pears.TempShift

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

Description

Individual shift details - once booked / worked / billed / paid etc. Also availability.

Columns

Column	Type	Null	Default	Comment
TempShiftID	char(20)	NOT NULL		
VacancyID	char(20)	NULL		
PersonID	char(20)	NULL		
PlacementID	char(20)	NULL		
TempTimeSheetID	char(20)	NULL		
ShiftDate	date	NOT NULL		
TimeFrom	time	NULL		
TimeTo	time	NULL		
State	char(1)	NOT NULL		Temp: Available,Unavailable,Holiday; Vacancy: Unfilled,Cancelled; Placement: Provisional,Booked,Worked,Cancelled
Note	char(100)	NULL		
TempShiftPlanID	char(20)	NULL		
EssentialSkillGradeID	char(4)	NULL		
BreakMinutes	smallint	NULL		
ReferenceCode	char(20)	NULL		
CancelReason	char(1)	NULL		
WhenCancelled	timestamp	NULL		
CBill	tinyint	NULL	0	Cancelled shift must still be billed. Clear when billed
CPay	tinyint	NULL	0	Cancelled shift must still be paid. Clear when paid
CRefill	tinyint	NULL	0	Cancelled shift still needs to be filled
ClientConfirmed	tinyint	NULL	0	
TempConfirmed	tinyint	NULL	0	
WhoCancelled	char(20)	NULL		

Column	Type	Null	Default	Comment
StaffID	char(20)	NULL		
WhenEntered	timestamp	NULL	current timestamp	
ConfirmedWith	char(20)	NULL		
TempShiftTypeID	char(2)	NULL		
AnalysisCode	char(20)	NULL		
RecoveryHours	smallint	NULL	0	
EffectiveTimeTo	time	NULL		
UnavailableReason	char(1)	NULL		
CandidateNotified	timestamp	NULL		
ClientNotified	timestamp	NULL		
AvailTemplateID	char(2)	NULL		
AWRNotes	long varchar	NULL		
ColourOverride	smallint	NULL	0	
TempShiftTemplateID	char(20)	NULL		
WhenClientConfirmed	timestamp	NULL		
WhenTempConfirmed	timestamp	NULL		
WhoTempConfirmed	char(20)	NULL		
WhoClientConfirmed	char(20)	NULL		
CancelAt	timestamp	NULL		

Primary Key

- TempShiftID

Foreign Keys

Constraint	Columns	References	Delete/update action
TempShiftCancelReason	CancelReason	pears.TempShiftCancelReason (TempShiftCancelReasonID)	
Vacancy	VacancyID	pears.vacancy (vacancyid)	
Person	PersonID	pears.Person (personid)	
Placement	PlacementID	pears.Placement (placementid)	
TempTimesheet	TempTimeSheetID	pears.TempTimeSheet (TempTimeSheetID)	
TempShiftPlan	TempShiftPlanID	pears.TempShiftPlan (TempShiftPlanID)	ON DELETE SET NULL
Staff	StaffID	pears.staff (staffid)	ON DELETE SET NULL
TempShiftType	TempShiftTypeID	pears.TempShiftType (TempShiftTypeID)	

Constraint	Columns	References	Delete/update action
WhoCancelled	WhoCancelled	pears.staff (staffid)	ON DELETE SET NULL
TempShiftUnavailableReason	UnavailableReason	pears.TempShiftUnavailableReason (TempShiftUnavailableReasonID)	
AvailabilityTemplate	AvailTemplateID	pears.AvailabilityTemplate (AvailTemplateID)	ON DELETE SET NULL
TempShiftTemplate	TempShiftTemplateID	pears.TempShiftTemplate (TempShiftTemplateID)	

Referenced By

Table	Constraint	Columns	Referenced columns
pears.AWRJobMaster	TempShift	TempShiftID	TempShiftID
pears.CardReaderLog	tempshift	MatchedToShiftID	TempShiftID
pears.CardReaderShift	tempshift	MatchedToShiftID	TempShiftID
pears.ExpenseBenefitShiftItem	TempShift	TempShiftID	TempShiftID
pears.IQAcShift	TempShift	TempShiftID	TempShiftID
pears.TempProvTimeSheetLine	TempShift	TempShiftID	TempShiftID
pears.TempProvTimeSheetShift	TempShift	TempShiftID	TempShiftID
pears.TempShiftInvoice	TempShift	TempShiftID	TempShiftID
pears.TempTimeSheetLine	TempShift	TempShiftID	TempShiftID

Indexes

Name	Type	Columns	Detail
TempShift_PDate	Index	PersonID, ShiftDate	
TempShift_VDate	Index	VacancyID, ShiftDate	
TempShift_PStateDate	Index	PersonID, State, ShiftDate	
TempShift_WhenEntered	Index	WhenEntered	
TempShift_ReferenceCode	Index	ReferenceCode	
TempShift_CancelAt	Index	CancelAt	
TempShift_AnalysisCode	Index	AnalysisCode	

Triggers

Name	Timing	Event
Shift_Confirm	before	update of “ClientConfirmed”, “TempConfirmed” order 1
Shift_Insert	before	insert order 1
TempShiftCancelled_ModifyPlan	after	update of “State” order 10
TempShiftAudit	after	update of “VacancyID”, “TimeFrom”,“TimeTo”,“ClientConfirmed”,“TempConfirmed”, “BreakMinutes”,“RecoveryHours” order 2
TempShiftCancelled_AWR	after	update of “State” order 11

Name	Timing	Event
psHealthTempShiftInsert	after	insert order 900
psHealthTempShiftUpdate	after	update of “shiftdate”, “timefrom”,“timeto”,“state” order 900
Shift_TrimUpdate	before	update order 13
AutoMatchShiftInsertUpdate	after	insert,update order 12

Original SQL

```
-- IQX database structure split by table
-- Source: IQXDatabaseStructure - with comments.sql
-- Table: "pears"."TempShift"
-- Table comment: Individual shift details - once booked / worked / billed /
paid etc. Also availability.
-- Statement count: 41

CREATE TABLE "pears"."TempShift" (
    "TempShiftID"                   CHAR(20) NOT NULL
   ,"VacancyID"                     CHAR(20) NULL
   ,"PersonID"                      CHAR(20) NULL
   ,"PlacementID"                   CHAR(20) NULL
   ,"TempTimeSheetID"               CHAR(20) NULL
   ,"ShiftDate"                     DATE NOT NULL
   ,"TimeFrom"                      TIME NULL
   ,"TimeTo"                        TIME NULL
   ,"State"                         CHAR(1) NOT NULL
   ,"Note"                          CHAR(100) NULL
   ,"TempShiftPlanID"               CHAR(20) NULL
   ,"EssentialSkillGradeID"         CHAR(4) NULL
   ,"BreakMinutes"                  SMALLINT NULL
   ,"ReferenceCode"                 CHAR(20) NULL
   ,"CancelReason"                  CHAR(1) NULL
   ,"WhenCancelled"                 TIMESTAMP NULL
   ,"CBill"                         tinyint NULL DEFAULT 0
   ,"CPay"                          tinyint NULL DEFAULT 0
   ,"CRefill"                       tinyint NULL DEFAULT 0
   ,"ClientConfirmed"               tinyint NULL DEFAULT 0
   ,"TempConfirmed"                 tinyint NULL DEFAULT 0
   ,"WhoCancelled"                  CHAR(20) NULL
   ,"StaffID"                       CHAR(20) NULL
   ,"WhenEntered"                   TIMESTAMP NULL DEFAULT CURRENT
TIMESTAMP
   ,"ConfirmedWith"                 CHAR(20) NULL
   ,"TempShiftTypeID"               CHAR(2) NULL
   ,"AnalysisCode"                  CHAR(20) NULL
   ,"RecoveryHours"                 SMALLINT NULL DEFAULT 0
   ,"EffectiveTimeTo"               TIME NULL COMPUTE
```

```
("pears"."AddRecoveryHours"("timefrom","timeto","recoveryhours"))
    ,"UnavailableReason"                CHAR(1) NULL
    ,"CandidateNotified"                TIMESTAMP NULL
    ,"ClientNotified"                   TIMESTAMP NULL
    ,"AvailTemplateID"                  CHAR(2) NULL
    ,"AWRNotes"                         long VARCHAR NULL
    ,"ColourOverride"                   SMALLINT NULL DEFAULT 0
    ,"TempShiftTemplateID"              CHAR(20) NULL
    ,"WhenClientConfirmed"              TIMESTAMP NULL
    ,"WhenTempConfirmed"                TIMESTAMP NULL
    ,"WhoTempConfirmed"                 CHAR(20) NULL
    ,"WhoClientConfirmed"               CHAR(20) NULL
    ,"CancelAt"                         TIMESTAMP NULL
    ,PRIMARY KEY ("TempShiftID" ASC)
)
GO


COMMENT ON COLUMN "pears"."TempShift"."State" IS
    'Temp: Available,Unavailable,Holiday; Vacancy: Unfilled,Cancelled;
Placement: Provisional,Booked,Worked,Cancelled'
GO


COMMENT ON COLUMN "pears"."TempShift"."CBill" IS
    'Cancelled shift must still be billed. Clear when billed'
GO


COMMENT ON COLUMN "pears"."TempShift"."CPay" IS
    'Cancelled shift must still be paid. Clear when paid'
GO


COMMENT ON COLUMN "pears"."TempShift"."CRefill" IS
    'Cancelled shift still needs to be filled'
GO


COMMENT ON TABLE "pears"."TempShift" IS
    'Individual shift details - once booked / worked / billed / paid etc.
Also availability.'
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "TempShiftCancelReason" ("CancelReason" ASC)
    REFERENCES "pears"."TempShiftCancelReason" ("TempShiftCancelReasonID")
```

```
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "Vacancy" ("VacancyID" ASC)
    REFERENCES "pears"."vacancy" ("vacancyid")
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "Person" ("PersonID" ASC)
    REFERENCES "pears"."Person" ("personid")
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "Placement" ("PlacementID" ASC)
    REFERENCES "pears"."Placement" ("placementid")
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "TempTimesheet" ("TempTimeSheetID" ASC)
    REFERENCES "pears"."TempTimeSheet" ("TempTimeSheetID")
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "TempShiftPlan" ("TempShiftPlanID" ASC)
    REFERENCES "pears"."TempShiftPlan" ("TempShiftPlanID")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "Staff" ("StaffID" ASC)
    REFERENCES "pears"."staff" ("staffid")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "TempShiftType" ("TempShiftTypeID" ASC)
    REFERENCES "pears"."TempShiftType" ("TempShiftTypeID")
GO


ALTER TABLE "pears"."TempShift"
```

```
    ADD FOREIGN KEY "WhoCancelled" ("WhoCancelled" ASC)
    REFERENCES "pears"."staff" ("staffid")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "TempShiftUnavailableReason" ("UnavailableReason" ASC)
    REFERENCES "pears"."TempShiftUnavailableReason"
("TempShiftUnavailableReasonID")
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "AvailabilityTemplate" ("AvailTemplateID" ASC)
    REFERENCES "pears"."AvailabilityTemplate" ("AvailTemplateID")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."TempShift"
    ADD FOREIGN KEY "TempShiftTemplate" ("TempShiftTemplateID" ASC)
    REFERENCES "pears"."TempShiftTemplate" ("TempShiftTemplateID")
GO


CREATE INDEX "TempShift_PDate" ON "pears"."TempShift"
    ( "PersonID","ShiftDate" )
GO


CREATE INDEX "TempShift_VDate" ON "pears"."TempShift"
    ( "VacancyID","ShiftDate" )
GO


CREATE INDEX "TempShift_PStateDate" ON "pears"."TempShift"
    ( "PersonID","State","ShiftDate" )
GO


CREATE INDEX "TempShift_WhenEntered" ON "pears"."TempShift"
    ( "WhenEntered" )
GO


CREATE INDEX "TempShift_ReferenceCode" ON "pears"."TempShift"
    ( "ReferenceCode" )
```

```
GO


CREATE INDEX "TempShift_CancelAt" ON "pears"."TempShift"
    ( "CancelAt" )
GO


CREATE INDEX "TempShift_AnalysisCode" ON "pears"."TempShift"
    ( "AnalysisCode" )
GO


CREATE TRIGGER "Shift_Confirm" BEFORE UPDATE OF "ClientConfirmed",
"TempConfirmed" ORDER 1 ON "pears"."TempShift"
REFERENCING NEW AS "new_shift"
FOR each ROW
BEGIN
  CALL
"TempShiftConfirmationChange"("new_shift"."tempshiftid","new_shift"."vacancy
id","new_shift"."personid","new_shift"."shiftdate","new_shift"."clientconfir
med","new_shift"."tempconfirmed");
  IF "new_shift"."state" = 'P' AND "new_shift"."clientconfirmed" = 1 AND
"new_shift"."tempconfirmed" = 1 THEN
    SET "new_shift"."state" = 'B'
  ELSE
    IF "new_shift"."state" = 'B'
AND("isnull"("new_shift"."clientconfirmed",0) = 0 OR
"isnull"("new_shift"."tempconfirmed",0) = 0) THEN
      SET "new_shift"."state" = 'P'
    END IF END IF;
  IF UPDATE("ClientConfirmed") THEN
    IF "varexists"('userstaffid') <> 0 THEN
      SET "new_shift"."whoclientconfirmed" = "userstaffid"
    END IF;
    SET "new_shift"."whenclientconfirmed" = CURRENT TIMESTAMP
  END IF;
  IF UPDATE("TempConfirmed") THEN
    IF "varexists"('userstaffid') <> 0 THEN
      SET "new_shift"."whotempconfirmed" = "userstaffid"
    END IF;
    SET "new_shift"."whentempconfirmed" = CURRENT TIMESTAMP
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."TempShift"."Shift_Confirm" IS
```

```
{CREATE TRIGGER Shift_Confirm
 BEFORE UPDATE OF ClientConfirmed,
TempConfirmed ORDER 1 ON pears.TempShift
REFERENCING NEW AS new_shift
FOR each ROW
BEGIN
  CALL
TempShiftConfirmationChange(new_shift.tempshiftid,new_shift.vacancyid,new_sh
ift.personid,new_shift.shiftdate,new_shift.clientconfirmed,new_shift.tempcon
firmed);
  IF new_shift.state = 'P' AND new_shift.clientconfirmed = 1 AND
new_shift.tempconfirmed = 1 THEN
    SET new_shift.state='B'
  ELSE
    IF new_shift.state = 'B' AND(isnull(new_shift.clientconfirmed,0) = 0 OR
isnull(new_shift.tempconfirmed,0) = 0) THEN
      SET new_shift.state='P'
    END IF
  END IF;
      IF UPDATE(ClientConfirmed) THEN
        IF varexists('userstaffid') <> 0 THEN
           SET new_shift.whoclientconfirmed=userstaffid
        END IF;
        SET new_shift.whenclientconfirmed=CURRENT TIMESTAMP
      END IF;
      IF UPDATE(TempConfirmed) THEN
        IF varexists('userstaffid') <> 0 THEN
           SET new_shift.whotempconfirmed=userstaffid
        END IF;
        SET new_shift.whentempconfirmed=CURRENT TIMESTAMP
      END IF;
END
}
GO


CREATE TRIGGER "Shift_Insert" BEFORE INSERT ORDER 1 ON
"pears"."TempShift"
REFERENCING NEW AS "new_shift"
FOR each ROW
BEGIN
  SET "new_shift"."referencecode" = "trim"("new_shift"."referencecode");
  CALL
"TempShiftConfirmationChange"("new_shift"."tempshiftid","new_shift"."vacancy
id","new_shift"."personid","new_shift"."shiftdate","new_shift"."clientconfir
med","new_shift"."tempconfirmed");
  IF "new_shift"."staffid" IS NULL THEN
    SET "new_shift"."staffid" = "userstaffid"
```

```
  END IF;
  IF("wpkmaintaingetswitchvalue"('AWRVISIBLE','','L') = 'Y')
AND(("new_shift"."state" = 'U') OR("new_shift"."state" = 'H')) THEN
    IF "new_shift"."state" = 'H' THEN
      CALL
"AWRInsertWeekly"("new_shift"."shiftdate","new_shift"."shiftdate",'H','H',"n
ew_shift"."personid",NULL,"new_shift"."AWRnotes")
    END IF;
    IF("new_shift"."state" = 'U') AND EXISTS(SELECT
"tempshiftunavailablereasonid" FROM "tempshiftunavailablereason" WHERE
"awraction" <> 0 AND "tempshiftunavailablereasonid" =
"new_shift"."unavailablereason") THEN
      CALL
"AWRInsertWeekly"("new_shift"."shiftdate","new_shift"."shiftdate","new_shift
"."unavailablereason",'P',"new_shift"."personid",NULL,"new_shift"."AWRnotes"
)
    END IF
  END IF
exception
  WHEN others THEN SET "new_shift"."staffid" = NULL
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."TempShift"."Shift_Insert" IS
{CREATE TRIGGER Shift_Insert
 BEFORE INSERT ORDER 1 ON
pears.TempShift
REFERENCING NEW AS new_shift
FOR each ROW
BEGIN
  SET new_shift.referencecode = TRIM(new_shift.referencecode);
  CALL
TempShiftConfirmationChange(new_shift.tempshiftid,new_shift.vacancyid,new_sh
ift.personid,new_shift.shiftdate,new_shift.clientconfirmed,new_shift.tempcon
firmed);
  IF new_shift.staffid IS NULL THEN
    SET new_shift.staffid=userstaffid
  END IF;
  IF (wpkmaintaingetswitchvalue('AWRVISIBLE','','L')= 'Y')
AND((new_shift.state = 'U') OR(new_shift.state = 'H')) THEN
    IF new_shift.state = 'H' THEN
      CALL
AWRInsertWeekly(new_shift.shiftdate,new_shift.shiftdate,'H','H',new_shift.pe
rsonid,NULL,new_shift.AWRnotes)
    END IF;
    IF(new_shift.state = 'U') AND EXISTS(SELECT tempshiftunavailablereasonid
FROM tempshiftunavailablereason WHERE awraction <> 0 AND
```

```
tempshiftunavailablereasonid = new_shift.unavailablereason) THEN
      CALL
AWRInsertWeekly(new_shift.shiftdate,new_shift.shiftdate,new_shift.unavailabl
ereason,'P',new_shift.personid,NULL,new_shift.AWRnotes)
    END IF
  END IF
exception
  WHEN others THEN
    SET new_shift.staffid=NULL
END
}
GO


CREATE TRIGGER "TempShiftCancelled_ModifyPlan" after UPDATE OF "State"
ORDER 10 ON "pears"."TempShift"
REFERENCING OLD AS "OldShift" NEW AS "NewShift"
FOR each ROW
WHEN("NewShift"."State" = 'C' AND "OldShift"."State" IN( 'B','P' ) AND
"NewShift"."CReFill" = 1)
BEGIN
  DECLARE @AdjustPlan SMALLINT;
  DECLARE @PlanTimeFrom TIME;
  DECLARE @PlanTimeTo TIME;
  DECLARE @PlanBreakMinutes SMALLINT;
  DECLARE @PlanRecoveryHours SMALLINT;
  SET @AdjustPlan = (SELECT "AdjustPlanOfCancelledShifts" FROM "TempDesk"
KEY JOIN "Vacancy" WHERE "Vacancy"."VacancyID" = "NewShift"."VacancyID");
  IF @AdjustPlan = 1 THEN
    SELECT "TimeFrom","TimeTo","BreakMinutes","RecoveryHours" INTO
@PlanTimeFrom,@PlanTimeTo,@PlanBreakMinutes,@PlanRecoveryHours FROM
"TempShiftPlan" WHERE "NewShift"."TempShiftPlanID" =
"TempShiftPlan"."TempShiftPlanID";
    IF "NewShift"."TimeFrom" <> @PlanTimeFrom OR "NewShift"."TimeTo" <>
@PlanTimeTo OR "isnull"("NewShift"."BreakMinutes",0) <>
"isnull"(@PlanBreakMinutes,0) OR "isnull"("NewShift"."RecoveryHours",0) <>
"isnull"(@PlanRecoveryHours,0) THEN
      UPDATE "TempShiftPlan"
        SET "TimeFrom" = "NewShift"."TimeFrom","TimeTo" =
"NewShift"."TimeTo","BreakMinutes" =
"NewShift"."BreakMinutes","RecoveryHours" = "NewShift"."RecoveryHours"
        WHERE "TempShiftPlan"."TempShiftPlanID" =
"NewShift"."TempShiftPlanID";
      CALL "AuditLog"('SHIFTPLAN',"NewShift"."TempShiftPlanID",'Shift Plan
modified to match cancelled shift',
      "string"('Time From: ',@PlanTimeFrom,', Time To: ',@PlanTimeTo,',
Break: ',@PlanBreakMinutes,', Recovery: ',@PlanRecoveryHours),
      "string"('Time From: ',"NewShift"."TimeFrom",', Time To:
```

```
',"NewShift"."TimeTo",', Break: ',"NewShift"."BreakMinutes",', Recovery:
',"NewShift"."RecoveryHours"))
    END IF
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."TempShift"."TempShiftCancelled_ModifyPlan" IS
{CREATE TRIGGER TempShiftCancelled_ModifyPlan
after UPDATE OF State
ORDER 10 ON pears.TempShift
REFERENCING OLD AS OldShift NEW AS NewShift
FOR each ROW
WHEN(NewShift.State = 'C' AND OldShift.State IN( 'B','P') AND
NewShift.CReFill = 1)
BEGIN
  DECLARE @AdjustPlan SMALLINT;
  DECLARE @PlanTimeFrom TIME;
  DECLARE @PlanTimeTo TIME;
  DECLARE @PlanBreakMinutes SMALLINT;
  DECLARE @PlanRecoveryHours SMALLINT;
  SET @AdjustPlan=(SELECT AdjustPlanOfCancelledShifts FROM TempDesk KEY JOIN
Vacancy WHERE Vacancy.VacancyID = NewShift.VacancyID);
  IF @AdjustPlan = 1 THEN
    SELECT TimeFrom,TimeTo,BreakMinutes,RecoveryHours INTO
@PlanTimeFrom,@PlanTimeTo,@PlanBreakMinutes,@PlanRecoveryHours FROM
TempShiftPlan WHERE NewShift.TempShiftPlanID =
TempShiftPlan.TempShiftPlanID;
    IF NewShift.TimeFrom <> @PlanTimeFrom OR NewShift.TimeTo <> @PlanTimeTo
OR isnull(NewShift.BreakMinutes,0) <> isnull(@PlanBreakMinutes,0) OR
isnull(NewShift.RecoveryHours,0) <> isnull(@PlanRecoveryHours,0) THEN
      UPDATE TempShiftPlan SET
        TimeFrom = NewShift.TimeFrom,TimeTo = NewShift.TimeTo,BreakMinutes =
NewShift.BreakMinutes,RecoveryHours = NewShift.RecoveryHours WHERE
        TempShiftPlan.TempShiftPlanID = NewShift.TempShiftPlanID;
      CALL AuditLog('SHIFTPLAN',NewShift.TempShiftPlanID,'Shift Plan
modified to match cancelled shift',
      string('Time From: ',@PlanTimeFrom,', Time To: ',@PlanTimeTo,', Break:
',@PlanBreakMinutes,', Recovery: ',@PlanRecoveryHours),
      string('Time From: ',NewShift.TimeFrom,', Time To:
',NewShift.TimeTo,', Break: ',NewShift.BreakMinutes,', Recovery:
',NewShift.RecoveryHours))
    END IF
  END IF
END
}
```

```
GO


CREATE TRIGGER "TempShiftAudit" after UPDATE OF "VacancyID",
"TimeFrom","TimeTo","ClientConfirmed","TempConfirmed",
"BreakMinutes","RecoveryHours" ORDER 2 ON "pears"."TempShift"
REFERENCING OLD AS "OldShift" NEW AS "NewShift"
FOR each ROW
WHEN(EXISTS(SELECT * FROM "AuditItems" WHERE "AreaName" = 'Shift' AND
"AuditFlag" = 1))
BEGIN
  DECLARE "AuditList" long VARCHAR;
  DECLARE "NewRefCode" CHAR(20);
  DECLARE "OldRefCode" CHAR(20);
  DECLARE "SerialNo" CHAR(20);
  DECLARE "NewConfirm" CHAR(60);
  DECLARE "OldConfirm" CHAR(60);
  SELECT "string"(',',"list"("ItemName"),',') INTO "AuditList" FROM
"AuditItems" WHERE "AreaName" = 'SHIFT' AND "AuditFlag" = 1;
  SELECT "ShiftSerialNumber" INTO "SerialNo" FROM "tempshiftplan" WHERE
"tempshiftplanid" = "NewShift"."TempShiftPlanID";
  SELECT "RefCode" INTO "NewRefCode" FROM "Vacancy" WHERE
"Vacancy"."VacancyID" = "NewShift"."VacancyID";
  IF "locate"("AuditList",',Temp Confirmed,') > 0 THEN
    IF UPDATE("TempConfirmed") THEN
      SELECT "Name" INTO "OldConfirm" FROM "staff" WHERE "staffID" =
"OldShift"."WhoTempConfirmed";
      SELECT "Name" INTO "NewConfirm" FROM "staff" WHERE "staffID" =
"NewShift"."WhoTempConfirmed";
      CALL "AuditLog"('SHIFT',"NewShift"."TempShiftID",'Shift ' ||
"SerialNo" || ' Temp Confirmed edited OurRef: ' || "NewRefCode",
      "string"("OldShift"."TempConfirmed",',
',"Dateformat"("OldShift"."WhenTempConfirmed",'dd/mm/yy hh:nn:ss'),',
',"OldConfirm"),
      "string"("NewShift"."TempConfirmed",',
',"Dateformat"("NewShift"."WhenTempConfirmed",'dd/mm/yy hh:nn:ss'),',
',"NewConfirm"))
    END IF
  END IF;
  IF "locate"("AuditList",',Client Confirmed,') > 0 THEN
    IF UPDATE("ClientConfirmed") THEN
      SELECT "Name" INTO "OldConfirm" FROM "staff" WHERE "staffID" =
"OldShift"."WhoClientConfirmed";
      SELECT "Name" INTO "NewConfirm" FROM "staff" WHERE "staffID" =
"NewShift"."WhoClientConfirmed";
      CALL "AuditLog"('SHIFT',"NewShift"."TempShiftID",'Shift ' ||
"SerialNo" || ' Client Confirmed edited OurRef: ' || "NewRefCode",
      "string"("OldShift"."ClientConfirmed",',
```

```
',"Dateformat"("OldShift"."WhenClientConfirmed",'dd/mm/yy hh:nn:ss'),',
',"OldConfirm"),
      "string"("NewShift"."ClientConfirmed",',
',"Dateformat"("NewShift"."WhenClientConfirmed",'dd/mm/yy hh:nn:ss'),',
',"NewConfirm"))
    END IF
  END IF;
  IF "locate"("AuditList",',Moving,') > 0 AND UPDATE("VacancyID") THEN
    SELECT "RefCode" INTO "OldRefCode" FROM "Vacancy" WHERE
"Vacancy"."VacancyID" = "OldShift"."VacancyID";
    CALL "AuditLog"('SHIFT',"NewShift"."TempShiftID",'Shift ' || "SerialNo"
|| ' moved between vacancies',
    "string"('Original OurRef: ',"OldRefCode",', Original VacancyID:
',"OldShift"."VacancyID"),
    "string"('New OurRef: ',"NewRefCode",', New VacancyID:
',"NewShift"."VacancyID"))
  ELSE
    IF "locate"("AuditList",',Times,') > 0 THEN
      IF UPDATE("RecoveryHours") THEN
        CALL "AuditLog"('SHIFT',"NewShift"."TempShiftID",'Shift ' ||
"SerialNo" || ' Recovery Hours edited OurRef: ' || "NewRefCode",
        "OldShift"."RecoveryHours",
        "NewShift"."RecoveryHours")
      END IF;
      IF UPDATE("BreakMinutes") THEN
        CALL "AuditLog"('SHIFT',"NewShift"."TempShiftID",'Shift ' ||
"SerialNo" || ' Break Minutes edited OurRef: ' || "NewRefCode",
        "OldShift"."BreakMinutes",
        "NewShift"."BreakMinutes")
      END IF;
      IF UPDATE("TimeFrom") THEN
        CALL "AuditLog"('SHIFT',"NewShift"."TempShiftID",'Shift ' ||
"SerialNo" || ' From edited OurRef: ' || "NewRefCode",
        "OldShift"."TimeFrom",
        "NewShift"."TimeFrom")
      END IF;
      IF UPDATE("TimeTo") THEN
        CALL "AuditLog"('SHIFT',"NewShift"."TempShiftID",'Shift ' ||
"SerialNo" || ' To edited OurRef: ' || "NewRefCode",
        "OldShift"."TimeTo",
        "NewShift"."TimeTo")
      END IF
    END IF
  END IF
END
GO
```

```
COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."TempShift"."TempShiftAudit"
IS
{CREATE TRIGGER TempShiftAudit
 after UPDATE OF VacancyID,
TimeFrom,TimeTo,ClientConfirmed,TempConfirmed,
BreakMinutes,RecoveryHours ORDER 2 ON pears.TempShift
REFERENCING OLD AS OldShift NEW AS NewShift
FOR each ROW
WHEN(EXISTS(SELECT * FROM AuditItems WHERE AreaName = 'Shift' AND AuditFlag
= 1))
BEGIN
  DECLARE AuditList long VARCHAR;
  DECLARE NewRefCode CHAR(20);
  DECLARE OldRefCode CHAR(20);
  DECLARE SerialNo CHAR(20);
  DECLARE NewConfirm CHAR(60);
  DECLARE OldConfirm CHAR(60);
  SELECT string(',',list(ItemName),',') INTO AuditList FROM AuditItems WHERE
AreaName = 'SHIFT' AND AuditFlag = 1;
  SELECT ShiftSerialNumber INTO SerialNo FROM tempshiftplan WHERE
tempshiftplanid = NewShift.TempShiftPlanID;
  SELECT RefCode INTO NewRefCode FROM Vacancy WHERE Vacancy.VacancyID =
NewShift.VacancyID;
    IF locate(AuditList,',Temp Confirmed,') > 0 THEN
      IF UPDATE(TempConfirmed) THEN
        SELECT Name INTO OldConfirm FROM staff WHERE staffID =
OldShift.WhoTempConfirmed;
        SELECT Name INTO NewConfirm FROM staff WHERE staffID =
NewShift.WhoTempConfirmed;
        CALL AuditLog('SHIFT',NewShift.TempShiftID,'Shift '||SerialNo||'
Temp Confirmed edited OurRef: ' || NewRefCode,
        string(OldShift.TempConfirmed,',
',Dateformat(OldShift.WhenTempConfirmed,'dd/mm/yy hh:nn:ss'),',
',OldConfirm),
        string(NewShift.TempConfirmed,',
',Dateformat(NewShift.WhenTempConfirmed,'dd/mm/yy hh:nn:ss'),',
',NewConfirm))
      END IF;
    END IF;
    IF locate(AuditList,',Client Confirmed,') > 0 THEN
      IF UPDATE(ClientConfirmed) THEN
        SELECT Name INTO OldConfirm FROM staff WHERE staffID =
OldShift.WhoClientConfirmed;
        SELECT Name INTO NewConfirm FROM staff WHERE staffID =
NewShift.WhoClientConfirmed;
        CALL AuditLog('SHIFT',NewShift.TempShiftID,'Shift '||SerialNo||'
Client Confirmed edited OurRef: ' || NewRefCode,
        string(OldShift.ClientConfirmed,',
```

```
',Dateformat(OldShift.WhenClientConfirmed,'dd/mm/yy hh:nn:ss'),',
',OldConfirm),
        string(NewShift.ClientConfirmed,',
',Dateformat(NewShift.WhenClientConfirmed,'dd/mm/yy hh:nn:ss'),',
',NewConfirm))
      END IF;
    END IF;
  IF locate(AuditList,',Moving,') > 0 AND UPDATE(VacancyID) THEN
    SELECT RefCode INTO OldRefCode FROM Vacancy WHERE Vacancy.VacancyID =
OldShift.VacancyID;
    CALL AuditLog('SHIFT',NewShift.TempShiftID,'Shift '||SerialNo||' moved
between vacancies',
    string('Original OurRef: ',OldRefCode,', Original VacancyID:
',OldShift.VacancyID),
    string('New OurRef: ',NewRefCode,', New VacancyID:
',NewShift.VacancyID))
  ELSE
    IF locate(AuditList,',Times,') > 0 THEN
      IF UPDATE(RecoveryHours) THEN
        CALL AuditLog('SHIFT',NewShift.TempShiftID,'Shift '||SerialNo||'
Recovery Hours edited OurRef: ' || NewRefCode,
        OldShift.RecoveryHours,
        NewShift.RecoveryHours)
      END IF;
      IF UPDATE(BreakMinutes) THEN
        CALL AuditLog('SHIFT',NewShift.TempShiftID,'Shift '||SerialNo||'
Break Minutes edited OurRef: ' || NewRefCode,
        OldShift.BreakMinutes,
        NewShift.BreakMinutes)
      END IF;
      IF UPDATE(TimeFrom) THEN
        CALL AuditLog('SHIFT',NewShift.TempShiftID,'Shift '||SerialNo||'
From edited OurRef: ' || NewRefCode,
        OldShift.TimeFrom,
        NewShift.TimeFrom)
      END IF;
      IF UPDATE(TimeTo) THEN
        CALL AuditLog('SHIFT',NewShift.TempShiftID,'Shift '||SerialNo||' To
edited OurRef: ' || NewRefCode,
        OldShift.TimeTo,
        NewShift.TimeTo)
      END IF
    END IF
  END IF
END
}
GO
```

```
CREATE TRIGGER "TempShiftCancelled_AWR" after UPDATE OF "State"
ORDER 11 ON "pears"."TempShift"
REFERENCING OLD AS "OldShift" NEW AS "NewShift"
FOR each ROW
WHEN("NewShift"."State" = 'C')
BEGIN
  CALL "AWRShiftCancel"("NewShift"."TempShiftID")
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."TempShift"."TempShiftCancelled_AWR" IS
{CREATE TRIGGER TempShiftCancelled_AWR
after UPDATE OF State
ORDER 11 ON pears.TempShift
REFERENCING OLD AS OldShift NEW AS NewShift
FOR each ROW
WHEN (NewShift.State = 'C')
BEGIN
  CALL AWRShiftCancel(NewShift.TempShiftID)
END
}
GO


CREATE TRIGGER "psHealthTempShiftInsert" after INSERT ORDER 900 ON
"pears"."tempshift"
REFERENCING NEW AS "new_ts"
FOR each ROW
BEGIN
  IF "psHealthCanSendShift"("new_ts"."tempshiftid",NULL) = 1 THEN
    CALL "psHealthInsertUpdateShift"("new_ts"."tempshiftid",NULL)
  END IF
END
GO


CREATE TRIGGER "psHealthTempShiftUpdate" after UPDATE OF "shiftdate",
"timefrom","timeto","state" ORDER 900 ON "pears"."tempshift"
REFERENCING NEW AS "new_ts"
FOR each ROW
BEGIN
  IF "psHealthCanSendShift"("new_ts"."tempshiftid",NULL) = 1 THEN
    CALL "psHealthInsertUpdateShift"("new_ts"."tempshiftid",NULL)
  END IF
END
```

```
GO


CREATE TRIGGER "Shift_TrimUpdate" BEFORE UPDATE ORDER 13 ON
"pears"."TempShift"
REFERENCING NEW AS "new_shift"
FOR each ROW
BEGIN
  SET "new_shift"."referencecode" = "trim"("new_shift"."referencecode")
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."TempShift"."Shift_TrimUpdate"
IS
{CREATE TRIGGER Shift_TrimUpdate
 BEFORE UPDATE ORDER 13 ON
pears.TempShift
REFERENCING  NEW AS new_shift
FOR each ROW
BEGIN
  SET new_shift.referencecode = TRIM(new_shift.referencecode)
END
}
GO


CREATE TRIGGER "AutoMatchShiftInsertUpdate" after INSERT,UPDATE ORDER 12 ON
"pears"."TempShift"
REFERENCING NEW AS "New_Shift"
FOR each ROW
WHEN("AutoMatchingEnabled" = 'Y')
BEGIN
  -- This trigger deals with filling and cancelling
  IF(SELECT "TempDesk"."AutoMatchOn" FROM "TempDesk" KEY JOIN "Vacancy" AS
"v" WHERE "v"."VacancyID" = "New_Shift"."VacancyID") = 1 THEN
    CALL
"AutoMatchRemoveShiftFromQueue"("New_Shift"."TempShiftPlanID","New_Shift"."s
tate","New_Shift"."CRefill")
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."TempShift"."AutoMatchShiftInsertUpdate" IS
{CREATE TRIGGER AutoMatchShiftInsertUpdate
 AFTER INSERT, UPDATE
```

```
ORDER 12 ON "pears"."TempShift"
REFERENCING NEW AS New_Shift
FOR EACH ROW
WHEN (AutoMatchingEnabled = 'Y')
BEGIN
-- This trigger deals with filling and cancelling
IF (SELECT TempDesk.AutoMatchOn FROM TempDesk KEY JOIN Vacancy v WHERE
v.VacancyID = New_Shift.VacancyID) = 1
 THEN CALL
AutoMatchRemoveShiftFromQueue(New_Shift.TempShiftPlanID,New_Shift.state,
New_Shift.CRefill);
END IF;
END
}
GO
```