COLEGIO DE POSTGRADUADOS
INSTITUCION DE ENSEÑANZA E INVESTIGACION EN CIENCIAS AGRICOLAS

CAMPUS MONTECILLO

POSTGRADO DE SOCIOECONOMÍA, ESTADÍSTICA E INFORMÁTICA
COMPUTO APLICADO

UN SISTEMA DE CONSULTAS PARAMÉTRICAS EN INTERNET PARA INVESTIGADORES AGRÍCOLAS

JUAN CARLOS ALARCÓN MALDONADO

TESIS

PRESENTADA COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS

MONTECILLO, TEXCOCO, EDO. DE MÉXICO
2008

La presente tesis**,** titulada: **Un Sistema de Consultas Paramétricas en Internet para Investigadores Agrícolas**, realizada por el alumno: **Juan Carlos Alarcón Maldonado**, bajo la dirección del Consejo Particular indicado, ha sido aprobada por el mismo y aceptada como requisito parcial para obtener el grado de :

MAESTRO EN CIENCIAS
SOCIOECONOMIA, ESTADISTICA E INFORMATICA
COMPUTO APLICADO

CONSEJO PARTICULAR

CONSEJERO : _____

DR. JUAN RICARDO BAUER MENGELBERG

ASESOR: _____

DR. DAVID H. DEL VALLE PANIAGUA

ASESOR: _____

M.C. EDGAR RAMÍREZ GALEANO

MONTECILLO, TEXCOCO, EDO. MÉXICO 18 DE JUNIO DEL 2008

AGRADECIMIENTOS

A LAURA, MI ESPOSA

QUE, UNA VEZ MAS, ME BRINDA SU TIEMPO, APOYO, PACIENCIA Y CARIÑO PARA CUMPLIR MIS OBJETIVOS.

A JUAN CARLOS Y MARIFER, MIS HIJOS

QUE SON LO MAS PRECIADO QUE DIOS ME HA DADO.

A MI MADRE

POR DARME LA VIDA Y ENSEÑARME A LUCHAR POR LO QUE UNO DESEA.

A MI TÍO MIGUEL,

DEL CUAL APRENDI QUE LOS LIMITES DEL SER HUMANO SON AQUELLOS QUE EL MISMO SE IMPONE

UN AGRADECIEMINTO ESPECIAL PARA EL DR. J. R. BAUER QUE, SIN DUDA, FUE UNA PARTE CRUCIAL PARA LA REALIZACION DE ESTA META.

UN SISTEMA DE CONSULTAS PARAMÉTRICAS EN INTERNET PARA INVESTIGADORES AGRÍCOLAS

Juan Carlos Alarcón Maldonado, M.C.
Colegio de Postgraduados, 2008

RESUMEN

CIMMYT (Centro Internacional de Mejoramiento de Maíz y Trigo) tiene datos de calidad sobre los experimentos que se realizan en todos sus centros experimentales, y los procesos de concentración de la información se han afinado notablemente. Sin embargo, el acceso a estos datos está limitado en dos aspectos: sólo se pueden acceder vía una red local, o solicitando el envío de un dispositivo con los datos de interés, y aunque hay algunos informes programados, para obtener algún otro, las demoras son muy considerables. El trabajo de investigación que se documenta en esta tesis de Maestría resultó en el sistema CROPFINDER, que resuelve ambas situaciones. Se creó una base de datos que permite al sistema ofrecer las consultas que necesitan los investigadores o colaboradores del Centro para ambos cultivos (maíz y trigo), puesto que usa las mismas estructuras a pesar de la diferencias en tecnología y estructura de datos entre los sistemas actuales para esos dos cultivos. Cada investigador puede crear sus propios informes, seleccionando los atributos de interés, mismos que se convierten en columnas de los cuadros resultantes, y aplicar una selección del tipo de variedades, localidades, fechas u otros criterios para limitar el cuadro a la información solicitada. El sistema arma una sentencia y ejecuta un comando tipo SQL, lo que a su vez resulta en un cuadro que aparece en el monitor del usuario, quien puede copiar este objeto a una hoja de cálculo Excel, en especial para imprimirlo total o parcialmente.

Palabras clave: Consultas paramétricas, maíz, trigo, trait

AN INTERNET BASED SYSTEM OF PARAMETERIZED QUERIES FOR AGRICULTURAL RESEARCHERS

Juan Carlos Alarcón Maldonado, M.C.
Colegio de Postgraduados, 2008

ABSTRACT

CIMMYT (International Maize and Wheat Improvement Center) has quality data about the experiments which are conducted in their different locations, and the processes which concentrate the information are now producing good results. However, the access to the data is limited in two ways. They are only available on a local network or through a special request, which will result in the sending of a file with the data of interest. Additionally, though there are some reports programmed to obtain information, to obtain a different report a special request must be formulated, which sometimes, if at all, is only satisfied after a considerable delay. The research described in this Master's Thesis resulted in a system called CROPFINDER, which removes both these limitations. A database which allows the system to offer the queries needed by the Center's researchers or cooperators in both crops (maize and wheat) was created, since it uses the same data structures notwithstanding the current difference in technology and data organization between the systems for these two crops. Furthermore, every researcher may create his own reports, by choosing the attributes which he needs, which will in turn constitute the columns of his reports. When executing a process to obtain a specific report, he may define a selection of the types of varieties, locations, dates or other criteria to limit the resulting report to those experiments which satisfy the conditions he specifies. The system constructs an SQL command, which in turn results in a table which appears on the user's screen, who may copy it into an Excel file, especially if he wishes to print any part of it.

Key words: Parameterized queries, maize, wheat, trait

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

INTRODUCCIÓN GENERAL

Introducción a la investigación

En muchos centros de investigación se ha enfatizado la recolección o recepción de información cada vez más precisa y actualizada. Naturalmente, esto es deseable, pero no alcanza, puesto que los que necesitan la información – para los cuales se la consigue – no pueden aprovecharla dado que las herramientas que se les ofrecen para su usufructo son insuficientes o no están disponibles en los lugares de trabajo de los interesados.

El proyecto de investigación que se describe se formuló precisamente para incrementar las posibilidades, en cuanto al uso de los datos, de los investigadores del Centro Internacional de Mejoramiento de Maíz y Trigo (CIMMYT), inicialmente para datos de maíz y trigo, pero contemplando que se pudieran aplicar a otros cultivos. Para ello, se estudiaron los datos que se recolectaban y las estructuras en las que se almacenaban en la computadora, sin dejar de analizar los procedimientos involucrados en la recepción y proceso de los datos que llegaban de todo el mundo.

El análisis arrojó varios resultados interesantes, entre los cuales vale la pena destacar que CIMMYT ha implementado una estandarización que le permite utilizar datos de diferentes centros experimantales en forma conjunta. Por el contrario, las tecnologías que se usan en los sistemas de Maíz y de Trigo son muy distintas. Finalmente, se analizaron los informes disponibles, y se determinaron dos grandes faltantes. En primer lugar, los datos no están disponibles excepto por peticiones específicas, las que resultan en envíos de un dispositivo o archivos al solicitante, tarea que realiza un sistema denominado IWIS (Payne et al. 2001). En segundo término en cuanto a las facilidades de obtener información, los reportes son fijos, lo que hace que sean insuficientes para ciertos interesados, mientras que otros reciben información superflua. Hay un procedimiento para solicitar nuevos reportes, pero en general el tiempo de respuesta entre la solicitud y la implementación del nuevo programa es relativamente largo.

Para establecer la terminología en cuanto a las características de interés que constituyen el objetivo principal del sistema de consultas, se definen con precisión los términos principales asociados a las variedades de las plantas. Todas estas definiciones provienen de “El Sistema de la UPOV de Protección de Variedades Vegetales” (UPOV, 2001).

Con un ejemplo que ilustra la clasificación taxonómica del trigo blando, se define la especie del siguiente modo.

División: Spermatophyta

Clase: Liliopsida (monocotiledónea)

Orden: Poales

Familia: Poaceae

Género: Triticum

Especie: Triticum aestivum L. (Trigo blando)

Se reproduce aquí en forma textual la definición de variedad vegetal del Convenio de la UPOV, que comienza declarando que se trata de "un conjunto de plantas de un solo taxón botánico del rango más bajo conocido...". Ello confirma que una variedad vegetal resulta de la subdivisión más baja de la especie. Sin embargo, para comprender mejor qué es una variedad vegetal, el Convenio de la UPOV, en su Artículo 1.vi),la define de la manera siguiente: "un conjunto de plantas de un solo taxón botánico del rango más bajo conocido que, con independencia de si responde o no plenamente a las condiciones para la concesión de un derecho de obtentor, pueda

- definirse por la expresión de los caracteres resultantes de un cierto genotipo o de una cierta combinación de genotipos,
- distinguirse de cualquier otro conjunto de plantas por la expresión de uno de dichos caracteres por lo menos,
- considerarse como una unidad, habida cuenta de su aptitud a propagarse sin alteración."

Los estudios se realizan con un tipo de semilla de una variedad. Por lo tanto es importante conocer datos adicionales de las semillas, mismas que se describen con su pedigrí y su historial[1].

En este trabajo, se decidió postergar el anuncio del resto de la tesis, para incluirlo en la sección de metodología, precisamente porque ésta fue la que dictó la organización del material. De este modo, tras anunciar en la sección siguiente los objetivos del proyecto, se describe la metodología y el orden de actividades y descripciones de las mismas. De este modo, tras enunciar los objetivos, se describen los antecedentes del proyecto de investigación.

OBJETIVOS

Objetivo General

Proveer a los investigadores del CIMMYT y a personal externo a dicho organismo, un modo de conseguir la información necesaria para realizar sus trabajos de investigación.

Objetivos especiales

En particular, se trata de poner a su disposición datos sobre variedades y semillas de Maíz y Trigo, con detalles acerca de los pedigrees e historiales de cada variedad.

Como componente importante de los datos proporcionados estarán los resultados de los experimentos efectuados para cada una de las variedades.

Se ofrecerá el servicio vía Internet, con una clave de acceso para cierto tipo de datos, mientras otros estarán disponibles para el público en general.

El sistema deberá ser el mismo para ambos cultivos, y adaptable a otros si se diera el caso de que se incluyeran otros en los proyectos de investigación.

[1] El vocablo pedigrí (del inglés *pedigree*): The pedigree notation used by CIMMYT and many other breeding institutions is based on what is commonly known as the Purdy method (Crop Science, Vol. 8, July-August 1968). This notation gives an explicit description of the crossing and selection history of a line. Such descriptions tend to become rather long after a few generations of crossing and are therefore given an abbreviated name. The problem then becomes one of recognizing and remembering what the abbreviations stand for. The problem is further compounded when several sister lines are given different abbreviations. Another source of confusion results from reassignment of names by commercial seed companies and national breeding programs. (ICIS, 2008)

METODOLOGÍA

Para el diseño del sistema se usó el Enterprise Architect y por ende, su metodología. Se decidió no incluir los documentos elaborados a detalle en esta tesis. Los pasos a seguir de acuerdo a la teoría que subyace a esta metodología son los siguientes.

- Se estudia a fondo el sistema, con el objetivo de formular la funcionalidad necesaria para los futuros usuarios, y detectar alguna situación desfavorable que pudiera impedir el uso extendido de los datos.
- Se crea un modelo especial para dar cabida a la difusión de los datos.
- Se definen los tipos de consultas a programar – con la aclaración de que no serán realmente consultas puntuales, sino consultas posibles donde los usuarios diseñarán y elaborarán sus consultas, pero sin necesidad de aplicar conocimientos de computación.
- Se implementa la base de datos, y se elaboran los programas que actualizan los datos a partir de la base de CIMMYT. Se definen procedimientos para ejecutar estos programas de modo de tener la información actualizada on line.
- Se elaboran los programas que permitirán la confección de consultas y sus resultados.
- Se documenta y divulga el sistema nuevo.
- Se atienden preguntas, se resuelven dudas y se recopilan sugerencias y críticas, mismas que se incorporan a la siguiente versión del sistema. Naturalmente, si surgen deficiencias o emergen errores, se corrigen de inmediato.

Esta lista de pasos o etapas del diseño e implementación a su vez sirvió de organizador del resto de este trabajo. Como parte del estudio del sistema, se incluye una sección de los antecedentes, seguida del Capítulo 1, que contiene la descripción técnica y funcional de los sistemas de Maiz y de Trigo del CIMMYT, con una breve descripción del servicio que proporcionan en cuanto a la divulgación de la información.

El Capítulo 2 está dedicado al diseño del sistema CROPFINDER, que es el que permitirá la nueva modalidad de las consultas a los datos existentes. Se describe

brevemente el servicio que prestará, para permitir la explicación de las entidades y campos de la base de datos que utilizará este nuevo sistema.

El Capítulo 3 está dedicado a la descripción del servicio, mostrando cómo funcionarán las consultas paramétricas, término que se definió previamente en la Introducción. Ahí se muestran las interfases con las que los usuarios pueden efectuar las tareas que ofrece el CROPFINDER: crear y ejecutar reportes, y guardarlos o imprimirlos si fuera necesario o conveniente.

El Capítulo 4 contiene la descripción técnica de los procesos que permiten ofrecer la funcionalidad del CROPFINDER, mientras que el Capítulo 5 refleja algunas mejoras sugeridas para la siguiente versión. El trabajo finaliza con una serie de conclusiones.

ÁREA DE ESTUDIO Y MATERIALES

El proyecto se enmarca en el área de Sistemas de Información, Bases de Datos, publicacíon de información en la Red de Redes y el concepto de consultas paramétricas.

Los materiales en cuanto a tecnologías fueron Visual Basic 6.0, ASP.NET, SQL scripts, EXCEL, ACCESS 2000, SLQ Server, en diversas versiones de WINDOWS, incluidas 2000, 2003 y XP Professional. Además se utilizó IIS de Windows para las pruebas y la depuración de los componentes de Internet.

Se utilizó el equipo del investigador, computadoras para el desarrollo, y la implementación de los componentes de Internet se instalaron en el site del CIMMYT

CAPÍTULO 1. LOS SISTEMAS DE INFORMACIÓN EN CIMMYT

1.1 Introducción

CIMMYT logró, tras muchos años de trabajo de numerosos grupos, estandarizar muchos de los datos que necesitan los investigadores para su trabajo. Se pueden mencionar algunos aspectos en especial, pero basta con decir que un investigador desea saber todo lo que hay publicado e investigado de alguna variedad de interés, incluyendo las características, el pedigrí, el historial de la variedad, y los resultados de experimentos realizados con dicha variedad. Como estas investigaciones se pudieron haber realizado en numerosas localidades[2], fue fundamental establecer procedimientos y procesos para concentrar la información y presentarla en forma uniformada, de tal modo que se pudieran comparar resultados aun cuando el registro original se hubiera realizado con sistemas, unidades de medida y nomenclaturas diversas.

CIMMYT tiene un sistema para TRIGO, y otro, diferente en varios aspectos, especialmente en los que se refieren a las tecnologías informáticas utilizadas, para el MAIZ. Se describe muy brevemente cada uno de estos sistemas, cuyo estudio concluye con una sección sobre el servicio que prestan a la comunidad del CIMMYT y al público en general.

1.2 El sistema de Maíz

Primero se describe, con ayuda de la Figura 1, el proceso y la naturaleza del servicio en forma gráfica. El investigador llena el libro de campo (*Fieldbook)* que emite el sistema y se le envía para tal efecto al que realizará el experimento. Cabe señalar que para la obtención de datos de trigo, el sistema IWIS (Payne *et al.* 2001) ofrece a los investigadores de campo dos opciones. Aquí también se genera un libro de campo que se envía, impreso, al investigador. El colaborador que hace el estudio puede llenar los datos, escribiendo los resultados, en dicho impreso, mismo que envía a CIMMYT para su captura e incorporación a las bases de datos. La otra modalidad es

[2] A pesar de que este vocablo no significa lo mismo en español, en los sistemas de CIMMYT se usa este término como traducción de "location" que es la designación de los centros de investigación u otros sitios donde se realizan experimentos.

que, en su computadora, el colaborador actualiza la información en un conjunto de hojas de cálculo, que para dicho fin se le proporcionaron como herramientas de trabajo. A continuación, envía dichos archivos (en Excel) para su proceso automático de incorporación a las base de datos.

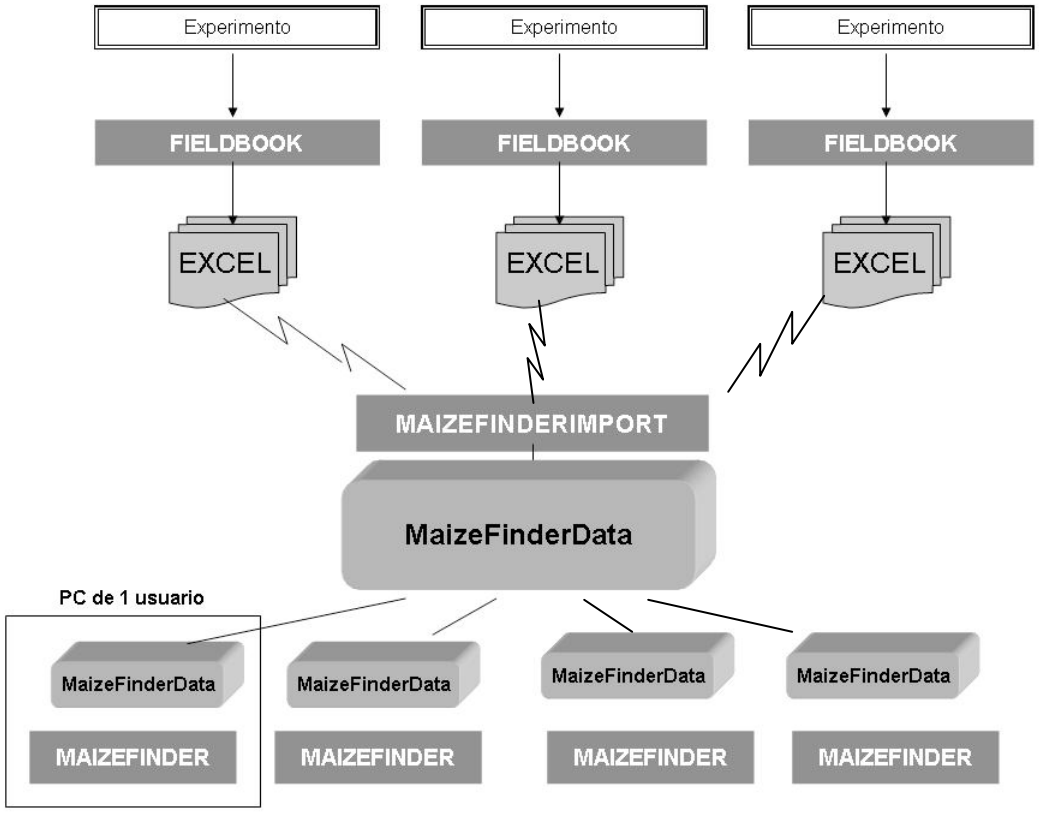

Figura 1. Diagrama de proceso de los datos de experimentos de Maíz

A continuación se describen las entidades y campos principales de la base de datos MaizeFinderData, que es la que se usa para proporcionar las consultas. La Figura 2 muestra en forma esquemática dichas tablas. El Apéndice A contiene descripciones de los campos de estas tablas.

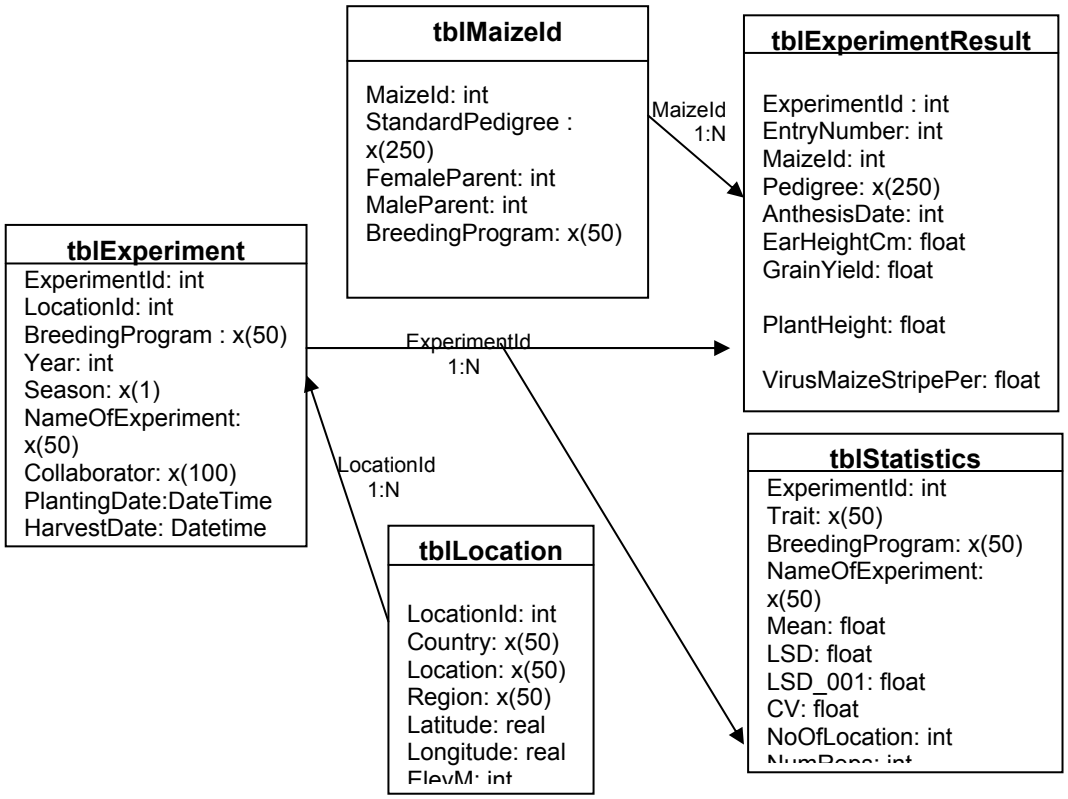

Figura 2. Principales tablas del sistema de "Maíz"

1.3 El sistema de Trigo

Para su usufructo, el sistema de Trigo usa 5 bases de datos, todas las cuales están comunicadas entre sí, como muestra la figura 3.

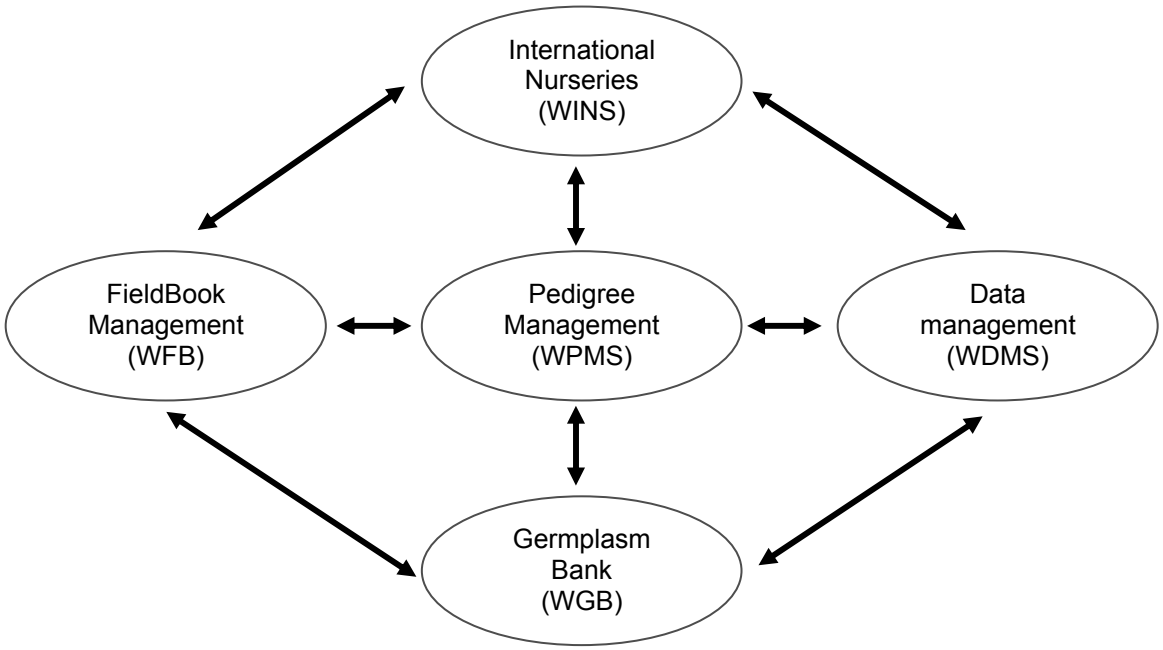

Figura 3. Las bases de datos de IWIS

A continuación se describen las principales tablas, que no están todas en una sola base. La nomenclatura de las tablas refleja la base a la cual pertenece.

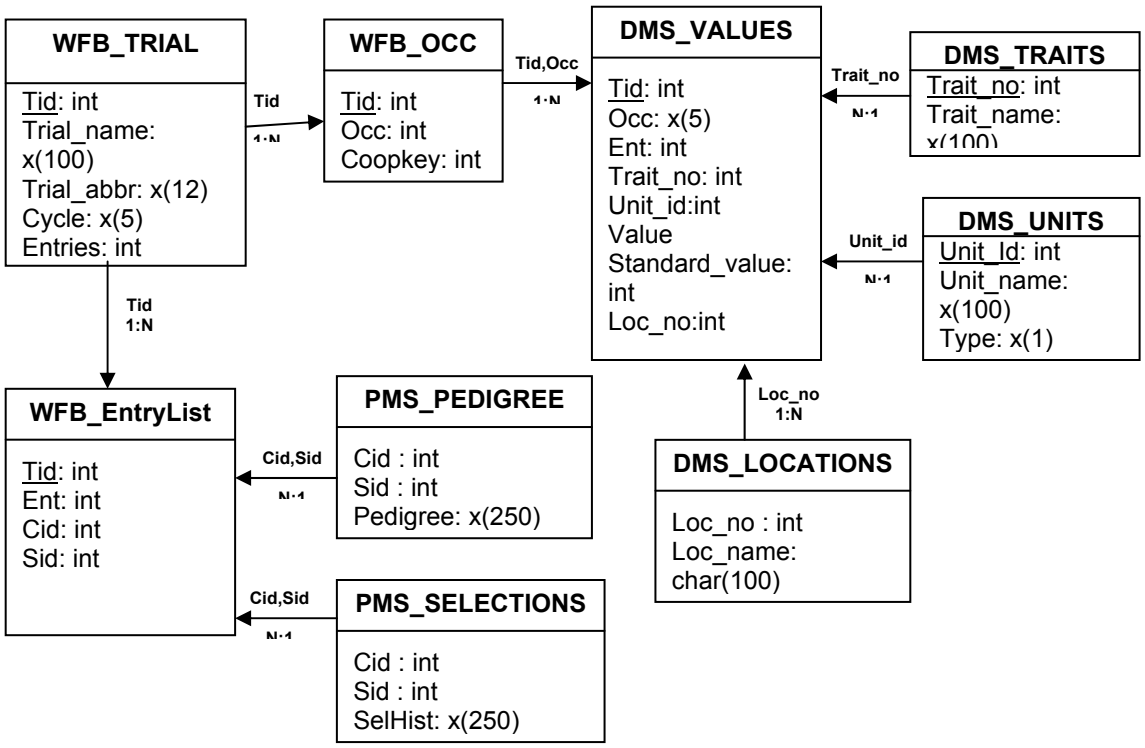

Figura 4. Principales tablas del sistema de “Trigo”

Se describen a continuación los principales conceptos, con el objetivo de permitir al lector entender la naturaleza del uso de los datos en el contexto de un investigador que pretende averiguar características de alguna variedad. En el Apéndice A hay una descripción más detallada de los campos.

Tid. Identificación de Trial (estudio).

Entries. Número de variedades que componen un estudio.

Cid. Identificación de pedigrí o cruza.

Sid. Identificación de Historia de selección.

Trait_no. Identificación de trait (característica).

Unit_id. Identificación de la unidad.

Occ. Número de conjunto de semilla de un estudio que se envió.

Coopkey. Identificador del cooperador o investigador.

Loc_no. Identificación de la localidad.

1.4 Comentarios sobre el servicio que presta el sistema de Trigo

Para el uso del personal que está conectado a la red local, hay una serie de consultas e informes. Aparecen los resultados en el monitor, pero se pueden guardar archivos en EXCEL o solicitar impresiones. Las consultas se hacen por estudio, y siempre salen todos los datos referentes al estudio seleccionado. Este sistema tiene una gran ventaja sobre el descrito para Maíz: los datos siempre están actualizados, puesto que se usan los del sistema que los concentra.

Si alguien necesitara un informe diferente a los disponibles, tendría que solicitarlo como proyecto de desarrollo, de modo que no sólo tardaría mucho tiempo en obtenerlo, sino que no tendría ninguna garantía de que su solicitud entrara al programa de trabajo de los grupos involucrados.

1.5 Comentarios sobre el servicio que presta el sistema de Maíz

Este sistema no está en una red local. Es una aplicación en la que se incorporan los datos recibidos con cierta periodicidad en la base de datos ACCESS, descrita anteriormente. Esta base se distribuye a los usuarios interesados, mismos que la instalan en su computadora y es ahí donde pueden hacer consultas, usando un sistema denominado MaizeFinder . La ventaja comparada con el de Trigo es que sí permite consultas a datos de más de un estudio, y con ciertas selecciones. Sin embargo, los datos no siempre estarán al día, sino usa los del último envío.

CAPÍTULO 2. EL SISTEMA CROPFINDER

2.1 Introducción

El diseño del nuevo sistema comenzó por la formulación de una estrategia en cuanto al modo en el que se ofrecería en el futuro la información, que se puede resumir como una consulta paramétrica disponible en Internet (ADR Infor S.L 2008). Esto implicó crear una estructura nueva de datos para poder ofrecer este servicio, como también la definición de los procesos para actualizar la misma a partir de los datos existentes, que se respetarían para no afectar la operación, misma que funciona y ha resultado en datos de calidad.

A continuación se especificaron los tipos de consultas que les podrían interesar a los investigadores, manteniendo siempre el criterio de que no sólo se proporcionarían los informes solicitados, sino los que pudieran aparecer en un futuro. Se estudió un caso concreto para abstraer el tipo de información que busca un investigador, y el nivel de detalle al cual le interesa obtener los datos. Naturalmente en esta etapa de diseño intervinieron investigadores y personal especializado en sistemas del organismo.

La determinación de la naturaleza de las consultas que se ofrecerían, resultó en la definición de los aspectos funcionales y técnicos, que se describen en tres secciones: el concepto de una consulta paramétrica y los componentes de las mismas; la base de datos diseñada para poder ofrecer estas consultas; el proceso necesario para actualizar la nueva base de datos a partir de las de los sistemas de recopilación de la información, tanto de Trigo como de Maíz. Para concluir la descripción se menciona el perfil de los usuarios y cómo se usa la identificación del que invoca el sistema para ofrecerle servicios.

2.2 Los componentes de las consultas paramétricas

Se ilustrará con un ejemplo el modo de crear un nuevo informe, consistente en las “columnas” que lo integrarán, por parte de un usuario que lo desea, en forma interactiva y sin conocimientos de bases de datos, estructuras o lenguajes de

programación. Para completar la descripción del servicio, se describe cómo se definen y almacenan criterios de selección de los datos que se incluirán en un reporte, los que constituirán las hileras del mismo. De este modo, el sistema CROPFINDER tiene los siguientes componentes, mismas que están en operación.

- Una base de datos compuesta por dos partes: los datos mismos y los elementos que permiten la construcción de una consulta en particular. Además, se guardan las consultas mismas.
- Los procesos que actualizan la base de datos con los que residen en las bases de trigo y de maíz.
- Los programas que permiten construir y ejecutar un informe.
- La documentación e instructivos para usuarios.

2.3 Diseño de la base de datos

Para ofrecer consultas por Internet a la información de CIMMYT, se diseñó una estructura de base de datos siguiendo los criterios recomendados para este tipo de bases (Silberschatz et al, 2005; Rainard 2002). Las bases son idénticas para ambos cultivos. La interfase de usuario se desarrolló utilizando ASP.NET (Esposito 2002). Se permitiría a un investigador construir su propio informe, es decir, especificar las columnas que desea. Además se le ofrecería la opción de definir criterios de inclusión de los registros (renglones) del informe. Para ello, se desarrollaron procedimientos de transferencia de datos a partir de las estructuras de datos existentes de cada cultivo, para incorporarlos a la nueva base de datos, misma que utiliza Microsoft SQL (Rainard 2002) y SQL Standard para los procedimientos de transferencia. Por supuesto, estos procedimientos son diferentes para los dos cultivos, debido a las características de almacenamiento de los datos de cada uno de ellos.

Se han separado las descripciones de las dos partes de la base de datos. En la Figura 5, se muestran las tablas de datos de CROPFINDER, como se denominó este servicio, y en la Figura 6 se mostrarán las tablas técnicas.

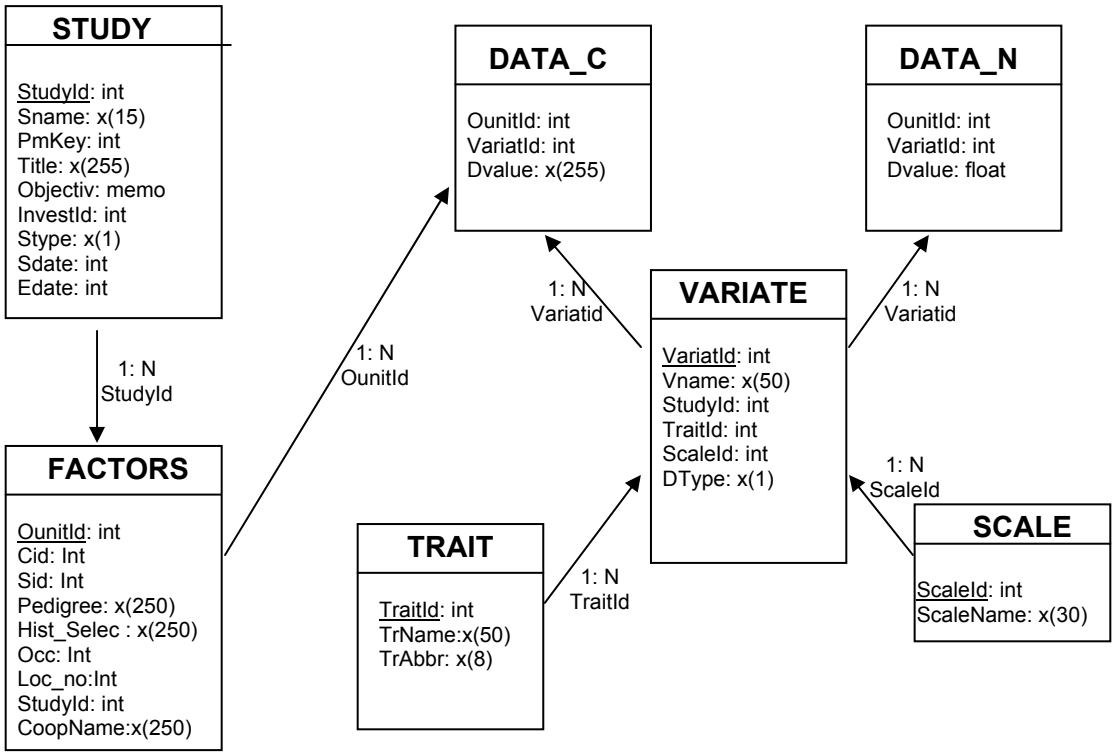

Figura 5. Tablas de las bases de datos que utiliza el sistema CROPFINDER

La tabla Study contiene la información general de un estudio. Un estudio representa una lista de una o más variedades (semillas) que se utilizarán en un experimento. Cada estudio tiene un identificador único, una abreviatura y el nombre completo del estudio, las fechas de inicio y final del estudio, el tipo de estudio, un identificador del investigador o usuario y , si se desea, un campo para hacer algún comentario o anotación referente al experimento.

La tabla Factors contiene los elementos que componen la lista de variedades de un estudio, más la información de identificación de cada uno de dichas variedades, el lugar donde se sembró el estudio, la identificación de su historia de selección, su *pedigree* y otros datos que pueden variar según el tipo de estudio. La información referente a las columnas de la tabla “Factors” dependerá del cultivo e información que se desee. La tabla “Factors” se agregó por motivos técnicos para facilitar la construcción de las consultas, al concentrar en una sola tabla información que originalmente se encontraba en varias de éstas.

Las tablas Data_N y Data_C contienen la información recolectada de las características que se observaron en el estudio de cada variedad. Si el tipo de valor de un *trait* es numérico, se almacena el valor en la tablas Data_N, mientras que se usa la tabla Data_C para los valores tipo cadena de caracteres.

En la tabla Variate se incluyen las características y las unidades de medición que se utilizan para el registro de la información observada. Las otras tablas son en realidad catálogos: Traits contiene la información de las características, mientras que Scale tiene las unidades de medición para las características.

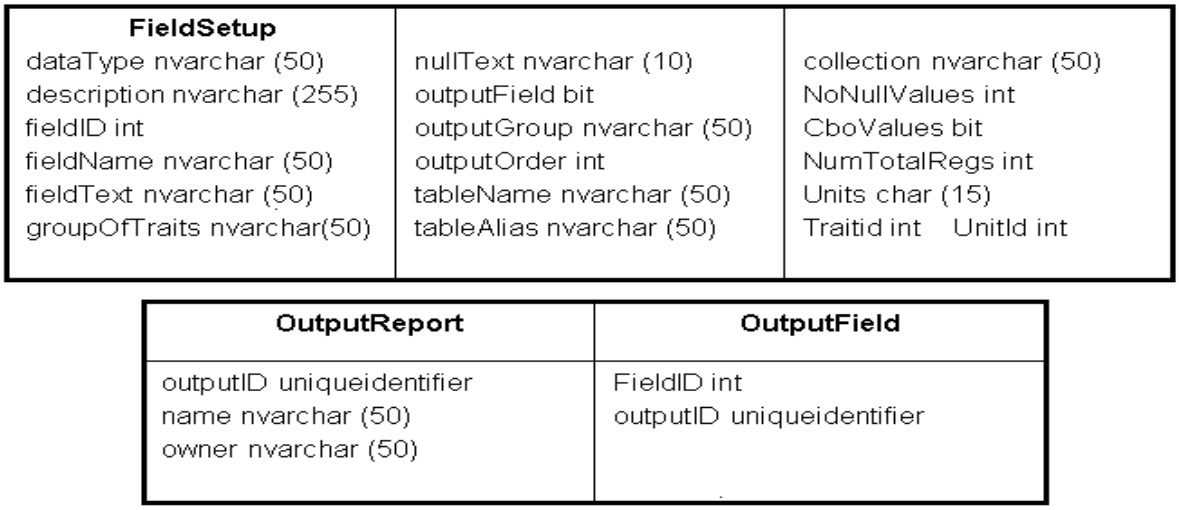

Figura 6. Tablas de control utilizadas para la construcción de sentencias SQL

La base de datos contiene, además, tablas de control que permiten construir las sentencias SQL que a su vez proporcionan las consultas paramétricas. La figura 6 muestra estas tablas en forma esquemática.

FieldSetup es la tabla esencial para la construcción de las sentencias SQL con las que se ofrecen las consultas paramétricas, puesto que contiene la información de los campos que componen la base de datos. Debido a que los valores de las características se encuentran concentrados en una misma tabla, es necesario diferenciar las características, por lo que se asignará un nombre alterno, tanto al campo original, como a la tabla a la que pertenece. A cada campo se le asigna un identificador numérico único, mismo que permitirá obtener la información correspondiente a ese campo, si el usuario la incluye en su consulta. En el paquete

de instalación de CROPFINDER se encuentran los “scripts” necesarios para crear y llenar esta tabla.

Todo reporte (informe) contiene varias columnas de información. En la tabla OutputReport se almacena una breve descripción del reporte (name) , la identificación del usuario(owner) que creó el reporte y un identificador único del informe, que designamos con OutputId, que sirve para ligar el reporte con las columnas que lo componen, que se almacenan en la tabla OutputField.

2.4 Procesos para transferir la información a las bases nuevas

Es importante señalar que se trata de procesos diferentes para el caso de cada uno de los cultivos. Por lo tanto, se describen los procesos por separado. La estructura de la base CROPFINDER es la misma para Maíz que para Trigo, pero son instancias separadas para los datos correspondientes.

2.4.1 Proceso de actualización de la base de CROPFINDER a partir de las bases de datos de IWIS (Trigo)

Se crearon procedimientos almacenados - de hecho, son SQL Scripts – que se incluyen en las bases de IWIS, para que cualquier cambio a datos de las tablas de dichas bases se refleje en forma automática en las tablas de CROPFINDER. Como no se pueden reproducir todos estos scripts, por la extensión de los mismos, se muestran los conceptos con dos ejemplos.

Se documenta aquí el efecto de la inclusión de información sobre la base de datos CROPFINDER. El cuadro 1 muestra el script principal que ejecuta esta actualización- Dichas instrucciones se incorporan a la tabla CRTL_OT de la base de datos WDMS de IWIS.

Cuadro 1. Proceso almacenado para que cambios en la tabla TRAITS se reflejen en el sistema CROPFINDER

declare @v_labelId integer, @v_factorid integer, @v_factorid_std integer,@v_studyid integer,	insert into factor values(@v_labelid, @v_factorId, 'SELECTION HISTORY',@v_StudyId ,251,1432,17,'C')

```
        @v_tid integer,@v_occ integer, @v_levelno
integer,
        @v_levelno_n integer,@v_levelno_c integer,
        @v_trial_name varchar(30), @v_location
varchar(30),
        @v_location_no integer, @v_cycle varchar(5),
        @v_flag_genotypes bit, @v_OunitID integer,
        @v_represno integer, @v_dmsattr integer

declare cursor_tid scroll cursor for
select distinct tid from crtl_ot where procesad = 'T'
and class = 'GE' and
tid in ( select distinct tid from wdms..randomizations)
and
tid in ( select distinct tid from wfb..wfb_occ where
fb_class = 'I' or fb_class = 'F') and last_update =
Now() order by tid

open cursor_tid
fetch cursor_tid into @v_tid
while (@@fetch_status = 0)
  begin
    if (select estatus from controlproceso) = 0
      return
    if exist(select * from study where studyid =
@v_tid)
       exec sp_delete_study @v_tid
    insert into control
    values(@v_tid,null,'F','Y')
    insert into proceso(tid,firstInsert)
    values(@v_tid,getdate())
    select @v_labelId = max(labelid), @v_factorid =
    max(labelid) from factor
   select @v_studyid = @v_tid
   insert into study(studyid,sname,title)
   select tid,trial_abbr, trial_name from wfb..wfb_trial
where
   tid = @v_tid
--  Get FactorIId
    if @v_factorId is null
       select @v_FactorId = 1  else
    exec sp_GetNextDmsAttrId @v_dmsattr output
    insert into dmsattr
    values(@v_dmsattr,801,'FACTOR',@v_labelid,
           'Selection history')
    select @v_labelid = @v_labelId + 1
    insert into factor
    values(@v_labelid, @v_factorId, 'CROSS
NAME',
           @v_StudyId ,251,92,17,'C')
    exec sp_GetNextDmsAttrId @v_dmsattr output
    insert into dmsattr
    values(@v_dmsattr,801,'FACTOR',@v_labelid,
           'Cross name')
--  Factor: Summary value
    select @v_labelid = @v_labelId + 1,
          @v_factorid_std = @v_labelid
    insert into factor
    values(@v_labelid, @v_factorId_std,
           'SUMMARY',@v_StudyId ,251,135,0,'C')
    exec sp_GetNextDmsAttrId @v_dmsattr output
    insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Sum
mary')
    -- Add Level_C and Level_N values
    select distinct @v_labelid = labelId, @v_factorid
= factorid
  from factor where studyid = @v_StudyId and
fname like
                        'Study'
    select distinct @v_trial_name = trial_abbr from
           wfb..wfb_trial where tid = @v_tid
    exec SP_GetLastLevelNo @v_levelno output
    insert level_c

values(@v_labelid,@v_factorid,@v_levelno,@v_trial
_name)
    select distinct @v_labelid = labelId, @v_factorid
= factorid from factor where studyid = @v_StudyId
and fname like 'Tid'
    insert level_n
```

```
        select @v_FactorId = @v_FactorId +1
--   Get LabelId
     if @v_labelid is null
        select @v_labelId = 1      else
        select @v_labelid = @v_labelId +1
--    Insert Study factor
     insert into factor
     values(@v_labelid, @v_factorId,
      'STUDY',@v_studyId,201,134,123,'C')
     exec sp_GetLastRepresNo @v_represno output
--    Insert Effect Study 1st. Representation
     insert into effect
     values(@v_represno,@v_factorid,@v_represno)
     exec sp_GetLastRepresNo @v_represno output
--    Insert Effect Study 2do. Representation
     insert into effect
     values(@v_represno,@v_factorid,@v_represno)
     select @v_labelid = @v_labelId +1
     insert into factor
     values(@v_labelid, @v_factorId,
      'TID',@v_studyId,217,1558,16,'N')

     exec sp_GetNextDmsAttrId @v_dmsattr output
     insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Trial
     Identification')
     select @v_labelid = @v_labelId +1,
            @v_factorId = @v_factorid +2
     insert into factor
     values(@v_labelid, @v_factorId,
              'OCC',@v_studyId,218,224,1428,'N')
     exec sp_GetNextDmsAttrId @v_dmsattr output
     insert into dmsattr
values(@v_dmsattr,801,'FACTOR',@v_labelid,'Occu
rrence')
--    Add Effect
     insert into effect
     values(@v_represno,@v_factorid,@v_represno)
     select @v_labelid = @v_labelId +1
     insert into factor
```

```
values(@v_labelid,@v_factorid,@v_levelno,@v_tid)
     exec SP_GetLastOindex @v_OunitID output
     select @v_flag_genotypes = 1
     update proceso
       set sp_add_level_c_n = getdate()
     where tid = @v_studyid
     declare cursor_occ scroll cursor for
     select distinct occ from wdms..crtl_ot where tid =
@v_tid and class = 'GE' and procesad = 'T' order by
occ
     open cursor_occ
     fetch cursor_occ into @v_occ
     while (@@fetch_status=0)
       begin
         select distinct @v_location = locname,
@v_location_no = l.codeno from cimmytlocs l,
wdmspgd..means_r m where tid = @v_tid and occ =
@v_occ and l.codeno = m.loc_no
         select @v_cycle = cycle from wfb..wfb_occ
where tid = @v_tid and occ = @v_occ
         exec SP_GetLastLevelNo @v_levelno output
         select distinct @v_labelid = labelId,
@v_factorid = factorid from factor where studyid =
@v_StudyId and fname like 'Occ'
         -- Add Occ
         insert level_n

values(@v_labelid,@v_factorid,@v_levelno,@v_occ
)
         select distinct @v_labelid = labelId,
@v_factorid = factorid from factor where studyid =
@v_StudyId and fname like 'Location'
         -- Add Location
         insert level_c
values(@v_labelid,@v_factorid,@v_levelno,@v_loc
ation)
         select distinct @v_labelid = labelId,
@v_factorid =
factorid from factor where studyid = @v_StudyId and
fname like 'Location_no'
```

```
      values(@v_labelid, @v_factorId,
      'LOCATION',@v_studyId,215,94,215,'C')
      exec sp_GetNextDmsAttrId @v_dmsattr output
      insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Loca
tion')
      select @v_labelid = @v_labelId +1
      insert into factor
      values(@v_labelid, @v_factorId,
       'LOCATION_NO',@v_studyId,215,95,215,'N')
      exec sp_GetNextDmsAttrId @v_dmsattr output
      insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Loca
tion')
      select @v_labelid = @v_labelId +1
      insert into factor
      values(@v_labelid, @v_factorId,
      'CYCLE',@v_studyId,219,1536,1454,'C')
      exec sp_GetNextDmsAttrId @v_dmsattr output
      insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Cycl
e')
      select @v_labelid = @v_labelId + 1,
            @v_factorid = @v_labelid
      insert into factor
      values(@v_labelid, @v_factorId,
'PLOT',@v_studyid
      ,205,146,1484,'N')
      exec sp_GetNextDmsAttrId @v_dmsattr output
      insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Plot')
--    Add Effect
      insert into effect
      values(@v_represno,@v_factorid,@v_represno)
      select @v_labelid = @v_labelId +1
      insert into factor
      values(@v_labelid, @v_factorId,
```

```
      -- Add Location_no
      insert level_n
values(@v_labelid,@v_factorid,@v_levelno,@v_loc
ation_no)
      select distinct @v_labelid = labelId,
@v_factorid = factorid from factor where studyid =
@v_StudyId and fname like 'Cycle'
      -- Add Cycle
      insert level_c

values(@v_labelid,@v_factorid,@v_levelno,@v_cyc
le)
      select distinct @v_labelid = labelId,
@v_factorid = factorid from factor where studyid =
@v_StudyId and fname like 'Summary'
      -- Add Summary Value
      insert level_c
values(@v_labelid,@v_factorid,@v_levelno,rtrim('Su
mmary' + '_' + rtrim(convert(char(3),@v_occ))))
      if @v_flag_genotypes = 1
        begin
          select @v_flag_genotypes = 0
        end
      fetch next from cursor_occ into @v_occ
    end
  close cursor_occ
  deallocate cursor_occ
  select @v_represno
  -- Add row data
    update proceso
      set sp_add_oindex = getdate()
    where tid = @v_studyid
    exec sp_add_oindex @v_tid,@v_represno,0
    update proceso
      set sp_add_variate_rowdata = getdate()
    where tid = @v_studyid
    exec Sp_add_Variate_RowData_new
          @v_tid,@v_represno
  -- End Add Raw Data

  -- Add Environmental values
```

```
'BLOCK',@v_studyid
      ,202,151,1483,'N')
     exec sp_GetNextDmsAttrId @v_dmsattr output
     insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Bloc
k')
     select @v_labelid = @v_labelId +1
     insert into factor
     values(@v_labelid, @v_factorId, 'SUB BLOCK',
              @v_StudyId,202,144,1483,'N')
     exec sp_GetNextDmsAttrId @v_dmsattr output
     insert into dmsattr
     values(@v_dmsattr,801,'FACTOR',@v_labelid,
       'Sub Block')
     select @v_labelid = @v_labelId + 1,
           @v_factorid = @v_labelid
     insert into factor
     values(@v_labelid, @v_factorId, 'ENTRY
NUMBER',
               @v_StudyId ,251,103,17,'N')
     exec sp_GetNextDmsAttrId @v_dmsattr output
     insert into dmsattr
     values(@v_dmsattr,801,'FACTOR',@v_labelid,
              'Entry Number')
--    Add Effect
     insert into effect
     values(@v_represno,@v_factorid,@v_represno)
     select @v_labelid = @v_labelId + 1
     insert into factor
     values(@v_labelid, @v_factorId,
'CID',@v_StudyId
               ,251,1430,17,'N')
     exec sp_GetNextDmsAttrId @v_dmsattr output
     insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Cid')
     select @v_labelid = @v_labelId + 1
     insert into factor
     values(@v_labelid, @v_factorId,
'SID',@v_StudyId
```

```
       exec sp_GetLastRepresNo @v_represno
output
       insert into effect
       select distinct
@v_represno,factorid,@v_represno from factor
where studyid = @v_tid and fname in ('Occ', 'Study')
       update proceso
         set sp_add_variate_env = getdate()
       where tid = @v_studyid
       exec sp_add_varieties_env
@v_tid,@v_represno
    -- End Environmental values
    -- Add Summary data
       exec sp_GetLastRepresNo @v_represno
output
       insert into effect
       select distinct
@v_represno,factorid,@v_represno from factor
where studyid = @v_tid and fname in ('Occ',
'Study','ENTRY NUMBER','SUMMARY')
       update proceso
         set sp_add_oindex_summary = getdate()
       where tid = @v_studyid
       exec sp_add_oindex_summary
@v_tid,@v_represno  -- Add OINDEX Summary
values
       update proceso
         set sp_add_variate_summary = getdate()
       where tid = @v_studyid
       exec sp_add_varieties_summary
@v_tid,@v_represno   -- Add Summary values
    -- End Summary data
     update control
       set estatus = 'T'
     where tid = @v_tid
     update proceso
       set finish = getdate()
     where tid = @v_studyid
    fetch next from cursor_tid into @v_tid
  end
close cursor_tid
```

```
            ,251,1600,17,'N')
    exec sp_GetNextDmsAttrId @v_dmsattr output
    insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Sid')
    select @v_labelid = @v_labelId + 1
    insert into factor
    values(@v_labelid, @v_factorId,
'GID',@v_StudyId
            ,251,91,17,'N')
    exec sp_GetNextDmsAttrId @v_dmsattr output
    insert into dmsattr

values(@v_dmsattr,801,'FACTOR',@v_labelid,'Gid')
    select @v_labelid = @v_labelId + 1
```

```
deallocate cursor_tid
```

El cuadro 2 ilustra el Script incluido como proceso almacenado, de modo que cambios en la tabla GE-VALUES de IWIS se vea reflejado en CROPFINDER

Cuadro 2. Proceso almacenado para que cambios en la tabla GE-VALUES se reflejen en el sistema CROPFINDER

```
create proc Sp_add_Variate_RowData
     @p_tid integer, @v_represno integer
as
declare @v_trait_no integer, @v_value_type varchar(3), @v_unitname varchar(40), @v_variatid integer,
      @v_variatid_std integer, @v_occ integer, @v_ent integer, @v_value varchar(50),
      @v_value_std varchar(50), @v_trait_abbr varchar(6), @v_ounit_id integer,
      @v_unit_id integer, @v_factorid integer,@v_dmsattr int, @v_standard int

declare cursor_randomizations scroll cursor for
select distinct r.occ,consecutive_entry_number from wdms..randomizations r inner join wdms..crtl_ot c
on r.tid = c.tid and r.occ = c.occ where
r.tid = @p_tid and procesad = 'T'  order by r.occ,consecutive_entry_number

/* Add Variate and Veffect */
exec Sp_Create_Trait_Unit @p_tid

declare cursor_trait_unit scroll cursor for
select distinct trait_no,unit_id,standard,scale_type from trait_unit_paso
```

```
where trait_no in (select distinct trait_no from wdms..crtl_ot where tid = @p_tid and procesad = 'T' and class
= 'GE') order by trait_no,standard,unit_id

open cursor_trait_unit
fetch cursor_trait_unit into @v_trait_no, @v_unit_id,@v_standard,@v_value_type
while @@fetch_status=0
  begin
    select @v_unitname = scale_name from wdms..units where unit_id = @v_unit_id
    select @v_trait_abbr = trait_abbr from wdms..traits where trait_no = @v_trait_no
    if not exists(select * from variate where traitid = (@v_trait_no + 1000) and scaleid = (@v_unit_id + 2000)
and studyid = @p_tid)
      begin
        exec Sp_getLastvariate @v_variatid output
        -- Add Variate
        if @v_standard = 0
          insert into Variate
          values(@v_variatid,rtrim(rtrim(@v_trait_abbr) + '_' + rtrim(@v_unitname))
                  ,@p_tid,'MV',@v_trait_no + 1000,@v_unit_id + 2000,0,@v_value_type)
        else
          insert into Variate
          values(@v_variatid,rtrim(rtrim(@v_trait_abbr) + '_STD_' + rtrim(@v_unitname))
                  ,@p_tid,'MV',@v_trait_no + 1000,@v_unit_id + 2000,0,@v_value_type)
        -- Add Veffect
        insert into veffect
        values(@v_represno,@v_variatid)

        exec sp_GetNextDmsAttrId @v_dmsattr output
        insert into dmsattr
        values(@v_dmsattr,802,'VARIATE',@v_variatid,rtrim(@v_trait_abbr))
      end
    fetch next from cursor_trait_unit into @v_trait_no, @v_unit_id,@v_standard,@v_value_type
  end
close cursor_trait_unit
deallocate cursor_trait_unit
/* End variate and veffects */
declare cursor_trait scroll cursor for
select distinct trait_no,unit_id from trait_unit_paso
where trait_no in (select distinct trait_no from wdms..crtl_ot where tid = @p_tid and procesad = 'T' and class
= 'GE') and standard = 0 order by trait_no,unit_id

open cursor_trait
```

```
fetch cursor_trait into @v_trait_no,@v_unit_id
while (@@fetch_status = 0)
  begin
    if exists (select * from wdms..ge_values_r where tid = @p_tid and trait_no = @v_trait_no and
                unit_id = @v_unit_id)
      begin
        select @v_ounit_id = min(ounitid)  from factor f , oindex o  where studyid = @p_tid and fname like
                 'Occ' and f.factorid = o.factorid and represno = @v_represno
        open cursor_randomizations
        fetch cursor_randomizations into @v_occ,@v_ent
        while (@@fetch_status=0)
          begin
            select @v_value = null,@v_value_std = null,@v_variatid = null,@v_variatid_std = null
            if exists(select * from wdms..ge_values_r where tid = @p_tid and occ = @v_occ and trait_no =
                      @v_trait_no and unit_id = @v_unit_id and ent = @v_ent)
              begin
                select @v_value = convert(char,real_value),@v_value_std  = convert(char,standard_value)
                from wdms..ge_values_r where tid = @p_tid and occ = @v_occ
                        and trait_no = @v_trait_no and unit_id = @v_unit_id and ent = @v_ent
                select @v_variatid = variatid from variate where traitid = (@v_trait_no + 1000) and scaleid =
                        (@v_unit_id + 2000) and studyid = @p_tid
                select @v_variatid_std = variatid from variate where traitid = (@v_trait_no + 1000) and
                         scaleid = (select unit_id + 2000 from trait_unit_paso where trait_no = @v_trait_no
                         and  standard=1) and studyid = @p_tid
                /* insert raw data */
                insert into data_n
                values(@v_ounit_id,@v_variatid,convert(float,@v_value))
                /* insert standard data */
                insert into data_n
                values(@v_ounit_id,@v_variatid_std,convert(float,@v_value_std))
              end

            select @v_ounit_id = @v_ounit_id + 1
            fetch next from cursor_randomizations into @v_occ,@v_ent
          end
          close cursor_randomizations
          fetch next from cursor_units_r into @v_unit_id
      end

    if exists (select * from wdms..ge_values_n where tid = @p_tid and trait_no = @v_trait_no and
                unit_id = @v_unit_id)
```

```
begin
  select @v_ounit_id = min(ounitid)  from factor f , oindex o  where
       studyid = @p_tid and fname like 'Occ' and f.factorid = o.factorid and represno = @v_represno
  open cursor_randomizations
  fetch cursor_randomizations into @v_occ,@v_ent
  while (@@fetch_status=0)
    begin
      select @v_value = null, @v_value_std = null, @v_variatid = null, @v_variatid_std = null

      if exists(select * from wdms..ge_values_n where tid = @p_tid and occ = @v_occ and
              trait_no = @v_trait_no and unit_id = @v_unit_id and ent = @v_ent)
        begin
          select @v_value = convert(char,number_value),@v_value_std = rtrim(standard_value)  from
                wdms..ge_values_n where tid = @p_tid and occ = @v_occ
               and trait_no = @v_trait_no and unit_id = @v_unit_id and ent = @v_ent
          select @v_variatid = variatid from variate where traitid = (@v_trait_no + 1000) and scaleid =
               (@v_unit_id + 2000) and studyid = @p_tid
          select @v_variatid_std = variatid from variate where traitid = (@v_trait_no + 1000) and
                scaleid = (select unit_id + 2000 from trait_unit_paso where trait_no = @v_trait_no and
                          standard=1) and studyid = @p_tid
          /* Insert raw data */
          insert into data_n
          values(@v_ounit_id,@v_variatid,convert(float,@v_value))
          /* Insert standard data */
          if (select scale_type from trait_unit_paso where trait_no = @v_trait_no and standard=1) = 'N'
            begin
              if rtrim(@v_value_std) <> '-'
                insert into data_n
                values(@v_ounit_id,@v_variatid_std,convert(float,@v_value_std))
            end
          else
            insert into data_c
            values(@v_ounit_id,@v_variatid_std,rtrim(@v_value_std))
        end

      select @v_ounit_id = @v_ounit_id + 1
      fetch next from cursor_randomizations into @v_occ,@v_ent
    end
    close cursor_randomizations
    fetch next from cursor_units_r into @v_unit_id
end
```

```
    if exists (select * from wdms..ge_values_t where tid = @p_tid and trait_no = @v_trait_no and unit_id =
@v_unit_id)
       begin
         select @v_ounit_id = min(ounitid)  from factor f , oindex o  where
              studyid = @p_tid and fname like 'Occ' and
              f.factorid = o.factorid and represno = @v_represno
         open cursor_randomizations
         fetch cursor_randomizations into @v_occ,@v_ent

         while (@@fetch_status=0)
           begin
             select @v_value = null, @v_value_std = null, @v_variatid = null, @v_variatid_std = null

             if exists(select * from wdms..ge_values_t where tid = @p_tid and occ = @v_occ and
                       trait_no = @v_trait_no and unit_id = @v_unit_id and ent = @v_ent)
               begin
                 select @v_value = rtrim(text_value),@v_value_std = rtrim(standard_value)  from
                          wdms..ge_values_t where tid = @p_tid and occ = @v_occ
                        and trait_no = @v_trait_no and unit_id = @v_unit_id and ent = @v_ent
                 select @v_variatid = variatid from variate where traitid = (@v_trait_no + 1000) and
                          scaleid = (@v_unit_id + 2000) and studyid = @p_tid
                 select @v_variatid_std = variatid from variate where traitid = (@v_trait_no + 1000) and
                          scaleid = (select unit_id + 2000 from trait_unit_paso where trait_no = @v_trait_no and
                                    standard=1) and studyid = @p_tid
                 /* Insert raw data */
                 insert into data_c
                 values(@v_ounit_id,@v_variatid,rtrim(@v_value))
                 /* Insert standard data */
                 if (select scale_type from trait_unit_paso where trait_no = @v_trait_no and standard=1) = 'N'
                   begin
                     if rtrim(@v_value_std) <> '-'
                       insert into data_n
                       values(@v_ounit_id,@v_variatid_std,convert(float,@v_value_std))
                   end
                 else
                   insert into data_c
                   values(@v_ounit_id,@v_variatid_std,rtrim(@v_value_std))
               end
             select @v_ounit_id = @v_ounit_id + 1
             fetch next from cursor_randomizations into @v_occ,@v_ent
           end
```

```
        close cursor_randomizations
        fetch next from cursor_units_r into @v_unit_id
    end
  if exists (select * from wdms..ge_values_d where tid = @p_tid and trait_no = @v_trait_no and
          unit_id = @v_unit_id)
    begin
      select @v_ounit_id = min(ounitid)  from factor f , oindex o  where
          studyid = @p_tid and fname like 'Occ' and
          f.factorid = o.factorid and represno = @v_represno
      open cursor_randomizations
      fetch cursor_randomizations into @v_occ,@v_ent
      while (@@fetch_status=0)
        begin
          select @v_value = null, @v_value_std = null, @v_variatid = null, @v_variatid_std = null
          if exists(select * from wdms..ge_values_d where tid = @p_tid and occ = @v_occ and
                  trait_no = @v_trait_no and unit_id = @v_unit_id and ent = @v_ent)
            begin
              select @v_value = rtrim(date_value),@v_value_std = convert(char,standard_value)  from
                   wdms..ge_values_d where tid = @p_tid and occ = @v_occ
                  and trait_no = @v_trait_no and unit_id = @v_unit_id and ent = @v_ent
              select @v_variatid = variatid from variate where traitid = (@v_trait_no + 1000) and
                   scaleid = (@v_unit_id + 2000) and studyid = @p_tid
              select @v_variatid_std = variatid from variate where traitid = (@v_trait_no + 1000) and
                   scaleid = (select unit_id + 2000 from trait_unit_paso where trait_no = @v_trait_no and
                        standard=1) and studyid = @p_tid
              /* Insert raw data */
              insert into data_c
              values(@v_ounit_id,@v_variatid,rtrim(@v_value))
              /* Insert standard data */
              if (select scale_type from trait_unit_paso where trait_no = @v_trait_no and standard=1) = 'N'
                insert into data_n
                values(@v_ounit_id,@v_variatid_std,convert(float,@v_value_std))
              else
                insert into data_c
                values(@v_ounit_id,@v_variatid_std,rtrim(@v_value_std))
            end
          select @v_ounit_id = @v_ounit_id + 1
          fetch next from cursor_randomizations into @v_occ,@v_ent
        end
        close cursor_randomizations
        fetch next from cursor_units_r into @v_unit_id
```

```
        end
        fetch next from cursor_trait into @v_trait_no
    end
close cursor_trait
deallocate cursor_trait
deallocate cursor_randomizations
```

Como parte fundamental del nuevo sistema, se definieron, elaboraron e instalaron estos procedimientos en las bases de datos de IWIS, se condujeron las pruebas necesarias para garantizar la exactitud de la información que tendría el nuevo sistema, y finalmente, se instalaron de modo que la información, que siempre estaba al día ya en IWIS, continúe teniendo este atributo en CROPFINDER:

2.4.2 Proceso de creación de la base de CROPFINDER a partir de la base de datos MaizeFinder

En este sistema, el proceso no puede ser automático. Se parte de una versión actualizada de la base de datos Maizefinder, probablemente cada vez que se actualice dicha base de datos, suceso que se debe a la llegada de información al sistema Maizeimport. El proceso que resulta es la actualización de la base de datos de Maizeimport, y una comparación con Maizefinder, que produce una actualización de esta última cuando se detectan diferencias (datos nuevos.) La base de datos Maizefinder del sistema en producción usa la tecnología ACCESS.

Para actualizar la base de datos CROPFINDER con datos de Maíz, primero se crea una versión de la base Maizefinder en SQL SERVER, puesto que eso facilita la actualización posterior de CROPFINDER. La base nueva se crea cada vez que llegan datos nuevos. De este modo, cuando se construye la base de datos Maizefinder en SQL SERVER, con los datos de la correspondiente en ACCESS; se ejecutarán procesos catalogados que reflejan las actualizaciones en la base CROPFINDER. La naturaleza de estos procesos, otra vez escritos en SQL Script, es similar a las que se usaron para ilustrar los procesos que crean la base de datos CROPFINDER para Trigo, aunque, naturalmente, son diferentes puesto que usan tablas distintas.

2.5 Perfil de los usuarios del sistema

Para que un usuario pueda usar el servicio de consultas, es decir, el sistema CROPFINDER, debe conectarse a dicho sistema, lo que se conoce como “login”. Le pide una identificación y una palabra clave, para efectos de autenticación. En muchos sistemas estos requisitos son para limitar el acceso al sistema, pero en el sistema que nos ocupa el motivo principal es poder ofrecer un servicio personalizado. En particular, a un usuario se le ofrecen *sus* reportes, es decir, los que él mismo construyó.

Es importante señalar que cualquier usuario – registrado – puede construir o ejecutar un informe. Sin embargo, en un futuro pudiera haber una solicitud de cambio al sistema, en el sentido de que sólo algunos usuarios pudieran realizar ciertas actividades.

CAPÍTULO 3. EL SERVICIO DE CONSULTAS EN INTERNET

3.1 Introducción

El sistema se ofrece por Internet, de modo que los usuarios trabajan con "páginas". Se muestra primero el armado de un reporten nuevo, que consiste en la especifiación de las columnas que contendrá, donde estas columnas se seleccionan de una lista de datos disponibles.

A continuación se muestra una corrida de una consulta, es decir, la invocación de un reporte previamente definido. Se verá que en ese paso, el usuario puede indicar filtros que harán que sólo aparezcan datos de los estudios que cumplan las condiciones especificadas. Por ejemplo, si le interesa conocer qué investiagciones se han realizado para una cierta variedad, puede indicar ésta como criterio de modo que no aparecerán en su informe, datos sobre estudios que no versaron sobre dicha variedad.

3.2 Construcción de una consulta o informe

Para armar un reporte, primero se invoca una página (de Internet) como la que se muestra en la Figura 4. Las leyendas aparecen en idioma inglés puesto que es el idioma oficial del Cimmyt. En ella, se muestran los campos que componen la base de datos utilizando el objeto "TreeView", del conjunto de objetos IEWebControls (Dhabi, 2003) a partir de una consulta a la tabla FieldSetup. Los campos se agrupan de acuerdo a la tabla en la que se encuentra la información correspondiente. De esta forma, los grupos "Factors" y "Study" están compuestos por los campos de esas tablas, mientras que el grupo "Traits" lo forman los campos definidos en la tabla "Variate", los que a su vez hacen referencia a las tablas Data_N y Data_C .

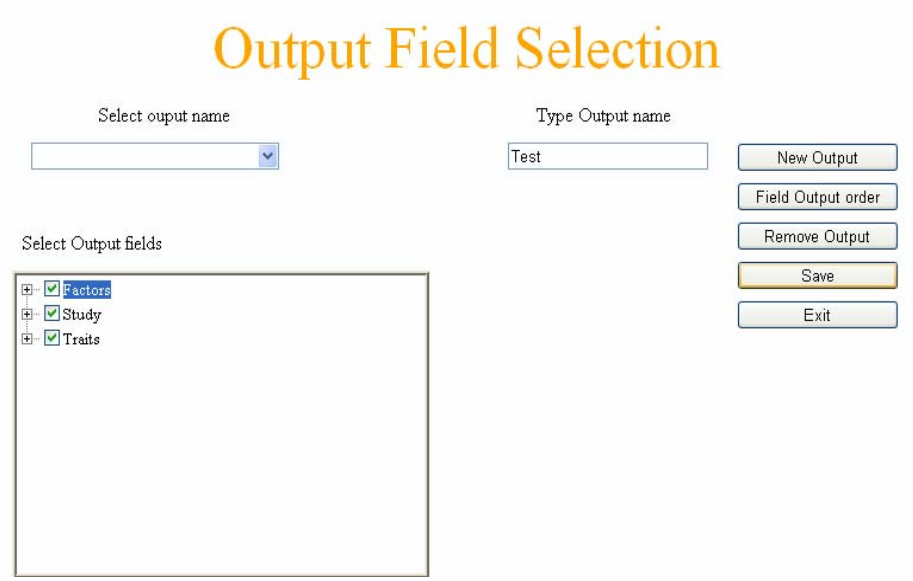

Figura 7. Los grupos de datos para su inclusión en el reporte

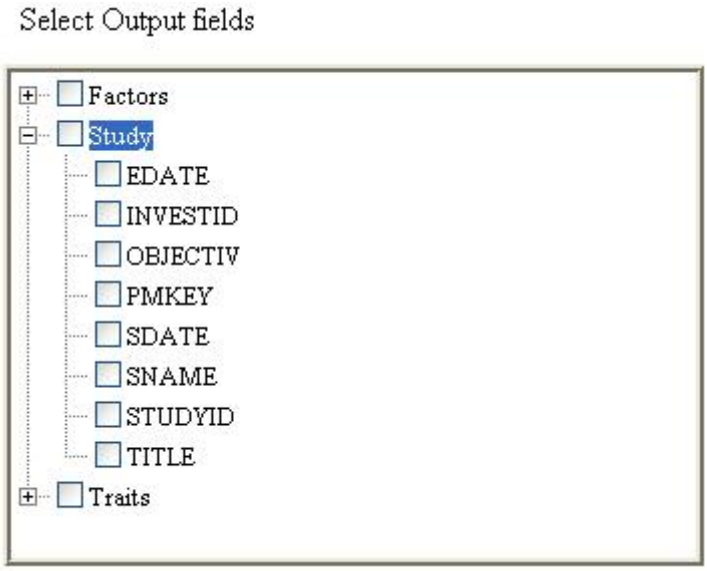

Figura 8. Detalle de campos de la tabla Study.

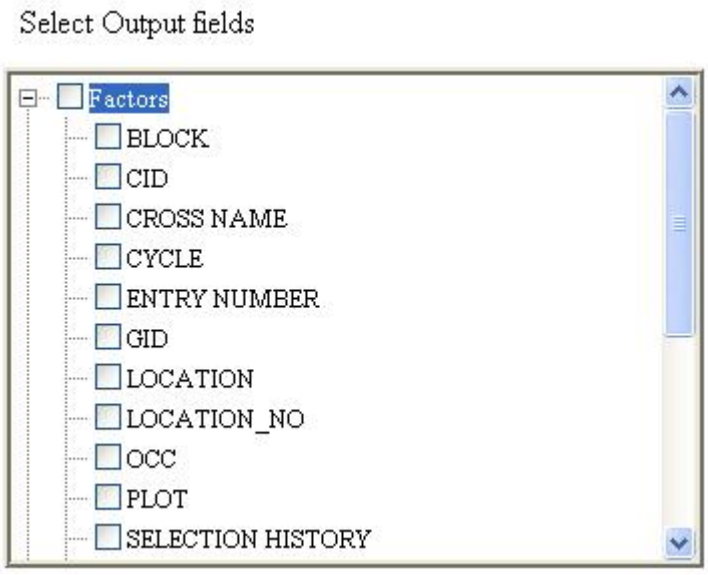

Figura 9. Detalle de campos de la tabla Factors

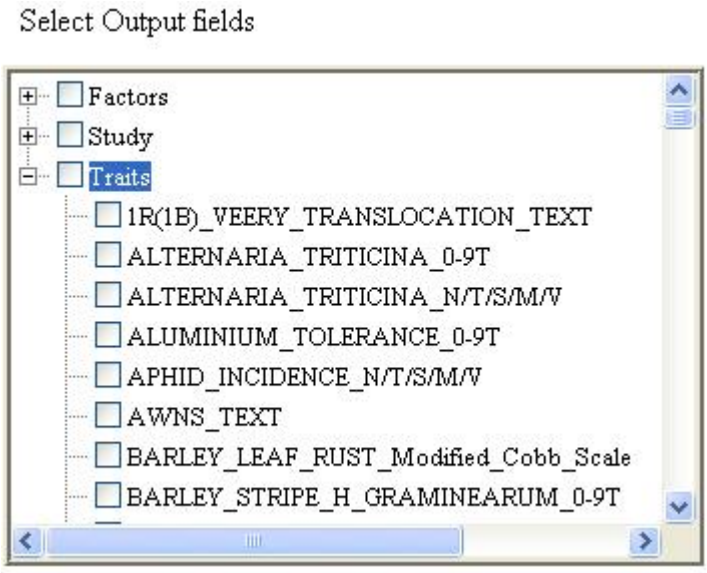

Figura 10. Algunos de los campos de la tabla Traits

El usuario deberá efectuar las siguientes acciones.

1. Seleccionar el botón “New Output”.
2. Teclear en el campo “Type Output Name” el nombre del reporte nuevo.

3. En el objeto “TreeView” (etiquetado Select Output fields) seleccionar los campos que formarán el reporte activando los “CheckBoxes” correspondientes. Para ello, invoca los campos de un grupo (con un click en el grupo), aparecen los campos que lo forman y selecciona los que desea para su informe.

4. Se usa el botón “Save” para guardar el reporte. Cuando el usuario - con un clic del ratón – usa este botón, la aplicación crea un registro en la tabla “OutputReport” con el nombre del reporte y le asigna un identificador. Posteriormente para cada objeto seleccionado del “TreeView”, almacena en la tabla “OutputField” los identificadores de los campos correspondientes a los “CheckBox” seleccionados (palomeados), incluyendo la identificación del reporte que se está definiendo para establecer las ligas necesarias.

3.3 Ejecución de un Reporte

Para obtener un reporte, mediante la ejecución del programa correspondiente, se selecciona uno de los reportes disponibles, mismos que aparecen en una lista con sus respectivos nombres. A continuación aparece una forma como la que se reproduce en la Figura 5, en la que pueden indicar los criterios de selección de registros – es decir, las hileras del informe producido. Estos criterios también se conocen como *filtros*. Siguiendo la terminología utilizada en la institución, se ha denominado *Advanced Query* este paso de selección de registros.

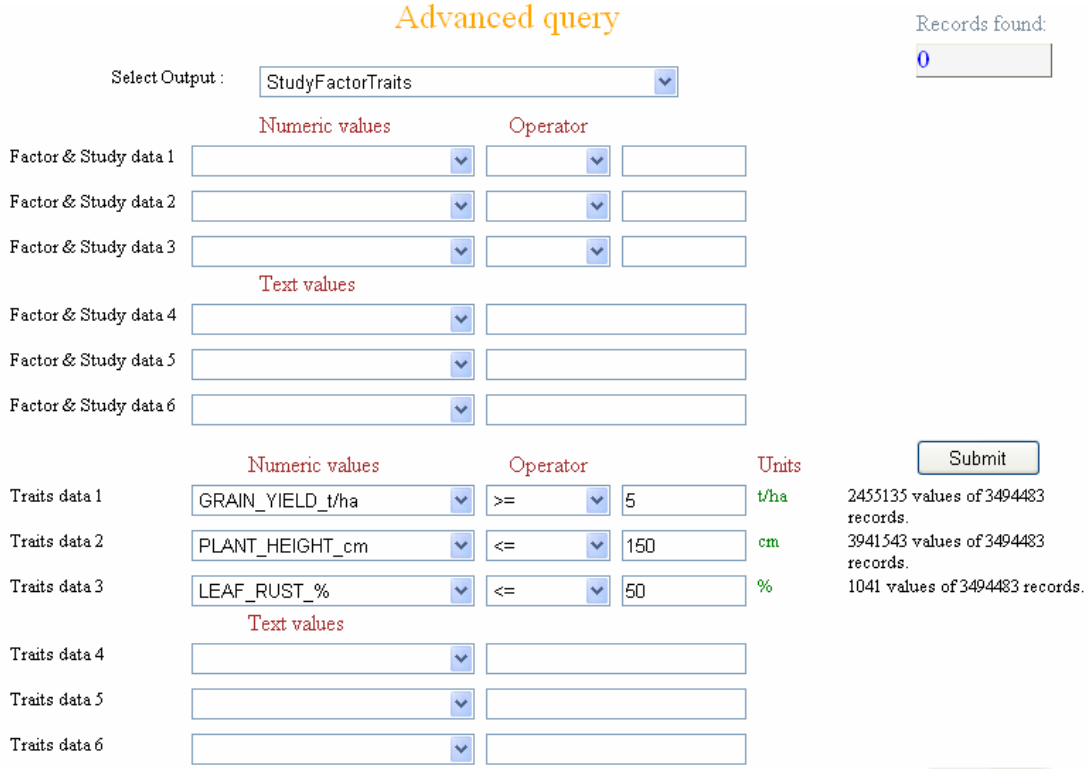

Figura 11. Pantalla para la ejecución de un reporte

Los pasos para la ejecución de un reporte son:

1. Ofrecer los reportes disponibles en una lista para que el usuario seleccione el que necesite o desee.
2. Indicar las condiciones o restricciones para la selección de los registros (hileras) que se incluirán en el reporte. Para ello se selecciona un concepto de la lista de alternativas disponibles (>, <, =,>=, <=). En la Figura 5, la primera condición impuesta a los registros en relación a una de las características fue indicada seleccionando el campo GRAIN-YIELD_t/ha, y a continuación, de la lista de operadores, se eligió el >=. Finalmente, se tecleó el valor 5. Esto hará que el informe no contenga datos de ningún estudio para el no se satisfaga dicha condición. Si el usuario indica más de un criterio de selección, el programa los interpretará como conjunciones de las operaciones correspondientes (en otras palabras, es condición-1 **Y** condición-2, etc.)

3. Una vez seleccionadas las restricciones, se oprime el botón “Submit”. El programa arma la consulta SQL, con el proceso que se describe en la siguiente sección.

3.4 El resultado de la consulta

La aplicación muestra el numero de registros (renglones) encontrados que cumplen con las restricciones definidas, dando la opción al usuario de poder verlos si éste lo considera pertinente. Esto se hace porque si el número es excesivo, puede ser que agregue algún filtro adicional para disminuir este número. Se recuerda al lector que hay muchos cientos de miles de datos, y una especificación inadecuada podría resultar en una consulta que no es de utilidad por contener demasiados datos.

Internamente, el resultado de la ejecución de la sentencia SQL es un Recordset, al que se le asocia un control “DataGrid” (MacDonal 2002) para mostrar la información. La Figura 12 muestra el resultado que aparece en el monitor del usuario.

Download

SNAME	LEAF_RUST_%	GRAIN_YIELD_t/ha	SELECTION HISTORY	CID	TID	CROSS NAME
04 IAT	10	7.24988412857056	CM92313-19Y-0H-0SY-5M-0RES-0HUA	20569	16450	CETTIA
04 IAT	50	6.08323621749878	-0AUS	149339	16450	EXCALIBUR
12 HTWYT	20	5.7166428565979	CMSS94Y02702T-030Y-0300M-0100Y-0100M-3Y-6M-0Y-0HTY	158634	17315	MUNIA/CHTO/3/PFAU/BOW//VEE#9/4/CHEN/AEGILOPS SQUARROSA (TAUS)//BCN
12 HTWYT	25	5.28331184387207	FPSS95B00315S-040Y-020M-040Y-020Y-7M-0Y-0HTY	281884	17315	BCN/3/ALD/PVN//YMI#6
12 HTWYT	30	5.68331098556519	CMSS92M02859T-015M-0Y-0Y-050M-12Y-1M-0Y-1KBY-0KBY-0M-0HTY	101469	17315	HD2136/SKA/5/TOB/CNO67//BB/4/NAI60*2//TT/SN64/3/LR64A/SN64/6/HD2257/7/SPB/8/I
12 HTWYT	50	5.99997615814209	CMSS92M02859T-015M-0Y-0Y-050M-12Y-1M-0Y-1KBY-0KBY-0M-0HTY	101469	17315	HD2136/SKA/5/TOB/CNO67//BB/4/NAI60*2//TT/SN64/3/LR64A/SN64/6/HD2257/7/SPB/8/I
12 SAWYT	0	7.91663503646851	CMSS93Y02628S-16Y-010SY-010Y-015SY-7Y-05B-0Y-0SY	120770	17310	DHARWAR DRY/NESSER

Figura 12. Resultados de una consulta

3.5 Impresión o generación de un archivo con la consulta

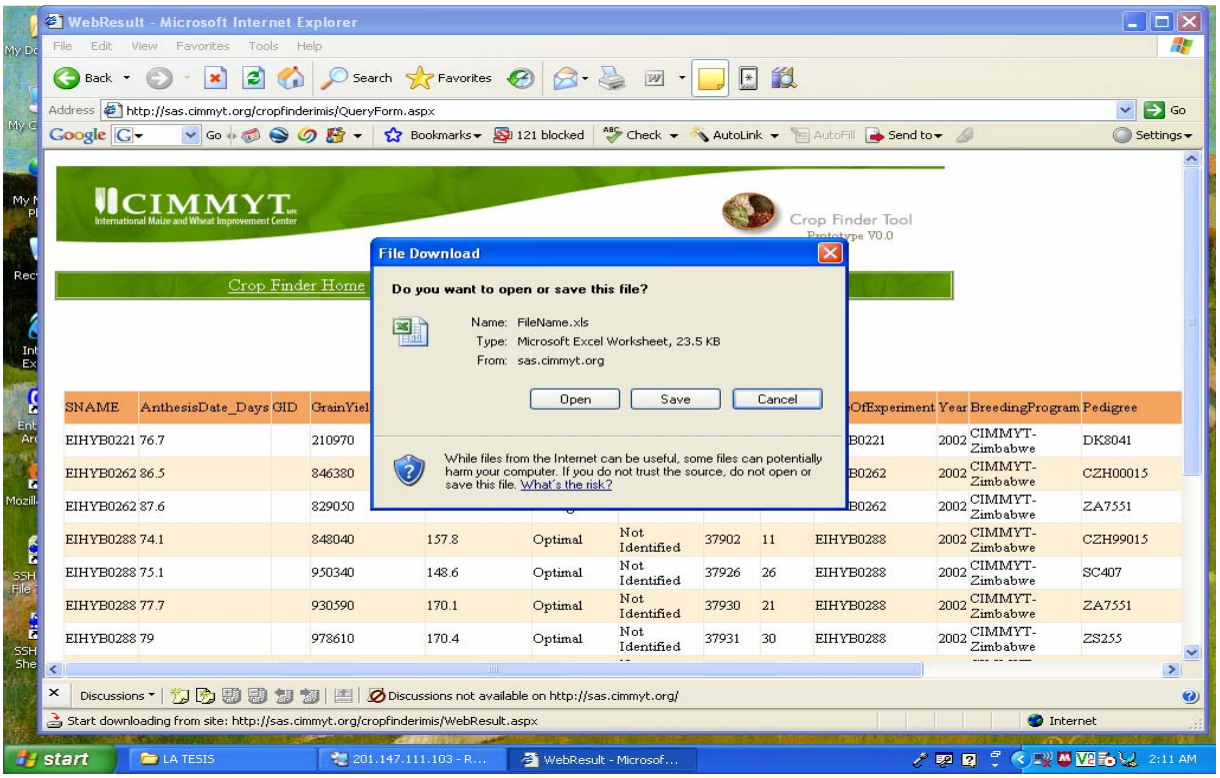

Figura 13. Diálogo para guardar el resultado de la consulta

Los resultados se pueden guardar en un archivo de formato “Excel” seleccionando el botón “Download”, en una forma como la que ilustra la Figura 13. Cuando se invoque esta función, el sistema solicitará al usuario, con la forma de la Figura 14, proveer el nombre y la ubicación del archivo generado,

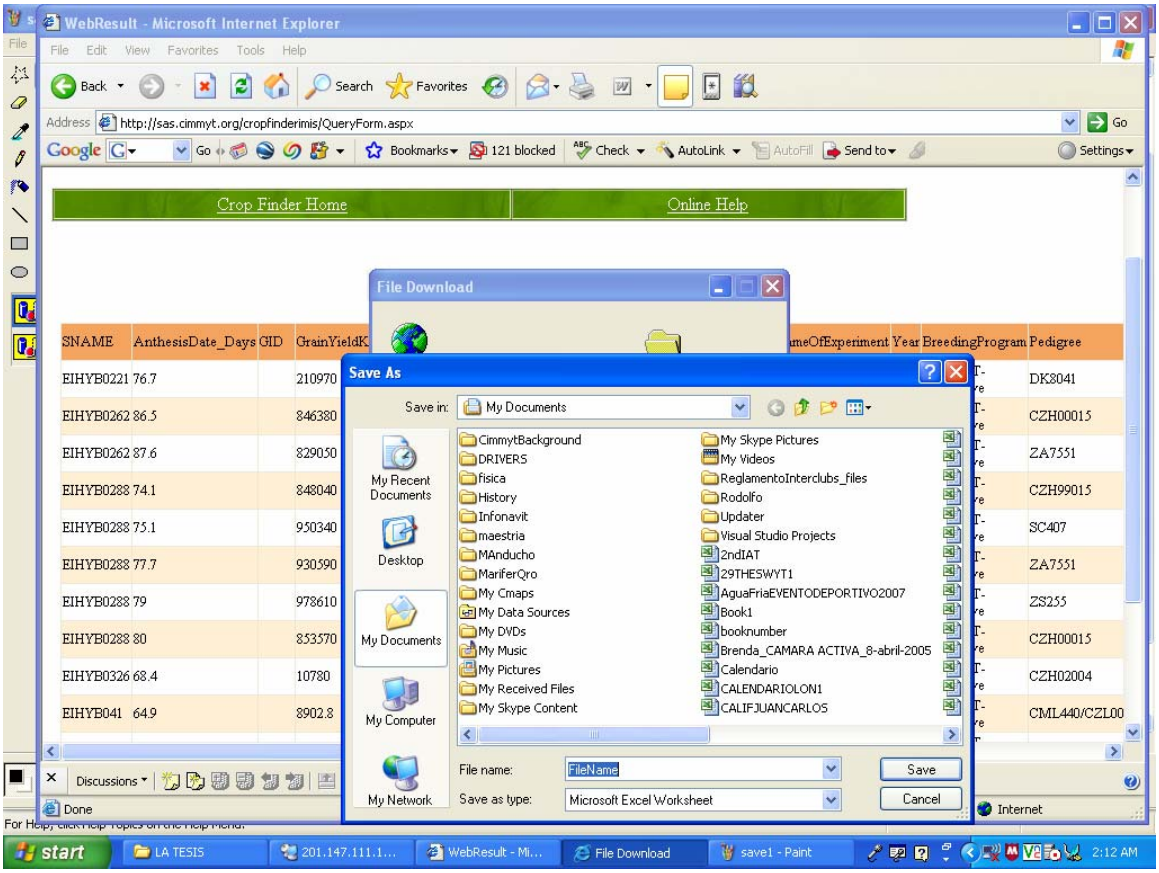

Figura 14. Diálogo para indicar el nombre y ubicación del archivo

CAPÍTULO 4. COMPONENTES TÉCNICAS DEL SERVICIO

4.1 Introducción

Para ofrecer los servicios de CROPFINDER a sus usuarios, se pueden aislar las siguientes componentes técnicas.

- La base de datos en la que se guardarán las consultas formuladas por los usuarios, misma que se describió en el Capítulo 2.
- La programación de la página de construcción de un reporte
- La programación de la página en la que el usuario selecciona el reporte que desea.
- La programación y los elementos que permiten la especificacíon de los criterios de selección de registros que integrarán el reporte.
- El proceso necesario para ejecutar una consulta, a partir de la definición y los criterios de selección.
- El uso de los objetos para mostrar el resultado.

4.2 La programación de la página de construcción de un reporte

El programa prepara una página en la que el componente principal es una lista de los reportes definidos, además de la inclusión de campos para la identificación del usuario. En esta misma página, el usuario puede indicar que desea un reporte nuevo, para lo cual indica el nombre del mismo en un campo de texto. El programa crea un registro nuevo en la tabla OutputReport con el nombre del reporte y un identificador que asigna, usando para ello una numeración consecutiva de los reportes definidos.

A continuación utiliza un objeto *treeview,* en el que pone los grupos de campos, que son Study, Factors y Traits – mismos que corresponden a las tres tablas de la base. El usuario puede seleccionar cualquiera de ellos una y otra vez, cosa que resulta en una lista de todos los campos del grupo seleccionado. Internamente, se asocia el objeto treeview a la tabla FieldSetup. Cuando el usuario termina de seleccionar los campos, el programa crea, para aquéllos marcados para su inclusión en el reporte, un registro en la tabla OutputField. Los campos de esta tabla son: el identificador del reporte y el identificador interno del campo seleccionado.

4.3 Programación de la página en la que el usuario solicita un reporte

El usuario selecciona el reporte de una lista que aparece en la página de inicio de la aplicación. Esta lista está asociada a la tabla OutputReport de los reportes definidos anteriormente. Se prepara la ejecución del reporte, que consiste en los criterios de selección y la consecuente consulta mostrada en un datagrid.

4.4 La programación de la página en la que un usuario especifica los criterios de selección

El programa crea un objeto treeview con los todos los campos, para que se indiquen los criterios de selección. Una vez más, el usuario selecciona un grupo de campos, tras lo cual el sistema muestra los campos de ese grupo. La selección de un campo hace que aparezcan dos objetos. Uno tipo combo, en el que se elige el operador de la lista (=, >, <, >=, <=). En el otro objeto, un campo de texto, se indica el valor aplicable a dicho criterio o filtro. El programa almacena estos criterios en memoria.

Cuando se indica que se desea el reporte con los criterios indicados, es decir, ya no hay otros filtros, el programa construye una sentencia SQl que se muestra en la sección 4.5, y tras ejecutar el comando, muestra el número de registros que devolvió la consulta. Si el usuario indica que desea agregar filtros, se repite el paso anterior. De lo contrario, se muestra la consulta en un objeto datagrid.

4.5 El proceso necesario para ejecutar una consulta

Cuando el programa recibe la confirmación de los criterios de selección, procede a crear un comando SQL. Se muestra cómo se construye la sentencia SQL para incorporar los criterios de selección, además de lo referente al reporte mismo, es decir, las columnas involucradas.

La construcción de la sentencia SQL se realiza en 4 etapas. En cada etapa se obtiene una cadena de caracteres. Al final se concatenan para formar la sentencia SQL final. SQLFinal = strSelect + strFrom + strJoin + strWhere

4.5.1 Construcción de la cadena strSelect

En la primera etapa, se crea la cadena de caracteres strSelect, basada en la identificación del reporte seleccionado por el usuario. Se crea una consulta (mostrada en el Cuadro 1), en la que se incluirán los campos que forman parte del reporte, que se ilustran en el Cuadro 1.

Cuadro 3. Consulta para determinar los campos del reporte

Select fieldname,fieldtext,tablename,tablealias,traitid,unitid from fieldsetup where fieldid in (select distinct fieldid from outputfield where outputid = '246BAA32-E123-4E12-B039-43699AF175F7')

4.5.2 Construcción de la cadena strFrom

Se puede apreciar que las tablas involucradas en esta consulta son “FieldSetup”, que contiene la información de los campos que componen la base de datos, y la tabla “OutputField”, con los identificadores de los campos que forman parte de un reporte. El cuadro 4 refleja el resultado de la consulta usada como ejemplo.

Cuadro 4. Resultado de la consulta a las columnas del reporte invocado

Field Name	Field Text	Table Name	Table Alias	TraitId	UnitId
Dvalue	GRAIN_YIELD_t/ha	data_n	data_n_GRAIN_YIELD_t/ha	1144	2065
Dvalue	PLANT_HEIGHT_cm	data_n	data_n_PLANT_HEIGHT_cm	1151	2019
Dvalue	LEAF_RUST_%	data_n	data_n_LEAF_RUST_%	1073	2017
TID	TID	Factors	Factors	Null	Null
CROSS NAME	CROSS NAME	Factors	Factors	Null	Null
CID	CID	Factors	Factors	Null	Null
SEL HISTORY	SELECTION HISTORY	Factors	Factors	Null	Null
SID	SID	Factors	Factors	Null	Null
SNAME	SNAME	Study	Study	Null	Null

A partir de este resultado se crean las colecciones de nombres de tablas (tablename), nombres alternos de tablas (tablealias) y la colección de campos de salida (output). La colección “tablename” contiene los nombres de todas las tablas

que figuran en el resultado de la consulta. En el caso del ejemplo, estará formada por los nombres *data_n, Factors y Study*. Los nombres alternos de las tablas están en la colección TableAlias; para nuestro caso, será *data_n_GRAIN_YIELD_t/ha, data_n_PLANT_HEIGHT_cm, data_n_LEAF_RUST_%, Factors y Study*. Por ultimo, "Output" es la colección de todos los registros del resultado de la consulta.

La cadena de caracteres strSelect se forma concatenando cada uno de los campos que integran el reporte a partir de la colección "Output". *For each Fieldname in Output Collection .. strSelect += [tablealias].fieldname as [fieldtext]*

4.5.3 Construcción de la cadena strFrom y strJoin

En la segunda etapa, se crean las cadenas de caracteres strFrom y strJoin. El primer paso en esta etapa es el análisis de la colección "TableName" creada en la primera etapa, con el propósito de identificar algún posible "Join" (reunión) que pueda requerir la sentencia final. Supongamos que nuestra colección "TableName" está compuesta por las tablas *Data_n y Study.* Estas tablas sólo se pueden relacionar utilizando la tabla "Factors" como intermedia, de modo que se tiene que agregar un "Join" a la sentencia SQL que no estaba implícita en las tablas que forman la colección "TableName". El cuadro 5 muestra las sentencias involucradas.

Cuadro 5. Sentencias que determinan las uniones necesarias

If Study table and Data(C or N) tables in Collection tablename but not Factors table then Add join name to join collection (Study-Data) .
If Study table and Factors table in Collection tablename but not Data(C or N) tables then Add join name to join collection (Study-Factors)
If Study table , Factors table and Data(C or N) table in Collection tablename then Add join name to join collection (Study-Factors-Data)

Una vez creada la colección "Join" se procede a formar las cadenas de caracteres strFrom y StrJoin. La cadena strFrom se crea con el primer evento al recorrer la colección "TableAlias", siempre y cuando en esta colección figuren nombres de tablas diferentes a "Study" y "Factors". Del mismo modo, la cadena strJoin se crea cuando hay más de un nombre de tabla diferente a "Study" y

“Factors”. En el Cuadro 6 se muesta la sentencia que recorre la colección “TableAlias”.

Cuadro 6. Sentencias SQL que producen las cadenas strFrom y StrJoin

For each tableAlias in Collection Table Alias. *if tablename not Factors or Study then* *if strFrom = “” then* ***strFrom** = “ From “ & tablealias* *strTablealiasFrom = Tablealias* *else **strjoin** += inner join tablename* *as tablealias on tablealias.oindex = strTablealiasfrom.oindex()* ***strjoin** += inner join tablename as tablealias on tablealias.oindex = strTablealiasfrom.oindex*	*if join_name = “Study-Data” then* ***strjoin**+= inner join Factors [Factors] on [strTablealiasFrom].ounitid = [Factors].ounitid inner join Study [Study] on [Study].studyId = [Factors].StudyId if join_name = “Study-Factors” then* ***strFrom** = “from [Factors] [Factors] “* ***strjoin** = “inner join Study [Study] on [Study].studyId = [Factors].StudyId”* *if join_name = “Study-Factors-Data” then* ***strjoin**+=inner join Factors [Factors] on [strtablealiasfrom].ounitid = [Factors].ounitid* *inner join Study [Study] on [Study].studyId = [Factors].StudyId*

4.5.4 Construcción de la cadena strWhere

En la tercera etapa se crea la cadena de caracteres strWhere, que refleja las condiciones seleccionadas antes de oprimir el botón “Submit” en la invocación del reporte. La aplicación recorre los objetos que definen las restricciones del reporte y añade la información de las condiciones a la colección “Collection”. De esta forma, la cadena de caracteres strWhere se crea a con la sentencia SQL: *For each QueryCondition in query Collection .. strWhere += [tablealias].[fieldtext] + operator + value*

4.5.5 Construcción de la cadena SQLfinal

Como se anunció previamente, se concatenan las cuatro cadenas obtenidas para obtener el resultado, es decir, la sentencia que se ejecutará para mostrar la consulta. De ese modo, SQLFinal = strSelect + strFrom + strJoin + strWhere. En el caso del ejemplo, se obtiene la sentencia SQL contenida en el Cuadro 7.

Cuadro 7. La sentencia con la que se realiza la consulta

```
Select distinct
[data_n_GRAIN_YIELD_t/ha].dvalue as
GRAIN_YIELD_t/ha],
[data_n_PLANT_HEIGHT_cm].dvalue as
[PLANT_HEIGHT_cm],
[data_n_LEAF_RUST_%].dvalue as
[LEAF_RUST_%], [factors].[TID] as [TID],
            [factors].[CROSS NAME] as
[CROSS NAME],  [factors].[GID] as [GID],
            [factors].[SELECTION HISTORY]
as [SELECTION HISTORY],
[study].[SNAME] as [SNAME]
   from data_n  [data_n_GRAIN_YIELD_t/ha]
   inner join data_n
[data_n_PLANT_HEIGHT_cm] on
[data_n_PLANT_HEIGHT_cm].ounitid =
[data_n_GRAIN_YIELD_t/ha].ounitid  and
data_n_PLANT_HEIGHT_cm].variatid in
    ( select distinct variatid from variate where
traitid = 1151 and scaleid =2019)
   inner join data_n [data_n_LEAF_RUST_%]
on
      [data_n_LEAF_RUST_%].ounitid =
      [data_n_GRAIN_YIELD_t/ha].ounitid
      and [data_n_LEAF_RUST_%].variatid in
      (select distinct variatid from variate where
          traitid = 1073 and scaleid = 2017)
   inner join factors [factors] on
[factors].ounitid =

[data_n_PLANT_HEIGHT_cm].ounitid
   inner join study [study] on [study].studyid =
[factors].studyid
   where [data_n_GRAIN_YIELD_t/ha].dvalue
>= 5 and [data_n_PLANT_HEIGHT_cm].dvalue
<= 150 and  [data_n_LEAF_RUST_%].dvalue
<= 50 and
        [data_n_GRAIN_YIELD_t/ha].variatid in
   (select distinct variatid from variate where
traitid = 1144 and scaleid = 2065)
```

4.6 Paso de los objetos para mostrar el resultado

El programa utiliza un objeto datagrid, al que asocia el recordset resultante de la consulta que construyó el programa con la sentencia SQL descrita.

Antes de mostrar el resultado, muestra al usuario el *recordcount* que le informa cuántos renglones tendrá su consulta, para permitirle agregar algún criterio de

selección adicional si fueran demasiados para resultar de utilidad. Cuando el usuario indica que desea ver el reporte, se muestra el objeto datagrid.

4.7 El proceso para guardar un reporte en un archivo de Excel

Como se describió en la sección 3.5, el usuario puede indicar que desea guardar el resultado de su consulta en un archivo EXCEL. Aparece un objeto dialogbox en el cual selecciona un directorio – o agrega uno nuevo – y a continuación se indica el nombre con el que desea almacenar los resultados. Esto a su vez le permitirá, con las herramientas de EXCEL, imprimir el cuadro en su totalidad, o seleccionar la parte del mismo que desea imprimir.

CAPÍTULO 5. EL FUTURO DEL SISTEMA CROPFINDER

5.1 El servicio que brinda

El sistema tuvo una aceptación inmediata, con comentarios muy favorables, puesto que ofrece a los investigadores dos aspectos novedosos y, por supuesto, muy solicitados anteriormente. De lejos el éxito mayor se debió a que la consecución de la información se realiza directamente solicitándola al que la tiene – el sistema, en lugar de tener que contar con la intervencion de personas que enviaran los datos al solicitante. El hecho de ofrecer los datos via Internet había sido una ambición de los investigadores, de modo que se convirtió en una prioridad en cuanto al plan de desarrollo de sistemas de CIMMYT.

El otro aspecto novedoso, que tuvo mucha aceptación, fue la posibilidad que se ofrece a un usuario del CROPFINDER de definir su propio “cuadro”, indicando cuáles de los datos le interesan en ese momento, pudiendo establecer criterios de selección que evitarán la inclusión de datos que no le sirvieran para el propósito de ese reporte.

Sin embargo, la necesidad de definir los criterios de selección (filtros) cada vez que se ejecuta un informe, no sólo resultó en algunas reclamaciones en ese sentido, sino que en ocasiones afectó el uso: hubo quienes dejaron de usar las consultas por ese motivo. Este tipo de situación se presenta con frecuencia en sistemas nuevos: los usuarios inicialmente están encantados por el acceso a los datos en tiempos muchísmimos más breves que anteriormente, pero de inmediato quieren “más”. Un servicio que acorta un periodo de dos semanas a dos minutos es criticado porque los usuarios afirman que “tarda mucho”, puesto que se les ocurre un modo de acortarlo aun más.

5.2 Algunos comentarios sobre las interfases

Las interfases de usuario son ágiles, en cuanto a las posibilidades y el fin para el cual fueron diseñadas. Como el sistema reacciona en forma casi instantánea a cualquier comando que se ejecuta en ellas – naturalmente contemplando algunas demoras causadas por la disponibilidad de los recursos informáticos que emplea, como es el proceso en el servidor y la transmisión de datos – se puede afirmar que

son idóneas: los usuarios las entienden y aprenden a usarlas con poco esfuerzo y sin lugar a confusiones. En la siguiente sección se comentará el único aspecto criticable de la versión que se adoptó, misma que estuvo motivada por una urgencia en el servicio solicitado, lo que a su vez implicó juntar dos aplicaciones en una misma interfase (la creación y la ejecución de los reportes.)

5.3 Cambios sugeridos

El sistema cumple su misión: permite a sus usuarios, aun los no locales, obtener la información que necesitan. Sin embargo, no sólo se formularon, como parte del proyecto, algunas sugerencias de mejoras, sino que algunas de éstas coincidieron con críticas acerca del modo de definir y usar los informes. Se mencionan sólo cuatro aspectos por la importancia que tienen en la naturaleza del servicio.

La necesidad de almacenar no sólo las columnas incluidas en el reporte, sino también los criterios de selección que algún usuario utilizará con cierta frecuencia, se considera que es el aspecto más importante de las mejoras que se tienen que incorporar cuanto antes. De hecho, se diseñaron dos tablas adicionales para guardar estos criterios, y se formularon primeras versiones de los procesos necesarios para guardar los criterios e invocarlos cuando el usuario los necesitara.

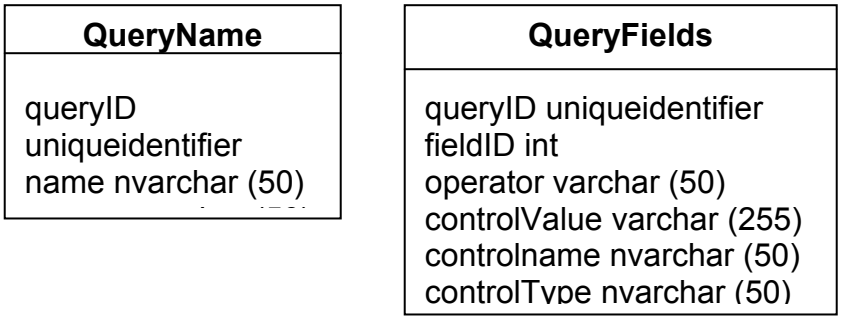

Figura 15. Las tablas planeadas para almacenar filtros para informes

Esta tabla no está instalada en el sistema, puesto que no se han definido en forma definitiva las facilidades que se proporcionarán. La figura 15 ilustra los campos que se han planeado.

Es importante señalar que un usuario podrá usar las mismas restricciones para varias consultas, puesto que es probable que necesite diversos cuadros para la

misma información En el sistema actual éste no es el caso, puesto que al seleccionar un reporte, se pierden las condiciones indicadas en el reporte anterior.

La separacíon de las dos aplicaciones (creación de un reporte y su ejecución) convendría para que los usuarios no se confundan, cosa que contribuirán a una mayor aceptación por parte de los mismos. En otras palabras, cuando el usuario se registra, selecciona una de las dos aplicaciones: la ejecución de un reporte previamente definido, que es la que se ejecuta por omisión, o la actualización de los reportes definidos (informes nuevos o modificaciones a los existentes. La especificación de los criterios de inclusión se ofrecería en ambos, de modo que se pueden definir criterios para su uso posterior, pero tambien indicar los filtros a la hora de ejecutar un reporte. En este último caso, se ofrecerán los criterios ya definidos y almacenados, mismos que se pueden modificar para el uso inmediato, pero también se pueden almacenar los criterios definidos en ese momento.

A pesar de que el CROPFINDER ya permite modificar el orden de las hileras del reporte, indicando un criterio para una columna, sería conveniente poder solicitar ordenamientos más complejos, indicando varias columnas como criterio de *sort*.

Otro aspecto de mejoras reside en que sería agradable, para ciertos informes, la posibilidad de “ordenar” las columnas, puesto que actualmente siempre salen en el orden de los elementos seleccionados. Esto es especialmente útil en consultas que tienen muchas columnas, de modo que para ver algunas de ellas se tienen que recorrer lateralmente las que aparecen, lo que en esta disciplina se conoce como *scrolling.* Al poner las columnas de más interés como iniciales, el usuario podría decidir, en base a la información contenida en estas columnas, si desea o no ver las restantes en un momento dado.

Un aspecto técnico que apareció al usar el explorador MOZILLA FireFox fue que, a pesar de que el objeto TreeView funciona correctamente, su aspecto no es el mismo que bajo Windows Explorer. Concretamente, como muestra este ejemplo en la Figura 16, la lista no está confinada al recuadro, como sucede en el Windows Explorer, sino que sobrepasa los límites del cuadro previsto. En las ilustraciones mostradas anterioremente de las consultas, se puede apreciar la diferencia, puesto que ahí las listas no se salen de los espacios planeados para las mismas.

Para solucionar este problema, se ha planteado el uso de la tecnología AJAX, que subsana esta y otras situaciones similares. De hecho, en la continuación del desarrollo del sistema, probablemente se incorpore el uso de AJAX.

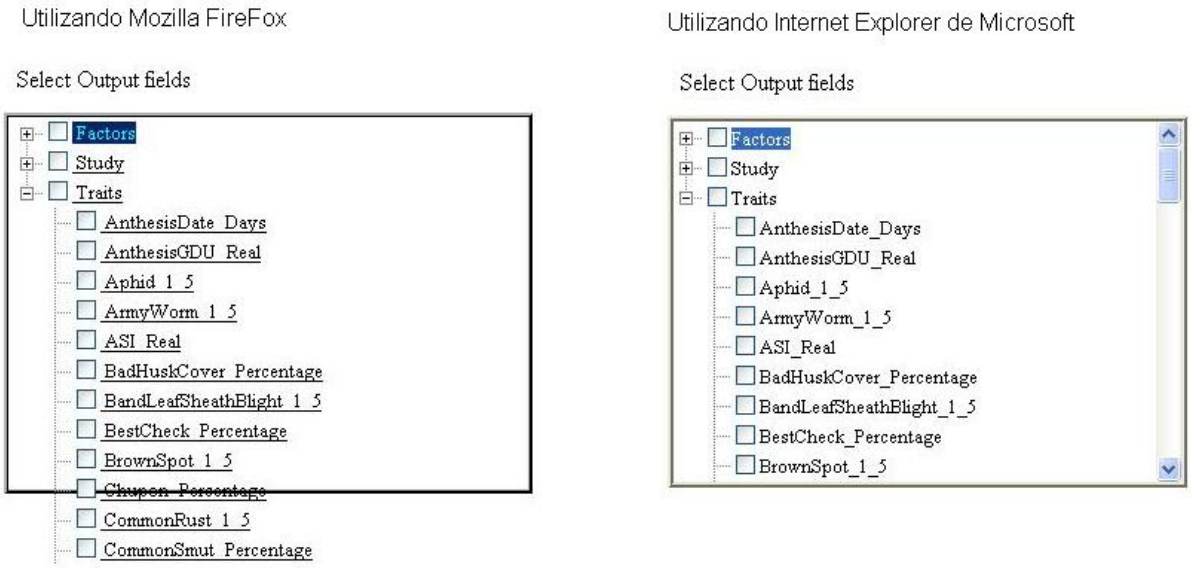

Figura 16. El cambio del aspecto del objeto TreeView con MOZILLA FireFox

CONCLUSIONES

A pesar de que llenaron un vacío importante y resultaron satisfactorias en cuanto a que permiten a sus usuarios, aun los no locales, obtener la información que necesitan, hubo críticas acerca del modo de definir y usar los informes. Esta situación es totalmente predecible para un estudioso de los sistemas de información, como se comentó en la sección del servicio que brinda el sistema. Por lo tanto, se contempla la elaboración de una versión del sistema que incorpore los elementos señalados en la sección anterior sobre cambios sugeridos. Estos cambios se efectuarán en un futuro próximo, respetando el hecho que el sistema se encuentra en producción y no se pueden realizar modificaciones sin consultar previamente a los investigadores que ya están haciendo uso del sistema. Adicionalmente, vale la pena señalar que en la teoría de sistemas de Bauer, utilizada en la concepción de este sistema, resulta conveniente postergar durante un periódo los cambios que no fueran imprescindibles para prestar el servicio: eso hace que se consolide el sistema, y cuando se ofrezcan las mejoras, éstas serán más apreciadas. De hecho, este tipo de postergación tiene otro impacto fundamental: obliga a los usuarios a entender su rol en el servicio, que es el de aprovechar y disfrutar el sistema, en lugar de asignarles una función que, aunque consideren propia, no lo es: ¡criticarlo! Finalmente, se analizaron aspectos relacionados con el rendimiento del sistema, mismo que resultó totalmente satisfactorio en cuanto a duraciones de los procesos mismos. Lamentablemente no se puede afirmar lo mismo sobre los tiempos de respuesta, que se ven afectados frecuentemente por la saturación del servidor en el que se ejecutan los procesos. El organismo que ofrece el servicio ya está trabajando en este aspecto fundamental del funcionamiento del site correspondiente.

LITERATURA CITADA

ADR Infor S.L.2008. Curso de Access XP avanzado – consultas paramétricas. http://www.adrformacion.com/cursos/aaxp/leccion3/tutorial2.html. Accesado 20 de mayo de 2008.

Dhami, H. 2003 IE_Web controls in VB.NET. www.devarticles.com/c/a/ASP.NET/IE-Web-Controls-in-VB.NET/. Accesado 16/01/2008.

Esposito D. 2002. Building WEB Solutions with ASP.NET and ADO.NETMicrosoft Press, USA.

ICIS – International Crop Information System. Wheat Pedigree. Annex 11. Definition of a Pedigree. Abril 11, 2008.http://www.icis.cgiar.org/icis/index.php/Wheat_Pedigree

MacDonal M. 2002. The Book of VB.NET. No Starch Press, Inc. USA.

Payne, T.S., Skovmand, B., Lopez, C.G., Brandon, E., McNab, A. The International Wheat Information System (IWIS): Wheat pedigree management system. GRIP 3, IWIS-Bib, CIMMYT publications, GIBLER. 2001. El Batan, Tex. (Mexico)

Rainard V. Building a Data Warehouse with examples in SQL Server.Apress 2008 USA.

Silberschatz, Korth, Sudarshan. Fundamentos de Bases de Datos. 4ta. Ed. 2002 McGraw-Hill/INTERAMERICANA DE ESPAÑA.

Skovmand, B., Fox, P.N. White, J.W. 1996. In The International Wheat Information System (IWIS): A tool to integrate research.1996. Wheat Breeding Society of Australia, 8. Proceedings; Canberra (Australia); 29 Sep - 4 Oct 1996.

UPOV - International Union for the Protection of New Varieties of Plants. 1961. El Sistema de la UPOV de Protección de Variedades Vegetales http://www.upov.int/en/about/upov_system.htm

APENDICE A. LAS BASES DE DATOS DE LOS SISTEMAS DE TRIGO Y MAÍZ DEL CIMMYT

Contenido

A.1. Tablas del sistema de trigo

A.1.1 Esquema de la base de datos de WPMS. Pedigree Management

PMS_PEDIGREES		**Pedigrees en WPMS**
CID	Int	Identificador de cruza.
SID	Int	Identificador de historia de selección.
PEDIGREE	varchar(250)	Descripción del pedigree.
PMS_NAMES		**Nombres de cruzas e Historias de Selección.**
CID	Int	Identificador de cruza.
SID	Int	Identificador de Historia de Selección.
CAN	char (25)	El nombre común del pedigree o historia de selección.
CAB	char (6)	Abreviación del pedigree..
CC3	char (3)	Código oficial del país. Este código es de 3 caracteres y se toma del standard, ISO 3166..
NYR	Smallint	El año en que la variedad se creo.
PRF	Bit	Indica si este nombre es el preferido.
PMS_NON_STANDARD		**Nombres no Standard para las historias de selección.**
CID	Int	Identificador de cruza.
SID	Int	Identificador de la historia de selección..
NSS	char (60)	Muestra parte de la historia de selección.
SYR	Smallint	Año en que se creo esta variedad.
LID	char (3)	Identificador de localidad. Este campo identifica el origen de la semilla
CC3	char (3)	Código oficial de 3 caracteres. Standard ISO 3166.

MEG	char (3)	Identificación del medio ambiente donde se sembró. Los valores mas comunes son : DRY Dryland, HOT Heat Stress,HR High Rainfall,IRR Irrigated Areas,OE Optimum Irrigated,SA Semi árido, TE Ambiente tropical y otras.
PMS_SELECTIONS		**Historias de Selección**
CID	Int	Identificador de la cruza.
SID	Int	Identificador de la historia de selección.
SYR	Smallint	El año cuando se selecciono la semilla.
LID	char (3)	Identificación del origen de la cruza o selección.
CC3	char (3)	Código oficial de letras para identificación del país.
SNU	Int	Número consecutivo de la selección.
MEG	char (3)	Identificación del medio ambiente donde se sembró. Los valores mas comunes son : DRY Dryland, HOT Heat Stress,HR High Rainfall,IRR Irrigated Areas,OE Optimum Irrigated,SA Semi árido, TE Ambiente tropical y otras
PSID	Int	Apuntador a la historia de selección anterior.

A.1.2. Esquema de la base de datos de WFB. Field Book Management.

WFB_TRIAL		**Trials (Estudios) en WFB**
Tid	Int	Identificador del trial (studio)
Trial_name	Char (30)	Nombre del trial (studio)
Trial_abbr	Char (8)	Abreviatura del trial
Cycle	Char (5)	Ciclo en que se creo el trial.
Mega_env	Char (3)	Identificación del medio ambiente donde se sembró. Los valores mas comunes son : DRY Dryland, HOT Heat Stress,HR High RainfallIRR Irrigated Areas,OE Optimum Irrigated,SA Semi árido, TE Ambiente tropical y otras
Offset	Int	Offset que se aplicara en las entradas del ensayo cuando la primera entrada no se plantara en el primer surco.
Entries	Int	Numero de entradas en el trial (estudio) incluyendo local checks.
Organization	Char (2)	División que creo el trial.

Program	Char (3)	El programa es una subdivisión de la organización. Valores que puede tener: B Barley,BW Bread Wheat DW Durum Wheat,HAR Harineros (Flour),RYE Rye,SPP Species TCL Triticale
Trial_status	Char (1)	Estatus del trial. Los valores que puede tomar son: S New trial F Frozen trial
WFB_OCC		**Ocurrencias para el trial**
Tid	Int	Identificador del trial.
Occ	Int	Identificador de la ocurrencia
Occ_name	Char (30)	Nombre de la ocurrencia
Occ_abbr	Char (8)	Abreviación de la ocurrencia.
LID	Char (3)	Identificación del origen de la cruza o selección..
Country_code	Char (3)	Identificador de 3 letras para el pais.
Cycle	Char (5)	Ciclo de plantación de la semilla
Mega_env	Char (3)	Identificación del medio ambiente donde se sembró. Los valores mas comunes son : DRY Dryland, HOT Heat Stress,HR High Rainfall, IRR Irrigated Areas,OE Optimum Irrigated,SA Semi árido, TE Ambiente tropical y otras.
Offset	Int	Offset que se aplicara en las entradas del ensayo cuando la primera entrada no se plantara en el primer surco.
Planting_date	Char (30)	Fecha en que fue plantado el studio. Format MM/DD/YYYY.
Harvest_date	Char (30)	Fecha de cosecha. Format MM/DD/YYYY.
Planting_Method	Char (2)	Método de plantación.
Planting_space	Int	Espacio entre plantas
Occ_status	Char (1)	“Status” de la ocurrencia. NO UTILIZADO.
Station_id	Int	Identificación de la estación.
Randomization_type	Char (1)	Tipo de randomización que se utilizara. Los valores que puede tomar esta campo son: A, F, N, R.
Coopkey	Int	Identificador del cooperador. Sirve de referencia para la base de datos WINS.
Printed_gn	Bit	1. Ya se imprimió hoja de notas. 0. No se ha impreso hoja de notas.
Db_administrator	Char (3)	Responsable del manejo de esta ocurrencia.
Fb_class	Char (1)	Clase o tipo de libro. Los valores que puede tomar son: N Nacional I Internacional W Banco de Germoplasma
WFB_ENTRYLIST		**Las entradas de un estudio**
Tid	Int	Identificación del estudio (Trial).
Ent	Int	Número de entrada.
Cid	Int	Identificador de cruza.
Sid	Int	Identificador de la historia de selección.

Meg	Char (3)	Identificación del medio ambiente donde se sembró. Los valores mas comunes son : DRY Dryland, HOT Heat Stress,HR High Rainfall,IRR Irrigated Areas,OE Optimum Irrigated,SA Semi árido, TE Ambiente tropical y otras
Ftid	Int	Identificador del trial origen de la madre.
Focc	Int	Ocurrencia origen de la madre.
Fent	Int	Entrada origen de la madre
Mtid	Int	Identificador del trial origen del padre.
Mocc	Int	Identificador de la ocurrencia origen del padre.
Ment	Int	Identificador de la entrada origen del padre
Stid	Int	Identificación del trial origen (STID) . Es el tid con el cual se define el origen de la selección o movimiento junto con SOCC y SENT.

WGBIS_KEYS		**Numero de accesión de las entradas en el banco de Germoplasma**
Tid	Int	Identificador del studio(trial)
Ent	Int	Numero de entrada .
Cid	Int	Identificación de cruza
Sid	Int	Identificador de la Historia de selección.
Accid	Int	Identificación de la accesión
Intrid	Char (10)	Numero de introducción en el banco.

A.1.3. Esquema de la base de datos de WDMS. Data Management.

DMS_GE_VALUES_D		**Datos genotípicos. Valores tipo "datetime"**
TID	int	Identificador de estudio (trial)
OCC	int	Ocurrencia
TRAIT_NO	int	Identificación del trait (característica)
ENT	int	Numero de entrada
DATE_VALUE	datetime	Valor de observación tipo "fecha"
DATE_AGR_AND_DES	datetime	Fecha de la toma del dato
UNIT_ID	int	Identificador de la unidad de medición

DMS_GE_VALUES_N		**Datos genotípicos. Valores tipo "entero".**
TID	int	Identificador de estudio (trial)
OCC	int	Ocurrencia
TRAIT_NO	int	Identificación del trait

		(característica)
ENT	int	Número de entrada
NUMBER_VALUE	int	Valor de observación tipo "entero"
DATE_AGR_AND_DES	datetime	Fecha de la toma del dato
UNIT_ID	int	Identificador de la unidad de medición
DMS_GE_VALUES_R		**Datos genotípicos. Valores tipo "real".**
TID	int	Identificador de estudio (trial)
OCC	int	Ocurrencia
TRAIT_NO	int	Identificación del trait (característica)
ENT	int	Número de entrada
REAL_VALUE	Real	Valor de observación tipo "real"
DATE_AGR_AND_DES	datetime	Fecha de la toma del dato
UNIT_ID	int	Identificador de la unidad de medición
DMS_GE_VALUES_T		**Datos genotípicos. Valores tipo "caracter"**
TID	int	Identificador de estudio (trial)
OCC	int	Ocurrencia
TRAIT_NO	int	Identificación del trait (caracteristica)
ENT	int	Número de entrada
TEXT_VALUE	varchar (255)	Valor de observación tipo "caracter"
DATE_AGR_AND_DES	datetime	Fecha de la toma del dato
UNIT_ID	int	Identificador de la unidad de medición
DMS_RANDOMIZATIONS		**Estudios con diseño.**
TID	int	Identificador del estudio (Trial)
OCC	int	Ocurrencia
CONSECUTIVE_ENTRY_NUMBER	int	Número consecutivo
BLOCK	int	Número de repetición
SUB_BLOCK	int	Número de bloque
RANDOMIZED_ENTRY	int	Identificación de la entrada
DMS_LOCALIDADES		**Información de localidades**
LOC_ID	int	Identificación de la localidad
SUB_DIV_ID	int	Identificación de la división de la localidad
PGM	int	Identificación del programa
LOCATION_NO	int	Numero de localidad

STATE	char (30)	Estado
FARM	char (40)	Nombre del lugar donde se siembra
PREF	bit	Indica si la información de esta localidad es preferida
TOWN	char (30)	Ciudad
INSTITUTION	char (30)	Nombre de la Institución
LONG_DEGREES	int	Grados de longitud de la localidad
LONG_MINUTES	int	Minutos de longitud de la localidad
LONG_GREENWICH_SIDE	char (1)	E. Este W. Oeste
LATI_DEGREES	int	Grados de latitud de la localidad
LATI_MINUTES	int	Minutos de latitud de la localidad
LATI_HEMISPHERE	char (1)	Hemisferio. N. Norte S. Sur
ALTITUDE	int	Altitud
DMS_TRAITS		**Información de traits**
TRAIT_NO	int	Identificación de trait
TRAIT_NAME	char (50)	Nombre del trait
TRAIT_ABBR	char (6)	Abreviatura del trait
CLASS	char (2)	Clase del trait. G. Genético E. Ambiente GE. Genotípico
FORMAT_VB_CELL	char (10)	Formato que se utiliza para mostrar el valor del trait.
FORMAT_VB_MEAN	char (10)	Formato del valor promedio que se utiliza para mostrar el valor del trait.
DMS_UNITS		**Información de unidades de medición**
UNIT_ID	int	Identificación de las unidades
SCALE_TYPE	char (1)	Tipo de unidad. D. Discreta C. Continua
SCALE_NAME	char (40)	Nombre de la unidad
VALUE_TYPE	char (1)	Tipo de valor. N. Entero R. Real T. Texto D. Datetime
		Tabla de control de
CRTL_OT		**procesamiento**
TID	int	Identificación del estudio(trial)
OCC	int	Ocurrencia
TRAIT_NO	Int	Identificador de trait
UNIT_DESC	char (40)	Nombre de la unidad de medida
UNIT_TYPE	char (1)	Tipo de unidad. D. Discreta C. Continua
DATE_AGR	char (11)	Fecha de toma de información
LAST_SUM	char (10)	Número verificar de proceso
NO_OBS	char (6)	Número de observaciones

		procesadas
PROCESAD	char (1)	Status de procesamiento. T Proceso correcto F. Proceso con error
STAT_DAT	char (250)	Cadena de caracteres con datos estadísticos
LAST_UPD	char (11)	Fecha de proceso
CLASS	char (2)	Tipo de trait evaluado
UNIT_TYPE_SOURCE	char (1)	Tipo de dato. R. Real N. Entero T. Caracter D. Fecha

A.1.4. Esquema de la base de datos de WINS. International Nurseries.

COOP		Información general de cooperadores
institute_id	smallint identity(1,1)	Identificación de institución. No utilizado
institute_name	char(40)	Nombre de la institución
Info_cycle	char(5)	Primer ciclo que se le envió semilla.
Phone	char(25)	Número de teléfono
Fax	char(25)	Fax
mail_address	char(255)	Dirección para recibir correo.
ship_address	char(255)	Dirección para recibir envíos de material,
Opdate_bw	char(7)	Fecha de siembra para trigo.
Opdate_tr	char(7)	Fecha de siembra para Triticale
opdate_ba	char(7)	Fecha de siembra para Cebada
Airline	char(20)	Aerolínea para envió de material.
Airport	char(20)	Aeropuerto para envió de material.
Coopkey	Int	Identificación del cooperador
Title	char(5)	Titulo del cooperador. Engineer, Doctor, etc.
Firstname	char(15)	Nombre
Lastname	char(20)	Apellido
email_address	char(60)	Email
Language	char(4)	Idioma del cooperador
institution_type	char(1)	Tipo de institución . (U) Educational (G) Government (C) Private (O) Others
coop_status	Bit	1 = Cooperador activo 0 = Cooperador inactivo

COUNTRY		Información de países
country_code	Int	Identificador del país
fao_country_code	Int	Identificador del país en FAO
iso_country_code	Int	Identificador del país en ISO
country_c3	char(3)	Código del país. 3 letras
country_c2	char(2)	Código del país. 3 letras
region_code_geographic	char(2)	Código de región geográfica.
region_code_economic	char(2)	Código de región economica
region_code_wheat	char(2)	Código de región para trigo
region_code_maize	char(2)	Código de región para maíz
long_country_name	char(44)	Nombre largo del país.
country_name	char(16)	Nombre corto del país.

A.1.5. Esquema de la base de datos de WGB. Germplasm Bank

ALTER_ACCID		**Identificación alterna en otros sistemas**
CID	int	Identificación de cruza
SID	int	Identificación de historia de selección
Alternate_Accid	char (10)	Identificación alterna de accesión.

BIO_STATUS		**Status biológico de la línea**
Status_Id	Int	Identificador consecutivo
Grip_Code	char (10)	Código de Grip
Status	char (40)	Status biológico.

CHARACTERIZATION		**Datas de caracterización**
TID	int	Identificación del estudio (Tid)
ENT	int	Numero de entrada
Daysan	int	Días de floración
Daysmt	int	Días de maduración
Plntht	Real	Altura de planta en centímetros
Grncol	char (255)	Color de grano
glmcol	char (255)	Color de gluma
intrid	char (10)	Numero de introducción
grow_habit	char (3)	Hábitat de crecimiento

COLLECTION		**Catalogo de colecciones**

Intrid	char(10)	Numero de introducción
Collect_id	char(20)	Identificación de colección
Miss_id	char(20)	Identificación de misión
Sample_Number	Int	Numero de semilla colectada
Collection_date	date	Día en que la colección fue recibida
Latitud	real	Latitud del lugar de la colección
Longitud	real	Longitud del lugar de la colección
Method_of_collect	char(25)	Método de colección
Local_ref	char(255)	Referencia física del lugar de colección
Gram_collected	Int	Gramos colectados

DONOR		**Datos de donadores**
Donor_Id	Int	Identificación del donador
Donor_last_name	char(30)	Apellido del donador
Donor_first_name	Char(30)	Nombre del donador
Donor_C3	Char(3)	Código de 3 letras del país del donador
Donor_org	Char(80)	Nombre de la organización donante
Donor_type	Char(10)	Tipo de donador

Logistic		**Logística Data**
Intrid	Char(10)	Numero de introducción
Storage_Address	Char(40)	Lugar de almacenamiento
Storage_Date	Date	Fecha en que el material fue almacenado
Seed_available	Int	Semilla disponible en gramos
Germination_date	Date	Fecha de ultima germinación
Germination_test	Int	Porcentaje de germinación
Moisture_test	Char(20)	Humedad de prueba
Num_regeneration	Int	Numero de regeneraciones
Increase_date	Date	Fecha de ultima regeneración
Increase_loc	Char(10)	Localidad donde se incremento la semilla
Critical_seed	Int	Numero minimode semilla dentro del banco
Critical_germination	Int	Germinación critica.(Numero mínimo de germinación dentro del banco)

MISSION		**Datos de las misiones**
Miss_Id	Char(20)	Identificación de la misión
Miss_descrip	Char(120)	Descripción de la misión
Miss_start	Char(15)	Fecha de inicio de la misión
Miss_end	Char(15)	Fecha final de la misión
Funding_Agency	Char(30)	Agencia patrocinadora

Passport		**Datos de pasaporte**
Intrid	Char(10)	Numero de introducción

Alternate_accession_ID	Char(20)	Número alterno de accesión
Origin_geographic_C3	Char(3)	Código de 3 letras del país
Origin_Inst_Id	Char(10)	Identificación del la institución
Donor_Id	Int	Identificador del donador
Introduction_date	Date	Fecha en que el material fue almacenado en el banco
Observations	Char(255)	Observaciones
Update_user	Char(15)	Identificacion del ultimo usuario que identifico este material
Update_date	Date	Fecha de ultima actualizacion

B.2. Tablas del sistema de maíz

Esquema de la base de datos de IMIS. International Maíze Information System

tblMaizeId		**Tabla con información de pedigree**
Maizeid	int	Identificador de pedigree
StandardPedigree	x(250)	Pedigree
FemaleParent	int	Identificador de la madre de la línea
MaleParent	int	Identificador del padre de la línea
BreedingProgram	x(50)	Nombre del programa que origino la línea

tblExperiment		**Tabla con información de Experimentos**
ExperimentID	int	Identificador del experimento
ExperimentCode	x(100)	Código de identificación del experimento
SeriesID	int	identificación de la serie, si el experimento pertenece a una serie
LocationID	int	identificación de la localidad donde se sembró el experimento
BreedingProgram	x(50)	Nombre del program responsable del experimento
NameOfSeries	x(50)	Nombre de la Serie
Year	int	Año de siembra del experimento
Season	x(1)	Ciclo de siembra. A. Ene-Jun B. Jul-Dic
NameOfExperiment	x(50)	Nombre del experimento
Series	x(2)	identificación de la serie. 2 caracteres
Collaborator	x(250)	Nombre del colaborador.
Management	x(50)	
PlantingDate	datetime	Fecha de siembra
HarvestDate	datetime	Fecha de cosecha
PlotSize	float	Tamaño en m2 del área de siembra
Observation	x(250)	

tblExperimentResult		**Tabla con valores de los experimentos**
EntryResultID	int	identificación del resultado
EntryCode	x(100)	Código de entrada
BreedingProgram	x(50)	Programa responsable del experimento
NameOfSeries	x(50)	Nombre de Serie
Year	int	Año de siembra

Season	x(1)	Ciclo de siembra. A Ene-Jun B. Jul-Dic
NameOfExperiment	x(50)	Nombre del experimento
EntryNumber	int	Numero de entrada
Site	x(50)	Localidad
ExperimentID	int	identificación del experimento
Series	x(2)	identificación de la serie
LocationIDNum	int	identificación de la localidad
MaizeID	int	identificación del pedigree
Pedigree	x(250)	
StockID	int	
LocalCheck	x(5)	Indicación si la entrada en un testigo
		A continuación el nombre de cada columna indica la característica (trait) definido.
AnthesisDate	float	
AnthesisGDU		
AnthracnosePer	float	
Aphid1_5	float	
ArmyWorm1_5	float	
ASI	float	
BacterialLeafStripe1_5	float	
BadHuskCoverPer	float	
BandLeafSheathBlight1_5	float	
BestCheckPer	float	
BlackBundleDiseasePer	float	
BrownSpot1_5	float	
CharcoalRotPer	float	
ChuponPer	float	
CommonRust1_5	float	
CommonSmutPer	float	
CornStuntPer	float	
CrazyTopPer	float	
CurvulariaLeafSpot1_5	float	
DaysToSilk	float	
DMBrownStripePer	float	
DMJavaPer	float	
DMPhilippinePer	float	
DMSorghumPer	float	
DMSugarcanePer	float	
EarAspect1_5	float	
EarHeightCm	float	
EarPosition	float	
EarRotAsp1_5	float	
EarRotFusGram1_5	float	
EarRotFusMon1_5	float	
EarRotPenicil1_5	float	

EarRotStenoc1_5	float
EarRotTotalPer	float
EarWormCm	float
EndospermHardness1_5	float
EPPNo	float
GrainMoisturePer	float
GrainTexture1_5	float
GrainYieldKg	float
GrayLeafSpot1_5	float
HeadSmutPer	float
LargeGrainBorer1_5	float
LateWiltPer	float
LeafBlightMaydis1_5	float
LeafBlightTurcicum1_5	float
LeafRolling1_5	float
LysinePer	float
MaizeWeevil1_5	float
MaturityDays	float
MaturityGDU	float
PhaesLeafSpot1_5	float
PlantAspect1_5	float
PlantHeightCm	float
PlantStandNo	float
PolysoraRust1_5	float
ProteinPer	float
RankNo	float
RankAvgNo	float
RelativeGrainYieldPer	float
RootLodgingPer	float
Rust1_5	float
SeedlingDiseasePer	float
SeedViability1_5	float
Senescence0_10	float
StalkRotBacterialPer	float
StalkRotFusMoniPer	float
StalkRotPer	float
StalkRotPythiumPer	float
StalkRotStenocarPer	float
StarkRotFusGramPer	float
StDevRank	float
StemBorer1_5	float
StemLodgingPer	float
StewartsWiltPer	float
Striga1_5	float
TarspotComplex1_5	float
TasselSize	float

TropicalRust1_5	float	
TryptophanPer	float	
VirusMaizeDwarfMosaicPer	float	
VirusMaizeMosaicPer	float	
VirusMaizeRayadoFinoPer	float	
VirusMaizeRoughDwarfPer	float	
VirusMaizeStreak1_5	float	
VirusMaizeStreakPer	float	
VirusMaizeStripePer	float	
VirusMalDeRioCuartoPer	float	
VirusSugarcaneMosaicPer	float	

tblLocation		**Tabla con información de localidades**
LocationID	int	Identificación de localidad
Country	x(50)	País
Location	x(50)	Nombre de la localidad
Region	x(50)	Región
Latitude	real	Latitud
Longitude	real	Longitud
ElevM	int	Elevación sobre nivel del mar
MAX_TEMP	real	Temperatura máxima registrada
MIN_TEMP	real	Temperatura mínima registrada
PRE	real	Precipitación pluvial
EVAP	real	Evaporación promedio
MegaEnviron	x(75)	Mega-ambiente
MEANTEMP	float	Promedio de temperatura

tblStatistics		**Tabla con datos estadísticos entre repeticiones por característica (trait)**
ExperimentId	int	Identificación de experimento
Trait	x(50)	Nombre de característica (trait)
BreedingProgram	x(50)	Programa responsable del experimento
NameOfExperiment	x(50)	Nombre del experimento
Mean	float	Promedio entre las observaciones
LSD	float	
CV	float	Desviación estándar
NoOfLocation	int	Identificación de localidad
NumReps	int	Número de repeticiones

APENDICE B. LA BASE DE DATOS DEL CROPFINDER

Study		**Tabla con la información general del estudio.**
StudyId	int	Identificación del estudio.
Sname	x(15)	Nombre corto del estudio
PmKey	int	
Title	x(255)	Nombre largo del estudio
Objective	memo	Descripción del objetivo del estudio
InvestId	int	Identificación del investigador o mejorador.
Stype	x(1)	Tipo de estudio.
Sdate	int	Fecha de cuando se va a sembrar el estudio
Edate	int	Cuando de cosecha del studio
Factors		**Tabla con información que define la información del germoplasma que compone un estudio**
Ounitid	int	Identificación de unidad de observación.
Cid	int	Identificación de cruza
Sid	int	Identificación de Historia de selección
Pedigree	x(250)	Pedigree
Hist_Selec	x(250)	Historia de selección de la línea seleccionada.
Occ	int	Numero de ocurrencia dentro del estudio.
Loc_no	int	Identificación de localidad
StudyId	int	Identificación del estudio.
Data_C		**Tablas de los valores observados de cada trait cuando los valores se expresan son cadenas de caracteres**
OunitId	int	Identificación de unidad de observación.
VariatId	int	Identificación de variable a observar.
Dvalue	x(250)	Valor de la observación.
Data_N		**Tablas de los valores observados de cada trait cuando son valores numéricos**
OunitId	int	Identificación de unidad de observación.
VariatId	int	Identificación de variable a observar.
Dvalue	float	Valor de la observación.
Trait		**Tabla con los nombres de las características (traits) a observar.**
TraitId	int	Identificación del trait
TrName	x(50)	Nombre del trait.
Trabbr	x(8)	Abreviatura del trait.

Scale		**Tabla con las escalas o unidades en que se pueden medir los traits.**
ScaleId	int	Identificación de la escala o la unidad.
ScaleName	x(50)	Nombre de la escala.
Variate		**Tabla que nos permite saber cual es el Trait y escala que se utilizo para la recolección de información.**
VariatId	int	Identificación de la variable que se va a medir.
Vname	x(50)	Nombre de la variable.
StudyId	int	Identificación del estudio.
TraitId	int	Identificación de trait que se va a medir.
ScaleId	int	Identificación de la escala se va a utilizar en la recolección de datos.
Dtype	x(1)	El valor en este campo determina si la información que se va a recolectar es numérico (N) o carácter (T).
		Información técnica utilizada para el armado y ejecución de los reportes
FieldSetUp		**Tabla que contiene información de la base de datos**
dataType	x(50)	Tipo de dato
descripción	x(255)	Breve descripción del campo
FieldID	int	Identificador del campo
Fieldname	x(50)	Nombre físico del campo en la base de datos
FieldText	x(50)	Nombre alterno del campo.
groupOfTraits	x(50)	Grupo al que pertenece el campo. Study, Factor o Trait
nullText	x(10)	Texto usado en caso de valor nulo del campo
outputField	bit	Bandera que indica si el campo se muestra al usuario
outputGroup	x(50)	Nombre del grupo al que pertenece el campo al definir condiciones de búsqueda
outputOrder	int	No utilizado
tableName	x(50)	Nombre físico de la tabla en la base de datos
tableAlias	x(50)	Nombre alterno de la tabla.
collection	x(50)	No usado
NoNullValues	int	Numero de valores no nulos del campo
CboValues	bit	Bandera que indica si el campo creara una lista de valores para escoger
NumTotalRegs	int	Numero total de registros de datos en la base de datos
Units	x(15)	Nombre de la unidad de medida
Traitid	int	Identificación del trait

Output Report		**Tabla con la información general del reporte o informe**
outputId	Uniqueidentifier	Identificador único del reporte
name	x(50)	Nombre del reporte
owner	x(50)	Nombre del propietario del reporte.

Output Field		**Tabla con la información de los campos que componen un reporte o informe**
FieldId	int	Identificador del campo
outputId	Unique identifier	Identificación del reporte