pears.Company

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

Description

Base Company / Client Record

Columns

Column	Type	Null	Default	Comment
companyid	char(20)	NOT NULL		
name	char(60)	NOT NULL		
keyname	char(60)	NOT NULL		
clientcode	char(12)	NULL		
fsflag	char(1)	NULL		
bms_id	char(12)	NULL		
addr1	char(40)	NULL		
addr2	char(40)	NULL		
addr3	char(40)	NULL		
town	char(30)	NULL		
county	char(30)	NULL		
country	char(30)	NULL		
postcode	char(20)	NULL		
alert	char(100)	NULL		
notes	long varchar	NULL		
invoiceaddress	smallint	NULL		
TempChargeCode	char(100)	NULL		Allows for the setting of of client or group based temp charge schemes
VatOnMargin	smallint	NULL		
TempShiftCode	char(12)	NULL		Allows for the setting of of client or group based standard shifts
status	char(1)	NOT NULL	'P'	
source	char(1)	NULL		
ParentCompanyID	char(20)	NULL		

Column	Type	Null	Default	Comment
currency	char(3)	NULL		Only used if params.homecurrency is set. Otherwise currency follows the default currency of the logged in user.
divisionid	char(20)	NULL		
staffid	char(20)	NULL		Made available in V2. Hidden by default
TransferNotes	long varchar	NULL		
RemoteSelfBill	tinyint	NULL		Identifies a company downloaded by a master agency which self-bills
TempHolidayCode	char(12)	NULL		Allows client specific holidays for rate script use
registrationdate	date	NULL	current date	Added V2.2.2.11 Will be null for existing companies
lastcontactevent	timestamp	NULL		Added V2.2.2.13 Will be null for existing companies
ExtraNotes	long varchar	NULL		
DefaultRateScheme	char(20)	NULL		
CompanyWarning	long varchar	NULL		
DefaultDocPack	char(20)	NULL		
SupplierCode	char(12)	NULL		Self bill account for secondary agency
OriginID	char(20)	NULL		This is the new Source
PrivateSector	smallint	NULL	0	Flag to indicate public/private
IsPortal	smallint	NULL	0	
AutoMatchUrgency	tinyint	NULL	0	Identifies automatch priority for a company
VATReverseCharge	tinyint	NULL		Construction rules on VAT charging
AllowPAYE	tinyint	NULL	1	
AllowLTD	tinyint	NULL	1	
AllowLTDF	tinyint	NULL	1	Limited + HMRCEngagement = F
AllowSelf	tinyint	NULL	1	
CompanySalesStatusID	char(20)	NULL		
TempComplianceCode	char(12)	NULL		Allows client specific compliance domains
DefaultVacancyRoleID	char(20)	NULL		
ETimesheetProv	tinyint	NOT NULL	0	
InvoiceWorkedShifts	smallint	NULL	0	Invoice Shifts off Company Accounts
QuestionDepartmentID	char(2)	NULL		Department used for question, defaults to user department if not set

Primary Key

- companyid

Foreign Keys

Constraint	Columns	References	Delete/update action
vacancyclass	source	pears.vacancyclass (classcode)	
staff	staffid	pears.staff (staffid)	
Company	ParentCompanyID	pears.Company (companyid)	ON DELETE SET NULL
division	divisionid	pears.Division (divisionid)	ON DELETE SET NULL
origin	OriginID	pears.Origin (OriginID)	
CompanySalesStatus	CompanySalesStatusID	pears.CompanySalesStatus (CompanySalesStatusID)	ON DELETE SET NULL
VacancyRole	DefaultVacancyRoleID	pears.VacancyRole (VacancyRoleID)	ON DELETE SET NULL
companystatus	status	pears.CompanyStatus (CompanyStatusID)	NOT NULL;
Department	QuestionDepartmentID	pears.Department (departmentid)	ON DELETE SET NULL

Referenced By

Table	Constraint	Columns	Referenced columns
pears.AccountOverrideInvoiceAddress	Company	InvoiceCompanyID	companyid
pears.AWRcompany	Company	companyid	companyid
pears.AWRWeeklyDetail	Company	CompanyID	companyid
pears.CardReaderOverrideSettings	company	Companyid	companyid
pears.CascadeDeliveryAddress	Company	CompanyID	companyid
pears.CascadedShift	Company	SecondaryAgencyID	companyid
pears.CascadedVacancy	Company	SecondaryAgencyID	companyid
pears.CascadeRule	Company	CompanyID	companyid
pears.Company	Company	ParentCompanyID	companyid
pears.CompanyAccount	Company	CompanyID	companyid
pears.CompanyDeptDocType	company	companyid	companyid
pears.CompanySDS	company	CompanyID	companyid
pears.EBTimeSheet	Company	SecondaryAgencyID	companyid
pears.employment	company	companyid	companyid
pears.IQXNetMessageCompanyRecipient	Company	CompanyID	companyid
pears.OffLimits	Company	CompanyID	companyid
pears.Pay_Employee	Company	SecondaryAgencyID	companyid
pears.PersonShiftPreference	Company	companyid	companyid
pears.StaffHistory	Company	CompanyID	companyid
pears.StatusHistory	Company	CompanyID	companyid
pears.TempDeskAgencyPoolMember	Company	CompanyID	companyid

Table	Constraint	Columns	Referenced columns
pears.TempJobType	Company	SecondaryAgencyID	companyid
pears.TempTimeSheet	Company	SecondaryAgencyID	companyid
pears.WithHolds	company	CompanyID	companyid

Indexes

Name	Type	Columns	Detail
company_keyname	Index	keyname	
company_postcode	Index	postcode	
cotown	Index	town	
company_clientcode	Index	clientcode	
person_lastcontactevent	Index	lastcontactevent	
company_notestext	Text index	notes	CONFIGURATION “SYS”.“default_char” IMMEDIATE REFRESH

Triggers

Name	Timing	Event
companyupdate	after	update of “name”, “clientcode”,“addr1”,“addr2”,“addr3”,“town”,“county”,“country”,“postcode”, “invoiceaddress”,“privatesector” order 3
updateinvaddress	before	update of “invoiceaddress” order 2
insertinvoiceaddress	before	insert order 1
WPK_company_SECAGENCY	after	insert,delete,update order 10
Company_InsertTrim	before	insert order 4
Company_UpdateTrim	before	update order 4
InsertStatusCompany	after	insert order 2
UpdateStatusCompany	after	update of “status”, “CompanySalesStatusID” order 2
CompanyAudit	before	update of “name”, “keyname”,“addr1”,“addr2”,“addr3”,“town”,“county”,“country”,“postcode”,“status”,“ParentCompanyID”,“divisionid”,“staffid”,“alert”, “originid”,“ClientCode”,“notes”,“PrivateSector”,“companywarning”,“CompanySalesStatusID” order 1
CompanyKeyWordsInsert	after	insert order 20
CompanyKeyWordsUpdate	after	update of “Name”, “Addr1”,“Addr2”,“Addr3”,“Town”,“County”,“PostCode” order 21

Original SQL

```
-- IQX database structure split by table
-- Source: IQXDatabaseStructure - with comments.sql
-- Table: "pears"."Company"
-- Table comment: Base Company / Client Record
-- Statement count: 56

CREATE TABLE "pears"."Company" (
    "companyid"                     CHAR(20) NOT NULL
   ,"name"                          CHAR(60) NOT NULL
   ,"keyname"                       CHAR(60) NOT NULL
CHECK("length"("trim"("keyname")) > 0)
   ,"clientcode"                    CHAR(12) NULL
   ,"fsflag"                        CHAR(1) NULL
   ,"bms_id"                        CHAR(12) NULL
   ,"addr1"                         CHAR(40) NULL
```

```
    ,"addr2"                        CHAR(40) NULL
    ,"addr3"                        CHAR(40) NULL
    ,"town"                         CHAR(30) NULL
    ,"county"                       CHAR(30) NULL
    ,"country"                      CHAR(30) NULL
    ,"postcode"                     CHAR(20) NULL
    ,"alert"                        CHAR(100) NULL
    ,"notes"                        long VARCHAR NULL
    ,"invoiceaddress"               SMALLINT NULL
    ,"TempChargeCode"               CHAR(100) NULL
    ,"VatOnMargin"                  SMALLINT NULL
    ,"TempShiftCode"                CHAR(12) NULL
    ,"status"                       CHAR(1) NOT NULL DEFAULT 'P'
    ,"source"                       CHAR(1) NULL
    ,"ParentCompanyID"              CHAR(20) NULL
    ,"currency"                     CHAR(3) NULL
    ,"divisionid"                   CHAR(20) NULL
    ,"staffid"                      CHAR(20) NULL
    ,"TransferNotes"                long VARCHAR NULL
    ,"RemoteSelfBill"               tinyint NULL
    ,"TempHolidayCode"              CHAR(12) NULL
    ,"registrationdate"             DATE NULL DEFAULT CURRENT DATE
    ,"lastcontactevent"             TIMESTAMP NULL
    ,"ExtraNotes"                   long VARCHAR NULL
    ,"DefaultRateScheme"            CHAR(20) NULL
    ,"CompanyWarning"               long VARCHAR NULL
    ,"DefaultDocPack"               CHAR(20) NULL
    ,"SupplierCode"                 CHAR(12) NULL
    ,"OriginID"                     CHAR(20) NULL
    ,"PrivateSector"                SMALLINT NULL DEFAULT 0
    ,"IsPortal"                     SMALLINT NULL DEFAULT 0
    ,"AutoMatchUrgency"             tinyint NULL DEFAULT 0
    ,"VATReverseCharge"             tinyint NULL
    ,"AllowPAYE"                    tinyint NULL DEFAULT 1
    ,"AllowLTD"                     tinyint NULL DEFAULT 1
    ,"AllowLTDF"                    tinyint NULL DEFAULT 1
    ,"AllowSelf"                    tinyint NULL DEFAULT 1
    ,"CompanySalesStatusID"         CHAR(20) NULL
    ,"TempComplianceCode"           CHAR(12) NULL
    ,"DefaultVacancyRoleID"         CHAR(20) NULL
    ,"ETimesheetProv"               tinyint NOT NULL DEFAULT 0
    ,"InvoiceWorkedShifts"          SMALLINT NULL DEFAULT 0
    ,"QuestionDepartmentID"         CHAR(2) NULL
    ,PRIMARY KEY ("companyid" ASC)
)
GO
```

```
COMMENT ON COLUMN "pears"."Company"."TempChargeCode" IS
    'Allows for the setting of of client or group based temp charge schemes'
GO


COMMENT ON COLUMN "pears"."Company"."TempShiftCode" IS
    'Allows for the setting of of client or group based standard shifts'
GO


COMMENT ON COLUMN "pears"."Company"."currency" IS
    'Only used if params.homecurrency is set. Otherwise currency follows the
default currency of the logged in user.'
GO


COMMENT ON COLUMN "pears"."Company"."staffid" IS
    'Made available in V2. Hidden by default'
GO


COMMENT ON COLUMN "pears"."Company"."RemoteSelfBill" IS
    'Identifies a company downloaded by a master agency which self-bills'
GO


COMMENT ON COLUMN "pears"."Company"."TempHolidayCode" IS
    'Allows client specific holidays for rate script use'
GO


COMMENT ON COLUMN "pears"."Company"."registrationdate" IS
    'Added V2.2.2.11 Will be null for existing companies'
GO


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


COMMENT ON COLUMN "pears"."Company"."SupplierCode" IS
    'Self bill account for secondary agency'
GO


COMMENT ON COLUMN "pears"."Company"."OriginID" IS
    'This is the new Source'
```

```
GO


COMMENT ON COLUMN "pears"."Company"."PrivateSector" IS
    'Flag to indicate public/private'
GO


COMMENT ON COLUMN "pears"."Company"."AutoMatchUrgency" IS
    'Identifies automatch priority for a company'
GO


COMMENT ON COLUMN "pears"."Company"."VATReverseCharge" IS
    'Construction rules on VAT charging'
GO


COMMENT ON COLUMN "pears"."Company"."AllowLTDF" IS
    'Limited + HMRCEngagement = F'
GO


COMMENT ON COLUMN "pears"."Company"."TempComplianceCode" IS
    'Allows client specific compliance domains'
GO


COMMENT ON COLUMN "pears"."Company"."InvoiceWorkedShifts" IS
    'Invoice Shifts off Company Accounts'
GO


COMMENT ON COLUMN "pears"."Company"."QuestionDepartmentID" IS
    'Department used for question, defaults to user department if not set'
GO


COMMENT ON TABLE "pears"."Company" IS
    'Base Company / Client Record'
GO


ALTER TABLE "pears"."Company"
    ADD FOREIGN KEY "vacancyclass" ("source" ASC)
    REFERENCES "pears"."vacancyclass" ("classcode")
GO
```

```
ALTER TABLE "pears"."Company"
    ADD FOREIGN KEY "staff" ("staffid" ASC)
    REFERENCES "pears"."staff" ("staffid")
GO


ALTER TABLE "pears"."Company"
    ADD FOREIGN KEY "Company" ("ParentCompanyID" ASC)
    REFERENCES "pears"."Company" ("companyid")
    ON DELETE SET NULL
GO


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


ALTER TABLE "pears"."Company"
    ADD FOREIGN KEY "origin" ("OriginID" ASC)
    REFERENCES "pears"."Origin" ("OriginID")
GO


ALTER TABLE "pears"."Company"
    ADD FOREIGN KEY "CompanySalesStatus" ("CompanySalesStatusID" ASC)
    REFERENCES "pears"."CompanySalesStatus" ("CompanySalesStatusID")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."Company"
    ADD FOREIGN KEY "VacancyRole" ("DefaultVacancyRoleID" ASC)
    REFERENCES "pears"."VacancyRole" ("VacancyRoleID")
    ON DELETE SET NULL
GO


ALTER TABLE "pears"."Company"
    ADD NOT NULL FOREIGN KEY "companystatus" ("status" ASC)
    REFERENCES "pears"."CompanyStatus" ("CompanyStatusID")
GO


ALTER TABLE "pears"."Company"
```

```
    ADD FOREIGN KEY "Department" ("QuestionDepartmentID" ASC)
    REFERENCES "pears"."Department" ("departmentid")
    ON DELETE SET NULL
GO


CREATE INDEX "company_keyname" ON "pears"."Company"
    ( "keyname" )
GO


CREATE INDEX "company_postcode" ON "pears"."Company"
    ( "postcode" )
GO


CREATE INDEX "cotown" ON "pears"."Company"
    ( "town" )
GO


CREATE INDEX "company_clientcode" ON "pears"."Company"
    ( "clientcode" )
GO


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


CREATE TEXT INDEX "company_notestext" ON "pears"."Company"
    ( "notes" ) CONFIGURATION "SYS"."default_char" IMMEDIATE REFRESH
GO


CREATE TRIGGER "companyupdate" after UPDATE OF "name",
"clientcode","addr1","addr2","addr3","town","county","country","postcode",
"invoiceaddress","privatesector" ORDER 3 ON "pears"."Company"
REFERENCING NEW AS "new_comp"
FOR each ROW
BEGIN
  UPDATE "companyaccount" SET "transferbatch" = 0 WHERE "companyid" =
"new_comp"."companyid" AND "transferbatch" > 0;
  IF "new_comp"."invoiceaddress" = 1 THEN
    UPDATE "companyaccount" KEY JOIN "company" SET
"companyaccount"."transferbatch" = 0 WHERE "company"."companyid" <>
"new_comp"."companyid"
```

```
      AND "company"."clientcode" = "new_comp"."clientcode" AND
"companyaccount"."transferbatch" > 0
  ELSE IF UPDATE("clientcode") THEN
      UPDATE "company" AS "c" SET "currency" = "b"."currency" FROM "company"
AS "c","company" AS "b"
        WHERE "c"."companyid" = "new_comp"."companyid" AND "b"."companyid" =
"getinvoicecompanyid"("new_comp"."clientcode");
      UPDATE "companyaccount" AS "c" SET "creditlimit" =
"b"."creditlimit","dayscredit" = "b"."dayscredit","groupinvoice" =
"b"."groupinvoice",
        "ernioninvoice" = "b"."ernioninvoice","invoiceemail" =
"b"."invoiceemail","onstop" = "b"."onstop","TheirRefRequiredInvoice" =
"b"."TheirRefRequiredInvoice",
        "Documenttemplateid" = "b"."Documenttemplateid","PDFInvCreate" =
"b"."PDFInvCreate","PDFInvIncTS" = "b"."PDFInvIncTS","FixedNI" =
"b"."FixedNI",
        "VATNumber" = "b"."VATNumber","VATExempt" =
"b"."VATExempt","PDFExtraDocs" = "b"."PDFExtraDocs","INVOICEMANAGEMENTGROUP"
= "b"."INVOICEMANAGEMENTGROUP","FixedWTR" = "b"."FixedWTR" FROM
        "companyaccount" AS "c","companyaccount" AS "b"
        WHERE "c"."companyid" = "new_comp"."companyid" AND "b"."companyid" =
"getinvoicecompanyid"("new_comp"."clientcode")
    END IF
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Company"."companyupdate" IS
{CREATE TRIGGER companyupdate
 after UPDATE OF "name",
"clientcode","addr1","addr2","addr3","town","county","country","postcode",
"invoiceaddress","privatesector" ORDER 3 ON "pears"."Company"
REFERENCING NEW AS "new_comp"
FOR each ROW
BEGIN
  UPDATE "companyaccount" SET "transferbatch" = 0 WHERE "companyid" =
"new_comp"."companyid" AND "transferbatch" > 0;
  IF "new_comp"."invoiceaddress" = 1 THEN
    UPDATE "companyaccount" KEY JOIN "company" SET
"companyaccount"."transferbatch" = 0 WHERE "company"."companyid" <>
"new_comp"."companyid"
      AND "company"."clientcode" = "new_comp"."clientcode" AND
"companyaccount"."transferbatch" > 0
  ELSE IF UPDATE("clientcode") THEN
      UPDATE "company" AS "c" SET "currency" = "b"."currency" FROM "company"
AS "c","company" AS "b"
        WHERE "c"."companyid" = "new_comp"."companyid" AND "b"."companyid" =
```

```
"getinvoicecompanyid"("new_comp"."clientcode");
      UPDATE "companyaccount" AS "c" SET "creditlimit" =
"b"."creditlimit","dayscredit" = "b"."dayscredit","groupinvoice" =
"b"."groupinvoice",
        "ernioninvoice" = "b"."ernioninvoice","invoiceemail" =
"b"."invoiceemail","onstop" = "b"."onstop","TheirRefRequiredInvoice" =
"b"."TheirRefRequiredInvoice",
        "Documenttemplateid" = "b"."Documenttemplateid","PDFInvCreate" =
"b"."PDFInvCreate","PDFInvIncTS" = "b"."PDFInvIncTS","FixedNI" =
"b"."FixedNI",
        "VATNumber" = "b"."VATNumber","VATExempt" =
"b"."VATExempt","PDFExtraDocs" = "b"."PDFExtraDocs","INVOICEMANAGEMENTGROUP"
= "b"."INVOICEMANAGEMENTGROUP","FixedWTR" = "b"."FixedWTR" FROM
        "companyaccount" AS "c","companyaccount" AS "b"
        WHERE "c"."companyid" = "new_comp"."companyid" AND "b"."companyid" =
"getinvoicecompanyid"("new_comp"."clientcode")
    END IF
  END IF
END
}
GO


CREATE TRIGGER "updateinvaddress" BEFORE UPDATE OF "invoiceaddress"
ORDER 2 ON "pears"."Company"
REFERENCING OLD AS "old_company" NEW AS "new_company"
FOR each ROW
WHEN("isnull"("new_company"."invoiceaddress",0) = 1
AND "isnull"("old_company"."invoiceaddress",0) = 0
AND "new_company"."clientcode" IS NOT NULL)
BEGIN
  UPDATE "company" SET "invoiceaddress" = NULL
    WHERE "clientcode" = "new_company"."clientcode"
    AND "companyid" <> "new_company"."companyid"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Company"."updateinvaddress"
IS
{CREATE TRIGGER updateinvaddress
 BEFORE UPDATE OF invoiceaddress
ORDER 2 ON pears.Company
REFERENCING OLD AS old_company NEW AS new_company
FOR each ROW
WHEN(isnull(new_company.invoiceaddress,0) = 1 AND
isnull(old_company.invoiceaddress,0) = 0 AND
new_company.clientcode IS NOT NULL)
```

```
BEGIN
  UPDATE company SET invoiceaddress = NULL WHERE
    clientcode = new_company.clientcode AND
    companyid <> new_company.companyid
END
}
GO


CREATE TRIGGER "insertinvoiceaddress" BEFORE INSERT ORDER 1 ON
"pears"."Company"
REFERENCING NEW AS "new_company"
FOR each ROW
WHEN("new_company"."invoiceaddress" = 1
AND "new_company"."clientcode" IS NOT NULL)
BEGIN
  UPDATE "company" SET "invoiceaddress" = NULL
    WHERE "clientcode" = "new_company"."clientcode"
    AND "companyid" <> "new_company"."companyid"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."Company"."insertinvoiceaddress" IS
{CREATE TRIGGER insertinvoiceaddress
 BEFORE INSERT ORDER 1 ON
pears.Company
REFERENCING NEW AS new_company
FOR each ROW
WHEN(new_company.invoiceaddress = 1 AND
new_company.clientcode IS NOT NULL)
BEGIN
  UPDATE company SET invoiceaddress = NULL WHERE
    clientcode = new_company.clientcode AND
    companyid <> new_company.companyid
END
}
GO


CREATE TRIGGER "WPK_company_SECAGENCY" after INSERT,DELETE,UPDATE ORDER 10
ON
"pears"."Company"
REFERENCING OLD AS "oldc" NEW AS "newc"
FOR each ROW
BEGIN
  DECLARE "doit" SMALLINT;
```

```
  IF inserting THEN IF "newc"."status" = 'A' THEN SET "doit" = 1
    END IF END IF;
  IF deleting THEN IF "oldc"."status" = 'A' THEN SET "doit" = 1
    END IF END IF;
  IF updating THEN IF(("newc"."status" = 'A') OR("oldc"."status" = 'A'))
THEN
      IF(("newc"."status" <> "oldc"."status") OR("oldc"."name" <>
"newc"."name")) THEN SET "doit" = 1
      END IF
    END IF END IF;
  IF "doit" = 1 THEN CALL "WPKTrackChange"('P','SECAGENCY')
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."Company"."WPK_company_SECAGENCY" IS
{CREATE TRIGGER WPK_company_SECAGENCY
 after INSERT,DELETE,UPDATE ORDER 10 ON
pears.Company
REFERENCING OLD AS oldc NEW AS newc
FOR each ROW
BEGIN
  DECLARE doit SMALLINT;
  IF inserting THEN IF newc.status = 'A' THEN SET doit=1
    END IF
  END IF;
  IF deleting THEN IF oldc.status = 'A' THEN SET doit=1
    END IF
  END IF;
  IF updating THEN IF((newc.status = 'A') OR(oldc.status = 'A')) THEN
      IF((newc.status <> oldc.status) OR(oldc.name <> newc.name)) THEN SET
doit=1
      END IF
    END IF
  END IF;
  IF doit = 1 THEN CALL WPKTrackChange('P','SECAGENCY')
  END IF END
}
GO


CREATE TRIGGER "Company_InsertTrim" BEFORE INSERT ORDER 4 ON
"pears"."company"
REFERENCING NEW AS "new_p"
FOR each ROW
BEGIN
```

```
  SET "new_p"."keyname" = "trim"("new_p"."keyname");
  SET "new_p"."clientcode" = "trim"("new_p"."clientcode")
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Company"."Company_InsertTrim"
IS
{CREATE TRIGGER Company_InsertTrim
 BEFORE INSERT ORDER 4 ON
pears.company
REFERENCING NEW AS new_p
FOR each ROW
BEGIN
  SET new_p.keyname = TRIM(new_p.keyname);
  SET new_p.clientcode = TRIM(new_p.clientcode)
END
}
GO


CREATE TRIGGER "Company_UpdateTrim" BEFORE UPDATE ORDER 4 ON
"pears"."company"
REFERENCING NEW AS "new_p"
FOR each ROW
BEGIN
  SET "new_p"."keyname" = "trim"("new_p"."keyname");
  SET "new_p"."clientcode" = "trim"("new_p"."clientcode")
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Company"."Company_UpdateTrim"
IS
{CREATE TRIGGER Company_UpdateTrim
 BEFORE UPDATE ORDER 4 ON
pears.company
REFERENCING NEW AS new_p
FOR each ROW
BEGIN
  SET new_p.keyname = TRIM(new_p.keyname);
  SET new_p.clientcode = TRIM(new_p.clientcode)
END
}
GO


CREATE TRIGGER "InsertStatusCompany" after INSERT ORDER 2 ON
```

```
"pears"."Company"
REFERENCING NEW AS "new_co"
FOR each ROW
BEGIN
  INSERT INTO "StatusHistory"(
"StatusHistoryID","companyid","staffid","newstatus" ) VALUES
    (
"uniquekey"("new_co"."companyid"),"new_co"."companyid","userstaffid","new_co
"."status" ) ;
  INSERT INTO "StatusHistory"(
"StatusHistoryID","companyid","staffid","newstatus","IsSales" ) VALUES
    (
"uniquekey"("new_co"."companyid"),"new_co"."companyid","userstaffid","new_co
"."CompanySalesStatusID",1 )
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."Company"."InsertStatusCompany" IS
{CREATE TRIGGER InsertStatusCompany
 after INSERT ORDER 2 ON
pears.Company
REFERENCING NEW AS new_co
FOR each ROW
BEGIN
  INSERT INTO StatusHistory (StatusHistoryID, companyid, staffid, newstatus)
    VALUES
    (uniquekey(new_co.companyid),new_co.companyid,
userstaffid,new_co.status);
  INSERT INTO StatusHistory (StatusHistoryID, companyid, staffid, newstatus,
IsSales)
    VALUES
    (uniquekey(new_co.companyid),new_co.companyid,
userstaffid,new_co.CompanySalesStatusID,1)
END
}
GO


CREATE TRIGGER "UpdateStatusCompany" after UPDATE OF "status",
"CompanySalesStatusID" ORDER 2 ON "pears"."Company"
REFERENCING OLD AS "old_co" NEW AS "new_co"
FOR each ROW
BEGIN
  DECLARE "ID" CHAR(20);
  IF UPDATE("status") THEN
    SELECT FIRST "StatusHistoryID" INTO "ID" FROM "StatusHistory" WHERE
```

```
"companyid" = "new_co"."companyid" AND "IsSales" = 0
      AND "datediff"("minute","whenentered",CURRENT TIMESTAMP) < 1 AND
"newstatus" = "old_co"."status" AND "oldstatus" = "new_co"."status" ORDER BY
"whenentered" DESC;
    IF "ID" IS NOT NULL THEN
      DELETE FROM "StatusHistory" WHERE "StatusHistoryID" = "ID"
    ELSE
      INSERT INTO "StatusHistory"(
"StatusHistoryID","companyid","staffid","oldstatus","newstatus" ) VALUES
        (
"uniquekey"("new_co"."companyid"),"new_co"."companyid","userstaffid","old_co
"."status","new_co"."status" )
    END IF
  END IF;
  IF UPDATE("CompanySalesStatusID") THEN
    SELECT FIRST "StatusHistoryID" INTO "ID" FROM "StatusHistory" WHERE
"companyid" = "new_co"."companyid" AND "IsSales" = 1
      AND "datediff"("minute","whenentered",CURRENT TIMESTAMP) < 1 AND
"newstatus" = "old_co"."CompanySalesStatusID" AND "oldstatus" =
"new_co"."CompanySalesStatusID" ORDER BY "whenentered" DESC;
    IF "ID" IS NOT NULL THEN
      DELETE FROM "StatusHistory" WHERE "StatusHistoryID" = "ID"
    ELSE
      INSERT INTO "StatusHistory"(
"StatusHistoryID","companyid","staffid","oldstatus","newstatus","IsSales" )
VALUES
        (
"uniquekey"("new_co"."companyid"),"new_co"."companyid","userstaffid","old_co
"."CompanySalesStatusID","new_co"."CompanySalesStatusID",1 )
    END IF
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."Company"."UpdateStatusCompany" IS
{CREATE TRIGGER UpdateStatusCompany
 after UPDATE OF STATUS, CompanySalesStatusID
ORDER 2 ON pears.Company
REFERENCING OLD AS old_co NEW AS new_co
FOR each ROW
BEGIN
  DECLARE ID CHAR(20);
 IF UPDATE(STATUS) THEN
   SELECT FIRST StatusHistoryID INTO ID FROM StatusHistory WHERE companyid =
new_co.companyid AND IsSales = 0
     AND datediff(MINUTE, whenentered, CURRENT TIMESTAMP) < 1 AND newstatus
```

```
= old_co.status AND oldstatus = new_co.status ORDER BY whenentered DESC;
   IF ID IS NOT NULL THEN
     DELETE FROM StatusHistory WHERE StatusHistoryID = ID
   ELSE
     INSERT INTO StatusHistory (StatusHistoryID, companyid, staffid,
oldstatus, newstatus)
      VALUES
      (uniquekey(new_co.companyid),new_co.companyid, userstaffid,
old_co.status, new_co.status)
   END IF;
 END IF;
 IF UPDATE(CompanySalesStatusID) THEN
   SELECT FIRST StatusHistoryID INTO ID FROM StatusHistory WHERE companyid =
new_co.companyid AND IsSales = 1
     AND datediff(MINUTE, whenentered, CURRENT TIMESTAMP) < 1 AND newstatus
= old_co.CompanySalesStatusID AND oldstatus = new_co.CompanySalesStatusID
ORDER BY whenentered DESC;
   IF ID IS NOT NULL THEN
     DELETE FROM StatusHistory WHERE StatusHistoryID = ID
   ELSE
     INSERT INTO StatusHistory (StatusHistoryID, companyid, staffid,
oldstatus, newstatus, IsSales)
      VALUES
      (uniquekey(new_co.companyid),new_co.companyid, userstaffid,
old_co.CompanySalesStatusID, new_co.CompanySalesStatusID, 1)
   END IF;
 END IF
END
}
GO


CREATE TRIGGER "CompanyAudit" BEFORE UPDATE OF "name",
"keyname","addr1","addr2","addr3","town","county","country","postcode","stat
us","ParentCompanyID","divisionid","staffid","alert",
"originid","ClientCode","notes","PrivateSector","companywarning","CompanySal
esStatusID" ORDER 1 ON "pears"."Company"
REFERENCING OLD AS "old_comp" NEW AS "new_comp"
FOR each ROW
WHEN(EXISTS(SELECT * FROM "AuditItems" WHERE "AreaName" = 'Company' AND
"AuditFlag" = 1))
BEGIN
  DECLARE @AuditList long VARCHAR;
  DECLARE @OldDescrip CHAR(250);
  DECLARE @NewDescrip CHAR(250);
  DECLARE @CompName CHAR(250);
  SELECT "string"(',',"list"("ItemName"),',') INTO @AuditList FROM
"AuditItems" WHERE "AreaName" = 'Company' AND "AuditFlag" = 1;
```

```
  SET @CompName = "old_comp"."Name";
  -- Status
  IF "locate"(@AuditList,',Status,') > 0 AND UPDATE("Status") THEN
    SELECT "name" INTO @OldDescrip FROM "status" WHERE "type" = 'C' AND
"status"."status" = "old_comp"."status";
    SELECT "name" INTO @NewDescrip FROM "status" WHERE "type" = 'C' AND
"status"."status" = "new_comp"."status";
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Status
Updated - ',@CompName),"string"("old_comp"."Status",' -
',@OldDescrip),"string"("new_comp"."Status",' - ',@NewDescrip))
  END IF;
  -- SalesStatus
  IF "locate"(@AuditList,',SalesStatus,') > 0 AND
UPDATE("companysalesstatusid") THEN
    SELECT "name" INTO @OldDescrip FROM "companysalesstatus" WHERE
"companysalesstatusid" = "old_comp"."companysalesstatusid";
    SELECT "name" INTO @NewDescrip FROM "companysalesstatus" WHERE
"companysalesstatusid" = "new_comp"."companysalesstatusid";
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Sales Status
Updated - ',@CompName),"string"("old_comp"."companysalesstatusid",' -
',@OldDescrip),"string"("new_comp"."companysalesstatusid",' -
',@NewDescrip))
  END IF;
  -- Division
  IF "locate"(@AuditList,',Division,') > 0 AND UPDATE("DivisionID") THEN
    SELECT "name" INTO @OldDescrip FROM "Division" WHERE "divisionid" =
"old_comp"."DivisionID";
    SELECT "name" INTO @NewDescrip FROM "Division" WHERE "divisionid" =
"new_comp"."DivisionID";
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Division
Updated - ',@CompName),@OldDescrip,@NewDescrip)
  END IF;
  -- Name
  IF "locate"(@AuditList,',Name,') > 0 AND UPDATE("Name") THEN
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Name Updated
- ',@CompName),"old_comp"."name","new_comp"."name")
  END IF;
  -- Notes
  IF "locate"(@AuditList,',Notes,') > 0 AND UPDATE("Notes") THEN
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Notes Updated
- ',@CompName),"old_comp"."notes","new_comp"."notes")
  END IF;
  IF "locate"(@AuditList,',Warning,') > 0 AND UPDATE("companywarning") THEN
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Warning
Updated -
',@CompName),"old_comp"."companywarning","new_comp"."companywarning")
  END IF;
  IF "locate"(@AuditList,',KeyName,') > 0 AND UPDATE("KeyName") THEN
```

```
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('KeyName
Updated - ',@CompName),"old_comp"."keyname","new_comp"."keyname")
  END IF;
  -- Parent
  IF "locate"(@AuditList,',Parent,') > 0 AND UPDATE("ParentCompanyID") THEN
    SELECT "name" INTO @OldDescrip FROM "Company" WHERE "companyid" =
"old_comp"."ParentCompanyID";
    SELECT "name" INTO @NewDescrip FROM "Company" WHERE "companyid" =
"new_comp"."ParentCompanyID";
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Parent
Company Updated - ',@CompName),@OldDescrip,@NewDescrip)
  END IF;
  -- Address
  IF "locate"(@AuditList,',Address,') > 0 AND(UPDATE("addr1") OR
UPDATE("addr2") OR UPDATE("addr3") OR UPDATE("town") OR UPDATE("county") OR
UPDATE("country") OR UPDATE("postcode")) THEN
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Address
Updated - ',@CompName),"string"("old_comp"."addr1",',
',"old_comp"."addr2",', ',"old_comp"."addr3",', ',"old_comp"."town",',
',"old_comp"."county",', ',"old_comp"."postcode",',
',"old_comp"."country"),"string"("new_comp"."addr1",',
',"new_comp"."addr2",', ',"new_comp"."addr3",', ',"new_comp"."town",',
',"new_comp"."county",', ',"new_comp"."postcode",', ',"new_comp"."country"))
  END IF;
  -- Consultant
  IF "locate"(@AuditList,',Consultant,') > 0 AND UPDATE("StaffID") THEN
    SELECT "userid" INTO @OldDescrip FROM "staff" WHERE "staffid" =
"old_comp"."StaffID";
    SELECT "userid" INTO @NewDescrip FROM "staff" WHERE "staffid" =
"new_comp"."StaffID";
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Consultant
Updated - ',@CompName),@OldDescrip,@NewDescrip);
    INSERT INTO "staffhistory"(
"staffhistoryid","companyid","oldstaff","newstaff","whoentered","whenentered
" ) VALUES(
"uniquekey"("new_comp"."StaffID"),"new_comp"."companyid","old_comp"."StaffID
","new_comp"."StaffID","userstaffid",CURRENT TIMESTAMP )
  END IF;
  -- Alert
  IF "locate"(@AuditList,',Alert,') > 0 AND UPDATE("Alert") THEN
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Alert Updated
- ',@CompName),"old_comp"."Alert","new_comp"."Alert")
  END IF;
  -- Source
  IF "locate"(@AuditList,',Source,') > 0 AND UPDATE("Originid") THEN
    SELECT "descrip" INTO @OldDescrip FROM "origin" WHERE "originid" =
"old_comp"."originid";
    SELECT "descrip" INTO @NewDescrip FROM "origin" WHERE "originid" =
```

```
"new_comp"."originid";
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Source
Updated - ',@CompName),@OldDescrip,@NewDescrip)
  END IF;
  -- Account Code
  IF "locate"(@AuditList,',Account Code,') > 0 AND UPDATE("ClientCode") THEN
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Account Code
- ',@CompName),"old_comp"."ClientCode","new_comp"."ClientCode")
  END IF;
  IF "locate"(@AuditList,',Private Sector Outside IR35,') > 0 AND
UPDATE("PrivateSector") THEN
    CALL "AuditLog"('COMPANY',"old_comp"."companyid","string"('Private
Sector Outside IR35 -
',@CompName),"old_comp"."PrivateSector","new_comp"."PrivateSector")
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Company"."CompanyAudit" IS
{CREATE TRIGGER CompanyAudit
 BEFORE UPDATE OF
name,keyname,addr1,addr2,addr3,town,county,country,postcode,STATUS,ParentCom
panyID,divisionid,staffid,alert,
originid,ClientCode,notes,PrivateSector, companywarning,
CompanySalesStatusID ORDER 1 ON pears.Company
REFERENCING OLD AS old_comp NEW AS new_comp
FOR each ROW
WHEN(EXISTS(SELECT* FROM AuditItems WHERE AreaName = 'Company' AND AuditFlag
= 1))
BEGIN
  DECLARE @AuditList long VARCHAR;
  DECLARE @OldDescrip CHAR(250);
  DECLARE @NewDescrip CHAR(250);
  DECLARE @CompName CHAR(250);
  SELECT string(',',list(ItemName),',') INTO @AuditList FROM AuditItems
WHERE AreaName = 'Company' AND AuditFlag = 1;
  SET @CompName=old_comp.Name;
  -- Status
  IF locate(@AuditList,',Status,') > 0 AND UPDATE(STATUS) THEN
    SELECT name INTO @OldDescrip FROM STATUS WHERE TYPE = 'C' AND
STATUS.status = old_comp.status;
    SELECT name INTO @NewDescrip FROM STATUS WHERE TYPE = 'C' AND
STATUS.status = new_comp.status;
    CALL AuditLog('COMPANY',old_comp.companyid,string('Status Updated -
',@CompName),string(old_comp.Status,' -
',@OldDescrip),string(new_comp.Status,' - ',@NewDescrip))
  END IF;
```

```
  -- SalesStatus
  IF locate(@AuditList,',SalesStatus,') > 0 AND UPDATE(companysalesstatusid)
THEN
    SELECT name INTO @OldDescrip FROM companysalesstatus WHERE
companysalesstatusid = old_comp.companysalesstatusid;
    SELECT name INTO @NewDescrip FROM companysalesstatus WHERE
companysalesstatusid = new_comp.companysalesstatusid;
    CALL AuditLog('COMPANY',old_comp.companyid,string('Sales Status Updated
- ',@CompName),string(old_comp.companysalesstatusid,' -
',@OldDescrip),string(new_comp.companysalesstatusid,' - ',@NewDescrip))
  END IF;
  -- Division
  IF locate(@AuditList,',Division,') > 0 AND UPDATE(DivisionID) THEN
    SELECT name INTO @OldDescrip FROM Division WHERE divisionid =
old_comp.DivisionID;
    SELECT name INTO @NewDescrip FROM Division WHERE divisionid =
new_comp.DivisionID;
    CALL AuditLog('COMPANY',old_comp.companyid,string('Division Updated -
',@CompName),@OldDescrip,@NewDescrip)
  END IF;
  -- Name
  IF locate(@AuditList,',Name,') > 0 AND UPDATE(Name) THEN
    CALL AuditLog('COMPANY',old_comp.companyid,string('Name Updated -
',@CompName),old_comp.name,new_comp.name)
  END IF;
  -- Notes
  IF locate(@AuditList,',Notes,') > 0 AND UPDATE(Notes) THEN
    CALL AuditLog('COMPANY',old_comp.companyid,string('Notes Updated -
',@CompName),old_comp.notes,new_comp.notes)
  END IF;
  IF locate(@AuditList,',Warning,') > 0 AND UPDATE(companywarning) THEN
    CALL AuditLog('COMPANY',old_comp.companyid,string('Warning Updated -
',@CompName),old_comp.companywarning,new_comp.companywarning)
  END IF;
  IF locate(@AuditList,',KeyName,') > 0 AND UPDATE(KeyName) THEN
    CALL AuditLog('COMPANY',old_comp.companyid,string('KeyName Updated -
',@CompName),old_comp.keyname,new_comp.keyname)
  END IF;
  -- Parent
  IF locate(@AuditList,',Parent,') > 0 AND UPDATE(ParentCompanyID) THEN
    SELECT name INTO @OldDescrip FROM Company WHERE companyid =
old_comp.ParentCompanyID;
    SELECT name INTO @NewDescrip FROM Company WHERE companyid =
new_comp.ParentCompanyID;
    CALL AuditLog('COMPANY',old_comp.companyid,string('Parent Company
Updated - ',@CompName),@OldDescrip,@NewDescrip)
  END IF;
  -- Address
```

```
  IF locate(@AuditList,',Address,') > 0 AND(UPDATE(addr1) OR UPDATE(addr2)
OR UPDATE(addr3) OR UPDATE(town) OR UPDATE(county) OR UPDATE(country) OR
UPDATE(postcode)) THEN
    CALL AuditLog('COMPANY',old_comp.companyid,string('Address Updated -
',@CompName),string(old_comp.addr1,', ',old_comp.addr2,',
',old_comp.addr3,', ',old_comp.town,', ',old_comp.county,',
',old_comp.postcode,', ',old_comp.country),string(new_comp.addr1,',
',new_comp.addr2,', ',new_comp.addr3,', ',new_comp.town,',
',new_comp.county,', ',new_comp.postcode,', ',new_comp.country))
  END IF;
  -- Consultant
  IF locate(@AuditList,',Consultant,') > 0 AND UPDATE(StaffID) THEN
    SELECT userid INTO @OldDescrip FROM staff WHERE staffid =
old_comp.StaffID;
    SELECT userid INTO @NewDescrip FROM staff WHERE staffid =
new_comp.StaffID;
    CALL AuditLog('COMPANY',old_comp.companyid,string('Consultant Updated -
',@CompName),@OldDescrip,@NewDescrip);
    INSERT INTO staffhistory (staffhistoryid, companyid, oldstaff, newstaff,
whoentered, whenentered) VALUES
(uniquekey(new_comp.StaffID),new_comp.companyid, old_comp.StaffID,
new_comp.StaffID, userstaffid, CURRENT TIMESTAMP)
  END IF;
  -- Alert
  IF locate(@AuditList,',Alert,') > 0 AND UPDATE(Alert) THEN
    CALL AuditLog('COMPANY',old_comp.companyid,string('Alert Updated -
',@CompName),old_comp.Alert,new_comp.Alert)
  END IF;
  -- Source
  IF locate(@AuditList,',Source,') > 0 AND UPDATE(Originid) THEN
    SELECT descrip INTO @OldDescrip FROM origin WHERE originid =
old_comp.originid;
    SELECT descrip INTO @NewDescrip FROM origin WHERE originid =
new_comp.originid;
    CALL AuditLog('COMPANY',old_comp.companyid,string('Source Updated -
',@CompName),@OldDescrip,@NewDescrip)
  END IF;
  -- Account Code
  IF locate(@AuditList,',Account Code,') > 0 AND UPDATE(ClientCode) THEN
    CALL AuditLog('COMPANY',old_comp.companyid,string('Account Code -
',@CompName),old_comp.ClientCode,new_comp.ClientCode)
  END IF;
  IF locate(@AuditList,',Private Sector Outside IR35,') > 0 AND
UPDATE(PrivateSector) THEN
    CALL AuditLog('COMPANY',old_comp.companyid,string('Private Sector
Outside IR35 - ',@CompName),old_comp.PrivateSector,new_comp.PrivateSector)
  END IF
END
```

```
}
GO


CREATE TRIGGER "CompanyKeyWordsInsert" after INSERT ORDER 20 ON
"pears"."Company"
REFERENCING NEW AS "NewRow"
FOR each ROW
BEGIN
  INSERT INTO "CompanyKeyWords"( "CompanyID","RefreshRequired" ) ON existing
UPDATE defaults off VALUES( "NewRow"."CompanyID",1 )
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."Company"."CompanyKeyWordsInsert" IS
{CREATE TRIGGER CompanyKeyWordsInsert
after INSERT  ORDER 20 ON
 pears.Company
 REFERENCING NEW AS NewRow
FOR each ROW
BEGIN
   INSERT INTO CompanyKeyWords(CompanyID,RefreshRequired) ON existing UPDATE
VALUES(NewRow.CompanyID,1);
END
}
GO


CREATE TRIGGER "CompanyKeyWordsUpdate" after UPDATE OF "Name",
"Addr1","Addr2","Addr3","Town","County","PostCode" ORDER 21 ON
"pears"."Company"
REFERENCING NEW AS "NewRow"
FOR each ROW
BEGIN
  UPDATE "CompanyKeyWords" SET "RefreshRequired" = 1 WHERE "CompanyID" =
"NewRow"."CompanyID"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."Company"."CompanyKeyWordsUpdate" IS
{CREATE TRIGGER CompanyKeyWordsUpdate
after  UPDATE OF Name,Addr1,Addr2,Addr3,Town,County,PostCode ORDER 21 ON
   pears.Company
  REFERENCING NEW AS NewRow
```

```
 FOR each ROW
 BEGIN
 UPDATE CompanyKeyWords SET RefreshRequired = 1 WHERE CompanyID =
NewRow.CompanyID;
 END
}
GO
```