Object Types vs. PL/SQL Types: A Practical Example

Dr. Paul Dorsey

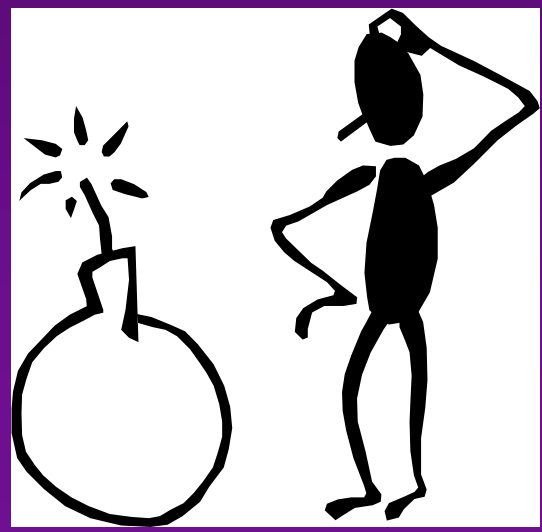
Dulcian, Inc.

www.dulcian.com

Evolution of SQL and PL/SQL

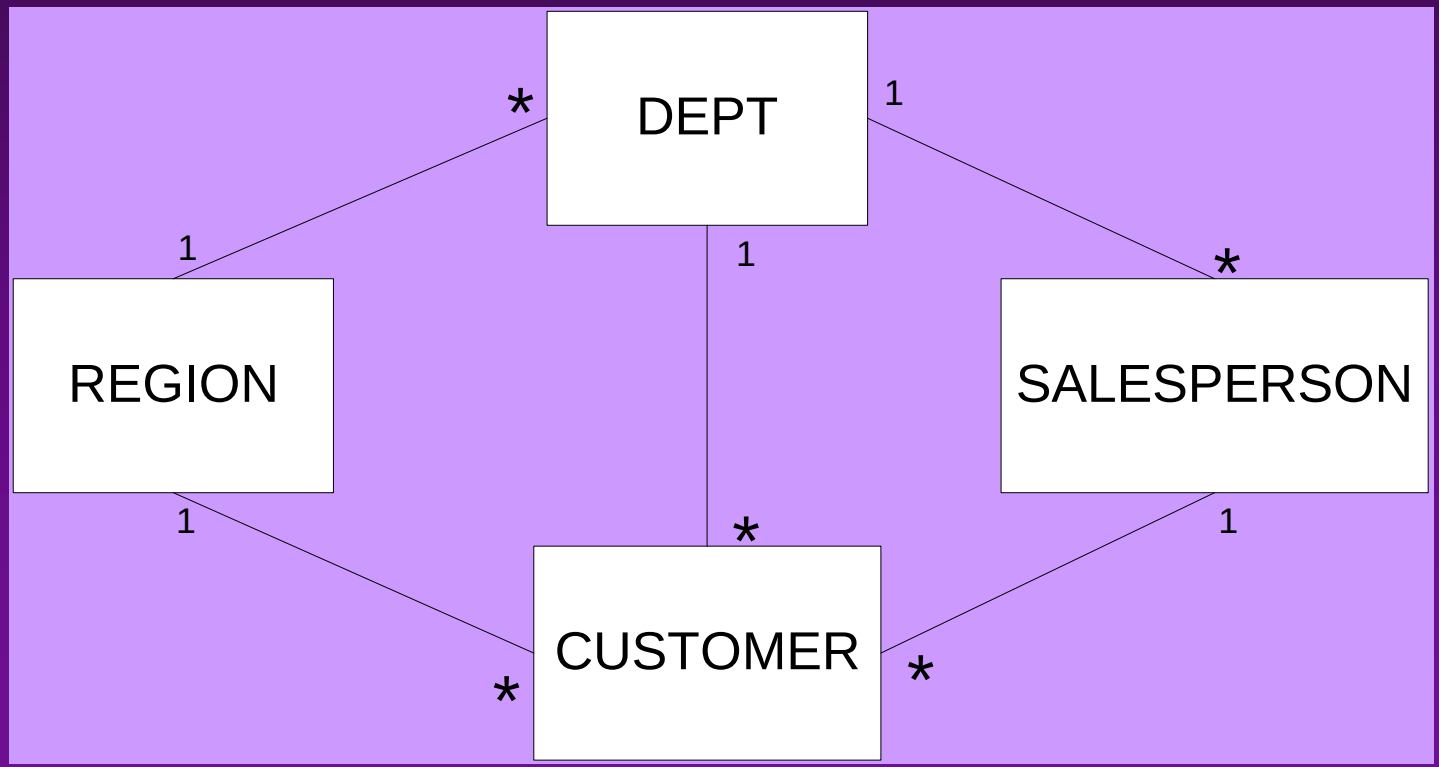
- Since release of the Oracle8 database, there have been new additions to SQL and PL/SQL.
  - Many developers are unaware of these.
  - Many developers who are aware of the additions may not be sure how to use them.
- This presentation will show a specific example of how new SQL and PL/SQL features were used to solve a specific real-world problem.

Part 1.
The Problem



Data Source

- A seemingly simple report was needed based on a small number of warehouse tables.
- Simplified data model



System description

- Customers are entered into the system.
- They move through different milestones.
  - Prospects > Leads > Actual Customers
    - Real system included 10 different milestones.
  - Need to track when each milestone is reached.
- Other customer attributes required tracking (DOB, height, weight).

Reporting Front-End

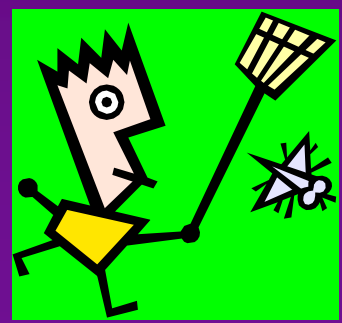
- Flexible reporting front-end needed.
- Users can specify any number of filters.
  - Example: “Customers over 40 years of age from California reaching the Lead milestone.”
- Report Display options
  - Example: Region
    Department
    Salesperson
- Report detail at Salesperson level
  - Breaks at Department and Region levels
  - Actual report had 6 levels.

Report details

- Users can specify:
  - Desired level of report detail
  - Location of breaks
  - Report columns
    - Used to group customers reaching a particular milestone
    - Up to 10 different columns needed for the report
  - Up to 20 statistics to appear within each cell
    - Ex. average number of phone calls, average age, customer count, etc.

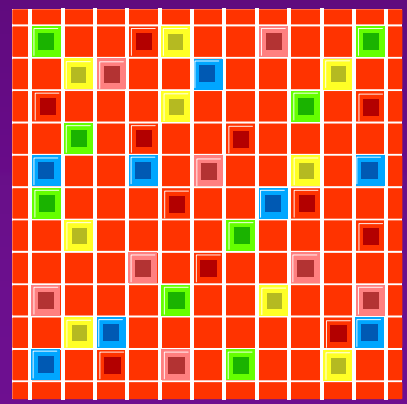
Reporting Requirements

- Generic filtering
  - 2-3 filter criteria
- On-the-fly structural specification
  - 3-5 levels of breaks (200-400 rows in the report)
- On-the-fly column specification
  - 5 or more columns
- On-the-fly reporting statistic specification
  - 4 or more statistic
- Number of Customers ~ 5-10 million range

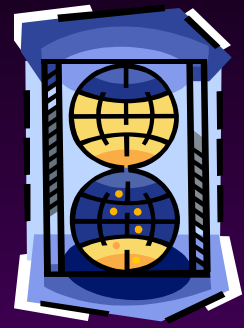


Attempted Solution 1

- Set up a dynamic matrix report using Oracle Reports.
  - Smart functions in each cell to calculate statistics
- Does not work because:
  - Each statistic requires overall filter criteria for the report.
  - For this report - 200 rows x 5 columns x 4 statistics = 4000 independent queries
  - Performance would be unacceptable.



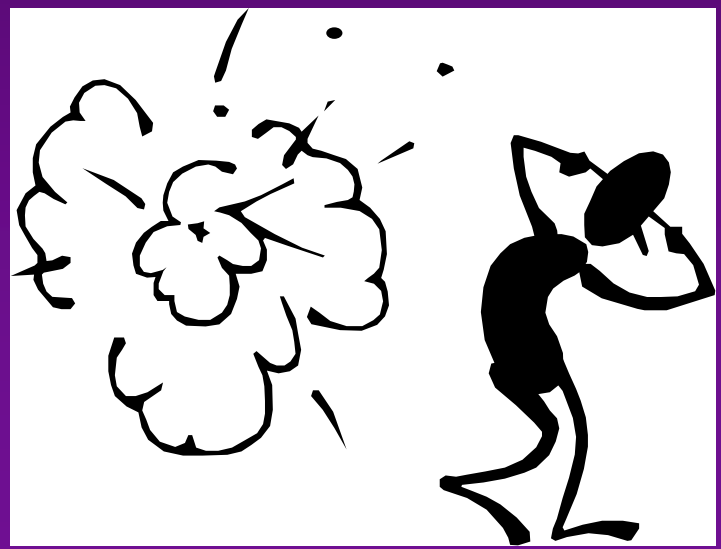
Attempted Solution 2



- Global temporary tables used
  - Good for building single use session-specific temporary tables.
  - CREATE TABLE command used with tables flagged as global temporary tables
- Actually used on a project to create a report with only 7 pre-defined rows
- To speed performance somewhat:
  - Report level and raw level filters applied first to populate 7 independent global temporary tables
  - Approach not scalable for complex report since 400 global temporary tables would be needed and up to 1500 queries.

DULCIAN INC.™

Part 2.
The solution that worked



DULCIAN INC.™

Specifications

- Simplify report structure
- Move report logic out of reporting tools into procedural code
- Create a complete image of final report
  - Including breaks and break values
- Need an entirely different architecture to leverage new SQL and PL/SQL additions

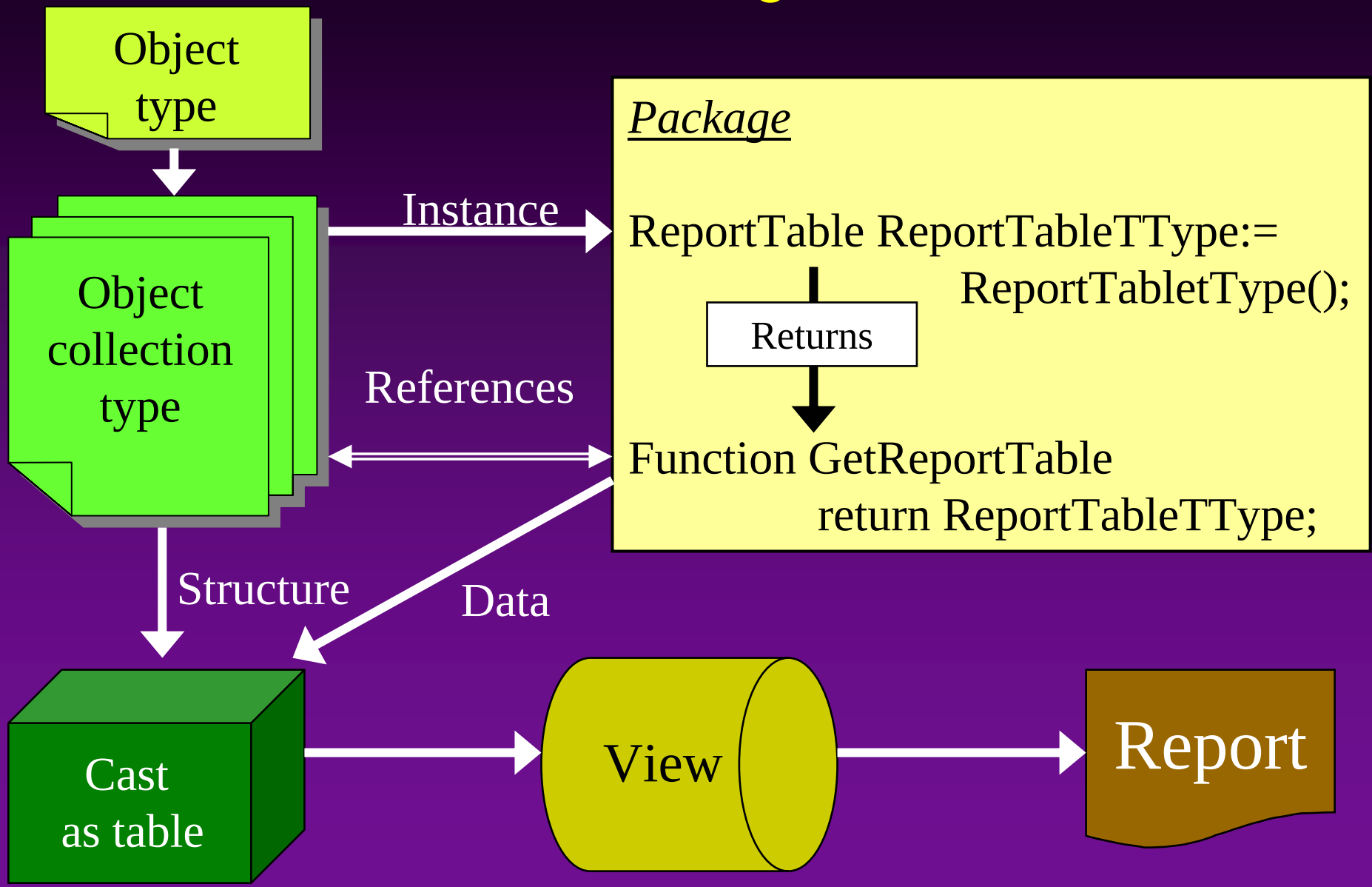
Required results

```
ID Number,
    ID_RFK Number,
    Region_OID number,
    Dept_OID number,
    CustmrCount_NR
  number,
    Break1_TX
  Varchar2(200),
    …
    Break10_TX
  Varchar2(200),
    Col1_TX
  Varchar2(200),
    …
    Col10_TX
  Varchar2(200),
    Level_NR Number,
    Order_NR Number,
    Populated_YN
  varchar2(1)
```

Items in this code are identified as follows:

- ID = System-generated ID
- RFK = Recursive foreign key link to track what rows roll up to what other rows
- Breaks 1-10 = Descriptive row text
- Columns 1-10 = CHR(10) delimited list of statistics values for the report
- Level = Row level in recursive hierarchy
- Order = Number of the row in the report
- Populated_YN = Used in processing for first detail then aggregated upward to build the report

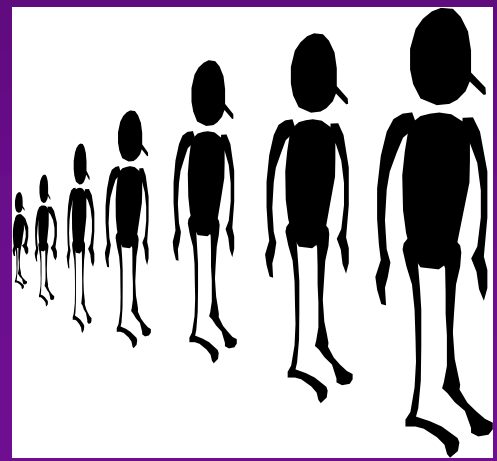
Logical structure



Step 1: Object type

- Report created as an object collection
- Necessary to create an object type first

```
CREATE OR REPLACE
type reporttableotype
as object
     (ID Number,
      ID_RFK Number, …)
```



Step 2: Object collection

- Object collection type must be built based on created object type:

```
CREATE OR REPLACE
type reporttablettype
as table of ReportTableOType;
```

- IMPORTANT: Once an object collection type has been created – cannot modify object type.
  - To modify
    - drop object collection type (invalidates dependent PL/SQL code!)
    - modify structural object type
    - recreate object collection type
    - recompile invalidated dependants

DULCIAN INC.

Step 3: Instantiated variable

- The “report table” is an instantiated package variable of type ReportTableType created using the following code:

```
ReportTable ReportTableTType:=
                              ReportTableTType();
```

- Definition of variable is placed in the package specification to make it accessible to other PL/SQL code

Step 4: GET-function

- Once the report “table” is populated, a function is created in the package to return the object collection and create the appropriate view.

```
FUNCTION GetReportTable
    RETURN ReportTableTType IS
BEGIN
  RETURN ReportTable;
END GetReportTable;
```

DULCIAN INC.™

Step 5: View

- Code to create view:

```
CREATE OR REPLACE VIEW v_reporttable (
    id, id_rfk, … )
AS
select r.*
from table(
         cast (OrgUnitReprt.GetReportTable()
                as ReportTableTType)
           ) r
```

Activities: Cleaning the report table

- Use DELETE method:
  - Object collection is a packaged variable – session-level resource.
  - It has to be cleaned BEFORE report data is collected.

```
ReportTable.delete;
```

Activities: Inserting into Report table

- Use the EXTEND command to create a new row (similar to an INSERT statement)

```
ReportTable.extend;
ReportTable(ReportTable.last):=
            ReportTableOType(V_ID,
                             V_ID_RFK,
                             …)
```

- When this is complete, it is possible to use SELECT * from view (`v_reporttable`) to see coding results

Limitations and Cautions

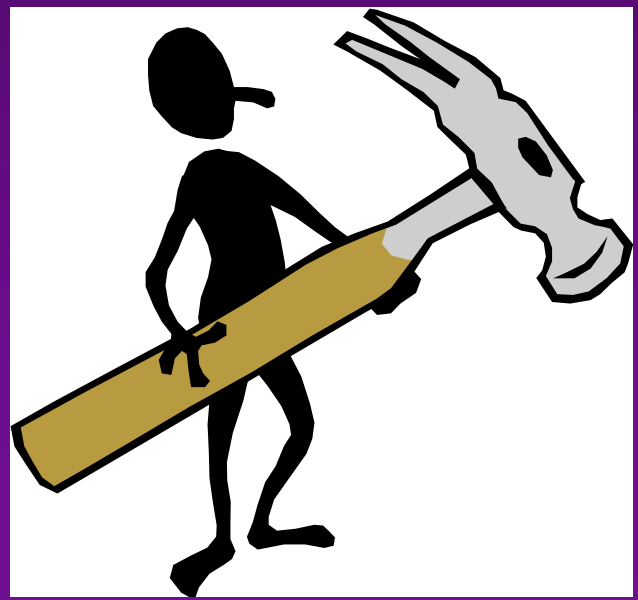
- CONNECT_BY commands do not work from these tables.
- Casting an object collection to a table in a loop requires a lot of time. (1/100th of a second).
  - Not good for processing 1million customers
- Looping should be done without casting.

Architectural features

- Report architecture is quite robust.
- Object collections can be treated like normal tables.
- Cursors loop through collections faster than a PL/SQL table
- Deploying the report table through the view is very useful.
  - Easy maintenance - changes of the algorithm are made in the package only
  - Portability - view can be used by any reporting utility

DULCIAN INC.™

Part 3.
Implementation



DULCIAN INC.™

Step 1: Prepare statistics

- Single query used to walk through all customers and update appropriate statistics based on values associated with the customer.
- Individual statistics placed into a PL/SQL table using a simple hash function to concatenate the row, column, statistic number.
- Allowed easy insert and retrieval of statistic values.

Step 2: Collect statistics

- Copy information from detail rows of report table to break columns in report
- Uses only information stored in report table and statistics.
- For complex statistics (e.g. truncated averages), RANK function used to calculate statistic in each cell where required.
- Code available on Dulcian (www.dulcian.com) and NYOUG (www.nyoug.org) websites.

DULCIAN INC.

Results (1)

- Report runs very quickly
  - Initial setup + report table population = .2-.3 seconds
  - Copying statistics to report + calculation of aggregate rollups = .1-.2 seconds
  - Approximately 10,000 customers/second can be processed depending upon machine power
- Solution works as long as the number of customers processed in any report is in the tens of thousands.

DULCIAN INC.

Results (2)

- Report's modular structure allows for easy modifications if needed.
- Performance varies little regardless of the number of statistics selected.
- Code can be reused among reports to speed creation of additional reports.
- 3 reports constructed this way support all managerial reporting requirements of a large government system.

Share your Knowledge: Call for Articles for the SELECT Journal

- Help contribute your knowledge to the larger Oracle community:
  - Make the SELECT Journal an even more valuable resource.
  - Articles wanted on topics of interest to the Oracle community.
  - Sign up to be a reviewer of articles.
- Submit articles, questions, … to select@ioug.org.

Contact Information

- Dr. Paul Dorsey – paul_dorsey@dulcian.com
- Dulcian website - www.dulcian.com