pears.TempShiftPlan

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

Description

Shift requirements. Linked to TempShift when booked / worked / billed / paid etc.

Columns

Column	Type	Null	Default	Comment	
TempShiftPlanID	char(20)	NOT NULL			
VacancyID	char(20)	NOT NULL			
EssentialSkill	char(15)	NULL		P	A;TagID;TagChoiceID;GradeTagID
EssentialSkillGradeID	char(4)	NULL			
Description	char(50)	NOT NULL			
TimeFrom	time	NULL			
TimeTo	time	NULL			
Moveable	smallint	NULL	0		
Minutes	smallint	NULL		Fill if shorter spell within the timespan	
BreakMinutes	smallint	NULL			
ShiftDate	date	NULL			
ReferenceRequired	char(1)	NULL		on <E>ntry, <C>onfirmation, <T>imesheet entry. Blank for not required	
ClientMustConfirm	smallint	NULL	0		
WhenEntered	timestamp	NULL	current timestamp		
ReferenceCode	char(20)	NULL			
EssentialSkillChoiceList	char(100)	NULL			
TempMustConfirm	smallint	NULL	0		
StaffID	char(20)	NULL			
OrderedBy	char(50)	NULL			
TempShiftTypeID	char(2)	NULL			

Column	Type	Null	Default	Comment	
AnalysisCode	char(20)	NULL			
ShiftSerialNumber	bigint	NULL	autoincrement		
TemporaryID	char(20)	NULL		Internal use only	
CascadeDateTime	timestamp	NULL		When next cascade will take place	
CascadeLevel	smallint	NULL	0	0 if has not yet been cascaded	
RecoveryHours	smallint	NULL	0		
TempShiftOrderReasonID	char(20)	NULL			
ClientNote	char(50)	NULL			
TemporaryPersonID	char(20)	NULL		Internal use only	
ColourOverride	smallint	NULL	0		
TempShiftTemplateID	char(20)	NULL			
MasterRosterShiftID	char(20)	NULL			

Primary Key

- TempShiftPlanID

Foreign Keys

Constraint	Columns	References	Delete/update action
Vacancy	VacancyID	pears.vacancy (vacancyid)	NOT NULL; ON DELETE CASCADE
Staff	StaffID	pears.staff (staffid)	ON DELETE SET NULL
TempShiftType	TempShiftTypeID	pears.TempShiftType (TempShiftTypeID)	
TempShiftOrderReason	TempShiftOrderReasonID	pears.TempShiftOrderReason (TempShiftOrderReasonID)	
TempShiftTemplate	TempShiftTemplateID	pears.TempShiftTemplate (TempShiftTemplateID)	
MasterRosterShift	MasterRosterShiftID	pears.MasterRosterShift (MasterRosterShiftID)	ON DELETE SET NULL

Referenced By

Table	Constraint	Columns	Referenced columns
pears.AutoMatchNotificationQueue	tempshiftplan	TempShiftPlanID	TempShiftPlanID
pears.AutoMatchShiftQueue	tempshiftplan	TempShiftPlanID	TempShiftPlanID
pears.CascadedShift	TempShiftPlan	TempshiftPlanID	TempShiftPlanID

Table	Constraint	Columns	Referenced columns
pears.IQXNetCandidateShifts	TempShiftPlan	TempShiftPlanID	TempShiftPlanID
pears.TempShift	TempShiftPlan	TempShiftPlanID	TempShiftPlanID
pears.TempShiftPlanAuthorise	TempShiftPlan	TempShiftPlanID	TempShiftPlanID
pears.TempShiftProgress	TempShiftPlan	TempShiftPlanID	TempShiftPlanID

Indexes

Name	Type	Columns	Detail
ShiftPlan_TempID	Index	TemporaryID	
ShiftPlan_SerialNo	Index	ShiftSerialNumber	
ShiftPlan_Cascade	Index	CascadeDateTime	
ShiftPlan_WhenEntered	Index	WhenEntered	
ShiftPlan_AnalysisCode	Index	AnalysisCode	

Triggers

Name	Timing	Event
ShiftPlan_Insert	before	insert order 1
TempShiftPlanAudit	after	update of “VacancyID”, “TimeFrom”,“TimeTo”,“Minutes”, “BreakMinutes”,“RecoveryHours”,“TempShiftOrderReasonID” order 2
psHealthTempShiftPlanInsert	after	insert order 900
psHealthTempShiftPlanUpdate	after	update of “shiftdate”, “timefrom”,“timeto” order 900
ShiftPlan_InsertTrim	before	insert order 2
ShiftPlan_UpdateTrim	before	update order 1
AutoMatchShiftPlanInsertUpdate	after	insert,update order 3

Original SQL

```
-- IQX database structure split by table
-- Source: IQXDatabaseStructure - with comments.sql
-- Table: "pears"."TempShiftPlan"
-- Table comment: Shift requirements. Linked to TempShift when booked /
worked / billed / paid etc.
-- Statement count: 32

CREATE TABLE "pears"."TempShiftPlan" (
    "TempShiftPlanID"                CHAR(20) NOT NULL
   ,"VacancyID"                      CHAR(20) NOT NULL
   ,"EssentialSkill"                 CHAR(15) NULL
   ,"EssentialSkillGradeID"          CHAR(4) NULL
   ,"Description"                    CHAR(50) NOT NULL
   ,"TimeFrom"                       TIME NULL
```

```
    ,"TimeTo"                       TIME NULL
    ,"Moveable"                     SMALLINT NULL DEFAULT 0
    ,"Minutes"                      SMALLINT NULL
    ,"BreakMinutes"                 SMALLINT NULL
    ,"ShiftDate"                    DATE NULL
    ,"ReferenceRequired"            CHAR(1) NULL
    ,"ClientMustConfirm"            SMALLINT NULL DEFAULT 0
    ,"WhenEntered"                  TIMESTAMP NULL DEFAULT CURRENT
TIMESTAMP
    ,"ReferenceCode"                CHAR(20) NULL
    ,"EssentialSkillChoiceList"     CHAR(100) NULL
    ,"TempMustConfirm"              SMALLINT NULL DEFAULT 0
    ,"StaffID"                      CHAR(20) NULL
    ,"OrderedBy"                    CHAR(50) NULL
    ,"TempShiftTypeID"              CHAR(2) NULL
    ,"AnalysisCode"                 CHAR(20) NULL
    ,"ShiftSerialNumber"            BIGINT NULL DEFAULT autoincrement
    ,"TemporaryID"                  CHAR(20) NULL
    ,"CascadeDateTime"              TIMESTAMP NULL
    ,"CascadeLevel"                 SMALLINT NULL DEFAULT 0
    ,"RecoveryHours"                SMALLINT NULL DEFAULT 0
    ,"TempShiftOrderReasonID"       CHAR(20) NULL
    ,"ClientNote"                   CHAR(50) NULL
    ,"TemporaryPersonID"            CHAR(20) NULL
    ,"ColourOverride"               SMALLINT NULL DEFAULT 0
    ,"TempShiftTemplateID"          CHAR(20) NULL
    ,"MasterRosterShiftID"          CHAR(20) NULL
    ,PRIMARY KEY ("TempShiftPlanID" ASC)
)
GO


COMMENT ON COLUMN "pears"."TempShiftPlan"."EssentialSkill" IS
    'P|A;TagID;TagChoiceID;GradeTagID'
GO


COMMENT ON COLUMN "pears"."TempShiftPlan"."Minutes" IS
    'Fill if shorter spell within the timespan'
GO


COMMENT ON COLUMN "pears"."TempShiftPlan"."ReferenceRequired" IS
    'on <E>ntry, <C>onfirmation, <T>imesheet entry. Blank for not required'
GO


COMMENT ON COLUMN "pears"."TempShiftPlan"."TemporaryID" IS
```

```
    'Internal use only'
GO


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


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


COMMENT ON COLUMN "pears"."TempShiftPlan"."TemporaryPersonID" IS
    'Internal use only'
GO


COMMENT ON TABLE "pears"."TempShiftPlan" IS
    'Shift requirements. Linked to TempShift when booked / worked / billed /
paid etc.'
GO


ALTER TABLE "pears"."TempShiftPlan"
    ADD NOT NULL FOREIGN KEY "Vacancy" ("VacancyID" ASC)
    REFERENCES "pears"."vacancy" ("vacancyid")
    ON DELETE CASCADE
GO


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


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


ALTER TABLE "pears"."TempShiftPlan"
    ADD FOREIGN KEY "TempShiftOrderReason" ("TempShiftOrderReasonID" ASC)
    REFERENCES "pears"."TempShiftOrderReason" ("TempShiftOrderReasonID")
```

```
GO


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


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


CREATE INDEX "ShiftPlan_TempID" ON "pears"."TempShiftPlan"
    ( "TemporaryID" )
GO


CREATE INDEX "ShiftPlan_SerialNo" ON "pears"."TempShiftPlan"
    ( "ShiftSerialNumber" )
GO


CREATE INDEX "ShiftPlan_Cascade" ON "pears"."TempShiftPlan"
    ( "CascadeDateTime" )
GO


CREATE INDEX "ShiftPlan_WhenEntered" ON "pears"."TempShiftPlan"
    ( "WhenEntered" )
GO


CREATE INDEX "ShiftPlan_AnalysisCode" ON "pears"."TempShiftPlan"
    ( "AnalysisCode" )
GO


CREATE TRIGGER "ShiftPlan_Insert" BEFORE INSERT ORDER 1 ON
"pears"."TempShiftPlan"
REFERENCING NEW AS "new_shift"
FOR each ROW
WHEN("new_shift"."staffid" IS NULL)
BEGIN
  SET "new_shift"."staffid" = "userstaffid"
```

```
exception
  WHEN others THEN
    SET "new_shift"."staffid" = NULL
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."TempShiftPlan"."ShiftPlan_Insert" IS
{CREATE TRIGGER ShiftPlan_Insert
 BEFORE INSERT ORDER 1 ON
pears.TempShiftPlan
REFERENCING NEW AS new_shift
FOR each ROW
WHEN(new_shift.staffid IS NULL)
BEGIN
  SET new_shift.staffid=userstaffid
exception
  WHEN others THEN
    SET new_shift.staffid=NULL
END
}
GO


CREATE TRIGGER "TempShiftPlanAudit" after UPDATE OF "VacancyID",
"TimeFrom","TimeTo","Minutes",
"BreakMinutes","RecoveryHours","TempShiftOrderReasonID" ORDER 2 ON
"pears"."TempShiftPlan"
REFERENCING OLD AS "OldShift" NEW AS "NewShift"
FOR each ROW
WHEN(EXISTS(SELECT * FROM "AuditItems" WHERE "AreaName" = 'ShiftPlan' AND
"AuditFlag" = 1))
BEGIN
  DECLARE "AuditList" long VARCHAR;
  DECLARE "NewRefCode" CHAR(20);
  SELECT "string"(',',"list"("ItemName"),',') INTO "AuditList" FROM
"AuditItems" WHERE "AreaName" = 'ShiftPlan' AND "AuditFlag" = 1;
  SELECT "RefCode" INTO "NewRefCode" FROM "Vacancy" WHERE
"Vacancy"."VacancyID" = "NewShift"."VacancyID";
  IF "locate"("AuditList",',Moving,') > 0 AND UPDATE("VacancyID") THEN
    CALL "AuditLog"('SHIFTPLAN',"NewShift"."TempShiftPlanID",'Shift Plan '
|| "NewShift"."ShiftSerialNumber" || ' moved between vacancies',
    "string"('Original OurRef: ',"NewRefCode",', Original VacancyID:
',"OldShift"."VacancyID"),
    "string"('New OurRef: ',"NewRefCode",', New VacancyID:
',"NewShift"."VacancyID"))
  END IF;
```

```
  IF "locate"("AuditList",',Times,') > 0 THEN
    IF UPDATE("RecoveryHours") THEN
      CALL "AuditLog"('SHIFTPLAN',"NewShift"."TempShiftPlanID",'Shift Plan '
|| "NewShift"."ShiftSerialNumber" || ' Recovery Hours edited OurRef: ' ||
"NewRefCode",
      "OldShift"."RecoveryHours",
      "NewShift"."RecoveryHours")
    END IF;
    IF UPDATE("Minutes") THEN
      CALL "AuditLog"('SHIFTPLAN',"NewShift"."TempShiftPlanID",'Shift Plan '
|| "NewShift"."ShiftSerialNumber" || ' Minutes edited OurRef: ' ||
"NewRefCode",
      "OldShift"."Minutes",
      "NewShift"."Minutes")
    END IF;
    IF UPDATE("BreakMinutes") THEN
      CALL "AuditLog"('SHIFTPLAN',"NewShift"."TempShiftPlanID",'Shift Plan '
|| "NewShift"."ShiftSerialNumber" || ' Break Minutes edited OurRef: ' ||
"NewRefCode",
      "OldShift"."BreakMinutes",
      "NewShift"."BreakMinutes")
    END IF;
    IF UPDATE("TimeFrom") THEN
      CALL "AuditLog"('SHIFTPLAN',"NewShift"."TempShiftPlanID",'Shift Plan '
|| "NewShift"."ShiftSerialNumber" || ' From edited OurRef: ' ||
"NewRefCode",
      "OldShift"."TimeFrom",
      "NewShift"."TimeFrom")
    END IF;
    IF UPDATE("TimeTo") THEN
      CALL "AuditLog"('SHIFTPLAN',"NewShift"."TempShiftPlanID",'Shift Plan '
|| "NewShift"."ShiftSerialNumber" || ' To edited OurRef: ' || "NewRefCode",
      "OldShift"."TimeTo",
      "NewShift"."TimeTo")
    END IF
  END IF;
  IF "locate"("AuditList",',Order Reason,') > 0 THEN
    IF UPDATE("TempShiftOrderReasonID") THEN
      CALL
"AuditLog"('SHIFTPLAN',"NewShift"."TempShiftPlanID","string"('Shift Plan
',"NewShift"."ShiftSerialNumber",' Our Ref: ',"NewRefCode",'  Order Reason
Changed - ',
      (SELECT "description" FROM "tempshiftorderreason" AS "r" WHERE
"r"."tempshiftorderreasonid" = "OldShift"."TempshiftorderreasonID"),' to ',
      (SELECT "description" FROM "tempshiftorderreason" AS "r" WHERE
"r"."tempshiftorderreasonid" = "newShift"."TempshiftorderreasonID")),
"OldShift"."TempshiftorderreasonID","NewShift"."TempshiftorderreasonID")
    END IF
```

```
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."TempShiftPlan"."TempShiftPlanAudit" IS
{CREATE TRIGGER TempShiftPlanAudit
 after UPDATE OF VacancyID,
TimeFrom,TimeTo,Minutes,
BreakMinutes,RecoveryHours,TempShiftOrderReasonID ORDER 2 ON
pears.TempShiftPlan
REFERENCING OLD AS OldShift NEW AS NewShift
FOR each ROW
WHEN(EXISTS(SELECT * FROM AuditItems WHERE AreaName = 'ShiftPlan' AND
AuditFlag = 1))
BEGIN
  DECLARE AuditList long VARCHAR;
  DECLARE NewRefCode CHAR(20);
  SELECT string(',',list(ItemName),',') INTO AuditList FROM AuditItems WHERE
AreaName = 'ShiftPlan' AND AuditFlag = 1;
  SELECT RefCode INTO NewRefCode FROM Vacancy WHERE Vacancy.VacancyID =
NewShift.VacancyID;
  IF locate(AuditList,',Moving,') > 0 AND UPDATE(VacancyID) THEN
    CALL AuditLog('SHIFTPLAN',NewShift.TempShiftPlanID,'Shift Plan
'||NewShift.ShiftSerialNumber||' moved between vacancies',
    string('Original OurRef: ',NewRefCode,', Original VacancyID:
',OldShift.VacancyID),
    string('New OurRef: ',NewRefCode,', New VacancyID:
',NewShift.VacancyID))
  END IF;
  IF locate(AuditList,',Times,') > 0 THEN
      IF UPDATE(RecoveryHours) THEN
        CALL AuditLog('SHIFTPLAN',NewShift.TempShiftPlanID,'Shift Plan
'||NewShift.ShiftSerialNumber||' Recovery Hours edited OurRef: ' ||
NewRefCode,
        OldShift.RecoveryHours,
        NewShift.RecoveryHours)
      END IF;
      IF UPDATE(Minutes) THEN
        CALL AuditLog('SHIFTPLAN',NewShift.TempShiftPlanID,'Shift Plan
'||NewShift.ShiftSerialNumber||' Minutes edited OurRef: ' || NewRefCode,
        OldShift.Minutes,
        NewShift.Minutes)
      END IF;
      IF UPDATE(BreakMinutes) THEN
        CALL AuditLog('SHIFTPLAN',NewShift.TempShiftPlanID,'Shift Plan
'||NewShift.ShiftSerialNumber||' Break Minutes edited OurRef: ' ||
```

```
NewRefCode,
        OldShift.BreakMinutes,
        NewShift.BreakMinutes)
      END IF;
      IF UPDATE(TimeFrom) THEN
        CALL AuditLog('SHIFTPLAN',NewShift.TempShiftPlanID,'Shift Plan
'||NewShift.ShiftSerialNumber||' From edited OurRef: ' || NewRefCode,
        OldShift.TimeFrom,
        NewShift.TimeFrom)
      END IF;
      IF UPDATE(TimeTo) THEN
        CALL AuditLog('SHIFTPLAN',NewShift.TempShiftPlanID,'Shift Plan
'||NewShift.ShiftSerialNumber||' To edited OurRef: ' || NewRefCode,
        OldShift.TimeTo,
        NewShift.TimeTo)
      END IF;
  END IF;
  IF locate(AuditList,',Order Reason,') > 0 THEN
      IF UPDATE(TempShiftOrderReasonID) THEN
        CALL AuditLog('SHIFTPLAN',NewShift.TempShiftPlanID, string('Shift
Plan ',NewShift.ShiftSerialNumber,' Our Ref: ' ,NewRefCode, '  Order Reason
Changed - ' ,
        (SELECT description FROM tempshiftorderreason r WHERE
r.tempshiftorderreasonid = OldShift.TempshiftorderreasonID), ' to ',
        (SELECT description FROM tempshiftorderreason r WHERE
r.tempshiftorderreasonid = newShift.TempshiftorderreasonID)) ,
        OldShift.TempshiftorderreasonID,   NewShift. TempshiftorderreasonID)
      END IF
  END IF
END
}
GO


CREATE TRIGGER "psHealthTempShiftPlanInsert" after INSERT ORDER 900 ON
"pears"."tempshiftplan"
REFERENCING NEW AS "new_rec"
FOR each ROW
BEGIN
  IF "psHealthCanSendShift"(NULL,"new_rec"."tempshiftplanid") = 1 THEN
    CALL "psHealthInsertUpdateShift"(NULL,"new_rec"."tempshiftplanid")
  END IF
END
GO


CREATE TRIGGER "psHealthTempShiftPlanUpdate" after UPDATE OF "shiftdate",
"timefrom","timeto" ORDER 900 ON "pears"."tempshiftplan"
```

```
REFERENCING NEW AS "new_rec"
FOR each ROW
BEGIN
  IF "psHealthCanSendShift"(NULL,"new_rec"."tempshiftplanid") = 1 THEN
    CALL "psHealthInsertUpdateShift"(NULL,"new_rec"."tempshiftplanid")
  END IF
END
GO


CREATE TRIGGER "ShiftPlan_InsertTrim" BEFORE INSERT ORDER 2 ON
"pears"."TempShiftPlan"
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"."TempShiftPlan"."ShiftPlan_InsertTrim" IS
{CREATE TRIGGER ShiftPlan_InsertTrim
 BEFORE INSERT ORDER 2 ON
pears.TempShiftPlan
REFERENCING NEW AS new_shift
FOR each ROW
BEGIN
  SET new_shift.referencecode = TRIM(new_shift.referencecode)
END
}
GO


CREATE TRIGGER "ShiftPlan_UpdateTrim" BEFORE UPDATE ORDER 1 ON
"pears"."TempShiftPlan"
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"."TempShiftPlan"."ShiftPlan_UpdateTrim" IS
{CREATE TRIGGER ShiftPlan_UpdateTrim
 BEFORE UPDATE ORDER 1 ON
```

```
pears.TempShiftPlan
REFERENCING NEW AS new_shift
FOR each ROW
BEGIN
  SET new_shift.referencecode = TRIM(new_shift.referencecode)
END
}
GO


CREATE TRIGGER "AutoMatchShiftPlanInsertUpdate" after INSERT,UPDATE ORDER 3
ON
"pears"."TempShiftPlan"
REFERENCING NEW AS "New_Shift"
FOR each ROW
WHEN("AutoMatchingEnabled" = 'Y')
BEGIN
  -- This trigger maintains the list of shifts to be checked for needing to
be (re-)matched
  -- DELETE is ignored as referential integrity will deal with that
  IF(SELECT "TempDesk"."AutoMatchOn" FROM "TempDesk" KEY JOIN "Vacancy" AS
"v" WHERE "v"."VacancyID" = "New_Shift"."VacancyID") = 1 THEN
    IF inserting THEN
      CALL
"AutoMatchAddShiftToQueue"("New_Shift"."TempShiftPlanID",'INSERT')
    ELSEIF updating THEN
      CALL
"AutoMatchAddShiftToQueue"("New_Shift"."TempShiftPlanID",'UPDATING')
    END IF
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."TempShiftPlan"."AutoMatchShiftPlanInsertUpdate" IS
{CREATE TRIGGER AutoMatchShiftPlanInsertUpdate
   AFTER INSERT, UPDATE
ORDER 3 ON pears.TempShiftPlan
REFERENCING NEW AS New_Shift
FOR EACH ROW
WHEN (AutoMatchingEnabled = 'Y')
BEGIN
-- This trigger maintains the list of shifts to be checked for needing to be
(re-)matched
-- DELETE is ignored as referential integrity will deal with that
IF (SELECT TempDesk.AutoMatchOn FROM TempDesk KEY JOIN Vacancy v WHERE
v.VacancyID = New_Shift.VacancyID) = 1
```

```
 THEN
    IF INSERTING THEN
        CALL AutoMatchAddShiftToQueue(New_Shift.TempShiftPlanID,'INSERT');
    ELSEIF UPDATING THEN
        CALL AutoMatchAddShiftToQueue(New_Shift.TempShiftPlanID,'UPDATING')
    END IF;
END IF;
END
}
GO
```