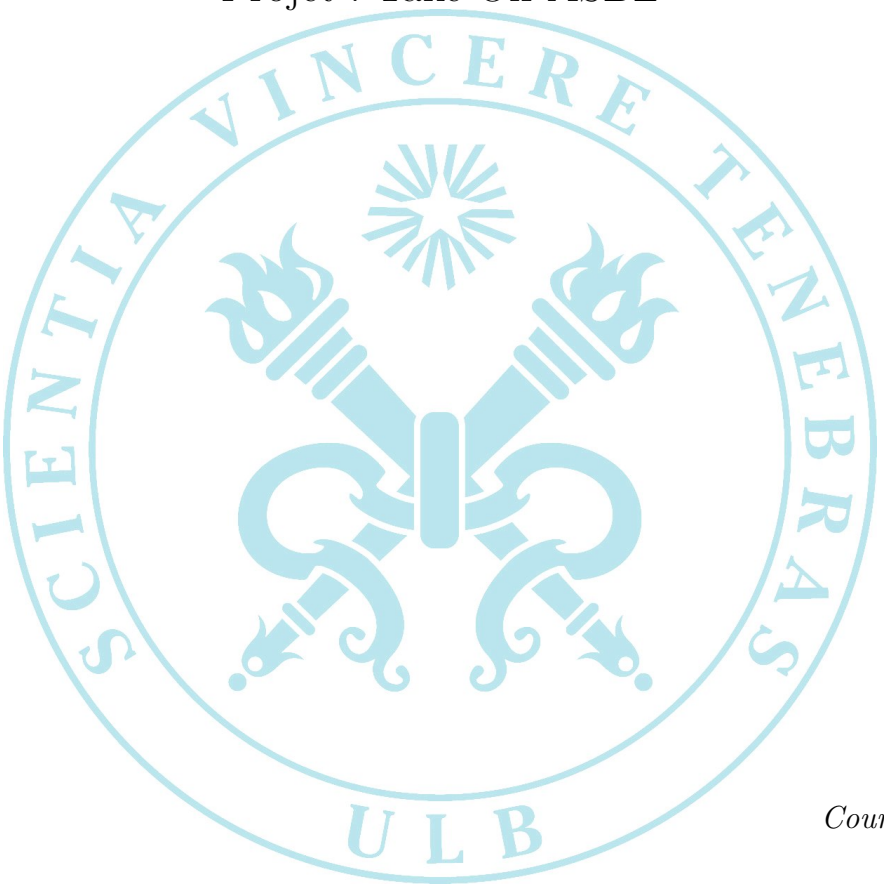
Université Libre de Bruxelles
Faculté de Philosophie et Lettres
Département des Sciences de
l'Information et de la Communication

ULB

Gestion de base de données

Projet : Take Off ASBL



Cours de M. A. Clève
STIC-B-505

John Divoy

STIC-5I
074728-58
jdivoy@ulb.ac.be

Année Académique 2012-2013

Table des matières

1 Introduction

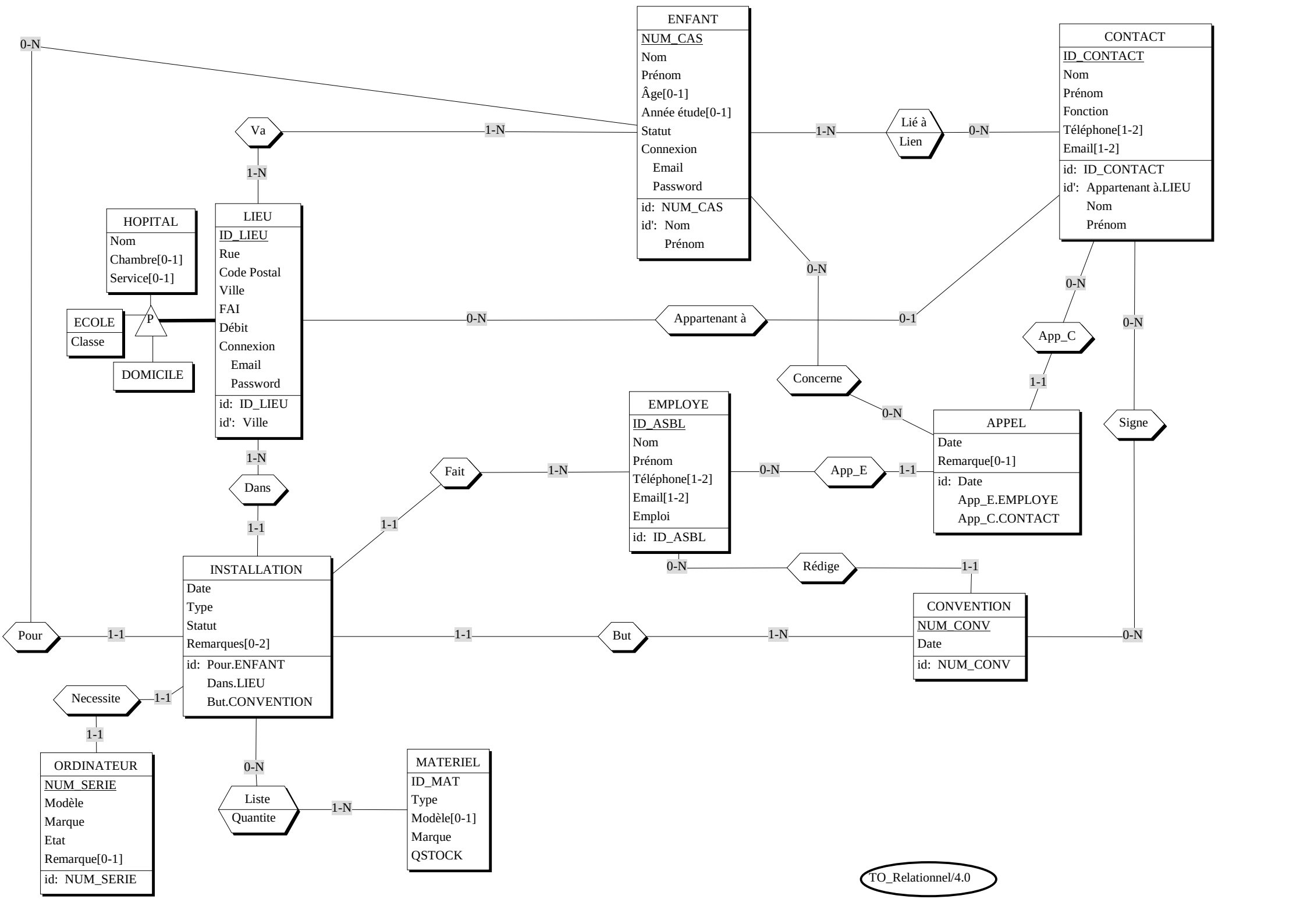
L'objectif du présent document est de présenter une structure de base de donnée potentielle dans le cadre des activités de l'A.S.B.L. Take Off dont l'objectif est, si tant est qu'il faille le rappeler, de « de mettre, gratuitement, à la disposition de l'enfant et de son école de base, les moyens informatiques qui vont lui permettre de garder le contact depuis l'hôpital ou son domicile » [1].

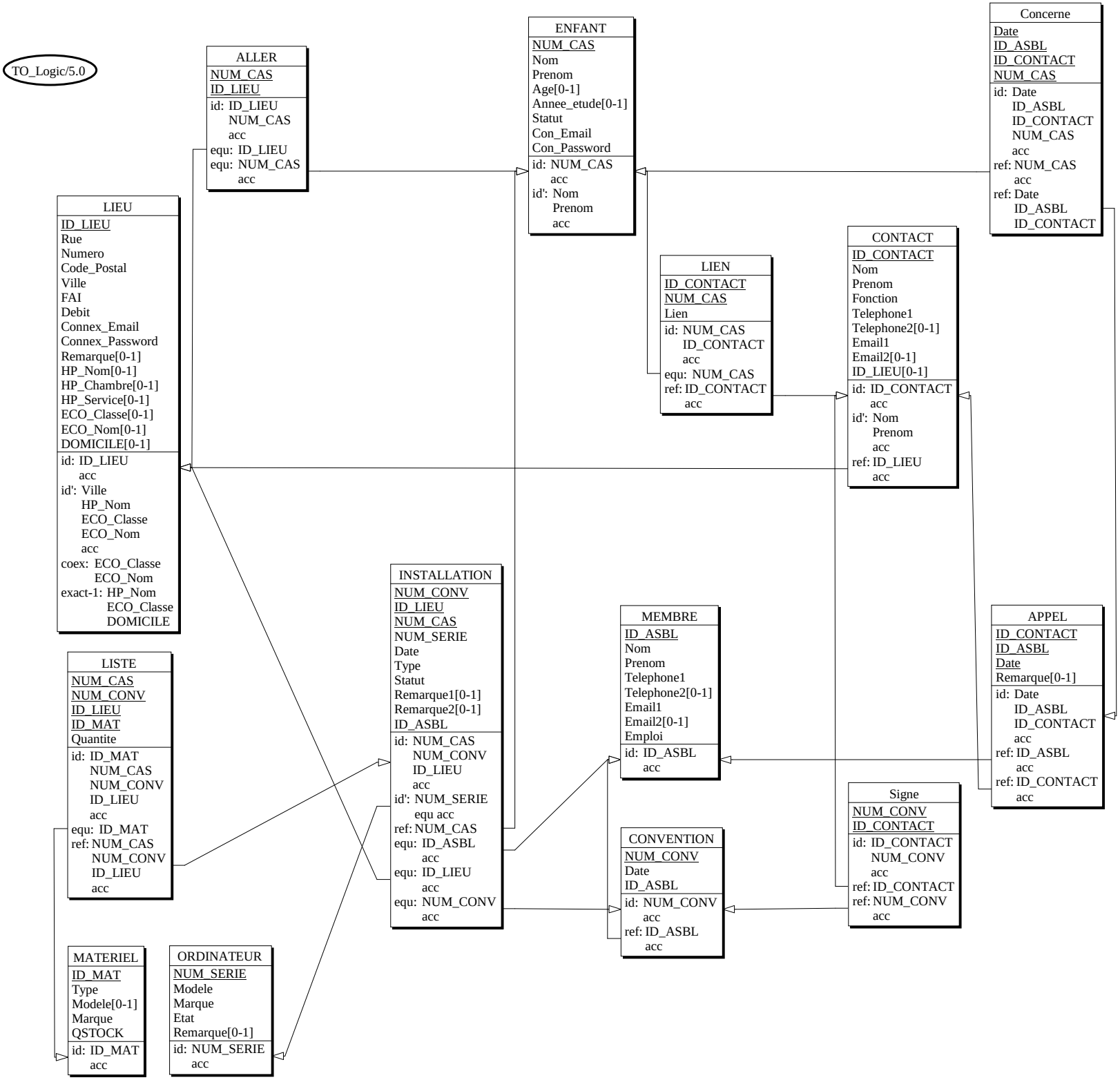
Nous présenterons donc dans un premier temps le schéma entité-relation de notre base de donnée et son équivalent logique. Nous ne reviendrons pas sur les différents arbitrages ni les différentes conventions graphiques usitées. Le Code SQL est également repris ci-dessous.

Nous avons également rédigé trois contraintes additionnelles sous forme de `trigger` ainsi que deux vues pouvant présenter un avantage pour les membres de l'A.S.B.L. Enfin, avant de tester notre code sur un système MySQL, nous avons rédigé quelques requêtes type afin d'exemplifier les capacités du système.

Nous avons également commencé à rédiger un rapport pour l'A.S.B.L. afin de leur transmettre les résultats de notre travail mais nous souhaiterions pouvoir proposer un interface à l'aide du module DAEZY, ce qui n'est pas encore le cas. Ce rapport étant inachevé et explicitant principalement les schémas et leurs conventions, nous n'avons pas jugé utile de le reprendre ici.

1. Take Off ASBL, *Garder une vie sociale et scolaire, quand on se retrouve isolé par la maladie ou un accident, c'est possible!* [En ligne]. < `http://www.asbl-takeoff-vzw.be/`>. (Consulté le 13/01/2014).


ENFANT
NUM_CAS
Nom
Prénom
Âge[0-1]
Année étude[0-1]
Statut
Connexion
Email
Password
id: NUM_CAS
id': Nom
Prénom
CONTACT
ID_CONTACT
Nom
Prénom
Fonction
Téléphone[1-2]
Email[1-2]
id: ID_CONTACT
id': Appartenant à.LIEU
Nom
Prénom
Va
Lié à
Lien
HOPITAL
Nom
Chambre[0-1]
Service[0-1]
LIEU
ID_LIEU
Rue
Code Postal
Ville
FAI
Débit
Connexion
Email
Password
id: ID_LIEU
id': Ville
ECOLE
Classe
P
DOMICILE
Appartenant à
Concerne
App_C
Signe
Dans
Fait
EMPLOYE
ID_ASBL
Nom
Prénom
Téléphone[1-2]
Email[1-2]
Emploi
id: ID_ASBL
App_E
APPEL
Date
Remarque[0-1]
id: Date
App_E.EMPLOYE
App_C.CONTACT
Rédige
INSTALLATION
Date
Type
Statut
Remarques[0-2]
id: Pour.ENFANT
Dans.LIEU
But.CONVENTION
Pour
But
CONVENTION
NUM_CONV
Date
id: NUM_CONV
Necessite
ORDINATEUR
NUM_SERIE
Modèle
Marque
Etat
Remarque[0-1]
id: NUM_SERIE
Liste
Quantite
MATERIEL
ID_MAT
Type
Modèle[0-1]
Marque
QSTOCK
0-N
1-N
1-1
TO_Relationnel/4.0


TO_Logic/5.0
ALLER
NUM_CAS
ID_LIEU
id: ID_LIEU
NUM_CAS
acc
equ: ID_LIEU
equ: NUM_CAS
acc
ENFANT
NUM_CAS
Nom
Prenom
Age[0-1]
Annee_etude[0-1]
Statut
Con_Email
Con_Password
id: NUM_CAS
acc
id': Nom
Prenom
acc
Concerne
Date
ID_ASBL
ID_CONTACT
NUM_CAS
id: Date
ID_ASBL
ID_CONTACT
NUM_CAS
acc
ref: NUM_CAS
acc
ref: Date
ID_ASBL
ID_CONTACT
LIEU
ID_LIEU
Rue
Numero
Code_Postal
Ville
FAI
Debit
Connex_Email
Connex_Password
Remarque[0-1]
HP_Nom[0-1]
HP_Chambre[0-1]
HP_Service[0-1]
ECO_Classe[0-1]
ECO_Nom[0-1]
DOMICILE[0-1]
id: ID_LIEU
acc
id': Ville
HP_Nom
ECO_Classe
ECO_Nom
acc
coex: ECO_Classe
ECO_Nom
exact-1: HP_Nom
ECO_Classe
DOMICILE
LIEN
ID_CONTACT
NUM_CAS
Lien
id: NUM_CAS
ID_CONTACT
acc
equ: NUM_CAS
ref: ID_CONTACT
acc
CONTACT
ID_CONTACT
Nom
Prenom
Fonction
Telephone1
Telephone2[0-1]
Email1
Email2[0-1]
ID_LIEU[0-1]
id: ID_CONTACT
acc
id': Nom
Prenom
acc
ref: ID_LIEU
acc
INSTALLATION
NUM_CONV
ID_LIEU
NUM_CAS
NUM_SERIE
Date
Type
Statut
Remarque1[0-1]
Remarque2[0-1]
ID_ASBL
id: NUM_CAS
NUM_CONV
ID_LIEU
acc
id': NUM_SERIE
equ acc
ref: NUM_CAS
equ: ID_ASBL
acc
equ: ID_LIEU
acc
equ: NUM_CONV
acc
MEMBRE
ID_ASBL
Nom
Prenom
Telephone1
Telephone2[0-1]
Email1
Email2[0-1]
Emploi
id: ID_ASBL
acc
APPEL
ID_CONTACT
ID_ASBL
Date
Remarque[0-1]
id: Date
ID_ASBL
ID_CONTACT
acc
ref: ID_ASBL
acc
ref: ID_CONTACT
acc
LISTE
NUM_CAS
NUM_CONV
ID_LIEU
ID_MAT
Quantite
id: ID_MAT
NUM_CAS
NUM_CONV
ID_LIEU
acc
equ: ID_MAT
ref: NUM_CAS
NUM_CONV
ID_LIEU
acc
CONVENTION
NUM_CONV
Date
ID_ASBL
id: NUM_CONV
acc
ref: ID_ASBL
acc
Signe
NUM_CONV
ID_CONTACT
id: ID_CONTACT
NUM_CONV
acc
ref: ID_CONTACT
ref: NUM_CONV
acc
MATERIEL
ID_MAT
Type
Modele[0-1]
Marque
QSTOCK
id: ID_MAT
acc
ORDINATEUR
NUM_SERIE
Modele
Marque
Etat
Remarque[0-1]
id: NUM_SERIE
acc

```
-- ********************************************
-- * SQL MySQL generation
-- *------------------------------------------
-- * DB-MAIN version: 9.1.5
-- * Generator date: Feb 14 2012
-- * Generation date: Wed Jan 13 18:29:55 2013
-- * LUN file: D:\Dropbox\MA2\Base de données\TAKE OFF.lun
-- * Schema: TO_Logic/5.0
-- ********************************************


-- Database Section
-- ________________

create database TO;
use TO;


-- Tables Section
-- ______________

create table ALLER (
     NUM_CAS char(10) not null,
     ID_LIEU char(10) not null,
     constraint ID_ALLER_ID primary key (ID_LIEU, NUM_CAS));

create table APPEL (
     ID_CONTACT char(10) not null,
     ID_ASBL char(10) not null,
     Date date not null,
     Remarque char(200),
     constraint ID_APPEL_ID primary key (Date, ID_ASBL, ID_CONTACT));

create table Concerne (
     Date date not null,
     ID_ASBL char(10) not null,
     ID_CONTACT char(10) not null,
     NUM_CAS char(10) not null,
     constraint ID_Concerne_ID primary key (Date, ID_ASBL, ID_CONTACT, NUM_CAS));

create table CONTACT (
     ID_CONTACT char(10) not null,
     Nom char(60) not null,
     Prenom char(60) not null,
     Fonction char(60) not null,
     Telephone1 char(60) not null,
     Telephone2 char(60),
     Email1 char(60) not null,
     Email2 char(60),
     ID_LIEU char(10),
     constraint ID_CONTACT_ID primary key (ID_CONTACT),
     constraint SID_CONTACT_ID unique (Nom, Prenom));

create table CONVENTION (
     NUM_CONV char(10) not null,
     Date date not null,
     ID_ASBL char(10) not null,
```

```
     constraint ID_CONVENTION_ID primary key (NUM_CONV));

create table ENFANT (
     NUM_CAS char(10) not null,
     Nom char(60) not null,
     Prenom char(60) not null,
     Age int,
     Annee_etude char(60),
     Statut char(60) not null,
     Con_Email char(60) not null,
     Con_Password char(60) not null,
     constraint ID_ENFANT_ID primary key (NUM_CAS),
     constraint SID_ENFANT_ID unique (Nom, Prenom));

create table INSTALLATION (
     NUM_CONV char(10) not null,
     ID_LIEU char(10) not null,
     NUM_CAS char(10) not null,
     NUM_SERIE char(1) not null,
     Date date not null,
     Type char(60) not null,
     Statut char(60) not null,
     Remarque1 char(60),
     Remarque2 char(60),
     ID_ASBL char(10) not null,
     constraint ID_INSTALLATION_ID primary key (NUM_CAS, NUM_CONV, ID_LIEU),
     constraint FKNecessite_ID unique (NUM_SERIE));

create table LIEN (
     ID_CONTACT char(10) not null,
     NUM_CAS char(10) not null,
     Lien char(60) not null,
     constraint ID_LIEN_ID primary key (NUM_CAS, ID_CONTACT));

create table LIEU (
     ID_LIEU char(10) not null,
     Rue char(60) not null,
     Numero char(10) not null,
     Code_Postal char(10) not null,
     Ville char(60) not null,
     FAI char(60) not null,
     Debit char(10) not null,
     Connex_Email char(60) not null,
     Connex_Password char(60) not null,
     Remarque char(200),
     HP_Nom char(60),
     HP_Chambre char(10),
     HP_Service char(60),
     ECO_Classe char(60),
     ECO_Nom char(60),
     DOMICILE char,
     constraint ID_LIEU_ID primary key (ID_LIEU),
     constraint SID_LIEU_ID unique (Ville, HP_Nom, ECO_Classe, ECO_Nom));

create table LISTE (
     NUM_CAS char(10) not null,
     NUM_CONV char(10) not null,
```

```
     ID_LIEU char(10) not null,
     ID_MAT char(10) not null,
     Quantite char(1) not null,
     constraint ID_LISTE_ID primary key (ID_MAT, NUM_CAS, NUM_CONV, ID_LIEU));

create table MATERIEL (
     ID_MAT char(10) not null,
     Type char(60) not null,
     Modele char(60),
     Marque char(60) not null,
     QSTOCK int not null,
     constraint ID_MATERIEL_ID primary key (ID_MAT));

create table MEMBRE (
     ID_ASBL char(10) not null,
     Nom char(60) not null,
     Prenom char(60) not null,
     Telephone1 varchar(60) not null,
     Telephone2 varchar(60),
     Email1 char(60) not null,
     Email2 char(60),
     Emploi char(60) not null,
     constraint ID_MEMBRE_ID primary key (ID_ASBL));

create table ORDINATEUR (
     NUM_SERIE char(1) not null,
     Modele char(1) not null,
     Marque char(1) not null,
     Etat char(1) not null,
     Remarque char(1),
     constraint ID_ORDINATEUR_ID primary key (NUM_SERIE));

create table Signe (
     NUM_CONV char(10) not null,
     ID_CONTACT char(10) not null,
     constraint ID_Signe_ID primary key (ID_CONTACT, NUM_CONV));


-- Constraints Section
-- ____________________

alter table ALLER add constraint FKALL_LIE
     foreign key (ID_LIEU)
     references LIEU (ID_LIEU);

alter table ALLER add constraint FKALL_ENF_FK
     foreign key (NUM_CAS)
     references ENFANT (NUM_CAS);

alter table APPEL add constraint FKApp_E_FK
     foreign key (ID_ASBL)
     references MEMBRE (ID_ASBL);

alter table APPEL add constraint FKApp_C_FK
     foreign key (ID_CONTACT)
     references CONTACT (ID_CONTACT);
```

```
alter table Concerne add constraint FKCon_ENF_FK
     foreign key (NUM_CAS)
     references ENFANT (NUM_CAS);

alter table Concerne add constraint FKCon_APP
     foreign key (Date, ID_ASBL, ID_CONTACT)
     references APPEL (Date, ID_ASBL, ID_CONTACT);

alter table CONTACT add constraint FKAppartenant_a_FK
     foreign key (ID_LIEU)
     references LIEU (ID_LIEU);

-- Not implemented
-- alter table CONVENTION add constraint ID_CONVENTION_CHK
--     check(exists(select * from INSTALLATION
--                  where INSTALLATION.NUM_CONV = NUM_CONV));

alter table CONVENTION add constraint FKRedige_FK
     foreign key (ID_ASBL)
     references MEMBRE (ID_ASBL);

-- Not implemented
-- alter table ENFANT add constraint ID_ENFANT_CHK
--     check(exists(select * from ALLER
--                  where ALLER.NUM_CAS = NUM_CAS));

-- Not implemented
-- alter table ENFANT add constraint ID_ENFANT_CHK
--     check(exists(select * from LIEN
--                  where LIEN.NUM_CAS = NUM_CAS));

alter table INSTALLATION add constraint FKPour
     foreign key (NUM_CAS)
     references ENFANT (NUM_CAS);

alter table INSTALLATION add constraint FKNecessite_FK
     foreign key (NUM_SERIE)
     references ORDINATEUR (NUM_SERIE);

alter table INSTALLATION add constraint FKFait_FK
     foreign key (ID_ASBL)
     references MEMBRE (ID_ASBL);

alter table INSTALLATION add constraint FKDans_FK
     foreign key (ID_LIEU)
     references LIEU (ID_LIEU);

alter table INSTALLATION add constraint FKBut_FK
     foreign key (NUM_CONV)
     references CONVENTION (NUM_CONV);

alter table LIEN add constraint FKLie_ENF
     foreign key (NUM_CAS)
     references ENFANT (NUM_CAS);

alter table LIEN add constraint FKLie_CON_FK
     foreign key (ID_CONTACT)
```

```
     references CONTACT (ID_CONTACT);

-- Not implemented
-- alter table LIEU add constraint ID_LIEU_CHK
--     check(exists(select * from ALLER
--                  where ALLER.ID_LIEU = ID_LIEU));

-- Not implemented
-- alter table LIEU add constraint ID_LIEU_CHK
--     check(exists(select * from INSTALLATION
--                  where INSTALLATION.ID_LIEU = ID_LIEU));

alter table LIEU add constraint COEX_LIEU
     check((ECO_Classe is not null and ECO_Nom is not null)
           or (ECO_Classe is null and ECO_Nom is null));

alter table LIEU add constraint EXTONE_LIEU
     check((HP_Nom is not null and ECO_Classe is null and DOMICILE is null)
           or (HP_Nom is null and ECO_Classe is not null and DOMICILE is null)
           or (HP_Nom is null and ECO_Classe is null and DOMICILE is not null));

alter table LISTE add constraint FKLIS_MAT
     foreign key (ID_MAT)
     references MATERIEL (ID_MAT);

alter table LISTE add constraint FKLIS_INS_FK
     foreign key (NUM_CAS, NUM_CONV, ID_LIEU)
     references INSTALLATION (NUM_CAS, NUM_CONV, ID_LIEU);

-- Not implemented
-- alter table MATERIEL add constraint ID_MATERIEL_CHK
--     check(exists(select * from LISTE
--                  where LISTE.ID_MAT = ID_MAT));

-- Not implemented
-- alter table MEMBRE add constraint ID_MEMBRE_CHK
--     check(exists(select * from INSTALLATION
--                  where INSTALLATION.ID_ASBL = ID_ASBL));

-- Not implemented
-- alter table ORDINATEUR add constraint ID_ORDINATEUR_CHK
--     check(exists(select * from INSTALLATION
--                  where INSTALLATION.NUM_SERIE = NUM_SERIE));

alter table Signe add constraint FKSig_CON_1
     foreign key (ID_CONTACT)
     references CONTACT (ID_CONTACT);

alter table Signe add constraint FKSig_CON_FK
     foreign key (NUM_CONV)
     references CONVENTION (NUM_CONV);


-- Index Section
-- _____________

create unique index ID_ALLER_IND
```

```
     on ALLER (ID_LIEU, NUM_CAS);

create index FKALL_ENF_IND
     on ALLER (NUM_CAS);

create unique index ID_APPEL_IND
     on APPEL (Date, ID_ASBL, ID_CONTACT);

create index FKApp_E_IND
     on APPEL (ID_ASBL);

create index FKApp_C_IND
     on APPEL (ID_CONTACT);

create unique index ID_Concerne_IND
     on Concerne (Date, ID_ASBL, ID_CONTACT, NUM_CAS);

create index FKCon_ENF_IND
     on Concerne (NUM_CAS);

create unique index ID_CONTACT_IND
     on CONTACT (ID_CONTACT);

create unique index SID_CONTACT_IND
     on CONTACT (Nom, Prenom);

create index FKAppartenant_a_IND
     on CONTACT (ID_LIEU);

create unique index ID_CONVENTION_IND
     on CONVENTION (NUM_CONV);

create index FKRedige_IND
     on CONVENTION (ID_ASBL);

create unique index ID_ENFANT_IND
     on ENFANT (NUM_CAS);

create unique index SID_ENFANT_IND
     on ENFANT (Nom, Prenom);

create unique index ID_INSTALLATION_IND
     on INSTALLATION (NUM_CAS, NUM_CONV, ID_LIEU);

create unique index FKNecessite_IND
     on INSTALLATION (NUM_SERIE);

create index FKFait_IND
     on INSTALLATION (ID_ASBL);

create index FKDans_IND
     on INSTALLATION (ID_LIEU);

create index FKBut_IND
     on INSTALLATION (NUM_CONV);

create unique index ID_LIEN_IND
```

```
     on LIEN (NUM_CAS, ID_CONTACT);

create index FKLie_CON_IND
     on LIEN (ID_CONTACT);

create unique index ID_LIEU_IND
     on LIEU (ID_LIEU);

create unique index SID_LIEU_IND
     on LIEU (Ville, HP_Nom, ECO_Classe, ECO_Nom);

create unique index ID_LISTE_IND
     on LISTE (ID_MAT, NUM_CAS, NUM_CONV, ID_LIEU);

create index FKLIS_INS_IND
     on LISTE (NUM_CAS, NUM_CONV, ID_LIEU);

create unique index ID_MATERIEL_IND
     on MATERIEL (ID_MAT);

create unique index ID_MEMBRE_IND
     on MEMBRE (ID_ASBL);

create unique index ID_ORDINATEUR_IND
     on ORDINATEUR (NUM_SERIE);

create unique index ID_Signe_IND
     on Signe (ID_CONTACT, NUM_CONV);

create index FKSig_CON_IND
     on Signe (NUM_CONV);
```

2 Contraintes additionnelles : Triggers

2.1 Existence d'au moins un contact par lieu où va l'enfant

De par la flexibilité et le caractère généraliste de notre structure, il nous paraît nécessaire d'imposer que à tout lieu où se rend un enfant doit appartenir au moins un contact. Pour ce faire, il faut créer le `trigger` suivant :

```
CREATE TRIGGER TRIG_CONT_LIEU_ENF
BEFORE INSERT ON ALLER
For each statement
BEGIN
  SELECT ID_CONTACT, ID_LIEU FROM CONTACT;
  IF CONTACT.ID_LIEU = ALLER.ID_LIEU
    INSERT INTO ALLER (NUM_CAS, ID_LIEU)
    VALUE=(new.NUM_CAS, new.ID_LIEU)
  END IF
END
```

2.2 Vérification de la disponibilité du matériel à l'encodage d'une installation

```
CREATE TRIGGER TRIG_MATERIEL
BEFORE INSERT ON LISTE
For each row
BEGIN
  SELECT ID_MAT, QSTOCK FROM MATERIEL;
  WHERE ID_MAT=new.ID_MAT
  IF new.Quantite > QSTOCK
     Raise exception 1035 'Matériel non disponible';
  END IF
END;
```

2.3 Adéquation du type de lieu avec le type d'installation

Puisque l'A.S.B.L. opère deux types d'installation, à savoir les PC_Classe et les PC_Enfants, nous voudrions spécifié que si le lieu d'installation est un hôpital ou un domicile, le type d'installation doit être un PC_Enfants alors que si celle-ci a lieu dans une école, le type d'installation doit être un PC_Classe. En cas de non respect de cette

condition, la saisie sera annulée. Cette contrainte particulière peut être spécifiée grâce au `trigger` suivant :

```
CREATE TRIGGER TRIG_INSTA.TP_LIEU.TP
BEFORE INSERT ON INSTALLATION
For each statement
BEGIN
  SELECT HP_Nom, ECO_Classe FROM LIEU;
  WHERE INSTALLATION.ID_LIEU = LIEU.ID_LIEU
  IF HP.Nom is not null OR DOMICILE is not null
    SELECT Type FROM INSTALLATION
    IF Type LIKE %Enfant%
       ABORT()
    END IF
  ELSE IF ECO_Classe is not null
    SELECT Type FROM INSTALLATION
    IF Type LIKE %Classe%
       ABORT
    END IF
  END IF
END
```

3 Vues

Nous proposons aussi de créer des vues, à savoir des tables virtuelles reprenant diverses informations issues de plusieurs tables « réelles » afin de faciliter la consultation et l'encodage de données.

3.1 Vue Installation - Lieu - Enfant

Il s'agit ici, de permettre l'affichage en un seul point toutes les informations relatives à une installation, son lieu et le cas concerné. Nous pensons qu'il s'agit d'une vue reprenant toutes les informations nécessaires aux techniciens lors d'une installation. Si ce n'est que le matériel à installer n'y est pas repris.

INSTALLATION				LIEU					ENFANT		
Date	Type	Statut	Rmq1	Id_Lieu	Rue	N°	CP	Ville	N°_cas	Nom	Prénom
			Rmq2	HP_Nom	Chambre		Service				
				Ecole_Nom	Classe						

Cette vue devra être créée à l'aide du code SQL suivant :

```
CREATE VIEW INSTA_LIEU_ENFANT
AS SELECT INSTALLATION.Date, INSTALLATION.Type, INSTALLATION.Statut,
INSTALLATION.Remarque1, INSTALLATION.Remarque2, LIEU.ID_LIEU, HP_Nom, HP_Chambre,
HP_Service, ECO_Nom, ECO_Classe, Rue, Numero, Code_Postal, Ville,
ENFANT.NUM_CAS, ENFANT.Nom, ENFANT.Prenom
FROM LIEU, INSTALLATION, ENFANT
WHERE LIEU.ID_LIEU=INSTALLATION.ID_LIEU
AND ENFANT.NUM_CAS=INSTALLATION.NUM_CAS
```

3.2 Vue Contact - Lieu installé - Enfant - Lien

CONTACT						INSTA_LIEU_ENFANT				LIEN
ID_Contact	Nom	Prenom	Fct	Tel1	Email1	HP_Nom	Service	Nom	Prenom	Lien
				Tel2	Email2	Ecole_Nom				

Cette vue, utilisant la vue précédente, devra être créée à l'aide du code SQL suivant :

```
CREATE VIEW CONTACT_LIEU_INSTA
AS SELECT CONTACT.Nom, CONTACT.Prenom, Fonction, Telephone1, Telephone2, Email1,
Email2, INSTA_LIEU_ENFANT.HP_Nom, INSTA_LIEU_ENFANT.HP_Service,
INSTA_LIEU_ENFANT.ECO_Nom, INSTA_LIEU_ENFANT.Nom as Nom_Enfant,
INSTA_LIEU_ENFANT.Prenom as Prenom_Enfant, LIEN.Lien
FROM CONTACT, INSTA_LIEU_ENFANT, LIEN
WHERE INSTA_LIEU_ENFANT.ID_LIEU = CONTACT.ID_LIEU
AND CONTACT.ID_CONTACT=LIEN.ID_CONTACT
```

4 Requêtes type

Nous proposons ici quelques requêtes SQL afin de démontrer les capacités potentielles du système. Il sera normalement possible, dans le cas d'une implémentation réelle, de récupérer toute données incluse selon des critères choisis et ce grâce aux fonctionnalités du langage SQL.

4.1 Liste du matériel pour un enfant, sur base du numéro de cas

```
SELECT distinct ID_LIEU, ID_MAT, Quantite;
FROM LISTE;
```

```
WHERE NUM_CAS="E_001";
GROUP BY ID_LIEU;
```

4.2 Liste de tous les cas non clôturés

```
SELECT NUM_CAS, Nom, Prenom;
FROM ENFANT;
WHERE Statut LIKE "%En cours%";
```

4.3 Création d'un mailing

4.3.1 Postal, relatif aux contacts d'un hôpital

Sur base de l'identifiant de l'hôpital :

```
SELECT HP_Nom, HP_Service, Nom, Prenom, Rue, Numero, Code_Postal, Ville;
FROM CONTACT, LIEU;
WHERE CONTACT.ID_LIEU = LIEU.ID_LIEU;
AND LIEU.ID_LIEU="L001";
```

Sur base du nom de l'hôpital :

```
SELECT HP_Nom, HP_Service, Nom, Prenom, Rue, Numero, Code_Postal, Ville;
FROM CONTACT, LIEU;
WHERE CONTACT.ID_LIEU = LIEU.ID_LIEU;
AND LIEU.ID_LIEU LIKE "%Saint-Luc%";
```

4.3.2 Électronique, relatif aux contacts d'une école

Sur base de l'identifiant de l'école :

```
SELECT ECO_Nom, Nom, Prenom, Email1, Email2;
FROM CONTACT, LIEU;
WHERE CONTACT.ID_LIEU = LIEU.ID_LIEU;
AND LIEU.ID_LIEU="L002";
```

Sur base du nom de l'école :

```
SELECT ECO_Nom, Nom, Prenom, Email1, Email2;
FROM CONTACT, LIEU;
WHERE CONTACT.ID_LIEU = LIEU.ID_LIEU;
AND LIEU.ID_LIEU LIKE "%Eglantiers%";
```

5 Interface et version test

Nous avons testé notre code de création de table ainsi que celui de création de vues sur une instance locale de `PHPMyAdmin`, ce qui fonctionne très bien. Cependant, nos codes de `triggers` semblent nécessiter des corrections puisqu'ils ne sont pas acceptés par le système. Nous n'avons pas eu l'occasion d'intégrer des données réelles [2], ni d'essayer, pour le moment et ce malgré de nombreuses tentatives, le module DEAZY proposé par le logiciel DB-MAIN, vraisemblablement suite à un problème avec notre compilateur java.

6 Conclusion

En conclusion sur ce projet, nous souhaiterions avant tout insister sur son aspect intéressant et pratique. En effet, de par la possibilité de travailler sur un cas réel, nous avons pu prendre pleinement conscience des difficultés relatives à la modélisation d'un domaine d'application spécifique tout en pouvant réellement l'implémenter. Nous trouvons donc que l'aspect projet du cours est un atout majeur qu'il serait intéressant de développer au maximum pour les années à venir.

Par rapport à l'organisation du cours, nous pensons qu'avoir un TP d'exercice relatif aux `triggers` pourrait être un avantage pour les prochains étudiants, tout comme le TP sur les requêtes SQL l'est pour le cours de MA1. De plus, nous trouvons qu'il pourrait être intéressant de regrouper tous les éléments relatifs à la modélisation lors du cours de MA1 afin d'éviter toute confusion entre les modélisations permises ou non. La matière serait certes plus réduite pour le cours de MA2 qui pourrait alors être centré sur les `trigger`, les vues et le projet.

2. Surtout de par l'absence totale de cohérence au sein des données fournies par l'A.S.B.L. et nous n'avons pas eu le temps de créer des fictives.