ER Model - Entity Relationship Model/Diagrams & Translation

Graphical representation of the data requirements for a database

Entities (rectangle)

* Things capable of an independent existence that can be uniquely identified
* Requires a separate table
* Every entity (unless it is a weak entity) must have a PK

Weak Entity (thick rectangle)

* Entity that cannot be uniquely identified by its attributes alone
* Must use a foreign key in conjunction with its attributes to create a PK, and the foreign key is typically a primary key of an entity it is related to
* Requires a separate table

Attributes (oval)

* Multivalued attributes require a separate table (example: an employee can have more than 1 phone number)
* Entities and relationships can both have attributes

Relationships (rhombus)

* Many-to-Many relationships require a separate table
* Attributes of a One-to-One or One-to-Many relationship can be stored at either end entity of the relationship

Is-a Relationship (triangle)

* http://stackoverflow.com/questions/18992653/entity-relationship-diagram-how-does-the-is-a-relationship-translate-into-table

Business Rules (Cardinality)

* At least 1
* At most 1
* Exactly 1

Normalization

Process of eliminating redundancy in a database by splitting up tables

Redundancies / Anomalies

* Things that go wrong
* Annoyances
* Re-typed data entries at insertion

Prime attribute – an attribute that is a PK or a subset of a PK

Functional Dependencies (FD)

* Relationship in which entity attributes can retrieve the dependent attributes
* Represented by an arrow to dependent attributes

P = prime

NP = non-prime

Steps

1. Identify CK’s
2. Identify prime attributes
3. 1NF
   1. For our purposes this will always be compliant
4. 2NF Violation
   1. Subset of CK 🡪 NP
5. 3NF Compliance
   1. LHS is CK (can be subset) or RHS is P
6. BCNF Compliance (Boyce Codd)
   1. LHS is CK

3NF Violation

* NP 🡪 NP

Triggers

Named PL/SQL units that are stored in the database and executed (fired) in response to a specified event associated with either a table, a view, a schema, or the database

Triggers can fire:

* Before/After the triggering statement executes
* Before/After each row that the triggering statement affects

Triggers can be **enabled** or **disabled**

FOR EACH ROW

* Trigger fires once for each row of the table that is affected by the triggering statement
* Absence of this option indicates that the trigger fires only once for each applicable statement

WHEN

* The expression in the clause is evaluated for each row that the trigger affects
* The trigger body executes on behalf of a given row IFF the expression evaluates to true

Accessing Column Values

* Within a trigger body of a row trigger, the PL/SQL code and SQL statements have access to the old and new column values of the current row
* The new column values are referenced using the NEW qualifier before the column name
  + :NEW.salary
* The old column values are referenced using the OLD qualifier before the column name
  + :OLD.salary

SET SERVEROUTPUT ON

CREATE OR REPLACE TRIGGER newEmpAddNewHire

AFTER INSERT ON empbb02

FOR EACH ROW

DECLARE

BEGIN

INSERT INTO newhires VALUES (:NEW.empno, :NEW.ename, :NEW.hiredate);

DBMS\_OUTPUT.PUT\_LINE('1 row added to newhires table.');

END;

/

Object-Relational Database Management System (ORDBMS)

Creating objects

The **CREATE TYPE** statement specifies the name of the type and its attributes, methods, and other properties

* CREATE TYPE Point AS OBJECT

PRAGMA RESTRICT\_REFERENCES(<Function Name>, <Assertion>)

* Asserts that a user-defined subprogram does not read or write database tables or package variables
* RNDB – Asserts that the subprogram reads no database state
* WNDS – Asserts that the subprogram writes no database state
* RNPS – Asserts that the subprogram reads no package state
* WNPS – Asserts that the subprogram writes no package state

CREATE TYPE address\_type AS OBJECT (

street VARCHAR2(20),

city VARCHAR2(14),

state VARCHAR2(14),

zip NUMBER(5),

MEMBER FUNCTION StreetName RETURN VARCHAR2);

/

The **CREATE TYPE BODY** statement contains the code for the methods that implement the type

* CREATE OR REPLACE TYPE BODY Point AS

CREATE OR REPLACE TYPE BODY address\_type AS

MEMBER FUNCTION streetName RETURN VARCHAR2 IS

st emped.addr.street%type;

BEGIN

SELECT e.addr.street INTO st FROM emped e WHERE e.addr.street = SELF.street;

RETURN SUBSTR(st,6,20);

END;

END;

/

SHOW ERRORS

BLOB and CLOB

Binary Large Object and Character Large Object

PL/SQL predefined packages

* DBMS\_LOB
* DBMS\_LOCK
* DBMS\_OUTPUT
* DBMS\_RANDOM

For use with PL\_SQL, there is a predefined package name DBMS\_LOB  
that is a pretty large package for use with large objects.  The code in  
load\_file.proc calls several of its methods.  Here is a list of some of  
those methods:  
  DBMS\_LOB.substring(blob, amount, offset) where amount is how many  
bytes you want to use  
  DBMS\_OUTPUT.write........  
  DBMS\_LOB.read........  
  DBMS\_LOB.getLength...  
  DBMS\_LOB.erase    same parameters as substring  
  DBMS\_LOB.fileopen(filename)  
  DBMS\_LOB.close(filename)  
  DBMS\_LOB.loadfromfile(blobLoc,filePtr,size)  
  DBMD\_LOB.isopen.....  
  DBMS\_LOB.instr    index of string  
There are other packages, e.g. DBMS\_LOCK, which only contains a sleep method.  
The most familiar is DBMS\_OUTPUT which contains PUT\_LINE.  
If you need random numbers inside a PL/SQL block, there is  
  DBMS\_RANDOM which contains random, seed, etc.  
  
These packages (about a dozen of them) make PL/SQL a more useful language  
for defining stored functions and procedures.

Database Admin Stuff

Predefined Roles

* Connect
* Resource
* DBA

GRANT SELECT ON empbb02 TO PUBLIC

REVOKE SELECT ON empbb02 FROM PUBLIC

ALTER SYSTEM KILL SESSION 'sid,serial#'

GRANT EXECUTE ON FUNCTION foo TO st67,st99  
  
GRANT ALL PRIVILEGES ON empbb02 TO st100

CREATE USER st11 identified by cs514 default tablespace users temporary tablespace temp quota 5m on users;  
GRANT student\_role to st11;  
ALTER USER st11 default role student\_role;

CREATE ROLE student\_role;  
GRANT create session, create table,create view,create sequence,create type,  
create any index,create synonym,create public synonym,create procedure,  
create trigger TO student\_role;

SELECT \* FROM session\_privs