pears.contactevent

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

Description

Details of Contacts with candidate & client contacts. Optional links to vacancies - progress etc etc.

Columns

Column	Type	Null	Default	Comment
contacteventid	char(20)	NOT NULL		
personid	char(20)	NULL		
staffid	char(20)	NOT NULL		
vacancyid	char(20)	NULL		
employmentid	char(20)	NULL		
progressid	char(20)	NULL		
placementid	char(20)	NULL		
contactdate	date	NOT NULL		
contacttime	time	NULL		
description	char(100)	NULL		
outcome	char(50)	NULL		
classcode	char(2)	NOT NULL		
who	char(1)	NULL		
notes	long varchar	NULL		
lettertext	long varchar	NULL		
followup	smallint	NULL	0	
wpdoc	char(10)	NULL		
callbackdate	date	NULL		
divisionid	char(20)	NULL		
WhenEntered	timestamp	NULL	current timestamp	
WhoEntered	char(20)	NULL		Auto-entered staffid, but no ref integ because it would break numerous key joins by making them ambiguous
callbacktime	time	NULL		
priority	smallint	NULL	5	1 - high, 5 - low
MIMEMessageID	char(20)	NULL		

Column	Type	Null	Default	Comment
SynetyGUID	char(100)	NULL		
speciality	char(50)	NULL		
forenames	char(30)	NULL		
surname	char(30)	NULL		

Primary Key

- contacteventid

Foreign Keys

Constraint	Columns	References	Delete/update action
person	personid	pears.Person (personid)	
staff	staffid	pears.staff (staffid)	NOT NULL;
progress	progressid	pears.progress (progressid)	ON DELETE SET NULL
placement	placementid	pears.Placement (placementid)	ON DELETE SET NULL
contactclass	classcode	pears.contactclass (classcode)	NOT NULL;
employment	employmentid	pears.employment (employmentid)	ON DELETE SET NULL
division	divisionid	pears.Division (divisionid)	ON DELETE SET NULL
vacancy	vacancyid	pears.vacancy (vacancyid)	
MIMEMessage	MIMEMessageID	pears.MIMEMessage (MIMEMessageID)	ON DELETE SET NULL

Referenced By

Table	Constraint	Columns	Referenced columns
pears.ContactEventEmailLog	ContactEvent	ContactEventID	contacteventid
pears.diary	contactevent	ContactEventID	contacteventid
pears.SentCVs	contactevent	contacteventid	contacteventid

Indexes

Name	Type	Columns	Detail
contactevent_contactdate	Index	contactdate	
Exp1	Index	contactdate, personid	
Exp2	Index	contactdate, contacttime	
Exp3	Index	contactdate, contacttime, staffid	
Test1	Index	staffid	
contactevent_callbackdate	Index	callbackdate	
contactevent_StaffContactDate	Index	staffid, contactdate	
contactevent_staffCallbackdate	Index	staffid, callbackdate	
contactevent_WhenEntered	Index	WhenEntered	

Triggers

Name	Timing	Event
ContactEvent_Insert	before	insert order 5
ContactEvent_Insert2	before	insert order 6
ContactEventAudit	before	update of “callbackdate”, “classcode”,“description” order 1

Original SQL

```
-- IQX database structure split by table
-- Source: IQXDatabaseStructure - with comments.sql
-- Table: "pears"."contactevent"
-- Table comment: Details of Contacts with candidate & client contacts.
Optional links to vacancies - progress etc etc.
-- Statement count: 28

CREATE TABLE "pears"."contactevent" (
    "contacteventid"                CHAR(20) NOT NULL
   ,"personid"                      CHAR(20) NULL
   ,"staffid"                       CHAR(20) NOT NULL
   ,"vacancyid"                     CHAR(20) NULL
   ,"employmentid"                  CHAR(20) NULL
   ,"progressid"                    CHAR(20) NULL
   ,"placementid"                   CHAR(20) NULL
   ,"contactdate"                   DATE NOT NULL
   ,"contacttime"                   TIME NULL
   ,"description"                   CHAR(100) NULL
   ,"outcome"                       CHAR(50) NULL
   ,"classcode"                     CHAR(2) NOT NULL
   ,"who"                           CHAR(1) NULL
   ,"notes"                         long VARCHAR NULL
   ,"lettertext"                    long VARCHAR NULL
   ,"followup"                      SMALLINT NULL DEFAULT 0
   ,"wpdoc"                         CHAR(10) NULL
   ,"callbackdate"                  DATE NULL
   ,"divisionid"                    CHAR(20) NULL
   ,"WhenEntered"                   TIMESTAMP NULL DEFAULT CURRENT
TIMESTAMP
   ,"WhoEntered"                    CHAR(20) NULL
   ,"callbacktime"                  TIME NULL
   ,"priority"                      SMALLINT NULL DEFAULT 5
   ,"MIMEMessageID"                 CHAR(20) NULL
   ,"SynetyGUID"                    CHAR(100) NULL
   ,"speciality"                    CHAR(50) NULL
   ,"forenames"                     CHAR(30) NULL
   ,"surname"                       CHAR(30) NULL
```

```
   ,PRIMARY KEY ("contacteventid" ASC)
)
GO


COMMENT ON COLUMN "pears"."contactevent"."WhoEntered" IS
    'Auto-entered staffid, but no ref integ because it would break numerous
key joins by making them ambiguous'
GO


COMMENT ON COLUMN "pears"."contactevent"."priority" IS
    '1 - high, 5 - low'
GO


COMMENT ON TABLE "pears"."contactevent" IS
    'Details of Contacts with candidate & client contacts. Optional links to
vacancies - progress etc etc.'
GO


ALTER TABLE "pears"."contactevent"
    ADD FOREIGN KEY "person" ("personid" ASC)
    REFERENCES "pears"."Person" ("personid")
GO


ALTER TABLE "pears"."contactevent"
    ADD NOT NULL FOREIGN KEY "staff" ("staffid" ASC)
    REFERENCES "pears"."staff" ("staffid")
GO


ALTER TABLE "pears"."contactevent"
    ADD FOREIGN KEY "progress" ("progressid" ASC)
    REFERENCES "pears"."progress" ("progressid")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."contactevent"
    ADD FOREIGN KEY "placement" ("placementid" ASC)
    REFERENCES "pears"."Placement" ("placementid")
    ON DELETE SET NULL
GO
```

```
ALTER TABLE "pears"."contactevent"
    ADD NOT NULL FOREIGN KEY "contactclass" ("classcode" ASC)
    REFERENCES "pears"."contactclass" ("classcode")
GO


ALTER TABLE "pears"."contactevent"
    ADD FOREIGN KEY "employment" ("employmentid" ASC)
    REFERENCES "pears"."employment" ("employmentid")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."contactevent"
    ADD FOREIGN KEY "division" ("divisionid" ASC)
    REFERENCES "pears"."Division" ("divisionid")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."contactevent"
    ADD FOREIGN KEY "vacancy" ("vacancyid" ASC)
    REFERENCES "pears"."vacancy" ("vacancyid")
GO


ALTER TABLE "pears"."contactevent"
    ADD FOREIGN KEY "MIMEMessage" ("MIMEMessageID" ASC)
    REFERENCES "pears"."MIMEMessage" ("MIMEMessageID")
    ON DELETE SET NULL
GO


CREATE INDEX "contactevent_contactdate" ON "pears"."contactevent"
    ( "contactdate" DESC )
GO


CREATE INDEX "Exp1" ON "pears"."contactevent"
    ( "contactdate" DESC,"personid" )
GO


CREATE INDEX "Exp2" ON "pears"."contactevent"
    ( "contactdate" DESC,"contacttime" DESC )
GO
```

```
CREATE INDEX "Exp3" ON "pears"."contactevent"
    ( "contactdate" DESC,"contacttime" DESC,"staffid" )
GO


CREATE INDEX "Test1" ON "pears"."contactevent"
    ( "staffid" )
GO


CREATE INDEX "contactevent_callbackdate" ON "pears"."contactevent"
    ( "callbackdate" )
GO


CREATE INDEX "contactevent_StaffContactDate" ON "pears"."contactevent"
    ( "staffid","contactdate" DESC )
GO


CREATE INDEX "contactevent_staffCallbackdate" ON "pears"."contactevent"
    ( "staffid","callbackdate" )
GO


CREATE INDEX "contactevent_WhenEntered" ON "pears"."contactevent"
    ( "WhenEntered" )
GO


CREATE TRIGGER "ContactEvent_Insert" BEFORE INSERT ORDER 5 ON
"pears"."contactevent"
REFERENCING NEW AS "new_ce"
FOR each ROW
WHEN("new_ce"."whoentered" IS NULL)
BEGIN
  SET "new_ce"."whoentered" = "userstaffid"
exception
  WHEN others THEN
    SET "new_ce"."whoentered" = NULL
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."contactevent"."ContactEvent_Insert" IS
{CREATE TRIGGER ContactEvent_Insert
 BEFORE INSERT ORDER 5 ON
```

```
pears.contactevent
REFERENCING NEW AS new_ce
FOR each ROW
WHEN(new_ce.whoentered IS NULL)
BEGIN
  SET new_ce.whoentered=userstaffid
exception
  WHEN others THEN
    SET new_ce.whoentered=NULL
END
}
GO


CREATE TRIGGER "ContactEvent_Insert2" BEFORE INSERT ORDER 6 ON
"pears"."contactevent"
REFERENCING NEW AS "new_ce"
FOR each ROW
BEGIN
  DECLARE "newwhen" TIMESTAMP;
  SET "newwhen" = "new_ce"."contactdate"+"new_ce"."contacttime";
  IF "newwhen" < CURRENT DATE+10 THEN
    IF "new_ce"."personid" IS NOT NULL THEN UPDATE "person" SET
"lastcontactevent" = "newwhen"
        WHERE "personid" = "new_ce"."personid" AND "newwhen" >
"isnull"("lastcontactevent",'2001-01-01')
    END IF;
    IF "new_ce"."vacancyid" IS NOT NULL THEN UPDATE "vacancy" SET
"lastcontactevent" = "newwhen"
        WHERE "vacancyid" = "new_ce"."vacancyid" AND "newwhen" >
"isnull"("lastcontactevent",'2001-01-01')
    END IF;
    IF "new_ce"."placementid" IS NOT NULL THEN UPDATE "placement" SET
"lastcontactevent" = "newwhen"
        WHERE "placementid" = "new_ce"."placementid" AND "newwhen" >
"isnull"("lastcontactevent",'2001-01-01')
    END IF;
    IF "new_ce"."employmentid" IS NOT NULL THEN UPDATE "employment" SET
"lastcontactevent" = "newwhen"
        WHERE "employmentid" = "new_ce"."employmentid" AND "newwhen" >
"isnull"("lastcontactevent",'2001-01-01');
      UPDATE "company" KEY JOIN "employment" SET
"company"."lastcontactevent" = "newwhen"
        WHERE "employmentid" = "new_ce"."employmentid" AND "newwhen" >
"isnull"("company"."lastcontactevent",'2001-01-01');
      UPDATE "person" KEY JOIN "employment" SET "person"."lastcontactevent"
= "newwhen"
        WHERE "employmentid" = "new_ce"."employmentid" AND "newwhen" >
```

```
"isnull"("person"."lastcontactevent",'2001-01-01')
    END IF END IF;
  IF "WPKMaintainGetSwitchValue"('CESTAFF','','L') = 'N' THEN
    SET "new_ce"."divisionid" = "isnull"((SELECT "divisionid" FROM "person"
WHERE "personid" = "new_ce"."personid"),
      (SELECT "company"."divisionid" FROM "company" KEY JOIN "employment"
WHERE "employmentid" = "new_ce"."employmentid"),
      (SELECT "company"."divisionid" FROM "vacancy" KEY JOIN "employment"
KEY JOIN "company" WHERE "vacancyid" = "new_ce"."vacancyid"),
      (SELECT "company"."divisionid" FROM "placement" KEY JOIN "vacancy" KEY
JOIN "employment" KEY JOIN "company" WHERE "placementid" =
"new_ce"."placementid"))
  END IF;
  IF "new_ce"."personid" IS NOT NULL AND "new_ce"."divisionid" IS NOT NULL
THEN
    IF "new_ce"."classcode" = any(SELECT "classcode" FROM
"ContClassPersonDivFilter" WHERE "divisionid" = "new_ce"."divisionid") THEN
      UPDATE "person" SET "FilteredLastContactEvent" = "newwhen" WHERE
"personid" = "new_ce"."personid" AND "newwhen" >
"isnull"("FilteredLastContactEvent",'2001-01-01')
    END IF END IF;
  IF "WPKUserPerformsRole"("new_ce"."whoentered",'CEAVAILALL') = 1 THEN
    SET "new_ce"."divisionid" = NULL
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."contactevent"."ContactEvent_Insert2" IS
{CREATE TRIGGER ContactEvent_Insert2
 BEFORE INSERT ORDER 6 ON
pears.contactevent
REFERENCING NEW AS new_ce
FOR each ROW
BEGIN
  DECLARE newwhen TIMESTAMP;
  SET newwhen = new_ce.contactdate+new_ce.contacttime;
  IF newwhen < CURRENT DATE+10 THEN
    IF new_ce.personid IS NOT NULL THEN UPDATE person SET lastcontactevent =
newwhen
        WHERE personid = new_ce.personid AND newwhen >
isnull(lastcontactevent,'2001-01-01')
    END IF;
    IF new_ce.vacancyid IS NOT NULL THEN UPDATE vacancy SET lastcontactevent
= newwhen
        WHERE vacancyid = new_ce.vacancyid AND newwhen >
isnull(lastcontactevent,'2001-01-01')
```

```
    END IF;
    IF new_ce.placementid IS NOT NULL THEN UPDATE placement SET
lastcontactevent = newwhen
        WHERE placementid = new_ce.placementid AND newwhen >
isnull(lastcontactevent,'2001-01-01')
    END IF;
    IF new_ce.employmentid IS NOT NULL THEN UPDATE employment SET
lastcontactevent = newwhen
        WHERE employmentid = new_ce.employmentid AND newwhen >
isnull(lastcontactevent,'2001-01-01');
      UPDATE company KEY JOIN employment SET company.lastcontactevent =
newwhen
        WHERE employmentid = new_ce.employmentid AND newwhen >
isnull(company.lastcontactevent,'2001-01-01');
      UPDATE person KEY JOIN employment SET person.lastcontactevent =
newwhen
        WHERE employmentid = new_ce.employmentid AND newwhen >
isnull(person.lastcontactevent,'2001-01-01')
    END IF
  END IF;
  IF WPKMaintainGetSwitchValue('CESTAFF','','L') = 'N' THEN
    SET new_ce.divisionid = isnull((SELECT divisionid FROM person WHERE
personid = new_ce.personid),
      (SELECT company.divisionid FROM company KEY JOIN employment WHERE
employmentid = new_ce.employmentid),
      (SELECT company.divisionid FROM vacancy KEY JOIN employment KEY JOIN
company WHERE vacancyid = new_ce.vacancyid),
      (SELECT company.divisionid FROM placement KEY JOIN vacancy KEY JOIN
employment KEY JOIN company WHERE placementid = new_ce.placementid))
  END IF;
  IF new_ce.personid IS NOT NULL AND new_ce.divisionid IS NOT NULL THEN
    IF new_ce.classcode = any(SELECT classcode FROM ContClassPersonDivFilter
WHERE divisionid = new_ce.divisionid) THEN
      UPDATE person SET FilteredLastContactEvent = newwhen WHERE personid =
new_ce.personid AND newwhen > isnull(FilteredLastContactEvent,'2001-01-01')
    END IF
  END IF;
  IF WPKUserPerformsRole(new_ce.whoentered,'CEAVAILALL') = 1 THEN
    SET new_ce.divisionid = NULL
  END IF
END
}
GO


CREATE TRIGGER "ContactEventAudit" BEFORE UPDATE OF "callbackdate",
"classcode","description" ORDER 1 ON
"pears"."contactevent"
```

```
REFERENCING OLD AS "old_ce" NEW AS "new_ce"
FOR each ROW
WHEN(EXISTS(SELECT * FROM "AuditItems" WHERE "AreaName" = 'Contact Event'
AND "AuditFlag" = 1))
BEGIN
  DECLARE "AuditList" long VARCHAR;
  DECLARE "OldDescrip" CHAR(10);
  DECLARE "NewDescrip" CHAR(10);
  DECLARE "StaffName" CHAR(25);
  SELECT "string"(',',"list"("ItemName"),',') INTO "AuditList" FROM
"AuditItems" WHERE "AreaName" = 'Contact Event' AND "AuditFlag" = 1;
  IF "locate"("AuditList",',Callback Date,') > 0 AND UPDATE("callbackdate")
THEN
    SELECT "old_ce"."callbackdate" INTO "OldDescrip";
    SELECT "new_ce"."callbackdate" INTO "NewDescrip";
    SELECT "userid" INTO "staffname" FROM "staff" WHERE "staffid" =
"old_ce"."staffid";
    CALL
"AuditLog"('CEVENT',"old_ce"."contacteventid","string"("old_ce"."description
",' Updated - Callback Date ',"old_ce"."contactdate",' by
','"staffname"),"OldDescrip","NewDescrip")
  END IF;
  IF "locate"("AuditList",',Type,') > 0 AND UPDATE("classcode") THEN
    SELECT "classdescrip" INTO "OldDescrip" FROM "contactclass" WHERE
"classcode" = "old_ce"."classcode";
    SELECT "classdescrip" INTO "NewDescrip" FROM "contactclass" WHERE
"classcode" = "new_ce"."classcode";
    SELECT "userid" INTO "staffname" FROM "staff" WHERE "staffid" =
"old_ce"."staffid";
    CALL
"AuditLog"('CEVENT',"old_ce"."contacteventid","string"("old_ce"."description
",' Updated - Type ',"olddescrip",' by
','"staffname"),"OldDescrip","NewDescrip")
  END IF;
  IF "locate"("AuditList",',Summary,') > 0 AND UPDATE("description") THEN
    SELECT "old_ce"."description" INTO "OldDescrip";
    SELECT "new_ce"."description" INTO "NewDescrip";
    SELECT "userid" INTO "staffname" FROM "staff" WHERE "staffid" =
"old_ce"."staffid";
    CALL
"AuditLog"('CEVENT',"old_ce"."contacteventid","string"("old_ce"."description
",' Updated - Summary ',"olddescrip",' by
','"staffname"),"OldDescrip","NewDescrip")
  END IF
END
GO
```

```
COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."contactevent"."ContactEventAudit" IS
{CREATE TRIGGER ContactEventAudit
 BEFORE UPDATE OF callbackdate,classcode,description,
 ORDER 1 ON pears.contactevent
REFERENCING OLD AS old_ce NEW AS new_ce
FOR each ROW
WHEN(EXISTS(SELECT* FROM AuditItems WHERE AreaName = 'Contact Event' AND
AuditFlag = 1))
BEGIN
  DECLARE AuditList long VARCHAR;
  DECLARE OldDescrip CHAR(10);
  DECLARE NewDescrip CHAR(10);
  DECLARE StaffName CHAR(25);
  SELECT string(',',list(ItemName),',') INTO AuditList FROM AuditItems WHERE
AreaName = 'Contact Event' AND AuditFlag = 1;
  IF locate(AuditList,',Callback Date,') > 0 AND UPDATE(callbackdate) THEN
    SELECT old_ce.callbackdate INTO OldDescrip ;
    SELECT new_ce.callbackdate INTO NewDescrip ;
    SELECT userid INTO staffname FROM staff WHERE staffid = old_ce.staffid ;
    CALL AuditLog('CEVENT',old_ce.contacteventid,string(old_ce.description,
' Updated - Callback Date ',old_ce.contactdate,' by
',staffname),OldDescrip,NewDescrip)
  END IF;
  IF locate(AuditList,',Type,') > 0 AND UPDATE(classcode) THEN
    SELECT classdescrip INTO OldDescrip FROM contactclass WHERE classcode =
old_ce.classcode ;
    SELECT classdescrip INTO NewDescrip FROM contactclass WHERE classcode =
new_ce.classcode ;
    SELECT userid INTO staffname FROM staff WHERE staffid = old_ce.staffid ;
    CALL AuditLog('CEVENT',old_ce.contacteventid,string(old_ce.description,
' Updated - Type ',olddescrip,' by ',staffname),OldDescrip,NewDescrip)
  END IF;
  IF locate(AuditList,',Summary,') > 0 AND UPDATE(description) THEN
    SELECT old_ce.description INTO OldDescrip ;
    SELECT new_ce.description INTO NewDescrip ;
    SELECT userid INTO staffname FROM staff WHERE staffid = old_ce.staffid ;
    CALL AuditLog('CEVENT',old_ce.contacteventid,string(old_ce.description,
' Updated - Summary ',olddescrip,' by ',staffname),OldDescrip,NewDescrip)
  END IF;
END
}
GO
```