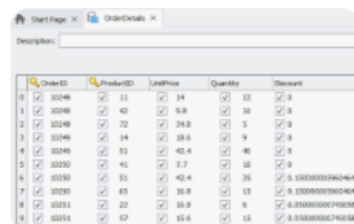
```

实操：使用函数coalesce(v1, v2, ... ,vn)返回v1到vn中第一个不是空值的值vi；可以用来对输入表进行优先排序

使用coalesce将关键列合并为一列（关键列要么包含相同的值要么包含空值）

什么是关键列：

A key column (or a combination of columns) is **used to uniquely identify rows in table and database table elements during comparison**. For this purpose, the key values must be different in each row. Usually, the key columns hold some unique identifiers, such as the employee IDs or SSNs.



	OrderID	ProductID	UnitPrice	Quantity	Discount
1	10248	11	14	12	0
2	10248	42	9.8	36	0
3	10248	72	34.8	3	0
4	10248	14	38.6	9	0
5	10248	51	42.4	40	0
6	10250	41	3.7	34	0
7	10250	51	42.4	35	0.13500000000000004
8	10250	85	35.9	15	0.13500000000000004
9	10251	22	35.9	9	0.13500000000000004
10	10251	57	35.6	15	0.13500000000000004

<https://support.smartbear.com/testing-with/checkpoints>

## Using Key Columns | TestComplete Documentation

[About featured snippets](#) • [Feedback](#)

广义派生操作（GAD）：将full Disjunction的几列合并为一列 GAD可以是用户定义的函数，从相同的语义列中选择一个作为结果

### 3.3 UNION方法

#### 3.3.0 简介

扩展完整性的自然想法（包含来自两个源的元组），但是简洁性很差

#### 3.3.1 Union, Outer Union, and Minimum Union

**联合(U)**运算符(有集合语义)结合了两个联合兼容关系的图元，并删除了完全重复的图元，即所有属性值都相同的图元。不确定性和矛盾很多，忽略了冲突，不保值但是保对象。

联合兼容：属性数量相同，属性数据类型匹配

**外联合**：将两个非联合兼容的表联合起来（填充空值）不保值、但是保对象，不保留唯一性，忽略冲突

**最小联合运算符**：删除子元组的外联合结果，只能进行很小一部分的简洁性改善，还是没有解决冲突

**一个元组t1包含另一个元组t2：**

- (1) 如果它们符合相同的模式；
- (2) 如果t2包含更多的null
- (3) 其他属性对应相等

#### 3.3.2 合并，按优先顺序合并

总结：两个外层连接的联合

```

( SELECT U1.Name, coalesce(U1.Age, U2.Age) AS Age,
    COALESCE(U1.Status, U2.Status) AS Status,
    COALESCE(U1.Address, U2.Address) as Address,
    COALESCE(U1.Field, U2.Field) as Field,
    U1.Library, U2.Phone
FROM U1 LEFT OUTER JOIN U2 ON U1.Name = U2.Name )
UNION
( SELECT U2.Name, COALESCE(U2.Age, U1.Age) AS Age,
    COALESCE(U2.Status, U1.Status) AS Status,
    COALESCE(U2.Address, U1.Address) as Address,
    COALESCE(U2.Field, U1.Field) as Field,
    U1.Library, U2.Phone
FROM U1 RIGHT OUTER JOIN U2 ON U1.Name = U2.Name )

```

优点：可以填补某一个元组在某一张表的空缺，也提供了优先选择权限（U1优先的生成一张表，U2优先的生成一张表，再将两张表UNION）

缺点：不幸的是，这对同一张表中的相应图元（姑且理解为key值相同的表）不起作用，并且也没有解决矛盾

### 3.3.3 提高联合结果的简洁性。

使用分组操作提高外延简洁性（内在简洁性在表连接处理语义相同的属性时就被处理了）

以这种方式提高简洁性需要一个或多个属性作为ID来识别相同的现实世界对象

只使用sql定义的聚合函数是远远不够的，需要用户使用用户自定义的sql支持的函数。在4.3.5会详细说明

```

SELECT Name, max(Age), vote(Status), most_recent(Address), concat(Field),
    coalesce(Library), coalesce(Phone)
FROM union(U1,U2)
GROUP BY Name

```

## 3.4其他技术

### 3.4.0 简介

以下技术既不是基于连接的，也不是基于联合的，许多技术包含了额外的信息，扩展了关系模型或现有的关系运算符，或结合运算符以融合数据。

#### 3.4.1 考虑到所有可能性

第一种处理方式：在元组中增加一列明确表示不确定性（离散的之或者连续的值都可以）。所有的关系代数运算都应考虑额外增加的一列，并且把所有元组连同增加的一列返回给用户。

第二种处理方式：“信任你的朋友”，增加一列信息表明数据是从哪里得到的

第三种处理方式：部分自然外连接：允许一个属性中的多个值以及它们作为正确值的概率，扩展了全分离操作。所有的值以及它是正确的值的概率都被留下了

第四种处理方式：通过概率分布模型模拟不确定性，最终只有一个值反馈给用户。



summary: 所有这些方法都实施了考虑所有可能性的策略, 利用**额外的信息**让用户有意识地在所有可能性中进行选择, 或提出最可能的价值。

### 3.4.2 只考虑一致的可能性

一个简单的想法: 只从数据库中返回一致的信息。(冗余会造成数据不一致, 比如说A的手机号同时存了两个值那就是不一致的)

**Table II.** Example Data Source, Representing Students from the First University ( $U_1$ ) Showing Extensional Completeness of 6/7 and Extensional Conciseness of 6/7 With Regard to the Considered Universe, all Seven Students in the Two Example Tables

Name	Age	Status	Address	Field	Library
Peter	⊥	0	TUB	Computer Science	P201
Alice	22	1	Berlin	Mechanical Engineering	A709
Bob	⊥	1	Boston	⊥	B321
Charly	25	1	⊥	Psychology	⊥
Paul	⊥	1	TUB	Architecture	⊥
Paul	26	1	Berlin	Arch.	P233
Frank	23	0	TUB	Mech. Engineering	F205

Data from data source  $U_1$ .

**Table III.** Example Data Source, Representing Students from a Second University ( $U_2$ ), Showing Extensional Completeness of 5/7 and Extensional Conciseness of 5/7 With Regard to the Considered Universe, All Seven Students in the Two Example Tables

Name	Age	Status	Address	Field	Phone
Alice	⊥	0	⊥	ME	030/12345
Bob	27	⊥	HUB	CS	030/54321
Charly	25	1	⊥	⊥	⊥
Alice	21	1	HUB	Mech. Eng.	030/98765
Eve	24	1	Berlin	CS/EE	030/55544
Eve	24	1	Berlin	CS/EE	030/55544
Frank	23	0	TUB	Mech. Engineering	⊥

Data from data source  $U_2$ .

例如在SELECT \* FROM U2的一致答案: Bob, Charly, 和Frank的元组。不一致的数据通过修复成为一致数据, 例如删除掉Alice或者Eve的其中一行。

在一个查询中所有可能的修复的集合是一个查询的一致答案。与之对应的是一个可能的答案, 他只要出现在一个元组中即可。找寻一致性答案是十分复杂的, 所以现有的策略通常都会为之做出妥协。

## 3.5 总结

常规的思路: 通过联合 (union) 来增加完整性, 因为每一张表存在的的不一致的实体都会在联合中被保留; 通过对相同语义属性的列的合并 (比如自然连接、分组和聚合) 使表示相同语义的列进行合并从而提高简洁性。

但是, 通常会有两列的列名不一样但语义相同 (无法用自然连接提高简洁性) 就需要人对其进行处理。

处理重复的值: 使用用户自定义的函数or聚合连接

具体处理流程的两大思路:

(1) 两个表的outer join, 然后是GAD运算, 语义上相等的列由一个函数组合

(2) 相同表的outer union (映射语义上相等的列), 然后由真实世界的ID分组, 用GAD运算中使用的相同函数聚合剩余的列

outer union的操作：默认情况下，UNION要求两个RowSets都有匹配的模式。OUTER UNION允许模式不同。如果一个RowSet缺少另一个RowSet所拥有的列，那么该行将被包含在结果中，并为缺少的列设置默认值。

数据融合的策略需要依赖于不同领域进行调整

解决空值冲突是最简单的，可以使用消除子元组或者使用COALESCE函数，只有应用函数或者使用分组和聚合才能解决不确定性。

- (1) 只使用SQL提供的功能（SQL99、基本操作和合并），我们可以融合数据，但只能应对不确定性。
- (2) 如果可以使用用户定义的函数，只要不存在源内重复，结合用户自定义函数的连接方法就能进一步解决不一致的问题。
- (3) 使用分组和用户定义的聚合函数，使我们也能处理源内重复和不一致的问题。

## 第四节 数据融合的信息系统

### 4.0简介

在这一节中，将不同的信息系统进行了概述，这些系统整合了来自不同来源的数据。观察它们融合数据和处理冲突的能力，对他们进行分类并逐个讲解其处理数据融合领域工作的能力。

### 4.1信息系统的分类

- (1) 关于数据融合的最先进的系统能够进行冲突解决（第4.3节）。
- (2) 下一类系统承认数据冲突并通过避免冲突来处理它们（第4.4节）。
- (3) 最简单的处理冲突的方式是无视冲突（第4.5节）。
- (4) 考虑所有剩余的系统，在整合不同来源的数据时不处理冲突（第4.6节）。

Table VIII. Main Architecture of Integration Systems and the Data Models Used in the Mediator/Sources (as well as the systems integration model)

System	Architecture	Internal Data Model (Mediator)	Source Data Model	Integration Model
Multibase	MDBMS	Daplex	Daplex	GaV
Hermes	MW	REL/OO	REL, OO, ...	GaV
Fusionplex	MW	REL	REL, OO, ...	GLaV
HumMer	MW	REL	REL, OO, ...	GaV
Ajax	APP	REL	REL	n/a
TSIMMIS	MW	OEM	OEM, REL, ...	GaV
SIMS/Ariadne	MW/MAS	Loom (OO)	REL, OO, SEMI, ...	LaV
Infomix	MW	SEMI	REL, XML, ...	GaV
Hippo	APP	REL	REL	n/a
ConQuer	APP	REL	REL	n/a
Rainbow	APP	REL	REL	n/a
Pegasus	DBMS	OO	OO, REL, ...	GaV
Nimble	MW	XML	XML	unknown
Carnot	MAS	Cyc	REL, SEMI, ...	LaV
InfoSleuth	MAS	Cyc	REL, SEMI, ...	LaV
Potter's Wheel	APP	REL	REL	n/a

上图展示了集成系统的主要架构和调解器/资源中使用的数据模型

对其中的参数进行解释：

**架构：**该系统实现了什么类型的架构？它是一个数据库管理系统（DBMS），一个多数据库管理系统（MDBMS），一个调节器-包装器系统（MW），一个多代理系统（MAS），还是一个独立的应用程序（APP）我们可以从架构的类型中推断出系统组件之间的耦合（紧密或松散）。



MDBMS：一个具有紧密耦合组件的数据库系统，将数据存储在不同的DBMS中，并使用一种特殊的语言来明确地直接访问分布式数据。

调解器-包裹器系统：由两类松散耦合的组件组成。调解器组件由用户使用标准语言进行查询，以透明地访问数据。数据存储分布在源中，并通过包装器提供，包装器在调解器和源之间翻译查询和数据。

多代理系统：是最松散耦合的系统架构，由相互作用的代理集合组成。这样一来，它就解除了调解器和包装器之间严格的等级分离。它可以被看作是并行数据管理系统（PDMSs）的前身

**内部数据类型：**调解器数据模型/内部数据模型。该系统在调解器中使用什么数据模型？如果它不是一个调解器-封装器系统，内部使用的是什么数据模型？它是基于关系世界（REL），基于一些面向对象的模型（OO），还是使用一些半结构化（SEMI）模型，如OEM、XML或DAPLEX

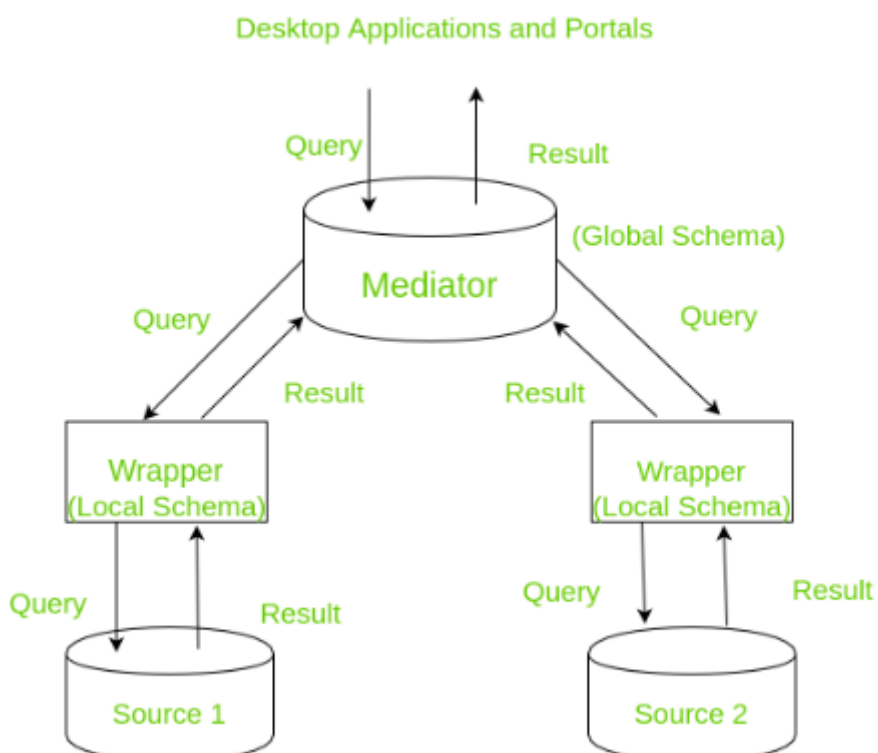
**集成模式。**系统使用哪种集成模型（GaV，LaV，GLaV）

GaV（Global-as-View）集成模型将全局模式（用于调解器或内部）表达为本地模式（用于数据源）的视图

LaV（Local-as-View）则将本地和全局模式反过来连接。它将本地模式表达为全局模式的视图。通过允许连接全局和局部模式的两种可能性

GLaV（Global-Local-as-View）试图结合两种模式的优点：LaV的更好的可扩展性和GaV的更简单的查询执行。

如图所示可以对集成模式进行理解：其实就是用调节器的模式表示源的形式还是用源的形式表示调节器的形式



**Table IX.** Query-Processing Abilities of Integration Systems and Categorization into Materialized/Virtual Integration System

System	Materialization	Manipulation	Access Method
Multibase	no	read-only	Daplex
Hermes	no	read-only	free text, logic based, graphical language
Fusionplex	no	read-only	ext. SQL
HumMer	no	read-only	ext. SQL
Ajax	yes	read-write	own language
TSIMMIS	no	read-only	MSL
SIMS/Ariadne	no	read-only	Loom query language
Infomix	no	read-only	SQL/Datalog
Hippo	no	read-only	SQL
ConQuer	no	read-only	SQL
Rainbow	no	read-only	SQL
Pegasus	no	read-write	HOSQL
Nimble	unknown	unknown	unknown
Carnot	no	read-only	SQL
InfoSleuth	no	read-only	SQL
Potter's Wheel	yes	read-write	GUI

上图中展示了集成系统的查询处理能力以及对物化/虚拟集成系统的分类

**物质化：**该系统是否将主系统中的来源数据进行物化（列数据库的名词），或者它是一个虚拟的集成系统，只提供对来源的访问。物化视图(Material View)是包括一个查询结果的数据库对象，它是远程数据的本地副本，或者用来生成基于数据表求和的汇总表。

**操作：**系统是否能够操纵来源中的数据，还是只允许读取访问？

**访问方法：**如何能够访问或查询该系统（查询语言、浏览、预制查询等）

上面的信息系统不包括第四组的信息系统，系统的结构和它的融合能力之间不存在关联性

## 4.2用于数据融合的信息系统的特性

**Table X.** General Conflict-Handling Properties of Systems (conflict types and data conflict awareness)

System	Conflict Types	Data Conflict Awareness
Multibase	schematic, data	yes, [Dayal 1983]
Hermes	schematic, data	yes, [Subrahmanian et al. 1995]
Fusionplex	schematic, object, data	yes, [Motro and Anokhin 2006]
HumMer	schematic, object, data	yes, [Bleiholder and Naumann 2006]
Ajax	schematic, object, data	yes
TSIMMIS	schematic, data	yes, [Papakonstantinou et al. 1996]
SIMS/Ariadne	schematic, data	yes, partly, [Ambite et al. 2001]
Infomix	schematic, data	yes, [Leone et al. 2005]
Hippo	schematic, object, data	yes
ConQuer	schematic, object, data	yes
Rainbow	schematic, object, data	yes
Pegasus	schematic, data	no
Nimble	unknown	unknown
Carnot	schematic	no
InfoSleuth	schematic	no
Potter's Wheel	schematic	no

**冲突类型。**在系统所使用的整合过程中，考虑了哪些类型的冲突，哪些类型的冲突有可能被处理：语义冲突（代表相同语义的不同属性）or 对象 or 数据级冲突

**数据冲突感知**系统是否意识到数据冲突？它们是否可能被自动处理或半自动处理

上图为系统的一般冲突处理属性（冲突类型和数据冲突感知）

**Table XI.** Solutions to Schema Matching/Mapping and Duplicate Detection in Integration Systems

System	Schema Matching/Schema Mapping	Duplicate Detection
Multibase	hand crafted	assumes global id
Hermes	hand crafted	assumes global id
Fusionplex	hand crafted, learned in Autoplex	assumes global id
HumMer	semi-automatically	global id generated by duplicate detection
Ajax	hand crafted	global id generated by similarity measure
TSIMMIS	wrapper generation	assumes global id
SIMS/Ariadne	wrapper generation	mapping table
Infomix	hand crafted	assumes global id
Hippo	hand crafted	assumes global id
ConQuer	hand crafted	assumes global id
Rainbow	hand crafted	assumes global id
Pegasus	hand crafted	assumes given mapping table
Nimble	unknown	mapping table, generated
Carnot	hand crafted	assumes global id
InfoSleuth	hand crafted	assumes global id
Potter's Wheel	n/a	n/a

**模式映射。**系统的**模式映射**能力是什么？如何创建映射以解决模式冲突？

**重复检测：**系统是如何检测和/或处理重复的对象的？它是否允许重复，是否假定有一个全局ID？

集成系统中模式匹配/映射和重复检测的解决方案

**Table XII.** Data Fusion Capabilities, Possible Strategies and How Fusion is Specified in Integration Systems

System	Fusion Possible	Fusion Strategy	Fusion Specification
Multibase	resolution	Trust your friends, Meet in the middle	manually, in query
Hermes	resolution	Keep up to date, Trust your friends, ...	manually, in mediator
Fusionplex	resolution	Keep up to date	manually, in query
HumMer	resolution	Keep up to date, Trust your friends, Meet in the middle, ...	manually, in query
Ajax	resolution	various	manually, in workflow definition
TSIMMIS	avoidance	Trust your friends	manually, rules in mediator
SIMS/Ariadne	avoidance	Trust your friends	automatically
Infomix	avoidance	No Gossiping	automatically
Hippo	avoidance	No Gossiping	automatically
ConQuer	avoidance	No Gossiping	automatically
Rainbow	avoidance	No Gossiping	automatically
Pegasus	ignorance	Pass it on	manually
Nimble	ignorance	Pass it on	manually
Carnot	ignorance	Pass it on	automatically
InfoSleuth	unknown	Pass it on	unknown
Potter's Wheel	ignorance	Pass it on	manually, transformation

**同对象的融合机制：**相同的现实世界对象的融合是否可能，有哪些类别的策略（无知、解决、回避）？

**融合策略：**哪些具体的融合策略有可能在系统中使用或在系统中实施？（相信某一个表、折中、用最新值、返回多个值、不管冲突值）

**融合的规范：**如何进行融合：以自动、半自动或手动方式？

每一张图都把信息系统分成了三类，分别是解决冲突、避免冲突和无视冲突

## 4.3 解决冲突的系统

### 4.3.0

这一组中的所有系统都允许全面解决冲突的数据，使用各种策略并遵循不同的实施范式。在大多数情况下，对这些系统的查询结果是一个完整而简洁的答案。

#### 4.3.1 Multibase数据库信息系统

Multibase：多数据库系统，源中的数据使用DAPLEX数据模型进行描述，允许对这些本地数据定义全局视图。并接受DAPLEX进行全局查询。查询优化通过对源的查询优化来完成

利用泛化原则将源数据中的值整合到全局视图中

处理冲突的方法：使用outer join操作，然后进行聚合，通过使用基本的聚合函数如min、max、sum、count、average或choice来融合冲突的值。不能解决源内重复数据之间的冲突。

#### 4.3.2 Hermes

同时整合了数据源和推理设施并通过调节器连接。调解器是用一个注释逻辑的变体来指定的，注释是概念、对象或价值被输入系统的时间和它的可靠性。换句话说，在调节其中可以生成每一个数据的时间以及其可靠性，用户也可以借助这个数据进行数据选择策略的制定。

可以解决模式和数据层面的冲突（使用信任朋友以及保持最新）

#### 4.3.3 Fusionplex

"plex"由三个系统组成。Multiplex，Fusionplex，以及Autoplex。

Multiplex是一个虚拟的多数据库系统，能够整合来自不同异质来源的信息，是其他两个系统的基础。

Fusionplex根据提供的数据源内容的质量信息作为额外的特征属性，增加了不一致识别和调和设施。

Autoplex增加了对自动添加新数据源的支持，

一般来说我们认为：integrated data sources（整合后的数据源）的模式是不同的或者是完全重复（有两行完全重复）的，但是一般的数据源可能存在重复的对象，以及不同和矛盾的对象表示。

Fusionplex允许用户在元组层面解决扩展不一致的问题。该系统根据一个全局键将代表同一对象的元组分组。然后，使用质量元数据（时间戳、成本、准确性、可用性、清除等），在额外的特征列中表示，只选择高质量的图元用于融合过程。用户对特征的重要性的偏好也被考虑在内。

接下来，融合函数（min, max, avg, any, .....）按属性应用于每个对象，以得出每个属性和对象的一个值，即最终的对象表示。最后，计算出最终表征的特征值。

在Autoplex中，每个数据源的规则被学习，什么样的查询方式能够对全局数据库产生良好的贡献，选择这些查询合并到全局数据库中。

#### 4.3.4 HumMer

一个综合信息系统，并允许对几个远程和异质数据源进行半自动的虚拟整合，他的整合步骤与上面介绍的数据整合的步骤类似，包括：模式匹配、重复检测和数据融合

用户可以在任何一个步骤与系统进行交互：例如判断元组是否是重复的，改变默认的模式映射关系等等；重复的检测被分到一个分组中，处理重复的方式是使用用户自定义的函数或者sql默认的基本聚合函数进行分组与聚合

#### 4.3.5 Ajax

提供了特殊的转换函数（例如单位间的转换、格式的转换）来识别语义相同的模式

重复检测（匹配）是通过对对象表示的分组分配键来完成的，而合并运算符通过聚合来解决现有的数据冲突

## 4.4. Conflict-Avoiding Systems

### 4.4.0简介

这一组的系统允许通过避免冲突来处理冲突的数据

#### 4.4.1 TSIMMIS

该系统通过特殊的封装器与调节器来传输数据，数据被表示为一个四元组（对象ID、标签、类型、值），没有像关系模型那样的定义模式。

在存在数据冲突的情况下，该系统不允许在调解器中解决冲突，而是使用简单的 "信任你的朋友" 策略来避免冲突，只允许在调解器规范期间指定一个首选来源

#### 4.4.2 SIMS&&Ariadne

查询的处理分三步进行。首先，选择所有相关的源，并重新制定以全局模式表示的查询，以便在本地源执行。第二，系统生成一个详细的查询执行计划。在第三步中，系统自动进行了查询优化，查询被重新表述为一个执行成本较低的语义等价计划

需要手动构建映射表来连接不同源中的ID，对象ID的值来自于一个指定的来源。

#### 4.4.3 Infomix

使用逻辑程序来指定底层不一致数据源的所有修复，并从那里得到查询的一致答案，逻辑编程是一种主要基于形式逻辑的编程范式。任何用逻辑编程语言编写的程序都是一组逻辑形式的句子，表达了一些问题领域的事实和规则。

然而，逻辑程序的固有复杂性限制了它只能用于小型数据库

#### 4.4.4 HIPPO

提供了一个高效计算修复的数据结构：冲突超图

相对于普通图而言，超图可以更加准确的描述存在多元关联的对象之间的关系。超图与普通图的主要不同在于图中边上顶点的个数的不同，在普通图中，一条边包含两个顶点，在超图中，边被称为超边，一条超边包含多个顶点。通过对图结构的遍历根据我们选定的冲突处理函数即可完成冲突数据的处理。

#### 4.4.5 ConQuer

给定一组关键的约束条件，该系统将SQL查询重写成一个只返回一致答案的等价查询，即使用这些约束对不一致的数据进行筛选从而返回一致数据。

#### 4.4.6 Rainbow

是一个查询框架，其作用是由用户给出完整性约束，并针对不一致数据进行处理。处理的方式包括3.3.2节中提到的优先合并或者3.2.3节中的匹配连接。

## 4.5 忽略冲突的系统

### 4.5.1 Pegasus

该系统使用面向对象的数据模型来表示数据，并提供合适的查询语言HOSQL，对该某一个的一个相同的对象有重复的表示，系统可以通过不同的对象表示中认识并标记冲突，但没有得到解决，需要查询者手工进行解决。

因为Pegasus是一个分布式数据库，所以在全局查询时进行了优化以减少通信成本。

### 4.5.2 Nimble

使用xml作为数据模型，重复数据的存在是可以被识别的，系统试图通过挖掘不同表格中可以连接的列来找到它们，没有明确的指出冲突的处理机制，一般而言是使用join操作以及手动操作进行冲突处理

#### 4.5.3 Carnot

作为全局数据模型，Carnot使用Cyc知识库，即建立现实世界知识模型，并通过建立一个复杂而完整的现实世界概念的本体来使其为信息系统所用。处理查询时将查询语句映射到cyc本体，在由其映射到数据源本身。其重要的工作就是将数据源映射到全局模式中，这个映射是由专家来做的，查询如果得到冲突也需要手动进行处理。

#### 4.5.4 InfoSleuth

是对于carnot的拓展，并将其扩展到更动态的WWW场景中，资源会随着时间的推移而被添加和删除，在查询时不清楚有哪些资源可用

重复的对象和可能冲突的值的问题既没有被明确认识到，也没有得到解决

#### 4.5.5 Potter's Wheel

只能对一个源进行操作，它的主要重点是用户对清洗操作的交互式定义：用户可以对数据进行定义转换、这样一来，列中的数值就可以被重新格式化，改变，复制到另一列，或分割成两列，复杂的转换组合不仅可以保存、重新加载，而且可以在样本上执行，也可以在整个数据集中执行。

不能进行重复检测，也无法进行合并。

## 4.6. Other Systems

### 4.6.0 简介

以下系统没有详细考虑重复检测或数据融合技术，在这里只做简要概述

#### 4.6.1 Research System

与所有其他概率数据库一样，这里采取的方法是将所有数据保存在数据库中，用质量元数据和lineage来注释进行综合表达，并提供一个强大的查询语言。这样他们就没有一个内置的重复对象的概念，并将冲突处理推迟到用户。

#### 4.6.2 Commercial DBMS

它们提供了使用第3节中提到的关系技术的全部功能（join、union、分组聚合等等），有时还通过供应商对用户定义的函数、聚合或程序的特定支持来扩展它们，数据融合阶段却经常被忽视，重复检测使用约束来体现，如果想要获得高完整性和高简洁性的结果需要用户自己编写自定义函数或语句来获得查询结果，应用通常只支持简单的操作，但并不负责组合他们。

#### 4.6.3 Peer Data Management Systems

查询可以针对任何一个对等体的模式提出，并沿着映射路径进行转换，以便所有参与的对等体都能对最终结果作出贡献，冲突不会被解决，而是被识别和隐藏。

#### 4.6.4 总结

在迄今为止建立的所有研究信息集成系统中，只有一些能够真正处理冲突的数据。大多数集成系统只处理模式冲突，同时只有少数系统能够识别和处理所有可能的源内和源外的数据冲突。

与研究系统相比，商业化的信息系统只有有限的数据融合能力，如果有的话。重复记录的问题在企业中得到了相对较大的关注，因此，除了标准的DBMSs之外，还有许多专门的软件产品能够检测数据中的重复部分

##

