La collecte digitalisée avec CSPro

Version 7.3

International Programs
Population Division
U.S. Census Bureau
4600 Silver Hill Road
Washington, DC 20233
cspro@lists.census.gov

Session 01 : Vue d’ensemble de CSPro, dictionnaire et formulaires

À la fin de cette leçon, les participants seront en mesure de :

- Identifier les différents modules et outils CSPro et leurs rôles dans le déroulement de travail de l’enquête
- Créer une application de saisie de données simple comprenant un dictionnaire et des formulaires
- Lancer une application de saisie de données sous Windows
- Lancer une application de saisie de données sur un appareil mobile et récupérez les données saisies
- Afficher les données saisies à l'aide de DataViewer

Vue d’ensemble de CSPro

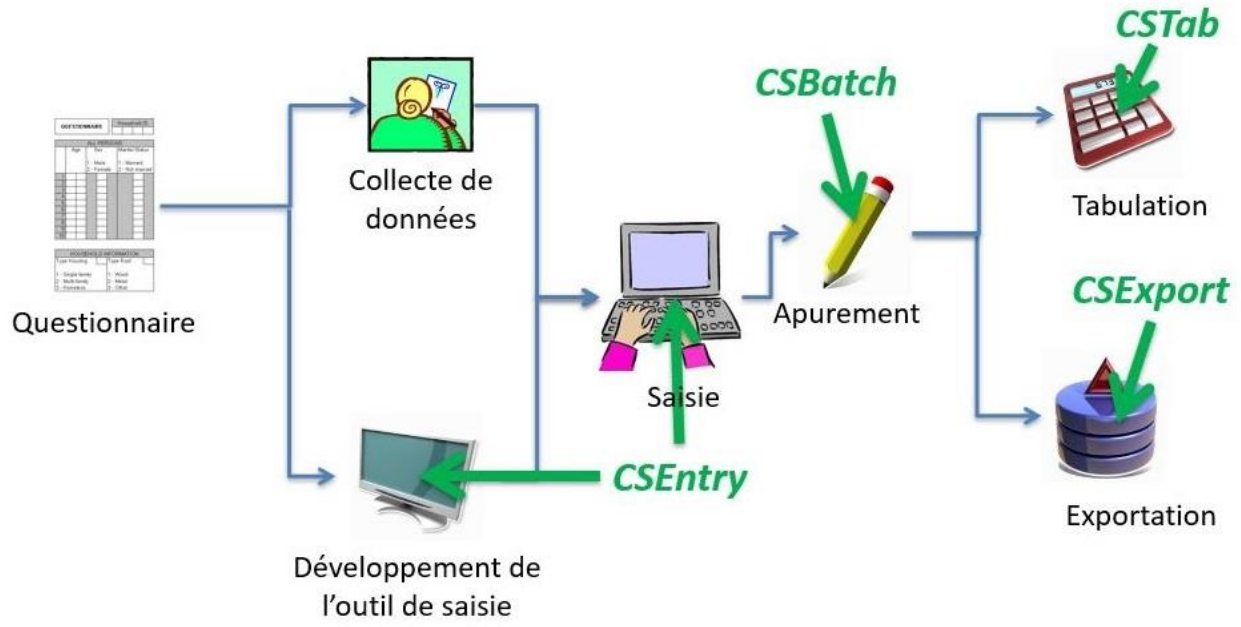
CSPro est une suite de logiciels pour le traitement des données des recensements et des enquêtes qui comprend des modules de collecte, d'apurement, de tabulation et de diffusion de données.

CSPro a une longue histoire. Il a été publié en 2000 et depuis ce temps a été utilisé dans plus de 100 pays à travers le monde. Il a été utilisé pour des recensements partout dans le monde ainsi que pour de nombreuses enquêtes auprès des ménages, notamment l'Enquête démographique et de santé (USAID), l'Enquête en grappes à indicateurs multiples (UNICEF) et l'Étude sur la mesure des niveaux de vie (Banque mondiale). La première version Android a été publiée en 2014. CSPro Android a déjà été utilisé dans la production d'enquêtes auprès des ménages et de recensements de la population dans plusieurs pays.

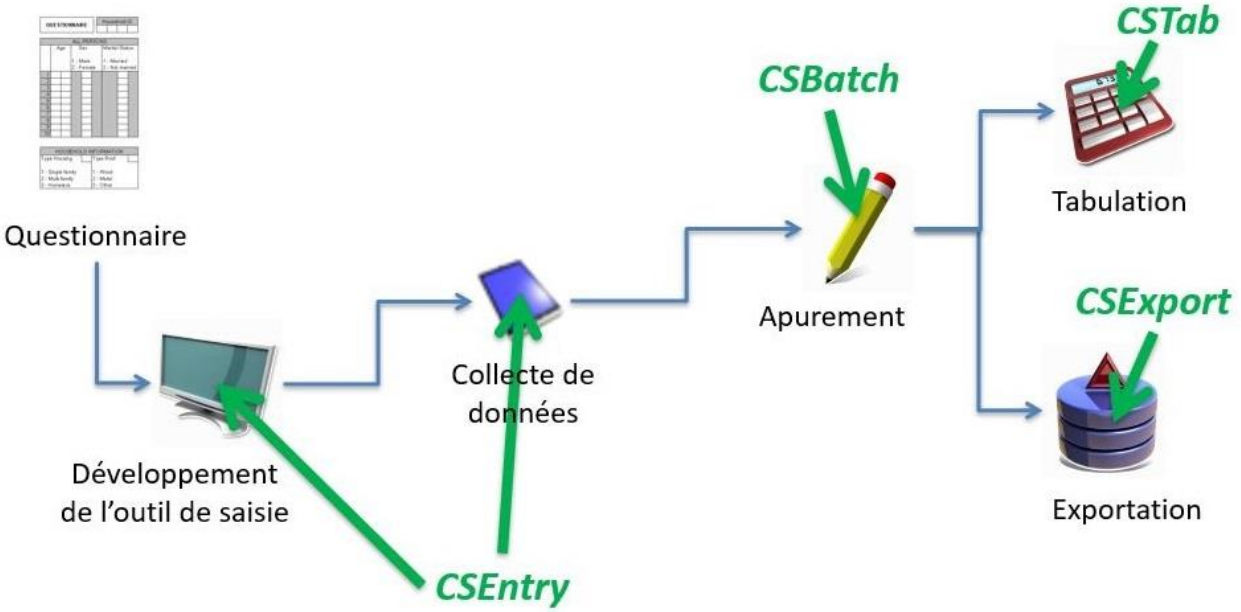
CSPro est un logiciel gratuit développé par le US Census Bureau et financé par l'USAID. Le US Census Bureau fournit un support client par courrier électronique gratuit. Vous pouvez envoyer des questions à cspro@lists.census.gov. Il existe également un forum d'utilisateurs où vous pouvez poser des questions et consulter les réponses aux questions posées par d'autres utilisateurs à l'adresse https://www.csprousers.org/forum/.

CSPro peut être utilisé à la fois pour la collecte traditionnel “PAPI” (paper and pencil interview) ainsi que pour le la collecte numérique “CAPI” (computer aided personal interview). Dans cet atelier, nous nous concentrerons sur la collecte de données sur les appareils mobiles avec la méthode “CAPI”.

Processus Traditionnelle



Processus Numérique



Matériel et logiciel requis

Pour exécuter les exemples de cet atelier, vous aurez besoin des éléments suivants :

- PC Windows avec CSPro version 7.3 ou ultérieure,
- Tablette Android avec CSEntry version 7.3 ou ultérieure.

Utilisation de CSPro Android

Avant d'apprendre à créer des applications de saisie de données dans CSPro, essayons de faire une petite collecte de données pour se familiariser avec le système.

Travail de groupe

Répartissez-vous en groupes de 3 à 4 personnes et utilisez les tablettes fournies pour vous interroger à l'aide de l'application «Apprendre à vous connaître». Interviewez chaque membre du groupe afin que nous ayons des données pour tous les participants de l'atelier. Lorsque vous avez terminé, appuyez sur le bouton de synchronisation () pour télécharger vos résultats sur le serveur.

Création d'une application de saisie de données

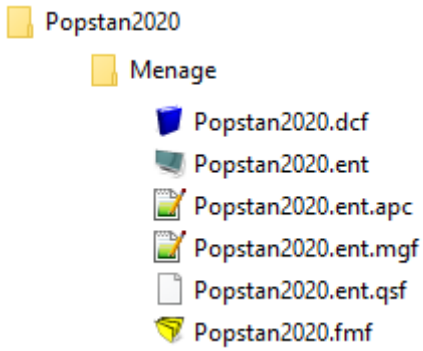
Pour la plupart des exemples de cet atelier, nous allons créer une application de saisie de données basée sur le questionnaire du recensement général de la population et de l'habitat de Popstan 2020 inclus dans le matériel de l'atelier. Avec les applications CAPI, un questionnaire papier ne suffit pas pour définir d'une manière compréhensive l'application de saisie de données. Nous avons également besoin d'un document de spécification décrivant les contrôles de cohérence, les sauts, les remplissages de texte, les messages d'erreur et d'autres aspects de l'application interactive qui ne sont pas définis sur un questionnaire papier. Prenez un moment pour examiner à la fois le questionnaire et le document de spécification.

Bien que les programmes de saisie de données CSPro puissent être exécutés sur des tablettes et des téléphones Android et Windows, pour développer une application d'enquête ou de recensement dans CSPro, vous utiliserez CSPro Designer, qui est exécuté sur des PC Windows. Lorsque vous lancez CSPro Designer, vous avez le choix entre "Data entry application" (Application de saisie de données) pour la saisie papier et "CAPI Data Entry Application" (Application de saisie de données CAPI) pour la collecte de données numériques à l'aide de téléphones / tablettes / ordinateurs portables. Les différences sont les suivantes :

- System controlled (chemin étroitement contrôlé)
- Texte des questions CAPI
- Extended Controls (boutons radios, cases à cocher, sélecteur de date…)

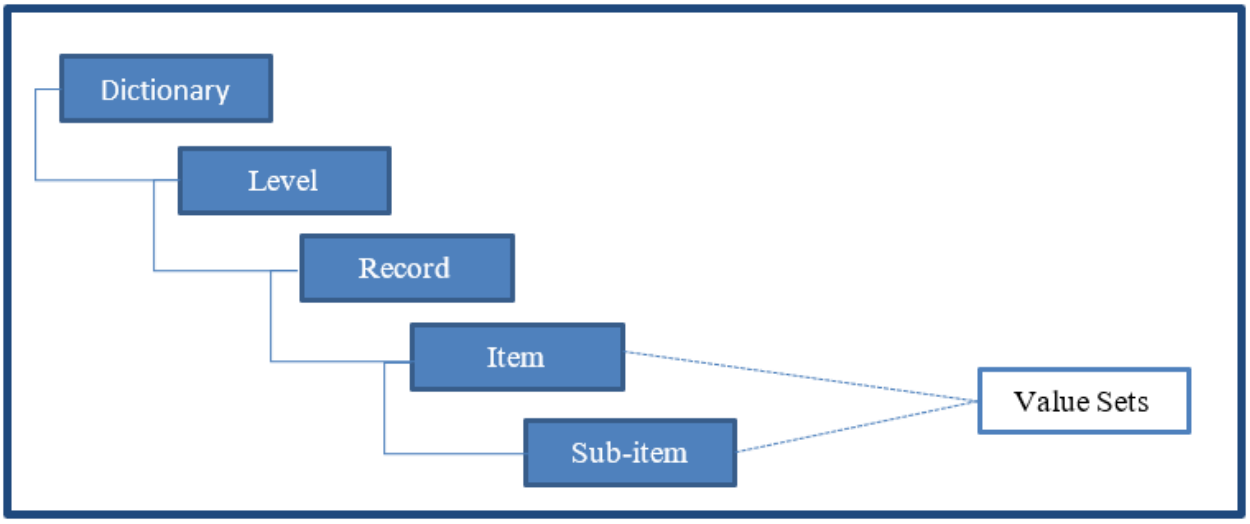
Étant donné que nous créons une application CAPI, nous choisirons "CAPI Data Entry Application". Nous nommerons l'application "Popstan2020" et nous utiliserons le même nom pour le dictionnaire. Étant donné que nous ajouterons éventuellement d'autres applications

telles que la fiche de numérotation des ménages, et l'application menu, nous allons créer la structure de dossiers suivante :



Dictionnaire de données

La première étape de la création de l'application consiste à définir le dictionnaire de données. Le dictionnaire de données regroupe toutes les variables et les réponses possibles de l'application et les organise dans les enregistrements et les niveaux. La hiérarchie du dictionnaire est la suivante :



Avant de définir les enregistrements et les variables, nous créons d'abord les id-items (les identifiants). Les identifiants définissent chaque cas (chaque questionnaire) de manière unique. C'est-à-dire que deux cas différents ne peuvent jamais partager les mêmes valeurs pour les variables identifiantes. Habituellement les identifiants sont des codes géographiques.

Quels sont les identifiants dans notre questionnaire ?

Province, district, zone du dénombrement, et numéro du ménage.

Pourquoi ne pas ajouter aussi le milieu de résidence, les coordonnées GPS, la date de l'entretien, l'heure de début et de fin car ils figurent dans la même section du questionnaire?

Parce qu'ils ne font pas partie des codes nécessaires pour identifier le questionnaire de manière unique.

Ajouter les identifiants au dictionnaire :

- Province (numérique, longueur 1)
- District (numérique, longueur 2)
- Zone du dénombrement (numérique, longueur 3)
- Numéro du ménage (numérique, longueur 3)

Propriétés des variables du dictionnaire :

- **Label :** texte affiché aux intervieweurs (pas le texte complet de la question, nous verrons où le mettre dans une leçon ultérieure)
- **Name :** utilisé pour désigner une variable dans la logique CSPro (les intervieweurs ne le verront jamais)
- **Start :** numéro de colonne du premier chiffre (caractère) de la variable
- **Len :** nombre de caractères / chiffres utilisés pour stocker la variable
- **Data type :** alpha pour les noms et autres textes, numérique pour les nombres et les réponses codées
- **Item type :** si la variable est un item (variable) ou un subitem (sous-variable)
- **Occ :** nombre d'occurrences pour les variables qui sont répétés dans l'enregistrement
- **Dec :** pour les nombres avec des fractions, nombre de chiffres après la virgule décimale
- **Dec char :** inclure ou non la virgule décimale dans les nombres décimaux
- **Zero fill :** pour les valeurs numériques ; s'il faut ajouter des zéros ou des blancs à la gauche du nombre lorsque le nombre de chiffres dans la valeur est inférieur à la longueur de la variable. En utilisant zero fill, on évite des problèmes avec les subitems. Nous pouvons activer le zero fill par défaut dans le menu des options.

Conseil : Notez que vous pouvez afficher les noms ou les libellés dans l’arborescence du dictionnaire, dans la partie gauche de l’écran, à l’aide du menu “View”. Vous pouvez également sélectionner "Append labels to names in tree” pour afficher les deux en même temps.

Quels sont les enregistrements utilisés dans notre enquête ?

Type d'enregistrement	Nom d'enregistrement	Section (s) du questionnaire
A	INTERVIEW_REC	A. Variables non identifiants (A6-A10)
B	INDIVIDUS_REC	B. Caractéristiques individuelles, C. Éducation, D. Fécondité
E	DECES_REC	E. Décès
F	HABITAT_REC	F. Caractéristiques de l'habitation
G	BIENS_REC	G. Biens du ménage

Notez que nous pourrions avoir des enregistrements séparés pour l'éducation et la fécondité, mais on a choisi de les combiner avec l'enregistrement individu. Cela simplifiera l'analyse ultérieurement car nous n'aurons pas à lier les différents enregistrements. Nous verrons plus tard que, même avec les enregistrements combinés, nous pouvons toujours mettre l'éducation et la fécondité dans des grilles séparées sur nos formulaires.

Commençons par créer les enregistrements de l'habitat et celui de l'individu.

Les propriétés des enregistrements :

- Label / name : comme pour les variables.
- Type Value : pour distinguer les différents enregistrements dans le fichier de données.
- Required : si l'enregistrement doit être saisi pour que le questionnaire soit au complet.
- Max : pour les enregistrements qui sont répétés, le nombre maximum d'occurrences autorisées. Généralement 1 pour les enregistrements uniques (comme l'enregistrement habitat) et un nombre plus important pour les enregistrements répétitifs (comme 30 pour les membres du ménage). Notez que CSPro n'alloue pas d'espace dans le fichier de données pour les occurrences qui ne sont pas utilisées. Il est donc préférable de faire preuve de prudence et d'autoriser des occurrences supplémentaires.

Quelles sont les propriétés de l'enregistrement habitat ?

- Type value : F (vous pouvez utiliser n'importe quoi, mais il vaut mieux utiliser quelque chose de significatif comme la lettre de la section)
- Required : non (nous ne collecterons pas la section F pour les logements vacants / refusés)
- Max : 1

Pour l'enregistrement de l'individu ?

- Type value : B
- Required : non (nous pouvons avoir des ménages vides)
- Max : 30 (le questionnaire a une limite de 10, mais aucune pénalité n'est ajoutée pour ajouter quelques cas supplémentaires au cas où)

Ajoutez maintenant quelques champs à l'enregistrement individu :

- Numéro d'ordre (longueur numérique 2)[1]
- Prénoms et Nom (longueur alpha 30)
- Lien de parenté (longueur numérique 1)
- Sexe (longueur numérique 1)
- Âge (longueur numérique 3)

Et à l'enregistrement de l'habitat :

- Nombre de pièces (longueur numérique 2)
- Type de logement principal (longueur numérique 1)

Les noms des variables

Chacun a son style préféré pour la dénomination des variables. Certains utilisent un nom descriptif tel que "LIEU_DE_NAISSANCE", d'autres préfèrent utiliser le numéro de la question tel que "B07", d'autres préfèrent une combinaison telle que "B07_LIEU_DE_NAISSANCE". Quelle que soit l'approche choisie, veillez simplement à ce que les utilisateurs de votre application et vos données puissent le comprendre facilement. Tous ceux qui travaillent sur la logique de votre application sauront-ils ce que c'est que B07 ?

Pour chacune de nos variables, nous devons ajouter les réponses possibles. Dans CSPro, l'ensemble des réponses valides d'une question s'appelle le "value set" (l'ensemble de valeurs). L'ensemble de valeurs dénombre toutes les réponses valides avec leurs étiquettes correspondantes. A l'absence du value set, l'intervieweur peut saisir n'importe quelle valeur (sauf le blanc), mais avec le value set, elles sont limitées aux options définies dans l'ensemble de valeurs. Sans valeurs définies, les utilisateurs peuvent même saisir des

[1] Le numéro de ligne n'est pas nécessaire dans CSPro lui-même car il est possible de déterminer le numéro de ligne à l'aide de la logique. Cependant, lors de l'exportation des données vers d'autres logiciels, il est souvent utile de les avoir. Nous verrons plus tard comment le remplir automatiquement lors de la saisie des données.

nombres négatifs. Pour cette raison, il est recommandé d'utiliser un ensemble de valeurs pour toutes les variables numériques.

Définissez les ensembles de valeurs pour certaines de nos variables sur la base des codes de réponses du questionnaire :

- Milieu de résidence (1- urbain, 2- périurbain, 3- rural)
- Province (copier / coller depuis Excel)
- Numéro d'ordre (intervalle : 1- 30, utilisez "1-30" comme étiquette pour la plage)
- Prénoms et Nom (aucune value set)
- Sexe (1 - male, 2 - female)
- Lien de parenté (voir le questionnaire)
- Age (utilisez une fourchette : 0-120, plus un code inconnu de 999). Nous pouvons associer le code 999 avec la valeur spéciale "missing". Cela indique à CSPro qu'il s'agit d'un code spécial indiquant que la réponse est inconnue. Nous verrons ultérieurement comment nous pouvons en tirer parti lorsque nous effectuons des contrôles de cohérence.
- Nombre de pièces (utilisez "create value set" pour générer les codes de 1 à 20)
- Type de logement principal (utilisez les codes du questionnaire)

Dictionary Macros

Il existe quelques fonctions utiles pour travailler avec les dictionnaires auxquelles vous pouvez accéder en cliquant avec le bouton droit de la souris sur le dictionnaire dans l'arborescence située à gauche de l'écran et en choisissant "Dictionary Macros". En particulier, vous pouvez copier / coller tous les ensembles de valeurs ou tous les noms / libellés des variables du dictionnaire vers / depuis Excel. Cela peut être utiliser pour créer des livres de code à partager avec des personnes n'ayant pas accès à CSPro. Il peut également être utilisé pour effectuer des modifications en masse sur des variables du dictionnaire, telles que la renumérotation des valeurs dans les ensembles de valeurs ou l'ajout de préfixes aux noms des variables.

Formulaires

Avant de pouvoir saisir des données, nous devons créer des formulaires de saisie de données. Pour commencer, cliquez sur la pile jaune de formulaires dans la barre d'outils. Pour suivre l'aspect du questionnaire papier, nous allons créer un formulaire pour chaque page du questionnaire papier.

Créez un formulaire pour la section A: Identification. Faites glisser et déposez les id-items sur le formulaire. Notez que nous pouvons faire glisser et déposer des éléments individuels ou des enregistrements entiers. En cliquant avec le bouton droit de la souris sur le formulaire dans l'arborescence de formulaires à gauche de l'écran, vous pouvez modifier le libellé et le nom du formulaire. Utilisons "A: Identification" pour le libellé et "IDENTIFICATION_FORM" comme nom.

Créez un formulaire pour la section B: Caractéristiques Individuelles. Faites glisser les éléments de l'enregistrement individu. Donnons le libellé de formulaire "B: Caractéristiques Individuelles" et nommons le "CARACTERISTIQUES_INDIVIDU_FORM". Notez que lorsque nous glissons l'enregistrement, nous avons l'option de mettre les variables dans un roster (grille) ou dans un formulaire à répéter. Si nous déposons les variables sur le formulaire d'identification, nous ne pouvons que les mètres en roster comme le formulaire d'identification ne peut pas être répétée. Pour notre exemple, utilisons un roster.

Lorsque nous créons les rosters, CSPro leur attribue automatiquement un nom se terminant par "000", par exemple "INDIVIDU000". Vous pouvez le voir dans l'arborescence des formulaires à gauche de l'écran. Nous pouvons le modifier en cliquant avec le bouton droit de la souris sur le roster dans l'arborescence des formulaires et en choisissant "Properties...". Appelons notre roster "CARACTERISTIQUES_INDIVIDU_ROSTER".

Créez un formulaire pour la section: F: Caractéristiques de l'habitation. Faites glisser et déposez les éléments de l'enregistrement de l'habitat.

Pour les enquêtes sur papier, nous passerions beaucoup de temps sur la mise en page, en ajoutant des étiquettes et des cadres supplémentaires pour que le formulaire ressemble exactement au questionnaire papier. Toutefois, lorsqu'il est affiché sur Android, le formulaire est affiché question par question, alors il ne vaut pas l'effort de faire la mise en page.

Lancer l'application sous Windows

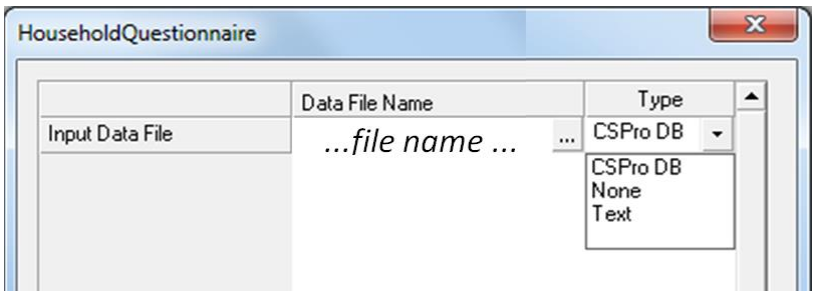
Cliquez sur l'icône de feu vert dans la barre d'outils pour exécuter l'application de saisie de données sur votre PC. Entrez le nom du fichier de données. Pour les fichiers de données, CSPro utilise l'extension ".csdb". Appelons notre fichier de données "menages.csdb".

Types de fichiers de données

Par défaut, CSPro stocke les données dans un fichier de base de données CSPro avec l'extension .csdb. Ce fichier contient non seulement les données elles-mêmes, mais également les notes, l'index, le statut de sauvegarde partielle et les métadonnées utilisées pour la synchronisation des données. Les fichiers de base de données CSPro peuvent être visualisés en double-cliquant dessus ou en utilisant l'outil DataViewer dans le menu Tools.

L'extension csdb est nouvelle dans CSPro 7.0. Dans les versions antérieures de CSPro, les notes, l'index et la sauvegarde partielle étaient stockés dans des fichiers supplémentaires qui accompagnaient le fichier de données lui-même. Contrairement au fichier texte utilisé par les versions précédentes de CSPro, il s'agit d'un fichier binaire qui ne peut pas être affiché à l'aide de l'outil TextViewer. Bien qu'il soit encore possible d'utiliser des fichiers texte dans CSPro 7.0, il est vivement recommandé d'utiliser CSPro DB à sa place. Des nouvelles fonctionnalités, telles que la synchronisation intelligente et les libellés de cas, ne sont pas prises en charge à l'aide de fichiers texte.

Lorsque vous lancez une application de saisie de données CSPro, vous pouvez sélectionner le nom du fichier de données et le type de fichier de données que vous souhaitez utiliser :



- **CSPro DB:** Ceci est un fichier de base de données CSPro. Il contient les données et les métadonnées.
- **None:** Il n'y a pas de fichier de données associé. Ceci est utile lors de la création de menus.
- **Text:** Ceci est un fichier texte UTF8. C'est ce que CSPro a utilisé dans les versions précédentes. Il est fourni pour la compatibilité.

Contrôler la taille du roster

Actuellement notre programme n'a aucun moyen d'arrêter la saisie des membres du ménage avant de compléter toutes les 30 lignes du roster. Pour limiter le nombre de lignes du roster au nombre de membres réels du ménage, l'intervieweur peut d'abord saisir le nombre de membres du ménage et puis le program peut l'utiliser pour limiter la taille du roster. Ajoutez une nouvelle variable, "nombre de membres du ménage", pour contrôler la taille du roster. À quel enregistrement devons-nous l'ajouter ? Cela ne peut pas être l'enregistrement individu, car nous ne voulons pas que cela se répète. Ajoutons-le à l'enregistrement des caractéristiques du ménage, mais placez-le sur le formulaire individu (section B). Cliquez avec le bouton droit sur le roster, choisissez "Properties" et définissez ce nouveau champ comme "occurrence control field".

Essayez de relancer l'application. Malheureusement, lorsque nous arrivons au formulaire individu, nous entrons dans le roster au lieu du champ "nombre de membres du ménage". Par défaut, CSEntry respecte l'ordre dans lequel les champs ont été ajoutés au formulaire. Nous pouvons modifier l'ordre en faisant glisser les champs dans l'arborescence des formulaires. Déplacez le champ "nombre de membres du ménage" dans l'arborescence de manière à ce qu'il vienne avant le roster. Maintenant, le champ de contrôle de la liste devrait fonctionner. Dans la prochaine leçon, nous utiliserons la logique pour sortir du roster sans le champ de contrôle.

Lancer l'application sur Android

Pour installer l'application sur un téléphone / une tablette, nous devons procéder comme suit :

1. Publiez le fichier .pen (menu File-> Publish Entry Application).
2. Connectez la tablette au PC à l'aide d'un câble USB. La tablette devrait apparaître comme une clé USB.
3. Copiez les fichiers Popstan2020.pen et Popstan2020.pff vers le répertoire CSEntry de la tablette.
4. Démarrer CSEntry sur la tablette.

5. Saisir les données (notez les différences entre Android et Windows).
6. Lorsque vous avez terminé, reconnectez la tablette à l'ordinateur et copiez le fichier menage.csdb de la tablette vers le PC. Sur certaines tablettes (Nexus 7 par exemple), la première fois que le fichier de données est créé, il peut ne pas s'afficher lors de la connexion au PC. Si tel est le cas, redémarrez la tablette et le fichier sera affiché.

Lorsque nous avons copié l'application sur la tablette, nous avons copié le fichier pen et le fichier pff. Le fichier pff contient les paramètres sur le lancement de l'application de saisie de données, y compris le nom du fichier de données à utiliser. Vous pouvez modifier le fichier pff en cliquant dessus avec le bouton droit de la souris dans l'Explorateur Windows et en choisissant "Modifier". Cela vous permet de modifier le nom du fichier de données, d'obliger l'application à démarrer en mode d'ajout ou mode de modification et de verrouiller certaines parties de l'interface utilisateur.

Sur Android, la liste des applications disponibles sur le périphérique est construite en recherchant tous les fichiers pff du répertoire CSEntry et ses sous-répertoires. Cela veut dire qu'un fichier pen sans sa pff n'apparaîtra pas dans la liste.

DataViewer

Double-cliquant sur le fichier menage.csdb l'ouvre dans l'outil DataViewer. DataViewer offre un moyen simple d'afficher les données saisies.

Sauvegarde partielle et arborescence de cas

La vérification d'une question figurant sur le troisième formulaire de l'enquête peut être pénible si l'on doit refaire la saisie de toutes les données jusqu'à la question que vous testez. Nous pouvons activer la sauvegarde partielle sous les options de saisie de données afin de pouvoir quitter la saisie de données, modifier l'application et revenir à l'endroit où nous nous sommes arrêtés. Pour modifier les options de saisie choisissez "Data Entry" du menu "Options". Pendant que nous sommes sur les options de saisie de données, nous pouvons également activer l'arborescence de cas sous Windows et Android pour faciliter la navigation dans le questionnaire pendant la vérification. L'arborescence de cas est activée par défaut sur Android car Android n'affiche qu'une question à la fois, mais il est désactivé par défaut sous Windows. Notez que sur les téléphones mobiles, comme il n'y a pas assez d'espace pour afficher l'arborescence et les questions en même temps, vous devez appuyer sur le grand "CS" vert dans le coin supérieur gauche de l'écran pour afficher l'arborescence.

En plus des options de saisie de données, les options du menu Options → Data Sources contient d'autres paramètres utiles. Par exemple, vous pouvez ici activer la sauvegarde partielle automatique à un intervalle régulier. Ceci est important pour les longues interviews car en cas de problème avec le logiciel ou l'appareil au cours d'interview, vous risquez de perdre des données si le cas n'a pas été partiellement sauvegardé.

Modification du dictionnaire après la collecte des données

Ajoutez la variable **F02 nombre de pièces** au dictionnaire et au formulaire juste avant le type de logement principal **F03**. N'oubliez pas de modifier l'ordre dans l'arborescence du formulaire afin qu'il soit posé avant **F03**. Exécutez à nouveau l'application et modifiez l'un des ménages existants. Pourquoi le nombre de membres du ménage est-il vide ? Notre dictionnaire ne correspond plus au fichier de données car l'introduction de la nouvelle variable à changer les positions dans le fichier des variables qui la suivent. Une fois les données saisies, évitez d'ajouter des variables au milieu d'un enregistrement. Il est bien d'ajouter à la fin de l'enregistrement, mais ajouter au milieu invalide les positions de départ des variables existantes. Si jamais vous devez ajouter une variable au milieu d'un enregistrement, vous pouvez utiliser l'outil de Reformat Data pour ajuster les anciens fichiers de données afin qu'ils correspondent au nouveau dictionnaire.

Sous-éléments

Ajoutons la **date de naissance (B06)** à l'application. Afin de pouvoir afficher la date sous forme de nombre à 8 chiffres et en plus de pouvoir afficher le jour, le mois et l'année individuellement, nous pouvons créer une variable avec des sous-éléments. Les sous-éléments sont des éléments constitués d'un sous-ensemble des chiffres de leur variable parente. Ajoutez la variable pour la date de naissance et les sous-éléments suivants :

- année de naissance (longeur 4)
- mois de naissance (longeur 2)
- jour de naissance (longeur 2)

Nous plaçons l'année d'abord suivi par le mois et puis le jour car ce format fonctionnera mieux avec les autres fonctionnalités de CSPro que nous verrons plus tard. Cliquez sur **Layout** dans la barre d'outils pour vous assurer que l'élément et les sous-éléments se chevauchent. Ajoutez les sous-éléments au formulaire, ajoutez les ensembles de valeurs pour chaque sous-élément et vérifier l'application. Notez que lorsque nous ajoutons les sous-éléments au formulaire, il n'est pas nécessaire de conserver le même ordre que dans le dictionnaire. Sur le formulaire, nous pouvons mettre le jour, le mois et puis l'année

Travail de groupe

Ajoutez un nouvel enregistrement au dictionnaire pour la section E du questionnaire (décès). Appelez cet enregistrement "Décès". Combien d'occurrences devrait-il avoir ? N'incluez pas les questions E01 et E02 dans le nouvel enregistrement car ils n'ont pas le même nombre d'occurrences que les autres variables de cette section. Ajoutez-les à l'enregistrement de l'habitat. Créez un nouveau formulaire pour la section E et ajoutez-y les champs pour créer un roster. Utilisez E02 comme occurrence control field dans la roster pour limiter le nombre de lignes au nombre de décès dans le ménage. Vérifier l'application sur Windows et sur Android.

Ensembles de valeurs liées

Notez que les questions **B07 (lieu de naissance)** et **B08 (résidence il y a 1 an)** ont le même ensemble de valeurs. Nous pourrions simplement copier et coller l'ensemble de valeurs dans les deux éléments, mais cela nous laisserait deux copies du même ensemble de valeurs. Si plus tard nous changeons l'un d'eux et oublions de changer l'autre, ils seront désynchronisés. Cela pourrait causer des problèmes de cohérence ultérieurement. Pour éviter ce problème, nous pouvons créer l'ensemble de valeurs de **B07**, puis le coller en tant qu'ensemble de valeurs lié ("linked value set") dans **B08**. Avec un ensemble de valeurs liées, CSPro ne stocke qu'une copie de l'ensemble de valeurs partagé entre les deux variables. Ainsi, si vous modifiez l'ensemble de valeurs, la modification est reflétée dans les deux variables.

Variables aux plusieurs occurrences

Ajoutons les questions **F04** et **F05** sur le nombre de logements à l'application. **F04** est une question simple oui / non. **F05** demande le nombre de logements de chacun des cinq types. Nous pourrions créer cinq variables différentes dans le dictionnaire : LOGEMENTS_CASES_RONDES, LOGEMENTS_INDIVIDUELLES, LOGEMENTS_JUMELEES, etc. Mais, pour simplifier notre dictionnaire, nous pouvons créer une seule variable dans le dictionnaire, LOGEMENTS, avec cinq occurrences : une pour chaque type. Lorsque nous faisons glisser cette variable sur le formulaire des caractéristiques du logement, nous obtenons un roster de 5 lignes.

Libellés des occurrences

Avec cette approche, notre formulaire n'indique pas les types de logements, mais nous pouvons résoudre ce problème en utilisant des libellés des occurrences. Sélectionnez la variable **F05** dans le dictionnaire et choisissez "Occurrence Labels ..." dans le menu Edition. Ajoutez les noms des cinq types de logement dans la grille qui s'affiche. Notez que vous pouvez copier à partir d'Excel et coller dans cette boîte de dialogue. Maintenant, lorsque nous faisons glisser la variable vers le formulaire, le roster indique le type de logement pour chaque ligne.

Cases à cocher

Nous pourrions également utiliser une variable à plusieurs occurrences pour la question **B11**, handicaps, mais cela peut être fait plus facilement à l'aide des cases à cocher. Les cases à cocher offrent une interface conviviale pour les questions à réponses multiples en présentant un seul écran avec une case à cocher pour chaque option plutôt que de présenter les options une par une.

Dans CSPro, les questions à réponses multiples se fassent sous forme de variables alpha dont la longueur est égale au nombre d'options pouvant être sélectionnées simultanément. L'ensemble de valeurs à une valeur pour chaque option qui est généralement une seule lettre. La valeur résultante est une chaîne de caractères contenant les valeurs pour chacun des éléments sélectionnés.

Pour les handicaps, nous utiliserons l'ensemble de valeurs suivant :

Aveugle	A
Sourd	B
Muet	C
Infirmité membres inférieurs	D
Infirmité membres supérieurs	E
Déficience mentale	F

Si l'intervieweur coche les cases Aveugle, Muet et Déficience mentale, la valeur de la variable sera "ACF". Nous verrons plus loin comment convertir la valeur alpha en une série de valeurs oui / non pour simplifier l'analyse.

Types de saisie

Si vous cliquez avec le bouton droit de la souris sur le champ des handicaps du formulaire et choisissez "Field properties ...", vous verrez que le “capture type” est “checkbox” (case à cocher). Il existe d’autres types de saisie :

- Text Box : saisie de données au clavier
- Radio button : choisissez une option parmi plusieurs avec un bouton radio pour chaque option
- Drop Down: choisissez une option parmi plusieurs sans boutons radio
- Combo Box: combinaison de saisie au clavier et une liste déroulante (même chose que Drop Down sous Windows)
- Checkbox : choix de plusieurs réponses avec des cases à cocher

Lorsque vous faites glisser une variable du dictionnaire vers le formulaire, CSPro définit le type de saisie en fonction de l'ensemble des valeurs de la variable. Si aucune ensemble de valeurs n'est définie lorsque vous déposez la variable, le type de saisie sera “Text Box”. Vous pouvez toujours modifier le type de capture à l'aide de la boîte de dialogue “Field Properties”.

Saisie des dates

Ajoutons la date de l'interview à la section d'identification. Dans quel enregistrement devrait-on l’ajouter ? Elle pourrait aller sur n'importe quel enregistrement à un seul occurrence tel que l’habitation, mais créons un nouvel enregistrement appelé INTERVIEW_REC pour

stocker les variables de la section A qui ne font pas partie des id-items et ajoutons-le ici. Ajoutons une variable à huit chiffres pour la date de l'interview. Nous pouvons également ajouter des sous-éléments pour l'année, le mois et le jour de l'interview. Faites glisser la variable de la date de l'interview sur le formulaire. Lorsque vous la faites glisser, n'utilisez pas les sous-éléments, mais seulement la variable même. Modifiez le type de capture de la date de l'interview en “Date” et choisissez le format de YYYYMMDD (4 chiffres pour l'année, suivi par 2 pour le mois et ensuite 2 pour le jour) pour correspondre à l'ordre des sous-éléments année, mois et jour. Avec le type de capture “Date” CSPro affiche un petit calendrier sur Windows et des champs roulantes sur Android.

Saisir les noms d'abord

Actuellement, l'intervieweur doit renseigner toutes les informations démographiques de la première personne avant de passer à la personne suivante du ménage. En pratique, il est plus simple de leur demander de ne renseigner que le nom, l’âge, le sexe et le lien de parenté de tous les membres du ménage, puis de ne renseigner que les informations restantes une fois que toutes les informations de base de tous les membres du ménage ont été saisies. Pour le faire, nous devons extraire uniquement **NUMERO_ORDRE**, **NOMS**, **SEXE**, **LIEN**, **AGE** et **DATE_DE_NAISSANCE** dans un roster à part. Enlever ces variables du roster **CARACTERISTIQUES_INDIVIDU_ROSTER,** créer un nouveau formulaire avant le **CARACTERISTIQUES_INDIVIDU_FORM** appelé **MEMBRES_MENAGE_FORM** et déposez **NUMERO_ORDRE** là-dessous pour faire un nouveau roster. Renommez ce nouveau roster **MEMBRES_MENAGE_ROSTER**. Déposez **NOMS** du dictionnaire sur **MEMBRES_MENAGE_ROSTER** pour l’ajouter à la grille. Notez que lorsque vous déposez une variable d’un enregistrement répété au-dessus d'un roster existant, il s’ajoute au roster, mais si vous la déposez en dehors de la grille, il en crée une nouvelle. Déposez les variables **SEXE**, **LIEN**, **AGE** et **DATE_DE_NAISSANCE** sur le nouveau roster. Supprimez le champ **NOMBRE_MEMBRES_MENAGE** du formulaire **CARACTERISTIQUES_INDIVIDU_FORM** et déposez-le sur **MEMBRES_MENAGE_FORM**. Dans l'arborescence des formulaires, déplacez le champ **NOMBRE_MEMBRES_MENAGE** de manière à ce qu'il vienne avant le **MEMBRES_MENAGE_ROSTER**. Définissez le champ de contrôle d'occurrence de **MEMBRES_MENAGE_ROSTER** comme **NOMBRE_MEMBRES_MENAGE**.

Conseil : Lorsque vous faites glisser des variables sur un formulaire avec un roster existante, déposez-les à l'intérieur de la grille ou vous obtiendrez un deuxième roster pour la nouvelle variable.

Blocs

La fonctionnalité de blocs vous permet de regrouper plusieurs champs dans une unité qu’on appelle un bloc. Les blocs sont principalement utiles pour deux raisons. Premièrement, ils fournissent le mécanisme permettant à CSEntry d’afficher plusieurs champs sur un même écran sous Android. Deuxièmement, ils vous permettent d’écrire des contrôles logiques à un endroit qui s’appliquent à un certain nombre de champs.

Ajoutons un bloc pour regrouper les champs de la date de naissance. Pour créer un nouveau bloc, sélectionnez les champs du jour, du mois et de l'année dans l'arborescence des formulaires. Pour sélectionner plusieurs champs on maintient la touche Ctrl enfoncée tout en cliquant sur les champs dans l'arborescence. Une fois que les champs soits sélectionnés, cliquez avec le bouton droit de la souris et sélectionnez “Add Block...”. Dans la boîte de dialogue "Block Properties", saisissez "Date de naissance" pour le libellé et "DOB_BLOCK" pour le nom. Laissez l'option "Show on same screen on mobile" cochée pour afficher les champs groupés sur le même écran sur les appareils mobiles.

Exercices

1. Ajoutez les champs restants de la section de l'habitat du questionnaire au dictionnaire et au formulaire de l'habitat (F06 à F13). Assurez-vous d'ajouter les ensembles de valeurs appropriés.
2. Ajoutez les champs restants de la section caractéristiques des individus (B) du questionnaire au dictionnaire et au formulaire caractéristiques des individus. Assurez-vous d'ajouter les ensembles de valeurs appropriés. Pour la langue parlée (B15), utilisez les cases à cocher.
3. Ajoutez les champs de la section C, Education, au dictionnaire. Ajoutez-les à l'enregistrement individu. Créez un nouveau formulaire pour la section C et déposez-y les variables du dictionnaire pour créer un roster. Définissez le champ de contrôle d'occurrence du roster comme le nombre de membres du ménage.
4. Ajoutez les heures de début et de fin de l'interview au formulaire d'identification après la date de de l'interview. Utilisez des subitems (sous-éléments) pour les heures et les minutes. Ajoutez des ensembles de valeurs appropriés.
5. Ajoutez un nouveau enregistrement et un nouveau formulaire pour la section G, biens appartenant au ménage. Étant donné que G01 (quantité et valeur) se répète, placez ces éléments dans leur propre enregistrement à plusieurs occurrences, sans inclure G02. Ce roster devrait inclure le code de possession (1-10), la quantité et la valeur par unité. Définissez les libellés d'occurrence dans l'enregistrement de G01 avec les noms des biens. Faites de G02 un champ de case à cocher unique et insérez-le dans l'enregistrement de l'habitat car il ne se répète pas. L'ensemble de valeurs définie pour G02 doit porter les noms de possession comme libellés avec les codes "A", "B", "C"…

Assurez-vous de vérifier votre application sous Windows et Android.

Session 02 : Sauts et Remplissages Automatiques

À la fin de cette leçon, les participants seront en mesure de :

- Utiliser les commandes skip et ask if pour sauter les champs
- Utilisez if then else pour faire des sauts conditionnels
- Terminez les rosters à l'aide de la commande endgroup
- Remplissez automatiquement les variables avec la logique

Sauts

Dans la question **F04** (autres types de logements), nous devons sauter la question **F05** (nombre de logements) s'ils répondent «non».

Pour sauter un champ, nous passons par la logique. Pour ajouter la logique à une application CSPro, cliquez sur le bouton logique  de la barre d'outils pour passer au mode logique. Dans ce mode, nous pouvons faire la programmation dans la langue de CSPro pour contrôler le programme. La logique est ajoutée à une PROC (procédure) généralement associé à un champ du formulaire. Pour accéder à la PROC d'un champ dans le mode logique, il suffit de cliquer sur le champ dans l'arborescence de formulaire à gauche. Nous allons ajouter notre saut à la PROC du champ **AUTRES_LOGEMENTS,** alors cliquez sur **AUTRES_LOGEMENTS** dans l'arborescence du formulaire.

Pour sauter un champ, nous utilisons la commande skip. Afin de sauter seulement quand la réponse est "non", nous devons combiner le saut avec la commande if comme le suivant :

```
PROC AUTRES_LOGEMENTS

if AUTRES_LOGEMENTS = 2 then
    skip to STATUT_OCCUPATION;
endif
```

La commande if n'exécute le code entre le then et le endif si la condition (AUTRES_LOGEMENTS = 2) est vraie, sinon elle passe directement à la première commande après le endif.

Cela sautera directement à STATUT_OCCUPATION pour les personnes qui répondent "non" à AUTRES_LOGEMENTS, en sautant le roster LOGEMENTS_ROSTER.

En plus de "=" pour vérifier si deux valeurs sont égales, vous pouvez utiliser les opérateurs suivants :

Opérateur	**Symbol**
égal à	=
différent de	<>
inférieur à	<
inférieur ou égal à	<=
supérieur à	>
supérieur ou égal à	> =

Par exemple, pour sauter **B08**, lieu de résidence il y'a un an, pour ceux qui sont âgés de moins d'un an, nous pourrions ajouter la logique suivante à la PROC de la question précédente :

```
PROC LIEU_RESIDENCE
 if AGE <1 then
    skip to SURVIE_MERE;
endif
```

Notez qu'une fois que le champ soit sauté, vous ne pouvez plus y accéder en cliquant dessus ou en revenant du champ suivant. CSEntry à noter que ce champ a été sauté et ne nous laissera plus y entrer tant que nous n'aurons pas changé la valeur de la variable **AGE**. C'est une différence importante entre le mode “system control” et le mode “operator control”.

Nous devrions ajouter un commentaire pour clarifier aux autres lecteurs de notre logique afin qu'ils comprennent ce que nous essayons de faire. Cela aidera non seulement les autres à comprendre notre logique, mais nous aidera également à la comprendre lorsqu'on y revient après quelques mois.

```
PROC LIEU_RESIDENCE
// Ne demandez pas lieu de résidence il y a un an pour ceux qui sont âgés de
// moins d'un an
if AGE < 1 then
    skip to SURVIE_MERE;
endif
```

Tout ce qui se trouve après le "//" est considéré comme un commentaire et est ignoré par CSPro. Vous pouvez également utiliser {} pour les commentaires multilignes. Il est recommandé d'utiliser beaucoup de commentaires pour documenter votre logique.

Une remarque sur le style de codage

Lors de la rédaction des instructions if, votre code sera plus lisible et plus facile à comprendre pour les autres si vous faites l'indentation des instructions entre les éléments then et endif. Pour faire indenter une ligne, tapper sur la touch "Tab" du clavier. Pour enlever l'indentation, appuyer sur "Shift" et tapper "Tab". Vous pouvez aussi utiliser la fonction de formatage automatique du CSPro à partir du menu "Edit".

Il est également utile d'utiliser une capitalisation cohérente. Nous recommandons toutes en majuscules pour les variables du dictionnaire comme **NOMBRE_MEMBRES_MENAGE** et toutes en minuscules pour les mots clés CSPro tels que if, then et endif. Enfin, vous devez utiliser autant que possible les commentaires pour aider les autres, et vous-même dans le futur, à mieux comprendre votre code.

Ask if

Une alternative au saut c'est la commande ask if. Il ignore la question actuelle si la condition n'est *PAS* remplie. Par exemple, pour utiliser ask if, afin de ne poser la question **LIEU_RESIDENCE_UN_AN** (**B08**) que pour les individus âgés d'un an et plus, nous pouvons utiliser la logique suivante :

```
PROC LIEU_RESIDENCE_UN_AN
preproc
ask if AGE >= 1
```

La commande ask if saute la question, sauf si la condition AGE >= 1 est remplie. C'est-à-dire que, si l'âge est inférieur à 1, la question sera ignorée. Notez que lors du passage de skip à ask si, la condition a été inversée passant de "<" à ">=".

La ligne preproc indique à CSPro d'exécuter cette logique *avant de* poser la question. Nos exemples précédents ne préciser pas preproc, donc, par défaut, la logique était exécutée dans la postproc. La logique de la postproc est exécutée *après* la saisie des données. La logique de la preproc est exécutée avant que les données ne soient saisies. Pour chaque variable, vous pouvez préciser une preproc, une postproc ou les deux. Ask if sera toujours codé dans le preproc du champ sauté, alors que skip peut être soit dans la preproc du champ sauté, soit dans la postproc du champ précédent.

Travail de groupe

Mettez en œuvre le saut pour les questions **F10** (lieu d'aisance privé) et **F11** (type de lieu d'aisance). Si la réponse à la question **F10**, type de lieu d'aisance, est "non", passez à la question **F11**, type de lieu d'aisance. Vous pouvez utiliser skip ou ask if. N'oubliez pas d'ajouter un commentaire à votre code et de le formater en utilisant la bonne l'indentation et capitalisation.

Conditions composées

Prenons un autre exemple, nous allons implémenter le saut pour la question **C01**. Nous devons ignorer le **C02**, le niveau d'instruction, si la fréquentation scolaire est "jamais fréquentée" (1) ou "ne sait pas" (9). Nous pouvons implémenter cela avec deux instructions if :

```
PROC FREQUENTATION_SCOLAIRE

// Ignorer le niveau d'instruction pour ceux qui n'ont jamais fréquenté l'école
if FREQUENTATION_SCOLAIRE = 1 then
   skip to ALPHABETISATION;
endif;

// Ignorer le niveau d'instruction pour ceux dont la fréquentation n'est pas connue
if FREQUENTATION_SCOLAIRE = 9 then
   skip to ALPHABETISATION;
endif;
```

Notez que dans CSPro, les commandes logiques doivent être séparées par des points-virgules (;). Cela indique à CSPro quand une commande se termine et que la commande suivante commence. Oublier de mettre le point-virgule à la fin d'une instruction est une erreur très courante commise par les nouveaux utilisateurs. Si vous oubliez le point-virgule, CSPro vous indiquera généralement qu'un point-virgule était attendu, bien que parfois il devient confus et vous affiche un message d'erreur moins informatif.

Nous pouvons simplifier ce code en combinant les deux instructions if en utilisant l'opérateur "or" (ou) pour créer une condition composée.

```
PROC FREQUENTATION_SCOLAIRE

// Ignorer le niveau d'instruction pour ceux dont la fréquentation est "jamais
// fréquentée" ou "ne sait pas"
if FREQUENTATION_SCOLAIRE = 1 or FREQUENTATION_SCOLAIRE = 9 then
   skip to ALPHABETISATION;
endif;
```

En plus de l'opérateur "or", vous pouvez également créer des conditions composées à l'aide de l'opérateur "and" (et). Une instruction composée liée par "or" est vraie si l'une ou l'autre des conditions est vraie mais une instruction composée utilisant "and" n'est vraie que si toutes les deux conditions sont vraies.

En plus de "ou", CSPro offre les opérateurs logiques suivants :

Opération	**Mot clé**
Négation d'une expression	not
Vraie si les deux expressions sont vraies	and
Vraie si l'une des expressions est vraie	or

Vous pouvez également utiliser des conditions composées avec ask if. Par exemple, nous voulons seulement poser la question **B14**, âge au premier mariage, pour ceux qui ont répondu "mariés", "veufs" ou "divorcés" pour l'état matrimonial.

```
PROC AGE_PREMIER_MARRIAGE
preproc
// Demandez uniquement l'âge au premier mariage pour ceux dont l'état matrimonial
// indique qu'ils sont / ont été mariés,
ask if ETAT_MATRIMONIAL= 2 or ETAT_MATRIMONIAL = 3 or
        ETAT_MATRIMONIAL = 4;
```

Une expression "or" est vraie lorsque les expressions de gauche ou de droite sont vraies. La question ne sera posée que si ETAT_MATRIMONIAL vaut 2, 3 ou 4.

Nous pouvons rendre notre expression encore plus simple en utilisant l'opérateur "in" qui vérifie si une valeur est comprise dans une plage :

```
PROC AGE_PREMIER_MARRIAGE
preproc
// Demandez uniquement l'âge au premier mariage pour ceux dont l'état matrimonial
// indique qu'ils sont / ont été mariés,
ask if ETAT_MATRIMONIAL in 2:4;
```

Questions “Autres (à préciser)”

Un modèle courant dans CSPro consiste à utiliser la commande ask if pour les questions avec une réponse “autres (à préciser)”. Prenons la question **F08**, matériau du toit, comme exemple. Premièrement, nous ajoutons une nouvelle variable alpha au dictionnaire pour capturer la valeur "autre". Nous pouvons l'appeler **MATERIAU_TOIT_AUTRE** et le déposer sur le formulaire afin qu'il vienne juste après **MATERIAU_TOIT**. Nous voulons seulement

demander la valeur "autre" s'ils ont sélectionné "autre" (le code 5) comme réponse pour **MATERIAU_TOIT**, sinon nous voulons le sauter. Nous pouvons le faire en utilisant ask if dans la preproc de **MATERIAU_TOIT_AUTRE**.

```
PROC MATERIAU_TOIT_AUTRE
preproc

// pose cette question uniquement si "Autre (à préciser)" est sélectionné dans le
// matériau du toit.
ask if MATERIAU_TOIT = 5;
```

Skip to next

Ajoutons maintenant le saut entre la langue des signes (**B12**) et l'état matrimonial (**B13**) pour passer à la personne suivante pour les membres du ménage âgés de moins de 10 ans. Dans ce cas, à quoi sautons-nous ? Nous ne pouvons pas passer à **B01** (numéro de ligne) car cela va essayer de nous renvoyer au numéro de ligne de la ligne en cours. Au lieu de cela, nous utilisons skip to next qui passe automatiquement au premier champ de la ligne suivante du roster. Nous allons mettre ce saut dans la postproc de **LANGUE_SIGNES**.

```
PROC LANGUES_SIGNES
// Sauter à la personne suivante pour les membres du ménage âgés de moins de 10
ans
if AGE < 10 then
   skip to next;
endif;
```

Ajoutons aussi le saut dans la section C, Éducation, pour les individus âgés moins de 3 ans. Nous ne sommes censés remplir la section éducation que pour ceux âgés de trois ans et plus. Nous pouvons utiliser skip to next pour sauter les questions relatives à l'éducation si l'âge est inférieur à trois ans. Nous devons faire cela dans la preproc de **C01**, le premier champ du fichier d'éducation, **FREQUENTATION_SCOLAIRE**.

Notez que, puisque ce champ a déjà une postproc, nous devons d'abord ajouter la preproc, puis le mot postproc de sorte que la logique originale ne fasse pas partie de la preproc.

```
PROC FREQUENTATION_SCOLAIRE
preproc

// Sauter la section éducation pour les membres du ménage moins de 3 ans
if AGE < 3 then
   skip to next;
endif;

// Sauter le niveau d'instruction pour ceux dont la fréquentation est "jamais
// fréquentée" ou "ne sait pas"
if FREQUENTATION_SCOLAIRE = 1 or FREQUENTATION_SCOLAIRE = 9 then
   skip to ALPHABETISATION;
endif;
```

Pré-remplissage

Remplissons automatiquement le numéro de ligne afin que l'intervieweur n'ait pas à le saisir. Nous pouvons le faire dans la preproc de **NUMERO_ORDRE** :

```
PROC NUMERO_ORDRE
preproc

// Remplir automatiquement le numéro de ligne
NUMERO_ORDRE = curocc();
```

La fonction curocc() nous donne le numéro d'occurrence actuel d'un enregistrement ou d'une variable en répétition. C'est-á-dire le numéro de rangée du roster sur laquelle nous nous trouvons actuellement.

Pour éviter d'avoir à taper la touche entrée pour parcourir ce champ, nous pouvons utiliser la commande noinput qui accepte la réponse attribuée et passe automatiquement au champ suivant comme si l'enquêteur avait utilisé la touche entrée.

```
PROC NUMERO_ORDRE
preproc

// Remplir automatiquement le numéro de ligne
NUMERO_ORDRE = curocc();
noinput;
```

Par ailleurs, nous pouvons rendre le champ non modifiable en cochant la case “Protected” (protégée) dans les propriétés du champ. Avec noinput, il est toujours possible de revenir en arrière et de modifier le champ, mais les champs protégés ne peuvent être modifiés que par la logique. Sachez que si vous ne définissez pas la valeur d'un champ protégé dans la logique *avant que* l'intervieweur n'arrive sur la variable, CSEntry vous donnera une erreur fatale.

Travail de groupe

Dans la question **B10**, le numéro de la ligne de la mère, remplissez automatiquement le code 88 (décédé) si la réponse à la question **B09**, survie de la mère, est "non". C'est-à-dire que si dans la question **B09**, l'individu a indiqué que sa mère n'est plus en vie, l'application de saisie mettra, de manière automatique, la réponse 88 (décédé) pour la question **B10** et continuerez à la question **B11**. Vous pouvez être tenté d'utiliser l'instruction skip pour cela, mais vous devez plutôt utiliser l'instruction noinput.

Endgroup

Au lieu d'utiliser le champ de contrôle d'occurrence du roster, nous pourrions demander à l'utilisateur s'il souhaite terminer la liste des membres du ménage. On ajoutera la question "Y a-t-il d'autres membres de ce ménage ?". Si la réponse est "non", on termine la saisie du roster. Pour ce faire, nous ajoutons d'abord une nouvelle variable au dictionnaire et au roster. Appelons-le **AUTRES_MEMBRES** et donnons-lui l'ensemble de valeurs Oui - 1, Non - 2. Nous le mettrons à la fin du roster des noms. Si l'intervieweur choisit "non", nous utilisons la commande endgroup qui termine la saisie du roster actuelle ou du formulaire à répétition.

```
PROC AUTRES_MEMBRES

// Quitter le roster quand il n'y a plus d'individus dans le ménage
if AUTRES_MEMBRES= 2 then
   endgroup;
endif
```

Avec cela, nous n'avons plus besoin d'utiliser le champ de contrôle d'occurrence du roster. Cependant, étant donné que les autres rosters (caractéristiques individuelles et éducation) dépendent toujours de la variable **NOMBRE_MEMBRES_MENAGE** en tant que champ de contrôle du roster, nous devons définir sa valeur après avoir complété la liste de noms. Nous pouvons en faire un champ protégé, le déplacer après la liste de noms et le définir à l'aide de la logique suivante :

```
PROC NOMBRE_MEMBRES_MENAGE
preproc
// Définit le nombre de membres du ménage en fonction de la liste des membres du
// ménage. Il est utilisé comme champ de contrôle d'occurrence pour les
// rosters qui suivent.
NOMBRE_MEMBRES_MENAGE = totocc(MEMBRES_MENAGE_ROSTER);
```

Ceci utilise la fonction totocc() qui donne le nombre total d'occurrences d'un enregistrement ou d'un élément répété. Autrement dit, cela nous donne le nombre total de lignes d'un roster.

Endlevel

Il existe également la commande endlevel qui est similaire à endgroup mais saute le reste du questionnaire en cours (à l'exception des applications à deux niveaux lorsque dans un nœud de deuxième niveau, la commande termine le nœud actuel).

Sauts avec les cases à cocher

Nous voulons seulement poser la question **B12**, langue des signes, si le membre du ménage a une déficience auditive. Nous souhaitons ignorer la question **B12** si la réponse “sourd” n’est pas cochée dans la question **B11**. Comment pouvons-nous dire en logique si l’option “sourd” est cochée ? “Sourd” est l'option B dans l'ensemble de valeurs. Nous devons donc déterminer si l'ensemble des options cochées contient la lettre B.

Dans la logique CSPro, nous pouvons utiliser la fonction pos() qui donne la position d'une chaîne de caractères dans une autre. Par exemple, pos("C", "CSPro") sera un, pos("P", "CSPro") sera trois et pos("o", "CSPro") sera cinq. Si la chaîne de recherche n'est pas trouvée, pos() retourne zéro. Dans notre cas, la lettre B n'est pas toujours à la même position dans la chaîne. Si l'intervieweur choisit uniquement l'option B, B sera le premier caractère de la chaîne, mais si l'intervieweur choisit A et B, le résultat sera "AB" et B sera en position deux. Cependant, tout ce qui nous importe est de savoir si B est dans la chaîne ou pas. Pour cela, nous pouvons vérifier le résultat de pos(). Si le résultat de pos() n’est pas zéro, alors B est dans la chaîne et si le résultat est zéro, B n'est pas dans la chaîne.

En utilisant pos(), la logique pour ignorer la langue des signes si “sourd” n'est pas cochée est la suivante:

```
PROC LANGUES_SIGNES
preproc
// ne pose cette question que si “sourd” (option B) n’est pas cochée,
ask if pos("B", HANDICAPS) > 0;
```

Notez que puisque nous ajoutons une préproc à une variable qui a déjà une postproc, nous devons ajouter explicitement le mot postproc.

```
PROC LANGUES_SIGNES
preproc
// ne pose cette question que si “sourd” (option B) n’est pas cochée,
ask if pos("B", HANDICAPS) > 0;
postproc
// Sauter à la personne suivante pour les membres du ménage âgés de moins
// de 10 ans
if AGE < 10 then
   skip to next;
endif;
```

Bien que cela fonctionne, cela introduit un problème avec le skip to next dans la postproc. Si nous avons un individu de moins de 10 ans qui n'est pas sourd, nous ne sautons plus la question de l'état matrimonial. En effet, ask if dans le preproc saut à la variable suivante et que la postproc n'est plus exécutée. Où pouvons-nous placer le skip to next afin que cela fonctionne pour ceux qui n'ont aucune difficulté à entendre ? Nous pouvons le déplacer dans la preproc de la variable **ETAT_MATRIMONIAL**.

```
PROC ETAT_MATRIMONIAL
preproc
// Sauter à la personne suivante pour les membres du ménage âgés de moins
// de 10 ans
if AGE < 10 then
   skip to next;
endif;
```

Exercices

1. Sautez la question **F07**, loyer mensuel, si le logement n’est pas loué en **F06**.
2. Dans la section E, décès, appliquez le saut de la question **E01**, décès dans les dernières 5 ans.
3. Dans la section E, décès, appliquez le saut pour la question **E08**, numéro de ligne de la mère du défunt, en fonction de l'âge au moment du décès **E07**.
4. Dans la section E, décès, appliquez le saut pour la question **E09**, décédé pendant la grossesse.
5. Dans la section E, décès, sauter les questions **E09** et **E10** pour les membres décédés du ménage qui ne sont PAS des femmes âgées de 12 à 50 ans.
6. Dans la section E, utilisez la logique pour pré-remplir le champ de numéro d’ordre **E03**. Assurez-vous que le numéro d’ordre est protégé.
7. Sautez toute la section C pour les individus âgés de moins de 3 ans.
8. Ajoutez une question supplémentaire à **F13** pour déterminer si le répondant souhaite donner la distance à l’eau en minutes ou en kilomètres. S'ils choisissent kilomètres, sautez la question de distance en minutes, sinon sautez la question de distance en km.
9. Ajoutez le champ autre (à préciser) et le saut à la question **F09**, matériau du mur.
10. Ajoutez le champ autre (à préciser) et le saut à la question **B15**, langues parlées.
11. Remplir automatiquement le code de bien pour la question **G01**.
12. Dans la question **G01**, biens du ménage, sautez le champ de valeur si la quantité est zéro.
13. Ajoutez des variables et un formulaire pour la section D: fécondité. Ajoutez les variables à l'enregistrement de líndividu, mais créez un nouveau formulaire avec son propre roster, comme nous l'avons fait pour la section C. Ne vous inquiétez pas de la dernière question du formulaire affichant le nombre total de naissances. Nous couvrirons cela dans une leçon ultérieure. Ajoutez les sauts pour la section de fécondité. N'oubliez pas de sauter la section entière pour les hommes et les femmes qui ne sont pas de 12 à 50 ans.

Session 03 : Contrôles de cohérence

À la fin de ce leçon les participants seront en mesure de :

- Utiliser la commande errmsg pour afficher des messages à l'intervieweur
- Utiliser la commande errmsg avec la clause select
- Utiliser if then else pour les contrôles de cohérence entre plusieurs variables
- Utiliser la commande warning pour les contrôles non-bloquants
- Créer et exécuter des plans de vérifications de contrôles de cohérence
- Créer des contrôles de cohérence avec les dates
- Déclarer et utiliser des variables de logique

Contrôles de cohérence à deux variables

Jusqu'à présent, nous avons été en mesure de limiter les réponses de chaque variable à un ensemble de réponses valides en utilisant des ensembles de valeurs. Et si nous voulions vérifier la cohérence entre deux variables différentes ?

Par exemple, empêchons l'intervieweur de saisir plus de chambres à coucher qu'il y a pièces dans la maison. Cliquez sur le champ **NOMBRE_CHAMBRES_COUCHER** dans l'arborescence des formulaires, puis cliquez sur logique dans la barre d'outils pour ouvrir l'éditeur de logique. Ajoutez le code suivant dans la proc de **NOMBRE_CHAMBRES_COUCHER** :

```
PROC NOMBRE_CHAMBRES_COUCHER

// Assurez que le nombre de chambres à coucher ne dépasse pas le nombre de
// chambres.
if NOMBRE_CHAMBRES_COUCHER > NOMBRE_PIECES then
   errmsg("Le nombre de chambres à coucher ne peut pas dépasser le nombre de
pièces");
   reenter;
endif;
```

L'instruction if exécute le code entre then et endif seulement si la condition (NOMBRE_CHAMBRES_COUCHER supérieur à NOMBRE_PIECES) est vraie. L'instruction errmsg affichera un message à l'utilisateur. La commande reenter oblige l'intervieweur à rester dans le champ actuel et ne lui permet pas de passer au champ suivant.

Notez que nous plaçons notre logique dans la proc de **NOMBRE_CHAMBRES_COUCHER** et pas dans la proc de **NOMBRE_PIECES**. En effet, lorsque nous sommes dans la proc de **NOMBRE_PIECES**, la valeur de **NOMBRE_CHAMBRES_COUCHER** n’a pas encore été saisie. Lors de la création de contrôles de cohérence, nous plaçons toujours la logique du contrôle dans la proc pour le dernier champ impliqué dans le contrôle.

Prenons un autre exemple. Les spécifications de contrôle demandent le suivant :

> • Si **LIEN_PARENTE** est épouse/époux du chef de ménage, **ETAT_MATRIMONIAL** ne doit pas être codé 1 (jamais marié), 2 (divorcé) ou 3 (veuf).

Dans quelle proc devrions-nous mettre cette vérification ? Puisque **ETAT_MATRIMONIAL** vient après **LIEN_PARENTE,** nous le mettrons dans la proc de la variable **ETAT_MATRIMONIAL**.

```
PROC ETAT_MATRIMONIAL

// Assurez que le conjoint du CM n'est pas célibataire, divorcé ou veuf
if (ETAT_MATRIMONIAL = 1 or ETAT_MATRIMONIAL= 3 or
   ETAT_MATRIMONIAL= 4) and LIEN_PARENTE = 2 then
   errmsg("L'état matrimoniale du conjoint ne peut être célibataire, divorcée ou
veuve");
   reenter;
endif;
```

Notez que lorsque vous combinez plusieurs expressions avec "and" et "or", les expressions "and" sont évaluées en premier, puis les expressions "or" sont évaluées, ce qui peut parfois conduire à des résultats inattendus. Vous pouvez utiliser des parenthèses pour forcer l'ordre d'évaluation que vous souhaitez, comme nous le faisons ci-dessus. Qu'advient-il de l'expression ci-dessus sans les parenthèses si la relation n'est pas 2 et que l'état matrimonial est 3 ?

Bien sûr, il existe des manières plus simples d’écrire cette vérification sans utiliser "or". Nous pouvons utiliser l'opérateur différent de ("<>") :

```
// Assurez-vous que le conjoint n'est pas célibataire
if ETAT_MATRIMONIAL<> 2 and LIEN_PARENTE = 2 then
   errmsg("L'état matrimoniale du conjoint ne peut être célibataire, divorcée ou
veuve");
   reenter;
endif;
```

Ou nous pouvons utiliser l’opérateur "in" :

```
// Assurez que le conjoint n'est pas célibataire
if ETAT_MATRIMONIAL in 1,3,4 and LIEN_PARENTE = 2 then
   errmsg("L'état matrimoniale du conjoint ne peut être célibataire, divorcée ou
veuve");
   reenter;
endif;
```

L'expression avec l'opérateur "in" sera vraie si la valeur correspond à l'un des nombres de la liste séparée par des virgules. Il prend également en charge les plages en séparant le début et la fin de la plage par deux points (:). Par exemple, **CHAMBRES_COUCHER** dans 1:4 sera vrai si **CHAMBRES_COUCHER** est égal à 1,2,3 ou 4.

Messages d'erreur améliorés

Nous pouvons améliorer notre message d'erreur en ajoutant des paramètres à la chaîne. Revenons à la vérification du nombre de chambres et affichons le nombre de chambres à coucher et le nombre de pièces dans le message :

```
PROC NOMBRE_CHAMBRES_COUCHER

// Assurez que le nombre de chambres à coucher ne dépasse pas le nombre de
// chambres.
if NOMBRE_CHAMBRES_COUCHER > NOMBRE_PIECES then
   errmsg("Nombre de chambres à coucher,%d, dépasse le nombre de pièces,%d",
         NOMBRE_CHAMBRES_COUCHER, NOMBRE_PIECES );
   reenter;
endif;
```

Les "%d" dans le message sont remplacés par les valeurs des variables qui suivent dans l'ordre dans lequel elles sont répertoriées.

Faisons de même pour le contrôle de l'état matrimonial et le lien de parenté :

```
// Assurez que le conjoint n'est pas célibataire
if ETAT_MATRIMONIAL in 1,3,4 and LIEN_PARENTE = 2 then
   errmsg("Le lien de parenté est le conjoint et l'état matrimonial est %d. Le conjoint
ne peut pas être célibataire, divorcé ou veuf. ", ETAT_MATRIMONIAL);
   reenter;
endif;
```

Il serait préférable que nous puissions inclure le nom de la personne dans le message d'erreur en tant qu'une information supplémentaire pour l'intervieweur. Comme le nom est une variable alpha, nous utilisons "%s" au lieu de "%d". (%d correspond à la valeur numérique seulement)

```
// Assurez que le conjoint n'est pas célibataire
if ETAT_MATRIMONIAL in 1,3,4 and LIEN_PARENTE = 2 then
   errmsg("%s a lien de parenté conjoint et l'état matrimonial est %d. Le conjoint ne
peut pas être célibataire, divorcé ou veuf. ", NOM, ETAT_MATRIMONIAL);
   reenter;
endif;
```

Pourquoi y a-t-il beaucoup d'espace supplémentaire après le nom dans le message d'erreur ? Rappelez-vous que les variables alpha dans le dictionnaire ont une longueur fixe. Cela signifie qu'ils sont remplis d'espaces vides. Nous pouvons supprimer les espaces vides à la fin de la variable en utilisant la fonction strip().

```
// Assurez que le conjoint n'est pas célibataire
if ETAT_MATRIMONIAL in 1,3,4 and LIEN_PARENTE = 2 then
   errmsg("%s a lien de parenté conjoint et l'état matrimonial est %d. Le conjoint ne
peut pas être célibataire, divorcé ou veuf.", strip(NOM), ETAT_MATRIMONIAL);
   reenter;
endif;
```

Enfin, plutôt que d'afficher la valeur numérique de l'état matrimonial, il serait préférable d'afficher le libellé la valeur saisie. Nous pouvons faire cela en utilisant "%l" au lieu de "%d". Cela affichera l'étiquette de la valeur définie plutôt que le code numérique.

```
// Assurez que le conjoint n'est pas célibataire
if ETAT_MATRIMONIAL in 1,3,4 and LIEN_PARENTE = 2 then
   errmsg("%s a lien de parenté conjoint et l'état matrimonial est %l. Le conjoint ne
peut pas être célibataire, divorcé ou veuf.", strip(NOM), ETAT_MATRIMONIAL);
   reenter;
endif;
```

Errmsg avec select

Si nous ajoutons la clause select à la commande errmsg, nous pouvons donner à l'utilisateur la possibilité de corriger l'un ou l'autre des champs impliqués dans l'incohérence. Nous pouvons utiliser ceci dans notre exemple de chambres :

```
PROC NOMBRE_CHAMBRES_COUCHER

// Assurez que le nombre de chambres à coucher ne dépasse pas le nombre de
// chambres.
if NOMBRE_CHAMBRES_COUCHER > NOMBRE_PIECES then
   errmsg("Nombre de chambres à coucher, %d, dépasse le nombre de pièces,%d",
         NOMBRE_CHAMBRES_COUCHER, NOMBRE_PIECES )
         select("Corriger le nombre de pièces", NOMBRE_PIECES ,
               "Corriger le nombre de chambres",
NOMBRE_CHAMBRES_COUCHER);
endif;
```

Avec le select on n'a plus besoin du reenter car CSEntry fera automatiquement le reenter au champ que l'intervieweur choisit.

Notez que la clause select fait partie de la commande errmsg. Il n'y a donc pas de point-virgule entre le select et le errmsg.

Ajoutons également une sélection à la vérification de l'état matrimonial :

```
// Assurez que le conjoint n'est pas célibataire
if ETAT_MATRIMONIAL in 1,3,4 and LIEN_PARENTE = 2 then
   errmsg("%s a lien de parenté conjoint et l'état matrimonial est %l. Le conjoint ne
peut pas être célibataire, divorcé ou veuf. ", strip(NOM), ETAT_MATRIMONIAL)
        select("Corriger l'état matrimonial", ETAT_MATRIMONIAL,
                "Corriger le lien de parenté", LIEN_PARENTE);
endif;
```

Contrôles non-bloquants

Si vous souhaitez autoriser l'utilisateur à ignorer l'erreur et continuer au champ suivant, vous pouvez ajouter une option à l'instruction select avec le mot clé continue :

```
PROC NOMBRE_CHAMBRES_COUCHER

// Assurez que le nombre de chambres à coucher ne dépasse pas le nombre de
// chambres.
if NOMBRE_CHAMBRES_COUCHER > NOMBRE_PIECES then
   errmsg("Nombre de chambres à coucher,%d, dépasse le nombre de pièces,%d",
        NOMBRE_CHAMBRES_COUCHER, NOMBRE_PIECES )
        select("Corriger le nombre de pièces", NOMBRE_PIECES ,
                "Corriger le nombre de chambres",
NOMBRE_CHAMBRES_COUCHER,
                "Ignorer l'erreur", continue);
endif;
```

Maintenant, la boîte de dialogue de message aura un troisième bouton intitulé "ignorer l'erreur" qui, lorsque vous cliquez dessus, passera au champ suivant, dans ce cas au type de logement.

C'est ce que l'on appelle un contrôle "non-bloquant". Les autres contrôles sont dits “bloquants” car ils ne permettent pas à l'intervieweur de continuer tant que l'incohérence n'est pas corrigée. L'avantage d'un contrôle non-bloquant c’est que l'intervieweur ne reste pas bloqué. Cependant, la qualité des données peut en souffrir. Dans un recensement CAPI, les contrôles non obligatoires sont généralement préférés afin d’empêcher les intervieweurs de rester bloqués et de réduire au minimum la durée des interviews.

CSPro fournit une fonction d’avertissement, warning, pour les contrôles non-bloquants. La commande warning est identique à errmsg sauf que si vous avez déjà ignoré le message une fois, lorsque vous naviguez dans le champ en cliquant sur l’arborescence ou en sortant de la sauvegarde partielle, le message ne s’affiche pas une deuxième fois. Pour que les pièces / chambres soient un contrôle non-bloquant, en plus d’ajouter l’option “ignorer l’erreur”, nous utiliserons warning au lieu de errmsg.

```
PROC NOMBRE_CHAMBRES_COUCHER

// Assurez que le nombre de chambres à coucher ne dépasse pas le nombre de
// chambres.
if NOMBRE_CHAMBRES_COUCHER > NOMBRE_PIECES then
   warning("Nombre de chambres à coucher, %d, dépasse le nombre de pièces,
%d",
         NOMBRE_CHAMBRES_COUCHER, NOMBRE_PIECES )
         select("Corriger le nombre de pièces", NOMBRE_PIECES ,
               "Corriger le nombre de chambres",
                  NOMBRE_CHAMBRES_COUCHER,
               "Ignorer l'erreur", continue);
endif;
```

Vérification des contrôles de cohérence

Lors de la vérification des contrôles de cohérence impliquant plusieurs variables, il est important de tester toutes les combinaisons possibles de ces variables. Pour le faire, nous pouvons créer une matrice de vérification. Par exemple, pour vérifier toutes les combinaisons possibles de notre vérification de cohérence entre **ETAT_MATRIMONIAL** et **LIEN_PARENTE,** nous utiliserons la matrice suivante qui indique le résultat attendu pour chaque combinaison des variables impliquées :

		État matrimonial		
Lien	Jamais marié (1)	Marié (2)	Divorcé (3)	Veuf (4)
Conjoint (2)	Erreur	OK	Erreur	Erreur
Non-conjoint (<> 2)	OK	OK	OK	OK

En utilisant cette matrice, vous pouvez tester chacune des 8 combinaisons possibles et vous assurer d'obtenir le résultat attendu. Cela garantit que tous les cas possibles ont été testés. En réunissant les matrices de vérification de tous les contrôles de cohérence de l'application, vous pouvez créer un plan de vérification pour l'application afin de vérifier que l'application fonctionne correctement lorsque des modifications sont apportées. Souvent, les modifications apportées à une partie du questionnaire peuvent avoir des effets inattendus sur les sauts et les contrôles de cohérence dans d’autres parties du questionnaire. Il est donc important de disposer d'un plan de vérification qu’on utilise après chaque série de modifications pour éviter tout problème.

Calculs

Dans la question **D02,** le répondant donne le nombre de garçons vivant dans le ménage, le nombre de filles vivant dans le ménage et le nombre total d'enfants vivant dans le ménage. Vérifions que la somme des filles et des garçons est égale au total.

```
if GARCONS_MENAGE + FILLES_MENAGE <> ENFANTS_MENAGE then
   errmsg("Nombre total d'enfants (%d) doit être égale à la somme du nombre de
garçons (%d) et le nombre de filles (%d)",
         ENFANTS_MENAGE , GARCONS_MENAGE , FILLES_MENAGE)
      select("Corriger le total", ENFANTS_MENAGE ,
            "Corriger le nombre de garçons", GARCONS_MENAGE ,
            "Corriger le nombre de filles", FILLES_MENAGE);
endif;
```

Les opérateurs mathématiques suivants sont permis dans la logique de CSPro :

Addition	+
Soustraction	-
Multiplication	*
Division	/
Modulo (reste)	%
Puissance (exposant)	^

Contrôles de cohérence avec les dates

Ajoutons un contrôle de cohérence entre la date de naissance et l'âge. Nous pouvons utiliser la fonction datediff() pour calculer l'âge en fonction de la date de l'interview et la date de naissance. Dans quelle proc devrions-nous mettre ce contrôle ? Nous devons avoir l'âge et la date de naissance complète (année, mois et jour) avant de pouvoir effectuer le contrôle. Nous pourrions donc faire cette vérification dans la postproc du dernier composant de date ou dans la postproc du bloc contenant tous les champs de date. La postproc du bloc est exécutée une fois que tous les champs du bloc ont été saisies. Sur mobile, lorsque les champs de bloc sont affichés sur le même écran, les procs des champs du bloc ne sont pas exécutés tant que l'utilisateur n'a pas terminé le bloc en entier et n'est passé à l'écran suivant. Pour assurer un comportement cohérent entre Windows et Android, il est donc recommandé de mettre tout le code de validation dans la postproc du bloc.

```
PROC DATE_NAISSAINCE_BLOCK

// Assurez que l'âge correspond à la date de naissance
if datediff(DATE_NAISSAINCE, DATE_INTERVIEW, "y") <> AGE then
   errmsg("L'âge (%d) ne correspond pas à la date de naissance (%d)",
            AGE, DATE_NAISSAINCE)
      select("Corriger l'âge", AGE,
               "Corriger la date de naissance", DATE_NAISSAINCE);
endif;
```

Cela fonctionne, mais il serait préférable que le message d'erreur informe l'intervieweur de la date de naissance calculée

```
PROC DATE_NAISSAINCE_BLOCK

// Assurez que l'âge correspond à la date de naissance
if datediff(DATE_NAISSAINCE, DATE_INTERVIEW, "y") <> AGE then
   errmsg("L'âge (%d) ne correspond pas à la date de naissance (%d). Basé sur la
date de naissance, l'âge doit être %d",
            AGE, DATE_NAISSAINCE,
            datediff(DATE_NAISSAINCE, DATE_INTERVIEW, "y") )
      select("Corriger l'âge", AGE,
               "Corriger la date de naissance", DATE_NAISSAINCE);
endif;
```

Nous devrions également traiter le cas où l'âge ou la date de naissance est inconnu.

```
PROC DATE_NAISSAINCE_BLOCK

// Assurez que l'âge correspond à la date de naissance
if AGE <> 999 et DATE_NAISSAINCE <> 99999999 then
   if datediff(DATE_NAISSAINCE, DATE_INTERVIEW, "y") <> AGE then
      errmsg("L'âge (%d) ne correspond pas à la date de naissance (%d). Basé sur la
date de naissance, l'âge doit être %d",
            AGE, DATE_NAISSAINCE,
            datediff(DATE_NAISSAINCE, DATE_INTERVIEW, "y") )
         select("Corriger l'âge", AGE,
                  "Corriger la date de naissance", DATE_NAISSAINCE);
   endif;
endif;
```

S'assurer d'avoir le nombre correct de chiffres “9” est un peu délicat. Comme nous avons utilisé la valeur spéciale missing pour 999 dans la valeur définie pour l'âge, nous pouvons utiliser plutôt missing dans notre instruction if. Nous pouvons faire la même chose pour la date de naissance.

```
PROC DATE_NAISSAINCE_BLOCK

// Assurez que l'âge correspond à la date de naissance
if AGE <> missing et DATE_NAISSAINCE <> missing then
   if datediff(DATE_NAISSAINCE, DATE_INTERVIEW, "y") <> AGE then
      errmsg("L'âge (%d) ne correspond pas à la date de naissance (%d). Basé sur la
de naissance, l'âge doit être %d",
            AGE, DATE_NAISSAINCE,
            datediff(DATE_NAISSAINCE, DATE_INTERVIEW, "y") )
         select("Corriger l'âge", AGE,
                "Corriger la date de naissance", DATE_NAISSAINCE);
   endif;
endif;
```

Cela rend non seulement notre code plus clair, mais également plus résistant aux modifications, car la logique fonctionnerait encore si nous changions ultérieurement la taille de la variable AGE de trois chiffres à deux.

Variables de logique

Il serait préférable que nous ne répétions pas deux fois le calcul du datediff. La répétition du code rend plus difficile sa maintenance. Nous pouvons corriger un bogue dans une copie du code mais oublier de le faire dans une autre. Pour éviter de nous répéter, nous pouvons déclarer une variable de logique contenant la valeur de notre calcul. Les variables de logique ressemblent aux variables du dictionnaire mais ne sont utilisées que dans les calculs logiques et ne sont pas affichées dans les formulaires ni enregistrées dans le fichier de données. Pour créer une variable de logique, nous la déclarons comme suit :

```
numeric unNombre;
string unAlphanumeric;
alpha(20) unAlphaNumericLonguerFixe;
```

Les variables du type numérique contiennent des nombres (y compris les nombres avec des fractions) et les variables du type string contiennent des valeurs alphanumériques (les chaînes de caractères). Les variables alpha ressemblent à des strings mais ont une longueur fixe. Dans les premières versions de CSPro, les variables de type string n'existaient pas et il était courant d'utiliser des variables du type alpha. Dans CSPro moderne, vous devriez toujours utiliser des variables de type string au lieu de variables alpha de longueur fixe.

Style de codage

Nous vous recommandons d'utiliser une casse mixte (cas de chameau) pour les variables de logique afin de les distinguer des variables du dictionnaire qui seront tout en majuscule.

Nous pouvons déclarer une variable qui contient le résultat du datediff de la manière suivante :

```
numeric ageCalcule;
ageCalcule = datediff(DATE_NAISSAINCE, DATE_INTERVIEW, "y");
```

Nous pouvons maintenant utiliser cette variable à la place de datediff:

```
if ageCalcule <> AGE then
    errmsg("L'âge (%d) ne correspond pas à la date de naissance (%d). Basé sur la
date de naissance, l'âge doit être %d",
            AGE, DATE_NAISSAINCE, ageCalcule)
        select("Corriger l'âge", AGE,
                "Corriger la date de naissance", DATE_NAISSAINCE);
  endif;
```

Il est possible de mettre la déclaration et l'initialisation de la variable en une seule ligne :

```
numeric ageCalcule = datediff(DATE_NAISSAINCE, DATE_INTERVIEW, "y");
```

Quand est-ce qu'on utilise les contrôles de cohérence ?

Il est tentant de placer des contrôles de cohérence sur chaque question de votre application afin d'optimiser la qualité des données, mais il est important de savoir que chaque contrôle ajouté à son prix. Au-delà du temps nécessaire pour mettre en œuvre et vérifier correctement chaque contrôle, vous devez également prendre en compte le temps supplémentaire nécessaire pour former les agents et prendre en charge une application plus complexe. Chaque vérification que vous ajoutez augmente le risque que votre application contient des bogues lorsqu'elle se rend sur le terrain. Contrairement à collecte sur papier, un problème de l'application sur le terrain peut signifier que l'agent est bloqué et abandonne un questionnaire entier. Pour les enquêtes, vous avez généralement un petit échantillon et des enquêteurs mieux formés, ce qui est moins préoccupant. Cependant, pour un recensement, les enquêteurs ne sont pas aussi expérimentés et leur soutien sur le terrain est beaucoup plus difficile. Dans un recensement, en raison du nombre de répondants, vous pouvez utiliser l'imputation pour corriger de nombreuses incohérences avec un impact minimal sur les résultats globaux. Dans tous les cas, le simple fait que l'instrument CAPI impose des contrôles de plage et des sauts automatiques constitue une amélioration significative de la qualité des données par rapport à un recensement sur papier. Pour une application de recensement CAPI, nous vous recommandons d'ajouter principalement des contrôles de cohérence sur les champs démographiques clés de la liste des membres du ménage et de corriger les incohérences potentielles dans d'autres parties du questionnaire après la collecte des données. De plus, nous recommandons d'utiliser principalement des contrôles non-bloquants pour que l'agent ne soit jamais bloqué par l'application.

Exercices

Réaliser les contrôles de cohérence suivants à partir des spécifications dans le document de spécifications du questionnaire. Vérifiez qu'ils sont corrects. Assurez-vous de tester tous les cas possibles (les matrices de test peuvent vous aider à cela).

1. Affichez un message d'erreur si le chef de ménage est âgé de moins de 12 ans.
2. Assurez que **B14** (âge au premier mariage) est inférieur ou égal à l'âge actuel (**B05**).
3. Affiche un message d'erreur si le niveau d'instruction (**C02**) est licence ou maîtrise/doctorat, mais que l'âge est inférieur à dix-huit. Faites-en un contrôle non-bloquants afin que l'intervieweur puisse ignorer l'erreur s'il le souhaite.
4. Ajoutez un message d'erreur si le niveau d'instruction (**C02**) est supérieur ou égale à «Élémentaires 5» et si la personne ne sait ni lire ni écrire (**C03**). Faites-en un contrôle non-bloquant.
5. Ajoutez un contrôle de cohérence dans la section **F** qui affiche un message d'erreur si le ménage dispose d'une toilette à chasse d'eau à la question **F11** et NE spécifie PAS également le robinet à l'intérieur de la maison pour la question **F12**. Utilisez une clause de “select” pour leur permettre de corriger soit **F11** ou **F12**.
6. Assurez que le nombre total d'enfants à **D03** est égal à la somme des garçons et des filles. Faites de même pour la question **D04**.
7. Vérifiez que la date à laquelle le décès est survenu dans la question **E05** est une date dans les cinq dernières années, car la section ne devrait inclure que les décès survenus pendant cette période. Utilisez la date de l'interview pour calculer le nombre d'années écoulées depuis le décès. Par exemple, si la date de l'interview est le 2019-07-10 et la date du décès est le 2012-07 ou le 2014-06, vous devez afficher un message d'erreur. Vous pouvez utiliser la fonction datediff pour calculer la différence entre la date de l’interview et la date du décès, comme nous l'avions fait pour la vérification de l'âge et de la date de naissance. Toutefois, vous devrez définir une date complète à partir du mois et de l'année du décès. Comme aucun jour n'est donné, utilisez 1 comme jour (supposer le premier jour du mois). Notez que la date est arrondie à l’année la plus proche. Pour effectuer le contrôle correctement, vous devez obtenir la différence en mois et la comparer à 60 (5 ans fois 12 mois).

Session 04 : Les rosters, les indices et les boucles

À la fin de cette leçon, les participants seront en mesure de :

- Mettre en œuvre des contrôles de cohérence comprenant plusieurs membres du ménage
- Comprendre l'ordre des PROC et l'endroit où placer les contrôles des données dans les rosters
- Utiliser des indices, totocc(), count(), seek() et curocc() avec les rosters
- Utiliser des boucles (do while) pour mettre en œuvre des contrôles sur les enregistrements / éléments répétitifs (rosters).
- Comprendre et utiliser des "valeurs spéciales" (notappl, missing et default) dans la logique
- Définir les libellés d'occurrence sur les rosters à partir de la logique

Les groupes et les indices

Ajoutons un contrôle au numéro de ligne de la mère pour s'assurer que le numéro de ligne entré correspond à une femme de 12 ans ou plus dans le ménage. Pour ce faire, nous devons relier le numéro de ligne entré dans **B10** à la ligne correspondante du roster dans la section B. Ceci est effectué via des indices pour obtenir l'âge et le sexe de la mère sur la base du numéro de ligne.

Pour tout élément répété, l'ajout d'un nombre entre parenthèses après le nom de la variable nous donne la valeur d'une occurrence particulière de la variable. Par exemple, AGE(1) sera l'âge du premier membre du ménage, AGE(2) l'âge du deuxième membre… Si nous omettons l'indice lors de l'exécution de la logique dans un PROC d'un roster, CSPro supposera que nous voulons celui pour la ligne actuelle. Cependant, dans ce cas, nous ne voulons pas le sexe et l'âge de la personne de la ligne en cours, nous voulons l'âge et le sexe de la ligne spécifiée par **NUMERO_ORDRE_MERE**. Pour cela, nous devons utiliser AGE(NUMERO_ORDRE_MERE) et SEXE(NUMERO_ORDRE_MERE).

```
PROC NUMERO_ORDRE_MERE

// Assurez que le numéro de ligne de la mère est le numéro de ligne d'une femme
// de plus de 12 ans
if not NUMERO_ORDRE_MERE in 87:88 then
   numeric sexeMere = SEXE(NUMERO_ORDRE_MERE);
   numeric ageMerge = AGE(NUMERO_ORDRE_MERE);
   numeric nomMere = strip(NOM(NUMERO_ORDRE_MERE));
   if sexeMere  = 1 then
      errmsg("Sexe de la mère %s ne doit pas être masculin", nomMere)
         select("Corriger numéro d'ordre de la mère",  NUMERO_ORDRE_MERE,
               "Corriger le sexe de" + nomMere,
               SEXE(NUMERO_ORDRE_MERE));
   endif;

   if ageMere <> missing and ageMere < 12 then
      errmsg("L'âge de la mère %s est de %d mais doit être au moins 12",
         nomMere, ageMere)
         select("Corriger numéro d'ordre de la mère",  NUMERO_ORDRE_MERE,
               "Corriger l'âge de" + nomMere),
                AGE(NUMERO_ORDRE_MERE));
   endif;
endif;
```

Blancs et valeurs spéciales

Que se passe-t-il si nous faisons référence à l'occurrence d'une ligne dans le roster de la section B qui n'existe pas ? Essayez de saisir un numéro de ligne pour la mère supérieure au nombre de lignes du roster de la section B. Nos contrôles d'âge et de sexe ne sont pas déclenchés. Quelles sont les valeurs de **NOM**, **SEXE** et **AGE** lorsqu'elles sont vides ? Affichons-les en utilisant errmsg et voyons.

```
errmsg("NOM= %s, AGE= %d, SEXE= %d", strip(NOM(NUMERO_ORDRE_MERE)),
      AGE(NUMERO_ORDRE_MERE), SEXE(NUMERO_ORDRE_MERE));
```

La variable alphanumérique **NOM** est simplement vide mais les valeurs numériques **AGE** et **SEXE** sont "notappl". Qu'est-ce que ça veut dire ? En générale, les champs numériques vides (ignorés ou non encore saisis) ont une valeur spéciale appelée notappl qui peut être utilisée dans les comparaisons en logique. Par exemple, pour vérifier si le **SEXE** est blanc, nous pouvons utiliser la comparaison suivante :

```
if SEXE(NUMERO_ORDRE_MERE) = notappl then
   // le sexe est vide, cette ligne doit être vide dans le roster de la section B
   errmsg("%d n'est pas une ligne valide dans la section B", NUMERO_ORDRE_MERE);
   reenter;
endif;
```

Il existe d'autres valeurs spéciales utilisées dans CSPro :

- Missing : utilisé comme alias pour les codes sans réponse / refusés (9, 99, 999…). Vous devez créer une entrée pour cela dans l'ensemble de valeurs.
- Default : résultat d'une erreur de lecture d'un fichier de données ou d'une erreur de calcul (comme essayer de calculer notappl + 2 ou d'essayer d'effectuer une division par zéro) ou de stocker une valeur dans une variable trop petite pour la contenir. Par exemple, si AGE est un champ à deux chiffres, AGE = 118 donnera la valeur default.

Obtenir la taille d'un roster

Dans notre message d'erreur, il serait bien de dire à l'intervieweur quel est le numéro de ligne maximum valide. Nous pouvons le faire en utilisant la fonction totocc() qui vous donne le nombre total d'occurrences d'un groupe.

```
if NUMERO_ORDRE_MERE > totocc(MEMBRES_MENAGE_ROSTER) then
    // Ceci est au-delà de la fin du roster
    errmsg("%d n'est pas une ligne valide dans la section B. Doit être comprise entre 1 et
%d.",
        NUMERO_ORDRE_MERE, totocc(MEMBRES_MENAGE_ROSTER));
    reenter;
endif;
```

Définition des libellés d'occurrence dans la logique

Si vous examinez l'arborescence du cas sous Android, vous verrez que les occurrences du roster sont affichées avec le libellé du roster et le numéro d'occurrence : "Caractéristiques des individus(1), Caractéristiques des individus(2) ...", ce qui n'est pas utile. En utilisant la logique, nous pouvons définir les libellés d'occurrence comme les noms des individus. Pour cela, nous utilisons la commande setocclabel() qui prend le nom du groupe (le roster ou le formulaire répété) et la chaîne de caractères à définir. Par exemple, pour définir le libellé d'occurrence de chaque ligne des différents rosters une fois que le nom soit saisie, nous pouvons procéder comme suit dans le postproc du champ **NOM** :

```
PROC NOM
setocclabel(MEMBRES_MENAGE_ROSTER(curocc()), strip(NOM)));
setocclabel(CARACTERISTIQUES_INDIVIDU_ROSTER(curocc()), strip(NOM));
setocclabel(EDUCATION_ROSTER(curocc()), strip(NOM));
setocclabel(FECONDITE_ROSTER(curocc()), strip(NOM));
```

Cela fonctionne bien lorsque nous ajoutons un nouveau cas. Toutefois, si nous ouvrons un cas existant en mode modification, les étiquettes d'occurrence ne sont pas définies tant que nous n'arrivons pas au roster des membres du ménage, même si les occurrences existent déjà. En mode de modification, tout en restant sur le roster des membres du ménage, vous pouvez faire défiler l'arborescence de cas pour afficher "Caractéristiques des individus(1),

Caractéristiques des individus(2) …” comme nous le faisions auparavant. Pour éviter cela, nous devons définir les libellés d’occurrence pour les rosters dès que nous ouvrons le cas.

En savoir plus sur les procs

Quel proc peut-on utiliser pour définir les libellés d'occurrence ? Nous pourrions utiliser le preproc du premier champ du première formulaire mais il y a une meilleure option. Il s'avère que chaque élément de l'arborescence de formulaire a sa propre PROC. Non seulement les variables ont des procs, mais il existe des procs pour les formulaires, les rosters, les niveaux et l'application elle-même. Tout ce que vous voyez dans l’arborescence de formulaires peut avoir une preproc et une postproc. Comprendre l'ordre dans lequel les procs sont exécutées est important pour comprendre la logique CSPro.

La règle générale est que :

1. Les éléments parent ont leur preproc exécutées en premier
2. Puis les procs des éléments enfants sont exécutées
3. Enfin, les postprocs des parents sont exécutées

Travail de groupe

Formez des équipes de trois à cinq personnes. Le formateur ajoute errmsg au postproc et preproc des éléments suivants : **POPSTAN2020_FF** (application), **POPSTAN2020_QUEST** (niveau), **MEMBRES_MENAGE_FORM** (formulaire), **MEMBRES_MENAGE_ROSTER** (liste), **NOM** (variable), **NOMBRE_PIECES** (variable). Chaque équipe reçoit des bouts de papier avec les noms de chaque preproc et postproc. AVANT d'exécuter l'application, les équipes doivent classer les feuillets dans l'ordre dans lequel les messages issus des procs seront affichés lors de l'exécution de l'application. Les équipes ont trois minutes pour compléter l'exercice. Ensuite, l'application est exécutée et les équipes voient si elles obtiennent les bons résultats.

Définir les libellés d'occurrence (deuxième essai)

Maintenant que nous comprenons bien l’ordre d'exécution des procs, dans quelle proc devrions-nous définir les libellés d'occurrence ? La preproc du questionnaire, bien sûr ! Nous pouvons faire quelque chose comme ce qui suit :

```
PROC POPSTAN2020_QUEST
preproc
setocclabel(MEMBRES_MENAGE_ROSTER(1), strip(NOM(1)));
setocclabel(MEMBRES_MENAGE_ROSTER(2), strip(NOM(2)));
setocclabel(MEMBRES_MENAGE_ROSTER(3), strip(NOM(3)));
```

Les Boucles

Le problème est que nous voulons faire setocclabel pour chaque membre du ménage, mais le nombre de membres du ménage varie d'un cas à l'autre. Ce qui précède ne fonctionne que si nous connaissons la taille du ménage à l'avance. Afin de gérer les ménages de toute taille, nous avons besoin d'une boucle. Nous pouvons utiliser une boucle do qui vous permet de répéter quelque chose pendant qu'une certaine condition soit vraie. Combien de fois devons-nous bouclé ? Nous utilisons la fonction totocc() qui nous donne le nombre total d'occurrences du roster.

```
PROC POPSTAN2020_QUEST
preproc

// Remplissez les libellés d'occurrence dans les rosters lors de la saisie d'un cas
// qui a déjà des données (mode de sauvegarde partielle ou modification).
// Si nous ne le faisons pas, alors l'arborescence de cas n'aura pas
// d'étiquettes d'occurrences correctes jusqu'à ce que nous passions par le roster
// des membres du ménage.
do numeric i = 1 while i <= totocc(MEMBRES_MENAGE_ROSTER)
   setocclabel(MEMBRES_MENAGE_ROSTER(i), strip(NOM(i)));
   setocclabel(CARACTERISTIQUES_INDIVIDU_ROSTER(i), strip(NOM(i)));
   setocclabel(EDUCATION_ROSTER(i), strip(NOM(i)));
   setocclabel(FECONDITE_ROSTER(i), strip(NOM(i)));
enddo;
```

Compter les chefs de ménage

Il est actuellement possible de saisir plusieurs chefs de ménage ou bien aucun chef de ménage. Comment pouvons-nous ajouter un contrôle de cohérence pour s'assurer qu'il y a exactement un chef de ménage ? Dans quelle procédure un tel contrôle irait-il ? Nous pouvons le mettre dans la postproc du roster puisque celle-ci sera exécutée après que toutes les lignes aient été saisies.
Comment détermine-t-on le nombre de chefs de ménage ? Dans chaque ligne de la liste, nous pouvons vérifier le lien de parenté pour voir si c'est le chef de ménage. Pour ce faire, nous avons besoin d'une variable de logique permettant de suivre le nombre de chefs de ménage.

```
numeric nombreChef = 0;
```

Maintenant, nous pouvons incrémenter la variable pour chaque chef de ménage que nous trouvons :

```
PROC MEMBRES_MENAGE_ROSTER

// Une fois que la liste de personnes est complète, vérifiez qu'il y a
// un chef de ménage.
numeric nombreChefs = 0;

if LIEN_PARENTE(1) = 1 then
   nombreChefs = nombreChefs + 1;
endif;

if LIEN_PARENTE(2) = 1 then
   nombreChefs = nombreChefs + 1;
endif;

if LIEN_PARENTE(3) = 1 then
   nombreChefs = nombreChefs + 1;
endif;

if nombreChefs <> 1 then
   errmsg("Le nombre de chefs de ménage doit être exactement un");
   reenter LIEN_PARENTE(1);
endif;
```

Comme il y a plus d'une ligne dans le roster, nous devons utiliser un indice pour indiquer à CSPro quelle ligne du roster consulter. LIEN_PARENTE(3) fait référence au lien de la 3e ligne du roster.

Ce qui précède ne fonctionne que si nous connaissons la taille du ménage à l'avance. Afin de gérer les ménages de toute taille, nous avons besoin d'une boucle. Nous pouvons utiliser une boucle do. Comme avec les libellés d'occurrence, nous pouvons boucler de 1 au nombre de lignes du roster.

```
PROC MEMBRES_MENAGE_ROSTER

// Une fois que la liste des membres est au complète, assurez-vous qu'il y a
// un chef de ménage exactement.
numeric nombreChefs = 0;
do numeric i = 1 while i <= totocc(MEMBRES_MENAGE_ROSTER)
   if LIEN_PARENTE(i) = 1 then
      nombreChefs = nombreChefs + 1;
   endif;
enddo;

if nombreChefs <> 1 then
   errmsg("Le nombre de chefs de ménage doit être exactement un");
   reenter LIEN_PARENTE(1);
endif;
```

Activité : Faire une boucle

Montrez un ménage de quatre membres sur le tableau et montrez le code ci-dessus à l'écran. Choisissez trois volontaires. Un volontaire joue le rôle de la variable nombreChefs, un deuxième joue le rôle de la variable i et un troisième est le directeur de la boucle. Donnez-leur des pancartes indiquant leur rôle pour que tout le monde sache quel rôle ils jouent. Les deux personnes jouant les variables doivent afficher leurs valeurs actuelles en indiquant le nombre avec leurs doigts. Le directeur est responsable de parcourir le code ligne par ligne et de mettre à jour les valeurs des variables jusqu'à la fin de la boucle, en expliquant ce qu'elles font étape par étape.

Maintenant que nous avons tous compris les boucles, voyons comment simplifier notre code en utilisant la fonction count() au lieu d'une boucle.

```
PROC MEMBRES_MENAGE_ROSTER

// Une fois que la liste des membres est au complète, assurez-vous qu'il y a
// un chef de ménage exactement.
numeric nombreChefs = count(INDIVIDU_REC where LIEN_PARENTE = 1);

if nombreChefs <> 1 then
   errmsg("Le nombre de chefs de ménage doit être exactement un");
   reenter LIEN_PARENTE(1);
endif;
```

Forcer le chef de ménage à la première ligne

Au lieu de permettre à l'intervieweur d'inscrire le chef de ménage sur n'importe quelle ligne du roster, nous pouvons simplifier notre logique si nous les forçons à inscrire le chef sur la première ligne. Nous pouvons le faire dans le postproc de **LIEN_PARENTE**.

```
PROC LIEN_PARENTE
// N'autorisez pas les non-chefs de ménage dans la première ligne
if LIEN_PARENTE<> 1 and curocc() = 1 then
   errmsg("Veuillez saisir le chef de ménage avant les autres membres du
ménage.");
   reenter;
endif;

// N'autorisez pas le chef de ménage sur une ligne autre que la première
if LIEN_PARENTE= 1 and curocc() <> 1 then
   errmsg("Un seul chef de ménage est autorisé. Le chef de ménage a déjà été
saisie.");
   reenter;
endif;
```

Cela remplace notre logique de compter le nombre de chefs de ménage dans la postproc du roster. Ceci est plus simple et simplifiera également les autres contrôles impliquant le chef de ménage. Par exemple, pour s'assurer que le chef de ménage a au moins 12 ans de plus que ses enfants, nous pouvons simplement comparer l'âge de l'individu à celui du chef. Comme le chef se trouve sur la première ligne, son âge sera toujours AGE(1). Comme le chef est saisie en premier, nous pouvons faire ce type de contrôle dans le postproc de l'âge au lieu de le faire dans le postproc du roster.

```
PROC AGE
// Comparez l'âge de l'enfant à l'âge du chef de ménage
if LIEN_PARENTE = 3 and AGE(1) - AGE < 12 then
   errmsg("Enfant %s est %d années mais le chef de ménage %s est âgée de %d.
Le parent doit être à moins 12 ans de plus que l'enfant.",
            strip(NOM),
            AGE,
            strip(NOM (1)),
            AGE (1))
            select("Corriger l'âge de " + strip(NOM), AGE,
                 "Corriger l'âge de " + strip(NOM(1)), AGE(1),
                 "Corriger le lien de parenté de " + strip(NOM), LIEN_PARENTE);
   endif;
enddo;
```

Seek

La question **D02** dans la section fécondité demande si une femme a des enfants qui vit avec elle dans le ménage. Si elle répond «oui», il doit y avoir au moins un membre du ménage qui lui a choisi comme mère en spécifiant son numéro d'ordre pour **B10**, numéro d'ordre de la mère. Nous pouvons utiliser la fonction seek pour trouver le numéro de ligne d'un enfant dont le numéro d'ordre de la mère correspond au numéro de ligne de la femme sur la ligne actuelle. Si aucun enfant de ce type n'existe, alors seek retournera zéro.

```
PROC EXISTES_ENFANTS_MENAGE

// Vérifiez la liste du ménage pour vous assurer que le nombre d'enfants
// dans la liste du ménage basée sur le numéro de la ligne mère correspond
// à cette réponse
numeric numeroLigneFemme = curocc();
numeric numeroLigneEnfant = seek(NUMERO_ORDRE_MERE = numeroLigneFemme);
if numeroLigneEnfant > 0 and EXISTES_ENFANTS_MENAGE= 2 then
   warning("Certains enfants du ménage ont indiqué %s comme leur mère", strip (NOM))
   select "Corriger avoir des enfants dans le ménage", EXISTES_ENFANTS_MENAGE,
        "Corriger numéro de ligne de la mère",
NUMERO_ORDRE_MERE(numeroLigneEnfant ),
        "Ignorer l'erreur", continue);
elseif numeroLigneEnfant = 0 and EXISTES_ENFANTS_MENAGE= 1 then
   warning("Aucun enfant dans le ménage n'a donnée %s comme mère", strip(NOM)).
   select ("Corriger avoir des enfants dans le ménage", EXISTES_ENFANTS_MENAGE,
        "Corriger le numéro de ligne de la  mère", NUMERO_ORDRE_MERE(1),
        "Ignorer l'erreur", continue);
endif;
```

Exercices

1. Assurez que la différence d'âge entre le chef de ménage et le petit-enfant ne soit pas inférieure à 24 ans. Si l'âge du chef est égal à X ans et l'âge de l'enfant, à Y ans, alors X-Y ≥ 24.
2. Affichez un message d'erreur si le chef de ménage a au moins un conjoint dans le ménage et que le statut matrimonial du chef de ménage n’est pas marié.
3. Afficher un message d'erreur si le chef de ménage et son conjoint sont du même sexe. Assurez que votre logique fonctionne avec les ménages polygames ainsi que les ménages avec un seul conjoint.
4. Associez les noms des personnes décédées, **E04**, avec les libellés d'occurrence de la section E. Assurez que les libellés sont correctement définis lors de la reprise d'une sauvegarde partielle.

Session 05 : Fonctionnalités CAPI

À la fin de cette leçon, les participants seront en mesure de :

- Ajouter du texte aux questions et de texte de l'aide
- Ajouter des paramètres dans le texte de la question
- Utiliser plusieurs langues dans le texte de la question, les messages d'erreur et le dictionnaire
- Utiliser les libellés d'occurrence comme des paramètres dans le texte de la question
- Personnaliser la liste des cas avec la commande setcaselabel.
- Construire des ensembles de valeurs dynamiques.

Le texte de la question

Nous pouvons ajouter du texte à chaque question de notre application en cliquant sur "CAPI Questions" dans la barre d'outils. Cela nous permet de saisir un texte de question littéral ; le texte que l'intervieweur lira mot pour mot. De plus, nous pouvons ajouter des instructions pour l'intervieweur.

Commençons par les premières questions de la section B. Pour **LIEN_PARENTE**, inscrivez "Quel est le lien de parenté de (nom) avec le chef de ménage ?" Pour **SEXE,** entrez "Quel est le sexe de (nom) ?". Lancez l'application sur Windows, puis sur mobile pour voir comment le texte de la question est affiché.

En plus du texte, nous pouvons ajouter des instructions à l'intervieweur. Par exemple pour **AGE,** nous pouvons mettre :

> Quel est l'âge de (nom) ?
>
> *Inscrivez l'âge en année révolues ("000" pour les enfants moins d'un an)*
> *Si inconnu inscrivez "999"*

Pour indiquer clairement à l'intervieweur qu'il s'agit d'une instruction, nous utilisons *l'italique* pour la distinguer de la question même. Nous pouvons également utiliser des couleurs ou des polices de caractère différentes. Vous pouvez utiliser le schéma de votre choix tant qu'il est cohérent partout dans l'application.

Si vous avez le texte de la question dans Word ou Excel, vous pouvez le copier et le coller dans CSPro pour préserver la mise en forme.

Le texte d'aide

En plus du texte de la question, nous pouvons fournir des instructions supplémentaires à l'intervieweur sous forme de "texte d'aide". Le texte d'aide n'est pas affiché par défaut, mais il pourra s'afficher à l'aide de la touche F2 sous Windows ou en appuyant sur l'icône d'aide à côté de la question sur Mobile.

Ajoutons le texte d'aide suivant au champ **NOM** dans la section B :

> *Inscrivez toutes les personnes vivant dans cette maison et ayant des dispositions communes pour cuisiner et manger.*

Lancez l'application sur Android et Windows et voyez comment le texte d'aide est affiché.

Paramètres dans le texte de la question

Il est possible que la question contienne des paramètres qu'on désire remplacer avec les valeurs des variables du dictionnaire. Par exemple, pour **SEXE**, au lieu de demander "Quel et le sexe de (nom) ?" nous pouvons remplacer "(nom)" avec le nom du répondant en utilisant **%NOM%** dans le texte de la question. Au moment de l'exécution, ceci sera remplacé par le contenu de la variable **NOM**.

Modifions le texte de **LIEN_PARENTE**, **SEXE** et **AGE** pour mettre **%NOM%** à la place de "(nom)".

Les conditions pour le texte de la question

Parfois, le texte de la question sera complètement différent en fonction d'autres facteurs et l'utilisation de variables de remplissage ne suffit pas pour réaliser les différences. Par exemple, le texte de la question pour le champ NOM devrait être :

> Quel est le nom du chef de ménage ? - pour la première ligne du roster
>
> Quel est le nom du prochain membre du ménage ? - pour les lignes restantes

Nous pouvons réaliser cela à partir de la fenêtre de conditions située sous la fenêtre de texte de la question. Par défaut, cette fenêtre contient une seule condition vide. Un clic droit sur les conditions vous permet d'ajouter des nouvelles conditions. Modifions la première condition pour que **Min Occ** et **Max Occ** soient toutes les deux égales à 1, puis ajoutons une deuxième condition où **Min Occ** est égal à 2 et **Max Occ** est égal à 30. Nous avons maintenant deux textes de question différents, l'un qui sera affiché pour la première occurrence et l'autre qui sera affiché pour le reste. Modifiez ces deux textes de questions comme ci-dessus et testez-le pour vous assurer qu'il fonctionne correctement.

	Min Occ	Max Occ	Condition
1	1	1	
2	2	30	

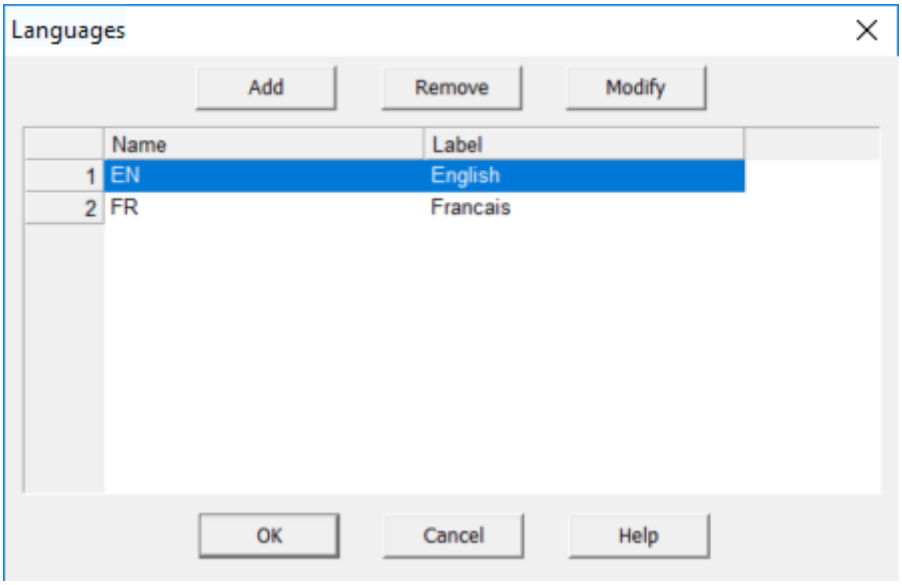
En plus des numéros d'occurrence, il est possible de spécifier une expression logique dans la colonne de condition pour déterminer si le texte doit être affiché ou non. Par exemple, si vous définissez la condition comme "SEXE = 1", le texte de la question sera affiché uniquement pour les hommes.

Saisie en plusieurs langues

Il est possible d'ajouter du texte de question, des libellés de dictionnaire et des messages d'erreur dans plusieurs langues.

Le texte des questions en plusieurs langues

Pour disposer de plusieurs langues pour le texte de la question, vous devez d'abord définir les langues au niveau de l'application de saisie. Cela se fait à partir du menu "CAPI Options". Sélectionnez "Define CAPI Languages". Cela fera apparaître la boîte de dialogue "Languages". Cliquez sur "Add" pour ajouter une langue, spécifiez le nom et le libellé à utiliser pour la langue et cliquez sur "OK". Le libellé est ce que l'intervieweur verra lors du choix d'une langue et le nom c'est ce qui sera utilisé en logique pour désigner la langue. Le nom doit être une abréviation de deux lettres correspondant à la langue de la norme ISO 639-1, telle que EN, FR et ES. Bien que vous puissiez utiliser n'importe quel nom de langue, si vous utilisez l'un des codes ISO 639-1, CSPro définit automatiquement la langue de votre application à la langue de votre appareil mobile (du paramètres système).



Lors de la création du texte de la question, vous entrez le texte de la question pour chaque langue spécifiée. Par exemple, regardons le texte de la question pour la variable **SEXE**. Il existe un menu déroulant pour les langues que nous avons spécifiées dans les étapes ci-dessus. Nous sélectionnons la langue, puis entrons le texte de la question dans cette langue. Pour ajouter les traductions anglaise des questions de sexe, de lien de parenté et d'âge, nous inscrivons :

What is %NAME%'s sex?

What is %NAME%'s relationship to the head of household?

How old is %NAME%?
Enter the age in completed years ("000" for children less than one year old)
If unknown enter "999"

Maintenant que vous avez créé le texte de la question dans les langues spécifiées, vous pouvez le sélectionner lors de la saisie des données. Sous Windows, vous pouvez accéder au menu "Options" et sélectionner "Change language" pour afficher un menu contenant les langues que nous avons définies. Cliquez sur la langue souhaitée et le texte de la question s'affichera dans cette langue. Sur les appareils mobiles, vous trouverez "Changer de langue" dans le menu lors de la saisie des données.

Les éléments du dictionnaire en plusieurs langues

Pour avoir plusieurs langues dans les ensembles de valeurs, vous devez d'abord définir les langues au niveau du dictionnaire. Ceci est similaire au processus que nous avons fait pour définir les langues pour le texte de la question ; toutefois, pour les éléments du dictionnaire, vous définissez les langues à partir du menu “Edit” du dictionnaire. Assurez-vous que vous êtes dans l'éditeur de dictionnaire (vous devrez peut-être cliquer sur le petit livre bleu dans la barre d'outils) et choisir "Edit" puis "Languages". Cette boîte de dialogue est identique à celle utilisée pour ajouter des langues aux textes de questions. Veillez à utiliser les mêmes noms et libellés pour les langues dans cette boîte de dialogue que ceux que vous avez saisis pour le texte de la question.

Une fois que vous avez ajouté des langues supplémentaires au dictionnaire, une liste déroulante de langues apparaît dans la barre d’outils. Pour ajouter une traduction pour un libellé de dictionnaire ou un libellé d’ensemble de valeurs, choisissez la langue dans le menu déroulant et modifiez le libellé comme vous le feriez normalement. Cela définira la version du libellé correspondante à la langue que vous avez choisie.

Messages d'erreur en plusieurs langues

Pour ajouter des traductions du texte spécifié dans la logique du programme, tels que des arguments du errmsg(), nous pouvons fournir des traductions dans le fichier de message. Pour ce faire, nous ajoutons les messages à l'onglet “Messages” au bas de l'éditeur de logique. Au lieu de mettre le texte du message dans la logique, nous étiquetons chaque message avec un numéro et utilisons ce numéro dans la logique. Dans l'onglet “Messages”, nous répertorions les numéros de message suivis du texte du message. Cela nous permet de fournir plusieurs versions du message ; un pour chaque langue. Marquez les différentes langues dans le fichier en utilisant "Language =" suivi du nom de la langue.

```
{Application 'POPSTAN2020' message file generated by CSPro}
Language=FR
100 Le chef de ménage doit avoir au moins 15 ans.
101 Corriger l'âge
102 Corriger le lien de parenté

Language=EN
100 Head of household must be at least 15 years old
101 Correct age
102 Correct relationship
```

Dans la logique vous pouvez passer le numéro de message comme argument à errmsg() en place de la chaîne de caractères. Dans la clause select, toutefois, vous devrez utiliser la fonction tr() ou maketext() autour du numéro du message, sinon CSPro ne saura pas qu'il s'agit d'un message à traduire.

```
PROC AGE

// Assurez que le chef de ménage a au moins 15 ans
if AGE < 15 and LIEN_PARENTE = 1 then
   errmsg(100)
      select(tr(101), AGE, tr(102), LIEN_PARENTE);
endif;
```

Maintenant, lorsque nous exécutons l'application avec la langue définie en anglais et nous rencontrons l'erreur, le message s'affiche en anglais.

Vous pouvez également utiliser un fichier de message distinct pour chaque langue. Pour ajouter un fichier de message supplémentaire, utilisez "Add Files" dans le menu "Files" et saisissez le nom du fichier de message en regard de "External message file".

Travail de groupe
Ajoutez le texte de la question et les traductions anglaises pour le texte de la question, les libellés, les ensembles de valeurs et les messages d'erreur relatifs à l'état matrimonial (**B13**).

Les libellés d'occurrence dans le texte de la question

Nous avons les libellés d'occurrence suivantes pour les types de logement dans la question **F05**.

- Case ronde traditionnelle
- Maison individuelle
- Maison jumelée
- Appartement
- Improvisé (kiosque / conteneur)

Nous pouvons utiliser ces libellés d'occurrence dans le texte de la question pour **F05**:

Combien y at-il de **%getocclabel%** dans ce ménage ?

Chaque fois que vous utilisez **%getocclabel%** dans le texte de la question, il est remplacé par le libellé de l'occurrence en cours. Avec ce qui précède, le texte de la question pour la première occurrence sera "Combien y at-il de **Case ronde traditionnelle** dans ce ménage ?" et le texte de la deuxième occurrence sera "Combien y a-t-il de logements **Maison individuelle** dans ce ménage ?" …

Libellés de cas

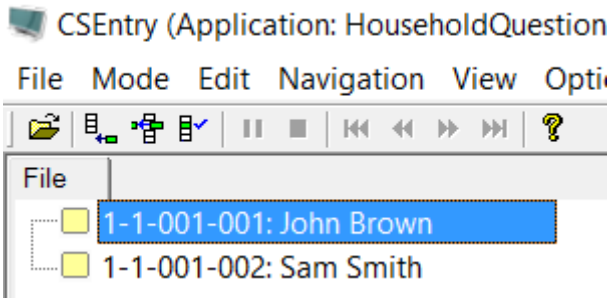
Par défaut, l'écran de liste des cas montre les id-items concaténés. Ce n'est pas très facile à lire pour un intervieweur. Vous pouvez personnaliser la liste des cas à l'aide de la commande setcaselabel. À titre d'exemple, définissons le libellé de cas sur la chaîne "*province-district-numéro de ménage : nom du chef de ménage*". Puisque nous avons besoin du nom du chef de ménage pour le faire, nous pouvons l'ajouter dans la postproc de **NOM**. Pour formater le libellé de cas à partir des variables, nous utilisons la fonction maketext(), qui fonctionne comme errmsg() mais crée une chaîne de caractères que nous pouvons utiliser dans notre logique au lieu d'afficher le message directement à l'écran.

```
PROC NOM

if curocc() = 1 then
    // Définit le libellé de cas pour avoir un format plus
    // facile à lire. Nous faisons cela lorsque nous obtenons
    // le nom du CM.
    string libelleCas = maketext("%d-%02d-%03d-%03d: %s",
        PROVINCE, DISTRICT, ZONE_DENOMBREMENT,
        NUMERO_MENAGE, strip(NOM));
    setcaselabel(POPSTAN2020_DICT, libelleCas);
endif;
```

Les %02d et %03d indiquent à CSPro de formater les nombres avec 2 et 3 chiffres numériques remplis de zéros respectivement. Si DISTRICT est égale à 1, le résultat sera "001" au lieu de "1". Les tailles 2 et 3 sont basées sur les longueurs de ces variables dans le dictionnaire. Au lieu d'utiliser %d, nous pouvons utiliser %v qui prendra automatiquement les paramètres de longueur et de remplissage zéro des variables du dictionnaire pour mettre en forme les valeurs.

```
PROC NOM

if curocc() = 1 then
    // Définit le libellé de cas pour avoir un format plus
    // facile à lire. Nous faisons cela lorsque nous obtenons
    // le nom du CM.
    string libelleCas = maketext("%v-%v-%v-%v: %s",
        PROVINCE, DISTRICT, ZONE_DENOMBREMENT,
        NUMERO_MENAGE, strip(NOM));
    setcaselabel(POPSTAN2020_DICT, libelleCas);
endif;
```

Maintenant, après avoir saisie un cas, nous avons une liste de cas beaucoup plus conviviale:



Notez que setcaselabel ne fonctionne qu'avec des fichiers de données de type CSPro DB.

Ensembles de valeurs dynamiques

Il est souvent utile de modifier les réponses de la question à partir de la logique. Cela peut être fait en utilisant la commande setvalueset. Commençons par un exemple simple. Actuellement, notre ensemble de valeurs pour le lien de parenté dans la section B comporte des libellés tels que "Fils / Fille" et "Frère / Soeur" pour permettre les deux sexes. Cependant, lorsque nous affichons cet ensemble de valeurs, nous connaissons déjà le sexe du membre du ménage en question. Alors, nous pourrions afficher "Fils" pour les hommes et "Fille" pour les femmes. Pour ce faire, nous créons deux nouveaux ensembles de valeurs pour le lien de parenté dans le dictionnaire : **LIEN_MALE_VS** et **LIEN_FEMALE_VS**. Ensuite, nous pouvons utiliser la logique pour choisir entre les ensembles de valeurs en fonction de la valeur de **SEXE**. Dans quelle proc faisons-nous cela ? Nous le faisons dans la proc ONFOCUS de **LIEN_PARENTE** puisque nous devons l'utiliser lorsque nous sommes dans ce champ. ONFOCUS est exécuté chaque fois que l'on entre dans le champ. Nous ne pouvons pas le faire dans la preproc car celle-ci n'est pas déclenchée lorsque vous revenez en arrière dans les questions.

```
PROC LIEN_PARENTE
onfocus
// Affiche la version masculin ou féminin de l'ensemble de valeurs en fonction du
sexe
// de la personne.
if SEXE = 1 then
   setvalueset(LIEN_PARENTE, LIEN_MALE_VS);
else
   setvalueset(LIEN_PARENTE, LIEN_FEMALE_VS);
endif;
```

Ensemble de valeur dynamique à partir d'un roster

Pour la question **A11**, numéro de ligne du répondant, nous souhaitons créer un ensemble de valeurs à partir des noms et des numéros de ligne des membres du ménage éligibles. Pour que cela fonctionne correctement, nous devons poser la question **A11** après avoir saisie les noms et les âges des membres du ménage. Créez une nouvelle variable

numérique dans le dictionnaire pour **A11** et placez-la sur le formulaire “membres du ménage” après le roster

Nous allons remplir l’ensemble de valeurs avec les noms et les numéros de ligne des membres éligibles du ménage. Selon la spécification, seuls les membres du ménage âgés de 12 ans et plus peuvent être le répondant. Par exemple, si nous avons le ménage suivant :

	Line number	Name	Sex	Relationship	Age
1	1	John Brown	1	1	39
2	2	Mary Brown	2	2	40
3	3	Bobby Brown	1	3	11
4	4	Jane Brown	2	7	22

Nous remplissons l’ensemble de valeurs comme suit :

Indice	Codes	Libellés
1	1	John Brown
2	2	Mary Brown
3	4	Jane Brown

Pour créer l’ensemble de valeurs à partir de la liste des membres du ménages nous avons besoin de la deuxième forme de setvalueset qui prend un objet du type valueset. Cela nous permettra de créer la liste des noms en logique plutôt que dans le dictionnaire.

Tout d'abord, nous devons déclarer un objet de type valueset. Un objet de type valueset est une variable comme string ou numeric, mais au lieu de contenir une seule valeur, il peut contenir tout un ensemble de valeurs complet avec ses codes et ses libellés.

```
PROC NUMERO_ORDRE_REPONDANT
onfocus
valueset repondantVS;
```

Nous pouvons ajouter des valeurs à l'ensemble de valeurs en appelant la fonction repondantVS.add(). Le point (“.”) entre “repondantVS” et “add” indique à CSPro d'appliquer l'opération d'ajout à l'ensemble de valeurs avant le point qui dans ce cas est l'ensemble de valeurs nommé repondantVS. Par exemple, pour ajouter John Brown à l’ensemble de valeurs, procédez comme suit :

```
repondantVS.add("John Brown", 1);
```

Donc, pour construire notre ensemble de valeurs, nous devons appeler la fonction add pour chaque membre du ménage de 12 ans et plus.

```
repondantVS.add("John Brown", 1);
repondantVS.add("Mary Brown", 2);
repondantVS.add("Jane Brown", 4);
```

Pour que cela fonctionne pour n'importe quel ménage, nous devons faire l'ajout dans une boucle sur les membres du ménage.

```
PROC NUMERO_ORDRE_REPONDANT
onfocus
// Créer l'ensemble de valeurs des membres du ménage de 12 ans
// et plus
valueset repondantVS;
do numerique i = 1 while i <= totocc(MEMBRES_MENAGE_ROSTER)
   if AGE(i) >= 12 then
      repondantVS.add(NOM(i), i);
   endif;
enddo;

setvalueset(NUMERO_ORDRE_REPONDANT, repondantVS);
```

Veillez à définir le type de capture du champ sur "Radio button" dans les propriétés du champ pour **NUMERO_ORDRE_REPONDANT**.

Dans un deuxième exemple, nous allons faire un ensemble de valeurs pour le numéro de ligne de la mère (**B10**). Pour cet ensemble de valeurs, nous ne montrons que les femmes du ménage âgées de plus de 12 ans et n'autorisons pas une femme à être sa propre mère.

```
// Création d'un ensemble de valeurs pour les femmes de plus de
// 12 ans dans la liste de ménages
valueset mereVS;

do numeric i= 1 while i <= totocc(MEMBRES_MENAGE_ROSTER)
   if SEXE(i) = 2 and AGE(i) > 12 and i <> curocc() then
      mereVS.add(NOM(i), i);
   endif;
enddo;
```

Nous devons également ajouter les codes des non-résidents et des personnes décédées dans l'ensemble de valeurs.

```
mereVS.motherVSet("Non-résident", 87);
mereVS.motherVSet("Décédé", 88);
```

Enfin, nous devons associer le nouvel ensemble de valeurs avec le champ :

```
setvalueset(NUMERO_ORDRE_MERE, mereVS);
```

Ensembles de valeurs dynamiques extraits des cases à cocher

La question **B16** (langue principale) ne doit afficher qu'un sous-ensemble des langues choisies dans **B15** (langues parlées). Nous pouvons le faire en utilisant également un ensemble de valeurs dynamiques ou on supprime les réponses qui n'ont pas été cochées dans **B15**. Puisque **B15** utilise des cases à cocher, il aura des codes alpha (A, B, C…) tandis que **B16** aura des codes numériques (1,2,3…). Nous devons donc passer de numérique à alpha pour déterminer si une langue donnée a été choisie. Nous pourrions le faire avec une série de if then else, mais une approche plus simple consiste à utiliser la chaîne **"ABCDEFGH"** pour convertir du numérique en alpha en recherchant le caractère à la position du code numérique. Par exemple, le code numérique 1 nous donnerait le caractère à la première position : A. Le code numérique 2 nous donnerait le caractère à la position 2, B etc…

```
PROC LANGUE_PRINCIPALE
onfocus

// Créer un ensemble de valeurs à partir d'éléments sélectionnés
// dans les langues parlées

// Déclarer une chaîne utilisée pour traduire de la case à cocher (codes alpha) en
// codes numériques
string langueCodeAlphas = "ABCDEFGH";

// Commencer avec l'ensemble de valeurs qui contient toutes les langues
valueset langueVS = LANGUE_PRINCIPALE_VS1;

// Boucler entre les codes numériques 1 à 8 et enlever
// les valeurs non-sélectionnées du valueset
do numeric langueCodeNum = 1 while langueCodeNum <= 8

   // Convertir le code numérique en code alpha à case à cocher
   // en le recherchant dans la chaîne
   string langueCodeAlpha = langueCodeAlphas[langueCodeNum:1];

   // Vérifie si la langue est sélectionnée dans la case à cocher
   if pos(languageCheckboxCode, LANGUES_PARLEES) = 0 then
      // La langue n'est pas sélectionnée. Enlevez-la de l'ensemble de valeurs
      langueVS.remove(langueCodeNum);
   endif;
enddo;
```

```
// Modifier de l'ensemble de valeurs du champ
setvalueset(LANGUE_PRINCIPALE, langueVS);
```

Que se passe-t-il si l'intervieweur ne choisit aucune langue dans **B15**? Dans ce cas, notre ensemble de valeurs dynamiques est vide. Nous devrions ajouter un contrôle à **B15** pour nous assurer qu'au moins une langue est choisie.

```
PROC LANGUAGES_PARLEES

// Assurez qu'au moins une langue est choisie
if length(strip(LANGUAGES_PARLEES)) = 0 then
   errmsg("Vous devez choisir au moins une langue");
   reenter;
endif;
```

Cases à cocher dynamiques

Faisons un ensemble de valeurs dynamiques pour la question **G2**, les biens du ménage achetés avec un prêt. Plutôt qu'une série de questions oui / non, nous afficherons cela en utilisant une seule variable avec des cases à cocher. Nous pourrions avoir une case à cocher pour chacun des 10 éléments de la liste des biens, mais il serait préférable que nous affichions uniquement les cases à cocher pour le sous-ensemble des biens que le ménage possède. Comment savons-nous si le ménage possède un bien ? Le ménage possède le bien si sa quantité est supérieure à zéro. Nous devons parcourir les lignes du roster et ajouter une case à cocher pour chaque article dont la quantité est supérieure à zéro. La difficulté est que comme ce sont des cases à cocher, nous devons donc utiliser des valeurs alpha.

Afin de créer un ensemble de valeurs avec des valeurs alpha, nous devons déclarer un objet de type valueset qui accepte les codes alpha.

```
valueset string biensVS;
```

Comme d'habitude, nous construisons l'ensemble de valeurs dans la proc onfocus du champ. Nous utilisons à nouveau l'astuce de l'alphabet pour obtenir les codes alpha à partir du numéro d'occurrence. Nous utilisons également la fonction getocclabel() pour extraire le libellé d'occurrence de la ligne actuelle du roster à utiliser dans l'ensemble de valeurs.

```
PROC BIENS_ACHETES_AVEC_PRET
onfocus

// Créer un ensemble de valeurs dynamiques à partir des biens
// dont la quantité > 0
valueset string biensVS;

string alphabet = "ABCDEFGHIJ";

do numeric numeroBien = 1 while numeroBien <= totocc(BIENS_ROSTER)
   // Vérifie si le ménage possède ce bien
   if QUANTITE(numeroBien) > 0 then
      // Ajouter à l'ensemble de valeurs
      string libelle = getocclabel(BIENS_ROSTER(numeroBien));
      biensVS.add(libelle, alphabet[numeroBien:1]);
   endif;
enddo;

setvalueset($, biensVS);
```

Exercices

1. Ajoutez le texte de la question pour les questions B06 à B08. Utilisez des paramètres pour intégrer le nom, comme nous l'avons fait dans les exemples.
2. Ajoutez une nouvelle langue, la langue de votre choix, au texte des questions et au dictionnaire. Traduisez le texte, les libellés et les ensembles de valeurs des questions B06-B08 dans cette nouvelle langue.
3. Ajoutez le texte de la question pour les questions de la section G et utilisez les libellés d’occurrence pour indiquer le nom des biens dans le text de la question pour les champs Quantité et Valeur.
4. Dans la question B06 (date de naissance), utilisez un ensemble de valeur dynamique pour le jour en fonction du mois (janvier: 1-31, février: 1-28, mars: 1-31…) afin que l’enquêteur ne puisse pas saisir une date non valide, telle que le 30 février ou le 31 avril. Bonus si vous pouvez gérer correctement les années bissextiles.
5. Pour E08 (numéro de ligne de la mère de la personne décédée), utilisez un ensemble de valeurs dynamiques pour afficher le nom de toutes les femmes éligibles figurant sur la liste des membres du ménage.

Session 06 : Fonctions, fichiers “lookup”, et navigation

À la fin de cette session, les participants seront en mesure de :

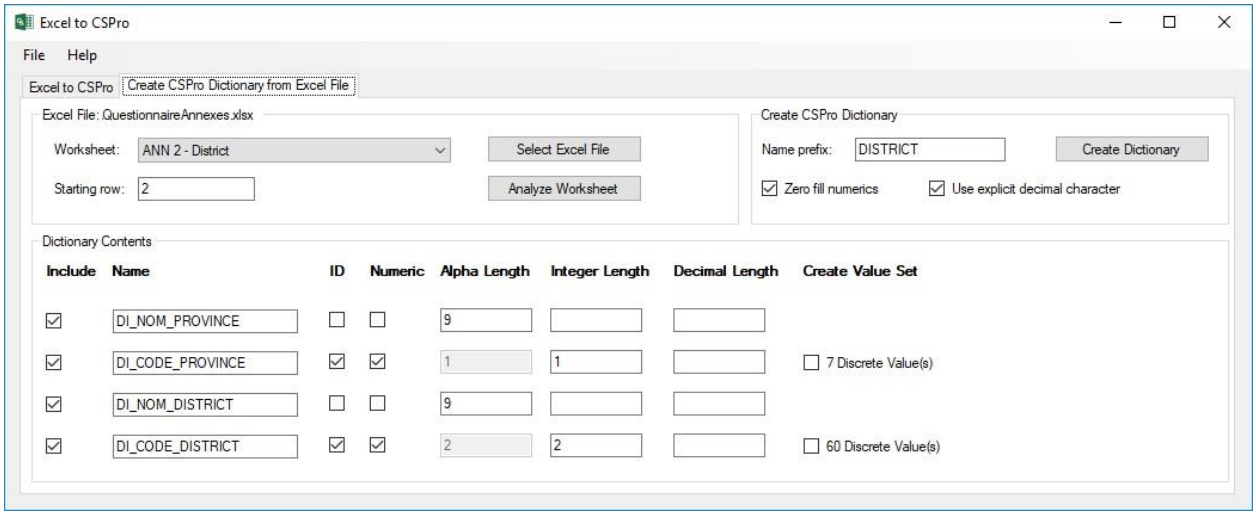
- Utiliser des fichiers de recherche (lookup) dans la saisie de données
- Créer des fonctions définies par l'utilisateur dans CSPro
- Étendre l'interface de CSEntry à l'aide des boutons du “userbar”
- Utiliser les commandes advance et move pour parcourir le questionnaire
- Utiliser la fonction visualvalue pour obtenir la valeur de variables "hors chemin"
- Utiliser la commande showarray pour afficher les tableaux

Fichiers “lookup”

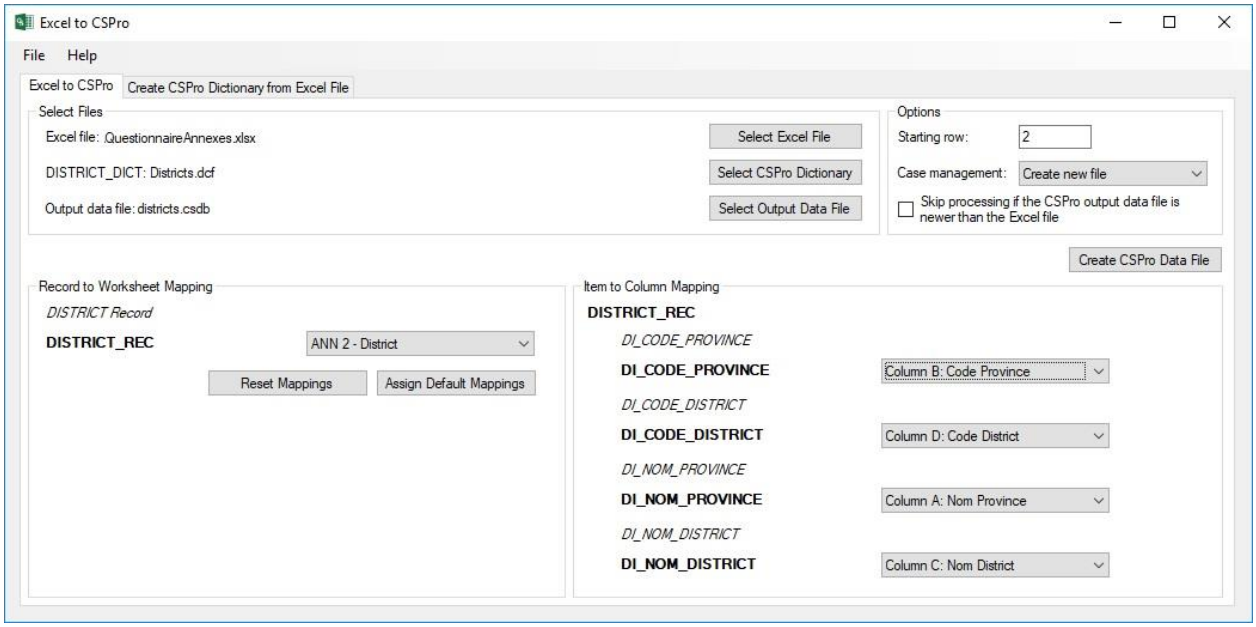
Il est actuellement possible pour l'intervieweur de saisir un code de district non valide pour la province qui a été sélectionnée. Afin de vérifier que le code de district est valide pour la province, nous devons utiliser la liste des codes de province et de district figurant en annexe 2 du questionnaire. Nous pouvons le faire en créant un fichier “lookup” (ficher de recherche) à partir de la feuille de calcul Excel. Nous pouvons ensuite utiliser ce fichier de recherche pour vérifier les districts valides.

D’abord, nous devons créer un dictionnaire pour notre fichier de recherche. Pour cette tâche, nous voulons pouvoir demander si la combinaison des codes de province et de district forme un district valide. Dans l'annexe 2 du questionnaire, nous avons un tableau avec le code de province, le code de district et le nom du district. Nous pouvons utiliser ceci comme fichier de recherche où les clés sont la province et le code du district et la valeur est le nom du district.

Nous pouvons créer un dictionnaire pour cela en utilisant l'outil Excel2CSPro à partir du menu “Tools”. Sélectionnez l'onglet "Create CSPro dictionary from Excel File", cliquez sur "Select Excel File" et accédez au fichier QuestionnaireAnnexes.xlsx. Choisissez "ANN2 - District" et cliquez sur "Analyze Worksheet". Ceci détecte les variables du dictionnaire à créer à partir des colonnes de la feuille de calcul. L'outil affichera quatre variables, une pour chaque colonne et définira les noms par défaut à partir des en-têtes de colonne de la feuille de calcul. Notre dictionnaire de fichier de recherche aura le code de province et le code de district en tant qu'id-items et les noms de province et de district en tant que variables normales. En regard des codes de province et de district, cochez la case "ID" pour indiquer qu'il doit s'agir d'éléments d'identification. Afin d'éviter les conflits de noms avec le dictionnaire principal, nous pouvons utiliser DI_CODE_PROVINCE_CODE, DI_CODE_DISTRICT, DI_NOM_PROVINCE et DI_NOM_DISTRICT en tant que noms de variable. Modifiez-les dans l'outil Excel2CSPro.



Nous devons nous assurer que les paramètres de longueur et de remplissage zéro des identifiants correspondent exactement à ceux du dictionnaire principal, sinon nous ne pourrons pas utiliser les variables du dictionnaire principal comme clés de la recherche. Vérifiez que la longueur des codes de province et de district correspond à celle du dictionnaire principal et cochez la case "Zero fill numerics" pour vous assurer que les identifiants sont remplis de zéros. Une fois que tous les paramètres sont corrects, cliquez sur “Create Dictionary” pour générer le fichier de dictionnaire CSPro. Sauvegardez-le sous "Districts.dcf" dans le dossier ménage.



Une fois que nous avons le dictionnaire, nous devons convertir la feuille de calcul Excel en fichier de données CSPro. Nous pouvons utiliser le premier onglet de l'outil Excel2CSPro pour ce faire. Sélectionnez à nouveau le fichier QuestionnaireAnnexes.xlsx avec le dictionnaire Districts que nous venons de créer. Spécifiez un nouveau fichier de données de sortie dans lequel écrire le fichier de recherche. Sous "Record to Worksheet Mapping", choisissez la deuxième feuille de travail "ANN 2 - District". Sous "Item to Column Mapping", liez les variables du dictionnaire aux colonnes appropriées de la feuille de calcul Excel :

DI_CODE_PROVINCE à "Colonne B: code province", DI_DISTRICT_CODE à "Colonne D: Code district", etc... Enfin, cliquez sur “Create CSPro Data File” pour générer le fichier. Enregistrez-le dans le dossier de données. Vérifiez le fichier dans DataViewer pour vous assurer qu'il a été converti correctement.

Enfin, dans la PROC du district, utilisez la commande loadcase() pour rechercher les codes de province et de district dans le fichier. Loadcase prend le nom du dictionnaire (**DISTRICT_DICT**) et les valeurs à utiliser comme clés (id-items) pour la recherche comme arguments. Par exemple, pour rechercher la province 3 dans le district 6, nous ferions :

```
loadcase(DISTRICT_DICT, 3, 6)
```

Si loadcase trouve un enregistrement dans le fichier de recherche avec le code de province 3 et le code de district 6, il renvoie 1 et définit les variables dans **DISTRICT_DICT** avec les valeurs l'enregistrement trouvée. Dans ce cas, cela signifie que loadcase définit les deux variables identifiantes **DI_CODE_PROVINCE**, **DI_CODE_DISTRICT** et en plus les variables **DI_NOM_DISTRICT** et **DI_NOM_PROVINCE**.

Nous pouvons utiliser cette information pour vérifier si les codes de province et de district sont valides dans la proc **DISTRICT** :

```
PROC DISTRICT

// Vérifiez que le code de district est valide pour la province sélectionnée.
if loadcase(DISTRICT_DICT, PROVINCE, DISTRICT) = 0 then
   errmsg("Le code de district %d n'est pas valide pour la province %l",
         DISTRICT, PROVINCE);
   reenter;
else
   errmsg("Vous avez sélectionné le district: %s", DI_NOM_DISTRICT);
endif;
```

Notez que nous utilisons les variables **PROVINCE** et **DISTRICT** du dictionnaire principal comme arguments pour loadcase, et non les id-items du dictionnaire districts (DI_CODE_PROVINCE, DI_CODE_DISTRICT). Avant d'exécuter la commande loadcase, les identifiants du dictionnaire externe sont tous vides. Ils ne sont définis que si loadcase a réussi.

Notez que lorsque vous exécutez cette application, vous devez également copier le fichier de recherche (le fichier .csdb), en plus de copier les fichiers pen et pff sur le périphérique Android.

Nous pouvons maintenant ajouter des variables alpha au dictionnaire principal, leur attribuer les noms de la province et du district et les afficher sur le formulaire sous forme de champs protégés :

```
// Vérifiez que le code de district est valide pour la province sélectionnée.
if loadcase(DISTRICT_DICT, PROVINCE, DISTRICT) = 0 then
   errmsg("Le code de district %d n'est pas valide pour la province %l",
         DISTRICT, PROVINCE);
   reenter;
else
   // Attribuez les noms de province et de district du fichier lookup aux variables du
   // dictionnaire principale afin que nous puissions les afficher sur le formulaire.
   NOM_PROVINCE = DI_NOM_PROVINCE;
   NOM_DISTRICT = DI_NOM_DISTRICT;
endif;
```

Mise à jour d'un fichier de recherche

Si les données de votre fichier de recherche changent une fois que votre enquête est sur le terrain, plutôt que de régénérer complètement le fichier de recherche, vous pouvez mettre à jour le fichier de données de recherche existant basé sur une nouvelle feuille de calcul Excel. Choisissez le fichier csdb existant comme fichier de données de sortie et sous "Case management", choisissez "Modify, add cases" ou "Modify, add, delete cases". Au lieu de supprimer et de régénérer le fichier de données, cela ne mettra à jour que les cas nouveaux ou modifiés dans le fichier Excel. Si vous choisissez "Modify, add, delete cases", vous supprimerez également les cas figurant dans le fichier de données mais non dans la feuille de calcul. L'utilisation de cette approche est importante si vous envisagez d'utiliser la synchronisation de données CSPro pour mettre à jour le fichier de recherche lors de la collecte de données. Si vous utilisez la synchronisation de données pour mettre à jour le fichier et générez un tout nouveau fichier de données, vous obtiendrez des doublons lorsque le nouveau fichier de recherche sera téléchargé sur les tablettes. Si, au lieu de cela, vous mettez à jour le fichier existant, la synchronisation des données mettra simplement à jour les fichiers de données sur les appareils qui le téléchargent.

Ensemble de valeurs dynamiques à partir d'un fichier lookup

A lieu d’utiliser le fichier de recherche pour vérifier les codes de district et de province, nous pouvons combiner le fichier de recherche avec setvalueset pour créer un ensemble de valeurs dynamique pour le district comprenant uniquement les districts de la province sélectionnée. Pour ce faire, nous devons extraire tous les districts de la province sélectionnée à partir du fichier de recherche. Nous pouvons le faire en utilisant la boucle forcase qui itère sur tous les cas dans un fichier de données externe. En lui-même, forcase parcourt chaque cas dans le fichier de données. Vous pouvez également ajouter une clause facultative where pour ne parcourir que les districts de la province sélectionnée.

```
PROC DISTRICT
onfocus
// Créer un ensemble de valeurs dynamiques de districts pour la province
// sélectionnée à l'aide du fichier de recherche
valueset districtsVS;
forcase DISTRICT_DICT where DI_CODE_PROVINCE = PROVINCE do
   districtsVS.add(DI_NOM_DISTRICT, DI_CODE_DISTRICT);
endfor;
setvalueset(DISTRICT, districtsVS);
```

Fonctions définies par l'utilisateur

Vous trouvez souvent que vous avez des blocs de logique identiques dans plusieurs procs de votre application. Cela peut poser problème si vous modifiez le code ultérieurement à un endroit pour corriger un bogue et oubliez de le changer à l'autre. Dans de telles situations, il est préférable de placer la logique dans une fonction définie par l'utilisateur que vous pouvez ensuite appeler à partir de toutes les procs où elle est utilisée.

Les fonctions définies par l'utilisateur sont définies dans la proc global. Tout ce qui est déclaré dans la proc global est disponible dans toutes les procs de votre programme. Si vous déclarez une variable de logique à l'intérieur de la proc d'un champ, elle est uniquement disponible à l'intérieur de cette proc, mais si vous la déclarez dans la proc global, vous pouvez l'utiliser n'importe où. Pour afficher la proc global, dans la vue logique, cliquez sur le premier élément de l'arborescence de formulaire. Cela montre toute la logique du programme à la fois : la proc global plus toutes les autres procs. En cliquant sur un autre élément de l'arborescence de formulaire, vous affichez uniquement la proc de cet élément.

Les fonctions définies par l'utilisateur peuvent prendre des arguments et renvoyer des valeurs, tout comme les fonctions intégrées de CSPro. Vous les appelez de la même façon que vous appelez les fonctions intégrées. Définissons une fonction que nous pouvons utiliser dans tous les endroits où nous vérifions si un membre du ménage est une femme en âge de procréer. Cette fonction sera passée l'indice (numéro de ligne) du membre du ménage figurant dans le roster des membres du ménage et renvoie un résultat d'un si la personne est une femme de plus de 12 ans et un résultat de zéro sinon. De cette façon, si nous décidons par la suite que nous devrions utiliser 13 ou 14 ans comme âge minimum, nous ne devons effectuer le changement qu'à un seul endroit.

```
// Déterminer si le membre du ménage figurant sur le roster du ménage est une
// femme à l'âge de procréer. Cette fonction prend comme argument l'indice (numéro
// d'occurrence) du membre du ménage.
function femmeAgeProcreer(indice)
   if SEXE(indice) = 2 and AGE(indice) >= 12 then
      femmeAgeProcreer = 1;
   else
      femmeAgeProcreer = 0;
   endif;
end
```

Nous pouvons maintenant utiliser cette fonction lors de la création des ensembles de valeurs pour le numéro de ligne de la mère de l'enfant (**B10**) et le numéro de ligne de la mère de la personne décédée (**E08**).

```
PROC NUMERO_ORDRE_MERE
onfocus
// Crée l'ensemble de valeurs pour la mère de l'enfant de toutes les femmes éligibles
// dans le roster des membres du ménage.
valueset mereVS;
do numeric i= 1 while i <= totocc(MEMBRES_MENAGE_ROSTER)
   if femmeAgeProcreer(i) = 1 and i <> curocc() then
      mereVS.add(NOM(i), i);
   endif;
enddo;
mereVS.add ("Non-résident", 87);
mereVS.add ("Décédé", 88);
setvalueset(NUMERO_ORDRE_MERE, mereVS);

PROC MERE_DU_DECEDE
onfocus
// Crée l'ensemble de valeurs pour la mère du défunt à partir de toutes les femmes
// éligibles dans le roster du ménage
valueset mereVS;
do numeric i = 1 while  i <= totocc(MEMBRES_MENAGE_ROSTER)
   if femmeAgeProcreer(i) = 1 then
      mereVS.add (NOM(i), i);
   endif;
enddo;
mereVS.add ("pas dans le ménage", 99);
setvalueset(MERE_DU_DECEDE, mereVS);
```

En regardant ces deux procs, nous voyons que nous pouvons remplacer le code partagé qui crée l'ensemble de valeurs avec une fonction.

```
// Créez un ensemble de valeurs de tous les membres du ménage éligibles d'être
// mère en remplissant l'ensemble de valeurs mèreVS transmis en tant que
// premier argument. Les étiquettes de l'ensemble de valeurs seront les noms des
// membres du ménage et les codes sont les numéros de ligne correspondants
// Le membre du ménage dans la ligne exclureIndice ne sera pas inclus dans
// l'ensemble de valeurs. Cela peut être utilisé pour exclure une personne d'être sa
// propre mère.
function creerEnsembleValeursMere(valuset mereVS, exclureIndice)
   do numeric i = 1 while i <= totocc(MEMBRES_MENAGE_ROSTER)
      if femmeAgeProcreer(i) = 1 and i <> exclureIndice then
         mereVS.add (NOM(i), i);
      endif;
   enddo;
end;

PROC NUMERO_ORDRE_MERE
onfocus

// Crée l'ensemble de valeurs la mère de l'enfant à partir de toutes
// les femmes éligibles dans le roster des membres du ménage.
// Exclure l'occurrence en cours pour que cette personne ne puisse pas être
// sa propre mère.
valuset mereVS;
creerEnsembleValeursMere(mereVS, curocc());

// Ajoute des entrées supplémentaires pour non-résident et décédé
mereVS.add("Non-résident", 87);
mereVS.add ("Décédé", 88);

setvalueset(NUMERO_ORDRE_MERE, mereVS);

PROC MERE_DU_DECEDE
onfocus
// Crée l'ensemble de valeurs pour la mère du défunt à partir de toutes les femmes
// éligibles dans le roster du ménage
valuset mereVS;
creerEnsembleValeursMere(mereVS, 0);

// Ajoute une entrée supplémentaire pour pas dans le ménage
mereVS.add("pas dans le ménage", 99);

setvalueset(MERE_DU_DECEDE, mereVS);
```

Fonctions dans le texte de la question

Une autre application des fonctions définies par l'utilisateur concerne les paramètres dans le texte de la question qui nécessitent un calcul. Par exemple, pour la question **D05,** nous souhaitons vérifier que la somme des enfants vivants avec la femme, les enfants vivants ailleurs et les enfants décédés correspond au nombre total de naissances. Pour ce faire, nous posons la question suivante :

> Pour confirmer, (nom) a un total de (naissances totales) enfants nés vivants dans sa vie. Est-ce que c'est correct ?

Nous pouvons utiliser le texte de la question pour cela, mais nous n'avons pas de variable du dictionnaire pour le nombre total de naissances. Nous avons seulement la variable oui / non **NAISSANCES_TOTALES_CORRECT**. Nous pourrions créer une variable de dictionnaire supplémentaire et le remplir, mais on peut plus simplement créer une fonction pour calculer le nombre total de naissances et l'utiliser comme paramètre dans le texte de la question.

```
// Calculez le nombre de naissances total de la femme pour afficher dans le texte
// de la question dans la section de fécondité
function naissancesTotales()
   naissancesTotales = ENFANTS_MENAGE + ENFANTS_AILLEURS +
                       ENFANTS_DECEDES;
end;
```

Dans le texte de la question, nous pouvons simplement ajouter l'appel de fonction naissancesTotales entouré de "%" pour indiquer que ça soit un paramètre à remplacer :

> Pour confirmer, %NOM% a un total de %naissancesTotales()% enfants nés vivants dans sa vie. Est-ce-que c'est correct ?

Que se passe-t-il lorsqu'un des champs de ce calcul est sauté ? La valeur devient nonappl qui gâche tout le calcul. Nous devons être un peu plus intelligents dans le calcul de notre total pour ignorer les valeurs sautées.

```
// Calculez le nombre de naissances total de la femme pour afficher dans le texte
// de la question dans la section de fécondité
function naissancesTotales()
   numeric total= 0;

   if ENFANTS_MENAGE <> notappl then
      total = total + ENFANTS_MENAGE ;
   endif;

   if ENFANTS_AILLEURS <> notappl then
      total = total + ENFANTS_AILLEURS;
   endif;
```

```
    if ENFANTS_DECEDES<> notappl then
       total = total + ENFANTS_DECEDES;
    endif;

    naissancesTotales = total;
end;
```

Le userbar

À partir de la logique, nous pouvons ajouter des boutons à l'interface utilisateur CSEntry qui appellent des fonctions définies par l'utilisateur. Ces boutons sont placés dans ce qu'on appelle le "userbar" (la barre d'utilisateur). Voici comment ajouter un bouton dans la barre d'utilisateur qui créera une boîte de dialogue errmsg qui dit "bonjour". Premièrement, nous définissons une fonction bonjour() dans la proc global:

```
function bonjour()
   errmsg("Bonjour");
end;
```

Ensuite, dans la preproc de l'application, nous l'ajoutons à la barre d'utilisateur :

```
PROC POPSTAN2020_FF
preproc
userbar(clear);
userbar(add button, "Bonjour", bonjour);
userbar(show);
```

Lors de l'ajout d'un bouton dans la preproc de l'application, il est important d'appeler d'abord userbar(clear) pour dégager les boutons du userbar, sinon nous pouvons nous retrouver avec deux ou trois copies du même bouton si nous lançons l'application plusieurs fois. Nous avons ajouté le bouton dans la preproc de l'application, mais vous pouvez ajouter ou supprimer des boutons dans n'importe quel proc afin que vous puissiez, par exemple, uniquement afficher un bouton dans un certain champ ou un certain roster.

Essayons un exemple plus intéressant. Ajoutons un bouton "Aller à ..." qui permettra à l'utilisateur de naviguer directement vers une section particulière du questionnaire.

```
// Fonction userbar pour accéder directement aux différentes parties
// du questionnaire.
fonction allera()
   numeric section= accept("Aller à quelle section ?",
                           "Identification",
                           "Membres du ménage",
                           "Caractéristiques individuelles");
   if section= 1 then
      skip to IDENTIFICATION_FORM;
   elseif section = 2 then
      skip to MEMBRES_MENAGE_FORM;
   elseif section = 3 then
      skip to CARACTERISTIQUES_INDIVIDU_FORM;
   endif;
end;

PROC POPSTAN2020_FF
preproc
userbar(clear);
userbar(add button, "Aller à ...", allera);
userbar(show);
```

Les commandes advance et move

Ce qui précède permettra à l'intervieweur de passer d'une section à une autre. Mais utiliser la commande skip signifie que si nous utilisons notre bouton *Aller à…* pour sauter par-dessus une section, cette section sera ignorée et ne sera pas enregistrée dans le fichier de données. Au lieu d'utiliser skip, nous pouvons utiliser la commande advance qui avance dans le questionnaire sans marquer les champs sautés. L'utilisation d'advance exécute également toutes les preprocs et postprocs des champs qui sont passés afin de s'assurer qu'aucune vérification de cohérence n'est manquée.

```
// Fonction userbar pour accéder directement aux différentes parties
// du questionnaire.
fonction allera()
   numeric section= accept("Aller à quelle section ?",
                           "Identification",
                           "Membres du ménage",
                           "Caractéristiques individuelles");
   if section= 1 then
      advance to IDENTIFICATION_FORM;
   elseif section = 2 then
      advance to MEMBRES_MENAGE_FORM;
   elseif section = 3 then
      advance to CARACTERISTIQUES_INDIVIDU_FORM;
   endif;
end;
```

Cela empêche la fonction de sauter des données lors de la navigation, mais elle a toujours une limitation. Nous ne pouvons naviguer à un seul sens dans le questionnaire. Pour revenir en arrière, nous devons utiliser reenter, mais dans notre fonction allera(), nous ne savons pas si l'utilisateur souhaite avancer ou reculer. Heureusement, CSPro fournit la commande move qui utilisera soit skip ou reenter selon le cas. Par défaut, lors du déplacement, move effectue un saut, mais vous pouvez ajouter le mot advance après le nom du champ pour lui faire faire un advance au lieu d'un skip.

```
// Fonction userbar pour accéder directement aux différentes parties
// du questionnaire.
fonction allera()
   numeric section= accept("Aller à quelle section ?",
                           "Identification",
                           "Membres du ménage",
                           "Caractéristiques individuelles");
   if section= 1 then
      move to IDENTIFICATION_FORM advance;
   elseif section = 2 then
      move to MEMBRES_MENAGE_FORM advance;
   elseif section = 3 then
      move to CARACTERISTIQUES_INDIVIDU_FORM advance;
   endif;
end;
```

Étendons notre fonction allera() pour permettre à l'intervieweur d'accéder directement à une personne figurant dans la liste des membres du ménage. S'ils choisissent l'option "membres du ménage", alors au lieu d'aller à la première personne du roster, nous leur montrerons une liste de tous les membres du ménage et leur laisserons choisir lequel aller. Pour cela, nous pouvons utiliser la fonction showarray(). Cette fonction prend une matrice de valeurs et les affiche dans une grille dans une boîte de dialogue. Il renvoie comme résultat de la fonction le numéro de la ligne choisi par l'utilisateur.

Dans CSPro, les variables du type matrice sont les variables array. Une variable de logique de type array est similaire à un élément de dictionnaire avec des occurrences. Un array numérique de longueur sept stocke sept valeurs numériques, chacun étant accessible via des indices.

```
array numeric codes(7);
```

showarray() prend un array à deux dimensions. Vous pouvez imaginer un tableau à deux dimensions sous forme de grille de variables ou de matrice. Vous déclarez un array à deux dimensions de la même manière que vous déclarez un array à une dimension, sauf que vous spécifiez la taille dans les deux dimensions : nombre de lignes et nombre de colonnes.

```
array string membresMenageArray(30, 3);
```

Dans notre cas, nous voulons 30 lignes maximum, une pour chaque individu, et nous utiliserons 3 colonnes pour pouvoir afficher le nom, le sexe et le lien de parenté de chaque personne.

Lorsque vous affectez une valeur à un tableau à deux dimensions, vous indiquez à la fois la ligne et la colonne dans lesquelles vous souhaitez placer la valeur :

```
// Définissez la valeur dans la ligne 4, colonne 2
membresMenageArray(4, 2) = "Ceci est la quatrième ligne, 2e colonne ";
```

Dans nos exemples, nous allons parcourir le roster des membres du ménage et ajouter le nom, le sexe et le lien de parenté de chacun dans notre matrice. Puisque ce sera plus que quelques lignes de code, mettons cela dans une fonction à part et appelons-la ensuite à partir de notre fonction allera().

```
// Afficher la liste des membres du ménage dans une boîte de dialogue pour que
// l'intervieweur en choisisse un. Renvoie le numéro de ligne de la personne
// sélectionnée ou zéro si la boîte de dialogue a été annulée.
fonction choisiRosterMenage()
   numeric i;
   do numeric i = 1 while i <= totocc(MEMBRES_MENAGE_ROSTER)
      membresMenageArray(i, 1) = strip(NOM(i));
      membresMenageArray(i, 2) = getlabel(SEXE, SEXE(i));
      membresMenageArray(i, 3) = getlabel(LIEN_PARENTE, LIEN_PARENTE(i));
   enddo;
   membresMenageArray(i, 1) = ""; // marque fin
   numeric ligneSelectionne= showarray(membresMenageArray, titre("Nom", "Sexe",
                                       "Lien"));
   choisiRosterMenage= ligneSelectionne;
end;

// Fonction userbar pour accéder directement aux différentes parties
// du questionnaire.
fonction allera()
   numeric section= accept("Aller à quelle section ?",
                           "Identification",
                           "Membres du ménage",
                           "Caractéristiques individuelles");
   if section= 1 then
      move to IDENTIFICATION_FORM advance;
   elseif section = 2 then
      numeric individu= choisiRosterMenage();
      if individu > 0 then
         move to NOM(individu) advance;
      endif;
   elseif section = 3 then
      move to CARACTERISTIQUES_INDIVIDU_FORM advance;
```

```
    endif;
end;
```

Notez que pour le sexe et le lien de parenté comme nous voulons les libellés de l'ensemble de valeurs, nous utilisons donc la fonction getlabel(). Cette fonction renvoie le libellé de la valeur définie en tant que valeur alpha. Il prend le nom de la variable (ou de l'ensemble de valeurs) et la valeur à utiliser. Par exemple, getlabel(SEXE, 1) renverra "Masculin". Dans notre boucle, getlabel(SEXE, SEXE(i)) renverra le libellé du sexe du membre du ménage.

Chemin et "visualvalue"

Notre fonction allera() fonctionne lorsque nous sommes dans la section B ou C, mais lorsque nous sommes dans la section A, le sexe et le lien de parenté sont vides. Que se passe-t-il ? En mode "system control", CSEntry enregistre les variables activées et désactivées dans le "chemin". Les champs dont vous avez déjà saisie les valeurs sont considérés comme "sur le chemin", mais ceux qui ont été sautés, même s'ils avaient une valeur avant d'être sautés, sont considérés comme "hors chemin". Comme nous l'avons déjà vu, les variables qui sont "hors chemin" sont considérées comme vides (notappl) en logique. Il se trouve que les champs plus en avance sur le formulaire du champ actuel sont également considérés "hors chemin" jusqu'à ce que vous les traversiez. L'idée est que ces champs n'ont pas encore été validés en exécutant leurs préproc et postproc avec les valeurs actuelles de tous les champs précédents et ne peuvent donc pas être considérés comme définitifs. Cela a pour effet que les valeurs de tous les champs situés après le champ actuel du questionnaire sont notappl dans les expressions dans la logique.

Comme nous pouvons le constater dans notre code actuel, cela ne s'applique qu'aux éléments numériques. Notre code fonctionne très bien pour le champ alpha **NOM**.

Vous pouvez voir quels champs sont "sur le chemin" en regardant les couleurs :

- Vert : sur le chemin
- Gris foncé : sauté
- Blanc : pas encore rempli
- Gris clair : protégé

Notez que le schéma de couleur des champs est différent en mode "operator control".

Heureusement, nous pouvons obtenir la valeur des champs "hors chemin" en utilisant la fonction visualvalue(). Elle renvoie la valeur actuellement visible dans le champ, même si le champ ait été sauté ou en avance sur le champ actuel. En utilisant ceci pour le lien de parenté et le sexe dans notre fonction, nous obtenons :

```
fonction choisiRosterMenage()
   numeric i;
   do numeric i = 1 while i <= totocc(MEMBRES_MENAGE_ROSTER)
      membresMenageArray(i, 1) = strip(NOM(i));
      membresMenageArray(i, 2) = getlabel(SEXE, visualvalue(SEXE(i)));
      membresMenageArray(i, 3) = getlabel(LIEN_PARENTE,
                                          visualvalue(LIEN_PARENTE(i)));
   enddo;
   membresMenageArray(i, 1) = ""; // marque fin
   numeric ligneSelectionne= showarray(membresMenageArray, titre("Nom", "Sexe",
                                       "Lien"));
   choisiRosterMenage= ligneSelectionne;
end;
```

Maintenant le sexe et le lien de parenté sont affichés correctement même lorsqu'on lance notre fonction de navigation à partir de la section A. Il ne reste plus maintenant qu'à utiliser le libellé du lien de parenté correspondant au sexe du membre du ménage :

```
if visualvalue(SEXE(i)) = 1 then
  membresMenageArray(i, 3) = getlabel(LIEN_MALE_VS,
                             visualvalue(LIEN_PARENTE(i)));
else
  membresMenageArray(i, 3) = getlabel(LIEN_FEMALE_VS,
                             visualvalue(LIEN_PARENTE(i)));
endif;
```

Définition des valeurs de champ une fois seulement

Remplissons d'une manière automatique l'heure de début de l'interview. Nous pouvons protéger le champ et définir la valeur dans la preproc, tout comme nous l'avons fait avec le la variable **NUMERO_ORDRE**. Nous pouvons utiliser la fonction systime() qui renvoie l'heure actuelle sous la forme d'un nombre formaté en fonction de la spécification de format transmise.

```
PROC HEURE_DEBUT_INTERVIEW
preproc
// Pré-remplissez l'heure de début de l'interview avec l'heure actuelle.
HEURE_DEBUT_INTERVIEW = systime("HHMM");
```

Cela fonctionne la première fois que nous visitons le champ, mais que se passe-t-il lorsque nous revenons à la question une minute ou deux plus tard ? Nous ne voulons enregistrer cette valeur que la première fois que l'intervieweur entre dans le champ et une fois qu'il est défini, nous ne voulons plus qu'il soit modifié. Nous pouvons le faire en comparant la valeur de **HEURE_DEBUT_INTERVIEW** à vide (notappl) dans la preproc et en définissant la valeur sur systime uniquement si elle est vide.

```
PROC HEURE_DEBUT_INTERVIEW
preproc
// Pré-remplissez l'heure de début de l'interview avec l'heure actuelle.
if HEURE_DEBUT_INTERVIEW = notappl then
   HEURE_DEBUT_INTERVIEW = systime("HHMM");
endif;
```

Mais cela ne semble pas fonctionner. Pourquoi ? Est-ce que le champ **HEURE_DEBUT_INTERVIEW** se trouve sur le chemin lorsque nous sommes dans sa preproc ? Ce n'est pas. Nous devons arriver à la postproc pour que la variable soit sur le chemin. Cependant, nous pouvons utiliser visualvalue() pour obtenir la valeur du champ dans la preproc:

```
PROC HEURE_DEBUT_INTERVIEW
preproc
// Pré-remplissez l'heure de début de l'interview avec l'heure actuelle.
if visualvalue(HEURE_DEBUT_INTERVIEW) = notappl then
   HEURE_DEBUT_INTERVIEW = systime("HHMM");
endif;
```

Nous pouvons également remplir automatiquement l'heure de fin de l'interview en utilisant systime(). Contrairement à l'heure de début, nous devons le faire à la fin de l'entretien, c'est-à-dire dans la postproc du questionnaire.

```
PROC POPSTAN2020_QUEST
postproc
// Remplir l'heure de fin de l'interview
if HEURE_FIN_INTERVIEW = notappl then
   HEURE_FIN_INTERVIEW = systime("HHMM");
endif;
```

Nous devrions rendre ce champ protégé. Cependant, si nous le faisons, nous obtenons une erreur lorsque nous ne le remplissons pas dans la section A du questionnaire. Bien que nous ne voulions normalement pas laisser un champ vide, nous devons dans ce cas créer une exception. Nous pouvons le faire en modifiant le “validation method” dans les propriétés du champ en "Allow out of range without confirmation” (Permettre les valeurs hors limites sans confirmation). Avec ce paramètre, vous pouvez laisser un champ vide ou saisir une valeur en dehors de l’ensemble de valeurs sans erreur.

Après avoir capturé l'heure de début et de fin de l'interview sous forme d'heures et de minutes, nous pouvons calculer la durée totale de l'interview en soustrayant l'heure de début à l'heure de fin. Cependant, ce calcul est délicat car nous devons gérer le cas où les minutes de fin sont inférieures aux minutes de début. Ce serait plus facile si, au lieu d’utiliser systime(), nous utilisons la fonction timestamp (horodatage), qui donne le temps total en secondes depuis le 1er janvier 1970. Si nous soustrayons le timestamp de début du timestamp pris à la fine, nous obtenons le temps total d’interview en secondes. Ajoutons des nouvelles variables protégées pour l'horodatage de début, l'horodatage de fin et la durée de

l'interview. Etant donné que ces champs figureront dans la section A mais ne seront remplis qu'à la fin du questionnaire, définissez la méthode de validation sur "Allow out of range without confirmation" comme nous l'avons fait avec INTERVIEW_END_TIME.

```
PROC TIMESTAMP_DEBUT_INTERVIEW
preproc
// Préremplit l'heure de début de l'interview avec l'heure actuelle.
if visualvalue(TIMESTAMP_DEBUT_INTERVIEW) = notappl then
   TIMESTAMP_DEBUT_INTERVIEW = timestamp();
endif;

PROC POPSTAN2020_QUEST
postproc

// Définition de l'heure de fin de l'interview et de la durée totale
if TIMESTAMP_FIN_INTERVIEW= notappl alors
   TIMESTAMP_FIN_INTERVIEW= timestamp();
   DUREE_INTERVIEW_MINUTES =
                  (TIMESTAMP_FIN_INTERVIEW - TIMESTAMP_DEBUT_INTERVIEW ) /60;
endif;
```

Si nous n'avons pas besoin d'afficher les heures de début et de fin sur le formulaire, nous pouvons éviter d'utiliser la fonction visualvalue. Nous pouvons simplement supprimer les variables du formulaire et les conserver dans le dictionnaire. Lorsque nous définissons les valeurs de ces variables dans la logique et que le cas est enregistré dans le fichier de donnés, les valeurs sont enregistrées dans le fichier de données, tout comme les variables du formulaire. Toutefois, contrairement aux variables du formulaire, les variables qui ne figurent pas dans le formulaire sont toujours considérées comme "sur le chemin", vous n'avez donc pas à vous soucier de l'utilisation de visualvalue.

Exercices

1. Modifiez le texte de la question pour **E01** en disant "Un membre de ce ménage est-il décédé au cours des dernières années, c'est-à-dire depuis (année) ?" où année est remplacée par l'année de l'interview moins 5. Par exemple, si l'année de l'interview est 2018, il convient de lire "Un membre de ce ménage est-il décédé au cours des dernières années, c'est-à-dire depuis 2013 ?". Utilisez la date de l'interview de la question **A6** pour calculer la valeur à insérer.
2. Étendez la fonction allera() pour inclure les sections restantes du questionnaire (D à G).
3. Ajoutez à la barre utilisateur un bouton appelé "Résumé du ménage" qui, une fois cliqué, utilise la fonction errmsg pour afficher un message indiquant le nom du chef de ménage et le nombre de membres du ménage par sexe. Par exemple : "Chef de ménage : John Brown, nombre total de membres : 5, femmes : 2, hommes : 3". Bonus si vous pouvez obtenir que cela fonctionne lorsque vous cliquez sur le bouton à partir de la section A. Conseil : les fonctions count et seek ne fonctionnera pas avec visualvalue ; vous devez donc utiliser une boucle pour trouver le nombre d'hommes et de femmes.
4. Mettre en œuvre une vérification des valeurs minimales et maximales des biens possédés par le ménage dans la question G01. Utilisez la feuille de calcul de l'annexe 4 pour créer un fichier de recherche (lookup) contenant le code du bien, la valeur minimale et la valeur maximale. Utilisez la commande loadcase avec ce fichier lookup pour rechercher les valeurs minimales et maximales du bien sélectionné et afficher un message d'erreur si la valeur par unité saisie est inférieure à la valeur minimale ou supérieure à la valeur maximale. Cela devrait être un contrôle non-bloquant.
5. Remplissez automatiquement la date de l'interview à l'aide de la commande sysdate() et protégez le champ. Assurez que la date ne soit pas modifiée si on revien dans le champ un autre jour.
6. Ajoutez la question A10, statut de l'interview, au dictionnaire et au formulaire. Nous voulons répondre à cette question au début de l'entretien. Si la réponse est le code 2 (pas de membre du ménage à la maison), 3 (vacant) ou 4 (refusé), puis terminer immédiatement le questionnaire sans entrer dans les questions suivantes. Cependant, si un répondant souhaite faire l'interview, le statut de l'interview doit être défini sur 5 (partiellement rempli) jusqu'à ce que le questionnaire complet soit rempli, puis sur 1 (complet). Pour mettre cela en œuvre, conservez la question dans sa position actuelle dans le questionnaire, mais utilisez un ensemble de valeur dynamique définie pour limiter les options, de sorte que le résultat de l'interview ne puisse pas être défini comme terminé tant que l'intégralité du questionnaire n'a pas été complétée. La première fois que le champ est saisi (avant toute valeur), l'ensemble de valeur doit être : 2 (pas de membre du ménage à la maison), 3 (vacant), 4 (refusé) 5 (Poursuivre l'interview). Si l'enquêteur choisit 2, 3 ou 4, mettez fin à l'interview, sinon, s'il choisit 5, passez au champ suivant. À la fin de l'interview (postproc du niveau), si la valeur est actuellement 5, définissez-la sur 1 (complété). Si l'intervieweur revient à A9 après que le champ ait été défini sur 1 (complété), affichez l'ensemble de valeurs définie telle qu'il est sur le questionnaire.

Session 07 : Multimédia et GPS

À la fin de cette session, les participants seront en mesure de :

- Utiliser des images dans l'ensemble de valeurs
- Afficher des documents externes et des fichiers multimédias à partir d'une application de saisie de données
- Prendre des photos dans une application de saisie de données
- Prendre les coordonnées GPS dans une application de saisie de données
- Afficher les points prises avec le GPS sur une carte
- Générez et affichez des rapports à partir d'une application de saisie de données.

Les images dans l'ensemble de valeurs

En plus de l'affichage de texte dans un ensemble de valeurs, vous pouvez également afficher des images. Pour ajouter une image à un ensemble de valeurs, sélectionnez la variable dans l'éditeur de dictionnaire et cliquez sur le bouton "..." à côté de la valeur. Naviguez pour choisir un fichier image. CSPro utilise les formats de fichier image standard : png, jpeg, bmp… Nous vous recommandons d'utiliser jpeg car ces fichiers sont généralement plus légers. Les images ne sont pas ajoutées au fichier pen par défaut. Vous devez donc les copier sur l'appareil lorsque vous copiez les fichiers pen et pff.

Ajoutons des images pour les différents types de toit dans la question **F08**. Copiez le dossier ImagesToit dans le répertoire de l'application. Ouvrez maintenant l'ensemble de valeurs de la variable **F08** et ajoutez chaque image à sa valeur respective. Nous n'avons pas d'image pour autre (à préciser), nous la laisserons donc sans image. Exécutez l'application sur Windows et sur le mobile et voyez les images. Notez que sur mobile, nous devons copier les images sur le périphérique, car elles ne font pas partie du fichier pen par défaut.

Le dossier de ressources

Au lieu de copier les fichiers image sur le périphérique mobile chaque fois qu'ils sont mis à jour, nous pouvons les mettre dans le répertoire de ressources de l'application. Tous les fichiers du dossier de ressources sont intégrés au fichier pen et extraits sur le périphérique lors de l'exécution de l'application. Cela facilite la distribution d'une application avec des fichiers supplémentaires. Créons un dossier de ressources, ajoutons-le à notre application et insérons nos fichiers image. Pour créer le fichier de ressources, créez simplement un nouveau dossier dans l'explorateur Windows. Nous allons créer le dossier "ressources" dans le dossier des ménages. Ensuite, dans CSPro, sélectionnez “Add Files…” dans le menu “File”, choisissez «Resource Folder» et accédez au nouveau dossier. Maintenant, placez tout le dossier ImagesToit dans le dossier de ressources et publiez encore le fichier pen.

Afficher des fichiers externes

La fonction view() lance des fichiers avec leur visualiseur par défaut. Elle prend le chemin du fichier à afficher. Le périphérique doit être doté d'une application capable d'afficher le type de fichier spécifié. La fonction détermine quelle application à lancer pour afficher le fichier en fonction de son extension. La plupart des appareils ont des applications intégrées pour afficher la plupart des formats de fichiers photo, musique et vidéo, mais tous ne disposent pas, par exemple, d'un lecteur intégré de fichier PDF. Si vous n'en avez pas sur votre appareil, vous pouvez toujours en télécharger un. La fonction view() peut également afficher des sites web si vous lui transmettez une URL.

```
view("/mnt/sdcard/csentry/picture.jpg");
view("/mnt/sdcard/csentry/audio.mp3");
view("/mnt/sdcard/csentry/movie.mp4");
view("/mnt/sdcard/csentry/document.pdf");
view("https://www.census.gov/data/software/cspro.html");
```

Ajoutons un bouton à la barre d'utilisateur pour afficher le manuel d'interview afin d'aider l'intervieweur :

```
// Afficher le manuel de l'intervieweur
function afficherManuelInterviewer()

view("/mnt/sdcard/csentry/Popstan2020/menage/ressources/ManuelInterviewer.pdf");
end;
```

Nous pouvons rendre notre code un peu plus flexible en utilisant la fonction pathname() pour obtenir le répertoire dans lequel l'application est stockée. Ainsi, si quelqu'un copie l'application dans un autre dossier du périphérique, il fonctionnera toujours :

```
// Affiche le manuelle de l'interviewer
function afficherManuelInterviewer()
   view(pathname(Application) + "ressources/ManuelInterviewer.pdf");
end;
```

Prendre des photos sur Android

Pour prendre une photo sur Android, vous pouvez utiliser la fonction execsystem, qui est un autre moyen de lancer des applications externes. Pour utiliser execsystem et prendre une photo, transmettez-lui une chaîne de caractères commençant par "camera:" suivie du chemin complet du fichier dans lequel enregistrer la photo:

```
execsystem("camera:/mnt/sdcard/csentry/photo.jpg");
```

Prendre des photos est possible uniquement sur Android. La fonction execsystem sur Windows ne fonctionne pas avec la caméra.

Notez que les photos ne sont pas enregistrées dans le fichier de données de l'application. Vous devez les enregistrer dans un dossier / fichier sur le périphérique. Le nom du fichier que vous utilisez devrait permettre de relier facilement la photo à l'interview pour laquelle elle a été prise. Pour ce faire, nous pouvons nommer le fichier de la photo en fonction des identifiants du questionnaire :

```
string nomFichierPhoto = pathname(application) +
        maketext("../Donnees/PhotosMenage/photo%v%v%v%v.jpg",
                PROVINCE, DISTRICT, ZONE_DENOMBREMENT,
                NUMERO_MENAGE);
execsystem("camera:" + nomFichierPhoto);
```

Notez que nous mettons la photo dans un sous-répertoire du répertoire de données nommé PhotosMenage. Nous devons créer ce dossier sur la tablette sinon les photos ne seront pas enregistrées.

Ajoutons l'option de prendre une photo du ménage dans la section A. Plutôt que d'ajouter un bouton de la barre d'utilisateur qui serait facile à oublier pour l'interviewer, nous allons en faire une partie du questionnaire en ajoutant une variable fictive au dictionnaire. Dans la postproc de la variable, nous appellerons execsystem() pour prendre la photo. Nous pouvons ajouter un code refusé si le ménage ne veut pas être pris en photo.

```
PROC PHOTO

if PHOTO = 1 then
   // prendre une photo
   string nomFichierPhoto = pathname(application) +
        maketext("../Donnees/PhotosMenage/photo%v%v%v%v.jpg",
                PROVINCE, DISTRICT, ZONE_DENOMBREMENT,
                NUMERO_MENAGE);
   execsystem("camera:" + nomFichierPhoto);
elseif PHOTO = 9 alors
   // refusé, passe au champ suivant sans photo
endif;
```

Cela fonctionne la première fois dans le questionnaire, mais si nous revenons à la question de la photo une deuxième fois, nous sommes obligés de prendre une autre photo. Nous pouvons éviter cela en ajoutant une autre option à l'ensemble de valeurs "Conserver la photo" que l'intervieweur sélectionnera une fois satisfait de la photo prise. Lorsque "conserver la photo" est sélectionné, nous ne lancerons pas l'appareil photo. Nous pouvons ajouter une autre option "Afficher photo" pour leur permettre de regarder la photo prise.

```
PROC PHOTO

string nomFichierPhoto = pathname(application) +
        maketext("../Donnees/PhotosMenage/photo%v%v%v%v.jpg",
                PROVINCE, DISTRICT, ZONE_DENOMBREMENT,
                NUMERO_MENAGE);

if PHOTO = 1 then
   // prendre / reprendre la photo
   execsystem("camera:" + nomFichierPhoto);
   // reenter pour que línterviewer puisse reprendre si ce n'est pas bien
   reenter;
elseif PHOTO = 2 then
   // voir la photo existante
   view(nomFichierPhoto);
   // reenter pour que línterviewer puisse reprendre si ce n'est pas bien
   reenter;
elseif PHOTO = 3 alors
   // Conserver la photo - permet de passer au champ suivant
elseif PHOTO = 9 alors
   // Aucune photo (refusée)
   // Supprime la photo si elle existe.
   filedelete(nomFichierPhoto);
endif;
```

Cela fonctionne sauf qu'il est possible pour l'intervieweur de sélectionner "Conserver la photo" même s'il n'y a pas de photo existante et d'essayer de visualiser une photo qui n'existe pas. Nous pouvons résoudre ce problème en affichant uniquement les options "Conserver la photo" et "Afficher la photo" dans l’ensemble de valeurs définie si la photo existe déjà.

```
PROC PHOTO
onfocus
string nomFichierPhoto = pathname(application) +
        maketext("../Donnees/PhotosMenage/photo%v%v%v%v.jpg",
                PROVINCE, DISTRICT, ZONE_DENOMBREMENT,
                NUMERO_MENAGE);

// Afficher l’ensemble de valeurs avec l’option de conserver la photo
// uniquement si la photo existe.
if fileexist(nomFichierPhoto) then
   setvalueset(PHOTO, PHOTO_VS_PHOTO_PRISE);
else
   setvalueset(PHOTO, PHOTO_VS_AUCUN_PHOTO);
endif;
```

Lecture de coordonnees GPS

Pour prendre un point GPS, vous devez d'abord démarrer le matériel GPS :

```
gps(open);
```

Ensuite, vous demandez une lecture GPS en lui donnant un délai d'expiration en secondes et une précision désirée facultative en mètres :

```
if gps(read, 60, 10)  then // Lire jusqu'à 60 secondes ou jusqu'à une précision de 10m
   errmsg("La latitude est %f, la longitude est %f", gps(latitude), gps(longitude));
else
   errmsg("Le signal GPS n'a pas pu être acquis");
endif;
```

Enfin, vous fermez le GPS :

```
gps(close);
```

En plus d'interroger la latitude et la longitude GPS, vous pouvez également interroger la précision, le nombre de satellites et l'altitude (bien que sur Android, l'altitude n'est pas très précise). Voir l'aide sur GPS pour plus de détails.

La longitude et la latitude indiquées sont en degrés dans le datum WGS84 et sont renvoyées sous forme de nombres décimaux.

Pour enregistrer les coordonnées GPS dans notre fichier de données, nous devons ajouter les variables appropriées à notre dictionnaire. Pour stocker les coordonnées de manière à pouvoir couvrir toute la terre avec suffisamment de précision, il convient de laisser l'espace pour trois chiffres avant la virgule, au moins cinq chiffres après la virgule, 1 chiffre pour la virgule même et un chiffre pour le signe moins. Ce qui fait un total minimum longueur de dix. Ajoutez les variables LATITUDE et LONGITUDE au dictionnaire et à la section A sur les formulaires.

Ajoutons un bouton utilisateur pour mettre à jour les coordonnées GPS du ménage :

```
// Prendre la position du ménage à partir du GPS
function lireGPS()
   gps(open);
   if gps(read, 60, 10)  then // Lire jusqu'à 60 secondes ou jusqu'à une précision
                              // de 10m
      LATITUDE = gps(latitude);
      LONGITUDE = gps(longitude);
   else
      errmsg("Le signal GPS n'a pas pu être acquis");
   endif;
   gps(close);
```

```
end;
```

Affichage des points GPS sur une carte

Si nous voulons être plus sophistiqués, nous pouvons afficher le point enregistré sur une carte afin que l'intervieweur puisse le visualiser et décider ensuite de l'utiliser ou non. Nous pouvons le faire en utilisant une variable de type map. Comme les variables de type valueset, les variables de type map ont des fonctions qui peuvent s'appeler à travers du point ("."). Par exemple, pour ajouter un marqueur à une carte, appelez :

```
map carte;
carte.addMarker(longitude, latitude);
```

Pour afficher la carte, appelez la fonction show(). Cela affiche une carte "Google Maps" contenant le marqueur qui a été ajouté.

Nous pouvons utiliser une variable de type map pour afficher une carte comportant un marqueur à la position GPS que nous venons de prendre :

```
// Prendre la position du ménage à l'aide du GPS
function lireGPS()
   gps(open);
   if gps(read, 60, 10)  then // Lire jusqu'à 60 secondes ou jusqu'à une précision
                              // de 10m
      // Afficher la carte pour que l'intervieweur puisse voir le résultat
      map carte;
      carte.addMarker(gps(latitude), gps(longitude));
      carte.show ();
      if accept("Enregistrer ce résultat", "Oui", "Non") = 1 then
         LATITUDE = gps(latitude);
         LONGITUDE = gps(longitude);
      endif;
   else
      errmsg("Le signal GPS n'a pas pu être acquis");
   endif;
   gps(close);
end;
```

Les variables de type map ont de nombreuses fonctions qui vous permettent de personnaliser la carte, les marqueurs, ajouter des boutons et lancer des fonctions logiques lorsque vous appuyez sur les marqueurs. Vous pouvez également utiliser vos propres fonds de cartes pour les situations dans lesquelles vous ne disposez pas de connexion Internet. Voir l'aide pour le type map pour en avoir plus de détails. Notez que les fonctions de map ne fonctionnent que sur Android.

Afficher la liste des cas sur la carte

Si vous avez des variables de latitude et de longitude dans votre dictionnaire de données définies pour chaque ménage, vous pouvez configurer l'écran de la liste des cas pour afficher les ménages sur une carte sous Android. Pour ce faire, ouvrez la boîte de dialogue “Mapping Options” du menu “Options” de CSPro. Cochez la case "Show the case listing on a map" et choisissez les variables de latitude et de longitude de votre dictionnaire. Lorsque vous exécutez l'application sur Android, l'écran de la liste des cas s'affiche avec une carte sur laquelle sera affiché un marqueur pour chaque ménage. Tapez sur un marqueur pour lancer la saisie de données pour le ménage y associé.

Travail de groupe

Créez une application CSPro pour la fiche de numérotation des ménages. Placez votre application dans le dossier Popstan2020/NumerotationMenages. Mettez chaque ménage dans un cas à part. C’est-à-dire de n'utilisez pas un roster / grille pour essayer de faire correspondre au questionnaire papier. Lors de la création du dictionnaire, utilisez "NU_" comme préfixe pour tous les noms des variables afin d'éviter les conflits avec les noms figurant dans le dictionnaire des ménages. Il n'est pas nécessaire de valider les identifiants à l'aide d'un fichier de recherche, comme nous l'avons fait pour le questionnaire ménage. Nous allons pré-remplir les identifiants dans le programme de menu demain. Pour chaque ménage, enregistrez automatiquement les coordonnées GPS dans le champ de latitude. Donnez à l'intervieweur la possibilité de prendre une photo du ménage mais permettez le refus. Nommez la photo en fonction des identifiants du ménage et placez les photos dans le répertoire Popstan2020/Donnees/PhotosNumerotation. Utilisez la boîte de dialogue “Mapping Options” pour afficher la liste des cas sur une carte. Sortez de la salle et testez votre application sur une tablette.

Rédaction et affichage de rapports

Avec la fonction view(), vous pouvez afficher des fichiers texte sur votre appareil. Nous pouvons utiliser ceci pour afficher les documents texte que nous écrivons à partir de CSPro. Pour écrire des fichiers à partir d’une application CSEntry, nous devons d’abord déclarer une variable de type file (fichier) :

```
file ficheTemp;
```

Ensuite, nous pouvons utiliser les commandes setfile(), filewrite() et close() pour ouvrir, écrire et fermer le fichier. Créons un bouton dans la barre utilisateur pour rédiger un rapport texte simple qui répertorie les membres du ménage :

```
// Écrire et afficher un rapport de résumé du ménage
function de afficherRapportMenage()
    string nomFichierRapport = pathname(application) +  "rapport.txt";
    file ficheTemp;
    setfile(ficheTemp, nomFichierRapport, create);
    filewrite(ficheTemp, "Rapport récapitulatif du ménage");
    filewrite(ficheTemp, "------------------------");
    filewrite(ficheTemp, "");
    filewrite(ficheTemp, "Province %d District %d ZD %d Numéro du ménage %d",
                    visualvalue(PROVINCE), visualvalue(DISTRICT),
                    visualvalue(ZONE_DENOMBREMENT),
                    visualvalue(NUMERO_MENAGE));
    filewrite(ficheTemp, "");
    filewrite(ficheTemp, "Membres du ménage:");
    filewrite(ficheTemp, "");
    filewrite(ficheTemp, "Nom Sexe Âge Lien de parenté");
    filewrite(ficheTemp, "--- ---- --- ---------------");
    do numeric i = 1 while i <= totocc(MEMBRES_MENAGE_ROSTER)
       filewrite(ficheTemp, "%s %-6s %3d %s",
                       NOM(i),
                       getlabel(SEXE, visualvalue(SEXE(i)))),
                       visualvalue(AGE(i)),
                       getlabel(LIEN_PARENTE, visualvalue(LIEN_PARENTE(i))));
    enddo;
    close(ficheTemp);

    view(nomFichierRapport );
end;
```

Si vous savez rédiger les pages web en HTML, vous pouvez l'utiliser pour les rapports au lieu des fichiers textes et bénéficier d'un formatage plus agréable en plus de l'intégration des images. CSPro possède également un moteur de modèles intégré pour générer des rapports HTML à l'aide de modèles. Voir "Templated reports" dans l'aide de CSPro pour plus de détails.

Exercices

1. Ajoutez un bouton dans la barre d'utilisateur pour afficher votre site web à l'aide de la fonction view().
2. Ajoutez les informations supplémentaires suivantes au rapport menáge. a. Nombre total de membres du ménage, total d'hommes, total de femmes. b. Liste des biens du ménage de la section G avec la quantité. Ne listez pas les biens dont la quantité est zéro.

Session 08 : Programmes de menu

À la fin de cette session, les participants seront en mesure de :

- Concevoir les écrans d'un programme de menu
- Utiliser des variables de type pff pour lancer une application CSPro à partir d'une autre application
- Créer un programme de menu pour lancer le programme de saisie de données principal
- Créer des menus dynamiques avec des ensembles de valeurs dynamiques

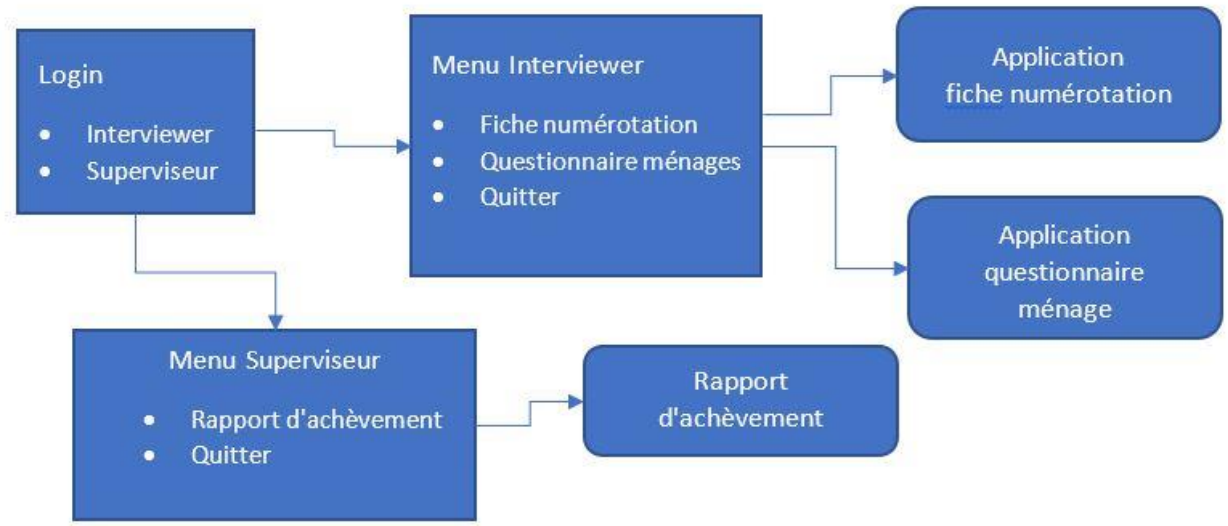
Qu'est-ce qu'un programme de menu ?

Un programme de menu est une application de saisie de données CSPro utilisée pour gérer le processus de travail de la saisie de données. Un programme de menu n'est pas utilisé pour saisir des données de l'interview proprement dites. Au lieu de cela, il lance d'autres programmes de saisie de données pour les interviews. Les programmes de menu ont généralement les fonctions suivantes :

- Lancer d'autres applications CSPro pour la collecte de données (souvent des éléments d'identification sont pré-remplis par le menu)
- Afficher des rapports sur le progrès de l'opération, et les statistiques sommaires
- Gérer l'accès des utilisateurs par les noms d'utilisateur / mot de passes
- Gérer les affectations des ménages et des zones de travail
- Transférer les données au serveur centrale ou au superviseur

Conception des écrans du menu

Une fois que vous avez choisi les fonctions que vous souhaitez avoir dans votre programme de menus, l'étape suivante consiste à concevoir les écrans et les liens entre les écrans. Cela se fait très facilement comme un diagramme comme celui ci-dessous pour un programme de menu assez simple.



Création du dictionnaire et des formulaires

Une fois que vous avez les écrans, l'étape suivante consiste à créer une nouvelle application de saisie de données pour votre programme de menu. Appelons notre application “menu” et mettons-le dans le dossier *Popstan2020/Menu*.

Chaque écran du menu sera une variable différente dans le dictionnaire. L’ensemble de valeurs pour la variable indiquera les choix de menu disponibles à l'interviewer pour cet écran. Dans l'exemple de menu ci-dessus, nous aurons trois variables :

- LOGIN (ensemble de valeurs : intervieweur - 1, superviseur - 2)
- MENU_PRINCIPAL_INTERVIEWER (ensemble de valeurs : fiche de numérotation - 1, questionnaire des ménages - 2, quitter - 9)
- MENU_PRINCIPAL_SUPERVISEUR (ensemble de valeurs : rapport de d’achèvement- 1, quitter- 9).

Nous conserverons la variable identifiant du nouveau dictionnaire que CSPro crée pour nous par défaut. Le dictionnaire du programme de menu n'a pas besoin d'identifiant car nous n'allons pas sauvegarder nos choix de menu dans le fichier de données. Cependant, CSPro nous demande au moins un identifiant, alors nous allons le conserver.

Créez les éléments dans le dictionnaire, puis créez un formulaire et déposez-les sur le formulaire. Ne déposez pas l'identifiant sur le formulaire. Nous pouvons le laisser en dehors du formulaire car nous n'enregistrons jamais de données à partir du programme de menu. Contrairement aux applications classiques de saisie de données, les programmes de menu ont tendance à ne pas avoir un chemin linéaire. En conséquence, l'ordre des variables sur le formulaire est moins important. Nous allons utiliser les commandes skip et reenter pour passer d’un menu à l’autre.

Logique de programme de menu

La logique de traitement du choix de menu pour chaque écran se trouve dans le post-proc du champ de menu. Par exemple, pour le champ LOGIN dans notre exemple, nous aurions la logique suivante :

```
PROC LOGIN

// Accédez au menu approprié pour le rôle choisi
if $ = 1 then
   skip to MENU_PRINCIPAL_INTERVIEWER;
else
   skip to MENU_PRINCIPAL_SUPERVISEUR;
endif;
```

Si l'utilisateur choisit de travailler en tant que l'intervieweur, nous passons au champ du menu principal de l'intervieweur pour afficher le menu de l'intervieweur. Sinon, nous passons au champ du menu principal du superviseur pour afficher le menu du superviseur.

La logique du champ du menu de l'intervieweur est similaire. Ici, nous allons créer des fonctions définies par l'utilisateur pour lancer la fiche de numérotation ou le questionnaire ménage.

```
PROC MENU_PRINCIPAL_INTERVIEWER

postproc
// Gère le choix de menu
if $ = 1 then
   lancerFicheNumerotation();
elseif $ = 2 then
   lancerQuestionnaireMenage();
elseif $ = 9 alors
   // Quitter
   stop(1);
endif;

// Afficher à nouveau le menu de l'intervieweur
reenter;
```

Il est important de veiller à ce que, après la postproc du champ de menu, nous ne laissions pas CSEntry passer au champ suivant, sinon après que l'enquêteur lance l'application de numérotation, il se retrouverait dans le champ suivant, qui est dans ce cas le menu du superviseur. Pour éviter cela, nous mettons un reenter à la fin de la postproc afin de revenir dans le même champ et afficher encore le menu de l'interviewer.

Il semble bizarre que lorsque nous revenons dans un menu, le choix précédent soit toujours sélectionné. Nous pouvons éviter cela en supprimant le choix dans la proc onfocus du champ.

```
onfocus
// Effacer le choix précédent
$ = notappl;
```

Le menu du superviseur est similaire à celui de l'intervieweur :

```
PROC MENU_PRINCIPAL_SUPERVISEUR

onfocus
// Effacer le choix précédent
$ = notappl;

postproc
// Gère le choix de menu
if $ = 1 then
    afficherRapportAchevement();
elseif $ = 9 alors
    // Quitter
    stop(1);
endif;

// Afficher à nouveau le menu du superviseur
reenter;
```

Lancer une application CSPro à partir d'une autre

Ecrivons maintenant la fonction qui lancera le programme de saisie de données du questionnaire ménage. Nous pouvons lancer d'autres programmes CSPro à partir de notre programme de saisie de données en utilisant une variable de type pff. Une variable pff représente un fichier pff. Vous pouvez charger une variable pff avec le contenu d'un fichier pff existant à l'aide de la fonction load() et lui transmettre le chemin du fichier. Vous pouvez ensuite exécuter l'application CSPro associée en appelant la fonction exec(). Cela démarrera l'application en utilisant les paramètres du fichier pff.

La logique suivante lancera une application de saisie de données à l'aide du fichier pff Popstan2020.pff situé dans le répertoire frère du répertoire du programme de menu.

```
function lancerQuestionnaireMenage()
    pff pffMenage;
    pffMenage.load(pathname(application) + "../Menage/Popstan2020.pff");
    pffMenage.exec();
end;
```

Ceci démarrera l'application de saisie de données et quittera immédiatement le menu. Le ".." dans le chemin menant à Popstan2020.pff indique à CSEntry d'accéder à un répertoire supérieur, c'est-à-dire d'accéder au répertoire parent du répertoire Menu et, de là, à Menage/Popstan2020.pff.

Retour au programme de menu après la saisie

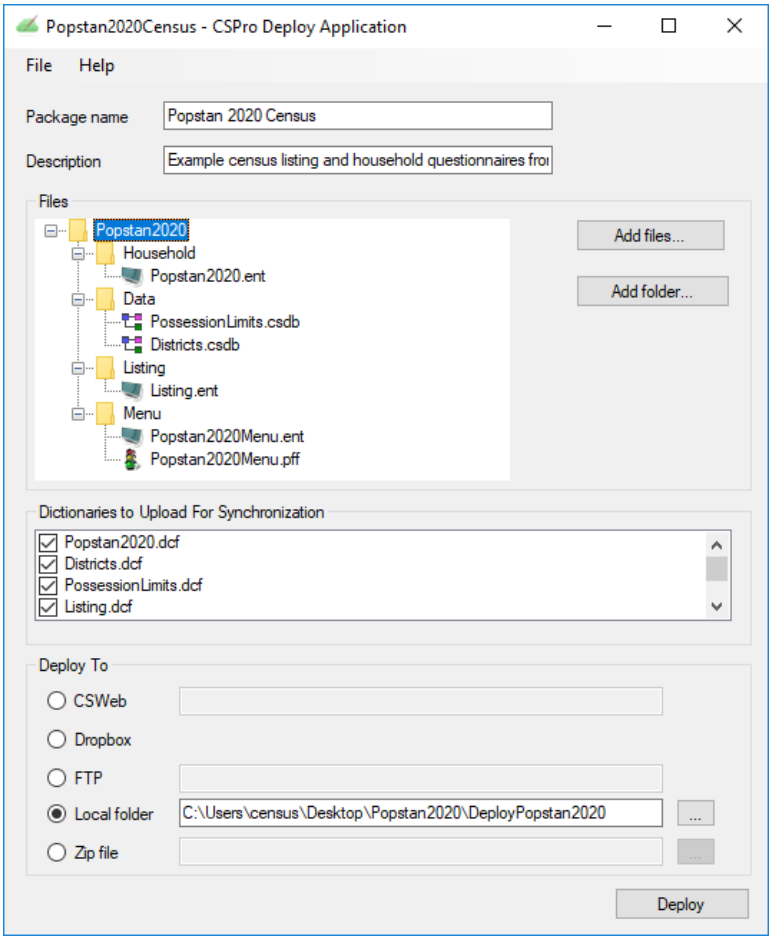
Lorsque nous quittons l'application questionnaire du ménage, nous souhaitons revenir au menu, mais le programme de menu s'est arrêté lorsque nous avons lancé l'application ménage. Pour le redémarrer, nous ajoutons un paramètre au fichier pff pour lui indiquer de démarrer le programme de menu à sa fermeture. Faites un clic droit sur Popstan2020.pff et choisissez "Modifier". Cela fera apparaître l'éditeur de fichiers pff. Dans le menu "Add", ajoutez une nouvelle entrée **On Exit PFF**. Utilisez le bouton "Browse" dans cette entrée pour localiser le fichier Menu.pff qui remplira son chemin d'accès.

Déploiement de plusieurs applications sur mobile

Lorsque vous copiez votre application sur votre téléphone ou votre tablette, il est important de conserver la même structure de dossiers que celle que vous avez sur votre PC. Dans notre cas, nous devons créer un dossier de menu distinct contenant les fichiers pen et pff du menu et ce dossier de menu doit se trouver dans le même répertoire parent que le dossier Menage afin que nous puissions utiliser le chemin "../Menage/" dans l'application menu pour indiquer le dossier contenant l'application d'entrée que nous lançons. Lorsque vous avez plusieurs applications comme celle que nous avons, il peut être fastidieux de créer les fichiers pen pour chaque application, puis de copier uniquement les fichiers pen et pff sur le périphérique mobile.

Pour simplifier le déploiement, utilisez l'outil "Deploy Applications" du menu Tools. Cet outil automatise le processus de publication et de déploiement. Faites glisser tout le dossier Popstan2020 dans la fenêtre "Files". L'outil ajoute toutes les applications de saisie (fichiers .ent) et tous les fichiers .pff situés dans ce dossier et ses sous-dossiers à la liste des fichiers à déployer. Par défaut, il n'ajoute pas de fichiers de données car ceux-ci ne servent souvent qu'à des tests. Cependant, pour notre application, nous devons ajouter les fichiers lookup pour les districts et les limites des valeurs des biens. Nous pouvons le faire en faisant glisser les fichiers csdb individuels sur la fenêtre des fichiers.

Une fois les fichiers ajoutés, inscrivez le nom et une description facultative pour l'application. Sous *Deploy To,* choisissez *Local Folder* et cliquez sur "..." pour choisir le dossier dans lequel déployer. Cliquez sur le bouton *Deploy*. Cela créera automatiquement des fichiers .pen pour chaque application de saisie de données et les copiera, ainsi que tous les autres fichiers de la fenêtre *Files*, dans le dossier de déploiement choisi. Le dossier de déploiement peut ensuite être copié directement sur le périphérique mobile. La structure des répertoires du dossier de déploiement correspond à la structure des répertoires que nous utilisons pour le développement et les tests sur le PC. Il n'y aura donc pas de problèmes de chemins.



Une fois que vous avez configuré le déploiement pour la première fois, vous pouvez enregistrer le fichier de spécification de déploiement afin de simplifier le processus pour les déploiements futurs.

Lors de la prochaine session, nous allons apprendre à utiliser cet outil pour déployer des applications sur un serveur en vue d'un déploiement via Internet.

Astuces pour les programmes de menu

Actuellement, le programme de menu apparaît dans la liste des applications sous Android sous le nom de "Menu" au lieu de quelque chose comme "Recensement de Popstan 2020". En outre, lorsque nous tapons sur Menu, nous devons taper sur "Nouveau cas", ce qui n'a aucun sens pour un menu. Nous pouvons résoudre ces deux problèmes en modifiant le fichier pff du menu. Faites un clic droit sur le fichier Menu.pff et choisissez "Modifier" pour lancer l'éditeur de pff. Changez le "Start Mode" en "Add" afin que nous n'ayons pas à appuyer sur "Nouveau cas". Entrez "RGPH 2020 Popstan" pour la description afin de changer ce qui apparaît dans la liste des applications. Pendant que nous y sommes, ouvrez le programme Menu dans le concepteur CSPro et dans les options de saisie de données, désactivez l'affichage de l'arborescence des cas car celui-ci n'est pas utile pour les programmes de menus. Enfin, dans les options de saisie des données, cochez la case "Automatically advance on selection". De cette façon, l'intervieweur n'aura pas à appuyer sur la touche suivant pour passer d'un champ à un autre dans le programme du menu.

Maintenant que nous avons le programme de menu pour lancer le programme de saisie de questionnaire ménage, nous ne voulons pas que l'application de questionnaire ménage apparaisse dans la liste des applications sur le périphérique mobile. Par défaut, CSPro crée la liste des applications à partir de tous les fichiers pff du répertoire CSEntry sur le périphérique. Pour empêcher la liste d'une application, vous pouvez modifier le fichier pff dans l'éditeur de pff et choisir "Hidden by default" ou "Never" pour l'option "Show in Application Listing".

Les menus dynamiques

Lorsque l'interviewer lance l'application de saisie de données, il peut actuellement saisir n'importe quel numéro de ménage, mais nous souhaitons le limiter à l'interview des ménages déjà numérotées dans la fiche de numérotation. Nous pouvons le faire en affichant les ménages du fichier de numérotation dans un ensemble de valeurs dynamique et en laissant l'intervieweur choisir parmi eux.

Afin de lire le fichier de numération, nous devons ajouter le dictionnaire de numération en tant que dictionnaire externe dans l'application de menu. Nous devrons également ajouter un nouvel élément de menu pour choisir le ménage. Ce sera une nouvelle variable dans le dictionnaire nommée **CHOISIR_MENAGE**. Nous aurons 9 chiffres pour contenir l'ensemble complet des identifiants (province, district, ZD, et numéro de ménage). Ce sera un champ alpha car il sera plus facile de travailler avec le type alpha plus tard. Le menu de l'intervieweur devra être modifié pour aller à ce nouveau champ au lieu d'appeler lancerQuestionnaireMenage().

```
PROC MENU_PRINCIPAL_INTERVIEWER

postproc
// Gère le choix de menu
if $ = 1 then
    lancerFicheNumerotation();
elseif $ = 2 then
   skip to CHOISIR_MENAGE;
elseif $ = 9 alors
    // Quitter
    stop(1);
endif;

// Afficher à nouveau le menu de l'intervieweur
reenter;
```

Dans la proc onfocus de **CHOISIR_MENAGE,** nous devons parcourir tous les cas du fichier de numération. Nous pouvons utiliser une boucle forcase pour boucler sur les ménages dans le fichier et construire l'ensemble de valeurs :

```
PROC CHOISIR_MENAGE
onfocus

valueset menagesVS;

// Parcourir tous les ménages du fichier de numérotation
// pour créer un ensemble de valeurs dynamiques des ménages
forcase NUMEROTATION_DICT do

   // Les codes sont les identifiants de ménage concaténés
   numeric code = maketext("%v%v%v%v",
      NU_PROVINCE, NU_DISTRICT, NU_ZONE_DENOMBREMENT,
      NU_NUMERO_MENAGE);

   // Les étiquettes ont un numéro de maison et le nom du CM
   string libelle = maketext("%v: (%s)", NU_NUMERO_MENAGE,
                     strip(NU_NOM_CHEF_MENAGE));

   menagesVS.add(libelle, code);
endfor;

setvalueset($, menagesVS);
```

Lorsque nous exécutons ceci, nous devrions voir une liste des ménages du fichier de numération en tant qu'ensemble de valeurs de la variable **CHOISIR_MENAGE**.

Bien que l'ensemble de valeurs dynamiques fonctionne correctement lorsque des ménages ont été numérotées, si le fichier de numération est vide, nous obtenons un ensemble de valeurs vide. Au lieu de cela, nous devrions afficher un message d'erreur et revenir au menu principal. Nous pouvons déterminer si l'ensemble de valeurs est vide en vérifiant la longueur de la liste de codes de l'ensemble de valeurs. Vous pouvez accéder aux listes de codes et de libellés d'un ensemble de valeurs en mettant le nom de la variable suivi d'un point (".") et puis "codes" ou "labels". Les propriétés des codes et labels de l'ensemble de valeurs sont des listes CSPro donc vous pouvez retrouver leurs tailles à l'aide de la fonction length().

```
PROC CHOISIR_MENAGE
onfocus

valueset menagesVS;

// Parcourir tous les ménages du fichier de numérotation
// pour créer un ensemble de valeurs dynamiques des ménages
forcase NUMEROTATION_DICT do

   // Les codes sont les identifiants de ménage concaténés ensemble
   numérique code = maketext("%v%v%v%v",
      NU_PROVINCE, NU_DISTRICT, NU_ZONE_DENOMBREMENT,
      NU_NUMERO_MENAGE);

   // Les étiquettes ont un numéro de maison et le nom de la tête
   string libelle = maketext("%v: (%s)", NU_NUMERO_MENAGE,
                     strip(NU_NOM_CHEF_MENAGE));

   menagesVS.add(libelle, code);
endfor;

if length(menagesVS.codes) = 0 then
   errmsg("Aucun ménage n'a été numérotée. Veuillez d'abord remplir la fiche des
ménages avant de procéder à l'interview.");
   reenter MENU_PRINCIPAL_INTERVIEWER;
endif;

setvalueset($, menagesVS);
```

Définition de l'attribut "Key" dans le pff

L'étape suivante consiste à gérer le choix de menu dans la postproc du champ **CHOISIR_MENAGE** afin de lancer le programme de saisie de données sur le ménage sélectionné. Pour ce faire, nous devrons transmettre les identifiants du ménage choisie à l'application de saisie du questionnaire ménage. Nous le faisons à travers l'attribut "Key" du pff. L'application de saisie lira alors automatiquement cette valeur à partir du pff et pré-remplira les variables identifiantes une fois lancé. Pour définir l'attribut "Key" dans le fichier pff, appelez la fonction setproperty() de la variable pff et transmettez-lui le nom et la valeur de l'attribut. La valeur de l'attribut "Key" doit être la chaîne entière des identifiants du ménage sélectionnée dans le champ **CHOISIR_MENAGE**.

```
function lancerQuestionnaireMenage()
    pff pffMenage;
    pffMenage.load(pathname(application) + "../Menage/Popstan2020.pff");
    pffMenage.setproperty("Key", CHOISIR_MENAGE);
    pffMenage.exec();
end;
```

CSEntry préremplit les identifiants à l'aide de la chaîne transmise dans l'attribut "Key". Rendons également le champ protégé s'il est prérempli. Nous pouvons le faire en utilisant setproperty.

```
PROC PROVINCE
preproc

// protège le champ s'il est prérempli
if visualvalue(PROVINCE) <> notappl then

    // protège le champ afin que l'interviewer ne puisse pas le modifier
    setproperty(PROVINCE, "Protected", "Yes");

endif;
```

La logique de **DISTRICT**, **ZONE_DENOMBREMENT** et **NUMERO_MENAGE** est pareil.

Utilisation d'un fichier lookup pour le login

Plutôt que de laisser l'utilisateur choisir son rôle, attribuons un code de personnel à chaque interviewer et superviseur, puis créez un fichier de recherche contenant les codes du personnel, ainsi que le rôle et le district / zone de dénombrement attribué à l'interviewer / superviseur. Voici un exemple fichier de personnel :

Code	Nom	Rôle (1 = interviewer, 2 = superviseur)	Province	District	ZD
001	Shemika Rothenberger	2	1	1	
002	Andrew Benninger	1	1	1	1
003	Angelica Swenson	1	1	1	2
004	Zelma Hawke	1	1	1	3
005	Willis Catron	1	1	1	4

Notez que pour le superviseur, nous laissons la zone de dénombrement vide, car ils sont affectés à un district entier et non à une zone de dénombrement. Le superviseur supervisera toutes les zones de dénombrements de son district.

Utilisons Excel2CSpro pour créer un nouveau dictionnaire appelé Personnel.dcf et convertir la feuille de calcul en un fichier de données nommé personnel.dat. Modifions le champ de login de sorte que l'utilisateur entre son code du personnel au lieu de choisir son rôle. Nous devrons augmenter la taille de la variable et supprimer son ensemble de valeurs. Dans la postproc, nous devons maintenant rechercher le code du personnel dans le fichier de personnel, le valider, puis passer au menu approprié en fonction du rôle.

```
PROC LOGIN

// Vérifier le code du personnel à l'aide d'un fichier lookup
if loadcase(PERSONNEL_DICT, LOGIN) = 0 then
   errmsg("Code du personnel non valide. Réessayer.");
   reenter;
endif;

// Aller au menu approprié pour le rôle choisi
if PE_ROLE = 1 then
   skip MENU_PRINCIPAL_INTERVIEWER;
else
   skip to MENU_PRINCIPAL_SUPERVISEUR;
endif;
```

Lorsque nous déployons notre application, nous devons nous rappeler d’ajouter le fichier de données du personnel à notre spécification de déploiement, sinon nous ne pourrions pas accéder à l’application menu.

Préservation du login lors du retour au menu

Actuellement, lorsque vous lancez le programme de questionnaire ménage et revenez au menu, vous devez ressaisir le code de personnel. Sauvegardons le code de login afin que vous n'ayez à le saisir qu'une seule fois. Nous pouvons le faire en utilisant les commandes saveesetting() et loadsetting(). Ces commandes stockent et extraient des valeurs dans un stockage de paramètres persistant. Les valeurs enregistrées dans les paramètres sont disponibles même après la fermeture et le redémarrage de CSEntry. Elles sont disponibles dans toutes les applications CSPro sur un même appareil. Nous enregistrerons le code de login dans la postproc de login après l'avoir validé :

```
PROC LOGIN

// Vérifier le code du personnel à l'aide d'un fichier lookup
if loadcase(PERSONNEL_DICT, LOGIN) = 0 then
   errmsg("Code du personnel non valide. Réessayer.");
   reenter;
endif;

// Sauvegarder le login pour ne pas avoir à le saisir à nouveau
savesetting("login", maketext("%v", LOGIN));

// Aller au menu approprié pour le rôle choisi
if PE_ROLE = 1 then
   skip to MENU_PRINCIPAL_INTERVIEWER;
else
   skip to MENU_PRINCIPAL_SUPERVISEUR;
endif;
```

Les paramètres sont toujours stockés sous forme de chaîne de caractères alphanumériques donc nous devons utiliser maketext pour convertir le numérique à une chaîne.

Dans la preproc, nous tenterons de récupérer le code de login à partir des paramètres. S'il n'est pas vide, nous l'utiliserons au lieu de demander à l'utilisateur de saisir le code.

```
PROC LOGIN
preproc
// Vérifiez s'il y a un code de login existant
if loadsetting("login") <> "" then
   LOGIN = tonumber(loadsetting("login"));
   noinput;
endif;
```

Enfin, nous devons effacer le paramètre sauvegardé lorsqu'on sort de l'application.

```
PROC MENU_PRINCIPAL_INTERVIEWER
postproc
// Gère le choix de menu
if $ = 1 then
   lancerFicheNumerotation();
elseif $ = 2 then
  skip to CHOISIR_MENAGE;
elseif $ = 9 alors
   // Supprimer login stocké dans les paramètres
   savesetting("login", "");
   // Quitter
   stop(1);
endif;
```

```
// Afficher à nouveau le menu de l'intervieweur
reenter;
```

Paramètres du pff

La section de paramètres du fichier pff vous permet de transmettre n'importe quelle valeur à votre programme de saisie de données. Utilisons ceci pour transmettre le code d'intervieweur choisi dans le programme de menu au programme de questionnaire ménage.

```
// Lancer l'application de saisie de données de questionnaires ménage
function lancerQuestionnaireMenage()
   pff pffMenage;
   pffMenage.load(pathname(application) + "../Menage/Popstan2020.pff");
   pffMenage.setproperty("Key", CHOISIR_MENAGE);
   pffMenage.setproperty("CODE_INTERVIEWER", PE_CODE);
   pffMenage.exec ();
end;
```

L'étape suivante consiste à modifier l'application de saisie des données du ménage pour préremplir le code de l'intervieweur à l'aide du paramètre du fichier pff. Cela peut être fait en utilisant la commande sysparm() qui récupère un paramètre par son nom dans le pff. Nous pouvons le faire dans la preproc de CODE_INTERVIEWER dans l'application de questionnaire ménage. sysparm() renvoie toujours son résultat sous forme de chaîne de caractères, nous devons donc le convertir en nombre.

```
PROC CODE_INTERVIEWER
// Pré-remplir avec le paramètre du pff
preproc
if sysparm("CODE_INTERVIEWER") <> "" then
   $ = tonumber(sysparm("CODE_INTERVIEWER"));

   // Protéger le champ
   setproperty($, "Protected", "Yes");
endif;
```

Le rapport d'achèvement

Ajoutons le rapport au menu du superviseur. Nous allons créer un rapport indiquant le nombre total de ménages par le résultat d'interview. Voici un exemple :

```
Rapport d'achèvement
--------------------
Province: 01 District: 01

Résultat de l'interview
-----------------------
Complété: 10
Pas de membre à la maison: 1
Vacant: 2
Refusé: 1
Partiellement rempli: 5
Total: 19
```

Pour générer ce rapport, nous devons parcourir tous les cas dans le dictionnaire des ménages et compter le nombre de cas pour chaque résultat d'interview. Nous pouvons utiliser forcase pour parcourir les ménages. Pour calculer les totaux pour chaque catégorie, nous créons une variable pour chaque résultat et incrémentons la variable appropriée chaque fois que nous rencontrons un ménage avec le résultat correspondant.

```
// Affiche le rapport d'achèvement pour le district du superviseur.
fonction afficherRapportAchevement()

    string nomFichierRapport = pathname(application) + "rapport.txt";
    file ficheTemp;
    setfile(ficheTemp, nomFichierRapport, create);

    filewrite(ficheTemp, "Rapport d'achèvement");
    filewrite(ficheTemp, "--------------------");
    filewrite(ficheTemp, "");
    filewrite(ficheTemp, "Province %v District %v",
                PE_PROVINCE,
                PE_DISTRICT);
    filewrite(ficheTemp, "");
    filewrite(ficheTemp, "Résultat de l'interview");
    filewrite(ficheTemp, "-----------------------");

    numérique complete = 0;
    numérique pasMembreMaison = 0;
    numérique vacant= 0;
    numérique refuse = 0;
    numérique partiel= 0;

    forcase POPSTAN2020_DICT do
       if RESULTAT_INTERVIEW = 1 then
          complet = complet + 1;
       elseif RESULTAT_INTERVIEW = 2 then
          pasMembreMaison = pasMembreMaison + 1;
```

```
        elseif RESULTAT_INTERVIEW = 3 then
            vacant = vacant + 1;
        elseif RESULTAT_INTERVIEW = 4 then
            refuse = refuse + 1;
        autre
            partiel = partiel + 1;
        endif;
    endfor;

    filewrite(ficheTemp, "Complété: %d", complete);
    filewrite(ficheTemp, "Pas de membre à la maison: %d", pasMembreMaison);
    filewrite(ficheTemp, "Vacant: %d", vacant);
    filewrite(ficheTemp, "Refusé: %d", refuse);
    filewrite(ficheTemp, "Partiellement rempli: %d", partiel);
    filewrite(ficheTemp, "Total: %d", complete + pasMembreMaison +
                                vacant + refuse + partiel);

    close(ficheTemp);
    view(nomFichierRapport);
    endif;
end;
```

Exercices

1. Ecrire la fonction lancerFicherNumerotation() dans le programme de menu. Il convient de configurer la variable pff pour exécuter le programme de saisie de la fiche de numérotation. Définissez l'attribut "Key" dans le pff sur la province, le district et la zone de dénombrement. Définissez le code du personnel en tant que paramètre du fichier pff. Pré-remplissez le code du personnel dans le programme de numérotation. Assurez-vous que le programme de numérotation retourne au programme de menu à la sortie.
2. Modifiez la logique dans onfocus et postproc du champ **CHOISIR_MENAGE** pour ajouter une option supplémentaire à la fin de l'ensemble de valeurs intitulée "Retour" qui reviendra dans le menu principal de l'intervieweur.
3. Modifiez l'ensemble de valeurs dynamiques que nous créons dans la proc onfocus du champ **CHOISIR_MENAGE** pour afficher le résultat de l'interview (**A10** dans le questionnaire ménage) en plus du numéro de ménage et le nom du chef. Pour ce faire, vous devez utiliser loadcase pour rechercher le cas dans le fichier de données des ménages. Vous devrez également gérer le cas où le ménage n'existe pas encore dans le fichier de données des ménages (un ménage qui a été numéroté mais pas encore dénombré).
4. Créez un nouveau menu appelé "Rapports" contenant des options pour le rapport d'achèvement (celui que nous avons déjà écrit) et un nouveau "Rapport de la population totale" que vous allez écrire. Ce nouveau menu devrait également avoir la possibilité de revenir au menu principal. Le menu "Rapports" doit être accessible à partir du menu principal du superviseur. Le nouveau "Rapport de la population totale" devrait ressembler au modèle suivant :

```
Rapport de la population totale
-------------------------------
Province : 01 District: 02
Hommes : 1020
Femmes : 1025
Total : 2045
```

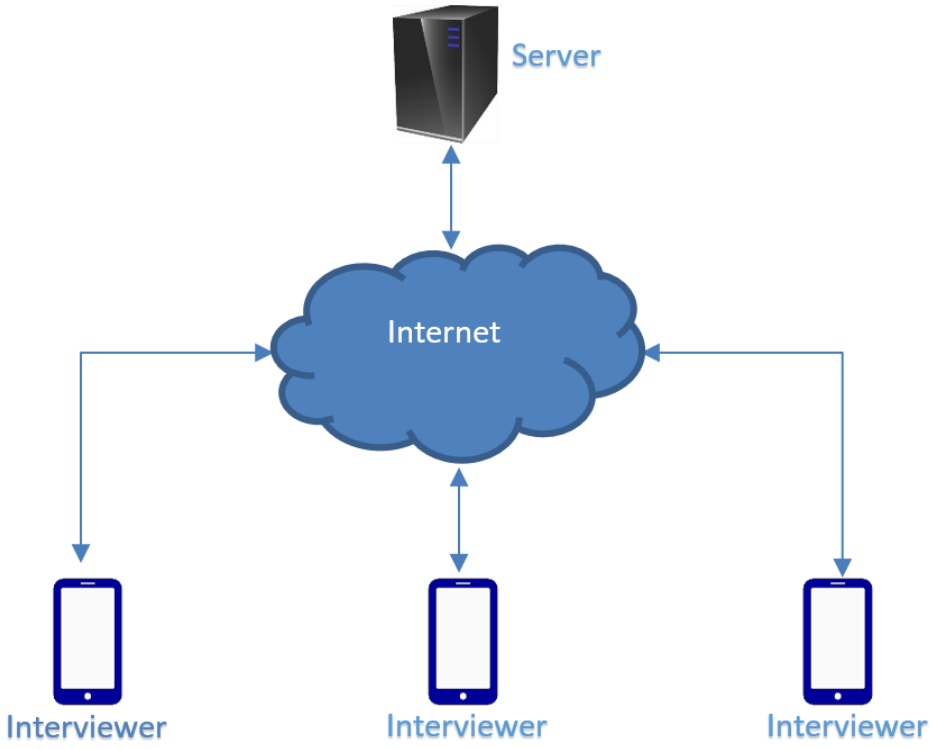
Session 09 : Synchronisation

À la fin de cette session, les participants seront en mesure de :

- Déployer des applications sur des appareils mobiles à travers l'internet
- Utiliser la boîte de dialogue des options de synchronisation pour ajouter la synchronisation à une application de saisie
- Créer des synchronisations avancées à l'aide de la logique CSPro
- Créer des synchronisations entre appareils pour la situation où il n'y a pas de connexion internet

Synchronisation dans CSPro

Après la collecte des données au terrain, vous devez récupérer les données de tous les appareils des intervieweurs pour créer un fichier de données combinées. Cela peut être fait manuellement en connectant chaque périphérique à un ordinateur, en copiant les fichiers de données un à un, et en les combinant à l'aide de l'outil CSPro "Concatenate Data", mais cela est plutôt gênant pour les enquêtes de grande envergure. Au lieu de cela, il est possible d'envoyer les données de chaque appareil à travers l'internet à un serveur central qui les combinera dans un seul fichier de données à utiliser pour le traitement ultérieur. Dans CSPro, ce processus s'appelle la synchronisation.



Direct synchronization between interviewers and central server over the internet

Le serveur de synchronisation

Pour utiliser la synchronisation par internet de CSPro, il vous faut un serveur connecter au réseau. CSPro est compatible avec trois types de serveurs de synchronisation :

- CSWeb : application web pouvant être installé dans un centre de données local ou hébergée sur un serveur du cloud. Idéal pour les grandes enquêtes. Nécessite des compétences dans la configuration et l'administration des site web et d'une base de données SQL.
- Dropbox : un service gratuit de synchronisation avec les serveurs du cloud utilisant un compte créé sur www.dropbox.com. Les données sont stockées sur les serveurs gérés par Dropbox aux États-Unis. Idéal pour les petites enquêtes lorsque les compétences et l'infrastructure nécessaires à la configuration de CSWeb ne sont pas disponibles.
- FTP : CSPro peut se synchroniser avec n'importe quel serveur FTP (File Transfer Protocol) accessible via le réseau. Idéal pour les petites enquêtes lorsque les compétences et l'infrastructure nécessaires à la configuration de CSWeb ne sont pas disponibles et que vous ne souhaitez pas que vos données soient stockées sur les serveurs de Dropbox.

Pour nos exemples de cet atelier, nous utiliserons un serveur CSWeb, mais utiliser Dropbox ou FTP est très similaire.

L'installation du serveur CSWeb n'entre pas dans le cadre de cette formation. Nous allons donc travailler avec un serveur déjà installé. Le processus de configuration d'un serveur CSWeb est bien décrit dans le guide de l'utilisateur CSWeb, qui est inclus dans la documentation de CSPro disponible dans le menu “Help" de CSPro.

Déploiement des applications

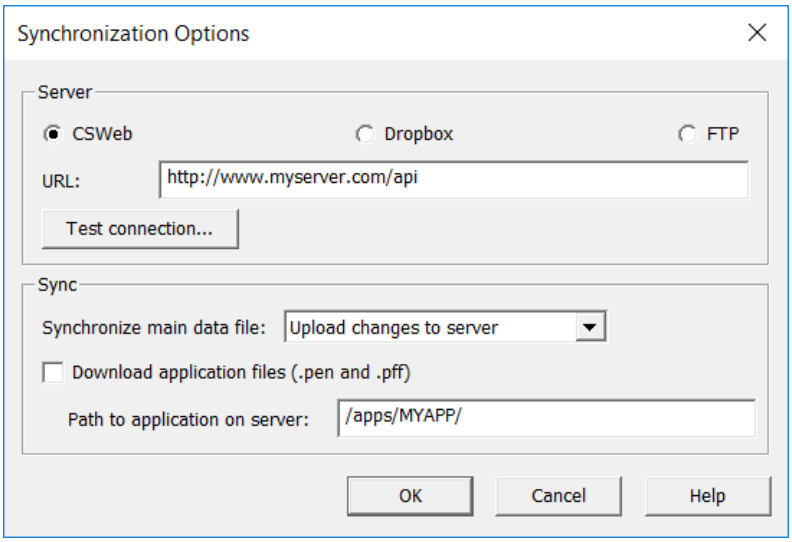
Au lieu de copier notre application sur un périphérique mobile à l'aide de la connexion USB, nous pouvons la télécharger depuis un serveur. Avant de faire cela, nous devons utiliser l'outil “Deploy Applications” pour envoyer l'application sur le serveur. Nous avons vu dans la session 8 comment utiliser l'outil “Deploy Applications” pour effectuer un déploiement dans un dossier local. Pour déployer sur un serveur, modifiez simplement l'option "Deploy To" par le type de serveur sur lequel télécharger. Vous pouvez déployer sur un serveur CSWeb, Dropbox ou FTP. L'outil de déploiement enverra tous les fichiers que vous avez déposés sur la liste de fichiers sur le serveur. Il téléchargera également les dictionnaires de votre application sur le serveur. Le téléchargement d'un dictionnaire sur le serveur est requis avant de pouvoir synchroniser le fichier de données associé à ce dictionnaire.

Une fois que vous avez téléchargé l'application sur le serveur, vous pouvez l'installer sur un appareil mobile en choisissant "Ajouter Application" dans le menu de l'écran de liste des applications. Cela vous demandera les détails du serveur, et puis il affiche une liste des applications disponibles sur ce serveur. Une fois que vous avez choisi une application, celle-ci est copiée dans le répertoire CSEntry de votre appareil et est prêt à lancer pour la saisie.

Vous pouvez également déployer des mises à jour de votre application sur le serveur et utiliser la même procédure pour télécharger l'application mise à jour sur votre appareil.

Ajout de la synchronisation à l'application de saisie de données

Pour pouvoir utiliser la synchronisation, vous devez d'abord ajouter les informations du serveur à l'application pour activer la synchronisation. Dans CSPro, choisissez "Synchronization…" dans le menu “Options”. Choisissez le type de serveur (CSWeb, Dropbox ou FTP). Pour CSWeb ou FTP, inscrivez l'URL du serveur (pour Dropbox, aucune URL n'est requise). Vous pouvez utiliser le bouton "Test connection…" pour vérifier que l'URL est correcte. Pour CSWeb, l'URL à utiliser pour la synchronisation dans CSPro se terminera généralement par "/api", tandis que l'URL à utiliser pour accéder à l'interface web se terminera généralement par "/ui". Le premier est pour CSPro lui-même de se connecter au serveur tandis que le second est pour l’utilisation uniquement dans le navigateur web.



Dans le menu déroulant, vous avez trois choix pour synchroniser le fichier de données principal de l’application :

- Upload changes to server : tous les nouveaux cas et tous les cas modifiés localement seront envoyer sur le serveur.
- Download changes from server : tous les nouveaux cas et les cas modifiés sur le serveur seront récupérées à partir du serveur.
- Sync local and remote changes : combine les deux premières options en envoyant des mises à jour au serveur et en récupérant des mises à jour à partir du serveur.

Dans la plupart des cas, vous voudrez choisir la première ou la troisième option. Utilisez la troisième option si vous souhaitez partager des cas entre plusieurs périphériques sur le terrain, car cela téléchargera chaque cas sur le serveur sur le périphérique local. Pour une opération d'enquête avec un grand nombre d'intervieweurs, évitez cela car cela vous obligerait à télécharger beaucoup de données. Pour notre test, nous allons utiliser la première option.

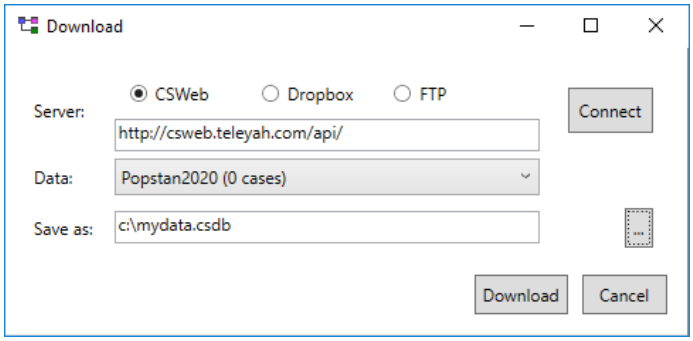
En plus de la synchronisation du fichier de données, vous pouvez également récupérer les fichiers pen et pff à partir du serveur. Cela fournit un moyen d'effectuer des mises à jour à distance de l'application pendant que les enquêteurs sont sur le terrain. Pour que cela fonctionne, vous devez placer une copie des fichiers pen et pff sur le serveur. Pour CSWeb, ceux-ci doivent être placés dans le dossier *csweb/files/* sur le serveur. Pour Dropbox, ils peuvent être placés n'importe où dans le dossier Dropbox. Pour FTP, ils doivent être placés dans le répertoire de base de l'utilisateur qui sera utilisé pour se connecter au serveur. Inscrivez le chemin sur le serveur où les fichiers ont été placés sous "Path to application on server". Notez qu'il s'agit d'une alternative à l'utilisation de l'outil “Deploy Applications” et “Ajouter Application” pour mettre à jour l'application. L'avantage de cette approche est qu'elle combine la synchronisation des données avec la mise à jour de l'application en une seule opération. L'inconvénient est que vous devez copier manuellement les fichiers .pen et .pff sur le serveur au lieu d'utiliser l'outil “Deploy Applications”.

Une fois que la configuration des paramètres de synchronisation est faite, publiez encore le fichier pen. Les paramètres de synchronisation sont stockés dans le fichier pen; ainsi, s'il n'est pas mis à jour, vous ne pourriez pas effectuer de synchronisation.

Exécution de la synchronisation sur le périphérique

Pour synchroniser l'application avec le serveur, lancez-la dans CSPro. Dans l'écran de liste des cas, choisissez "Synchronize…" dans le menu File sur le PC ou appuyez sur l'icône de synchronisation () sur le mobile.

Téléchargement du fichier de données combinées

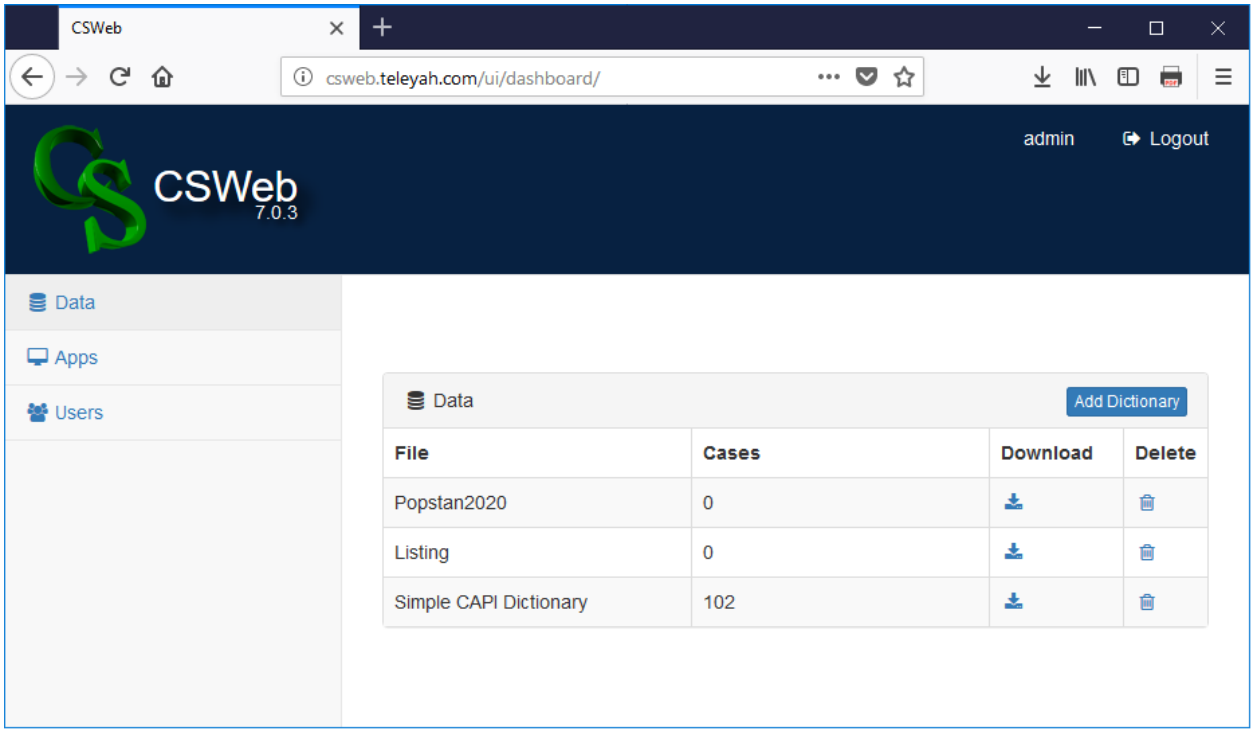


Une fois que vous avez utilisé la synchronisation pour envoyer les données des mobiles sur le serveur, vous pouvez télécharger le fichier de données à partir du serveur à l'aide de l'outil DataViewer. Cela téléchargera un fichier de données unique qui combine toutes les données envoyées à partir de tous les appareils synchronisés avec le serveur. Ouvrez

DataViewer, selectionnez “Download”du menu “File”, choisissez le type de serveur, inscrivez la même URL que celle utilisée dans la boîte de dialogue des options de synchronisation, cliquez sur "Connect" pour télécharger la liste des fichiers de données disponibles sur le serveur, choisissez le fichier de données et le nom d’un fichier pour enregistrer les données téléchargées et cliquez sur "Download" pour effectuer le transfert.

Utilisation de l'interface du serveur Web CSPro

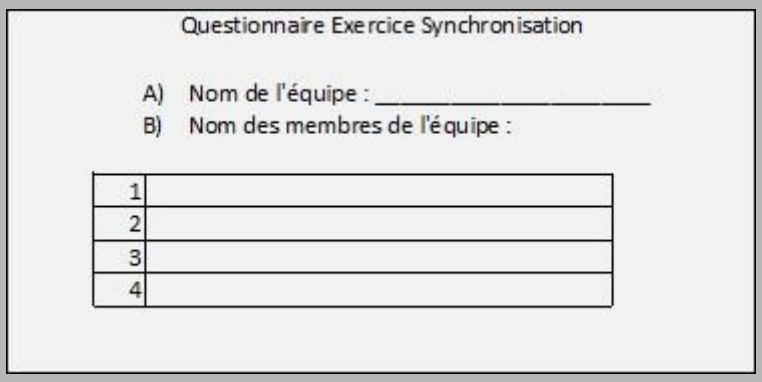
Le serveur CSWeb dispose d'une interface web à laquelle vous pouvez accéder via un navigateur. L'URL de l'interface web est similaire à celle utilisée dans CSPro, sauf que "api" à la fin est remplacé par "ui". L'interface web vous permet de gérer les applications déployées, les dictionnaires de données et les comptes d'utilisateurs.



Si vous utilisez CSWeb, vous pouvez éviter de remplir la boîte de dialogue de téléchargement en cliquant sur l'icône "Download" dans l'interface CSWeb du navigateur. Cela téléchargera un fichier pff. Lorsque vous ouvrez le fichier pff téléchargé, il lancera automatiquement l’outil DataViewer sur le PC et commencera à télécharger les données combinées. Une fois que vous avez téléchargé le fichier de données unepremière fois, vous pouvez utiliser la fonction "Synchronize" de DataViewer pour ne télécharger que les données nouvellement mises à jour, au lieu de télécharger le fichier entier une deuxième fois. Si vous utilisez Dropbox ou FTP, CSPro créera automatiquement un fichier pff nommé *DownloadData-<nom de votre dictionnaire>.pff* dans votre Dropbox ou sur votre serveur FTP qui lancera DataViewer pour télécharger les données.

Travail de groupe - Course de synchronisation

Séparez-vous en groupes de 3 à 4 personnes et créez une petite application CSPro pour le questionnaire suivant :


Questionnaire Exercice Synchronisation

A) Nom de l'équipe : _____

B) Nom des membres de l'équipe :

1	
2	
3	
4	

Chaque équipe doit utiliser le nom de son équipe comme nom du dictionnaire de leur application. Si deux équipes utilisent le même nom de dictionnaire, elles se synchroniseront toutes les deux sur le même fichier de données sur le serveur et nous voudrions éviter cela.

Déployez votre application sur le serveur suivant :

URL : http://csweb.teleyah.com/api
Nom d'utilisateur de l'administrateur : test
Mot de passe de l'administrateur : password

Ajoutez une synchronisation à votre application en utilisant les détails du serveur ci-dessus.

Choisissez de télécharger des données sur le serveur.
Ne synchronisez PAS les fichiers de l'application (pen et pff).

Lancez votre application, saisissez un cas et synchronisez pour envoyer vos données au serveur.

La première équipe à télécharger ses données sur le serveur est le gagnant.

Synchronisation à partir de la logique

Parfois, nous avons besoin de plus de contrôle sur la synchronisation que ne le prévoit la boîte de dialogue des options de synchronisation. Par exemple, si une application contient des dictionnaires externes, si vous souhaitez synchroniser avec un univers ou s'il existe plusieurs applications à synchroniser à la fois. CSPro fournit des fonctions logiques pour exécuter des synchronisations à partir de la logique de l'application permettant des synchronisations plus complexes.

Ajoutons la synchronisation à notre application de menu qui téléchargera le fichier de données principal et téléchargera la mise à jour des fichiers pen et pff pour l'application de menu et pour le questionnaire ménage et la fiche de numérotation. Nous allons ajouter une nouvelle option dans le menu principal du superviseur intitulée "Synchroniser avec le siège".

Pour faire la synchronisation à partir de la logique, nous utilisons d'abord la commande syncconnect() et à lui transmettre les informations du serveur. Si cette fonction réussit, nous sommes connectés au serveur et nous pouvons faire des appels aux commandes syncdata() et syncfile() pour synchroniser les fichiers de données et les autres fichiers. Enfin, nous appelons syncdisconnect() pour terminer la session.

Les commandes syncdata() et syncfile() fonctionnent différemment. syncdata() ne peut être utilisé qu'avec des fichiers de données au format CSPro DB. Avec syncdata(), CSPro garde l'historique des cas dans le fichier qui ont déjà été téléchargés sur le serveur afin d'éviter le transfert du même cas deux fois. Cela réduit la quantité de données transférées et réduit au minimum les coûts et les temps de transfert des données. Cela évite également à un intervieweur d'écraser les modifications apportées par un autre intervieweur qui travaille sur des cas différents dans le même fichier. syncfile() peut être utilisé avec tout type de fichier, y compris les fichiers texte, les images et les fichiers d'application (pen et pff). Il ne regarde pas le contenu du fichier, il envoie ou recupere tout le fichier, écrasant toute version existante. Lors de la synchronisation de fichiers de données, vous devez toujours utiliser syncdata().

La commande syncdata() prend deux arguments: la direction et le nom du dictionnaire. La direction peut être PUT (envoyer les données sur le serveur), GET (récupérer les données du serveur) ou BOTH (envoyer et récupérer). Ces options correspondent au menu déroulant de la boîte de dialogue des options de synchronisation. Le deuxième argument est le nom d'un dictionnaire de l'application qui correspond au fichier de données à synchroniser. Notez que ce dictionnaire doit être ajouté à l'application en tant que dictionnaire externe.

Dans notre application, nous souhaitons synchroniser le dictionnaire du questionnaire ménage, nous devons donc d'abord l'ajouter au programme de menu en tant que dictionnaire externe. Nous pouvons maintenant l'utiliser dans l'appel de syncdata().

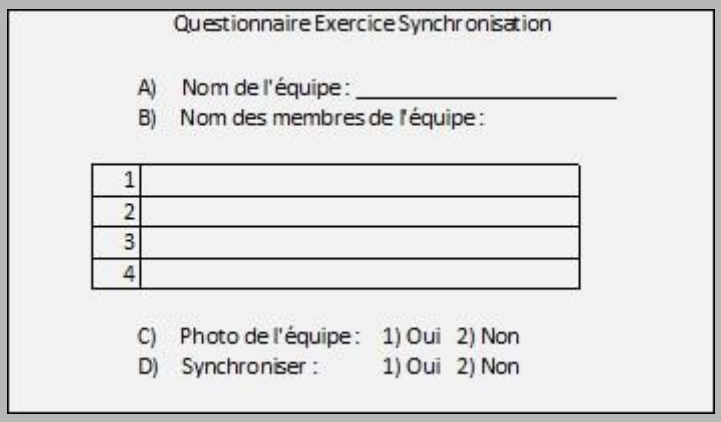
Nous ajouterons également des appels à syncfile() pour télécharger les dernières versions des programmes d'application à partir du serveur. Cette fonction prend la direction (PUT ou GET), le chemin du répertoire source et le chemin du répertoire destination. Dans le cas de GET, le répertoire source est le chemin sur le serveur et le répertoire destination est le répertoire local du périphérique.

```
// Télécharger les cas sur le serveur web du siège et télécharger la dernière
// version des fichiers de l'application
function syncAvecSiege()

    // Se connecter au serveur
    if syncconnect(CSWeb, "http://csweb.teleyah.com/api") then

        // Synchroniser les cas du fichier de données du ménage avec
        // le serveur
        syncdata(BOTH, POPSTAN2020_DICT);

        // Télécharger la dernière version de l'application de menu
        syncfile(GET, "/Popstan2020/Menu/Menu.pen", "./Menu.pen");
        syncfile(GET, "/Popstan2020/Menu/Menu.pff", "./Menu.pff");

        // Obtenir la dernière version du programme de saisie de questionnaire ménage
        syncfile(GET, "/Popstan2020/Menage/Popstan2020.pen",
                "../Menage/Popstan2020.pen");

        // Se déconnecter du serveur
        syncdisconnect();

    endif;
end;
```

Maintenant que nous avons ajouté la logique de synchronisation au menu, nous publions encore les fichiers pen et les copions dans le répertoire “files” sur le serveur. Cela nous permettra d’exécuter le programme de menu et de tester notre synchronisation.

Travail de groupe - Course de synchronisation II

Dans vos équipes, modifiez le questionnaire de l'exercice de synchronisation pour prendre une photo de l'équipe et l'enregistrer dans un fichier.



Ajoutez deux questions, C et D, à votre application. Pour la question C, prenez une photo si l'intervieweur répond "oui". Pour la question D, ajoutez une logique à la postproc de la question afin qu'elle se synchronise avec le serveur si l'utilisateur sélectionne "oui". Utilisez syncfile() avec PUT pour placer la photo dans le répertoire */photos/* sur le serveur.

Lancez votre application, saisir un cas et synchronisez pour envoyer votre photo au serveur.

La première équipe à télécharger sa photo sur le serveur est le gagnant.

La sécurité des données

Il est important d'assurer la sécurité de votre serveur et de vos données. La sécurité des ordinateurs et des réseaux est un problème complexe qui dépasse largement le cadre de cette formation. Voici quelques considérations de sécurité à prendre en compte. Pour toute grande enquête ou opération de recensement, vous devriez consulter un expert en cybersécurité pour vous assurer que vos données sont bien sécurisées.

Utiliser HTTPS sur le serveur web pour le transfert de données
Cela crypte la transmission de données par internet. Cela nécessite l'installation d'un certificat SSL sur le serveur. Sans SSL, les mots de passe des serveurs sont envoyés non chiffrés et sont vulnérables aux pirates.

Utiliser un réseau privé
Pour les recensements et les enquêtes de grande envergure, vous pouvez éventuellement travailler avec le fournisseur de télécommunication pour créer un réseau privé auquel seuls vos appareils peuvent accéder. Cela évite de placer votre serveur sur l'internet public où il pourrait être vulnérable aux attaques. Cela permet également de limiter vos appareils pour pouvoir uniquement accéder à votre serveur et non aux autres sites web que vous ne souhaitez pas que vos intervieweurs utilisent.

Utiliser le cryptage de périphérique sur les périphériques Android
En définissant un code PIN sur votre tablette ou votre téléphone Android, vous activez le cryptage des fichiers sur le périphérique à l'aide d'un cryptage matériel renforcé. Cela rend très difficile pour quiconque sans le code PIN de récupérer des données sur le périphérique.

Comment gérer les mots de passe ?
Il est plus pratique d'avoir un seul mot de passe partagé par tous les appareils et codé en dur dans le fichier pen, mais il est plus sûr d'utiliser un nom d'utilisateur / mot de passe différent pour chaque intervieweur et de le saisir à chaque synchronisation. Pour un grand nombre d'intervieweurs, la gestion des mots de passe et la réinitialisation des mots de passe pour ceux qui ont oublié leurs mots de passe pourraient constituer un fardeau de gestion important. Vous devrez trouver le juste équilibre entre la sécurité et la commodité. Si vous souhaitez coder en dur le mot de passe dans votre fichier de pen, vous pouvez passer le nom d'utilisateur et le mot de passe dans l'appel à syncconnect(). Si vous ne fournissez pas de nom d'utilisateur ni de mot de passe lors de cet appel, l'intervieweur est invité à saisir le mot de passe lors de sa première synchronisation avec le périphérique. Après cela, le nom d'utilisateur et le mot de passe sont enregistrés sur l'appareil. Il n'est donc pas nécessaire de les saisir encore pour les synchronisations ultérieures. Si vous ne souhaitez pas que le mot de passe soit enregistré sur le périphérique, vous pouvez utiliser la fonction prompt() pour que l'intervieweur le saisisse avant chaque synchronisation, et puis transmettre le mot de passe reçu de prompt() à syncconnect().

Synchronisation avec un univers

La commande syncdata() possède un troisième paramètre facultatif, l'univers. Il s'agit d'une chaîne de caractères que CSPro tente de faire correspondre aux éléments de l'identifiant de cas pour limiter les données transférées. Si l'univers est fourni, seuls les cas dont l'identifiant de cas correspond à l'univers sont synchronisés. Par exemple, pour limiter la synchronisation à la province 1, district 2, nous appellerions :

```
syncdata(BOTH, HOUSEHOLDQUESTIONNAIRE_DICT, "102");
```

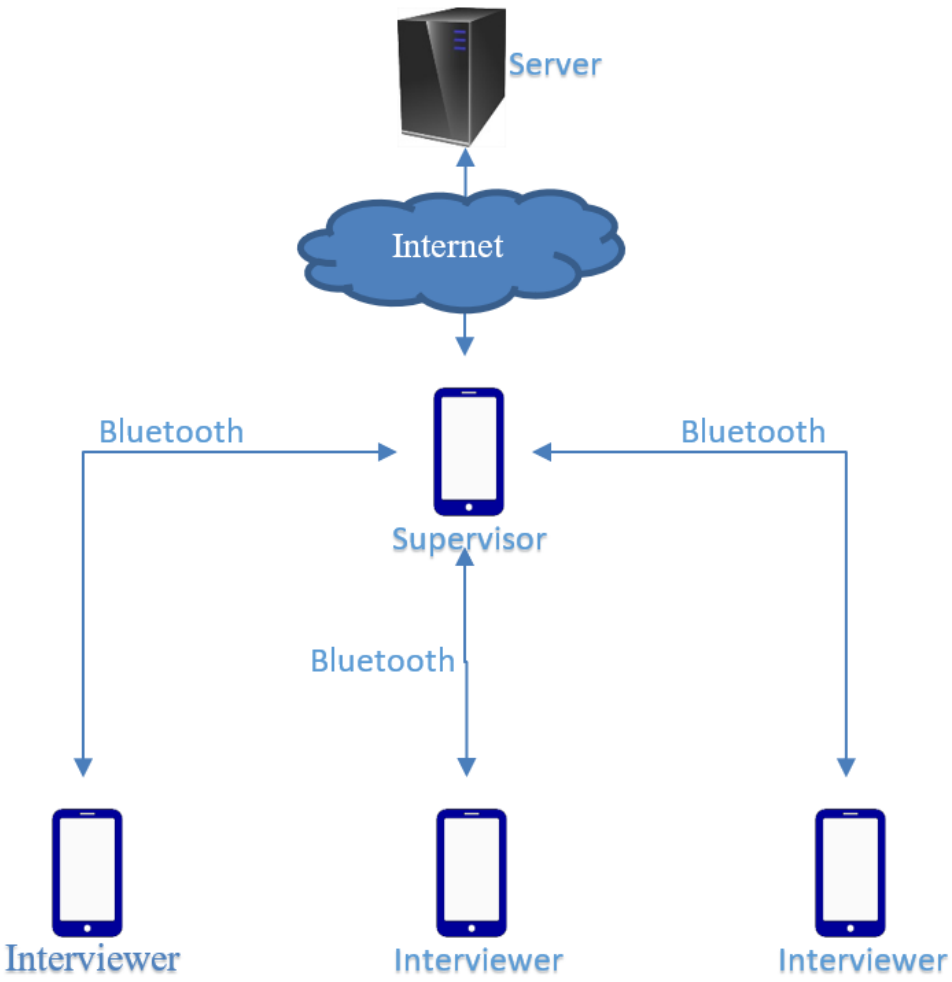
Cela ne synchronisera que les cas dont les identifiants commencent par "102". Comme la province et d'un chiffre et le district est de 2, la synchronisation est donc limitée aux cas de la province 1 et du district 2. De même, si nous utilisions "1" comme univers, tous les cas de la province 1 seraient synchronisés.

Ceci est utile lorsqu'il y a un grand nombre d'intervieweurs et que vous ne voulez pas que chaque intervieweur télécharge les données des autres. Vous pouvez spécifier la zone géographique attribuée à l'intervieweur en tant qu'univers, de sorte que seuls les cas de cette zone soient synchronisés.

Synchronisation pair à pair à l'aide de Bluetooth

Dans certains endroits, les intervieweurs n'ont pas un accès à l'internet fiable et ne peuvent pas se synchroniser directement avec le serveur. Dans ce cas, vous pouvez utiliser une

synchronisation pair à pair. Les intervieweurs se synchronisent avec la tablette ou l'ordinateur portable d'un superviseur par Bluetooth. Plus tard, le superviseur se rend à un endroit où il peut se connecter à l'internet et il se synchronise avec le serveur central.



Synchronization with supervisor over Bluetooth when interviewers do not have internet

Pour mettre en œuvre la synchronisation Bluetooth, nous devons ajouter une logique à la fois au dispositif de superviseur et au celui de l'interviewer. Un des deux périphériques agit comme un serveur, traitant les demandes de synchronisation émanant du périphérique client, et l'autre, le client, envoie les commandes syncfile() et syncdata() de la même manière que s'il le faisait à un serveur CSWeb, Dropbox ou FTP. Pour notre exemple, faisons-en sorte que le périphérique superviseur soit le serveur et que le périphérique intervieweur soit le client. C'est arbitraire, on pourrait facilement inverser les rôles.

Pour étendre le programme de menu afin de faire la synchronisation entre superviseur et interviewer, nous allons ajouter une nouvelle option de menu au menu principal du superviseur "sync avec l'interviewer" et une nouvelle option de menu au menu principal de l'intervieweur "sync avec le superviseur".

La logique de la tablette de l'intervieweur est presque identique à celle utilisée pour la synchronisation web. La seule différence c'est les arguments de syncconnect(). Lorsque

vous utilisez Bluetooth, syncconnect() n'a pas besoin d'URL, de nom d'utilisateur ou de mot de passe.

```
syncconnect(Bluetooth)
```

La fonction syncconnect() cherchera tous les périphériques Bluetooth à proximité et présentera une liste à l'intervieweur qui pourra choisir le périphérique auquel se connecter. Si vous connaissez à l'avance le nom du périphérique auquel vous souhaitez vous connecter, vous pouvez également le spécifier en tant que deuxième paramètre de syncconnect(). Dans ce cas, syncconnect() essaiera de se connecter directement au périphérique nommé.

```
syncconnect(Bluetooth, "Supervisor03")
```

Le reste de la logique est identique à celui de la synchronisation web, à l'exception du fait que les chemins sources changent pour correspondre aux chemins du périphérique superviseur.

```
function syncAvecSuperviseur()
   // Se connecter au périphérique du superviseur
   // Nous n'utilisons pas le nom du périphérique auquel se connecter,
   // ce qui permet à l'intervieweur/ de sélectionner le périphérique dans une liste de
   // périphériques à proximité.
   if syncconnect(bluetooth) then

      // Sync le fichier de données principal.
      syncdata(BOTH, POPSTAN2020_DICT);

      // Téléchargez les derniers fichiers d'application du superviseur.
      syncfile(GET, "./Menu.pen", "./Menu.pen");
      syncfile(GET, "./Menu.pff", "./Menu.pff");

      // Comme l'application actuelle se trouve dans le dossier Menu, nous devons
      // utiliser "../Menage" pour monter d'un niveau et descendre dans le dossier
      // Menage pour les fichiers de l'application de ménage.
      syncfile(GET, "../Menage/Popstan2020.pen",
            "../Menage/Popstan2020.pen");

      syncdisconnect();
   endif;
end;
```

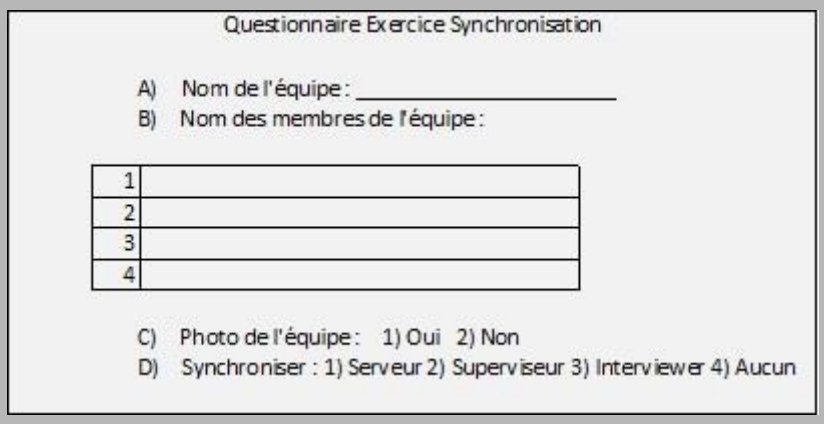
La logique pour le superviseur est encore plus simple. Nous appelons la commande syncserver() qui exécute le serveur Bluetooth et attend les connexions des périphériques clientes.

```
function syncAvecInterviewer ()
   // Lancer le serveur Bluetooth pour recevoir les données de l'intervieweur.
   syncserver(Bluetooth);
end;
```

Par défaut, le répertoire racine du serveur Bluetooth est le répertoire de l'application menu.

Travail de groupe - Course de synchronisation III

Dans vos équipes, modifiez l'application de synchronisation pour ajouter la synchronisation Bluetooth.



Modifiez la question D pour pouvoir synchroniser via Bluetooth.

Lancez votre application sur deux appareils, complétez le cas sur un appareil et synchronisez-la pour envoyer votre photo à l'autre appareil via Bluetooth. Utilisez le deuxième appareil à synchroniser pour envoyer votre photo au serveur web.

La première équipe à télécharger sa photo sur le serveur est le gagnant.

Exercices

1. Modifiez les fonctions syncAvecSiege() et syncAvecSuperviseur() dans le programme de menu pour synchroniser également le fichier de données de la fiche de numérotation des ménages et le fichier lookup de personnel. Le fichier de numérotation doit être synchronisé à l'aide de BOTH et le fichier du personnel doit être téléchargé à partir du serveur à l'aide de syncdata avec GET.
2. Modifiez les fonctions syncAvecSiege() et syncAvecSuperviseur() afin que seules les données de la zone géographique attribuée à l'intervieweur / superviseur soient synchronisées. Pour l'intervieweur, cela signifie que seuls les ménages de la zone de dénombrement attribuée sont synchronisés et que pour le superviseur, seuls les ménages du district affecté sont synchronisés.

Session 10 : Apurement et exportation

À la fin de cette session, les participants seront en mesure de :

- Faire l’apurement et l’imputation dans une application “batch edit”
- Utilisez une application “batch edit” pour ajouter des variables calculées et des variables recodées
- Utiliser une application “batch edit” pour convertir les valeurs des cases à cocher en variables oui / non
- Utiliser l'outil d'exportation pour transférer des données de CSPro vers d'autres logiciels.

Création d'une application “batch”

L’application “batch edit” (traitements par lots) est comme une application de saisie de données mais sans les formulaires. Il est destiné à être exécuté après la saisie des données pour détecter et résoudre les problèmes dans le fichier de données. Une application traitement par lots prend un fichier de données d'entrée et exécute une logique dessus. Il génère un rapport, appelé fichier de liste, et éventuellement un fichier de données de sortie, qui est une version modifiée du fichier d'entrée. Une application traitement par lots ne modifie jamais le fichier d'entrée.

Pour créer une application traitement par lots, choisissez File -> New à partir de CSPro, puis choisissez “Batch Edit Application”. Ensuite sélectionnez le dictionnaire. C'est généralement le même dictionnaire que vous avez utilisé pour la saisie de données.

L'interface utilisateur dans l’application “batch” est similaire à celle des applications de saisie de données, à l'exception du fait qu'il n'y a pas de formulaire. L’onglet avec l'arborescence des éléments des formulaires est remplacé par un onglet avec un arborescence de “Edits” qui affiche les variables et les enregistrements du dictionnaire. Tout comme dans la saisie de données, vous ajoutez la logique aux procs. Cliquant sur une variable ou sur un roster dans l'arborescence affiche sa proc dans la fenêtre de logique. Lorsqu’on lance l’application “batch”, toutes les procs sont exécutées mais au lieu de s'exécuter de manière interactive, tous les messages d'erreur sont écrits dans un fichier journal pour être revus une fois que le programme entier a terminé.

Vérification des données

Pour ajouter des contrôles de cohérence, nous procédons comme dans notre application de saisie de données en ajoutant une logique à la proc appropriée. Commençons par un simple contrôle pour nous assurer que l'âge au premier mariage n'est pas supérieur à l'âge.

```
PROC AGE_PREMIER_MARRIAGE

// Vérifie si l'âge au premier mariage est inférieur à l'âge actuel
if AGE_PREMIER_MARRIAGE> AGE then
   errmsg("Âge au premier mariage supérieur à l'âge");
endif;
```

Nous lançons ensuite l'application, mais nous avons d'abord besoin d'un fichier de données. Vous pouvez utiliser le fichier Popstan2020Brutes.csdb. Lancez l'application sur ces données pour la tester. Après l'exécution de l'application, nous voyons le fichier "listing" (fichier journal). Le fichier journal signale que nous avons un cas dans le fichier de données dans lequel cette erreur existe.

```
Process Messages

*** Case [3691141112] has 1 messages (0 E / 0 W / 1U)
    U   -20  Âge au premier mariage supérieur à l'âge

User unnumbered messages:

   Line    Freq  Pct.  Message text                                          Denom
   ----    ----  ----  ------------                                         -----
     20       1     -  Age at first marriage greater than age                   -

CSPRO Executor Normal End
```

Pour déterminer où se trouve le problème, nous pouvons ouvrir le cas pour lequel l'erreur a été signalé. Le message dans le fichier journal contient les identifiants de cas que nous pouvons utiliser pour l'afficher. Vous pouvez copier l'identifiant à partir du fichier journal et l'utiliser avec l'option "Find Case…" du menu "Edit" de CSEntry.

Informations sommaires

Si vous ajoutez le mot-clé summary après la commande errmsg, les cas d'erreur individuels ne sont pas écrits dans le fichier journal. Avec summary, seul le nombre de fois total que chaque erreur est rencontrée est écrit dans le fichier. Cela peut être utile lorsque vous travaillez avec des fichiers de données volumineux comportant de nombreuses erreurs, ce qui rend le fichier journal très lourd.

```
PROC AGE_PREMIER_MARRIAGE

// Vérifie si l'âge au premier mariage est inférieur à l'âge actuel
if AGE_PREMIER_MARRIAGE> AGE then
   errmsg("Âge au premier mariage supérieur à l'âge") summary;
endif;
```

Désormais, lorsque nous exécutons l'application, le fichier journal ne contient plus les cas individuels dans lesquels les erreurs se sont produites.

```
User unnumbered messages:

   Line     Freq  Pct.  Message text                                          Denom
   ----     ----  ----  ------------                                         -----
     20        1     -  Âge au premier mariage supérieur à l'âge                    -

CSPRO Executor Normal End
```

Si vous ajoutez le mot-clé denom, CSPro calcule le pourcentage de cas où l'erreur est survenue en divisant l'effectif de l'erreur par le dénominateur qui vous l'avez fourni. Le dénominateur à utiliser dépend de ce qui est compté. Dans notre cas, il s'agit du nombre total de membres du ménage âgés de 10 ans ou plus.

```
PROC GLOBAL
numeric nombreMembres10AnsPlus =  0;

PROC AGE_PREMIER_MARRIAGE

if AGE >= 10 then
   nombreMembres10AnsPlus = nombreMembres10AnsPlus + 1;

   // Vérifier l'âge au premier mariage inférieur à l'âge actuel
   if AGE_PREMIER_MARRIAGE > AGE then
      errmsg("Âge au premier mariage supérieur à l'âge") summary
            denom = nombreMembres10AnsPlus;
   endif;
endif;
```

Lorsque nous lançons le programme, nous constatons que 0,2% des personnes de 10 ans et plus qui ont un âge au premier mariage supérieur à leur âge.

```
User unnumbered messages:

   Line     Freq  Pct.  Message text                                          Denom
   ----     ----  ----  ------------                                         -----
     20        1   0.2  Age at first marriage greater than age                 463

CSPRO Executor Normal End
```

Correction des erreurs

En plus de l'utilisation de l'application de traitement par lots pour trouver les erreurs, vous pouvez également l'utiliser pour corriger les problèmes. C'est-à-dire pour l'apurement des données. Cela se fait en modifiant les valeurs variables dans la logique de votre program. Limitons simplement la valeur de l'âge au premier mariage pour qu'il ne soit jamais supérieur à l'âge.

```
PROC AGE_PREMIER_MARRIAGE

// Vérifie si l'âge au premier mariage est inférieur à l'âge actuel
if AGE_PREMIER_MARRIAGE> AGE then
   errmsg("Âge au premier mariage supérieur à l'âge. Limitation de l'âge au premier
mariage à l'âge.");
   AGE_PREMIER_MARRIAGE = AGE;
endif;
```

Lorsque nous effectuons cette opération, nous spécifions un fichier de sortie : Popstan2020Apure.csdb. Les modifications apportées ne seront apportées qu'au fichier de sortie. Nous pouvons ensuite relancer l'application de traitement par lots sur le fichier de sortie pour nous assurer que nous n'avons plus de message d'erreur.

Au lieu de simplement attribuer la valeur de AGE_PREMIER_MARRIAGE nous pouvons utiliser la commande impute qui fait l'affectation comme l'opérateur « = » mais génère également un rapport montrant bien les valeurs qui ont été imputées.

```
// Vérifie si l'âge au premier mariage est inférieur à l'âge actuel
if AGE_PREMIER_MARRIAGE> AGE then
   errmsg("Âge au premier mariage supérieur à l'âge. Limitation de l'âge au premier
mariage à l'âge.");
   impute(AGE_PREMIER_MARRIAGE, AGE);
endif;
```

Le rapport d'imputation sera ouvert dans TextViewer après l'exécution de l'application, mais pour le voir, vous devrez aller au menu “Window” et choisir le fichier qui se termine par ".frq.lst".

```
                          IMPUTE FREQUENCIES                          Page    1

_____________________________________________________________________________
       Imputed Item AGE_PREMIER_MARRIAGE: Age au premier marriage - all occurrences

                                  _____________________________ _____________
 Categories                             Frequency       CumFreq       %  Cum %
_____________________________ _____________________________ _____________
 40                                             1             1   100.0  100.0
_____________________________ _____________________________ _____________
 TOTAL                                          1             1   100.0  100.0
```

Variables calculées

Il est souvent utile d'ajouter d’autres variables dans le fichier de données après la collecte qui sont calculées à partir des variables collectées. Par exemple, ajoutons une variable oui / non / ne sais pas à l’enregistrement de l’habitat qui détermine si le chef du ménage est un enfant. Premièrement, nous ajoutons la nouvelle variable **CHEF_ENFANT** au dictionnaire (à la fin de l'enregistrement de l’habitat afin de ne pas gâcher nos données existantes). Ensuite, nous ajoutons une logique dans la proc de notre nouvelle variable pour imputer la valeur. Le chef du ménage est un enfant si l'âge du chef est inférieur à 18.

```
PROC CHEF_ENFANT

// Calculer la variable selon l'âge du chef de ménage
if AGE(1) < 18 then
   // Chef moins de 18 ans
   impute(CHEF_ENFANT, 1);
elseif AGE (1) = missing then
   // Age du chef inconnu
   impute(CHEF_ENFANT, missing);
else
   // Age du chef supérieur à 18
   impute(CHEF_ENFANT, 2);
endif;
```

Lancez le programme et consultez le rapport d'imputation pour savoir combien de ménages ont des chefs de moins de 18 ans dans notre ensemble de données.

Conversion des cases à cocher en variables oui / non

La case à cocher constitue une interface agréable, mais les données résultantes sont difficiles à interpréter et à gérer dans d'autres logiciels. Pour faciliter l'exportation, nous pouvons convertir la case à cocher en une série de variables avec des options oui / non. Convertissons la variable **LANGUES_PARLEES** en une série de variables oui / non. Créez une série de nouvelles variables oui / non ; une variable pour chaque langue. Mettez ces variables à la fin de l'enregistrement individu. Ajoutez des ensembles de valeurs avec Oui - 1 et Non - 2.

Pour chacune de ces nouvelles variables, nous voulons définir la valeur sur oui si la langue correspondante est cochée dans **LANGUES_PARLEES**. Quelle fonction utilisons-nous pour déterminer si un élément dans un champ de case à cocher est coché ? Comme auparavant, nous utilisons pos().

```
PROC LANGUE_FRANCAISE_PARLEE
if pos("A, LANGUES_PARLEES) > 0 then
   LANGUAGE_FRANCAISE_PARLEE = 1;
else
   LANGUAGE_FRANCAISE_PARLEE = 2;
endif;
```

```
PROC LANGUE_ANGLAISE_PARLEE
if pos("B", LANGUES_PARLEES) > 0 then
   LANGUE_ANGLAISE_PARLEE = 1;
else
   LANGUE_ANGLAISE_PARLEE = 2;
endif;

PROC LANGUE_ESPAGNOLE_PARLEE
if pos("C", LANGUES_PARLEES) > 0 then
   LANGUE_ESPAGNOLE_PARLEE= 1;
else
   LANGUE_ESPAGNOLE_PARLEE= 2;
endif;
```

La logique pour les autres langues suit le même schéma.

Outil d'exportation de données

L'outil d'exportation de données CSPro est disponible dans le menu “Tools” sous “Export Data”. Lorsque vous démarrez l'outil, vous êtes invité à fournir un dictionnaire de données. Choisissez le dictionnaire Popstan2020. À partir de l’écran principal d’exportation, vous pouvez choisir les enregistrements / variables à exporter en cochant les cases correspondantes. Pour commencer, sélectionnons simplement les éléments d’identification et les premiers champs de la section F. Au bas de la fenêtre d’exportation, vous pouvez choisir le format de fichier dans lequel vous souhaitez exporter. Pour cet exercice, nous choisirons un fichier CSV pouvant être facilement ouvert dans Excel. Pour lancer l'exportation, cliquez sur le feu vert. Vous êtes invité à choisir le fichier de données. Nous allons utiliser le fichier de données Popstan2020Apure.csdb. Enfin, vous êtes invité à entrer le nom des fichiers exportés. Une fois l'exportation terminée, les fichiers exportés sont affichés dans l’outil TextViewer. Ouvrons-les dans Excel et voyons ce que nous avons. Notez que chaque ménage est enregistré sur une ligne distincte du fichier Excel.

PROVINCE	DISTRICT	ENUMERATION_AREA	AREA TYPE	HOUSEHOLD_NUMBER	F01	F02	F03
2	14	214	1	1	3	1	3
3	27	301	1	1	9	9	1
1	1	345	1	5	1	1	1
3	26	222	2	2	1	4	1
4	37	422	2	1	1	5	1
2	2	214	3	1	1	2	4
1	1	101	1	1	2	5	1

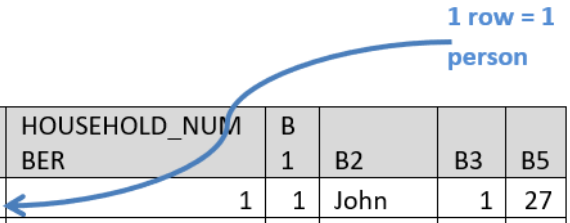
Exportation des enregistrements répétés

Essayons d'exporter quelques champs de la section B: numéro de ligne, nom et sexe. Remarquez que pour chaque variable que nous avons exportée, nous obtenons 30 colonnes. Pourquoi donc ? Nous obtenons une colonne pour chaque occurrence de

l'enregistrement. CSPro exporte toujours chaque ménage sur une seule ligne et, puisqu'il compte jusqu'à 30 personnes dans le ménage, il génère 30 colonnes pour chaque variable. Même les occurrences vides génèrent toujours des colonnes dans la feuille de calcul.

B1_01	B1_02	B1_03	B1_04	B1_05	B1_06	B1_07	B1_08	B1_09	B1_10	B1_11	B1_12	B1_13	B1_14	B1_15	
1	2														
1	2	3													
2															
1	2	3	4												
1	2	3	4												
1	2	3													
1	2	3	4	5	6	7									

Avoir une colonne pour chaque occurrence peut compliquer l'analyse des données. Par exemple, dans ce fichier, il est plutôt difficile de faire quelque chose d'aussi simple que de compter le nombre total de personnes. Si, au lieu de choisir le paramètre par défaut consistant à placer plusieurs occurrences d'enregistrement dans une seule ligne ("All in One Record"), essayez de sélectionner "As Separate Records" (enregistrements séparés). Nous avons maintenant chaque individu du ménage dans une rangée séparée. Lors de cette opération, il est important d'inclure les éléments d'identification du ménage afin d'indiquer clairement les membres de chaque ménage.



PROVINCE	DISTRICT	ENUMERATION_AREA	AREA_TYPE	HOUSEHOLD_NUMBER	B1	B2	B3	B5
2	14	101	1	1	1	John	1	27
2	14	101	1	1	2	Mary	2	26
2	27	200	1	1	1	Jane	2	40
2	27	200	1	1	2	Billy	1	13
2	27	200	1	1	3	Tina	2	13

Exportation de plusieurs types d'enregistrement

Maintenant, essayons d'exporter les premiers articles à la fois de l'enregistrement individu (B) et de l'enregistrement des décès (E). Avec nos paramètres actuels, l'outil nous avertit que seuls les éléments du premier enregistrement seront exportés. Pourquoi ? Le problème est que si vous mettez chaque enregistrement sur sa propre ligne, certaines lignes auront des membres du ménage et d'autres des décès, mais les colonnes ne correspondront pas.

La solution consiste à exporter chaque enregistrement dans un fichier à part en sélectionnant l'option "Multiple Files (one for each record type)" - *Plusieurs fichiers (un pour chaque type d'enregistrement)* sous "Number of Files Created". Cela génère deux fichiers : INDIVIDU.csv qui contient les membres du ménage et DECES.csv qui contient les décès.

PERSON.csv

PROVINCE	DISTRICT	ENUMERATION_AREA	AREA_TYPE	HOUSEHOLD_NUMBER	B1	B2	B3	B5
2	14	101	1	1	1	John	1	27
2	14	101	1	1	2	Mary	2	26
2	27	200	1	1	1	Jane	2	40
2	27	200	1	1	2	Billy	1	13
2	27	200	1	1	3	Tina	2	13

DEATHS.csv

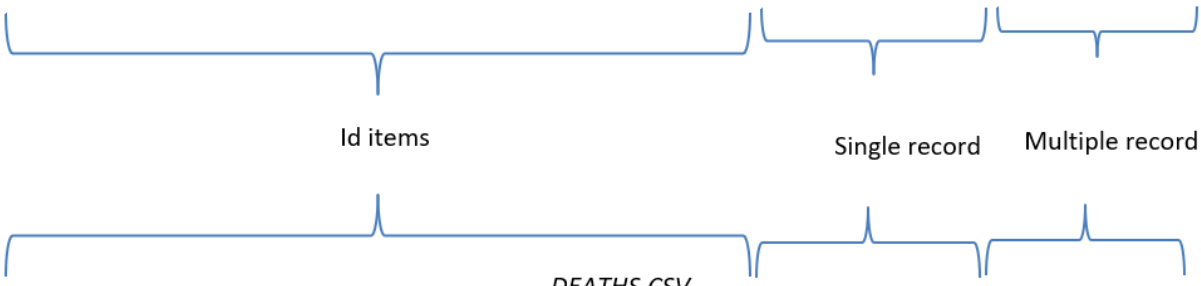
PROVINCE	DISTRICT	ENUMERATION_AREA	AREA_TYPE	HOUSEHOLD_NUMBER	E3	E4	E5
2	14	101	1	1	1	Erwin	20140211
2	14	101	1	1	2	Carmine	20161201
2	27	200	1	1	1	Arnold	20150113

Les ménages de ces deux fichiers peuvent être liés entre eux en fonction des variables identifiantes.

Il est également possible de faire en sorte que les données d'exportation joignent des enregistrements uniques et multiples. Si vous sélectionnez cette option, un fichier sera généré pour chaque enregistrement avec plusieurs occurrences et les colonnes sélectionnées pour tous les enregistrements uniques seront ajoutées à chaque ligne dans chacun des fichiers exportés. Par exemple, si nous choisissons le statut d'occupation (**F06**) et le type de logement (**F03**) dans l'enregistrement unique de l'habitat et choisissons également les premières variables de l'enregistrement des individus, **F06** et **F03** seront ajoutés à chacun des deux fichiers exportés : INDIVIDUS.csv et DECES.csv.

PERSON.CSV

PROVINCE	DISTRICT	ENUMERATION_AREA	AREA_TYPE	HOUSEHOLD_NUMBER	F03	F06	B1	B2
2	14	101	1	1	1	1	1	John
2	14	101	1	1	1	1	2	Mary
2	27	200	1	1	2	3	1	Jane
2	27	200	1	1	2	3	2	Billy
2	27	200	1	1	2	3	3	Tina



DEATHS.CSV

PROVINCE	DISTRICT	ENUMERATION_AREA	AREA_TYPE	HOUSEHOLD_NUMBER	F03	F06	E3	E4
2	14	101	1	1	1	1	1	Erwin
2	14	101	1	1	2	1	2	Carmine
2	27	200	1	1	1	3	1	Arnold

Il n'est pas possible de joindre des enregistrements répétés à d'autres enregistrements multiples à partir de l'exportation.

Notez que dans certains cas, il peut être plus facile d'exporter les enregistrements uniques et multiples sans les joindre, puis de les rejoindre après dans l'autre logiciel après avoir effectué l'importation.

Importation de données dans SAS, SPSS, Stata et R

Lors de l'exportation de données ver les logiciels statistiques, CSPro génère à la fois un fichier de données et un script à exécuter dans le logiciel statistique lui-même pour effectuer l'importation. Pour plus d'informations sur l'exécution de ce script pour chaque logiciel, reportez-vous à l'aide en ligne relative à Export Data et voir la section "How to…".

Sauvegarde de la spécification d'exportation

Vous pouvez enregistrer vos paramètres d'exportation dans un fichier de spécification d'exportation CSPro (.exf). Vous pouvez ensuite double-cliquer sur ce fichier ou l'ouvrir à partir de l'outil d'exportation de données pour récupérer toutes les sélections effectuées.

Exercices

1. Modifiez l'application de traitement par lots pour ajouter un contrôle pour un individu ayant lien de parenté époux / épouse mais dont l'état matrimonial n'est pas marié. Ecrire un message dans le fichier journal pour chaque cas trouvé. Cela devrait être fait dans l'application “batch”, PAS dans l'application de saisie de données.
2. Modifiez l’application de traitement par lots afin d’ajouter une vérification du nombre total de pièces (**F01**) supérieure au nombre de chambres à coucher (**F02**).
3. Modifiez l’application de traitement par lots pour ajouter un contrôle de cohérence du type de logement principal (**F03**) avec le nombre total de chaque type de logement en **F05**. Par exemple, si le logement principal dans **F03** est une maison individuelle, le nombre total de maisons individuelles dans **F05** ne doit pas être égal à zéro.
4. Modifiez l'application de traitement par lots pour convertir les handicaps (**B11**) de chaîne de caractères (utilisé pour les cases à cocher) en une série de variables numériques oui / non. Créez les nouvelles variables dans le dictionnaire et écrivez une logique pour définir la valeur de chacune des nouvelles variables en fonction des sélections effectuées dans **B11**.
5. À l'aide du fichier de données de test, Popstan2020Apure.csdb, exportez les membres du ménage ainsi que les variables identifiantes vers le logiciel de votre choix (Excel, SPSS, Stata, SAS ou R). Utilisez ce logiciel pour trouver le nombre total de personnes par sexe (total, hommes, femmes) ainsi que le nombre de personnes par sexe pour le district 01.
6. Exportez l'enregistrement des décès et l'enregistrement des individus avec les variables identifiantes vers le logiciel de votre choix. Vous devez exporter tous les enregistrements en même temps, pas un par un. Combien de ménages ont plus d'un décès ? Combien de ménages ont des enfants de moins de 5 ans ?