C++数据库类型及转换应用

陈哲瀚 陆文莲
江苏省横林高级中学，江苏 常州 213101

[摘要]随着计算机的不断发展，C++是在 C 语言的基础上开发的一种通用编程语言，其具有丰富的功能和灵活性，应用广泛，可以用于开发复杂的数据库应用程序。通过使用 C++语言，可以对数据库进行设定、管理和操作，实现办公数据库的自动化转换。但是在不同编译器下，C++数据类型会有所差异。因此，文章进一步探究和分析 C++数据库类型及转换应用，以期为其应用提供借鉴和参考。

[关键词]C++数据库类型；转换；应用
DOI：10.33142/sca.v7i10.13644 中图分类号：TP3 文献标识码：A

C++Database Types and Conversion Application

CHEN Zhehan, LU Wenlian
Jiangsu Henglin Senior High School, Changzhou, Jiangsu, 213101, China

Abstract: With the continuous development of computers, C++is a general-purpose programming language developed on the basis of C language. It has rich functions and flexibility, and is widely used for developing complex database applications. By using the C++language, it is possible to set up, manage, and operate databases, achieving automated conversion of office databases. However, under different compilers, the data types of C++may vary. Therefore, the article further explores and analyzes the types and conversion applications of C++databases, in order to provide reference and guidance for their application.

Keywords: C++database type; transformation; application

引言

在软件开发过程中，经常需要涉及与数据库的交互，如：存储用户信息、管理产品数据等。C++作为一种功能丰富的编程语言，可以通过多种方式与数据库进行集成，优化数据库操作的性能和效率。不同类型数据库的转换是一种常见的操作，其对于保证程序的正确性和性能至关重要。通过规范的数据转换和数据库操作，以及合适的异常处理机制，能够及时捕获和处理数据库操作中可能出现的异常，保证程序的稳定性和数据的完整性。

因此，结合 C++数据库类型和转换平台，分析语言技术的相关应用，可以有效地提升软件开发的效率和质量，实现高效的数据库操作和管理，确保软件系统的稳定性和可靠性[1]。

1 C++语言特点和数据库类型

1.1 C++语言特点

C 语言以其简洁和高效被广泛使用，而 C++在此基础上增加了面向对象编程的支持，丰富了编程的表达能力。目前，C++支持多种编程范式，可以为计算机用户提供更多的语言决策，这是传统语言编制不具备的功能。现阶段，C++不仅从计算机语言方面提供处理依据，也能在实际程序操控中完成自动化处理，形成了相对稳定的程序语言环境。例如：C++可面向对象编程、泛型编程和过程化编程，进一步提升了数据处理效率。

1.2 C++数据库类型

随着计算机软硬件技术快速发展，程序语言编写流程有了多方面提升， 数据库结构层次设定的方式不一样，可以满足不同类型的数据应用要求。目前，常用 C++数据类型有八种（见表 1），主要以 80×86 处理器的 ViusalC++ 2012，以及 gcc4.8.1 的长度。这说明 C++数据类型的长度和取值范围，会受到编译器和处理器架构的影响。如果在不同的编译器下进行编译，数据类型的长度和取值范围可能会有所不同。因此，在跨平台开发或者在不同编译器下编写程序时，需要格外留意数据类型的一致性，以确保程序的可移植性和稳定性[2]。

表 1 常见的 C++数据类型

类型名	长度（字节）	取值范围
bool	1	False, true
char	1	- 128-127
signed char	1	- 128-127
unsigned char	1	0-255
signed short	2	- 32768-32767
unsigned short	2	0-65535
signed int	4	- 2147483648-2147483647
unsigned int	4	0-4294967295

2 C++数据库转换应用

在计算机操作系统中数据库起到非常重要的作用，是

管理和存储数据的核心组成部分。当数据库资源出现流失或损坏，则会直接影响系统的性能和应用价值。随着技术的发展和业务需求的变化，原有的数据库结构已无法满足新的需求。所以需要对数据库进行结构上的升级和改造，以便更好地支持新的功能和性能需求。通过结合C++的特点和设计合理的控制方案，解决传统数据库系统存在的功能性缺失问题，如：性能瓶颈、扩展性差、灵活性不足等，从而提升数据库的整体功能和性能。在数据库中，数据的转换是指将数据从一种形式或结构转换为另一种形式或结构的过程。因此，转换被视为数据库资源多功能应用的一个新方法，不仅充分展示了C++的应用层次，还可以帮助企业或个人构建一个更加稳定和高效的数据程序平台。类型转换可以分为两种，以下详细介绍。

2.1 隐式类型转换

2.1.1 定义

C++设定了一些标准化的规则，规定了如何在不同类型之间进行转换。当程序中涉及不同数据类型的操作时，C++会根据这些规则自动进行数据类型的转换。这种转换在编译时由编译器完成，程序员无须手动干预，因此它是“自动”的，所以称之为“隐式类型转换”[3]。例如：在代码中将一个整型变量与一个浮点型变量进行运算，C++会自动将整型变量转换为浮点型，以确保运算的准确性，避免数据损失。隐式类型转换有助于简化代码，使程序员不必频繁地进行手动转换，从而降低出错的可能性。同时，也能提高代码的可读性，使逻辑更加清晰。隐式类型转换原则见图1。

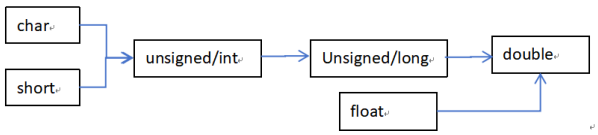

图1 隐式类型转换原则

2.1.2 发生隐式类型转换的常见类型

第一，多种数据类型的算术表达式中。即：

```
1  Int a=2;
2  float b=3.4;
3  double d=2.2;
4  a+b+c;
```

第二，将一种数据类型赋值给另外一种数据类型变量。即：

```
1  Int a=2;
2  float b=3.4;
3  long double d=2.2;
4  b=a;
5  d=a;
```

第三，函数调用时，若实参表达式与形参的类型不相符。即：

```
1  int Min（int a，int b）{
2  return a<b?a:b;
3  }
4  int a=2;
5  float b=3.4;
6  int x= Min（b，a+3.5）;
```

第四，函数返回时，如果返回表达式的值与函数返回类型不同。

```
1  double add（int a，Int b）{
2  return a+b;
3  }
```

2.2 显式类型转换

2.2.1 定义

显示类型转换（或强制类型转换）是指程序员明确地将一种数据类型转换为另一种数据类型的过程。与隐式类型转换不同，显示类型转换需要程序员的手动干预，通常是通过特定的语法来实现的。

在某些情况下，自动的隐式类型转换可能无法满足程序的需求。例如，程序员可能需要将一个较大的数据类型（如浮点型）转换为较小的数据类型（如整型）。在这种情况下，显示类型转换可以帮助程序员明确地控制转换过程，确保数据以期望的方式进行处理[4]。C++提供了更加严格的类型转换，提供更好地控制转换过程，其使用格式为：目标类型变量=xxxx cast〈目标类型〉(源类型变量)。

2.2.2 显式类型转换新增转换

第一，静态类型转换（static cast）。其是一种用于进行隐式转换的逆向转换的方式，不仅可以用于基本数据类型之间的转换，也可以将void*转换为其他类型的指针。还可以用于自定义类型之间的转换，这种转换被称为向上造型，允许将派生类对象看作基类对象，以便进行统一的处理。需要注意的是，它不能用于两个高风险的转换，即整型和指针之间的相互转换、不同类型的指针或引用之间的转换，以避免程序错误和数据损失。第二，重解释类型转换（reinterpret cast）。常用于进行指针或引用之间的转换，无论它们之间是否存在继承或类型关系。还可以用于指针和整数类型之间的转换，通常用于处理底层内存操作或者进行指针地址和整数之间的互相转换。第三，常类型转换（const_cast）。用于移除指针或引用的常属性，主要应用场景是在需要修改本来被标记为常量的数据时使用。例如修改一个指向常量的指针以允许写入操作。第四，动态类型转换 （dynamic_cast ）。主要用于处理多态类型的指针或引用的向下转型，在执行转换时会进行运行时的类型检查，避免不安全的转型操作。

2.3 自己声明的类型转换

自己声明的类型转换是指在自定义的类中，通过编写特定的代码，使得该类可以与其他类型进行转换。这样的

转换是针对特定类而言的，不同于系统自带的隐式或显式类型转换。在C++中，当使用用户自定义类型进行数据转换时，编译器并不会自动知道如何进行这种转换，所以会导致编译错误或不符合预期的行为。因此，可以定义一个专门的函数，使得编译系统知道怎样将一种类型转换为另一种类型。一种是转换构造函数，在类的构造函数中定义一个接受其他类型作为参数的构造函数，这样编译器就能通过这个构造函数进行类型转换。另一种是类型转换函数，在类中定义一个类型转换操作符，使得对象可以被隐式或显式地转换成其他类型。这种类型转换函数的返回值是目标类型，函数名是operator后跟目标类型名，通过定义这个函数，可以实现从类对象到标准类型的转换[5]。

3 C++数据库转换应用问题和解决方法

3.1 隐式类型转换

在C++中，隐式类型转换主要是依靠编译器进行自动类型转换，但在转换应用中很容易出现数据准确丢失或运行错误等现象。

例如：

```
1  int a=10;
2  double b=3.14;
3
4  double c=a/b; //预期结果为3.3333，但实际结果为3
```

在代码中，a是int类型的变量，b是double类型的变量，而a/b的结果被编译器自动转换为int类型，进而导致结果的小数部分被截断。针对这种问题，我们可以利用以下两种方式进行解决。

（1）显式转换

为了更好地避免出现隐式类型转换中自动转换造成的结果错误，在C++中可以利用显示类型对数据进行转换。通过static_cast操作符，可以明确指定需要进行的类型转换，使代码更加清晰明了。例如：在本次代码中将4 double c=a/b修改为4 double c=static_cast<double>(a)/b，这是在明确告知编译器需要将a转换为double类型，进而得到预期结果。

（2）优化计算顺序

优化计算顺序是提高程序性能和可读性的重要手段。在C++及其他编程语言中，通过合理安排计算顺序，可以减少不必要的运算、提高效率，并使代码更易于理解。例如：在本次代码中将4 double c=a/b修改为4 double c=a/（double）b，这是将b转换为double类型，使得结果为3.333。

通过合理地应用显式类型转换和优化计算顺序，可以避免隐式类型转换可能引起的结果错误，确保程序的准确性和可靠性。这种做法有助于提高代码的可读性和性能，避免潜在的数据损失问题。

3.2 字符串和数值类型转换

在C++中，经常需要将字符串转换为数值类型以便进行数值计算，或者将数值类型转换为字符串以便输出或存储。但是在转换中会出现一些问题，影响转换效果。例如：字符串转换为数值类型，

```
1 #include <iostream>
2 #include <string>
3
4 int main() {
5 std::string str = “12345”;
6
7 int num = std::stoi(str); // string to int
8 long long_num = std::stol(str); // string to long
9 float float_num = std::stof(str); // string to float
10 double double_num = std::stod(str); // string to double
11
12 std::cout << num << std::endl; // 输出：12345
13 std::cout << long_num << std::endl; // 输出：12345
14 std::cout << float_num << std::endl; // 输出：12345.0
14 std::cout << double_num << std::endl; // 输出：12345.0
16
17 return 0;
18 }
```

在字符串和数值类型转换中，针对字符串转换为数值类型可以利用stoi、stol、stof、stod；针对数值类型转换为字符串可以利用to_string。正确的数据类型转换，可以提高程序的性能和正确性。由于不同的转换函数适用于不同的场景，所以在实际编程中，应该根据具体需求选择合适的转换函数，并结合异常处理机制，创建更健壮的程序，有效地处理字符串和数值类型之间的转换操作，不仅能提高代码的可读性和维护性，还能有效地避免运行时错误，提升用户体验。

4 C++转换应用实例

随着经济全球化发展，我国的工业经济逐步走向世界舞台，为此工业信息化生产成为实现这一目标的重要平台。在工业生产过程中，计算机技术和程序语言的应用变得至关重要，不仅可以提高生产效率和质量，还可以实现生产过程的自动化和智能化。

C++作为一种高级编程语言，与现有的计算机控制系统无缝对接，可以更精确地设计和规划产品的结构和生产

流程，包括从原型设计到实际加工中的各个环节，确保生产过程的高效性和准确性。还可以依照数据库整体标准对生产数据的管理和优化，不仅涉及数据存储和查询的效率，还包括对生产过程中可能出现的变化和需求的快速响应能力。C++编程能力的强大使得工业生产中的技术成果可以更直接地转化为实际的产品和服务，这种直接性和实用性，提升了工业生产的效率和质量，同时也降低了成本和资源浪费。通过 C++技术在工业生产中的应用，企业可以建立更为先进和智能化的作业平台，有效提高生产的灵活性和响应速度，并为企业带来更大的竞争优势和市场份额。

另外，在时代的不断变化下，经济行业也需要转型升级，以适应新的市场需求和全球化竞争的挑战。这就意味着我国工业生产需要及时更新技术应对策略，以确保生产过程的高效性、质量和安全性。通过优化企业内部的管理体制，提高生产效率、降低成本、增强创新能力。在企业办公自动化系统中 C++起到重要作用，首先，可以处理各种类型的原始数据，包括文本、数字、图像等。通过编写高效的程序，将这些数据从一种格式转换成另一种格式，使得数据能够被不同的系统或应用程序所理解和使用。其次，企业的不同业务和领域可能对数据有不同的需求。C++的灵活性使得它能够根据具体的业务需求进行定制开发，从而能够处理特定行业或领域的数据。

通过转换和处理原始数据，C++提高了数据的可操控性，使得企业可以更方便地分析、管理和决策，从而提高业务运作的效率和准确性，这对于提升企业的竞争力和市场反应速度至关重要。

5 结语

综上所述，支持多种编程是 C++应用特点，无论是企业或个人用户，必须按照 C++标准设定数据库，从而实现了原始数据的自动化转变。针对数据库结构类型存在的不足，计算机编程时要按照 C++数据库类型及其转换方式操作，进而体现出 C++的可操控性特点，实现数据程序语言的优化转变。

在进行跨平台开发时，程序员需要注意不同编译器对数据类型的长度和取值范围的影响，确保程序的可移植性。在数据库操作中，通过合理使用隐式和显式类型转换，程序员可以有效地管理和处理数据，提高应用程序的性能和可靠性。随着对数据库应用需求的不断演进，C++的类型转换技术将继续在软件开发中发挥重要作用，推动软件行业朝着更加智能化和高效化的方向发展。

[参考文献]

[1]张浩,朱志强,陈志强.基于 CPU 环境的多类型数据库同步方法、装置及设备[J].信息技术与信息化,2024(2):84-87.
[2]李力,蓝天飞,郭越凡,等.基于 c++的三维闪电回击数据再处理系统设计[J].湖北大学学报(自然科学版),2024,46(2):289-296.
[3]胡平,吴福祥.跨语言同词化数据库与词汇类型学研究[J].当代语言学,2023,25(4):562-576.
[4]郭静.基于多类型宏的 Access 数据库用户多角色登录验证系统设计[J].自动化技术与应用,2020,39(12):150-153.
[5]李业田.常用数据库类型介绍与解析[J].电子世界,2018(20):101.

作者简介：陈哲瀚（2007.5—），毕业院校：江苏省横林高级中学，所学专业：C++，目前就读学校：江苏省横林高级中学。