**Regular Expressions in Oracle**

[Square brackets] – valid options for single characters

[A-Z] – range of characters

{10} – count

(AB|CD) – valid options for groups of characters

\@ - look for a specific character

. (dot) – any character

+ – count of one or more

[ ]? – optional

Example for postal codes:

[A-Z][0-9][A-Z][0-9][A-Z][0-9]

Shorter:

([A-Z][0-9]){3}

Email: bob123@xyz.com

.+ \@ .+ \. .+

Can use regex in statements like SELECT, etc. by using REGEXP\_LIKE operator (note it’s regex**p**, not regex). Example: SELECT \* FROM test WHERE NOT REGEXP\_LIKE(prov, ‘(AB|BC|MB)’);. This will

**Hierarchical Query (for self-joins)**

Structure: Bob

/ \

Ed Mary

/ \ \

Tim Sally Robert

Output in Oracle:

* Bob
  + Ed
    - Tim
    - Sally
  + Mary
    - Robert

select lpad(' ',3\*level)||name from payroll

start with sin=1

connect by prior sin=manager\_sin

The above command will do a hierarchical query. LPAD adds padding to the left. Here, it’s defined as a single space (‘ ‘) multiplied by 3 \* the “level” of the query output. LEVEL is a keyword in Oracle.

select ' ' as "ID", rpad(lpad(' ', 3\*level)||id||' '||name, 25) as "Name",

parent\_id as "Parent ID",

quantity\_in\_parent as "Quantity in Parent"

from Items

start with id=1

connect by prior id=parent\_id

Above example – outputs a hierarchical structure with ID numbers concatenated to the name

PIVOT – makes a pivot table, like in Excel

Simple pivot table:

with datax as (select deptno, job, sal from emp)

select \* from datax

pivot

(sum(sal) for deptno in (10,20,30))

Making reports:

TTITLE – Top title

BTITLE – Bottom title

SPOOL will only work with file types that can be opened in NotePad

**SDLC – System Development Life Cycle**

* Structured way to deliver systems on time, on budget, with desired features

Time

/ \

Cost ---- Features

* ↑Features: ↑Time/Cost
* ↓Cost: ↓Features, ↑Time
* ↓Time: ↓Features, ↑Cost

SDLC Parts:

1. Feasibility Study – planning – determine if you can do it and if it has benefits
   1. Look at 3 risks:
      1. People – do they have skills needed and are they available?
      2. Technology – look at disk needed, network speeds, CPU speeds, memory needed, upgrades to operating systems, etc…
      3. Money – determine costs and benefits
         1. Evaluate NPU, IRR, and paybacks (how quickly return on investment arrives)
2. Analysis – talk to everyone to get exact requirements
   1. Talk to managers, customers, technical staff, users, suppliers
   2. Gather information about reports needed, security needed, input forms needed, interfaces needed, calculations needed, processing speeds expected
   3. Ways to gather information:
      1. Interviews
      2. Surveys
      3. Observation (Job Analysis)
         1. Observe
         2. Perform job yourself
      4. Document study
3. Design – draw diagrams to explain system
   1. Major diagrams
      1. Table relationships, class diagrams, flowchart, etc…
      2. Chen diagram, “Michael Jackson” diagram, “Yourdon” diagram, action chart, state chart, “Warnier-Orr” diagram, data flow diagram
      3. HIPO (pronounced hype-oh)

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Rate | 1. Read rates 2. Read person 3. Add time 4. If hours > 40 then overtime 5. Print checks 6. GoTo 2 | Cheques |
| Workers |  |
| Time\_Cards |  |

1. Implementation – built, test and install system
   1. Build
      1. Top down (traditional)
         1. Design, no functional code (like working buttons), no calculations, etc. Like making art: broad sketch, then refine
      2. Bottom up (object oriented)
         1. Build functional classes, build screen and reports
   2. Test (i – iii done by technical staff)
      1. Unit – test each program
      2. Integration – programs fit together
      3. System – check interfaces, security, processing speeds…
      4. Acceptance – (by users)
         1. Alpha – use fake data
         2. Beta – use real data at one location
   3. Install. Install options:
      1. Cutover – remove old, put in new
      2. Parallel – put in new, but keep old for a month – hard; users usually prefer to use old system; requires tracking to see if anyone is using new system
      3. Phased – install a piece of the new system at a time

In-class Exercise: Monorail system vs warehouse system

|  |  |  |
| --- | --- | --- |
| Area | Same | Different |
| Software | Code needs to:   * Follow circuitous route * Make decisions at branch routes * Avoid collisions | Monorails don’t need to find a starting point, as rails are circuitous. The warehouse trollies would need start points |
| Database | * Locations in warehouse/monorail stops/etc | Warehouse database would have product data; monorail database would just have stop locations |
| Hardware | * Path sensors * Motion sensor * Radio communications | Monorail follows a rail path; warehouse trollies follow special paint paths |

On the board:

|  |  |
| --- | --- |
| Same | Different |
| * Database of routes * Code to read routes/avoid same routes * TCP/IP communication to other equipment * Code to read sensors * Code to control motors * Screens to display jobs * Reports to show jobs done * Programs to enter routes | * Calculations for stopping distance * Looping code for train * Delays at train stops * Max load size different for each system * Program run times * Math symbol to control motors * Override actions |

**5 Types of Programs in Oracle (PL SQL)**

PLSQL is based on Algol, from 1968

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Types** | **Commands Allowed** | **Parameters** | | | | **Command to Start** |
| **Input** | | **Output** | |
| **Min** | **Max** | **Min** | **Max** |
| Script | All | 0 | Infinite | 0 | 0 | START |
| Procedure | All | 0 | Infinite | 0 | Infinite | EXEC |
| Function | SELECT | 0 | Infinite | 1 | 1 | SELECT UPPER (name) … WHERE UPPER (name) = ‘…’  X:=UPPER(name); |
| Trigger | SELECT, INSERT, UPDATE, DELETE | 0 | 0 | 0 | 0 | Runs automatically when a table is changed |
| Package (like a .zip with related functions and procedures | All | 0 | 0 | 0 | 0 | EXEC |

This script will allow user to input parameters for statements:

--VERIFY shows SQL commands running in-between prompts

SET VERIFY OFF

-- These ACCEPT statements only accept NUMBER, CHAR, or DATE

ACCEPT min\_price NUMBER PROMPT 'Enter minimum price: '

--Create a list that will show every size available:

SELECT DISTINCT inv\_size FROM inventory ORDER BY inv\_size;

ACCEPT size CHAR PROMPT 'Enter size to show: '

--Create a list of all available colours in that size:

SELECT DISTINCT color FROM inventory WHERE UPPER(inv\_size) = UPPER('&size') ORDER BY color;

ACCEPT colour CHAR PROMPT 'Enter colour to show: '

SELECT \*

FROM inventory

WHERE inv\_price >= &min\_price

AND UPPER(inv\_size) = UPPER('&size')

AND UPPER(color) = UPPER('&colour');

SET VERIFY ON

Heading