Stored Procedures

Key Benefits

- Encapsulate complex SQL logic into reusable units
- Handle data validation and batch operations efficiently
- Callable by applications or other database procedures
- Improved security through parameterised execution

Common Use Cases

Data migration scripts, complex reporting queries, batch processing operations, and multi-table transaction management.

Oracle PL/SQL Procedure Example

```
CREATE OR REPLACE PROCEDURE update_employee_salary(
 p_employee_id IN NUMBER,
 p_salary_increase IN NUMBER
)
AS
BEGIN
 UPDATE employees
 SET salary = salary + p_salary_increase,
 last_modified = SYSDATE
 WHERE employee_id = p_employee_id;

 COMMIT;

 DBMS_OUTPUT.PUT_LINE('Salary updated successfully');
```

Database Triggers: Automated Event Responses

Key Benefits

- Automate actions in response to DML events (INSERT, UPDATE, DELETE)
- Enforce complex business rules and data integrity
- Facilitate auditing and logging changes to tables
- Maintain derived data and synchronize related tables
- Provide transparent execution without requiring application intervention

Common Trigger Types

- **BEFORE Triggers:** Execute prior to the DML event. Useful for data validation or manipulating data before it's stored.
- **AFTER Triggers:** Execute after the DML event. Ideal for auditing, logging, or performing actions based on the final state of the data.
- **INSTEAD OF Triggers:** Used on views that are not otherwise modifiable. They convert DML operations on the view into corresponding DML operations on the underlying base tables.

Oracle PL/SQL Trigger Example: Employee Salary Update

```
CREATE OR REPLACE TRIGGER trg_update_salary
BEFORE UPDATE OF salary ON employees
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE(
    'Salary changed from ' || :OLD.salary ||
    ' to ' || :NEW.salary
  );
END;
/
```