pears.vacancy

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

Description

Header record for Vacancies.

Columns

Column	Type	Null	Default	Comment
vacancyid	char(20)	NOT NULL		
departmentid	char(2)	NOT NULL		
employmentid	char(20)	NOT NULL		
staffid	char(20)	NULL		
expiry	date	NULL		
entrydate	date	NULL		
status	char(1)	NULL		
startdate	date	NULL		
ontargetearn	numeric(12,2)	NULL		
salary	numeric(12,2)	NULL		
clientrate	numeric(12,2)	NULL		
temprate	numeric(12,2)	NULL		
temp	smallint	NOT NULL	0	
position	char(50)	NULL		
notes	long varchar	NULL		
othernotes	long varchar	NULL		
whynotfilled	char(50)	NULL		
classcode	char(1)	NULL		
refcode	char(20)	NULL		
noofposts	integer	NULL	0	
ErNI	double	NULL		
HolidayAllowance	double	NULL		
Discount	double	NULL		
FinishDate	date	NULL		
TempDeskID	char(20)	NULL		
TheirRef	char(50)	NULL		

Column	Type	Null	Default	Comment
ContractRef	char(20)	NULL		
PayrollIdentifier	char(1)	NULL		For external payroll link
AnalysisCode	char(20)	NULL		
clientdepartment	char(30)	NULL		
TransferBatch	integer	NULL		
ExtNumber	integer	NULL		
Currency	char(3)	NULL		
RecipientID	char(20)	NULL		
PriceListID	char(20)	NULL		
Funder1	char(12)	NULL		
Funder2	char(12)	NULL		
Funder3	char(12)	NULL		
Role	char(35)	NULL		Allows autogeneration of position for recipient based vacancies
Funder1Limit	double	NULL		
Funder2Limit	double	NULL		
Funder3Limit	double	NULL		
SiteName	char(50)	NULL		
SiteContact	char(100)	NULL		
addr1	char(40)	NULL		
addr2	char(40)	NULL		
addr3	char(40)	NULL		
town	char(30)	NULL		
county	char(30)	NULL		
country	char(30)	NULL		
postcode	char(20)	NULL		
SitePhoneNumbers	char(100)	NULL		
TempJobTypeID	char(20)	NULL		
Invaddr1	char(40)	NULL		
Invaddr2	char(40)	NULL		
Invaddr3	char(40)	NULL		
Invtown	char(30)	NULL		
Invcounty	char(30)	NULL		
Invcountry	char(30)	NULL		
Invpostcode	char(20)	NULL		
DateBecameFinal	date	NULL		
SiteFax	char(50)	NULL		
SiteEmail	char(250)	NULL		
CascadeDateTime	timestamp	NULL		When next cascade will take place
Cascaded	smallint	NULL	0	
CascadeLevel	smallint	NULL	0	0 if has not yet been cascaded

Column	Type	Null	Default	Comment
WhenEntered	timestamp	NULL	current timestamp	
SendShiftsToPayroll	smallint	NULL	0	
WorkMonday	tinyint	NULL		
WorkTuesday	tinyint	NULL		
WorkWednesday	tinyint	NULL		
WorkThursday	tinyint	NULL		
WorkFriday	tinyint	NULL		
WorkSaturday	tinyint	NULL		
WorkSunday	tinyint	NULL		
WorkNormalHours	double	NULL		
WorkStartTime	time	NULL		
documenttemplateid	char(12)	NULL		
PDFInvIncTS	tinyint	NULL	0	
lastcontactevent	timestamp	NULL		Added V2.2.2.13 Will be null for existing vacancies
InvoiceEmail	char(250)	NULL		
Funder1Ref	char(20)	NULL		
Funder2Ref	char(20)	NULL		
Funder3Ref	char(20)	NULL		
AWRTempJobTypeID	char(20)	NULL		No ref. integ. to avoid breaking existing key join code
AWRFixedNI	double	NULL		%
AWRFixedWTR	double	NULL		%
ExtraHols	double	NULL		Needs Calc version > 3.0
QuestionnaireUpdated	timestamp	NULL		Used to update requirements
BlockETimesheets	tinyint	NULL	0	
extranotes	long varchar	NULL		
OriginID	char(20)	NULL		This is the new Source
NotTaxable	smallint	NULL	0	Flag to override client public sector
salaryto	numeric(12,2)	NULL		
InvAttnOf	char(100)	NULL		
ComplianceDomainID	char(20)	NULL		
WarningMessage	long varchar	NULL		

Primary Key

- vacancyid

Foreign Keys

Constraint	Columns	References	Delete/update action
department	departmentid	pears.Department (departmentid)	NOT NULL;
employment	employmentid	pears.employment (employmentid)	NOT NULL;
staff	staffid	pears.staff (staffid)	
vacancyclass	classcode	pears.vacancyclass (classcode)	
TempDesk	TempDeskID	pears.tempdesk (tempdeskid)	
person	RecipientID	pears.Person (personid)	ON DELETE SET NULL
TempPriceList	PriceListID	pears.TempPriceList (TempPriceListID)	ON DELETE SET NULL
TempJobType	TempJobTypeID	pears.TempJobType (TempJobTypeID)	ON DELETE SET NULL
origin	OriginID	pears.Origin (OriginID)	
ComplianceDomain	ComplianceDomainID	pears.ComplianceDomain (ComplianceDomainID)	ON DELETE SET NULL
iqacdocumenttemplate	documenttemplateid	pears.IQacDocumentTemplate (DocumentTemplateID)	ON DELETE SET NULL

Referenced By

Table	Constraint	Columns	Referenced columns
pears.AWRJobMaster	Vacancy	VacancyID	vacancyid
pears.AWRvacancy	vacancy	vacancyid	vacancyid
pears.BroadbeanCandidate	vacancy	vacancyid	vacancyid
pears.BroadbeanChannels	vacancy	vacancyid	vacancyid
pears.BroadbeanVacancy	vacancy	vacancyid	vacancyid
pears.CardReaderOverrideSettings	vacancy	VacancyID	vacancyid
pears.CascadedVacancy	Vacancy	VacancyID	vacancyid
pears.CascadeRule	Vacancy	VacancyID	vacancyid
pears.contactevent	vacancy	vacancyid	vacancyid
pears.diary	vacancy	vacancyid	vacancyid
pears.DocPackValidation	Vacancy	VacancyID	vacancyid
pears.MasterRoster	Vacancy	VacancyID	vacancyid
pears.Placement	vacancy	vacancyid	vacancyid
pears.progress	vacancy	vacancyid	vacancyid
pears.StaffHistory	Vacancy	VacancyID	vacancyid
pears.StatusHistory	Vacancy	VacancyID	vacancyid
pears.TempJobRate	vacancy	VacancyID	vacancyid
pears.TempProvTimeSheet	Vacancy	VacancyID	vacancyid
pears.TempSecAgencyVacRateScheme	Vacancy	VacancyID	vacancyid
pears.TempShift	Vacancy	VacancyID	vacancyid

Table	Constraint	Columns	Referenced columns
pears.TempShiftPlan	Vacancy	VacancyID	vacancyid
pears.TempShiftTemplateAllowed	Vacancy	VacancyID	vacancyid
pears.vacancy_team	vacancy	vacancyid	vacancyid
pears.VacancyEtip	vacancy	vacancyID	vacancyid
pears.VacancyLimit	vacancy	VacancyID	vacancyid
pears.VacancyOverrideRateScheme	Vacancy	VacancyID	vacancyid
pears.VacancyRoleAllocation	vacancy	VacancyID	vacancyid
pears.WithHolds	vacancy	VacancyID	vacancyid

Indexes

Name	Type	Columns	Detail
vacancy_entrydate	Index	entrydate	
vacancy_refcode	Index	refcode	
vacancy_extnumber	Index	ExtNumber	
vacancy_transferbatch	Index	TransferBatch	
vacancy_status	Index	status	
person_lastcontactevent	Index	lastcontactevent	
vacancy_desk_status	Index	TempDeskID, status	
VacancyWhenEntered	Index	WhenEntered	
vacancy_notestext	Text index	notes, othernotes	CONFIGURATION “SYS”.“default_char” IMMEDIATE REFRESH

Triggers

Name	Timing	Event
VacancyFinal	after	update of “Status” order 1
VacancyNoLongerFinal	after	update of “Status” order 2
vacancy_lsmupd	before	update of “TheirRef” order 2
vacancy_insert	before	insert order 1
VacancyAudit	before	update of “employmentid”, “departmentid”,“staffid”,“expiry”,“entrydate”,“status”,“startdate”,“salary”,“position”,“noofposts”,“FinishDate”, “TempDeskID”,“RefCode”,“TheirRef”,“Payrollidentifier”,“othernotes”,“nottaxable”,“extranotes”,“sitename”,“siteemail”,“addr1”,“addr2”,“addr3”,“postcode”,“sitecontact”,“town”,“county”,“country”,“documenttemplateid” order 6
vacancy_insert2	after	insert order 1
vacancy_delete	before	delete order 1
psHealthVacancyUpdate	after	update of “recipientid” order 900
vacancy_UpdateTrim	before	update order 7
InsertStatusVacancy	after	insert order 2
UpdateStatusVacancy	after	update of “status” order 3
VacancyKeyWordsInsert	after	insert order 20
VacancyKeyWordsUpdate	after	update of “refcode”, “position”,“siteemail”,“sitename”,“sitecontact”,“employmentid” order 21

Original SQL

```
-- IQX database structure split by table
-- Source: IQXDatabaseStructure - with comments.sql
-- Table: "pears"."vacancy"
-- Table comment: Header record for Vacancies.
-- Statement count: 59

CREATE TABLE "pears"."vacancy" (
```

```
    "vacancyid"                 CHAR(20) NOT NULL
   ,"departmentid"              CHAR(2) NOT NULL
   ,"employmentid"              CHAR(20) NOT NULL
   ,"staffid"                   CHAR(20) NULL
   ,"expiry"                    DATE NULL
   ,"entrydate"                 DATE NULL
   ,"status"                    CHAR(1) NULL
   ,"startdate"                 DATE NULL
   ,"ontargetearn"              NUMERIC(12,2) NULL
   ,"salary"                    NUMERIC(12,2) NULL
   ,"clientrate"                NUMERIC(12,2) NULL
   ,"temprate"                  NUMERIC(12,2) NULL
   ,"temp"                      SMALLINT NOT NULL DEFAULT 0
   ,"position"                  CHAR(50) NULL
   ,"notes"                     long VARCHAR NULL
   ,"othernotes"                long VARCHAR NULL
   ,"whynotfilled"              CHAR(50) NULL
   ,"classcode"                 CHAR(1) NULL
   ,"refcode"                   CHAR(20) NULL
   ,"noofposts"                 INTEGER NULL DEFAULT 0
   ,"ErNI"                      DOUBLE NULL
   ,"HolidayAllowance"          DOUBLE NULL
   ,"Discount"                  DOUBLE NULL
   ,"FinishDate"                DATE NULL
   ,"TempDeskID"                CHAR(20) NULL
   ,"TheirRef"                  CHAR(50) NULL
   ,"ContractRef"               CHAR(20) NULL
   ,"PayrollIdentifier"         CHAR(1) NULL
   ,"AnalysisCode"              CHAR(20) NULL
   ,"clientdepartment"          CHAR(30) NULL
   ,"TransferBatch"             INTEGER NULL
   ,"ExtNumber"                 INTEGER NULL
   ,"Currency"                  CHAR(3) NULL
   ,"RecipientID"               CHAR(20) NULL
   ,"PriceListID"               CHAR(20) NULL
   ,"Funder1"                   CHAR(12) NULL
   ,"Funder2"                   CHAR(12) NULL
   ,"Funder3"                   CHAR(12) NULL
   ,"Role"                      CHAR(35) NULL
   ,"Funder1Limit"              DOUBLE NULL
   ,"Funder2Limit"              DOUBLE NULL
   ,"Funder3Limit"              DOUBLE NULL
   ,"SiteName"                  CHAR(50) NULL
   ,"SiteContact"               CHAR(100) NULL
   ,"addr1"                     CHAR(40) NULL
   ,"addr2"                     CHAR(40) NULL
   ,"addr3"                     CHAR(40) NULL
   ,"town"                      CHAR(30) NULL
```

```
    ,"county"                       CHAR(30) NULL
    ,"country"                      CHAR(30) NULL
    ,"postcode"                     CHAR(20) NULL
    ,"SitePhoneNumbers"             CHAR(100) NULL
    ,"TempJobTypeID"                CHAR(20) NULL
    ,"Invaddr1"                     CHAR(40) NULL
    ,"Invaddr2"                     CHAR(40) NULL
    ,"Invaddr3"                     CHAR(40) NULL
    ,"Invtown"                      CHAR(30) NULL
    ,"Invcounty"                    CHAR(30) NULL
    ,"Invcountry"                   CHAR(30) NULL
    ,"Invpostcode"                  CHAR(20) NULL
    ,"DateBecameFinal"              DATE NULL
    ,"SiteFax"                      CHAR(50) NULL
    ,"SiteEmail"                    CHAR(250) NULL
    ,"CascadeDateTime"              TIMESTAMP NULL
    ,"Cascaded"                     SMALLINT NULL DEFAULT 0
    ,"CascadeLevel"                 SMALLINT NULL DEFAULT 0
    ,"WhenEntered"                  TIMESTAMP NULL DEFAULT CURRENT
TIMESTAMP
    ,"SendShiftsToPayroll"          SMALLINT NULL DEFAULT 0
    ,"WorkMonday"                   tinyint NULL
    ,"WorkTuesday"                  tinyint NULL
    ,"WorkWednesday"                tinyint NULL
    ,"WorkThursday"                 tinyint NULL
    ,"WorkFriday"                   tinyint NULL
    ,"WorkSaturday"                 tinyint NULL
    ,"WorkSunday"                   tinyint NULL
    ,"WorkNormalHours"              DOUBLE NULL
    ,"WorkStartTime"                TIME NULL
    ,"documenttemplateid"           CHAR(12) NULL
    ,"PDFInvIncTS"                  tinyint NULL DEFAULT 0
    ,"lastcontactevent"             TIMESTAMP NULL
    ,"InvoiceEmail"                 CHAR(250) NULL
    ,"Funder1Ref"                   CHAR(20) NULL
    ,"Funder2Ref"                   CHAR(20) NULL
    ,"Funder3Ref"                   CHAR(20) NULL
    ,"AWRTempJobTypeID"             CHAR(20) NULL
    ,"AWRFixedNI"                   DOUBLE NULL
    ,"AWRFixedWTR"                  DOUBLE NULL
    ,"ExtraHols"                    DOUBLE NULL
    ,"QuestionnaireUpdated"         TIMESTAMP NULL
    ,"BlockETimesheets"             tinyint NULL DEFAULT 0
    ,"extranotes"                   long VARCHAR NULL
    ,"OriginID"                     CHAR(20) NULL
    ,"NotTaxable"                   SMALLINT NULL DEFAULT 0
    ,"salaryto"                     NUMERIC(12,2) NULL
    ,"InvAttnOf"                    CHAR(100) NULL
```

```
   ,"ComplianceDomainID"             CHAR(20) NULL
   ,"WarningMessage"                 long VARCHAR NULL
   ,PRIMARY KEY ("vacancyid" ASC)
   ,CONSTRAINT "TempVacancyMustHaveTempDesk" CHECK("Temp" = 0 OR
"TempDeskID" IS NOT NULL)
)
GO


COMMENT ON COLUMN "pears"."vacancy"."PayrollIdentifier" IS
    'For external payroll link'
GO


COMMENT ON COLUMN "pears"."vacancy"."Role" IS
    'Allows autogeneration of position for recipient based vacancies'
GO


COMMENT ON COLUMN "pears"."vacancy"."CascadeDateTime" IS
    'When next cascade will take place'
GO


COMMENT ON COLUMN "pears"."vacancy"."CascadeLevel" IS
    '0 if has not yet been cascaded'
GO


COMMENT ON COLUMN "pears"."vacancy"."lastcontactevent" IS
    'Added V2.2.2.13 Will be null for existing vacancies'
GO


COMMENT ON COLUMN "pears"."vacancy"."AWRTempJobTypeID" IS
    'No ref. integ. to avoid breaking existing key join code'
GO


COMMENT ON COLUMN "pears"."vacancy"."AWRFixedNI" IS
    '%'
GO


COMMENT ON COLUMN "pears"."vacancy"."AWRFixedWTR" IS
    '%'
GO
```

```
COMMENT ON COLUMN "pears"."vacancy"."ExtraHols" IS
    'Needs Calc version > 3.0'
GO


COMMENT ON COLUMN "pears"."vacancy"."QuestionnaireUpdated" IS
    'Used to update requirements'
GO


COMMENT ON COLUMN "pears"."vacancy"."OriginID" IS
    'This is the new Source'
GO


COMMENT ON COLUMN "pears"."vacancy"."NotTaxable" IS
    'Flag to override client public sector'
GO


COMMENT ON TABLE "pears"."vacancy" IS
    'Header record for Vacancies.'
GO


ALTER TABLE "pears"."vacancy"
    ADD NOT NULL FOREIGN KEY "department" ("departmentid" ASC)
    REFERENCES "pears"."Department" ("departmentid")
GO


ALTER TABLE "pears"."vacancy"
    ADD NOT NULL FOREIGN KEY "employment" ("employmentid" ASC)
    REFERENCES "pears"."employment" ("employmentid")
GO


ALTER TABLE "pears"."vacancy"
    ADD FOREIGN KEY "staff" ("staffid" ASC)
    REFERENCES "pears"."staff" ("staffid")
GO


ALTER TABLE "pears"."vacancy"
    ADD FOREIGN KEY "vacancyclass" ("classcode" ASC)
    REFERENCES "pears"."vacancyclass" ("classcode")
GO
```

```
ALTER TABLE "pears"."vacancy"
    ADD FOREIGN KEY "TempDesk" ("TempDeskID" ASC)
    REFERENCES "pears"."tempdesk" ("tempdeskid")
GO


ALTER TABLE "pears"."vacancy"
    ADD FOREIGN KEY "person" ("RecipientID" ASC)
    REFERENCES "pears"."Person" ("personid")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."vacancy"
    ADD FOREIGN KEY "TempPriceList" ("PriceListID" ASC)
    REFERENCES "pears"."TempPriceList" ("TempPriceListID")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."vacancy"
    ADD FOREIGN KEY "TempJobType" ("TempJobTypeID" ASC)
    REFERENCES "pears"."TempJobType" ("TempJobTypeID")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."vacancy"
    ADD FOREIGN KEY "origin" ("OriginID" ASC)
    REFERENCES "pears"."Origin" ("OriginID")
GO


ALTER TABLE "pears"."vacancy"
    ADD FOREIGN KEY "ComplianceDomain" ("ComplianceDomainID" ASC)
    REFERENCES "pears"."ComplianceDomain" ("ComplianceDomainID")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."vacancy"
    ADD FOREIGN KEY "iqacdocumenttemplate" ("documenttemplateid" ASC)
    REFERENCES "pears"."IQacDocumentTemplate" ("DocumentTemplateID")
    ON DELETE SET NULL
GO
```

```
CREATE INDEX "vacancy_entrydate" ON "pears"."vacancy"
    ( "entrydate" DESC )
GO


CREATE INDEX "vacancy_refcode" ON "pears"."vacancy"
    ( "refcode" )
GO


CREATE INDEX "vacancy_extnumber" ON "pears"."vacancy"
    ( "ExtNumber" )
GO


CREATE INDEX "vacancy_transferbatch" ON "pears"."vacancy"
    ( "TransferBatch" )
GO


CREATE INDEX "vacancy_status" ON "pears"."vacancy"
    ( "status" )
GO


CREATE INDEX "person_lastcontactevent" ON "pears"."vacancy"
    ( "lastcontactevent" DESC )
GO


CREATE INDEX "vacancy_desk_status" ON "pears"."vacancy"
    ( "TempDeskID","status" )
GO


CREATE INDEX "VacancyWhenEntered" ON "pears"."vacancy"
    ( "WhenEntered" )
GO


CREATE TEXT INDEX "vacancy_notestext" ON "pears"."vacancy"
    ( "notes","othernotes" ) CONFIGURATION "SYS"."default_char" IMMEDIATE
REFRESH
GO


CREATE TRIGGER "VacancyFinal" after UPDATE OF "Status"
```

```
ORDER 1 ON "pears"."vacancy"
REFERENCING OLD AS "OldVac" NEW AS "NewVac"
FOR each ROW
WHEN((SELECT FIRST "isnull"("Status"."Final",0) FROM "Status"
  WHERE "type" = 'V' AND "NewVac"."Status" = "Status"."Status") = 1
AND(SELECT FIRST "isnull"("Status"."Final",0) FROM "Status"
  WHERE "type" = 'V' AND "OldVac"."Status" = "Status"."Status") = 0)
BEGIN
  UPDATE "vacancy" SET "Vacancy"."DateBecameFinal" = CURRENT DATE WHERE
"NewVac"."VacancyID" = "VacancyID"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."vacancy"."VacancyFinal" IS
{CREATE TRIGGER VacancyFinal
 after UPDATE OF STATUS
ORDER 1 ON pears.vacancy
REFERENCING OLD AS OldVac NEW AS NewVac
FOR each ROW
WHEN((SELECT FIRST isnull(STATUS.Final,0) FROM STATUS WHERE
  TYPE = 'V' AND NewVac.Status = STATUS.Status) = 1 AND
(SELECT FIRST isnull(STATUS.Final,0) FROM STATUS WHERE
  TYPE = 'V' AND OldVac.Status = STATUS.Status) = 0)
BEGIN
  UPDATE vacancy SET Vacancy.DateBecameFinal = CURRENT DATE WHERE
NewVac.VacancyID = VacancyID
END
}
GO


CREATE TRIGGER "VacancyNoLongerFinal" after UPDATE OF "Status"
ORDER 2 ON "pears"."vacancy"
REFERENCING OLD AS "OldVac" NEW AS "NewVac"
FOR each ROW
WHEN((SELECT FIRST "isnull"("Status"."Final",0) FROM "Status"
  WHERE "type" = 'V' AND "NewVac"."Status" = "Status"."Status") = 0
AND(SELECT FIRST "isnull"("Status"."Final",0) FROM "Status"
  WHERE "type" = 'V' AND "OldVac"."Status" = "Status"."Status") = 1)
BEGIN
  UPDATE "vacancy" SET "Vacancy"."DateBecameFinal" = NULL WHERE "VacancyID"
= "NewVac"."VacancyID"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
```

```
"pears"."vacancy"."VacancyNoLongerFinal" IS
{CREATE TRIGGER VacancyNoLongerFinal
 after UPDATE OF STATUS
ORDER 2 ON pears.vacancy
REFERENCING OLD AS OldVac NEW AS NewVac
FOR each ROW
WHEN((SELECT FIRST isnull(STATUS.Final,0) FROM STATUS WHERE
  TYPE = 'V' AND NewVac.Status = STATUS.Status) = 0 AND
(SELECT FIRST isnull(STATUS.Final,0) FROM STATUS WHERE
  TYPE = 'V' AND OldVac.Status = STATUS.Status) = 1)
BEGIN
  UPDATE vacancy SET Vacancy.DateBecameFinal = NULL WHERE VacancyID =
NewVac.VacancyID
END
}
GO


CREATE TRIGGER "vacancy_lsmupd" BEFORE UPDATE OF "TheirRef"
ORDER 2 ON "pears"."vacancy"
REFERENCING NEW AS "new_vac"
FOR each ROW
WHEN("new_vac"."transferbatch" > 0)
BEGIN
  SET "new_vac"."transferbatch" = 0
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."vacancy"."vacancy_lsmupd" IS
{CREATE TRIGGER vacancy_lsmupd
 BEFORE UPDATE OF TheirRef
ORDER 2 ON pears.vacancy
REFERENCING NEW AS new_vac
FOR each ROW
WHEN(new_vac.transferbatch > 0)
BEGIN
  SET new_vac.transferbatch=0
END
}
GO


CREATE TRIGGER "vacancy_insert" BEFORE INSERT ORDER 1 ON
"pears"."vacancy"
REFERENCING NEW AS "new_vacancy"
FOR each ROW
BEGIN
```

```
  DECLARE "autonumvacs" SMALLINT;
  DECLARE "lastvacnum" INTEGER;
  SET "new_vacancy"."contractref" = "trim"("new_vacancy"."contractref");
  SET "new_vacancy"."refcode" = "trim"("new_vacancy"."refcode");
  SET "new_vacancy"."sitename" = "trim"("new_vacancy"."sitename");
  SET "new_vacancy"."position" = "trim"("new_vacancy"."position");
  SET "new_vacancy"."theirref" = "trim"("new_vacancy"."theirref");
  IF "new_vacancy"."refcode" IS NULL THEN SELECT
"autovacnumber","nextvacnumber" INTO "autonumvacs","lastvacnum" FROM
"params";
    IF "isnull"("autonumvacs",0) <> 0 THEN SET "new_vacancy"."refcode" =
"isnull"("lastvacnum",0)+1;
      UPDATE "params" SET "nextvacnumber" = "isnull"("nextvacnumber",0)+1
    END IF
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."vacancy"."vacancy_insert" IS
{CREATE TRIGGER vacancy_insert
 BEFORE INSERT ORDER 1 ON
pears.vacancy
REFERENCING NEW AS new_vacancy
FOR each ROW
BEGIN
  DECLARE autonumvacs SMALLINT;
  DECLARE lastvacnum INTEGER;
  SET new_vacancy.contractref = TRIM(new_vacancy.contractref);
  SET new_vacancy.refcode = TRIM(new_vacancy.refcode);
  SET new_vacancy.sitename = TRIM(new_vacancy.sitename);
  SET new_vacancy.position = TRIM(new_vacancy.position);
  SET new_vacancy.theirref = TRIM(new_vacancy.theirref);
  IF new_vacancy.refcode IS NULL THEN SELECT autovacnumber,nextvacnumber
INTO autonumvacs,lastvacnum FROM params;
    IF isnull(autonumvacs,0) <> 0 THEN SET
new_vacancy.refcode=isnull(lastvacnum,0)+1;
      UPDATE params SET nextvacnumber = isnull(nextvacnumber,0)+1
    END IF
  END IF
END
}
GO


CREATE TRIGGER "VacancyAudit" BEFORE UPDATE OF "employmentid",
"departmentid","staffid","expiry","entrydate","status","startdate","salary",
"position","noofposts","FinishDate",
```

```
"TempDeskID","RefCode","TheirRef","Payrollidentifier","othernotes","nottaxab
le","extranotes","sitename","siteemail","addr1","addr2","addr3","postcode","
sitecontact","town","county","country","documenttemplateid" ORDER 6 ON
"pears"."vacancy"
REFERENCING OLD AS "old_vac" NEW AS "new_vac"
FOR each ROW
WHEN(EXISTS(SELECT * FROM "AuditItems" WHERE "AreaName" = 'Vacancy' AND
"AuditFlag" = 1))
BEGIN
  DECLARE @AuditList long VARCHAR;
  DECLARE @OldDescrip CHAR(250);
  DECLARE @NewDescrip CHAR(250);
  DECLARE @CompName CHAR(250);
  DECLARE @CRLF CHAR(2);
  SET @CRLF = "char"(10)+"char"(13);
  SELECT "string"(',',"list"("ItemName"),',') INTO @AuditList FROM
"AuditItems" WHERE(("AreaName" = 'Company Contact' AND "AuditFlag" = 1)
OR("AreaName" = 'Vacancy' AND "AuditFlag" = 1));
  SET @CompName = (SELECT "string"("Company"."Name",' -
','"old_vac"."position",'('"old_vac"."RefCode",')') FROM "Company" KEY JOIN
"Employment" WHERE "Employment"."EmploymentID" = "old_vac"."EmploymentID");
  -- Status
  IF "locate"(@AuditList,',Status,') > 0 AND UPDATE("Status") THEN
    SELECT "name" INTO @OldDescrip FROM "status" WHERE "type" = 'V' AND
"status"."status" = "old_vac"."status";
    SELECT "name" INTO @NewDescrip FROM "status" WHERE "type" = 'V' AND
"status"."status" = "new_vac"."status";
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Status Updated
- ',@CompName),"string"("old_vac"."Status",' -
',@OldDescrip),"string"("new_vac"."Status",' - ',@NewDescrip))
  END IF;
  -- Salary
  IF "locate"(@AuditList,',Salary,') > 0 AND UPDATE("Salary") THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Salary Updated
- ',@CompName),"string"("old_vac"."Salary"),"string"("new_vac"."Salary"))
  END IF;
  -- Position
  IF "locate"(@AuditList,',Position,') > 0 AND UPDATE("Position") THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Position
Updated -
',@CompName),"string"("old_vac"."Position"),"string"("new_vac"."Position"))
  END IF;
  -- No of Posts
  IF "locate"(@AuditList,',No Of Posts,') > 0 AND UPDATE("NoOfPosts") THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('No of Posts
Updated -
',@CompName),"string"("old_vac"."NoOfPosts"),"string"("new_vac"."NoOfPosts")
)
```

```
  END IF;
  -- DATES
  IF "locate"(@AuditList,',Dates,') > 0 THEN
    IF UPDATE("entrydate") THEN
      CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Entry Date
Updated -
',@CompName),"string"("old_vac"."entrydate"),"string"("new_vac"."entrydate")
)
    END IF;
    IF UPDATE("startdate") THEN
      CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Start Date
Updated -
',@CompName),"string"("old_vac"."startdate"),"string"("new_vac"."startdate")
)
    END IF;
    IF UPDATE("finishdate") THEN
      CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Finish Date
Updated -
',@CompName),"string"("old_vac"."finishdate"),"string"("new_vac"."finishdate
"))
    END IF;
    IF UPDATE("expiry") THEN
      CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Expiry Date
Updated -
',@CompName),"string"("old_vac"."expiry"),"string"("new_vac"."expiry"))
    END IF END IF;
  -- Consultant
  IF "locate"(@AuditList,',Consultant,') > 0 AND UPDATE("StaffID") THEN
    SELECT "userid" INTO @OldDescrip FROM "staff" WHERE "staffid" =
"old_vac"."StaffID";
    SELECT "userid" INTO @NewDescrip FROM "staff" WHERE "staffid" =
"new_vac"."StaffID";
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Consultant
Updated - ',@CompName),@OldDescrip,@NewDescrip);
    INSERT INTO "staffhistory"(
"staffhistoryid","vacancyid","oldstaff","newstaff","whenentered" ) VALUES(
"uniquekey"("new_vac"."StaffID"),"new_vac"."vacancyid","old_vac"."StaffID","
new_vac"."StaffID",CURRENT TIMESTAMP )
  END IF;
  -- Department
  IF "locate"(@AuditList,',Department,') > 0 AND UPDATE("DepartmentID") THEN
    SELECT "name" INTO @OldDescrip FROM "Department" WHERE "DepartmentID" =
"old_vac"."DepartmentID";
    SELECT "name" INTO @NewDescrip FROM "Department" WHERE "DepartmentID" =
"new_vac"."DepartmentID";
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Department
Updated - ',@CompName),@OldDescrip,@NewDescrip)
  END IF;
```

```
  -- TempDesk
  IF "locate"(@AuditList,',Temp Desk,') > 0 AND UPDATE("TempDeskID") THEN
    SELECT "name" INTO @OldDescrip FROM "TempDesk" WHERE "TempDeskid" =
"old_vac"."TempDeskID";
    SELECT "name" INTO @NewDescrip FROM "TempDesk" WHERE "TempDeskid" =
"new_vac"."TempDeskID";
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('TempDesk
Updated - ',@CompName),@OldDescrip,@NewDescrip)
  END IF;
  -- Our Ref
  IF "locate"(@AuditList,',Our Ref,') > 0 AND UPDATE("RefCode") THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Our Ref
Updated -
',@CompName),"string"("old_vac"."RefCode"),"string"("new_vac"."RefCode"))
  END IF;
  -- Payrollidentifier
  IF "locate"(@AuditList,',Payroll Identifier,') > 0 AND
UPDATE("Payrollidentifier") THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Payroll
Identifier Updated -
',@CompName),"string"("old_vac"."Payrollidentifier"),"string"("new_vac"."Pay
rollidentifier"))
  END IF;
  -- ExtraNotes
  IF "locate"(@AuditList,',Extra Notes,') > 0 AND UPDATE("ExtraNotes") THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Extra Notes
Updated -
',@CompName),"string"("old_vac"."ExtraNotes"),"string"("new_vac"."ExtraNotes
"))
  END IF;
  -- OtherNotes
  IF "locate"(@AuditList,',Job Description,') > 0 AND UPDATE("OtherNotes")
THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Job
Description Ref Updated -
',@CompName),"string"("old_vac"."OtherNotes"),"string"("new_vac"."OtherNotes
"))
  END IF;
  -- Site Details
  IF "locate"(@AuditList,',Site Details,') > 0 AND(UPDATE("country") OR
UPDATE("county") OR UPDATE("town") OR UPDATE("sitecontact")
    OR UPDATE("postcode") OR UPDATE("addr1") OR UPDATE("addr2") OR
UPDATE("addr3") OR UPDATE("siteemail") OR UPDATE("sitename")) THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Site Details
Updated - ',@CompName),
"string"("old_vac"."sitename",@CRLF,"old_vac"."sitecontact",@CRLF,"old_vac".
"addr1",@CRLF,"old_vac"."addr2",@CRLF,"old_vac"."addr3",@CRLF,"old_vac"."tow
n",@CRLF,"old_vac"."county",@CRLF,"old_vac"."country",@CRLF,"old_vac"."postc
```

```
ode",@CRLF,"old_vac"."siteemail"),
"string"("new_vac"."sitename",@CRLF,"new_vac"."sitecontact",@CRLF,"new_vac".
"addr1",@CRLF,"new_vac"."addr2",@CRLF,"new_vac"."addr3",@CRLF,"new_vac"."tow
n",@CRLF,"new_vac"."county",@CRLF,"new_vac"."country",@CRLF,"new_vac"."postc
ode",@CRLF,"new_vac"."siteemail"))
  END IF;
  -- Their Ref
  IF "locate"(@AuditList,',Their Ref,') > 0 AND UPDATE("TheirRef") THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Their Ref
Updated -
',@CompName),"string"("old_vac"."TheirRef"),"string"("new_vac"."TheirRef"))
  END IF;
  IF "locate"(@AuditList,',Override IR35 Public Sector,') > 0 AND
UPDATE("nottaxable") THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid","string"('Override IR35
Public Sector - ',@CompName),"old_vac"."nottaxable","new_vac"."nottaxable")
  END IF;
  -- Contact
  IF "locate"(@AuditList,',Contact,') > 0 AND UPDATE("employmentid") THEN
    CALL "AuditLog"('VACANCY',"old_VAC"."vacancyid",'Contact left company -
vacancies reassigned',
    (SELECT "p"."name" FROM "employment" AS "e" KEY JOIN "person" AS "p"
WHERE "e"."employmentid" = "old_vac"."employmentid"),(SELECT "p"."name" FROM
"employment" AS "e" KEY JOIN "person" AS "p" WHERE "e"."employmentid" =
"new_vac"."employmentid"))
  END IF;
  IF "locate"(@AuditList,',Contact Left Company,') > 0 AND
UPDATE("employmentid") THEN
    CALL "AuditLog"('COMPANY',(SELECT "c"."Companyid" FROM "Company" AS "c"
KEY JOIN "employment" AS "e" WHERE "e"."employmentid" =
"old_VAC"."employmentid"),'Contact left company - Vacancies reassigned',
    (SELECT "p"."name" FROM "employment" AS "e" KEY JOIN "person" AS "p"
WHERE "e"."employmentid" = "old_vac"."employmentid"),(SELECT "p"."name" FROM
"employment" AS "e" KEY JOIN "person" AS "p" WHERE "e"."employmentid" =
"new_vac"."employmentid"))
  END IF;
  --  Override Invoice Layout
  IF "locate"(@AuditList,',Override Invoice Layout,') > 0 AND
UPDATE("documenttemplateid") THEN
    SET @OldDescrip = (SELECT "name" FROM "iqacdocumenttemplate" WHERE
"documenttemplateid" = "old_vac"."documenttemplateid");
    SET @NewDescrip = (SELECT "name" FROM "iqacdocumenttemplate" WHERE
"documenttemplateid" = "new_vac"."documenttemplateid");
    CALL "AuditLog"('VACANCY',"old_vac"."VacancyID","string"('Override
Invoice Layout Updated - ',@CompName),@OldDescrip,@NewDescrip)
  END IF
END
GO
```

```
COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."vacancy"."VacancyAudit" IS
{CREATE TRIGGER VacancyAudit
 BEFORE UPDATE OF
employmentid,departmentid,staffid,expiry,entrydate,STATUS,startdate,salary,P
OSITION,noofposts,FinishDate,
TempDeskID,RefCode,TheirRef,Payrollidentifier,othernotes,nottaxable,extranot
es,sitename,siteemail,addr1,addr2,addr3,postcode,sitecontact,town,county,cou
ntry,documenttemplateid ORDER 6 ON pears.vacancy
REFERENCING OLD AS old_vac NEW AS new_vac
FOR each ROW
WHEN(EXISTS(SELECT* FROM AuditItems WHERE AreaName = 'Vacancy' AND AuditFlag
= 1))
BEGIN
  DECLARE @AuditList long VARCHAR;
  DECLARE @OldDescrip CHAR(250);
  DECLARE @NewDescrip CHAR(250);
  DECLARE @CompName CHAR(250);
  DECLARE @CRLF CHAR(2);
  SET @CRLF = "char"(10)+"char"(13);
  SELECT string(',',list(ItemName),',') INTO @AuditList FROM AuditItems
WHERE (AreaName = 'Company Contact' AND AuditFlag = 1) OR (AreaName =
'Vacancy' AND AuditFlag = 1);
  SET @CompName=(SELECT string(Company.Name,' -
',old_vac.position,'(',old_vac.RefCode,')') FROM Company KEY JOIN Employment
WHERE Employment.EmploymentID = old_vac.EmploymentID);
  -- Status
  IF locate(@AuditList,',Status,') > 0 AND UPDATE(STATUS) THEN
    SELECT name INTO @OldDescrip FROM STATUS WHERE TYPE = 'V' AND
STATUS.status = old_vac.status;
    SELECT name INTO @NewDescrip FROM STATUS WHERE TYPE = 'V' AND
STATUS.status = new_vac.status;
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Status Updated -
',@CompName),string(old_vac.Status,' -
',@OldDescrip),string(new_vac.Status,' - ',@NewDescrip))
  END IF;
  -- Salary
  IF locate(@AuditList,',Salary,') > 0 AND UPDATE(Salary) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Salary Updated -
',@CompName),string(old_vac.Salary),string(new_vac.Salary))
  END IF;
  -- Position
  IF locate(@AuditList,',Position,') > 0 AND UPDATE(POSITION) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Position Updated -
',@CompName),string(old_vac.Position),string(new_vac.Position))
  END IF;
  -- No of Posts
```

```
  IF locate(@AuditList,',No Of Posts,') > 0 AND UPDATE(NoOfPosts) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('No of Posts Updated -
',@CompName),string(old_vac.NoOfPosts),string(new_vac.NoOfPosts))
  END IF;
  -- DATES
  IF locate(@AuditList,',Dates,') > 0 THEN
    IF UPDATE(entrydate) THEN
      CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Entry Date Updated -
',@CompName),string(old_vac.entrydate),string(new_vac.entrydate))
    END IF;
    IF UPDATE(startdate) THEN
      CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Start Date Updated -
',@CompName),string(old_vac.startdate),string(new_vac.startdate))
    END IF;
    IF UPDATE(finishdate) THEN
      CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Finish Date Updated
- ',@CompName),string(old_vac.finishdate),string(new_vac.finishdate))
    END IF;
    IF UPDATE(expiry) THEN
      CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Expiry Date Updated
- ',@CompName),string(old_vac.expiry),string(new_vac.expiry))
    END IF
  END IF;
  -- Consultant
  IF locate(@AuditList,',Consultant,') > 0 AND UPDATE(StaffID) THEN
    SELECT userid INTO @OldDescrip FROM staff WHERE staffid =
old_vac.StaffID;
    SELECT userid INTO @NewDescrip FROM staff WHERE staffid =
new_vac.StaffID;
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Consultant Updated -
',@CompName),@OldDescrip,@NewDescrip);
    INSERT INTO staffhistory (staffhistoryid, vacancyid, oldstaff, newstaff,
whenentered) VALUES (uniquekey(new_vac.StaffID),new_vac.vacancyid,
old_vac.StaffID, new_vac.StaffID, CURRENT TIMESTAMP)
  END IF;
  -- Department
  IF locate(@AuditList,',Department,') > 0 AND UPDATE(DepartmentID) THEN
    SELECT name INTO @OldDescrip FROM Department WHERE DepartmentID =
old_vac.DepartmentID;
    SELECT name INTO @NewDescrip FROM Department WHERE DepartmentID =
new_vac.DepartmentID;
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Department Updated -
',@CompName),@OldDescrip,@NewDescrip)
  END IF;
  -- TempDesk
  IF locate(@AuditList,',Temp Desk,') > 0 AND UPDATE(TempDeskID) THEN
    SELECT name INTO @OldDescrip FROM TempDesk WHERE TempDeskid =
old_vac.TempDeskID;
```

```
    SELECT name INTO @NewDescrip FROM TempDesk WHERE TempDeskid =
new_vac.TempDeskID;
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('TempDesk Updated -
',@CompName),@OldDescrip,@NewDescrip)
  END IF;
  -- Our Ref
  IF locate(@AuditList,',Our Ref,') > 0 AND UPDATE(RefCode) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Our Ref Updated -
',@CompName),string(old_vac.RefCode),string(new_vac.RefCode))
  END IF;
  -- Payrollidentifier
  IF locate(@AuditList,',Payroll Identifier,') > 0 AND
UPDATE(Payrollidentifier) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Payroll Identifier
Updated -
',@CompName),string(old_vac.Payrollidentifier),string(new_vac.Payrollidentif
ier))
  END IF;
  -- ExtraNotes
  IF locate(@AuditList,',Extra Notes,') > 0 AND UPDATE(ExtraNotes) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Extra Notes Updated -
',@CompName),string(old_vac.ExtraNotes),string(new_vac.ExtraNotes))
  END IF;
  -- OtherNotes
  IF locate(@AuditList,',Job Description,') > 0 AND UPDATE(OtherNotes) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Job Description Ref
Updated -
',@CompName),string(old_vac.OtherNotes),string(new_vac.OtherNotes))
  END IF;
  -- Site Details
  IF locate(@AuditList,',Site Details,') > 0 AND (UPDATE(country) OR
UPDATE(county) OR UPDATE(town) OR UPDATE(sitecontact)
  OR UPDATE(postcode)OR UPDATE(addr1)OR UPDATE(addr2)OR UPDATE(addr3)OR
UPDATE(siteemail)OR UPDATE(sitename)) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Site Details Updated -
',@CompName),
string(old_vac.sitename,@CRLF,old_vac.sitecontact,@CRLF,old_vac.addr1,@CRLF,
old_vac.addr2,@CRLF,old_vac.addr3,@CRLF,old_vac.town,@CRLF,old_vac.county,@C
RLF,old_vac.country,@CRLF,old_vac.postcode,@CRLF,old_vac.siteemail),
string(new_vac.sitename,@CRLF,new_vac.sitecontact,@CRLF,new_vac.addr1,@CRLF,
new_vac.addr2,@CRLF,new_vac.addr3,@CRLF,new_vac.town,@CRLF,new_vac.county,@C
RLF,new_vac.country,@CRLF,new_vac.postcode,@CRLF,new_vac.siteemail))
  END IF;
  -- Their Ref
  IF locate(@AuditList,',Their Ref,') > 0 AND UPDATE(TheirRef) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Their Ref Updated -
',@CompName),string(old_vac.TheirRef),string(new_vac.TheirRef))
  END IF;
```

```
  IF locate(@AuditList,',Override IR35 Public Sector,') > 0 AND
UPDATE(nottaxable) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,string('Override IR35 Public
Sector - ',@CompName),old_vac.nottaxable,new_vac.nottaxable)
  END IF;
  -- Contact
  IF locate(@AuditList,',Contact,') > 0 AND UPDATE(employmentid) THEN
    CALL AuditLog('VACANCY',old_VAC.vacancyid,'Contact left company -
vacancies reassigned',
 (SELECT p.name FROM employment AS e KEY JOIN person AS p WHERE
e.employmentid = old_vac.employmentid),(SELECT p.name FROM employment AS e
KEY JOIN person AS p WHERE e.employmentid = new_vac.employmentid))
  END IF;
  IF locate(@AuditList,',Contact Left Company,') > 0 AND
UPDATE(employmentid) THEN
    CALL AuditLog('COMPANY',(SELECT c.Companyid FROM Company AS c KEY JOIN
employment AS e WHERE e.employmentid = old_VAC.employmentid),'Contact left
company - Vacancies reassigned',
 (SELECT p.name FROM employment AS e KEY JOIN person AS p WHERE
e.employmentid = old_vac.employmentid),(SELECT p.name FROM employment AS e
KEY JOIN person AS p WHERE e.employmentid = new_vac.employmentid))
  END IF;
  --  Override Invoice Layout
  IF locate(@AuditList,',Override Invoice Layout,') > 0 AND
UPDATE(documenttemplateid) THEN
  SET @OldDescrip=(SELECT name FROM iqacdocumenttemplate WHERE
documenttemplateid = old_vac.documenttemplateid);
  SET @NewDescrip=(SELECT name FROM iqacdocumenttemplate WHERE
documenttemplateid = new_vac.documenttemplateid);
    CALL AuditLog('VACANCY',old_vac.VacancyID,string('Override Invoice
Layout Updated - ',@CompName),@OldDescrip,@NewDescrip)
  END IF;
END
}
GO


CREATE TRIGGER "vacancy_insert2" after INSERT ORDER 1 ON
"pears"."vacancy"
REFERENCING NEW AS "new_vacancy"
FOR each ROW
BEGIN
  DECLARE "rateschemeid" CHAR(20);
  DECLARE "dynam" INTEGER;
  IF "new_vacancy"."temp" = 1 AND "new_vacancy"."tempjobtypeid" IS NULL THEN
    SELECT "t"."tempjobtypeid","isnull"("t"."dynamic",0) INTO
"rateschemeid","dynam"
      FROM "tempjobtype" AS "t" JOIN "company" AS "c" ON
```

```
"c"."defaultratescheme" = "t"."tempjobtypeid" KEY JOIN "employment" AS "e"
      WHERE "e"."employmentid" = "new_vacancy"."employmentid" AND
"t"."secondaryagencyid" IS NULL
      AND("string"(',',"c"."tempchargecode",',') LIKE
"string"('%,',"t"."clienttempchargecode",',%') OR "t"."clienttempchargecode"
IS NULL)
      AND("t"."DepartmentID" = "new_vacancy"."departmentid" OR
"t"."departmentid" IS NULL);
    IF "rateschemeid" IS NULL THEN
      SELECT "t"."tempjobtypeid","isnull"("t"."dynamic",0) INTO
"rateschemeid","dynam"
        FROM "tempjobtype" AS "t","company" AS "c" KEY JOIN "employment" AS
"e"
        WHERE "e"."employmentid" = "new_vacancy"."employmentid" AND
"t"."secondaryagencyid" IS NULL
        AND "t"."description" LIKE 'Global Default%'
        AND("string"(',',"c"."tempchargecode",',') LIKE
"string"('%,',"t"."clienttempchargecode",',%') OR "t"."clienttempchargecode"
IS NULL)
        AND("t"."DepartmentID" = "new_vacancy"."departmentid" OR
"t"."departmentid" IS NULL)
    END IF;
    IF "rateschemeid" IS NOT NULL THEN
      UPDATE "vacancy" SET "tempjobtypeid" = "rateschemeid" WHERE
"vacancyid" = "new_vacancy"."vacancyid";
      IF "dynam" = 0 THEN
        INSERT INTO "tempjobrate"(
"ChargeRate","EndDate","Grade","PayRate","StartDate","TempJobRateID","TempPa
yBandID","VacancyID" ) (
          SELECT
"ChargeRate","EndDate","Grade","PayRate","StartDate","uniquekey"("tempjobrat
emasterid"),"TempPayBandID","new_vacancy"."vacancyid"
            FROM "tempjobratemaster" WHERE "tempjobtypeid" = "rateschemeid")
      END IF END IF END IF;
  IF "new_vacancy"."temp" = 1 AND
"wpkmaintaingetswitchvalue"('AWRVISIBLE','','L') = 'Y' THEN
    INSERT INTO "AWRvacancy"(
"vacancyid","AWRStatus","benefits","notes","holidays","pay","bonus","extraho
ls" ) SELECT "new_vacancy"."vacancyid",
        "AWRStatus","benefits","notes","holidays","pay","bonus","extrahols"
FROM "AWRcompany" WHERE "companyid" = (SELECT "companyid" FROM "employment"
WHERE "employmentid" = "new_vacancy"."employmentid")
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."vacancy"."vacancy_insert2" IS
```

```
{CREATE TRIGGER vacancy_insert2
after INSERT ORDER 1 ON
"pears"."vacancy"
REFERENCING NEW AS "new_vacancy"
FOR each ROW
BEGIN
  DECLARE "rateschemeid" CHAR(20);
  DECLARE "dynam" INTEGER;
  IF "new_vacancy"."temp" = 1 AND "new_vacancy"."tempjobtypeid" IS NULL THEN
    SELECT "t"."tempjobtypeid","isnull"("t"."dynamic",0) INTO
"rateschemeid","dynam"
      FROM "tempjobtype" AS "t" JOIN "company" AS "c" ON
"c"."defaultratescheme" = "t"."tempjobtypeid" KEY JOIN "employment" AS "e"
      WHERE "e"."employmentid" = "new_vacancy"."employmentid" AND
"t"."secondaryagencyid" IS NULL
      AND("string"(',',"c"."tempchargecode",',') LIKE
"string"('%,',"t"."clienttempchargecode",',%') OR "t"."clienttempchargecode"
IS NULL)
      AND("t"."DepartmentID" = "new_vacancy"."departmentid" OR
"t"."departmentid" IS NULL);
    IF "rateschemeid" IS NULL THEN
      SELECT "t"."tempjobtypeid","isnull"("t"."dynamic",0) INTO
"rateschemeid","dynam"
        FROM "tempjobtype" AS "t","company" AS "c" KEY JOIN "employment" AS
"e"
        WHERE "e"."employmentid" = "new_vacancy"."employmentid" AND
"t"."secondaryagencyid" IS NULL
        AND "t"."description" LIKE 'Global Default%'
        AND("string"(',',"c"."tempchargecode",',') LIKE
"string"('%,',"t"."clienttempchargecode",',%') OR "t"."clienttempchargecode"
IS NULL)
        AND("t"."DepartmentID" = "new_vacancy"."departmentid" OR
"t"."departmentid" IS NULL)
    END IF;
    IF "rateschemeid" IS NOT NULL THEN
      UPDATE "vacancy" SET "tempjobtypeid" = "rateschemeid" WHERE
"vacancyid" = "new_vacancy"."vacancyid";
      IF "dynam" = 0 THEN
        INSERT INTO "tempjobrate"(
"ChargeRate","EndDate","Grade","PayRate","StartDate","TempJobRateID","TempPa
yBandID","VacancyID" ) (
          SELECT
"ChargeRate","EndDate","Grade","PayRate","StartDate","uniquekey"("tempjobrat
ematerid"),"TempPayBandID","new_vacancy"."vacancyid"
            FROM "tempjobratemaster" WHERE "tempjobtypeid" = "rateschemeid")
      END IF
    END IF END IF;
  IF "new_vacancy"."temp" = 1 AND
```

```
"wpkmaintaingetswitchvalue"('AWRVISIBLE','','L') = 'Y' THEN
    INSERT INTO "AWRvacancy"(
"vacancyid","AWRStatus","benefits","notes","holidays","pay","bonus","extraho
ls" ) SELECT "new_vacancy"."vacancyid",
        "AWRStatus","benefits","notes","holidays","pay","bonus","extrahols"
FROM "AWRcompany" WHERE "companyid" = (SELECT "companyid" FROM "employment"
WHERE "employmentid" = "new_vacancy"."employmentid")
  END IF
END
}
GO


CREATE TRIGGER "vacancy_delete" BEFORE DELETE ORDER 1 ON
"pears"."vacancy"
REFERENCING OLD AS "old_vacancy"
FOR each ROW
BEGIN
  UPDATE "person" SET "ExclusiveVacancyID" = NULL WHERE "ExclusiveVacancyID"
= "old_vacancy"."vacancyid"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."vacancy"."vacancy_delete" IS
{CREATE TRIGGER vacancy_delete
 BEFORE DELETE ORDER 1 ON
pears.vacancy
REFERENCING OLD AS old_vacancy
FOR each ROW
BEGIN
  UPDATE person SET ExclusiveVacancyID = NULL WHERE ExclusiveVacancyID =
old_vacancy.vacancyid
END
}
GO


CREATE TRIGGER "psHealthVacancyUpdate" after UPDATE OF "recipientid"
ORDER 900 ON "pears"."vacancy"
REFERENCING NEW AS "new_rec"
FOR each ROW
BEGIN
  IF "psHealthCanSendVacancy"("new_rec"."vacancyid") = 1 THEN
    CALL "psHealthInsertUpdateVacancy"("new_rec"."vacancyid")
  END IF
END
GO
```

```
CREATE TRIGGER "vacancy_UpdateTrim" BEFORE UPDATE ORDER 7 ON
"pears"."vacancy"
REFERENCING NEW AS "new_vacancy"
FOR each ROW
BEGIN
  SET "new_vacancy"."contractref" = "trim"("new_vacancy"."contractref");
  SET "new_vacancy"."refcode" = "trim"("new_vacancy"."refcode");
  SET "new_vacancy"."sitename" = "trim"("new_vacancy"."sitename");
  SET "new_vacancy"."position" = "trim"("new_vacancy"."position");
  SET "new_vacancy"."theirref" = "trim"("new_vacancy"."theirref")
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."vacancy"."vacancy_UpdateTrim"
IS
{CREATE TRIGGER vacancy_UpdateTrim
 BEFORE UPDATE ORDER 7 ON
pears.vacancy
REFERENCING NEW AS new_vacancy
FOR each ROW
BEGIN
  SET new_vacancy.contractref = TRIM(new_vacancy.contractref);
  SET new_vacancy.refcode = TRIM(new_vacancy.refcode);
  SET new_vacancy.sitename = TRIM(new_vacancy.sitename);
  SET new_vacancy.position = TRIM(new_vacancy.position);
  SET new_vacancy.theirref = TRIM(new_vacancy.theirref)
END
}
GO


CREATE TRIGGER "InsertStatusVacancy" after INSERT ORDER 2 ON
"pears"."Vacancy"
REFERENCING NEW AS "new_co"
FOR each ROW
BEGIN
  INSERT INTO "StatusHistory"(
"StatusHistoryID","Vacancyid","staffid","newstatus" ) VALUES
    (
"uniquekey"("new_co"."Vacancyid"),"new_co"."Vacancyid","userstaffid","new_co
"."status" )
END
GO
```

```
COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."vacancy"."InsertStatusVacancy" IS
{CREATE TRIGGER InsertStatusVacancy
 after INSERT ORDER 2 ON
pears.Vacancy
REFERENCING NEW AS new_co
FOR each ROW
BEGIN
  INSERT INTO StatusHistory (StatusHistoryID, Vacancyid, staffid, newstatus)
    VALUES
    (uniquekey(new_co.Vacancyid),new_co.Vacancyid,
userstaffid,new_co.status)
END
}
GO


CREATE TRIGGER "UpdateStatusVacancy" after UPDATE OF "status"
ORDER 3 ON "pears"."Vacancy"
REFERENCING OLD AS "old_co" NEW AS "new_co"
FOR each ROW
BEGIN
  DECLARE "ID" CHAR(20);
  SELECT FIRST "StatusHistoryID" INTO "ID" FROM "StatusHistory" WHERE
"Vacancyid" = "new_co"."Vacancyid"
    AND "datediff"("minute","whenentered",CURRENT TIMESTAMP) < 1 AND
"newstatus" = "old_co"."status" AND "oldstatus" = "new_co"."status" ORDER BY
"whenentered" DESC;
  IF "ID" IS NOT NULL THEN
    DELETE FROM "StatusHistory" WHERE "StatusHistoryID" = "ID"
  ELSE
    INSERT INTO "StatusHistory"(
"StatusHistoryID","Vacancyid","staffid","oldstatus","newstatus" ) VALUES
      (
"uniquekey"("new_co"."Vacancyid"),"new_co"."Vacancyid","userstaffid","old_co
"."status","new_co"."status" )
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."vacancy"."UpdateStatusVacancy" IS
{CREATE TRIGGER UpdateStatusVacancy
 after UPDATE OF STATUS
ORDER 3 ON pears.Vacancy
REFERENCING OLD AS old_co NEW AS new_co
FOR each ROW
```

```
BEGIN
  DECLARE ID CHAR(20);
   SELECT FIRST StatusHistoryID INTO ID FROM StatusHistory WHERE Vacancyid =
new_co.Vacancyid
     AND datediff(MINUTE, whenentered, CURRENT TIMESTAMP) < 1 AND newstatus
= old_co.status AND oldstatus = new_co.status ORDER BY whenentered DESC;
   IF ID IS NOT NULL THEN
     DELETE FROM StatusHistory WHERE StatusHistoryID = ID
   ELSE
     INSERT INTO StatusHistory (StatusHistoryID, Vacancyid, staffid,
oldstatus, newstatus)
      VALUES
      (uniquekey(new_co.Vacancyid),new_co.Vacancyid, userstaffid,
old_co.status, new_co.status)
   END IF;
END
}
GO


CREATE TRIGGER "VacancyKeyWordsInsert" after INSERT ORDER 20 ON
"pears"."vacancy"
REFERENCING NEW AS "NewRow"
FOR each ROW
BEGIN
  INSERT INTO "VacancyKeyWords"( "VacancyID","RefreshRequired" ) ON existing
UPDATE defaults off VALUES( "NewRow"."VacancyID",1 )
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."vacancy"."VacancyKeyWordsInsert" IS
{CREATE TRIGGER VacancyKeyWordsInsert
after INSERT  ORDER 20 ON
 pears.vacancy
 REFERENCING NEW AS NewRow
FOR each ROW
BEGIN
   INSERT INTO VacancyKeyWords(VacancyID,RefreshRequired) ON existing UPDATE
VALUES(NewRow.VacancyID,1);
END
}
GO


CREATE TRIGGER "VacancyKeyWordsUpdate" after UPDATE OF "refcode",
"position","siteemail","sitename","sitecontact","employmentid" ORDER 21 ON
```

```
"pears"."Vacancy"
REFERENCING NEW AS "NewRow"
FOR each ROW
BEGIN
  UPDATE "VacancyKeyWords" SET "RefreshRequired" = 1 WHERE "VacancyID" =
"NewRow"."VacancyID"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."vacancy"."VacancyKeyWordsUpdate" IS
{CREATE TRIGGER VacancyKeyWordsUpdate
after  UPDATE OF refcode, POSITION, siteemail, sitename, sitecontact,
employmentid ORDER 21 ON
   pears.Vacancy
  REFERENCING NEW AS NewRow
 FOR each ROW
 BEGIN
 UPDATE VacancyKeyWords SET RefreshRequired = 1 WHERE VacancyID =
NewRow.VacancyID;
 END
}
GO
```