Langages d'exploitation des bases de données

Projet final

DroneX

Table of Contents

Résumé 4
Problématique 4
Mise en situation 4
Pratiques de l'entreprise 5
Preuve de concept 5
Découpage du projet 6
Immersion et familiarisation du domaine 7
Drone vs UVS 7
Domaines opérationnels 7
Domaines d'application 8
Caractéristiques d'un drone 9
Équipements 10
Exemples de drone 11
Spécificités du client 16
Identification d'un drone 16
Localisation 18
Stratégie de localisation 18
Bureau des employés 19
Localisation des équipements 20
États 21
Concept d'état 21
Diagramme logique des états et de leurs transitions 22
Notes liées aux états 23
Conception préliminaire 24
Fichiers mis à votre disposition 25
Fichier dbdiagram 25
Fichiers CSV 25
Fichiers PGSQL 25
Tâches à réaliser 26
0. Analyse de la conception 26
1. Rédaction du script de création des objets fondamentaux 27
2. Remplissage des tables de référence 28

3. Remplissage des tables de référence immuables 29
4. Modèles de drone 30
5. Localisations et employés 31
6. Acquisition de drones 34
7. Gestion des états 37
8. Requêtes 38
Notions complémentaires 41
Tables de référence et tables de référence immuables 41
Notes techniques 42
Surcharge de fonctions 42
Fonctions utilitaires de PostgreSQL 42
Fonctions utilitaires mises à votre disposition 43
Manipulation de tableau avec une requête SQL 44
Exécution de scripts avec `psql` 45
Contraintes 47
Stratégie d'évaluation 47
Remise 47

Résumé

Vous devez concevoir et réaliser une base de données permettant de répondre à une problématique spécifique.

Ce projet se veut un deuxième contact avec ce que pourrait être un mandat en industrie (avec quelques simplifications évidemment). Pour y arriver, vous devrez réaliser plusieurs étapes distinctes, mais interdépendantes :

1. comprendre les besoins client
2. comprendre une conception de base de données proposée;
3. production d'une série de scripts réalisant les tâches suivants :
    a. création de l'infrastructure
    b. création de procédures et fonctions facilitant la manipulation des données
    c. création de déclencheurs variés
    d. création de procédures et fonctions simulant l'insertion de données
    e. création de requêtes répondant à des questions précises

Ce deuxième projet met l'emphase sur la notion d'automatisation des opérations liées à la base de données. Ces automatisations visent plusieurs considérations :

- augmentation de l'intégrité en offrant davantage de robustesse à ce que les six contraintes de base permettent
- offrir une interface de programmation :
    - masquant la complexité de l'infrastructure de la base de données
    - uniforme pour les divers logiciels utilisant la base de données
- automatiser certaines tâches redondantes du côté serveur :
    - remplissage automatique de champs plus complexes
    - validation de contraintes plus complexes
- réaliser de simples simulations pour alimenter la base de données

Problématique

Mise en situation

Vous travaillez pour une jeune firme cherchant à décrocher un important contrat de la compagnie `DroneX`. Ce contrat vise le développement de toutes les applications de gestion de l'entreprise. `DroneX` est une entreprise spécialisée dans la location de drones pour des domaines aussi variés que pour les loisirs, les besoins commerciaux, industriels, scientifiques et d'intervention.

Ce contrat est pour votre patron l'opportunité d'accéder aux ligues majeures. En effet, la législation liée à l'utilisation de drones a été récemment modifiée et leur usage est maintenant plus facile grâce à une règlementation stricte. Depuis, l'utilisation des drones dans l'industrie a explosé et `DroneX` se trouve en situation de croissance rapide. Votre entreprise a réussi à franchir les premières étapes du processus d'appel d'offres et elle se situe parmi les finalistes. La décision définitive se prendra sur l'évaluation d'une preuve de concept qui doit être présentée.

Pratiques de l'entreprise

Pour mieux cibler le projet, il est important de comprendre que l'entreprise vise le développement d'au moins cinq applications internes et une application externe visant chacune des tâches différentes :

- applications internes :
    - administration : principalement axé sur les opérations financières de l'entreprise
    - ressources humaines : principalement axé sur la gestion des employés
    - ventes : principalement axé sur la gestion des clients
    - inventaire : principalement axé sur les fournisseurs et les stocks dans l'entreprise
    - opération : principalement axé sur l'état des équipements et des opérations en cours
- application externe disponible via un navigateur web :
    - portail client : offrant aux clients l'opportunité
        - d'avoir une vue d'ensemble sur ses opérations (« *dashboard* »)
        - faire certaines opérations (location, …)

Pour `DroneX`, ce développement est l'opportunité de consolider les outils logiciels existants et, surtout, de mettre de l'avant certaines pratiques importantes et spécifiques à la culture de l'entreprise. Notamment :

- sa stratégie d'identification de localisation des bureaux dans ses divers immeubles
- la méthode d'identification d'un drone
- le concept de gestion des opérations basé sur l'état d'un drone normalisant une pratique opérationnelle stricte

Preuve de concept

Le leader technique a identifié que la plus grande valeur du projet est de centraliser :

- les données
- les opérations sur les données

Cette approche met l'accent sur une architecture centralisée utilisée par des outils logiciels décentralisés.

Cette architecture met l'emphase sur le développement d'outils génériques d'accès aux données en offrant une interface de programmation optimale pour tous les développeurs d'application. Cette approche permet le développement d'un noyau robuste, uniforme et puissant pour tous les logiciels applicatifs.

En fait, ce noyau supporte entièrement la logique d'affaire de l'entreprise. C'est le « *backend* » qui supporte les opérations fondamentales. Les différentes applications à développer ne sont que des portails utilisateurs offrant des interfaces graphiques usager interagissant simplement avec le système, le « *frontend* ».

Baser une preuve de concept avec une telle approche est audacieuse, mais permet de démontrer la puissance du produit envisagé et la qualité de l'équipe de développement.

Sachant que le client connaît déjà les enjeux techniques du développement logiciel à court, moyen et long terme et qu'il envisage une longue durée de vie pour ce projet, il sera facile de justifier cette approche.

Découpage du projet

Pour réaliser la preuve de concept, le leader technique divise l'équipe de développement en trois équipes de développement :

1. le noyau, le « *backend* » et le moteur de l'entreprise
2. l'application interne liée aux opérations, un premier exemple de « *frontend* »
3. l'application externe du portail client, un deuxième exemple de « *frontend* »

Évidemment, toutes ces parties ne seront développées que partiellement et présenteront au client un aperçu du potentiel de la solution envisagée.

Pour des raisons évidentes de démonstration et pour des considérations techniques, certains outils supplémentaires devront être mis en place pour permettre un accès à des données venant des systèmes non-développés. De plus, une simulation permettra l'insertion de données pertinentes pour faire la démonstration.

Dans le but de livrer une démonstration minimale, mais fonctionnelle du système, le travail à réaliser est axé sur les opérations quotidiennes de l'entreprise. Ainsi, on désire démontrer nos capacités à produire les parties névralgiques du projet.

Pour cette mise en situation, vous faites partie de l'équipe 1.

Immersion et familiarisation du domaine

Lors du développement d'un projet technique, il est indispensable de se familiariser avec le domaine technique du client. Il est essentiel de développer un minimum de vocabulaire technique afin de comprendre le client et de discuter avec lui. C'est un gage minimum de respect, de compétence et de professionnalisme.

Drone vs UVS

L'historique de l'entreprise explique l'utilisation du mot drone au sens général. `DroneX` a démarré ses activités avec la vente et la location de drones aériens dédiées à la photographie. Maintenant l'entreprise adopte entièrement le terme générique d'UVS (« ***U**nmanned **V**ehicle **S**ystem* ») à même l'utilisation du mot drone.

Domaines opérationnels

Le diagramme suivant présente les différents domaines opérationnels des drones classiques. Il est important de comprendre que certains drones font partie de plusieurs domaines opérationnels (cas classique des drones amphibies).

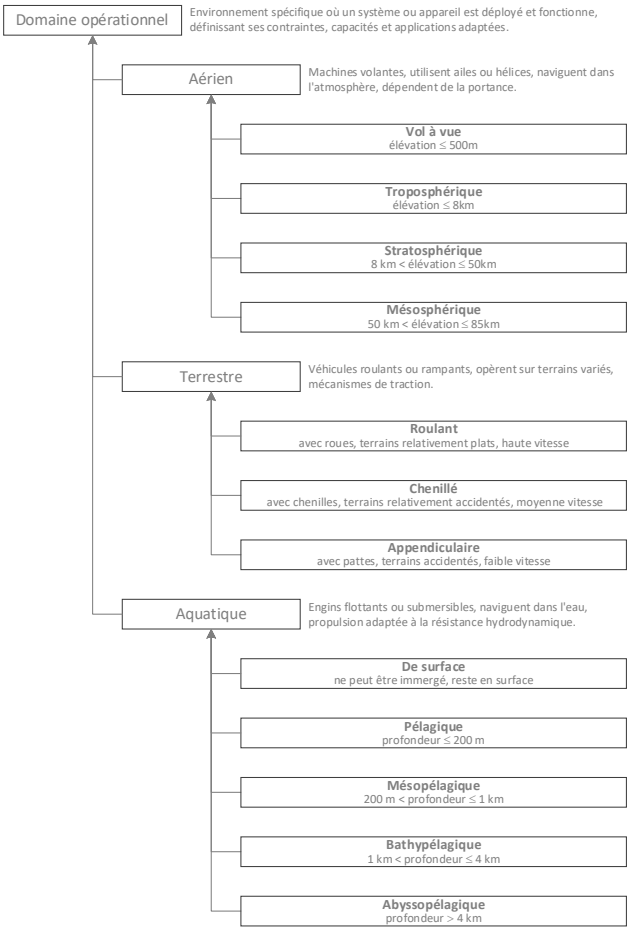


Diagramme présentant les différents domaines opérationnels

Domaines d’application

La liste suivante présente des domaines d’application possibles. La liste est non-exhaustives et n’est pas détaillée.

- Inspection et maintenance
    - Infrastructure
    - Industrielle
    - Réseaux
    - Maritime
    - Ferroviaire
- Photogrammétrie, cartographie et modélisation
    - Relevés topographiques
    - Mesures bathymétriques
    - Modélisation 3D
- Surveillance et sécurité
    - Surveillance des frontières
    - Surveillance urbaine
    - Sécurité des installations
- Recherche et sauvetage
    - Opérations de sauvetage en montagne
    - Recherche en milieu aquatique
    - Détection de personnes disparues
- Agriculture et gestion des ressources naturelles
    - Agriculture de précision
    - Gestion des forêts
    - Gestion des ressources halieutiques (pêche)
- Communication et réseautage
    - Extension de réseaux mobiles
    - Systèmes de communication d'urgence
- Transport et livraison
    - Livraison de colis
    - Transfert médical
- Loisirs et divertissement
    - Courses de drones
    - Photographie et vidéographie aériennes
    - Spectacle de drones
- Enseignement et formation
    - Formation au pilotage
    - Éducation multidisciplinaire (Science, Technologie, Ingénierie et Mathématiques)
- Sondage et exploration
    - Exploration minière
    - Exploration archéologique
- Études environnementales et climatologiques
    - Surveillance de la qualité de l'air

    - Suivi des changements climatiques
- Médias et reportage
    - Journalisme d'investigation
    - Couverture d'événements en direct
- Santé et soins médicaux
    - Livraison de fournitures médicales
    - Transport d'échantillons médicaux
- Intervention d'urgence
    - Gestion des catastrophes
    - Évacuations médicales
- Opérations commerciales
    - Publicité
    - Logistique

Caractéristiques d'un drone

La liste suivante présente les caractéristiques importantes à connaître lorsqu'on évalue l'utilisation d'un drone. Encore une fois, cette liste est largement non-exhaustive.

1. Autonomie de déplacement :
   Mesure du temps que le drone peut passer en déplacement sans ravitaillement.
2. Autonomie décisionnelle :
   Détermine le niveau de supervision nécessaire à son opération.
3. Portée de communication :
   Distance maximale à laquelle le drone peut être contrôlé à partir de la station au sol.
4. Capacité de charge utile :
   Poids maximal que le drone peut transporter en plus de son propre poids.
5. Résolution de la caméra :
   Qualité des images ou vidéos que le drone peut capturer.
6. Vitesse maximale :
   Vitesse maximale que le drone peut atteindre lors de ses déplacements.
7. Stabilité de déplacement :
   Aptitude du drone à maintenir une trajectoire de déplacement stable dans diverses conditions environnementales.
8. Maniabilité :
   Facilité avec laquelle le drone peut être piloté et effectuer des mouvements complexes.
9. Systèmes de navigation :
   Les méthodes utilisées pour la localisation et la navigation, comme le GPS ou la navigation inertielle.
10. Résistance aux intempéries :
    Capacité du drone à fonctionner dans des conditions météorologiques défavorables comme la pluie et le vent fort.
11. Fiabilité :
    Probabilité que le drone fonctionne sans défaillance sur une période de temps donnée.

12. Coût réel :
    Prix total de possession, incluant l'achat initial, la maintenance, les coûts d'opération, les éventuelles mises à niveau et le remplacement de pèces.
13. Redondance des systèmes :
    Présence de systèmes de secours pour les fonctions critiques comme la propulsion ou la navigation.
14. Sécurité des données :
    Mesures prises pour protéger les données collectées ou transmises par le drone.
15. Interfaçage avec d'autres systèmes :
    Capacité du drone à se connecter et à fonctionner en tandem avec d'autres drones, d'autres systèmes ou plateformes.
16. Réglementation et conformité :
    Aptitude du drone à satisfaire aux normes et réglementations locales ou internationales.
17. Extensibilité :
    Possibilité d'ajouter des fonctionnalités supplémentaires ou des modules au drone.

Équipements

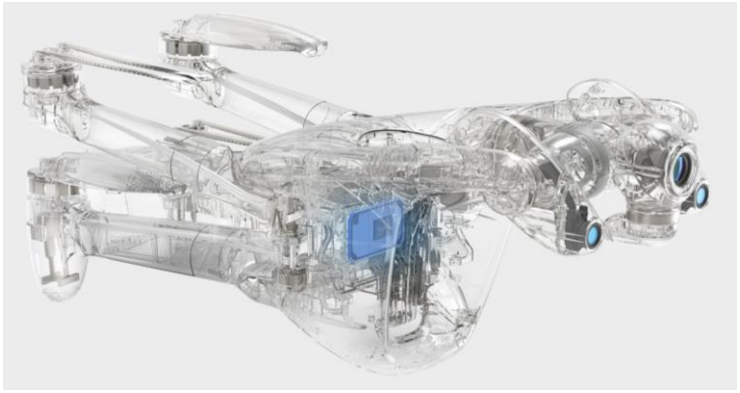
Les équipements possibles sur un drone sont très variés et ne sont pas considérés pour ce projet.

Exemples de drone

Voici quelques exemples réels de drone.

DJI Enterprise
Phantom 4 RTK
(voir)

Utilisé principalement pour la photogrammétrie et la cartographie.



Parrot
Anafi
(voir)

Utilisé pour la photographie, la photogrammétrie et la vidéographie

DJI Enterprise
Agras T40
(voir)

Ce drone est spécifiquement conçu pour l'agriculture et est capable de pulvériser des liquides sur de grandes étendues de terres agricoles.

Il est équipé de multiples buses de pulvérisation et peut couvrir une grande surface en un seul vol, améliorant ainsi l'efficacité de l'arrosage et de l'application de pesticides.

AgEagle
eBee X
(voir)

Employé pour la cartographie et les relevés topographiques.

Parrot
Disco

Utilisé pour le loisir et la vidéographie.

Quantum Systems
Trinity F90+
(voir)

Principalement utilisé pour la cartographie et la surveillance.

Lilium
Jet
(voir)

Un drone taxi visant le transport de passagers, encore en phase de développement mais avec des essais de vol réussis.

Présente aussi la capacité de faire des décollage et atterrissage verticaux.

Bell
V-247 Vigilant
(voir)

Un concept de drone militaire à décollage et atterrissage verticaux.

CLEARPATH Robotics
Jackal
(voir)

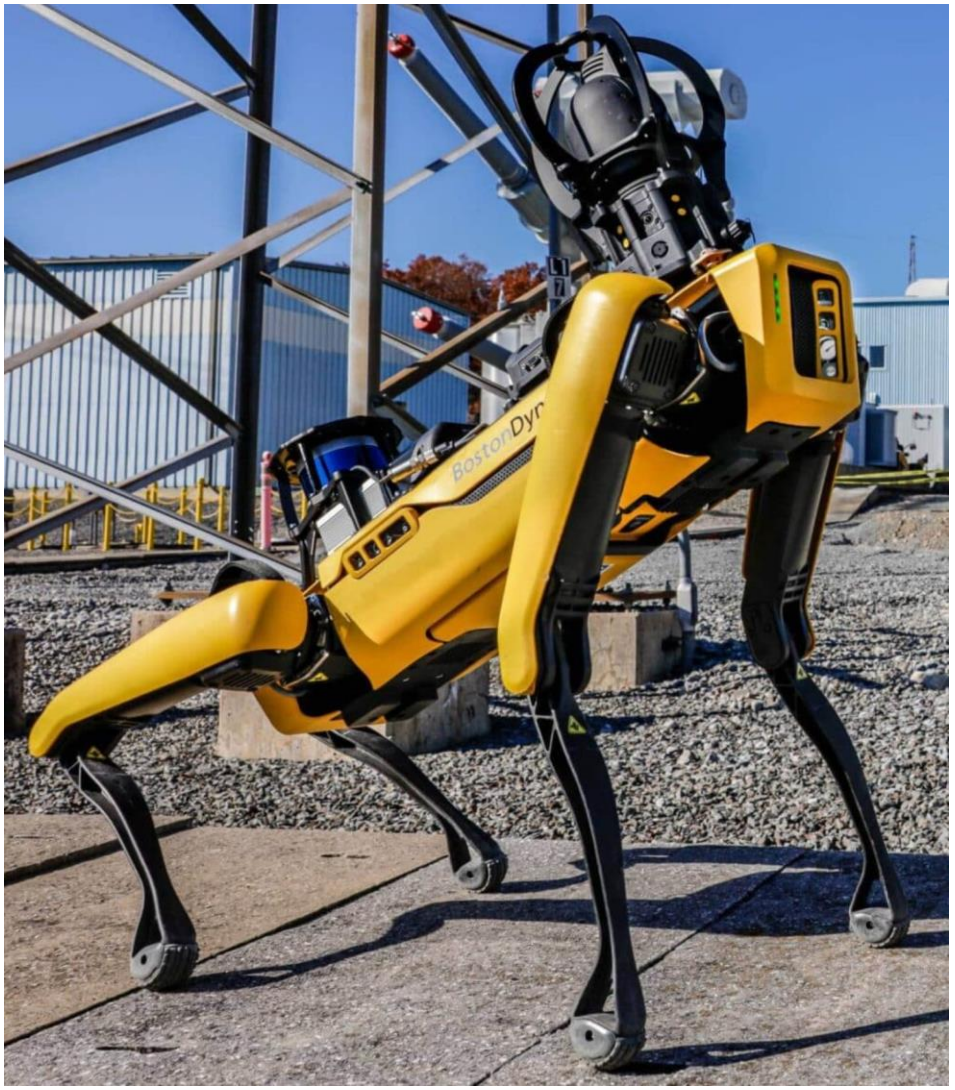
Utilisé dans la recherche et le développement.

Teledyne
PackBot 510
(voir)

Employé pour la détection d'explosifs et les opérations de recherche et de sauvetage.

Boston Dynamics
Spot
(voir)

Employé pour la recherche, ains que pour l'inspection et la surveillance de sites industriels.



OceanAlpha
SL40
(voir)

Utilisé pour les mesures bathymétriques et les études environnementales.

Blueye
X3
(voir)

Employé pour l'inspection sous-marine et la recherche.

Deep Trekker
Revolution NAV
(voir)

Conçu pour des inspections en profondeur et la recherche en environnement abyssal.

NASA
DragonFly
(voir)

Un projet de la NASA prévu pour étudier la lune Titan de Saturne. Ce drone, en phase de conception, est censé voler dans l'atmosphère de Titan pour collecter des données.

Spécificités du client

La compagnie `DroneX` existe depuis un bon moment et elle possède sa propre culture. Elle a déjà des habitudes spécifiques bien ancrées à travers toutes ses opérations.

Il est essentiel de les connaître afin que le projet soit conforme et respecte les pratiques internes.

Identification d'un drone

Lors de l'acquisition d'un drone, ce dernier possède déjà un numéro de série établie par l'entreprise manufacturière. Toutefois, `DroneX` baptise chacun de ses drones et appose deux étiquettes permettant son identification. La nomenclature `drone id` est utilisée pour faire référence à cet identifiant.

Le `drone id` est exactement de 20 caractères et construit selon cette règle :

`AAABBB-CCCCCC-DDDDDD`

- `AAA`
  - toujours 3 caractères
  - les 3 premières lettres en majuscule de l'entreprise manufacturière du drone
  - si le nom de l'entreprise possède moins de 3 lettres, le caractère `x` minuscule est utilisé comme substitue pour toutes les lettres manquantes
  - si le nom de l'entreprise possède des espaces ou caractères spéciaux, il sont supprimés avant l'analyse
- `BBB`
  - toujours 3 caractères suivis d'un tiret
  - les lettres suivantes du nom du modèle de drone en majuscule :
    - la première
    - celle à la position médiane
    - la dernière lettre
  - si le nom du modèle possède 2 lettres, on substitue celle du centre par un `x` minuscule et les lettres existantes sont positionnées au début et à la fin
  - si le nom du modèle possède 1 lettre, on substitue la première et la dernière par un `x` minuscule
  - si le nom du modèle possède des espaces ou caractères spéciaux, il sont supprimés avant l'analyse
- `CCCCCC`
  - toujours 6 caractères suivis d'un tiret
  - le nombre de jours écoulés entre la date d'achat et le 1er janvier 2000
  - lorsqu'il y a moins de 6 chiffres, on fait du remplissage (« *padding* ») à gauche avec le caractère `0`
- `DDDDDD`
  - toujours 6 caractères
  - un numéro séquentiel parmi tous les drones de l'entreprise débutant à 1000 et incrémentant de 10
  - lorsqu'il y a moins de 6 chiffres, on fait du remplissage (« *padding* ») à gauche avec le caractère `0`

    - chaque caractère de `0` à `9` est substitué ainsi :
        - `0-1-2-3-4-5-6-7-8-9`
          `↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓`
        - `Z-U-D-T-Q-C-S-P-H-N`

Par exemple, le drone ayant ces informations :

- Entreprise manufacturière : Parrot
    - => PAR
- Modèle : Disco
    - => DSO
- Date d'achat : 12 avril 2020
    - 7407 jours écoulés depuis le 1 janvier 2000
    - => 007407
- Le n^e acheté par l'entreprise : 2772
    - donne le numéro 028710
    - =>ZDHPUZ

Sera nommé :

`PARDSO-007407-ZDHPUZ`

Localisation

Stratégie de localisation

`DroneX` a mis en place un système de localisation de ses bureaux, espaces de travail, ateliers et entrepôts qui permet une reconnaissance facile des lieux. L'information est mise dans une chaîne de caractères faciles à décodée nommée « *location tag* » ou, en français, *étiquette de localisation*.

L'approche consiste à préciser la localisation de l'endroit la plus vague vers l'endroit la plus précise. Le « *location tag* » est construite selon ces principes importants :

1. immeuble
2. étage
3. zone de l'immeuble, identifié par un code référant à une couleur
4. numéro de salle
5. identifiant spécialisé selon le contexte utilitaire dans la salle

La chaîne de caractères est construite ainsi :

```
AA BBB.CCC-DDD[.xxxxx]
```

Où :

1. `AA`
    - correspond à l'immeuble concerné
    - toujours deux caractères, suivi par un espace
    - l'entreprise ne possède que deux immeubles identifiés par :
        - `GZ`
        - `XB`
    - la validation de ces chaînes est obligatoire
    - l'entreprise vise étendre ses opérations et faire l'acquisition d'autres immeubles dans un avenir rapproché
2. `BBB`
    - correspond à l'étage
    - toujours trois caractères, suivi par un point
    - pour les étages ayant moins de 3 chiffres, on fait du remplissage (« *padding* ») à gauche avec le caractère `0`
    - pour le rez-de-chaussée, le code 000 est utilisé
    - pour les étages sous-terrain, le premier caractère est S et le niveau est identifié par seulement les deux derniers caractères
3. `CCC`
    - la zone de l'immeuble, représenté aussi par une couleur
    - toujours trois caractères, suivi par un tiret
    - les codes sont les suivants :
        - `RED` : rouge
        - `GRE` : vert
        - `BLU` : bleu
        - `YEL` : jaune

        - `MAG` : magenta
        - `ORA` : orange
        - `WHI` : blanc
        - `BLA` : noir
    - il est envisageable que d'autres couleurs soient spécifiées dans le futur.
4. `DDD`
    - correspond au numéro de salle
    - le numéro est un nombre entre 100 et 899 (les bornes sont inclusives)
    - puisque chaque zone est divisée en 8 sections, la centaine indique la section dans la zone
    - les deux derniers chiffres représentent :
        - une position relative de la salle par rapport à une autre dans la même zone
        - la valeur incrémentale indique une distribution suivant un parcours circulaire
        - autant que possible, suivant le sens horaire des points cardinaux (nord, est, sud et ouest).
5. `[.xxxx]`
    - les crochets `[…]` indiquent que cette partie est optionnelle selon le contexte
    - correspond à un ajout spécialisé (dépend du contexte)
    - de taille variable (dépend du contexte)
    - lorsque présent, toujours précédé par un point

Sans la partie optionnelle, l'étiquette possède exactement 14 caractères.

Bureau des employés

Le bureau d'un employé se trouve dans une salle suivant la nomenclature suivante pour le motif `[.xxxxx]` :

`.ABB`

Où :

- `A`
    - un code pour le type d'espace
    - toujours précédé d'un point
    - toujours un caractère de `A` à `J`
- `BB`
    - représente une position relative de l'espace dans la salle, comme pour `DDD` :
        - la valeur incrémentale indique une distribution suivant un parcours circulaire
        - autant que possible, suivant le sens horaire des points cardinaux (nord, est, sud et ouest).
    - toujours deux caractères représentant un nombre entre `10` à `89`

Il existe un bureau par défaut pour les nouveaux employés. Il s'agit du bureau :

`GZ 000.WHI-100.A10`

Au total, la chaîne de caractère a une longueur de 18 caractères.

Localisation des équipements

Les équipements se trouvent dans une salle suivant la nomenclature suivante pour le motif `[.xxxxx]` :

`.ABCDD`

Où :

- `A`
  - un code pour le type d'espace
  - toujours précédé d'un point
  - toujours l'un de ces six symboles : `<`, `^`, `>`, `v` (la lettre `v` minuscule), `_` (le soulignement) ou `x` (la lettre `x` minuscule)
- `B`
  - indique l'endroit de stockage dans la pièce
  - toujours un seul caractère
  - il ne peut y avoir que 20 étagères par salle, donc un code de 1 à 20
  - représenté par une lettre de `A` à `T`
- `C`
  - Indique la hauteur de la tablette
  - toujours un seul caractère, une lettre dont la position alphabétique représente la hauteur à partir du sol
  - par exemple, la lettre `A` indique la première tablette, `B` indique la deuxième tablette et ainsi de suite
  - La lettre `Z` est une exception représentant l'espace de stockage directement au sol, il ne peut donc y avoir que 25 tablettes au maximum
- `DD`
  - représente le casier de stockage
  - toujours deux caractères
  - un nombre de `00` à `99`
  - lorsque le nombre est inférieur à 10, le premier caractère est `0`.

La localisation par défaut des équipements est le quai de réception. Ce dernier est indiqué ainsi :

`XB 000.MAG-600.^IZ00`

Au total, la chaîne de caractère a une longueur de 20 caractères.

Lorsque l'équipement est loué ou perdu, l'endroit d'entreposage est mis à la valeur nulle.

États

Concept d'état

Dans l'entreprise, chaque drone peut se retrouver dans un état de traitement spécifique. Cet état communique des informations sur la situation dans laquelle le drone se trouve. Par exemple :

- il est possible de savoir à quel endroit il se trouve
- s'il est disponible pour la location
- s'il a des tâches spécifiques à faire comme une inspection, des tests, des réparations, etc.
- qui est le dernier employé ayant travaillé sur le drone
- depuis combien de temps le drone est dans cet état

De plus une mécanique rigide est en place pour qu'un drone puisse passer d'un état à un autre, mais seulement d'un état valide à un état valide. Cette approche est basée sur le fait qu'à un état spécifique, il existe deux seuls états suivants possibles correspondant au succès des tâches reliées à l'état courant :

- accepté : passe à l'état suivant normal
- refusé : un problème est arrivé, on doit retourner à un état antérieur de validation

Finalement, il existe deux états particuliers qui n'ont pas d'états suivants. Ces états sont nommés états terminaux. Il s'agit des cas où le drone est :

- hors d'usage : il est utilisé à l'interne pour ses pièces de remplacement
- perdu : le drone a été volé à l'interne ou perdu/volé par le client lors de la location

Ce mécanisme rigoureux a été mis en place depuis quelques années suite à l'inconsistance des opérations à l'interne. Trop d'oublies ont laissé place à des erreurs humaines qui ont eu des impacts financiers et réputationnels importants et dommageables pour l'entreprise. Cette approche contraignante assure une qualité de production uniforme à travers l'entreprise, les employés et le temps.

Diagramme logique des états et de leurs transitions

Le diagramme suivant présente les états existants et leurs liaisons.

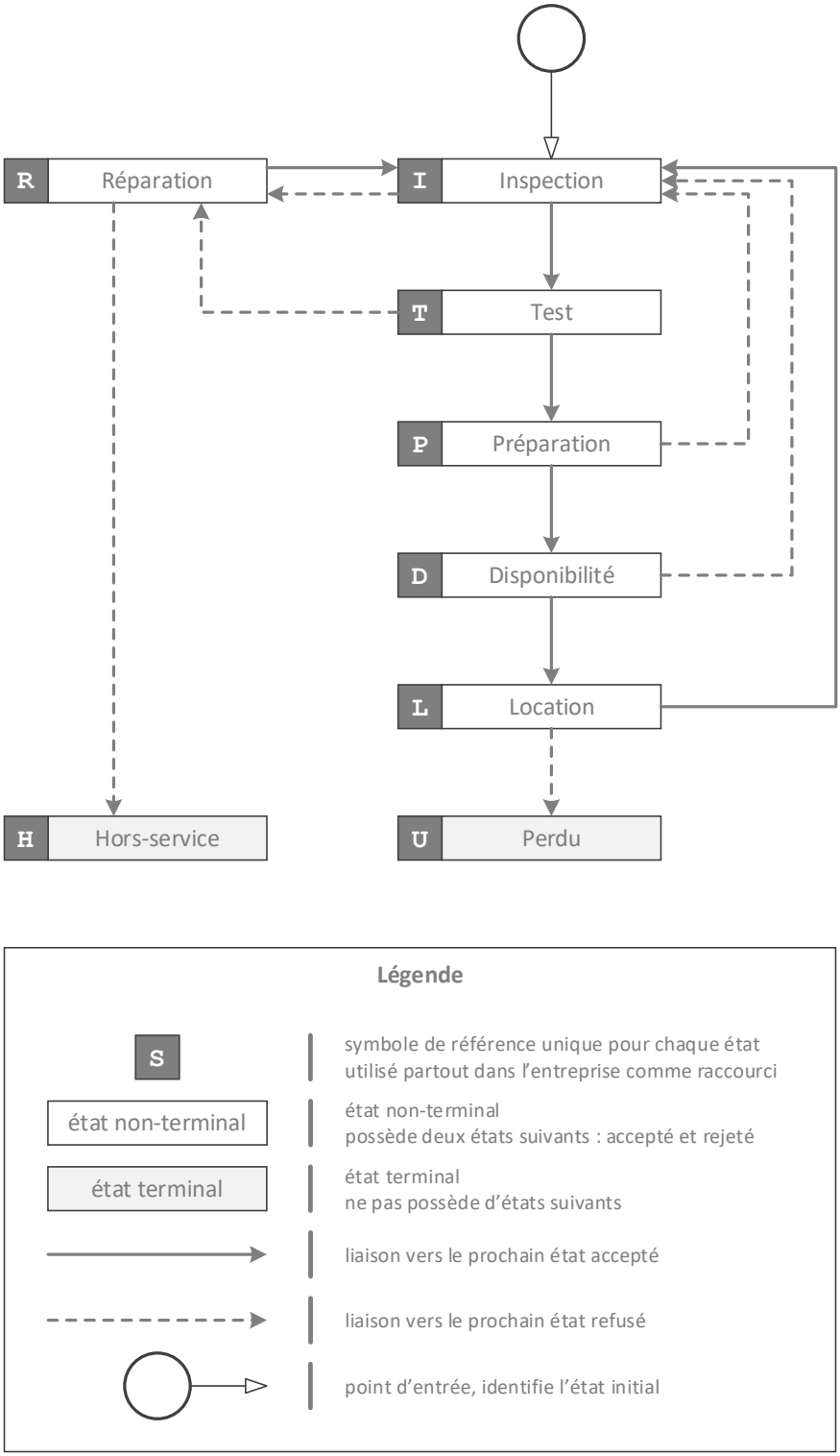


Diagramme d'états

Notes liées aux états

Pour plusieurs états, les pratiques de l'entreprise obligent que des commentaires soient présents pour donner des détails sur les évènements liés. Ces notes sont catégorisées en 5 types :

- observation générale
- observation problématique
- entretient effectué
- réparation effectuée
- équipement remplacé.

Pour chaque note, il existe un ensemble d'informations pertinentes aidant l'entreprise à améliorer ses pratiques internes. Pour chaque note on retrouve :

- la note fait référence à quel robot et à quel état dans le temps
- la catégorie de note
- la date liée à la note
- l'employé qui a fait la note
- un texte donnant les détails sur l'évènement.

De plus, il est fréquent d'avoir plusieurs notes sur un même état-drone. Ces notes sont complémentaires et structurées de façon à servir à des outils d'automatisation. Par exemple, ces informations pourront servir à alimenter :

- des outils de recherche pour identifier des problèmes récurrents
- des outils de recherche pour identifier des solutions éprouvées à un problème
- une intelligence artificielle
- des statistiques d'opération
- des rapports de coûts et d'opportunités.

Finalement, il y a des règles strictes concernant certaines transitions. Lorsqu'on passe d'un état à un autre, une validation est faite pour s'assurer que des notes essentielles ont été saisies dans des contextes précis. Voici les notes obligatoires à avoir selon le contexte :

- dans tous les cas où une transition de type *rejeté* passe d'un état à un autre :
    - une note d'observation problématique
- pour l'état **R**éparation :
    - une note d'observation problématique
    - et, soit une note de réparation ou d'entretient effectué

Conception préliminaire

Voici la conception préliminaire d'une partie de la base de données réalisée par le responsable technique.

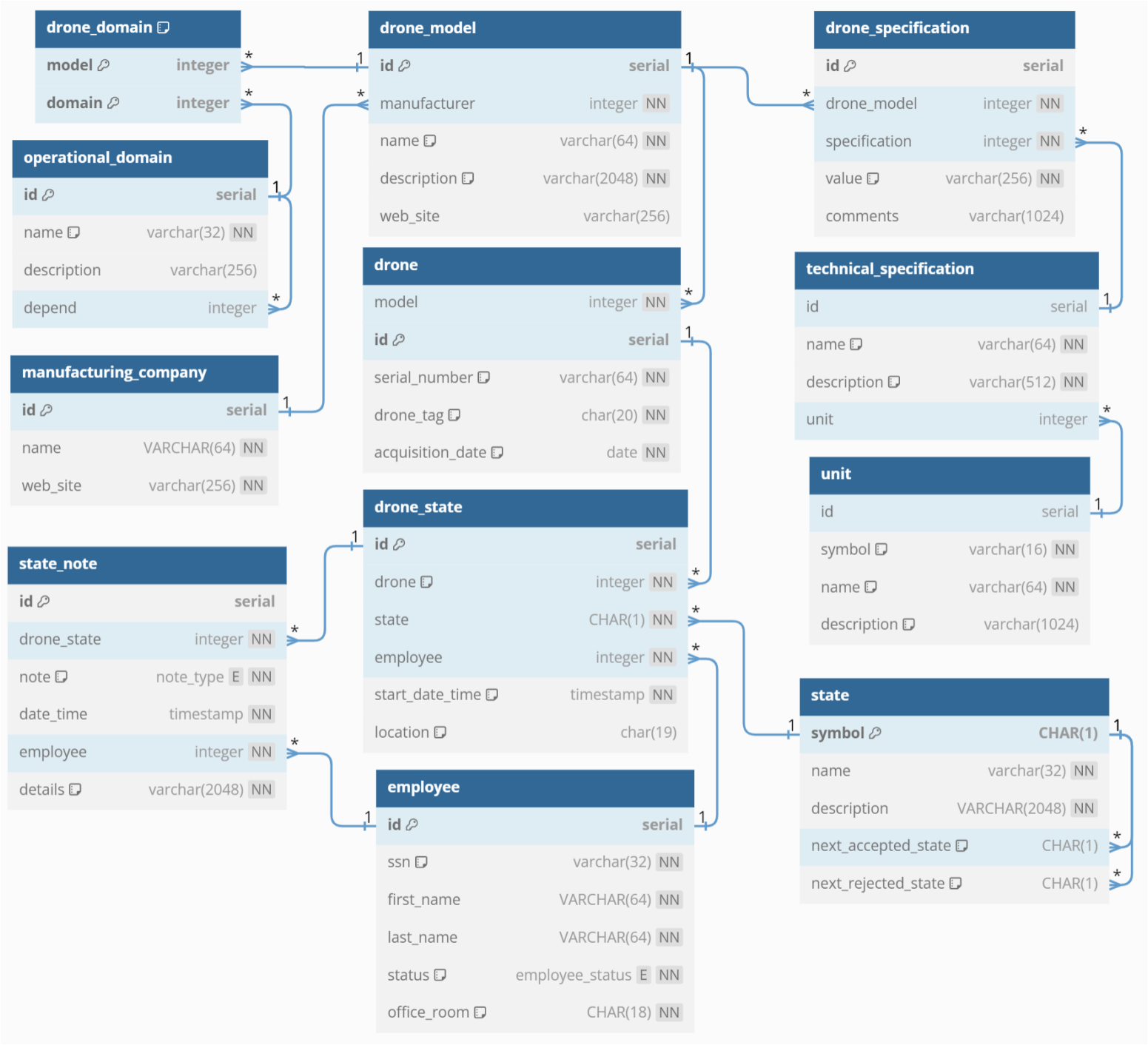


Diagramme relationnel de la base de données

Fichiers mis à votre disposition

Fichier dbdiagram

Vous avez en pièce jointe le fichier texte permettant de produire le diagramme `dbdiagram` de la page précédente. Ce document est précieux car il contient, sous forme de notes et commentaires, plusieurs informations complémentaires non visibles sur le schéma.

Ce fichier s'intitule `C42-P2_script_dbdiagram.txt`.

Fichiers CSV

Vous avez à votre disposition plusieurs fichiers `CSV` contenant les données à insérer dans les tables. Ces fichiers sont donnés pour faciliter le travail à réaliser.

- `states.csv` : les états possibles — données immuables
- `operational_domain.csv` : les domaines opérationnels — données immuables
- `units.csv` : les unités de mesure disponibles — vous pouvez en ajouter
- `technical_specifications.csv` : les spéc. tech. disponibles — vous pouvez en ajouter
- `drone_manufacturers.csv` : les manufacturiers de drone — vous pouvez en ajouter
- `drone_models.csv` : plusieurs modèles de drone existants — vous pouvez en ajouter
- `drone_model_specifications.csv` : quelques spécifications techniques sur les drones — vous pouvez en ajouter

Il vous appartient de transformer ces données afin de créer un script fonctionnel en `SQL`. Il ne suffit pas de charger ces données directement à partir du fichier `CSV`.

Plusieurs approches sont possibles :

- faire les adaptations manuellement
- faire les adaptations à l'aide d'outil spécialisé comme `Excel` et `Matlab`
- faire les adaptations via un programme informatique personnalisé réalisé par vous (en `Python` par exemple)
- faire les adaptations à l'aide de langages spécialisés ou de bibliothèques spécialisées comme le langage `R` ou les bibliothèques `NumPy` ou `Panda` de `Python`.

Prenez l'approche qui vous semble la plus pertinente selon le contexte.

Fichiers PGSQL

Vous retrouverez dans ces fichiers des outils génériques à la programmation qui sont susceptibles de vous aider pour la réalisation du projet. Sont présentes quelques procédures et fonctions utilitaires classiques servant à faciliter certains calculs et à la production de données.

- `C42-P2_00_MATH_UTILITIES.pgsql` — fonctions mathématiques simples
- `C42-P2_00_RANDOM_UTILITIES.pgsql` — fonctions de génération aléatoire

Tâches à réaliser

Le projet se découpe en tâches progressives et, pour chacune d'entre elles, vous avez des scripts à produire. N'oubliez pas de mettre un en-tête pertinent pour chacun, de respecter la norme de codage et de les documenter intelligemment.

Plusieurs éléments sont à prendre à considérations :

- La plus grande partie du DDL se retrouve dans le script 1. Toutefois, plusieurs autres objets sont ajoutés dans la base de données un peu partout ailleurs. Pour **tous** les scripts concernés, vous devez détruire les objets s'ils existent (`DROP IF EXISTS`) avant leur création (`CREATE`).
- ...

0. Analyse de la conception

Il vous incombe d'examiner la conception proposée et de discerner les relations entre ce schéma et l'objectif du projet en accord avec les usages de l'entreprise.

Assurez-vous d'avoir une compréhension suffisante du projet et de la conception afin qu'il n'y ait pas d'ambiguïtés. Posez les questions le plus rapidement possible. D'abord en équipe, ensuite avec le responsable technique (rôle joué par l'enseignant).

Cette conception est relativement simple, mais répond aux exigences minimales d'une preuve de concept. Quoique largement incomplète, elle possède tout de même l'essentiel d'une infrastructure rigoureuse et stricte pour faire les démonstrations requises.

Un des points importants de ce projet est d'observer que les contraintes standards de `SQL` sont malheureusement insuffisantes pour garantir l'intégrité d'un projet réel d'entreprise. Votre rôle est de faire le lien entre cette conception et toutes les nuances données dans les explications précédentes. Les étapes suivantes sont suffisamment détaillées pour vous aider à bien comprendre comment automatiser les éléments manquants.

La conception donnée est considérée complète et permet la réalisation du projet. Il ne faut pas créer d'autres tables.

1. Rédaction du script de création des objets fondamentaux

Vous devez créer un script créant les objets fondamentaux du projet :

- les énumérations
- les tables
- les colonnes
- les 6 contraintes disponibles en `SQL` :
    - valeur obligatoire non nulle
    - unique
    - validation
    - valeur par défaut
    - clé primaire
    - clé étrangère

  N'oubliez pas de respecter la norme de codage pour la déclaration de ces contraintes.

- les vues[1]
- les index[1]

Ce script doit être intitulé : `C42-P2_01_CREATE_INFRA.pgsql`

Attention, vous avez le droit d'utiliser `dbdiagram` pour créer le code `SQL`. Toutefois, c'est votre responsabilité de l'ajuster, de l'adapter et de le formater afin qu'il réponde exactement à l'énoncé et à la norme de codage utilisée dans le cadre du cours.

[1] On vous demande de créer $\lfloor n / 2 \rfloor$ vues pertinentes et $\lceil (n + 1) / 2 \rceil$ index pertinents (où *n* fait référence au nombre d'étudiants dans l'équipe). Il est suggéré de réaliser la création des vues et des index plus tard durant le développement du projet. La suite du projet vous donnera des indices importants sur quelles vues et quels index peuvent être pertinents. L'utilisation du `DQL` et l'analyse de fréquence est certainement l'indice le plus important.

2. Remplissage des tables de référence

On vous demande de créer un script insérant les données des tables de référence `unit` et `technical_specification`. Pour comprendre tous les détails, voir la section Tables de référence et tables de référence immuables.

Ce script doit réaliser :

- une première section avec des opérations du `DML`
  - via une première transaction, insérer le contenu des données de la table `unit`
  - via une deuxième transaction, insérer le contenu des données de la table `technical_specification`
- une deuxième section avec des opérations du `DDL`
  - créer la fonction du déclencheur `forbid_dml_operations`
  - créer les deux déclencheurs demandés associés aux tables `unit` et `technical_specification`
  - n'oubliez pas de supprimer les objets nécessaires pour qu'ils puissent être créés de nouveau
- Optionnellement, si vous le jugez pertinent, vous pouvez créer des requêtes préparées, procédures ou fonctions facilitant le travail à faire. Attention, ces outils temporaires doivent être supprimés de la base de données lorsque leur usage n'est plus pertinent.

Vous avez à votre disposition deux fichiers `CSV` contenant les données à insérer. Vous pouvez en ajouter d'autres si vous le voulez. Ces fichiers sont nommés :

- `units.csv`
- `technical_specifications.csv`.

Ce script doit être nommé :

`C42-P2_02_POPULATE_REFERENCE.pgsql`

3. Remplissage des tables de référence immuables

On vous demande de créer un script insérant les données des tables de référence `state` et `operational_domain`. Pour comprendre tous les détails, voir la section Tables de référence et tables de référence immuables.

Ce script doit :

- une première section avec des opérations du `DML`
    - via une première transaction, insérer le contenu des données de la table `operational_domain`
        - observation, il y a une notion de dépendance circulaire ici :
            - au niveau du `DDL` à cause de la clé étrangère `depend`
            - toutefois, il n'y a pas de problème au niveau du `DML` pour les opérations d'insertion – les données sont hiérarchisées d'une façon rendant possible les insertion sans problème
    - via une seconde transaction, insérer le contenu des données de la table `state`
        - attention, il y a une notion de dépendance circulaire ici :
            - au niveau du `DDL` à cause des clés étrangères `next_accepted_state` et `next_rejected_state`
            - aussi au niveau du `DML` pour les opérations d'insertion à cause de la nature même des données – les données font référence les unes aux autres
            - ici, on vous demande de régler le problème proprement et vous ne pouvez pas simplement désactiver et réactiver les `triggers`
- une deuxième section avec des opérations du `DDL`
    - on suppose la fonction du déclencheur `forbid_dml_operations` créée et disponible
    - créer les deux déclencheurs demandés associés aux tables `state` et `operational_domain`
    - n'oubliez pas de supprimer les objets nécessaires pour qu'ils puissent être créés de nouveau
- Optionnellement, si vous le jugez pertinent, vous pouvez créer des requêtes préparées, procédures ou fonctions facilitant le travail à faire. Attention, ces outils temporaires doivent être supprimés de la base de données lorsque leur usage n'est plus pertinent.

Vous avez à votre disposition deux fichiers `CSV` contenant les données à insérer. Vous ne pouvez pas ajouter d'autres données. Ces fichiers sont nommés :

- `states.csv`
- `operational_domains.csv`.

Ce script doit être nommé :

`C42-P2_03_POPULATE_REF_IMMUTABLE.pgsql`

4. Modèles de drone

On vous demande de mettre en place certains outils et les données concernant les modèles de drones. Ce script est divisé en deux sections différentes :

- Vous devez créer trois fonctions/procédures utilitaires simplifiant les insertions de données concernant les modèles de drone. Vous devez créer les fonctions/procédures répondant aux déclarations suivantes :
  - `add_drone_manufacturer(name_value, web_site_value)`
    - `name_value`, le nom de l'entreprise manufacturière
    - `web_site_value`, l'adresse web de l'entreprise
    - ne retourne rien
    - effectue l'insertion d'une entreprise manufacturière de drone
  - `add_drone_model(name_value, manufacturer, domains_value, description_value, web_site_value)`
    - `name_value`, le nom du modèle de drone
    - `manufacturer`, le nom de l'entreprise manufacturière (l'entreprise doit existé dans la table des entreprises)
    - `domains_value` : les noms des domaines opérationnels concernés par ce modèle de drone.
      - chaque drone peut avoir plusieurs domaines opérationnels
      - dans cet exemple, on propose de donner tous les domaines opérationnels dans une seule chaîne de caractères en utilisant le séparateur suivant : `'$'` (signe de dollar)
      - par exemple, on pourrait représenter un robot terrestre et aquatique ainsi :

        ```
        'Roulant&De surface'
        ```

      - pour plus de détail, voir la section Manipulation de tableau avec une requête SQL
    - `description_value`, une courte description du modèle
    - `web_site_value`, la page web décrivant le produit
  - `add_drone_specification(model_name, spec_name, spec_value, spec_comments)`
    - `model_name` : le nom du modèle (le modèle doit exister dans la base de données)
    - `spec_name` : le nom de la spécification technique (la spécification doit exister)
    - `spec_value` : la valeur de la spécification
    - `spec_comments` : un commentaire lié à cette spécification
- En utilisant les fonctions/procédures créées, vous devez faire le remplissage des données fournies. Vous avez à votre disposition trois fichiers où se trouvent les données minimales à insérer. Ces fichiers sont :
  - `drone_manufacturers.csv`
  - `drone_models.csv`
  - `drone_model_specifications.csv`

Votre script doit être nommé : `C42-P2_04_POPULATE_DRONE_MODELS.pgsql`

5. Localisations et employés

On vous demande de mettre en place certains outils et les données liées aux employés. Ce script est divisé en trois sections différentes :

- Vous devez créer plusieurs utilitaires simplifiant les insertions de données concernant les localisations, les employés et les embauches. Vous devez créer les fonctions/procédures répondant aux déclarations suivantes :
    - `get_localisation_tag(building_code CHAR(2), floor_level INTEGER, color_tag CHAR(3), room_number INTEGER) -> CHAR(14)`
        - `building_code`, le code de l'immeuble
        - `floor_level`, l'étage du local
        - `color_tag`, le code de couleur
        - `room_number`, le numéro du local
        - retourne le « *localisation tag* » générique
        - cette fonction/procédure doit valider tous les intrants – si l'un d'entre eux est invalide, la tâche est interrompue avec un message d'erreur pertinent
    - `get_office_localisation_tag(building_code CHAR(2), floor_level INTEGER, color_tag CHAR(3), room_number INTEGER, office_type CHAR(1), office_number INTEGER) -> ?`
        - `building_code`, le code de l'immeuble
        - `floor_level`, l'étage du local
        - `color_tag`, le code de couleur
        - `room_number`, le numéro du local
        - `office_type`, le type de bureau
        - `room_number`, le numéro du bureau
        - retourne le « *localisation tag* » pour les bureaux
        - cette fonction/procédure doit valider tous les intrants – si l'un d'entre eux est invalide, la tâche est interrompue avec un message d'erreur pertinent
    - `get_office_localisation_tag() -> ?`
        - cette surcharge retourne le « *localisation tag* » par défaut pour les bureaux
    - `get_storage_localisation_tag(building_code CHAR(2), floor_level INTEGER, color_tag CHAR(3), room_number INTEGER, office_type CHAR(1), storage_cabinet INTEGER, shelf_height INTEGER, storage_bin INTEGER) -> ?`
        - `building_code`, le code de l'immeuble
        - `floor_level`, l'étage du local
        - `color_tag`, le code de couleur
        - `room_number`, le numéro du local
        - `office_type`, le type de bureau
        - `storage_cabine`, le numéro d'étagère
        - `shelf_height`, la hauteur de la tablette

        - `storage_bin`, le numéro du casier de stockage
        - retourne le « *localisation tag* » pour l'entreposage
        - cette fonction/procédure doit valider tous les intrants – si l'un d'entre eux est invalide, la tâche est interrompue avec un message d'erreur pertinent
    - `get_ storage _localisation_tag() -> ?`
        - cette surcharge retourne le « *localisation tag* » par défaut pour les entrepôts
    - ```
      hire(ssn_value employee.ssn%TYPE,
           last_name_value employee.last_name%TYPE,
           first_name_value employee.first_name%TYPE,
           probation BOOLEAN DEFAULT TRUE,
           office_room_value employee.office_room%TYPE
                             DEFAULT get_office_localisation_tag())
      ```
        - `ssn_value`, le numéro social de l'employé
        - `last_name_value`, le nom de famille de l'employé
        - `first_name_value`, le prénom de l'employé
        - `probation`, si vrai, le statut de l'employé est `probation` sinon `regular`
        - `office_room_value`, le « *location tag* » de l'employé
        - ne retourne rien
        - cette fonction/procédure ajoute le nouvel employé
- Vous devez créer 3 fonctions/procédures utilitaires simulant les données de cette section répondant aux déclarations suivantes :
    - `simulate_office_localisation_tag() -> ?`
        - retourne un « *localisation tag* » de bureau valide
        - le « *localisation tag* » est généré aléatoirement
        - 20% du temps, la localisation par défaut est retournée
        - lorsqu'on génère un localisation aléatoirement, 85% du temps l'immeuble `GZ` est retenu
    - `simulate_storage_localisation_tag() -> ?`
        - retourne un « *localisation tag* » d'entrepôt valide
        - le « *localisation tag* » est généré aléatoirement
        - 40% du temps, la localisation par défaut est retournée
        - lorsqu'on génère un localisation aléatoirement, 75% du temps l'immeuble B`G` est retenu
    - ```
      simulate_hiring(last_name_value employee.last_name%TYPE,
                      first_name_value employee.first_name%TYPE,
                      ssn_value employee.ssn%TYPE DEFAULT NULL)
      ```
        - ne retourne rien
        - effectue l'insertion d'un nouvel employé dont la plupart des données sont générées aléatoirement
        - si le numéro social donné est nul
            - il est généré aléatoirement et est constitué de 9 caractères numériques
            - le premier ne peut être 0
            - il est important que le numéro social généré soit différent de ceux présents dans la base de données (il faut que l'insertion réussisse toujours)

        - 75% du temps, le nouvel employé a le statut `probation`
        - Utilise le numéro de bureau généré par la fonction de génération aléatoire
- Finalement, en utilisant `simulate_hiring`, vous devez faire l'ajout de nouveaux employés. Vous devez ajoutés, dans l'ordre, tous les nouveaux employés suivants :
    - *n* membres de l'équipe de travail (tous les étudiants)
        - chacun avec un numéro social bidon suivant ce patron `111111111`, `222222222`, `333333333`, `...`
    - *n* enseignants que vous avez eu pendant la technique
        - chacun avec un numéro social généré aléatoirement
    - où *n* représente le nombre d'étudiants dans l'équipe de travail

Votre script doit être nommé : `C42-P2_05_HIRING.pgsql`

6. Acquisition de drones

On vous demande de mettre en place certains outils et les données liées à l'acquisition de drones. Ce script est divisé en trois sections différentes :

- Vous devez créer plusieurs utilitaires simplifiant les insertions de données concernant la génération des `drone_tag` et l'acquisition de drones. Vous devez créer les fonctions/procédures répondant aux déclarations suivantes :
    - `generate_ drone_tag(`
      `model_name drone_model.name%TYPE,`
      `drone_acquisition drone.acquisition_date%TYPE)`
      `RETURNS drone.drone_tag%TYPE`
        - `model_name`, le modèle de drone acheté
        - `drone_acquisition`, la date d'acquisition du drone
        - retourne le `drone_tag` généré
        - cette fonction/procédure doit utiliser un outil de votre cru pour généré la partie séquentielle de la section `DDDDDD` du `drone_tag`
        - cette fonction/procédure doit valider le `model_name` et retourner un message d'erreur pertinent si le modèle n'existe pas
        - l'utilisation d'une séquence est à considérer pour cette partie
    - `add_drone_acquisition(`
      `model_name drone_model.name%TYPE,`
      `serial_drone drone.drone_tag%TYPE,`
      `registering_employee employee.ssn%TYPE,`
      `registering_timestamp drone_state.start_date_time%TYPE,`
      `receiving_employee employee.ssn%TYPE,`
      `receiving_date drone.acquisition_date%TYPE,`
      `unpacking_employee employee.ssn%TYPE)) -> ?`
        - `model_name`, le nom du modèle de drone
        - `serial_drone`, le numéro de série du fabriquant (pas le `drone_tag`)
        - `registering_employee`, l'employé faisant l'enregistrement dans le système
        - `registering_timestamp`, l'horodatage lié à l'enregistrement
        - `receiving_employee`, l'employé qui a fait la réception de la marchandise
        - `receiving_date`, le date de réception de la marchandise
        - `unpacking_employee`, l'employé qui a fait le déballage du drone
        - effectue les 3 tâches liées à l'acquisition d'un drone dans l'entreprise :
            - effectue l'insertion des informations dans la table `drone`
                - crée aussi automatiquement le `drone_tag`
                - utilise `receiving_date` comme date d'acquisition du drone
            - détermine le premier état du drone
                - réalise une insertion de données dans la table `drone_state`
                - utilise les données `registering_employee` pour l'employé et `registering_timestamp` pour l'horodatage lié à l'état
            - insère une note essentielle pour ce premier état (suivant les directives du client) :
                - la note est de type observation générale

        - utilise les données `registering_employee` pour l'employé et `registering_timestamp` pour l'horodatage lié à l'état
        - le détail est la chaîne de caractères suivante (sur 2 lignes) :
          ```
          Received by : Jean-Marc Deschamps on 2021-05-29
          Unpacked by : Marcel L'Italien
          ```
- Vous devez créer 3 fonctions/procédures utilitaires simulant les données de cette section répondant aux déclarations suivantes :
    - ```
      simulate_drone_acquisition(
          model_name drone_model.name%TYPE,
          registering_employee employee.ssn%TYPE,
          registering_timestamp drone_state.start_date_time%TYPE)
                                                                  -> ?
      ```
        - effectue l'insertion d'un nouveau drone (acquisition simulée)
        - le numéro de série du drone est généré aléatoirement suivant ce patron :
            - doit débuter par un caractère allant de `A` à `J`
            - une suite de caractères allant de `A` à `J`, de `0` à `9` ou avec les signes de ponctuations suivant `.` (point), `-` (tiret) et `:` (deux-points)
            - doit terminer par caractères numériques
            - au total, le numéro de série doit être une chaîne de caractères variant de 6 à 12 caractères
            - ATTENTION, il faut que le numéro de série soit unique dans la BD. S'assurer que le numéro généré soit bien unique.
        - tous les employés concernés sont le même, le `registering_employee` donné
        - la date de réception est 1 heure avant l'horodatage donné
    - ```
      simulate_drone_acquisition(
              ref_timestamp drone_state.start_date_time%TYPE) -> ?
      ```
        - effectue l'insertion d'un nouveau drone (acquisition simulée)
        - les informations suivantes sont déterminées aléatoirement parmi ceux disponibles dans la base de données :
            - le `drone_model`
            - le `registreing_employee`
            - le `receiving_employee`
        - à 75% du temps, le `unpacking_employee` est le même que le `registering_employee` sinon il est le même que le `receiving_employee`
        - le `registering_timestamp` est le `ref_timestamp`
        - toutefois, le `receiving_date` est une journée avant le `ref_timestamp`
        - le numéro de série est généré aléatoirement selon la même règle que la fonction précédente
    - ```
      simulate_drone_acquisition(n INTEGER,
            from_timestamp TIMESTAMP, to_timestamp TIMESTAMP) -> ?
      ```
        - appelle la surcharge précédente *n* fois en répartissant les horodatages dans l'intervalle donnés
        - la valeur n doit être strictement positive sinon un message d'erreur pertinent est affiché
      ```

- Finalement, vous devez faire l'acquisition de 100 drones en faisant au moins un appelle à :
    - `add_drone_acquisition`
    - les trois surcharges de : `simulate_drone_acquisition`

Votre script doit être nommé : `C42-P2_06_DRONE_ACQUISITION.pgsql`

7. Gestion des états

Cette partie devient le point culminant du projet. On vous demande de mettre en place les outils d'automatisation de la gestion des états de drone.

Contrairement aux sections précédentes où vous étiez fortement guidé, la conception des éléments de cette partie vous appartient. Toutefois, vous avez objectifs très spécifiques :

- Vous devez créer un déclencheur interdisant les opérations de modification et de suppression sur la table `drone_state`.
- Vous devez créer un déclencheur validant si une opération d'insertion dans la table drone_state est valide. Ce déclencheur doit valider tous les éléments suivants :
    - Si le nouvel état est possible selon l'état actuel et en respectant le diagramme des transitions.
    - Si le nouvel horodatage est supérieur à l'actuel.
    - S'il existe les notes adéquates selon le type de transition :
        - Pour les transitions rejetées :
            - une note `problematic_observation`
        - Pour les transitions acceptées de type `R` :
            - une note `problematic_observation`
            - une note `maintenance_performed` ou `repair_completed`
- Une fonction facilitant le passage d'un état à un autre. La fonction doit considérer les éléments suivants :
    - le `drone_tag` et/ou le `id` du drone
    - l'employé et la localisation
    - un booléen indiquant si l'état est accepté ou refusé

  Cette fonction insère l'horodatage courant.
- Une fonction facilitant l'insertion de note pour l'état courant.
- Une fonction de simulation simulant une transition pour un drone. Pour cette fonction on considère ces informations :
    - le `drone_tag` et/ou le `id` du drone
    - l'horodatage désiré
    - en pourcentage entre 0 et 1, indiquant la possibilité de transition acceptée.
- Une fonction de simulation simulant *n* transitions pour un drone. Pour cette fonction, on considère ces informations :
    - le nombre d'insertions à faire
    - le `drone_tag` et/ou le `id` du drone
    - deux horodatages indiquant l'intervalle de temps à remplir (chaque insertion distribue le temps dans l'intervalle donné)
    - en pourcentage entre 0 et 1, indiquant la possibilité de transition acceptée.
- Une autre fonction pratiquement identique à la précédente qui détermine aléatoirement le drone concerné. Les mêmes paramètres sont utilisés sauf l'identification du drone.

Attention, vos différentes fonctions doivent tenir compte du fait qu'un drone perdu ou hors-service ne peut changer d'état. Si cette situation arrive, un message doit informer l'utilisateur.

Finalement, effectuer plusieurs changements d'états (plusieurs centaines au total).

8. Requêtes

On vous demande de créer un script nommé `C42-P2_08_DQL.pgsql` ayant les requêtes répondant aux questions suivantes :

1. Pour chacun des drones que possède l'entreprise, afficher :
   - nom du modèle
   - `drone_tag`
   - numéro de série
   - son état (on désire le nom complet et pas seulement le symbole)
   - la date à partir de laquelle cet état a été effectif

   On désire le tout trié en ordre décroissant de :
   - le nom du modèle (croissant)
   - date d'acquisition des drones (décroissant)
   - `drone_tag` (croissant)
2. On désire connaître quels sont les drones disponibles pour la location. Cette requête doit présenter :
   - le domaine opérationnel (s'il y en a plusieurs, une chaîne de caractères où chacun est séparé par une virgule)
   - nom du modèle
   - `drone_tag`
   - la date depuis quand il est disponible
   - sa localisation.

   On désire le résultat trié par :
   - le domaine opérationnel (croissant)
   - le nom du modèle (croissant)
   - la date depuis quand il est disponible (croissant) Donner le nombre d'inspections que chaque employé a fait.
3. Pour un drone donné, montrer l'historique des activités liées. Pour chaque activité on désire :
   - l'état (avec le nom au complet)
   - la date
   - le nom et prénom de l'employé lié (prénom + espace + nom)
   - le nombre de notes disponibles
4. Dans le but de faire une promotion pour la location à rabais, on désire connaître quels sont les drones qui sont disponibles pour la location depuis longtemps. Plus précisément, on désire la liste des drones qui est disponible à la location depuis `T`. Le rabais en pourcentage *R* est établi comme suit :

   $$R = 0.005\,x^2 + 0.05\,x + 5$$

   R doit être arrondi au 2.5% le plus près et plafonné à 50%

   où :
   - `T` = la moitié de la moyenne des délais requis pour faire une location
   - *x* = le temps écoulé depuis `T` (en jours sans fractions)
   -

Cette requête doit présenter :
    - le modèle du drone
    - le `drone_tag`
    - *x*
    - *R*
5. Pour chaque employé, on désire connaitre le nombre de fois qu'il.elle a accepté.e positivement et négativement la transition d'un état à un autre (on considère toutes les transitions). On désire cette présentation :
    - prénom
    - nom de famille
    - le nombre de transitions acceptées
    - le nombre de transitions rejetées
    - le ratio de transitions rejetées

    où le ratio correspond au pourcentage des rejets des transitions qu'il.elle a effectuées
    le tout trié en nombre total de transitions effectuées (décroissant)
6. On désire connaître pour chaque modèle de drone :
    - le nombre d'unités toujours en opération
    - le nombre d'unités hors service
    - le nombre d'unités perdu
    - le nombre de locations totales parmi toutes les unités
    - le nombre de réparations totales parmi toutes les unités
7. Ajoutez deux requêtes de votre crue qui soit pertinente.

Soyez conscient que certaines requêtes peuvent bénéficier de clarification. Surtout, n'hésitez pas à poser des questions.

Pour chacune des requêtes du script, vous devez respecter la structure suivante en y ajoutant les informations adéquates :

```
-- ========================================================
-- Requête #
-- Objectif :    ...
--               ...
--               ...
-- Évaluation :  ...
--               ...
--               ...
-- Réalisé par : ...
-- Aidé par    : ...
-- ========================================================
votre requête
-- ========================================================
```

Vous devez donc indiquer :

- le numéro de la requête créée;
- l'objectif : ce que fait la requête;
- votre évaluation du résultat obtenu : certaines requêtes sont complexes et il est possible que vous ayez de la difficulté à la réaliser; dans ce cas, mettez ce que vous avez réussi à faire et expliquez ce qui fonctionne et ce qui ne fonctionne pas – soyez concis et précis.

ATTENTION : les informations demandées doivent être complétées adéquatement sinon vous aurez automatiquement zéro.

Notions complémentaires

Tables de référence et tables de référence immuables

Une table de référence est une table dont le contenu est connu à la création et qui ne change pratiquement jamais pendant le projet. En fait, il peut arriver que le contenu d'une table change, mais ce sont des évènements plus rares ou exceptionnels. Par exemple représentant bien ce concept est une table contenant les noms des pays.

On retrouve souvent la notion où le contenu d'une table peut changer un peu, mais n'a pas d'impact au niveau des opérations. Par exemple, on peut décider d'avoir une liste de choix prédéfinis, mais qu'il soit possible d'avoir des ajouts ultérieurs. En général, on autorise les opérations du DML pour l'insertion et la mise à jour, mais on interdit les suppressions. Cette approche garantit que jamais il n'y aura de problème avec des valeurs existantes qui peuvent être référées dans le système. Par exemple, une table de couleurs pourrait contenir 100 couleurs prédéfinies par l'entreprise. Il n'y a pas de mal à en ajouter quelques autres, mais ce sont des situations plus rares. Par exemple, l'entreprise décide, une fois par année, d'ajouter trois nouvelles couleurs tendance associées à son image.

Il existe aussi la situation où le contenu d'une table ne doit jamais changer. Ces tables de références sont considérées comme étant immuables. Selon le contexte applicatif, leurs modifications pourraient engendrer des situations problématiques soit dans les logiques opérationnelles ou simplement dans la logique d'affaires. Ces tables sont appelées tables de référence immuables et toutes les opérations du DML sont interdites.

Plusieurs approches sont envisageables pour réaliser une telle restriction (gestion des droits usager, police d'accès aux lignes …). Toutefois, pour ce projet, on vous demande d'utiliser spécifiquement des déclencheurs pour répondre au problème.

Dans ce projet, il existe :

- deux tables de référence
  - `unit` : représente les unités de mesure (n'a aucune dépendance)
  - `technical_specification` : représente une spécification technique précise du domaine du génie
- deux tables de référence immuables :
  - `state` : représente l'infrastructure des états et de leur relation selon la pratique de l'entreprise
  - `operational_domain` : représente l'arbre des domaines opérationnels existants

Vous devez créer une seule fonction déclencheur (« *trigger function* ») nommée `forbid_dml_operations` interrompant les opérations en cours et affichant spécifiquement ce message d'erreur :

```
Opération interdite [_ OPERATION_NAME_] sur la table _TABLE_NAME_.
```

Où :

- `_OPERATION_NAME_` est le nom en majuscule de l'opération du `DML` concernée par l'évènement
- `_TABLE_NAME_` est le nom en majuscule de la table concernée par l'évènement
- les crochets `[...]` font partie de la chaîne de caractères.

Ensuite, pour chacune des quatre tables mentionnées, vous devez créer un trigger pertinent (quatre au total).

Notes techniques

Surcharge de fonctions

`PostgreSQL` supporte la notion de surcharge de fonction. Cet outil technique est non seulement pratique, mais aussi pertinent. La surcharge offre une meilleure uniformité pour l'interface de programmation disponible pour les équipes de développement. Toutefois, une mise en garde importante est à faire dans le cas d'écriture de scripts créant des fonctions.

L'exemple suivant présente une mise en situation classique créant des problèmes parfois difficiles à déboguer. Vous voulez créer une fonction `FUNC` et votre conception préliminaire implique que la fonction ait un seul argument. Vous créez cette fonction avec la clause `CREATE OR REPLACE FUNCTION …` Génial! Par la suite, vous modifiez votre conception pour la fonction prenne un paramètre de type différent ou plusieurs autres paramètres. Vous changez la signature et appelez de nouveau `CREATE OR REPLACE FUNCTION …` Maintenant, pensant avoir remplacé la première version de la fonction, vous avez créé une deuxième version avec surcharge. Imaginez faire cette opération plusieurs fois.

Par la suite, lorsque vous appelez votre fonction et sans que vous ne vous en rendiez compte, l'appel se fait avec une version antérieure qui ne change pas et vous ne comprenez pas pourquoi vous avez toujours une certaine erreur.

Soyez prudent et assurez-vous de bien supprimer vos fonctions obsolètes.

Fonctions utilitaires de PostgreSQL

Voici quelques références vers des fonctions existantes pour des tâches spécifiques.

- Remplissage (« *padding* ») sur une chaîne de caractères :
  - voir les fonctions `lpad` et `rpad`
- Nombre de jours entre deux dates :
  - voir la soustraction entre les dates
- Extraction et substitution de caractères dans une chaîne de caractères :
  - extraction d'une sous-chaîne de caractères, voir la fonction `substring`
  - substitutions spécifique de caractères, voir la fonction `translate`
- Conversion de caractères `ascii` à un nombre entier et vice versa :
  - le code `ascii` (l'entier) d'un caractère, voir la fonction `ascii`
  - le caractère d'un code `ascii`, voir la fonction `chr`

Fonctions utilitaires mises à votre disposition

Vous avez plusieurs fonctions utilitaires qui pourront vous aider pour :

- mieux comprendre les procédures et fonctions en SQL et PL/pgSQL (via le code source)
- vous aider à réaliser plusieurs tâches du projet

Voici un résumé des fonctions disponibles et des fichiers où elles se trouvent.

- `C42-P2_00 MATH_UTILITIES.pgsql`
  - `clamp` (plusieurs surcharges)
- `C42-P2_00 RANDOM_UTILITIES.pgsql`
  - ATTENTION, ce fichier dépend de `MATH_UTILITIES` pour la fonction `clamp`. Donc, exécuter le script dépendant avant celui-ci
  - `random_event`
  - `random_integer`
  - `random_real`
  - `random_char`
  - `random_string`
  - `random_time`
  - `random_date`
  - `random_timestamp` (avec 2 surcharges)

Vous pouvez considérer ces scripts comme des outils génériques indépendants de ce projet. Ainsi, vous devez les exécuter avant les vôtres pour pouvoir les utiliser.

Manipulation de tableau avec une requête SQL

Un sujet important qui n'a pas été couvert pendant la session est la possibilité d'utiliser certains autres types, notamment les tableaux. C'est un vaste sujet qui mériterait quelques périodes de cours si nous avions plus de temps.

Néanmoins, pour une partie du projet, on vous demande d'explorer un peu cette notion pour régler un cas précis. Supposons que nous désirons diviser une chaîne de caractères en sous-chaîne de caractères selon un certain séparateur. Par exemple, dans la chaîne de caractères suivante, on utilise le caractère `;` pour séparer les noms des enseignants.

```
Jean-Marc;Marcel;Frédéric;Éric;Gedefroy
```

La fonction `string_to_array` permet de diviser la chaîne de caractères initiale en sous-chaîne de caractères avec le séparateur désiré. Par exemple, l'instruction suivante:

```
SELECT string_to_array('Jean-Marc;Marcel;Frédéric;Éric;Gedefroy', ';') AS tableau;
```

donnera le résultat suivant :

	tableau text[]
1	{Jean-Marc,Marcel,Frédéric,Éric,Gedefroy}

Il est important de remarquer le type `text[]`. Les crochets `[...]` indiquent que ce n'est pas une colonne de type `text` mais plutôt une colonne de type tableau de texte (`array of texts`). De plus, la valeur présente bien les éléments du tableau avec les accolades et les éléments séparés par une virgule `{ ..., ..., ... }`. Donc, le résultat est un tableau de cinq valeurs.

NOTE IMPORTANTE : Attention de ne pas utiliser ce concept inadéquatement et briser l'un des concepts fondamentaux du modèle relationnel : l'atomicité des valeurs. Cet outil est très puissant et offre des possibilités très intéressantes. Mais comme tous les outils en informatique, il est important de savoir quand les utiliser.

Il est possible d'extraire les valeurs de ce tableau pour les mettre dans une table conventionnelle. C'est la fonction `unnest` qui permet cette opération. Par la suite, on peut utiliser tous les outils `SQL` et `PL/pgSQL` pour réaliser les tâches requises.

Voici le résultat obtenu de la requête utilisant cette stratégie :

```
SELECT unnest(string_to_array('Jean-Marc;Marcel;Frédéric;Éric;Gedefroy', ';')) AS Enseignant;
```

On remarque cette fois-ci que le type de la colonne est `text` tel qu'attendu et que la règle d'atomicité est respectée.

	enseignant text
1	Jean-Marc
2	Marcel
3	Frédéric
4	Éric
5	Gedefroy

Exécution de scripts avec `psql`

`pgAdmin` offre une interface utilisateur graphique intuitive et facile à utiliser. Toutefois, une interface en ligne de commande est aussi offerte.

`psql` est l'outil de ligne de commande de `PostgreSQL`. Il permet aux utilisateurs :

- d'exécuter des requêtes `SQL`
- d'exécuter des scripts
- d'administrer les bases de données
- d'utiliser des fonctionnalités avancées comme les « import/export » de données
- d'établir une connexion à des bases de données distantes

`psql` est apprécié pour sa flexibilité, sa puissance, son efficacité et sa faible empreinte au niveau des ressources utilisées. Toutefois, comme tout outil en ligne de commande, beaucoup plus de temps est nécessaire à son apprentissage.

Évidemment, le cours est suffisamment chargé comme ça pour que nous ne l'ayons pas présenté. Toutefois, pour ceux qui sont intéressés, c'est un outil de gestion très pratique pour exécuter plusieurs scripts à la fois. Faire ce type d'exécution en *batch* est l'une des tâches les plus fréquentes et vraiment plus agréable à faire que d'exécuter manuellement les scripts les uns à la suite des autres avec `pgAdmin`.

Pour y arriver, il suffit de créer un script supplémentaire qui exécutera les scripts désirés. Ensuite, il suffit d'exécuter ce nouveau script uniquement avec `psql`.

Voici les étapes:

1. préparation :
    a. créer les scripts du projet
    b. créer un script supplémentaire qui exécutera les scripts originaux, pour ce projet vous pouvez le nommer `C42-P2-main.pgsql`
    `c.` vous trouverez une partie du contenu de ce script plus bas
2. exécution
    a. démarrer `psql`
        i. vous pouvez le démarrer directement dans une console (voir la documentation)
        ii. vous pouvez le démarrer à même `pgAdmin`, en cliquant sur `>_`
    b. exécuter manuellement quelques instructions pour déterminer l'encodage et le dossier de travail courant (le `CWD`) :

        ```
        \! chcp 65001
        SET client_encoding = 'UTF8';
        \cd /en/absolu/le/nom/du/dossier/CWD/
        ```

        S'il y a des espaces, mettre le chemin entre apostrophe pour indiquer une chaîne de caractères
    c. exécuter le script créé en 1b avec la commande suivante :

        ```
        \i le_nom_du_fichier_script.pgsql
        ```

        Par exemple :

        ```
        \i C42-P2-main.pgsql
        ```

Le petit exemple de code suivant montre à quoi pourrait ressembler un tel script

```
--
-- 1. Lancer psql
-- 2. Exécuter manuellement ces commandes une seule fois au démarrage de psql
--    a. \! chcp 65001
--    b. SET client_encoding = 'UTF8';
--    c. \cd /en/absolu/le/nom/du/dossier/CWD/current/working/directory/
-- 3. Exécuter ce script :
--       \i le_nom_de_ce_fichier_script.pgsql
--
--
\! cls -- ne vide pas tout à fait la console
       -- mais déplace le curseur de la ligne courante tout en haut

\echo
\echo
\echo
\echo
\echo
\echo ------------------------------------------------------------------------------
\echo ------------------------------------------------------------------------------
\echo ------------------------------------------------------------------------------
\echo ------------------------------------------------------------------------------
\echo ------------------------------------------------------------------------------
\echo -- START C42 SCRIPTS
\echo ------------------------------------------------------------------------------
\echo ------------------------------------------------------------------------------
\echo ------------------------------------------------------------------------------
\echo ------------------------------------------------------------------------------
\echo ------------------------------------------------------------------------------
\! chcp
SHOW client_encoding;
\echo
\echo
\echo
\echo
\echo

\echo PHASE 0 - Insertion des scripts utilitaires
\echo ------------------------------------------------------------------------------
\i 'C42-P2_00 MATH_UTILITIES.pgsql'
\i 'C42-P2_00 RANDOM_UTILITIES.pgsql'
\echo
\echo

\echo PHASE 1 - Crée l''infrastructure nominale
\echo ------------------------------------------------------------------------------
\i 'C42-P2_01_CREATE_INFRA.pgsql'
\echo
\echo

... ... ...
```

Contraintes

Plusieurs contraintes sont à respecter pour ce travail :

- Vous devez travailler en équipe de 5 ou de 6 étudiants.
- Il est très important que tous les membres de l'équipe travaillent équitablement, car l'évaluation finale tient compte de plusieurs critères, dont la répartition de la tâche de travail.
- Vous devez travailler avec le SGBDR PostgreSQL utilisé tout au long de la session.

Stratégie d'évaluation

L'évaluation se fera en 2 parties. D'abord, l'enseignant évaluera le projet remis et assignera une note de groupe pour le travail. Ensuite, chaque équipe devra remettre un fichier Excel dans lequel sera soigneusement reportée une cote représentant la participation de chaque étudiant dans la réalisation du projet. Cette évaluation est faite en équipe et un consensus doit être trouvé.

Une pondération appliquée sur ces deux évaluations permettra d'assigner les notes finales individuelles.

Ce projet est long et difficile. Il est conçu pour être réalisé en équipe. L'objectif est que chacun prenne sa place et que chacun laisse de la place aux autres.

Ainsi, trois critères sont évalués :

- **participation** (présence en classe, participation active, laisse participer les autres, pas toujours en train d'être sur Facebook ou sur son téléphone, concentré sur le projet, pas en train de faire des travaux pour d'autres cours, …)
- **réalisation** (répartition du travail réalisé : conception, modélisation, rédaction de script, documentation, …)
- **impact** (débrouillardise, initiative, amène des solutions pertinentes, motivation d'équipe, …)

Remise

Vous devez créer un fichier de format `zip` dans lequel vous insérez :

- tous les scripts demandés
- optionnellement, vos impressions et commentaires sur ce travail `commentaires.txt`
- le fichier Excel rempli sur la participation active des membres du groupe `evaluation.xlsx`

Vous devez remettre votre projet une seule fois sur Lea après avoir nommé votre fichier :

NomPrenomEtudiant1_NomPrenomEtudiant2[_NomEtudiantN].zip