# pears.employment

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

## Description

Link between Person and Company defining employment.

## Columns

| Column | Type | Null | Default | Comment |
|---|---|---|---|---|
| temp | smallint | NOT NULL | 0 | |
| **employmentid** | char(20) | NOT NULL | | |
| personid | char(20) | NOT NULL | | |
| companyid | char(20) | NOT NULL | | |
| startdate | date | NULL | | |
| leavedate | date | NULL | | |
| noreemploy | smallint | NULL | | |
| note | char(100) | NULL | | |
| salary | numeric(12,2) | NULL | | |
| position | char(50) | NULL | | |
| department | char(30) | NULL | | |
| Concurrent | smallint | NULL | 0 | 1 used by shift placements to prevent availability blocks |
| ExtendedNotes | long varchar | NULL | | |
| lastcontactevent | timestamp | NULL | | Added V2.2.2.13 Will be null for existing employment |
| UnsubscribeToMarketing | tinyint | NULL | 0 | |
| ownerstaffid | char(20) | NULL | | |

## Primary Key

- employmentid

## Foreign Keys

| Constraint | Columns | References | Delete/update action |
|---|---|---|---|
| person | personid | pears.Person (personid) | NOT NULL; |
| company | companyid | pears.Company (companyid) | NOT NULL; |

## Referenced By

| Table | Constraint | Columns | Referenced columns |
|---|---|---|---|
| pears.CompanyAccount | Employment | AccountsContact | employmentid |
| pears.CompanyAccount | Employment2 | TimesheetContact | employmentid |
| pears.CompanySDS | employment | EmploymentID | employmentid |
| pears.contactevent | employment | employmentid | employmentid |
| pears.diary | employment | employmentid | employmentid |
| pears.ETip | Employment | EmploymentID | employmentid |
| pears.IQXNetUserLink | Employment | EmploymentID | employmentid |
| pears.Placement | employment | employmentid | employmentid |
| pears.vacancy | employment | employmentid | employmentid |
| pears.VacancyRoleAllocation | employment | EmploymentID | employmentid |

## Indexes

| Name | Type | Columns | Detail |
|---|---|---|---|
| employment_leavedate | Index | leavedate | |
| person_lastcontactevent | Index | lastcontactevent | |

## Triggers

| Name | Timing | Event |
|---|---|---|
| employmentleaveupdate | after | update of “leavedate” order 1 |
| EmploymentAudit | before | update of “leavedate”, “startdate”,“position”,“salary”,“note”,“extendednotes”, “UnsubscribeToMarketing” order 1 |
| EmploymentUnsubscribe | after | update of “UnsubscribeToMarketing” order 2501 |
| EmploymentAudit_insert | before | insert order 1 |

## Original SQL

```
-- IQX database structure split by table
-- Source: IQXDatabaseStructure - with comments.sql
-- Table: "pears"."employment"
-- Table comment: Link between Person and Company defining employment.
-- Statement count: 16
```

```
CREATE TABLE "pears"."employment" (
    "temp"                          SMALLINT NOT NULL DEFAULT 0
   ,"employmentid"                  CHAR(20) NOT NULL
   ,"personid"                      CHAR(20) NOT NULL
   ,"companyid"                     CHAR(20) NOT NULL
   ,"startdate"                     DATE NULL
   ,"leavedate"                     DATE NULL
   ,"noreemploy"                    SMALLINT NULL
   ,"note"                          CHAR(100) NULL
   ,"salary"                        NUMERIC(12,2) NULL
   ,"position"                      CHAR(50) NULL
   ,"department"                    CHAR(30) NULL
   ,"Concurrent"                    SMALLINT NULL DEFAULT 0
   ,"ExtendedNotes"                 long VARCHAR NULL
   ,"lastcontactevent"              TIMESTAMP NULL
   ,"UnsubscribeToMarketing"        tinyint NULL DEFAULT 0
   ,"ownerstaffid"                  CHAR(20) NULL
   ,PRIMARY KEY ("employmentid" ASC)
)
GO


COMMENT ON COLUMN "pears"."employment"."Concurrent" IS
    '1 used by shift placements to prevent availability blocks'
GO


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


COMMENT ON TABLE "pears"."employment" IS
    'Link between Person and Company defining employment.'
GO


ALTER TABLE "pears"."employment"
    ADD NOT NULL FOREIGN KEY "person" ("personid" ASC)
    REFERENCES "pears"."Person" ("personid")
GO


ALTER TABLE "pears"."employment"
    ADD NOT NULL FOREIGN KEY "company" ("companyid" ASC)
    REFERENCES "pears"."Company" ("companyid")
GO
```

```
CREATE INDEX "employment_leavedate" ON "pears"."employment"
    ( "leavedate" DESC )
GO


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


CREATE TRIGGER "employmentleaveupdate" after UPDATE OF "leavedate"
ORDER 1 ON "pears"."employment"
REFERENCING OLD AS "old_emp" NEW AS "new_emp"
FOR each ROW
BEGIN
  DECLARE "idum" SMALLINT;
  BEGIN
    UPDATE "placement" AS "l" JOIN "sync_row" AS "s" ON "s"."record_type" =
'L' AND "s"."record_id" = "l"."placementid" SET "s"."batch" = 0 WHERE
"l"."employmentid" = "new_emp"."employmentid" AND "s"."batch" > 0
  exception
    WHEN others THEN SET "idum" = 0
  END;
  INSERT INTO "placementextension"(
"placementextensionid","placementid","leavedate","startdate","whochanged","w
henchanged","notes","oldleavedate" )
    SELECT
"uniquekey"("placementid"),"placementid","new_emp"."leavedate","new_emp"."st
artdate","userstaffid",CURRENT TIMESTAMP,"string"('leavedate
changed'),"old_emp"."leavedate"
      FROM "placement" AS "l" WHERE "l"."employmentid" =
"new_emp"."employmentid" AND "new_emp"."leavedate" <> "old_emp"."leavedate"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."employment"."employmentleaveupdate" IS
{CREATE TRIGGER employmentleaveupdate
 after UPDATE OF leavedate
ORDER 1 ON pears.employment
REFERENCING OLD AS old_emp NEW AS new_emp
FOR each ROW
BEGIN
  DECLARE idum SMALLINT;
  BEGIN
```

```
    UPDATE placement AS l JOIN sync_row AS s ON s.record_type = 'L' AND
s.record_id = l.placementid SET s.batch = 0 WHERE l.employmentid =
new_emp.employmentid AND s.batch > 0;
  exception
    WHEN others THEN SET idum=0
  END;
  INSERT INTO placementextension(placementextensionid, placementid,
leavedate, startdate,  whochanged, whenchanged,notes, oldleavedate)
  SELECT uniquekey(placementid), placementid, new_emp.leavedate,
new_emp.startdate, userstaffid, CURRENT TIMESTAMP, string('leavedate
changed') ,  old_emp.leavedate
  FROM placement l WHERE l.employmentid = new_emp.employmentid  AND
new_emp.leavedate != old_emp.leavedate
END
}
GO


CREATE TRIGGER "EmploymentAudit" BEFORE UPDATE OF "leavedate",
"startdate","position","salary","note","extendednotes",
"UnsubscribeToMarketing" ORDER 1 ON "pears"."Employment"
REFERENCING OLD AS "old_emp" NEW AS "new_emp"
FOR each ROW
WHEN(EXISTS(SELECT * FROM "AuditItems" WHERE(("AreaName" = 'Company Contact'
AND "AuditFlag" = 1) OR("AreaName" = 'Placement' AND "AuditFlag" = 1)
OR("AreaName" = 'Person History' AND "AuditFlag" = 1))))
BEGIN
  DECLARE @AuditList long VARCHAR;
  DECLARE @ContactName CHAR(250);
  DECLARE @CName CHAR(250);
  DECLARE @Placement CHAR(20);
  DECLARE @Company CHAR(20);
  DECLARE @OurRef CHAR(20);
  DECLARE @History CHAR(1);
  SET @ContactName = (SELECT "name" FROM "person" WHERE "personid" =
"old_emp"."personid");
  SET @Placement = (SELECT FIRST "placementid" FROM "placement" WHERE
"employmentid" = "old_emp"."employmentid");
  SET @OurRef = (SELECT FIRST "refcode" FROM "placement" WHERE
"employmentid" = "old_emp"."employmentid");
  SET @Company = (SELECT "Companyid" FROM "Company" WHERE "Companyid" =
"old_emp"."Companyid");
  SET @CName = (SELECT "name" FROM "Company" WHERE "Companyid" =
"old_emp"."Companyid");
  IF EXISTS(SELECT * FROM "person" WHERE "personid" = "old_emp"."personid"
AND "status" = 'L') THEN
    SET @History = NULL
  ELSE SET @History = 'Y'
```

```
  END IF;
  SELECT "string"(',',"list"("ItemName"),',') INTO @AuditList FROM
"AuditItems" WHERE(("AreaName" = 'Company Contact' AND "AuditFlag" = 1)
OR("AreaName" = 'Placement' AND "AuditFlag" = 1) OR("AreaName" = 'Person
History' AND "AuditFlag" = 1));
  -- Leave Date
  IF "locate"(@AuditList,',Leave Date,') > 0 AND UPDATE("LeaveDate")
AND(@Placement IS NULL) AND(@History IS NULL) THEN
    CALL "AuditLog"('COMPANY',@Company,"string"('Leave Date Updated -
',@ContactName),"old_emp"."leavedate","new_emp"."leavedate")
  END IF;
  IF "locate"(@AuditList,',To,') > 0 AND UPDATE("LeaveDate") AND(@History =
'Y') THEN
    CALL "AuditLog"('PERSON',"old_emp"."personid","string"(@CName,' History
Leave Date Updated - '),"old_emp"."leavedate","new_emp"."leavedate")
  END IF;
  -- Start Date
  IF "locate"(@AuditList,',Start Date,') > 0 AND UPDATE("StartDate")
AND(@Placement IS NULL) AND(@History IS NULL) THEN
    CALL "AuditLog"('COMPANY',@Company,"string"('Start Date Updated -
',@ContactName),"old_emp"."startdate","new_emp"."startdate")
  END IF;
  IF "locate"(@AuditList,',To,') > 0 AND UPDATE("StartDate") AND(@History =
'Y') THEN
    CALL "AuditLog"('PERSON',"old_emp"."personid","string"(@CName,' History
Start Date Updated - '),"old_emp"."startdate","new_emp"."startdate")
  END IF;
  IF "locate"(@AuditList,',Placement Leave Date,') > 0 AND
UPDATE("LeaveDate") AND(@Placement IS NOT NULL) THEN
    CALL "AuditLog"('PLACEMENT',@Placement,"string"('Leave Date Updated Our
Ref. - ',@OurRef),"old_emp"."leavedate","new_emp"."leavedate")
  END IF;
  -- Start Date
  IF "locate"(@AuditList,',Placement Start Date,') > 0 AND
UPDATE("StartDate") AND(@Placement IS NOT NULL) THEN
    CALL "AuditLog"('PLACEMENT',@Placement,"string"('Start Date Updated Our
Ref. - ',@OurRef),"old_emp"."startdate","new_emp"."startdate")
  END IF;
  -- Job Title
  IF "locate"(@AuditList,',Job Title,') > 0 AND UPDATE("position")
AND(@Placement IS NULL) AND(@History IS NULL) THEN
    CALL "AuditLog"('COMPANY',@Company,"string"('Job Title Updated -
',@ContactName),"old_emp"."position","new_emp"."position")
  END IF;
  IF "locate"(@AuditList,',Job Title,') > 0 AND UPDATE("position")
AND(@History = 'Y') THEN
    CALL "AuditLog"('PERSON',"old_emp"."personid","string"(@CName,' History
Job Title Updated - '),"old_emp"."position","new_emp"."position")
```

```
  END IF;
  IF "locate"(@AuditList,',Salary,') > 0 AND UPDATE("salary") AND(@History =
'Y') THEN
    CALL "AuditLog"('PERSON',"old_emp"."personid","string"(@CName,' History
Salary Updated - '),"old_emp"."salary","new_emp"."salary")
  END IF;
  IF "locate"(@AuditList,',Note,') > 0 AND UPDATE("note") AND(@History =
'Y') THEN
    CALL "AuditLog"('PERSON',"old_emp"."personid","string"(@CName,' History
Note Updated - '),"old_emp"."note","new_emp"."note")
  END IF;
  IF "locate"(@AuditList,',Details,') > 0 AND UPDATE("ExtendedNotes")
AND(@History = 'Y') THEN
    CALL "AuditLog"('PERSON',"old_emp"."personid","string"(@CName,' History
Details Updated - '),"old_emp"."ExtendedNotes","new_emp"."ExtendedNotes")
  END IF;
  -- Unsubscribe To Marketing
  IF "locate"(@AuditList,',Unsubscribe To Marketing,') > 0 AND
UPDATE("UnsubscribeToMarketing") THEN
    CALL "AuditLog"('PERSON',"old_emp"."personid","string"('Unsubscribe To
Marketing - ',@ContactName,' -
',@CNAME),"string"("old_emp"."UnsubscribeToMarketing"),"string"("new_emp"."U
nsubscribeToMarketing"))
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."employment"."EmploymentAudit"
IS
{CREATE TRIGGER EmploymentAudit
 BEFORE UPDATE OF leavedate, startdate, POSITION, salary, note,
extendednotes,UnsubscribeToMarketing
 ORDER 1 ON pears.Employment
REFERENCING OLD AS old_emp NEW AS new_emp
FOR each ROW
WHEN(EXISTS(SELECT * FROM AuditItems WHERE (AreaName = 'Company Contact' AND
AuditFlag = 1) OR (AreaName = 'Placement' AND AuditFlag = 1) OR (AreaName =
'Person History' AND AuditFlag = 1)))
BEGIN
  DECLARE @AuditList long VARCHAR;
  DECLARE @ContactName CHAR(250);
  DECLARE @CName CHAR(250);
  DECLARE @Placement CHAR(20);
  DECLARE @Company CHAR(20);
  DECLARE @OurRef CHAR(20);
  DECLARE @History CHAR(1);
  SET @ContactName=(SELECT name FROM person WHERE personid =
```

```
old_emp.personid);
  SET @Placement=(SELECT FIRST placementid FROM placement WHERE employmentid
= old_emp.employmentid);
  SET @OurRef=(SELECT FIRST refcode FROM placement WHERE employmentid =
old_emp.employmentid);
  SET @Company=(SELECT Companyid FROM Company WHERE Companyid =
old_emp.Companyid);
  SET @CName=(SELECT name FROM Company WHERE Companyid = old_emp.Companyid);
  IF EXISTS(SELECT * FROM person WHERE personid = old_emp.personid AND
STATUS = 'L' )
    THEN SET @History = NULL
    ELSE SET @History = 'Y'
  END IF;
  SELECT string(',',list(ItemName),',') INTO @AuditList FROM AuditItems
WHERE (AreaName = 'Company Contact' AND AuditFlag = 1) OR (AreaName =
'Placement' AND AuditFlag = 1) OR (AreaName = 'Person History' AND AuditFlag
= 1);
  -- Leave Date
  IF locate(@AuditList,',Leave Date,') > 0 AND UPDATE(LeaveDate)  AND
(@Placement IS NULL) AND (@History IS NULL) THEN
    CALL AuditLog('COMPANY',@Company,string('Leave Date Updated -
',@ContactName),old_emp.leavedate,new_emp.leavedate)
  END IF;
  IF locate(@AuditList,',To,') > 0 AND UPDATE(LeaveDate) AND (@History =
'Y') THEN
    CALL AuditLog('PERSON',old_emp.personid,string(@CName,' History Leave
Date Updated - '),old_emp.leavedate,new_emp.leavedate)
  END IF;
  -- Start Date
  IF locate(@AuditList,',Start Date,') > 0 AND UPDATE(StartDate) AND
(@Placement IS NULL) AND (@History IS NULL) THEN
    CALL AuditLog('COMPANY',@Company,string('Start Date Updated -
',@ContactName),old_emp.startdate,new_emp.startdate)
  END IF;
  IF locate(@AuditList,',To,') > 0 AND UPDATE(StartDate) AND (@History =
'Y') THEN
    CALL AuditLog('PERSON',old_emp.personid,string(@CName,' History Start
Date Updated - '),old_emp.startdate,new_emp.startdate)
  END IF;
  IF locate(@AuditList,',Placement Leave Date,') > 0 AND UPDATE(LeaveDate)
AND (@Placement IS NOT NULL) THEN
    CALL AuditLog('PLACEMENT',@Placement,string('Leave Date Updated Our Ref.
- ',@OurRef),old_emp.leavedate,new_emp.leavedate)
  END IF;
  -- Start Date
  IF locate(@AuditList,',Placement Start Date,') > 0 AND UPDATE(StartDate)
AND (@Placement IS NOT NULL) THEN
    CALL AuditLog('PLACEMENT',@Placement,string('Start Date Updated Our Ref.
```

```
- ',@OurRef),old_emp.startdate,new_emp.startdate)
  END IF;
  -- Job Title
  IF locate(@AuditList,',Job Title,') > 0 AND UPDATE(POSITION) AND
(@Placement IS NULL) AND (@History IS NULL) THEN
    CALL AuditLog('COMPANY',@Company,string('Job Title Updated -
',@ContactName),old_emp.position,new_emp.position)
  END IF;
  IF locate(@AuditList,',Job Title,') > 0 AND UPDATE(POSITION) AND (@History
= 'Y') THEN
    CALL AuditLog('PERSON',old_emp.personid,string(@CName,' History Job
Title Updated - '),old_emp.position,new_emp.position)
  END IF;
  IF locate(@AuditList,',Salary,') > 0 AND UPDATE(salary) AND (@History =
'Y') THEN
    CALL AuditLog('PERSON',old_emp.personid,string(@CName,' History Salary
Updated - '),old_emp.salary,new_emp.salary)
  END IF;
  IF locate(@AuditList,',Note,') > 0 AND UPDATE(note) AND (@History = 'Y')
THEN
    CALL AuditLog('PERSON',old_emp.personid,string(@CName,' History Note
Updated - '),old_emp.note,new_emp.note)
  END IF;
  IF locate(@AuditList,',Details,') > 0 AND UPDATE(ExtendedNotes) AND
(@History = 'Y') THEN
    CALL AuditLog('PERSON',old_emp.personid,string(@CName,' History Details
Updated - '),old_emp.ExtendedNotes,new_emp.ExtendedNotes)
  END IF;
  -- Unsubscribe To Marketing
  IF locate(@AuditList,',Unsubscribe To Marketing,') > 0 AND
UPDATE(UnsubscribeToMarketing) THEN
    CALL AuditLog('PERSON',old_emp.personid,string('Unsubscribe To Marketing
- ',@ContactName,' -
',@CNAME),string(old_emp.UnsubscribeToMarketing),string(new_emp.UnsubscribeT
oMarketing))
  END IF;
END
}
GO


CREATE TRIGGER "EmploymentUnsubscribe" after UPDATE OF
"UnsubscribeToMarketing"
ORDER 2501 ON "pears"."Employment"
REFERENCING OLD AS "old_name" NEW AS "new_name"
FOR each ROW /* WHEN( search_condition ) */
BEGIN
  DECLARE "emailaddr" CHAR(100);
```

```
  DECLARE "existingemail" INTEGER;
  IF "new_name"."UnsubscribeToMarketing" = 1 THEN
    SELECT "isnull"("getphone"('E','E-mail',"old_name"."Employmentid"),'')
INTO "emailaddr";
    SELECT "count"("email") INTO "existingemail" FROM
"UnsubscribeToMarketing" WHERE "email" = "emailaddr";
    IF "emailaddr" <> '' AND "existingemail" = 0 THEN
      INSERT INTO "UnsubscribeToMarketing"( "email" ) VALUES( "emailaddr" )
;
      UPDATE "mailerselectionmember" SET "unsubscribe" = 1 WHERE
"datesubscribed" IS NOT NULL AND "record" = "old_name"."employmentid" AND
"mailerselectionid" = any(SELECT "mailerselectionid" FROM "mailerselection"
WHERE "type" = 'E');
      DELETE FROM "mailerselectionmember" WHERE "record" =
"old_name"."employmentid" AND "datesubscribed" IS NULL AND
"mailerselectionid" = any(SELECT "mailerselectionid" FROM "mailerselection"
WHERE "type" = 'E')
    END IF
  ELSE
    DELETE FROM "UnsubscribeToMarketing" WHERE "email" = any(SELECT
"getphone"('E','E-mail',"old_name"."Employmentid"))
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."employment"."EmploymentUnsubscribe" IS
{CREATE TRIGGER EmploymentUnsubscribe
 AFTER UPDATE OF UnsubscribeToMarketing
 ORDER 2501 ON Employment
  REFERENCING OLD AS old_name NEW AS new_name
 FOR EACH ROW /* WHEN( search_condition ) */
 BEGIN
   DECLARE emailaddr CHAR(100);
   DECLARE existingemail INTEGER;
   IF new_name.UnsubscribeToMarketing = 1 THEN
      SELECT isnull(getphone('E','E-mail', old_name.Employmentid),'') INTO
emailaddr  ;
     SELECT COUNT(email) INTO existingemail FROM UnsubscribeToMarketing
WHERE email=emailaddr;
      IF emailaddr<>'' AND  existingemail=0 THEN
        INSERT INTO UnsubscribeToMarketing (email) VALUES (emailaddr);
        UPDATE mailerselectionmember SET unsubscribe=1 WHERE datesubscribed
IS NOT NULL AND record=old_name.employmentid AND mailerselectionid IN
(SELECT mailerselectionid FROM mailerselection WHERE TYPE='E');
        DELETE FROM mailerselectionmember WHERE record=old_name.employmentid
AND datesubscribed IS NULL AND mailerselectionid IN (SELECT
```

```
mailerselectionid FROM mailerselection WHERE TYPE='E');
      END IF;
   ELSE
     DELETE FROM UnsubscribeToMarketing WHERE email IN ( SELECT
getphone('E','E-mail', old_name.Employmentid) )
   END IF;
 END
}
GO


CREATE TRIGGER "EmploymentAudit_insert" BEFORE INSERT ORDER 1 ON
"pears"."Employment"
REFERENCING NEW AS "new_emp"
FOR each ROW
BEGIN
  DECLARE @AuditList long VARCHAR;
  SELECT "string"(',',"list"("ItemName"),',') INTO @AuditList FROM
"AuditItems" WHERE("AreaName" = 'Company Contact' AND "AuditFlag" = 1);
  IF "locate"(@AuditList,',New Company Contact,') > 0 THEN
    CALL "AuditLog"('COMPANY',"new_emp"."companyid",'New Contact','',(SELECT
"name" FROM "person" WHERE "personid" = "new_emp"."personid"))
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."employment"."EmploymentAudit_insert" IS
{CREATE TRIGGER EmploymentAudit_insert
 BEFORE INSERT
 ORDER 1 ON pears.Employment
REFERENCING NEW AS new_emp
FOR each ROW
BEGIN
  DECLARE @AuditList long VARCHAR;
  SELECT string(',',list(ItemName),',') INTO @AuditList FROM AuditItems
WHERE (AreaName = 'Company Contact' AND AuditFlag = 1);
  IF locate(@AuditList,',New Company Contact,') > 0 THEN
    CALL AuditLog('COMPANY',new_emp.companyid,'New Contact','',(SELECT name
FROM person WHERE personid = new_emp.personid))
  END IF;
END
}
GO
```

From:
https://iqxusers.co.uk/iqxhelp/ - **iqx**

Permanent link:
**https://iqxusers.co.uk/iqxhelp/doku.php?id=database:tables:pears_employment**

Last update: **2026/08/07 19:24**