



**Collectif francophone pour l'enseignement libre de l'informatique**

## **Modèle logique relationnel**

*Interface de programmation d'application*

**MLD\_10**

Christina KHNAISSER ([Christina.KHNAISSER@USherbrooke.ca](mailto:Christina.KHNAISSER@USherbrooke.ca))

Luc LAVOIE ([Luc.LAVOIE@USherbrooke.ca](mailto:Luc.LAVOIE@USherbrooke.ca))

*(les auteurs sont cités en ordre alphabétique nominal)*

—

*Scriptorum/Scriptorum/MLD\_10-EMIRA, version 0.1.4.f, en date du 2024-05-12*

*— document de travail, ne pas citer —*

## Sommaire

...

## Mise en garde

Le présent document est en cours d’élaboration ; en conséquence, il est incomplet et peut contenir des erreurs.

## Historique

diffusion	resp.	description
2025-05-12	LL	Revue et corrections diverses.
2024-08-09	CK	Récupération de notes diverses.

## Table des matières

Introduction.....	4
1. ÉMIR.....	5
1.1. Règles de dénomination .....	5
1.2. Évaluation.....	5
1.3. Modification .....	6
1.4. Insertion .....	7
1.5. Retrait .....	7
1.6. Synthèse .....	8
2. Comportement strict ou non.....	8
Conclusion.....	9
Références .....	10
Définitions .....	11

## Introduction

Le présent document a pour but de présenter les éléments nécessaires pour définir une interface applicative (ou interface machine-machine IMM) pour manipuler et interroger une base de données relationnelle.

*À développer*

*Évolution du document*

Le présent document tire son origine de l'expérience d'enseignement des auteurs.

*Contenu des sections*

- La section 2 présente l'ÉMIR.
- La section 3 présente son extension l'ÉMIRA

## 1. ÉMIR

Une IMM de type ÉMIR permet de consulter (évaluer, extraire), mettre à jour (modifier), créer (insérer) et déclasser (retirer) des informations (des données) relatives à chaque entité de référence depuis une base ou un entrepôt de données.

Les routines d'un ÉMIR appartiennent à l'une des catégories suivantes :

- Évaluation
- Modification
- Insertion
- Retrait

Lors du développement d'une base de données, plusieurs entités (relations) sont accompagnées d'un ensemble de routines en permettant la gestion, communément appelées collectivement CRUD (*create, retrieve, update, delete*) ou ÉMIR (évaluation, modification, insertion, retrait).

Attendu l'uniformité de comportement des routines en regard de l'entité de référence et le grand nombre de routines, il est souvent convenu de catégoriser les routines d'un ÉMIR et de leur donner des règles de dénomination afin de faciliter le repérage des routines et la systématisation de la sémantique.

Les prochaines sous-sections définissent un ensemble de règles accompagnées d'exemples.



En fait, les deux systèmes ne sont pas équivalents. Le système CRUD est un cas particulier du système ÉMIR : l'évaluation d'une expression relationnelle est plus large que la « recherche » d'un tuple dans une table.



Qu'entend-on par « entité de référence » ?

Les ÉMIR sont souvent élaborés à partir des entités du modèle conceptuel afin de faciliter l'interaction avec les ASTBD (application ou service tributaire de la BD). Ces entités sont souvent représentées par la jointure de plusieurs relations (notamment pour cause de normalisation). L'utilisation des entités conceptuelles prend ainsi en charge cette complexité et en soulage les développeurs des ASTBD.

Une entité de référence est donc soit une entité du modèle conceptuel (possiblement modélisée par plusieurs relations du modèle logique), soit une relation du modèle logique représentant un intérêt applicatif.

### 1.1. Règles de dénomination

```
<routine> ::= <entité> <catégorie> <spécialisation>
<entité> ::= IDENT
<catégorie> ::= '_EVA' | '_MOD' | '_INS' | '_RET'
<spécialisation> ::= ('_' IDENT)*
```

- L'identifiant de <entité> est le même que celui de la table principale qui la représente.
- La <catégorie> est formée des trois premières du nom de la catégorie de la routine.
- L'interprétation de la <spécialisation> est propre à chaque classe.

### 1.2. Évaluation

Une routine d'évaluation est une fonction qui retourne une valeur.

La catégorie est \_EVA.

Si aucune spécialisation n'est indiquée, aucun paramètre n'est présent et tous les tuples de l'entité sont retournés.

#### Exemple\_EVA

```
create or replace function "EMIR".Resultat_EVA ()
  returns table
  (
    matricule Matricule,
    TE        TypeEval,
    activite  SigleCours,
    trimestre Trimestre,
    note      Note
  )
begin atomic
  select * from Resultat;
end;

/*TEST*/select * from "EMIR".Resultat_EVA ();

create or replace function "EMIR".Resultat_EVA_note_accumulee
  -- Note accumulée totale d'un étudiant pour une activité et un trimestre donnés
  (_matricule Matricule, _activite SigleCours, _trimestre Trimestre)
  returns Integer
return (
  select sum(note)
  from Resultat
  where matricule = _matricule and activite = _activite and trimestre = _trimestre
);

/*TEST*/select "EMIR".Resultat_EVA_note_accumulee ('15113150', 'IFT187', '20133');
```

### 1.3. Modification

Une routine de modification est une procédure modifiant la valeur d'un ou plusieurs tuples des relations représentant l'entité.

Une modification de l'entité.

La catégorie est `_MOD`.

Si aucune spécialisation n'est indiquée, tous les attributs de la relation doivent être présents au titre de paramètre. Le tuple coïncidant avec la clé candidate détermine le tuple à être modifié. Si aucun tuple ne correspond, aucune modification n'est faite.

En outre, l'usage veut qu'une procédure soit définie pour chacun des attributs non clés afin de permettre d'en permettre la modification individuelle.

#### Exemple\_MOD

```
create or replace procedure "EMIR".Resultat_MOD
  (
    _matricule Matricule,
    _activite  SigleCours,
    _te        TypeEval,
    _trimestre Trimestre,
    _note      Note
  )
begin atomic
  update resultat
  set note = _note
  where matricule = _matricule
     and activite = _activite
```

```

        and te = _te
        and trimestre = _trimestre;
end;

/*TEST*/call "EMIR".Resultat_MOD ('15113150', 'IFT187', 'TP', '20133', 85) ;

```

## 1.4. Insertion

Une routine d'insertion est une procédure ajoutant un ou plusieurs tuples dans les relations représentant l'entité.

La catégorie est `_INS`.

Si aucune spécialisation n'est indiquée, tous les attributs de l'entité doivent être présents au titre de paramètre. Le tuple coïncidant avec la clé candidate détermine le tuple à être créé. Si un tuple correspond déjà à cette clé, la procédure lève une exception. (ce comportement est dit strict).

### Exemple `_INS`

```

create or replace procedure "EMIR".Resultat_INS
(
    _matricule Matricule,
    _activite SigleCours,
    _te TypeEval,
    _trimestre Trimestre,
    _note Note
)
begin atomic
    insert into Resultat
    ( matricule, activite, te, trimestre, note)
    values
    (_matricule, _activite, _te, _trimestre, _note);
end
;

/*TEST*/call "EMIR".Resultat_MOD ('15113150', 'IFT187', 'PR', '20133', 75) ;

```

## 1.5. Retrait

Une routine de retrait est une procédure retirant un ou plusieurs tuples des relations représentant l'entité.

La catégorie est `_RET`.

Si aucune spécialisation n'est indiquée, les attributs de la clé primaire doivent être présents. Le tuple coïncidant avec la clé candidate détermine le tuple à être retiré. Si aucun tuple ne correspond déjà à cette clé, la procédure lève une exception. (ce comportement est dit strict).

### Exemple `_RET`

```

create or replace procedure "EMIR".Resultat_RET
(
    _matricule Matricule,
    _activite SigleCours,
    _te TypeEval,
    _trimestre Trimestre
)
begin atomic
    delete from Resultat
    where matricule = _matricule
        and activite = _activite
        and te = _te

```

```

        and trimestre = _trimestre;
end;

/*TEST*/call "EMIR".Resultat_RET ('15113150', 'IFT187', 'TP', '20133') ;

```

## 1.6. Synthèse

Typiquement, pour chaque entité du modèle l'ÉMIR proposera :

- la routine `_EVA` de base et quelques routines spécialisées ;
- la routine `_MOD` de base, une routine spécialisée pour chacun des attributs non clés et quelques autres routines spécialisées ;
- la routine `_INS` de base et parfois quelques routines spécialisées ;
- la routine `_RET` de base et parfois quelques routines spécialisées.

## 2. Comportement strict ou non

Les procédures d'insertion et de retrait (voire même d'insertion) peuvent avoir un comportement strict ou non strict.

Plusieurs solutions ont été envisagées

### Explicitation du comportement en tout temps

La dénomination de la procédure comprend le type de comportement

- `'_S'` pour strict.
- `'_N'` pour non strict.
- Le comportement doit être omis pour les fonctions d'évaluation, mais toujours présent pour les procédures.

```

<routine> ::= <entité> <catégorie> <comportement> <spécialisation>
<comportement> ::= ('_S' | '_N')?

```

### Explicitation du comportement déviant

Un comportement explicite commun est documenté pour l'ensemble de l'interface, les procédures déviantes doivent porter la marque du comportement déviant.

### Paramétrage du comportement en tout temps

Un paramètre (`_strict` Boolean) est ajouté à la procédure permettant de moduler le comportement. Ce paramètre peut éventuellement avoir une valeur par défaut. Dans ce cas, il est suggéré de le placer en position finale.



## Conclusion

La définition d'IMM de type ÉMIR offre plusieurs avantages, dont ceux-ci :

- augmentation de la couverture des tests unitaires ;
- augmentation de l'indépendance lors l'évolution de la mise en oeuvre et de l'ajout des fonctionnalités aux ASTBD comme à la BD elle-même ;
- réduction de l'effort de développement ASTBD ;
- réduction du volume des transactions entre ASTBD et la BD ;
- maintien des propriétés ACID durant une plus grande partie du traitement et donc de l'adéquation des ASTBD.

## Références

### PostgreSQL\_17

- <https://docs.postgresql.fr/17/sql-createfunction.html>
- <https://docs.postgresql.fr/17/sql-createprocedure.html>

## Définitions

### ACID

Acronyme désignant conjointement les propriétés d'atomicité, de cohérence, d'isolation et rémanence (ou *durability* en anglais) relativement au traitement transactionnel.

### ASTBD

Application ou service tributaire d'une base de données.

### ÉMIR

Une interface machine-machine (IMM) permettant de consulter (évaluer, extraire), mettre à jour (modifier), créer (insérer) et déclasser (retirer) des informations (des données) depuis un système d'information (en particulier depuis une base ou un entrepôt de données).

### ÉMIRA

Variante de l'ÉMIR auxquelles s'ajoutent des routines d'archivage.

### IMM

Interface machine-machine (interface de programmation ou *application programming interface* - API), par opposition à une interface personne-machine (IPM).

Produit le 2025-05-19 05:18:31 UTC



**Collectif francophone pour l'enseignement libre de l'informatique**