pears.AWRJobMaster

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

Description

Links Person records with Vacancies for AWR purposes.

Columns

Column	Type	Null	Default	Comment
AWRJobMasterID	char(20)	NOT NULL		
PlacementID	char(20)	NULL		
PersonID	char(20)	NULL		
Bonus	long varchar	NULL		
Notes	long varchar	NULL		
AWRCheckDate	date	NULL		
AWRRecvdDate	date	NULL		
Benefits	long varchar	NULL		
Holidays	long varchar	NULL		
Pay	long varchar	NULL		
AWRStatus	smallint	NOT NULL	1	1=Not known 2= Applies 3 = Not Applies
Grade	char(4)	NULL		
Speciality	char(100)	NULL		
TempShiftID	char(20)	NULL		
WhenEntered	date	NULL	current date	
VacancyID	char(20)	NULL		
ExtraHols	double	NULL		
AWRLinkCode	char(20)	NULL		
OverrideDate	date	NULL		
OverrideSystem	smallint	NULL	1	1=Use System 2= Qualified 3 = Not Qualified

Primary Key

- AWRJobMasterID

Foreign Keys

Constraint	Columns	References	Delete/update action
Person	PersonID	pears.Person (personid)	ON DELETE CASCADE
TempShift	TempShiftID	pears.TempShift (TempShiftID)	
Vacancy	VacancyID	pears.vacancy (vacancyid)	
Placement	PlacementID	pears.Placement (placementid)	ON DELETE CASCADE

Referenced By

Table	Constraint	Columns	Referenced columns
pears.AWRWeeklyDetail	AWRJobMaster	AWRJobMasterID	AWRJobMasterID
pears.AWRWeeklyQualified	AWRJobMaster	AWRJobMasterID	AWRJobMasterID
pears.TempTimeSheetLine	AWRJobMaster	AWRJobMasterID	AWRJobMasterID

Indexes

Name	Type	Columns	Detail
AWRJobMaster_AWRLinkCode	Index	AWRLinkCode	
AWRJobMasterVacIDPersID	Index	VacancyID, PersonID	

Triggers

Name	Timing	Event
awrjobmaster_update	after	update of “AWRStatus”, “AWRCheckDate”,“AWRLinkCode” order 1
awrjobmaster_updatetwo	after	update of “OverrideDate”, “OverrideSystem” order 2
awrjobmaster_updatethree	after	update of “ExtraHols” order 3
awrjobmaster_insert	after	insert order 1

Original SQL

```
-- IQX database structure split by table
-- Source: IQXDatabaseStructure - with comments.sql
-- Table: "pears"."AWRJobMaster"
-- Table comment: Links Person records with Vacancies for AWR purposes.
-- Statement count: 18

CREATE TABLE "pears"."AWRJobMaster" (
    "AWRJobMasterID"                CHAR(20) NOT NULL
   ,"PlacementID"                   CHAR(20) NULL
   ,"PersonID"                      CHAR(20) NULL
   ,"Bonus"                         long VARCHAR NULL
   ,"Notes"                         long VARCHAR NULL
```

```
    ,"AWRCheckDate"                    DATE NULL
    ,"AWRRecvdDate"                    DATE NULL
    ,"Benefits"                        long VARCHAR NULL
    ,"Holidays"                        long VARCHAR NULL
    ,"Pay"                             long VARCHAR NULL
    ,"AWRStatus"                       SMALLINT NOT NULL DEFAULT 1
    ,"Grade"                           CHAR(4) NULL
    ,"Speciality"                      CHAR(100) NULL
    ,"TempShiftID"                     CHAR(20) NULL
    ,"WhenEntered"                     DATE NULL DEFAULT CURRENT DATE
    ,"VacancyID"                       CHAR(20) NULL
    ,"ExtraHols"                       DOUBLE NULL
    ,"AWRLinkCode"                     CHAR(20) NULL
    ,"OverrideDate"                    DATE NULL
    ,"OverrideSystem"                  SMALLINT NULL DEFAULT 1
    ,PRIMARY KEY ("AWRJobMasterID" ASC)
)
GO


COMMENT ON COLUMN "pears"."AWRJobMaster"."AWRStatus" IS
    '1=Not known 2= Applies 3 = Not Applies'
GO


COMMENT ON COLUMN "pears"."AWRJobMaster"."OverrideSystem" IS
    '1=Use System 2= Qualified 3 = Not Qualified'
GO


COMMENT ON TABLE "pears"."AWRJobMaster" IS
    'Links Person records with Vacancies for AWR purposes.'
GO


ALTER TABLE "pears"."AWRJobMaster"
    ADD FOREIGN KEY "Person" ("PersonID" ASC)
    REFERENCES "pears"."Person" ("personid")
    ON DELETE CASCADE
GO


ALTER TABLE "pears"."AWRJobMaster"
    ADD FOREIGN KEY "TempShift" ("TempShiftID" ASC)
    REFERENCES "pears"."TempShift" ("TempShiftID")
GO
```

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


ALTER TABLE "pears"."AWRJobMaster"
    ADD FOREIGN KEY "Placement" ("PlacementID" ASC)
    REFERENCES "pears"."Placement" ("placementid")
    ON DELETE CASCADE
GO


CREATE INDEX "AWRJobMaster_AWRLinkCode" ON "pears"."AWRJobMaster"
    ( "AWRLinkCode" )
GO


CREATE INDEX "AWRJobMasterVacIDPersID" ON "pears"."AWRJobMaster"
    ( "VacancyID","PersonID" )
GO


CREATE TRIGGER "awrjobmaster_update" after UPDATE OF "AWRStatus",
"AWRCheckDate","AWRLinkCode" ORDER 1 ON "pears"."awrjobmaster"
REFERENCING OLD AS "old_awr" NEW AS "new_awr"
FOR each ROW
BEGIN
  DECLARE "ref" CHAR(20);
  -- audit
  SELECT "refcode" INTO "ref" FROM "placement" WHERE "placementid" =
"new_awr"."placementid";
  IF UPDATE("AWRStatus") THEN
    CALL "AuditLog"('AWRROLE',"old_awr"."AWRJobMasterID",
    "string"('AWR Status Placement OurRef - ',"ref",(IF "new_awr"."Grade" IS
NOT NULL THEN "string"(', Grade - ',"new_awr"."Grade")
    ELSE ''
    endif),IF "new_awr"."speciality" IS NOT NULL THEN "string"(', Speciality
- ',"new_awr"."speciality")
    ELSE ''
    endif),CASE "old_awr"."AWRStatus" WHEN 1 THEN 'Not Known' WHEN 2 THEN
'AWR Applies' WHEN 3 THEN 'AWR not Applicable' END,
    CASE "new_awr"."AWRStatus" WHEN 1 THEN 'Not Known' WHEN 2 THEN 'AWR
Applies' WHEN 3 THEN 'AWR not Applicable' END);
    UPDATE "AWRJobMaster" AS "a" SET "a"."AWRStatus" = "new_awr"."AWRStatus"
WHERE "AWRLinkCode"
       = "new_awr"."awrlinkcode" AND "a"."awrjobmasterid" <>
"new_awr"."awrjobmasterid"
```

```
  END IF;
  IF UPDATE("AWRCheckDate") THEN
    CALL "AuditLog"('AWRROLE',"old_awr"."AWRJobMasterID",
    "string"('AWR Checked Date Placement OurRef - ',"ref",(IF
"new_awr"."Grade" IS NOT NULL THEN "string"(', Grade - ',"new_awr"."Grade")
    ELSE ''
    endif),IF "new_awr"."speciality" IS NOT NULL THEN "string"(', Speciality
- ',"new_awr"."speciality")
    ELSE ''
endif),"string"("old_awr"."AWRCheckDate"),"string"("new_awr"."AWRCheckDate")
);
    UPDATE "AWRJobMaster" AS "a" SET "a"."AWRCheckDate" =
"new_awr"."AWRCheckDate" WHERE "AWRLinkCode"
       = "new_awr"."awrlinkcode" AND "a"."awrjobmasterid" <>
"new_awr"."awrjobmasterid"
  END IF;
  IF UPDATE("AWRLinkCode") AND "new_awr"."AWRLInkCode" <>
"new_awr"."AWRJobMasterID" AND "new_awr"."AWRLinkCode" IS NOT NULL THEN
    -- linking to other record copy
    UPDATE "AWRJobMaster" AS "newrec"
      SET "newrec"."AWRStatus" = "master"."AWRStatus","newrec"."benefits" =
"master"."benefits","newrec"."notes" = "master"."notes","newrec"."holidays"
= "master"."holidays",
      "newrec"."pay" = "master"."pay","newrec"."bonus" =
"master"."bonus","newrec"."extrahols" =
"master"."extrahols","newrec"."overridedate" = "master"."overridedate",
      "newrec"."overridesystem" =
"master"."overridesystem","newrec"."AWRcheckdate" = "master"."AWRcheckdate"
FROM
      "AWRJobMaster" AS "master"
      WHERE "newrec"."AWRJobMasterID" = "new_awr"."AWRJobMasterID"
      AND "master"."AWRJobMasterID" = "new_awr"."AWRLinkCode" AND
"master"."AWRLinkCode" = "new_awr"."AWRLinkCode"
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."AWRJobMaster"."awrjobmaster_update" IS
{CREATE TRIGGER awrjobmaster_update
 after UPDATE OF AWRStatus,
AWRCheckDate,AWRLinkCode ORDER 1 ON pears.awrjobmaster
REFERENCING OLD AS old_awr NEW AS new_awr
FOR each ROW
BEGIN
  DECLARE REF CHAR(20);
  -- audit
```

```
  SELECT refcode INTO REF FROM placement WHERE placementid =
new_awr.placementid;
  IF UPDATE(AWRStatus) THEN
    CALL AuditLog('AWRROLE',old_awr.AWRJobMasterID,
    string('AWR Status Placement OurRef - ',REF,(IF new_awr.Grade IS NOT
NULL THEN string(', Grade - ',new_awr.Grade)
    ELSE ''
    endif),IF new_awr.speciality IS NOT NULL THEN string(', Speciality -
',new_awr.speciality)
    ELSE ''
    endif),CASE old_awr.AWRStatus WHEN 1 THEN 'Not Known' WHEN 2 THEN 'AWR
Applies' WHEN 3 THEN 'AWR not Applicable'
    END,CASE new_awr.AWRStatus WHEN 1 THEN 'Not Known' WHEN 2 THEN 'AWR
Applies' WHEN 3 THEN 'AWR not Applicable'
    END);
    UPDATE AWRJobMaster AS a SET a.AWRStatus = new_awr.AWRStatus WHERE
AWRLinkCode
       = new_awr.awrlinkcode AND a.awrjobmasterid <> new_awr.awrjobmasterid
  END IF;
  IF UPDATE(AWRCheckDate) THEN
    CALL AuditLog('AWRROLE',old_awr.AWRJobMasterID,
    string('AWR Checked Date Placement OurRef - ',REF,(IF new_awr.Grade IS
NOT NULL THEN string(', Grade - ',new_awr.Grade)
    ELSE ''
    endif),IF new_awr.speciality IS NOT NULL THEN string(', Speciality -
',new_awr.speciality)
    ELSE ''
    endif),string(old_awr.AWRCheckDate),string(new_awr.AWRCheckDate));
    UPDATE AWRJobMaster AS a SET a.AWRCheckDate = new_awr.AWRCheckDate WHERE
AWRLinkCode
       = new_awr.awrlinkcode AND a.awrjobmasterid <> new_awr.awrjobmasterid
  END IF;
  IF UPDATE(AWRLinkCode) AND new_awr.AWRLInkCode <> new_awr.AWRJobMasterID
AND new_awr.AWRLinkCode IS NOT NULL THEN
    -- linking to other record copy
    UPDATE AWRJobMaster AS newrec SET
      newrec.AWRStatus = master.AWRStatus,newrec.benefits =
master.benefits,newrec.notes = master.notes,newrec.holidays =
master.holidays,
      newrec.pay = master.pay,newrec.bonus = master.bonus,newrec.extrahols =
master.extrahols,newrec.overridedate = master.overridedate,
      newrec.overridesystem = master.overridesystem,newrec.AWRcheckdate =
master.AWRcheckdate FROM
      AWRJobMaster AS master
      WHERE newrec.AWRJobMasterID = new_awr.AWRJobMasterID
      AND master.AWRJobMasterID = new_awr.AWRLinkCode AND master.AWRLinkCode
= new_awr.AWRLinkCode
  END IF
```

```
END
}
GO


CREATE TRIGGER "awrjobmaster_updatetwo" after UPDATE OF "OverrideDate",
"OverrideSystem" ORDER 2 ON "pears"."awrjobmaster"
REFERENCING OLD AS "old_awr" NEW AS "new_awr"
FOR each ROW
BEGIN
  DECLARE "ref" CHAR(20);
  -- audit
  SELECT "refcode" INTO "ref" FROM "placement" WHERE "placementid" =
"new_awr"."placementid";
  IF UPDATE("OverrideSystem") THEN
    CALL "AuditLog"('AWRROLE',"old_awr"."AWRJobMasterID",
    "string"('AWR Override Placement OurRef - ',"ref",(IF "new_awr"."Grade"
IS NOT NULL THEN "string"(', Grade - ',"new_awr"."Grade")
    ELSE ''
    endif),IF "new_awr"."speciality" IS NOT NULL THEN "string"(', Speciality
- ',"new_awr"."speciality")
    ELSE ''
    endif),CASE "old_awr"."OverrideSystem" WHEN 1 THEN 'No' WHEN 2 THEN
'Qualified' WHEN 3 THEN 'Not Qualified' END,
    CASE "new_awr"."OverrideSystem" WHEN 1 THEN 'No' WHEN 2 THEN 'Qualified'
WHEN 3 THEN 'Not Qualified' END)
  END IF;
  IF UPDATE("OverrideDate") THEN
    CALL "AuditLog"('AWRROLE',"old_awr"."AWRJobMasterID",
    "string"('AWR Override Date Placement OurRef - ',"ref",(IF
"new_awr"."Grade" IS NOT NULL THEN "string"(', Grade - ',"new_awr"."Grade")
    ELSE ''
    endif),IF "new_awr"."speciality" IS NOT NULL THEN "string"(', Speciality
- ',"new_awr"."speciality")
    ELSE ''
    endif),"old_awr"."OverrideDate",
    "new_awr"."OverrideDate")
  END IF;
  UPDATE "AWRJobMaster" AS "a" SET "a"."OverrideDate" =
"new_awr"."OverrideDate","a"."OverrideSystem" = "new_awr"."OverrideSystem"
WHERE "AWRLinkCode"
     = "new_awr"."awrlinkcode" AND "a"."awrjobmasterid" <>
"new_awr"."awrjobmasterid"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
```

```
"pears"."AWRJobMaster"."awrjobmaster_updatetwo" IS
{CREATE TRIGGER awrjobmaster_updatetwo
 after UPDATE OF OverrideDate,
OverrideSystem ORDER 2 ON pears.awrjobmaster
REFERENCING OLD AS old_awr NEW AS new_awr
FOR each ROW
BEGIN
  DECLARE REF CHAR(20);
  -- audit
  SELECT refcode INTO REF FROM placement WHERE placementid =
new_awr.placementid;
  IF UPDATE(OverrideSystem) THEN
    CALL AuditLog('AWRROLE',old_awr.AWRJobMasterID,
    string('AWR Override Placement OurRef - ',REF,(IF new_awr.Grade IS NOT
NULL THEN string(', Grade - ',new_awr.Grade)
    ELSE ''
    endif),IF new_awr.speciality IS NOT NULL THEN string(', Speciality -
',new_awr.speciality)
    ELSE ''
    endif),CASE old_awr.OverrideSystem WHEN 1 THEN 'No' WHEN 2 THEN
'Qualified' WHEN 3 THEN 'Not Qualified'
    END,CASE new_awr.OverrideSystem WHEN 1 THEN 'No' WHEN 2 THEN 'Qualified'
WHEN 3 THEN 'Not Qualified'
    END)
  END IF;
  IF UPDATE(OverrideDate) THEN
    CALL AuditLog('AWRROLE',old_awr.AWRJobMasterID,
    string('AWR Override Date Placement OurRef - ',REF,(IF new_awr.Grade IS
NOT NULL THEN string(', Grade - ',new_awr.Grade)
    ELSE ''
    endif),IF new_awr.speciality IS NOT NULL THEN string(', Speciality -
',new_awr.speciality)
    ELSE ''
    endif),old_awr.OverrideDate,
    new_awr.OverrideDate)
  END IF;
  UPDATE AWRJobMaster AS a SET a.OverrideDate =
new_awr.OverrideDate,a.OverrideSystem = new_awr.OverrideSystem WHERE
AWRLinkCode
     = new_awr.awrlinkcode AND a.awrjobmasterid <> new_awr.awrjobmasterid
END
}
GO


CREATE TRIGGER "awrjobmaster_updatethree" after UPDATE OF "ExtraHols"
ORDER 3 ON "pears"."awrjobmaster"
REFERENCING OLD AS "old_awr" NEW AS "new_awr"
```

```
FOR each ROW
BEGIN
  DECLARE "ref" CHAR(20);
  -- audit
  SELECT "refcode" INTO "ref" FROM "placement" WHERE "placementid" =
"new_awr"."placementid";
  CALL "AuditLog"('AWRROLE',"old_awr"."AWRJobMasterID",
  "string"('AWR Extra Holidays Placement OurRef - ',"ref",(IF
"new_awr"."Grade" IS NOT NULL THEN "string"(', Grade - ',"new_awr"."Grade")
  ELSE ''
  endif),IF "new_awr"."speciality" IS NOT NULL THEN "string"(', Speciality -
',"new_awr"."speciality")
  ELSE ''
  endif),"old_awr"."ExtraHols",
  "new_awr"."ExtraHols");
  UPDATE "AWRJobMaster" AS "a" SET "a"."ExtraHols" = "new_awr"."ExtraHols"
WHERE "AWRLinkCode"
     = "new_awr"."awrlinkcode" AND "a"."awrjobmasterid" <>
"new_awr"."awrjobmasterid"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."AWRJobMaster"."awrjobmaster_updatethree" IS
{CREATE TRIGGER awrjobmaster_updatethree
 after UPDATE OF ExtraHols
ORDER 3 ON pears.awrjobmaster
REFERENCING OLD AS old_awr NEW AS new_awr
FOR each ROW
BEGIN
  DECLARE REF CHAR(20);
  -- audit
  SELECT refcode INTO REF FROM placement WHERE placementid =
new_awr.placementid;
  CALL AuditLog('AWRROLE',old_awr.AWRJobMasterID,
  string('AWR Extra Holidays Placement OurRef - ',REF,(IF new_awr.Grade IS
NOT NULL THEN string(', Grade - ',new_awr.Grade)
  ELSE ''
  endif),IF new_awr.speciality IS NOT NULL THEN string(', Speciality -
',new_awr.speciality)
  ELSE ''
  endif),old_awr.ExtraHols,
  new_awr.ExtraHols);
  UPDATE AWRJobMaster AS a SET a.ExtraHols = new_awr.ExtraHols WHERE
AWRLinkCode
     = new_awr.awrlinkcode AND a.awrjobmasterid <> new_awr.awrjobmasterid
END
```

```
}
GO


CREATE TRIGGER "awrjobmaster_insert" after INSERT ORDER 1 ON
"pears"."awrjobmaster"
REFERENCING NEW AS "new_awr"
FOR each ROW
BEGIN
  IF "new_awr"."AWRLinkCode" IS NULL THEN
    UPDATE "AWRJobMaster" SET "AWRLinkCode" = "AWRJobMasterID" WHERE
"AWRJobMasterID" = "new_awr"."AWRJobMasterID"
  END IF;
  --not linking, so inherit values from AWRvacancy
  IF "new_awr"."AWRLinkCode" IS NULL OR "new_awr"."AWRLinkCode" =
"new_awr"."AWRJobMasterID" THEN
    IF("new_awr"."benefits" IS NULL) AND("new_awr"."notes" IS NULL)
AND("new_awr"."holidays" IS NULL) AND("new_awr"."pay" IS NULL)
AND("new_awr"."bonus" IS NULL) AND("new_awr"."extrahols" IS NULL) THEN
      UPDATE "AWRJobMaster" SET "AWRStatus" = "v"."AWRStatus","benefits" =
"v"."benefits","notes" = "v"."notes","holidays" = "v"."holidays",
        "pay" = "v"."pay","bonus" = "v"."bonus","extrahols" =
"v"."extrahols" FROM
        "AWRvacancy" AS "v" JOIN "AWRJobMaster" ON "v"."vacancyid" =
"AWRJobMaster"."vacancyid" WHERE "AWRJobMasterID" =
"new_awr"."AWRJobMasterID"
    END IF
  ELSE --linked, so inherit from the linked AWRJobMasterRecord
    UPDATE "AWRJobMaster" AS "newrec"
      SET "newrec"."AWRStatus" = "master"."AWRStatus","newrec"."benefits" =
"master"."benefits","newrec"."notes" = "master"."notes","newrec"."holidays"
= "master"."holidays",
      "newrec"."pay" = "master"."pay","newrec"."bonus" =
"master"."bonus","newrec"."extrahols" =
"master"."extrahols","newrec"."overridedate" = "master"."overridedate",
      "newrec"."overridesystem" =
"master"."overridesystem","newrec"."AWRcheckdate" = "master"."AWRcheckdate"
FROM
      "AWRJobMaster" AS "master"
      WHERE "newrec"."AWRJobMasterID" = "new_awr"."AWRJobMasterID"
      AND "master"."AWRJobMasterID" = "new_awr"."AWRLinkCode" AND
"master"."AWRLinkCode" = "new_awr"."AWRLinkCode"
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
```

```
"pears"."AWRJobMaster"."awrjobmaster_insert" IS
{CREATE TRIGGER awrjobmaster_insert
 after INSERT ORDER 1 ON
pears.awrjobmaster
REFERENCING NEW AS new_awr
FOR each ROW
BEGIN
  IF new_awr.AWRLinkCode IS NULL THEN
    UPDATE AWRJobMaster SET AWRLinkCode = AWRJobMasterID WHERE
AWRJobMasterID = new_awr.AWRJobMasterID
  END IF;
  --not linking, so inherit values from AWRvacancy
  IF new_awr.AWRLinkCode IS NULL OR new_awr.AWRLinkCode =
new_awr.AWRJobMasterID THEN
    IF(new_awr.benefits IS NULL) AND(new_awr.notes IS NULL)
AND(new_awr.holidays IS NULL) AND(new_awr.pay IS NULL) AND(new_awr.bonus IS
NULL) AND(new_awr.extrahols IS NULL) THEN
      UPDATE AWRJobMaster SET AWRStatus = v.AWRStatus,benefits =
v.benefits,notes = v.notes,holidays = v.holidays,
        pay = v.pay,bonus = v.bonus,extrahols = v.extrahols FROM
        AWRvacancy AS v JOIN AWRJobMaster ON v.vacancyid =
AWRJobMaster.vacancyid WHERE AWRJobMasterID = new_awr.AWRJobMasterID
    END IF
  ELSE
    --linked, so inherit from the linked AWRJobMasterRecord
    UPDATE AWRJobMaster AS newrec SET
      newrec.AWRStatus = master.AWRStatus,newrec.benefits =
master.benefits,newrec.notes = master.notes,newrec.holidays =
master.holidays,
      newrec.pay = master.pay,newrec.bonus = master.bonus,newrec.extrahols =
master.extrahols,newrec.overridedate = master.overridedate,
      newrec.overridesystem = master.overridesystem,newrec.AWRcheckdate =
master.AWRcheckdate FROM
      AWRJobMaster AS master
      WHERE newrec.AWRJobMasterID = new_awr.AWRJobMasterID
      AND master.AWRJobMasterID = new_awr.AWRLinkCode AND master.AWRLinkCode
= new_awr.AWRLinkCode
  END IF
END
}
GO
```