全面探讨PL_SQL的复合数据类型 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/167/2021_2022__E5_85_A8_E9_9D_A2_E6_8E_A2_E8_c102_167095.htm PL/SQL有两种复合数据结构：记录和集合。记录由不同的域组成，集合由不同的元素组成。在本文中我们将讨论记录和集合的类型、怎样定义和使用记录和集合。 PL/SQL 记录 记录是PL/SQL的一种复合数据结构，scalar数据类型和其他数据类型只是简单的在包一级进行预定义，但复合数据类型在使用前必须被定义，记录之所以被称为复合数据类型是因为他由域这种由数据元素的逻辑组所组成。域可以是scalar数据类型或其他记录类型，它与c语言中的结构相似，记录也可以看成表中的数据行，域则相当于表中的列，在表和虚拟表（视图或查询）中非常容易定义和使用，行或记录中的每一列或域都可以被引用或单独赋值，也可以通过一个单独的语句引用记录所有的域。在存储过程或函数中记录也可能有参数。 创建记录 在PL/SQL中有两种定义方式：显式定义和隐式定义。一旦记录被定义后，声明或创建定义类型的记录变量，然后才是使用该变量。隐式声明是在基于表的结构或查询上使用%TYPE属性，隐式声明是一个更强有力的工具，这是因为这种数据变量是动态创建的。 显式定义记录 显式定义记录是在PL/SQL程序块中创建记录变量之前在声明部分定义。使用type命令定义记录，然后在创建该记录的变量。语法如下： TYPE record_type IS RECORD (field_definition_list). field_definition_list是由逗号分隔的列表。 域定义的语法如下： field_name data_type_and_size [NOT NULL][{:=|DEFAULT} default_value] 域名必须服从与表

或列的命名规则相同的命名规则。下面我们看一个例子：DELCARE TYPE stock_quote_rec IS RECORD (symbol stock.symbol%TYPE ,bid NUMBER(10,4) ,ask NUMBER(10,4) ,volume NUMBER NOT NULL:=0 ,exchange VARCHAR2(6) DEFAULT NASDAQ). real_time_quote stock_quote_rec. variable 域定义时的%TYPE属性用于引用数据库中的表或视图的数据类型和大小,而在此之前程序不知道类型和大小。在上面的例子中记录域在编译时将被定义为与列SYMBOL相同的数据类型和大小,当代码中要使用来自数据库中的数据时，在变量或域定义中最好使用%TYPE来定义。 隐式定义记录 隐式定义记录中，我们不用描述记录的每一个域。这是因为我们不需要定义记录的结构，不需要使用TYPE语句，相反在声明记录变量时使用%ROWTYPE命令定义与数据库表，视图，游标有相同结构的记录，与TYPE命令相同的是它是一种定义获得数据库数据记录的好方法。 DECLARE accounter_info accounts%ROWTYPR. CURSOR xactions_cur(acct_no IN VARCHAR2) IS SELECT action,timestamp,holding FROM portfolios WHERE account_nbr=acct_no . xaction_info xactions_cur%ROWTYPE. variable 有一些PL/SQL指令在使用隐式定义记录时没有使用%ROWTYPE属性，比如游标FOR循环或触发器中的:old和:new记录。 DELCARE CURSOR xaction_cur IS SELECT action,timeamp,holding FROM portfolios WHERE account_nbr=37 . BEGIN FOR xaction_rec in xactions_cur LOOP IF xactions_rec.holding=ORCL THEN notify_shareholder. END IF. END LOOP. 使用记录 用户可以给记录赋值、将值传递给其他程序。记录作为一种复合数据结

构意味作他有两个层次可用。用户可以引用整个记录，使用0select into或fetch转移所有域，也可以将整个记录传递给一个程序或将所有域的值赋给另一个记录。在更低的层次，用户可以处理记录内单独的域，用户可以给单独的域赋值或者在单独的域上运行布尔表达式，也可以将一个或更多的域传递给另一个程序。 引用记录 记录由域组成，访问记录中的域使用点（.）符号。我们使用上面的例子看看 DELCARE TYPE stock_quote_rec IS RECORD (symbol stock.symbol%TYPE ,bid NUMBER(10,4) ,ask NUMBER(10,4) ,volume NUMBER NOT NULL:=0 ,exchange VARCHAR2(6) DEFAULT NASDAQ). TYPE detailed_quote_rec IS RECORD (quote stock_quote_rec ,timestamp date ,bid_size NUMBER ,ask.size NUMBER ,last_tick VARCHAR2(4)). real_time_detail detail_quote_rec. BEGIN real_time_detail.bid_size:=1000. real_time_detail.quote.volume:=156700. log_quote(real_time_detail.quote). 给记录赋值 给记录或记录中的域赋值的方法有几种，可以使用SELECT INTO或FETCH给整个记录或单独的域赋值， 可以将整个记录的值赋给其他记录，也可以通过给每个域赋值来得到记录，以下我们通过实例讲解每一种赋值方法。 1、使用SELECT INTO 使用SELECT INTO给记录赋值要将记录或域放在INTO子串中，INTO子串中的变量与SELECT中列的位置相对应。 例： DECLARE stock_info1 stocks%ROWTYPE. stock_info2 stocks%ROWTYPE. BEGIN SELECT symbol,exchange INTO stock_info1.symbol,stock_info1.exchange FROM stocks WHERE symbol=ORCL. SELECT * INTO stock_info2 FROM stocks

WHERE symbol=ORCL. 2、使用FETCH 如果SQL语句返回多行数据或者希望使用带参数的游标，那么就要使用游标，这种情况下使用FETCH语句代替INSTEAD INTO是一个更简单、更有效率的方法，但在安全性较高的包中FETCH的语法如下： FETCH cursor_name INTO variable. 我们改写上面的例子： DECLARE CURSOR stock_cur(symbol_in VARCHAR2) IS SELECT symbol,exchange,begin_date FROM stock WHERE symbol=UPPER(symbol_in). stock_info stock_cur%ROWTYPE BEGIN OPEN stock_cur(ORCL). FETCH stock_cur INTO stock_info. 使用赋值语句将整个记录复制给另一个记录是一项非常有用的技术，不过记录必须精确地被声明为相同的类型，不能是基于两个不同的TYPE语句来获得相同的结构。 例： DECLARE TYPE stock_quote_rec IS RECORD (symbol stocks.symbol%TYPE ,bid NUMBER(10,4) ,ask number(10,4) ,volume NUMBER). TYPE stock_quote_too IS RECORD (symbol stocks.symbol%TYPE ,bid NUMBER(10,4) ,ask number(10,4) ,volume NUMBER). --这两个记录看上去是一样的，但实际上是不一样的 stock_one stocks_quote_rec. stock_two stocks_quote_rec. --这两个域有相同的数据类型和大小 stock_also stock_rec_too；--与stock_quote_rec是不同的数据类型 BEGIN stock_one.symbol:=orcl. stock_one.volume:=1234500. stock_two：=stock_one.--正确 syock_also：=stock_one.--错误，数据类型错误 stock_also.symbol:=stock_one.symbol. stock_also.volume:=stock_one.volume. 记录不能用于INSERT语句和将记录直接用于比较，下面两种情况是错误的： INSERT INTO stocks VALUES (stock_record). 和 IF

stock_rec1>stock_rec2 THEN 要特别注意考试中试题中有可能用%ROWTYPE来欺骗你，但这是错误的，记住这一点。还有可能会出现用记录排序的情况，ORACLE不支持记录之间的直接比较。对于记录比较，可以采用下面的两个选择：. 设计一个函数，该函数返回scalar数据类型，使用这个函数比较记录，如 IF sort_rec(stock_one)>sort_rec(stock_two) THEN . 可以使用数据库对象，数据库对象可以使用order或map方法定义，允许oracle对复合数据类型进行比较。关于数据库对象的讨论已经超越了本文的范围，要详细了解数据库对象，可以查阅oracle手册。 PL/SQL集合 集合与其他语言中的数组相似，在ORACLE7.3及以前的版本中只有一种集合称为PL/SQL表，这种类型的集合依然保留，就是索引（INDEX_BY）表，与记录相似，集合在定义的时候必须使用TYPE语句，然后才是创建和使用这种类型的变量。 集合的类型 PL/SQL有三种类型的集合 . Index_by表 . 嵌套表 . VARRAY 这三种类型的集合之间由许多差异，包括数据绑定、稀疏性(sparsity)、数据库中的存储能力都不相同。绑定涉及到集合中元素数量的限制，VARRAY集合中的元素的数量是有限，Index_by和嵌套表则是没有限制的。稀疏性描述了集合的下标是否有间隔，Index_by表总是稀疏的，如果元素被删除了嵌套表可以是稀疏的，但VARRAY类型的集合则是紧密的，它的下标之间没有间隔。 Index_by表不能存储在数据库中，但嵌套表和VARRAY可以被存储在数据库中。 虽然这三种类型的集合有很多不同之处，但他们也由很多相似的地方：. 都是一维的类似数组的结构 . 都有内建的方法 . 访问由点分隔 Index_by表 Index_by表集合的定义语法如下： TYPE type_name IS

TABLE OF element_type [NOT NULL] INDEX BY BINARY_INTERGET.