pears.DuplicateVacancyAndPlacement

Generated schema reference. Regenerate this page from the SQL unload; keep hand-maintained business notes in the narrative namespace.

Original SQL

```
CREATE FUNCTION "pears"."DuplicateVacancyAndPlacement"(
  /* Application Maintained Function / Procedure - DO NOT EDIT*/
  IN @OldVacancyID CHAR(20),IN @NewVacancyID CHAR(20),IN @NewEmploymentID
CHAR(20),IN @OldPlacementID CHAR(20) DEFAULT NULL,IN @NewPlacementID
CHAR(20) DEFAULT NULL,IN @CloseOffOld CHAR(1) DEFAULT 'N' )
RETURNS SMALLINT
BEGIN
  DECLARE @VacancyAllowed CHAR(250);
  DECLARE @OldCandEmploymentID CHAR(20);
  DECLARE @NewCandEmploymentID CHAR(20);
  DECLARE @NewCandEmploymentCompanyID CHAR(20);
  DECLARE @ExplainStringFrom CHAR(250);
  DECLARE @ExplainStringTo CHAR(250);
  DECLARE @MovedPerson CHAR(50);
  --Check new vacancy allowed
  SET @VacancyAllowed = (SELECT "vacancyallowed"(@NewEmploymentID));
  IF @VacancyAllowed <> '' THEN RETURN(1)
  END IF;
  BEGIN atomic
    INSERT INTO "vacancy"(
"VacancyID","DepartmentID","EmploymentID","Status","temp","Position","Contra
ctRef",
"ClientDepartment","NoOfPosts","StartDate","FinishDate","TempDeskID","StaffI
D","SiteName","SiteContact",
"addr1","addr2","addr3","town","county","country","postcode","SitePhoneNumbe
rs","Notes","OtherNotes",
"ErNi","HolidayAllowance","Discount","Salary","Expiry","TheirRef","Currency"
,"PayrollIdentifier",
"TempJobTypeID","InvAddr1","InvAddr2","InvAddr3","InvTown","InvCounty","InvC
ountry","InvPostCode",
"ClassCode","AnalysisCode","EntryDate","Role","CascadeDateTime","WorkMonday"
,"WorkTuesday","WorkWednesday","WorkThursday","WorkFriday","WorkSaturday","W
orkSunday","WorkStartTime","WorkNormalHours" )
      SELECT
@NewVacancyID,"DepartmentID",@NewEmploymentID,"Status","temp","Position","Co
```

```
ntractRef",
"ClientDepartment","NoOfPosts","StartDate","FinishDate","TempDeskID","StaffI
D","SiteName","SiteContact",
"addr1","addr2","addr3","town","county","country","postcode","SitePhoneNumbe
rs","Notes","OtherNotes",
"ErNi","HolidayAllowance","Discount","Salary","Expiry","TheirRef","Currency"
,"PayrollIdentifier",
"TempJobTypeID","InvAddr1","InvAddr2","InvAddr3","InvTown","InvCounty","InvC
ountry","InvPostCode",
        "ClassCode","AnalysisCode",CURRENT
DATE,"Role","CascadeDateTime","WorkMonday","WorkTuesday","WorkWednesday","Wo
rkThursday","WorkFriday","WorkSaturday","WorkSunday","WorkStartTime","WorkNo
rmalHours"
        FROM "vacancy" WHERE "vacancyid" = @OldVacancyID; //NB Vacancy
refcode inserted BY TRIGGER IN Vacancy TABLE
    -- Copy questionnaire
    INSERT INTO "tagvalue"(
"taglocation","tagid","tagchoiceid","id","value","textvalue" )
      SELECT
"taglocation","tagid","tagchoiceid",@NewVacancyID,"value","textvalue" FROM
"tagvalue"
        WHERE "id" = @OldVacancyID AND "taglocation" LIKE 'V%';
    -- Copy requirements
    INSERT INTO "criterion"( "searchlocation",
      "id","critid","sourcelocation","dictionaryid",
"tagid","tagchoiceid","notflag","matchtype","textvalue","value","uppervalue"
,"taganytotal",
      "extravalue","extrauppervalue","extramatchtype" )
      SELECT
"searchlocation",@NewVacancyID,"critid","sourcelocation","dictionaryid",
"tagid","tagchoiceid","notflag","matchtype","textvalue","value","uppervalue"
,"taganytotal",
        "extravalue","extrauppervalue","extramatchtype" FROM "criterion"
        WHERE "id" = @OldVacancyID AND "searchlocation" LIKE 'V%';
    -- Copy rates
    INSERT INTO "TempJobRate"(
"TempJobRateID","VacancyID","TempPayBandID","PayRate","ChargeRate","StartDat
e","EndDate","Grade" )
      SELECT(SELECT
"uniquekey"("TempJobRateID")),@NewVacancyID,"TempPayBandID","PayRate","Charg
eRate","StartDate","EndDate","Grade"
        FROM "TempJobRate" WHERE "vacancyid" = @OldVacancyID;
    -- Copy T & A rules
    INSERT INTO "cardreaderoverridesettings"(
"CardReaderSettingsID","Companyid","defaultshiftlength","MachineID","maxbrea
klength","MaxEarlyness","MaxExtraBreakTime","MaxLateness","maxshiftlength","
MinBeforeExtending","MinBeforeExtendStart","MinBreakLength","VacancyID" )
      SELECT
```

```
"uniquekey"("VacancyID"),"Companyid","defaultshiftlength","MachineID","maxbr
eaklength","MaxEarlyness","MaxExtraBreakTime","MaxLateness","maxshiftlength"
,"MinBeforeExtending","MinBeforeExtendStart","MinBreakLength",@NewVacancyID
FROM "cardreaderoverridesettings"
        WHERE "vacancyid" = @OldVacancyID;
    -- Cascade Rules
    INSERT INTO "CascadeRule"(
"CascadeRuleID","CompanyID","VacancyID","AgencyStoredSelectionID","CascadeHo
urs","HoursFrom","CascadeLevel","CustomFunction","VacancyRule" )
      SELECT
"uniquekey"("CascadeRuleID"),"CompanyID",@NewVacancyID,"AgencyStoredSelectio
nID","CascadeHours","HoursFrom","CascadeLevel","CustomFunction","VacancyRule
"
        FROM "CascadeRule" WHERE "vacancyid" = @OldVacancyID;
    -- PLACEMENT
    IF @OldPlacementID IS NOT NULL AND @NewPlacementID IS NOT NULL THEN
      -- Candidates Employment
      SET @OldCandEmploymentID = (SELECT "EmploymentID" FROM "Placement"
WHERE "PlacementID" = @OldPlacementID);
      SET @NewCandEmploymentID = "uniquekey"(@OldCandEmploymentID);
      SET @NewCandEmploymentCompanyID = (SELECT "CompanyID" FROM
"Employment" WHERE "EmploymentID" = @NewEmploymentID);
      INSERT INTO "Employment"(
"EmploymentID","temp","personid","companyid","startdate","leavedate","noreem
ploy","note","salary","position","department","Concurrent","ExtendedNotes" )
        SELECT
@NewCandEmploymentID,"temp","personid",@NewCandEmploymentCompanyID,"startdat
e","leavedate","noreemploy","note","salary","position","department","Concurr
ent","ExtendedNotes"
          FROM "Employment" WHERE "EmploymentID" = @OldCandEmploymentID;
      -- Placement
      INSERT INTO "placement"(
"placementid","placedate","employmentid","vacancyid","departmentid","staffid
","temp","clientrate","temprate","note","State","RefCode","TheirRef","Contra
ctRef","DaysPerWeek",
"TransferBatch","Currency","TempJobTypeID","ExtendedNotes","ExpenseBenefitSc
hemeID","WorkMonday","WorkTuesday","WorkWednesday","WorkThursday","WorkFrida
y","WorkSaturday","WorkSunday","WorkNormalHours","WorkStartTime","documentte
mplateid" )
        SELECT
@NewPlacementID,"placedate",@NewCandEmploymentID,@NewVacancyID,"departmentid
","staffid","temp","clientrate","temprate","note","State","RefCode","TheirRe
f","ContractRef","DaysPerWeek",
"TransferBatch","Currency","TempJobTypeID","ExtendedNotes","ExpenseBenefitSc
hemeID","WorkMonday","WorkTuesday","WorkWednesday","WorkThursday","WorkFrida
y","WorkSaturday","WorkSunday","WorkNormalHours","WorkStartTime","documentte
mplateid"
          FROM "Placement" WHERE "PlacementID" = @OldPlacementID;
```

```
      -- Placement Rates
      INSERT INTO "TempJobRate"(
"TempJobRateID","PlacementID","TempPayBandID","PayRate","ChargeRate","StartD
ate","EndDate","Grade" )
        SELECT(SELECT
"uniquekey"("TempJobRateID")),@NewPlacementID,"TempPayBandID","PayRate","Cha
rgeRate","StartDate","EndDate","Grade"
          FROM "TempJobRate" WHERE "PlacementID" = @OldPlacementID;
      -- Placement Elements
      INSERT INTO "PlacementElement"(
"PlacementElementTypeID","PlacementID","Value","Volatile" )
        SELECT
"UniqueKey"("PlacementElementTypeID"),"PlacementID","Value","Volatile"
          FROM "PlacementElement" WHERE "PlacementID" = @OldPlacementID;
      -- Descriptive Text
      SET @ExplainStringFrom = (SELECT "string"("Company"."Name",' -
','"Person"."Name",' - ',"Vacancy"."RefCode") FROM "Vacancy" KEY JOIN
"Employment" KEY JOIN("Person","Company") WHERE "Vacancy"."VacancyID" =
@OldVacancyID);
      SET @ExplainStringTo = (SELECT "string"("Company"."Name",' -
','"Person"."Name",' - ',"Vacancy"."RefCode") FROM "Vacancy" KEY JOIN
"Employment" KEY JOIN("Person","Company") WHERE "Vacancy"."VacancyID" =
@NewVacancyID);
      SET @MovedPerson = (SELECT "Person"."Name" FROM "Person" KEY JOIN
"Employment" KEY JOIN "Placement" WHERE "PlacementID" = @OldPlacementID);
      -- Close off old Placement
      IF @CloseOffOld = 'Y' THEN
        UPDATE "Employment" SET "LeaveDate" = "StartDate","ExtendedNotes" =
"trim"("string"('Employment cancelled, transferred to
',@ExplainStringTo,"char"(13),"char"(13),"ExtendedNotes"))
          WHERE "EmploymentID" = @OldCandEmploymentID;
        UPDATE "Placement" SET "ExtendedNotes" = "trim"("string"('Placement
cancelled, transferred to
',@ExplainStringTo,"char"(13),"char"(13),"ExtendedNotes"))
          WHERE "PlacementID" = @OldPlacementID
      END IF;
      -- Add explanatory text to new Employment & Placement
      UPDATE "Employment" SET "ExtendedNotes" = "trim"("string"('Employment
transfered from ',@ExplainStringFrom,"char"(13),"char"(13),"ExtendedNotes"))
        WHERE "EmploymentID" = @NewCandEmploymentID;
      UPDATE "Placement" SET "ExtendedNotes" = "trim"("string"('Placement
transfered from ',@ExplainStringFrom,"char"(13),"char"(13),"ExtendedNotes"))
        WHERE "PlacementID" = @NewPlacementID;
      -- Update Audit Trail
      CALL "AuditLog"('VACANCY',@OldVacancyID,'Placement Cancelled and
moved',"string"(@ExplainStringFrom,' - ',@MovedPerson),@ExplainStringTo);
      CALL "AuditLog"('VACANCY',@NewVacancyID,'Copied
Vacancy',@ExplainStringFrom,@ExplainStringTo);
```

```
      CALL "AuditLog"('PLACEMENT',@OldPlacementID,'Placement Cancelled and
moved',@ExplainStringFrom,@ExplainStringTo);
      CALL "AuditLog"('PLACEMENT',@NewPlacementID,'Moved
Placement',@ExplainStringFrom,@ExplainStringTo)
    // END OF placement SECTION
    END IF
  END; // END OF atomic statment
  RETURN(0)
exception
  WHEN others THEN RETURN(2)
END
```