Archeologia e Calcolatori
Supplemento 2, 2009, 209-222

PYARCHINIT – PYTHON, QGIS E POSTGRESQL PER LA GESTIONE DEI DATI DI SCAVO

1. Come nasce pyArchInit

Il progetto pyArchInit nasce principalmente dalla necessità sempre più presente nella comunità di archeologi di informatizzare la documentazione degli scavi, utilizzando software che gestiscano dati alfanumerici, multimediali e topografici in una soluzione unica. Al momento, oltre alle tradizionali schede fornite dall'ICCD (Istituto Centrale per il Catalogo e la Documentazione), non vi sono indicazioni specifiche da parte delle Soprintendenze competenti né sull'obbligatorietà di una consegna in formato digitale né sulle norme di implementazione di schede informatizzate, lasciando la produzione di questo tipo di documentazione alla scelta dell'archeologo che opera sul campo. Questo ha portato negli ultimi anni allo sviluppo di numerose soluzioni, con l'utilizzo di software open source, gratuito oppure proprietario con licenze a pagamento; tuttavia nessuna di queste soluzioni ha ancora saputo soddisfare appieno le esigenze dell'archeologia portando sempre più gruppi di ricerca a realizzare in maniera indipendente proprie applicazioni.

Ad oggi le Soprintendenze tendono a richiedere ancora la documentazione di scavo in formato cartaceo e non dettano leggi in materia di informatizzazione dei dati; si giunge così ad avere supporti disomogenei e banche dati spesso poco allineabili oppure realizzate con software a pagamento che obbligherebbero gli enti ad acquistarne le licenze per poterli gestire.

Per tali motivi l'archeologia guarda ormai da anni con interesse all'uso di software open source gratuito, che offre una serie di vantaggi imprescindibili per lo sviluppo di applicazioni: codice aperto e modificabile a seconda delle esigenze, licenze gratuite che abbassano i costi di installazione e permettono ad un gruppo di sviluppo un libero uso del software su più macchine, possibilità di ridistribuire il proprio prodotto liberamente ottenendo in feedback la crescita della propria soluzione da parte di terzi, favorendo l'omogeneità delle soluzioni adottate.

2. Obiettivi

pyArchInit è stato progettato su tali premesse e mira a soddisfare le esigenze con una soluzione unica e un pacchetto di software limitato che garantisca nel tempo stabilità, sviluppo e facilità di installazione e aggiornamento. Obiettivo finale è la realizzazione di una piattaforma GIS ad alta scalabità tra sistemi operativi differenti, in cui schedatura alfanumerica, geometrie GIS e

dati multimediali risiedano all'interno di un unico sistema per mantenere il più possibile l'integrità del dato di partenza, offrano all'utente un approccio all'uso molto veloce e solido, rimanendo tuttavia aperto a modifiche e personalizzazioni da parte di altri sviluppatori.

La prima parte del pacchetto che è stata programmata e realizzata è il sistema di gestione delle schede di Unità Stratigrafica (pyarchinit_US) proprio per la necessità di gestire nell'immediato la documentazione dei cantieri in corso. Al momento è stato scartato l'uso di una schedatura su server remoti per due motivi: la legislazione carente in materia di conservazione di dati statali, quali sono le documentazioni di scavo, non offre garanzie agli archeologi che si vogliono avvalere di questa tecnologia per la schedatura; problematica è anche la necessità di schedare i dati sul campo, dove non sempre è possibile avere una connessione Internet; inoltre, l'uso di collegamenti wireless per ogni computer utilizzato su ogni scavo, aumenterebbe i costi di un problema risolvibile con una semplice query SQL per l'allineamento dati su di un unico server locale. Tuttavia il sistema è stato pensato anche per poter gestire in futuro i dati via web, sia per le fasi di data-entry, sia per una consultazione mediante Web-GIS.

3. The cookbook

Come è buona prassi, prima di iniziare ad implementare l'intero programma, sono stati passati al vaglio linguaggi di programmazione, database e software GIS che soddisfacessero appieno le esigenze richieste dal progetto. Il primo principio applicato è l'utilizzo esclusivo di software FOSS (Free Open Source Software), in modo da agganciarsi a progetti esterni oppure dare la possibilità a terzi di inserirvisi; altro aspetto importante è la necessità di avere un funzionamento stabile di ogni singolo componente all'interno dei tre principali sistemi operativi in uso: Windows, Linux e Mac OS X.

Il software di scripting, base per tutto il programma, doveva possedere librerie complete e semplici da utilizzare per sviluppare interfacce grafiche, gestire la connessione ai database e l'aggancio al GIS. Il database più adatto doveva avere la possibilità di immagazzinare e gestire in maniera fluida grosse moli di dati, sia alfanumerici che spaziali. Il programma GIS, invece, doveva dare la possibilità di georeferenziare planimetrie raster, interagire con il linguaggio di programmazione scelto e avere accesso ai dati geografici e alfanumerici del database, senza tralasciare una buona strumentazione di base, tra cui sistemi per la digitalizzazione delle geometrie e gli strumenti di analisi topografica. Dopo un'attenta analisi, basata anche sulla facilità d'uso di questi strumenti da parte di un utente medio, sono stati scelti i seguenti strumenti:

1. GIS: QGIS/GRASS;
2. Linguaggio di programmazione: Python; moduli Python:

I. Database API: psycopg2;
II. GUI (Graphical User Interface) = pyQT;
III. ORM (Object-Relationship Mapper): SQLalchemy;
3. Librerie grafiche per interfacce: Qt;
4. Database: PostgreSQL: Libreria geografica: PostGIS.

Vediamo ora nello specifico i software scelti per questo progetto:

– QGIS/GRASS: software GIS altamente versatile, con interfaccia grafica e sistema di gestione molto intuitivi, fondamentale per un programma complesso come il GIS, che nel nostro caso deve essere utilizzato da utenti di livello medio, le cui competenze maggiori sono in campo archeologico. Ha un set base di strumenti per la digitalizzazione ed è in grado di gestire layer di tipo shapefile e tabellari PostgreSQL/PostGIS. Accetta la lettura di tabelle spaziali virtuali realizzate tramite le view di PostgreSQL. Offre un sistema di plug-in realizzati in Python, che permette agli utenti di realizzare proprie soluzioni in maniera abbastanza rapida e con nozioni di programmazione basilari. Ha un funzionamento molto stabile e uguale su tutti i sistemi operativi. Grazie al binding tra QGIS e Python, è possibile accedere alle librerie pyQGIS, per sfruttare tutte le funzioni di interfaccia ed elaborazione geometrie proprie di QGIS.
– Python: è stato scelto per la sua semplicità e pulizia del codice e per la potenza di sviluppo. È un linguaggio che permette ad utenti medi di entrare in breve tempo nel mondo della programmazione. Inoltre, la sua integrazione all'interno di QGIS ha reso quasi obbligatoria la sua scelta. Offre numerosi moduli per personalizzare le proprie applicazioni e nel caso di pyArchInit vengono utilizzate in particolare:

– SQLalchemy: un ORM (Object Relationship Model) che permette di realizzare modelli complessi per la gestione di database, indipendentemente dal software utilizzato. pyArchInit fin dall'inizio è stato pensato per poter migrare su altri database a seconda di chi utilizza il codice. SQLalchemy, semplicemente indicandogli quale database si sta utilizzando (PostgreSQL, MySQL, Sqlite o Oracle), converte le chiamate nel codice SQL compatibile con il vostro motore;
– Psycopg2: è l'API utilizzato per generare la connessione tra Python o SQLalchemy e il database PostgreSQL;
– PyQT: è il modulo utilizzato per sfruttare appieno le librerie grafiche QT per lo sviluppo delle interfacce;
– Reportlab: viene utilizzato per l'output dei dati in formato PDF, al momento non ancora implementato.

PostgreSQL: dal momento che pyArchInit dovrà gestire una serie complessa di dati, non era possibile separare a livello di database la catastazione dei dati alfanumerici e topografici; utilizzare *shapefiles* e tabelle DBF per la raccolta dati rende il sistema di difficile amministrazione. L'uso di database

come file Base di OpenOffice o prodotti con licenze a pagamento tipo Filemaker o MS Access garantiscono elevati standard qualitativi nella gestione del dato alfanumerico, ma un dispendio elevato di energie nel mantenere allineati i Database con le relative geometrie, a patto di non utilizzarli come *front-end*.

Quindi, la miglior soluzione percorribile è sfruttare un GeoDB, un database in grado di gestire dati alfanumerici e geometrie in un'unica soluzione, che nel nostro caso andava nella direzione di un binomio QGIS + PostgreSQL/PostGIS.

PostgreSQL è uno dei database open source più avanzati, offrendo una gestione molto pratica di grandi quantità di informazioni oltre ad un sistema di programmazione interna che aumenta le possibilità di plasmare l'organizzazione del database sui modelli in cui il mondo reale viene interpretato. Tra i numerosi vantaggi che fanno prediligere l'utilizzo di questo database rispetto ad altri in campo GIS è la possibilità di sfruttare PostGIS, un'estensione spaziale che fornisce il sistema di gestione dati su cui sono basati i programmi di elaborazione spaziale. Inoltre PostgreSQL permette la creazione delle VIEW, ovvero delle "tabelle" non fisicamente presenti nel database che eseguono dei *join* complessi tra altre tabelle. In pratica si tratta di *query* SQL che vengono eseguite ogni volta che sono chiamate, senza andare ad occupare spazio ulteriore e senza richiedere una manutenzione specifica, garantendo un allineamento costante dei dati, nel nostro caso catastati nella tabella di scheda US e nelle tabelle che contengono le geometrie poligonali di unità stratigrafiche positive e negative e i poligoni relativi alle caratterizzazioni di strato.

QT: è una libreria grafica multi-piattaforma per la realizzazione di interfacce grafiche mediante *widgets*. Lo sviluppo delle GUI, grazie ad un *tool* visuale, risulta molto veloce e intuitivo; si ottengono soluzioni molto chiare e il loro aspetto rimane invariato da un sistema operativo all'altro, senza richiedere grossi aggiustamenti. Molto semplici da installare sotto Mac OS X e Linux (nel nostro caso Ubuntu), pongono qualche problematica in più sotto Windows, richiedendo alcuni accorgimenti che vanno un po' oltre le conoscenze di un utente base. La possibilità di disegnare a video le interfacce, convertibili e gestibili via Python tramite le librerie pyQT, mettono qualsiasi utente nelle condizioni di realizzare in tempi rapidi prodotti di buona qualità; grazie alle librerie pyQGIS divengono implementabili applicazioni GIS standalone o plug-in importabili in QGIS.

4. Struttura e utilizzo

pyArchInit è stato pensato per avere una struttura ordinata e aperta il più possibile ad integrazioni future. Al momento può essere sfruttato sia come applicazione standalone per l'inserimento delle schede di Unità Stratigrafica,

sia come plug-in per QGIS, che oltre alla catastazione delle US, mette l'utente in grado di sfruttare il GIS di scavo passando dai dati verso le geometrie e viceversa. La struttura di pyArchInit mantiene separati tutti gli ambiti del progetto, migliorando il sistema di sviluppo e permettendo a terzi utenti di modificare a proprio piacimento non solo il codice, ma l'intero pacchetto relativo ad un'unica porzione, offrendo una lettura veloce e immediata dello schema gestionale. Presentiamo di seguito la struttura del plugin pyarchinit_US, i relativi moduli e le loro funzioni, oltre allo schema del database.

4.1 *Struttura del plugin pyarchinit_US*

1 *pyarchinitus*: directory principale;

1.1 *__init__.py, pyarchinitUSplugin.py*: caricamento e gestione dell'attivazione del plugin sotto QGIS;
1.2 *pyarchinit_US_mainapp.py*: è l'applicazione main del sistema e contiene sia le funzioni legate ai pulsanti dell'interfaccia, che le funzioni di base per la gestione del database (controlli, avvisi, sistema di ricerca, etc.);
1.3 *sortpanelmain.py*: gestione funzioni e interfaccia del sistema di ordinamento dei dati;
1.4 *settings.py*: è il file che contiene i settaggi per la connessione al database ed eventuali percorsi verso directory esterne, web link, etc.;

2 *modules*: directory contenente i moduli per la gestione di: database, funzioni GIS, interfacce e *script* di uso generico;

2.1 *db*: directory dei moduli relativi alla gestione del database;
2.1.1 *pyarchinit_conn_strings.py:* gestisce le stringe di connessione al database;
2.1.2 *pyarchinit_db_manager.py*: modulo contenente tutte le funzioni per la gestione del database, come la creazione delle tabelle, sistema di inserimento dati, ricerca, eliminazione, etc.;
2.1.3 *pyarchinit_db_mapper.py*: contiene le classi per mappare le tabelle e permettere ad SQLalchemy di gestirle indipendentemente dal database che viene utilizzato;
2.1.4 *pyarchinit_db_structure.py*: definisce la struttura dell'intero database;
2.2 *gis*: moduli per l'interazione con librerie GIS come pyQGIS, GDAL, etc.;
2.2.1 *pyarchinit_pyQGIS.py*: funzioni di interazione tra QGIS e database;
2.3 *gui*: directory dei files relativi all'interfaccia grafica;
2.3.1 *pyarchinit_US_ui.ui*: file contiene il codice dell'interfaccia della scheda US disegnata tramite il Designer delle QT;
2.3.2 *pyarchinit_US_ui.py*: file per la gestione dell'interfaccia della scheda US in Python, generato via pyQT;
2.3.3 *sort_panel.ui*: file contenente il codice dell'interfaccia del pannello di ordinamento dati disegnato tramite il Designer delle QT;

2.3.4 *sort_paneli.py*: file per la gestione dell'interfaccia del pannello di ordinamento dati in Python, generato via pyQT;
2.3.5 *resources.qrc*: file contenente le immagini legate alla GUI generata dal Designer delle QT;
2.3.6 *resources_rc.py*: conversione del file resources.qrc in codice Python, generato via pyQT;
2.3.7 *pyQt4_generator.py*: file per la conversione in codice Python di tutte le interfacce realizzate con il Designer delle QT;
2.3.8 *qrc_generator.py*: file per la conversione in codice Python del file resources. qrc;
2.3.9 *images*: directory dedicata alle immagini utilizzate per lo sviluppo delle G.U.I.;
2.4 *utility*: contiene i moduli utilizzati per operazioni generiche come la gestione di liste, tuple, dizionari in Python, script di elaborazione dei dati, manutenzione delle cartelle o dei files;
2.4.1 *pyArchInit_backup_dir.py*: sistema di backup del plugin utilizzato in fase di sviluppo;
2.4.2 *pyArchInit_utility.py*: collezione di funzioni per la gestione ed elaborazione dei dati, come conversioni, automazione di script di analisi, etc.;
3 icons: directory per le icone del progetto.

4.2 *Struttura del database*

database:

– pyArchInit_db:

Tabelle:

– us_table: è la tabella che gestisce i dati alfanumerici delle US; l'univocità della scheda è garantita mediante un controllo tramite pyarchinit_db_manager.py su sito, area, us; l'identità del record è garantita dall'Idus, un numero intero univoco sfruttato per gestire le ricerche tra le geometrie rappresentate in QGIS e i dati alfanumerici contenuti in PostgreSQL;
– pyunitastratigrafiche: tabella spaziale di tipo poligono che accoglie le geometrie relative a US positive e negative;
– pyuscaratterizzazioni: tabella spaziale di tipo poligono che accoglie tutte le caratterizzazioni di strato, sia a livello di rappresentazione grafica degli elementi reali individuati in strato (per esempio, frammenti di laterizi o pietra, macchie di concotto, reperti individuati in strato, etc.) sia delle convenzioni grafiche (simboli per rappresentare, carboni, calce, malta, etc.);
– pyusview: è la query di tipo VIEW che permette l'allineamento tra i dati della tabella us_table e le geometrie pyunitastratigrafiche; il join avviene sull'identità tra i campi: sito, area, us;

– pyuscaratterizzazioniview: è la query di tipo VIEW che permette l'allineamento tra i dati della tabella us_table e le geometrie pyuscaratterizzazioni; il join avviene sull'identità tra i campi: sito, area, us.

4.3 *Requisiti d'installazione*

Attualmente il plug-in in via di perfezionamento non è stato reso pubblico, ma un volta terminata la prima fase di debug, il codice sarà liberato sul sito www.pyarchinit.altervista.org.

Per poter funzionare correttamente, il plug-in richiede pochi accorgimenti ed è necessario aver installato:

1. QGIS, versione 1.0 o superiore.
2. PostgreSQL versione 8.2 o superiore con supporto PostGIS.
3. Python 2.5, le librerie pyQt4 (e relative SIP), sqlalchemy 3.0 e psycopg2.
4. In PostgreSQL sarà necessario creare un Database chiamato pyarchinit_db (o con un altro nome che dovrà essere dichiarato in settings.py).
5. Aprire il file pyarchinitus/settings.py e inserire i dati relativi alla vostra connessione con il Database.

4.4 *Installazione e attivazione*

Una volta inserito il plug-in nella cartella apposita di QGIS (che varia a seconda del sistema operativo su cui vi trovate), basterà avviare QGIS e dal Plugin Manager selezionare pyArchInit: il plug-in apparirà sia nella lista che nell'interfaccia grafica dedicata ai plugin di QGIS. Al primo avvio, pyArchInit penserà a creare le tabelle necessarie alla gestione delle vostre Unità Stratigrafiche.

Una volta installate tutte le tabelle, apparirà l'interfaccia grafica per l'inserimento dati relativo alle Unità Stratigrafiche; l'interfaccia è molto semplice e intuitiva da usare ed è stata divisa in due aree: la porzione superiore di tipo statico con i *tool* di gestione dei dati; la parte inferiore dinamica, con una serie di pannelli dedicati al data entry.

La parte superiore è stata suddivisa in 5 sezioni:

1. sistema di gestione delle funzioni GIS;
2. pulsantiera per la gestione dei dati alfanumerici; le funzioni sviluppate sono di: immissione dati, salvataggio, ordinamento, navigazione, eliminazione;
3. informazioni relative al database: numero di record presenti, record utilizzati e modalità d'uso dell'interfaccia: navigazione, inserimento dati o ricerca;
4. area dotata di una pulsantiera per la navigazione tra i record, immissione e salvataggio dati;
5. campi relativi all'identificatore univoco della scheda US composto dal sito indagato, area di scavo, numero di unità stratigrafica progressivo; sono stati

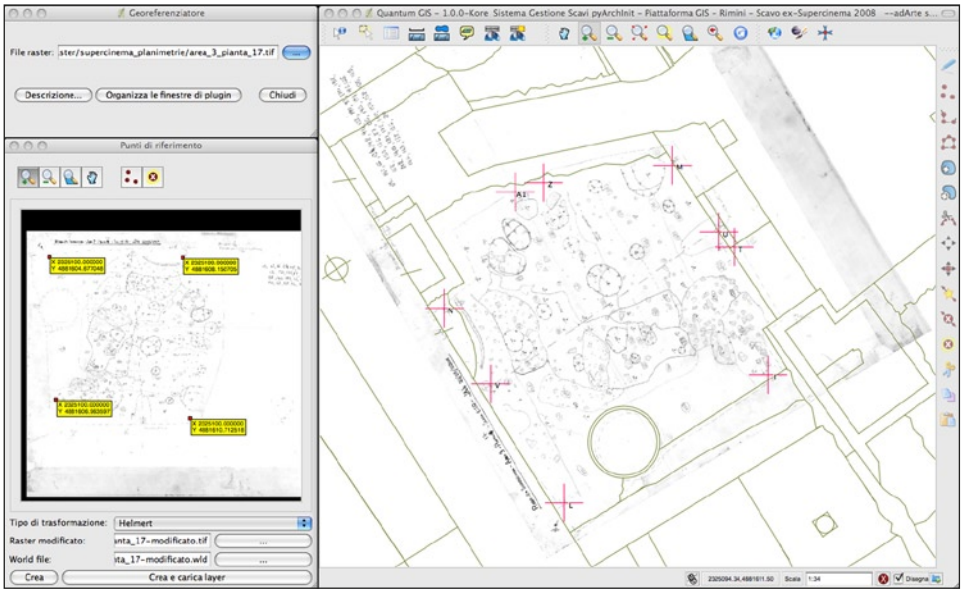

Fig. 1 – Georeferenziazione della pianta di strato mediante QGIS.

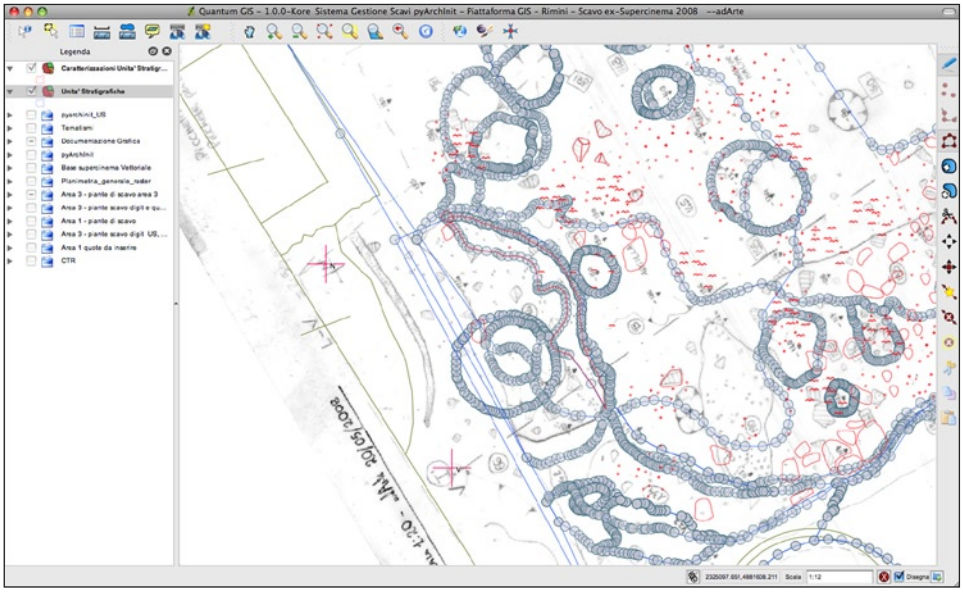

Fig. 2 – Digitalizzazione delle Unità Stratigrafiche presenti nella pianta su QGIS.

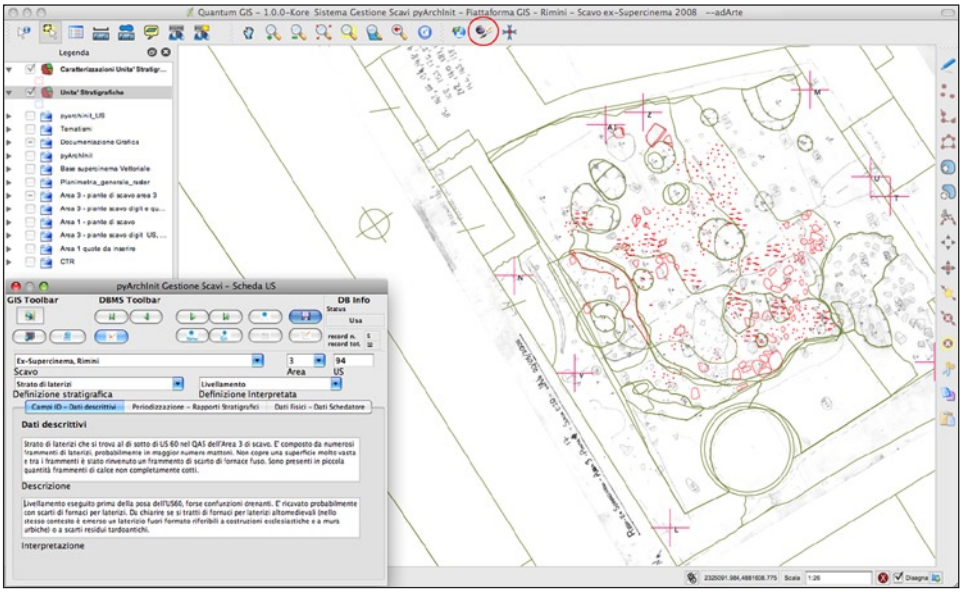

Fig. 3 – Compilazione delle schede US mediante pyArchInit attivato come plugin su QGIS. Nel cerchietto è visibile il pulsante di attivazione del plugin.

aggiunti in questa parte di interfaccia anche i dati relativi alla definizione stratigrafica e interpretata per migliorare la lettura della scheda.

La parte inferiore prevede tre pannelli distinti per la compilazione della scheda:

1. dati descrittivi: contengono la descrizione stratigrafica e la sua interpretazione;
2. periodizzazione-Rapporti Stratigrafici-Strutture;
3. dati da scavo: segnalazione dello schedatore, anno di scavo, dati fisici di strato, etc.

L'esperienza sul campo insegna che compilazione della scheda US e rappresentazione grafica avvengono in momenti separati; se poi vogliamo informatizzare sia i dati alfanumerici che quelli topografici, è probabile che dovremo considerare che queste fasi avvengano in tempi, luoghi e computer differenti. Per questo pyArchInit prevede due ambienti di gestione indipendenti per schede e disegni; l'archeologo che realizza le schede e chi disegna non dovranno lavorare in sincronia, ma ognuno porterà avanti in maniera autonoma il proprio lavoro (Figg. 1, 2, 3) e sarà pyArchInit, una volta migrati i dati in un unico server, a gestire i dati e fare i *join* tramite le tabelle di *view*.

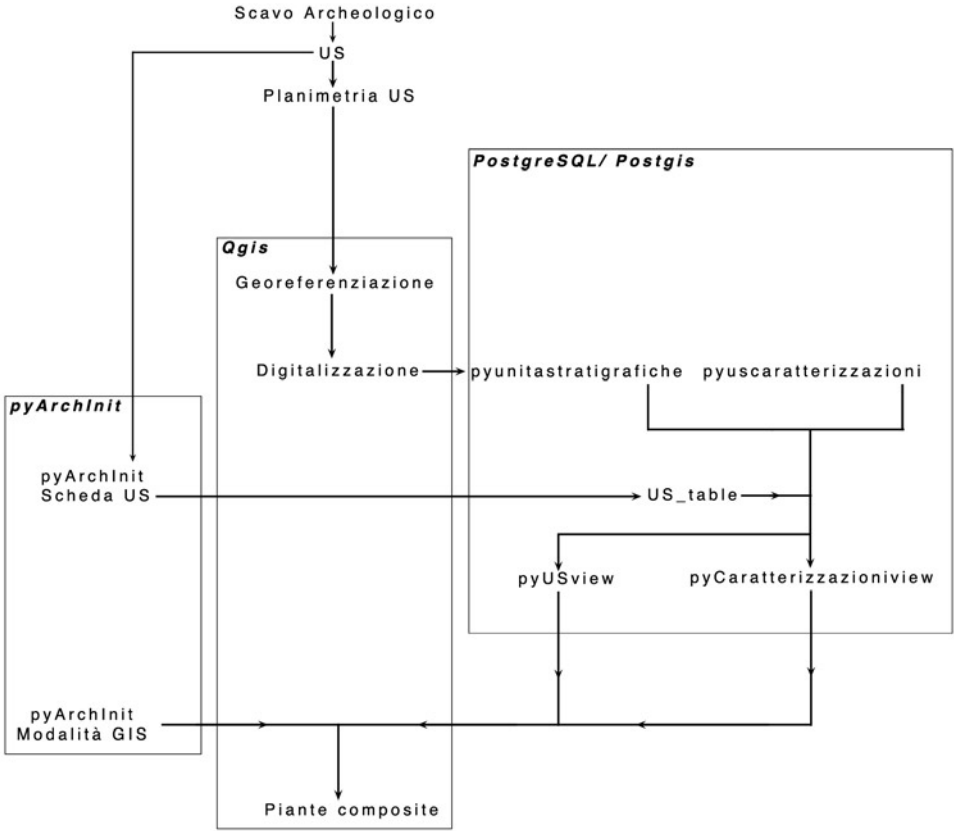

Fig. 4 – Schema del processo di catastazione e output dei dati relativi alle Unità Stratigrafiche attraverso pyArchInit.

Un limite al momento risulta la migrazione dei dati alfanumerici da computer a computer, gestita in parte in maniera manuale, tramite query SQL e con l'ausilio di una funzione scritta in Python e contenuta nel file pyArchInit_db_manager.py, che riallinea il numero progressivo dell'IDus. Meno problematico risulta l'allineamento delle geometrie, che una volta esportate in formato shapefile, possono essere semplicemente importate in PostgreSQL e copiate e incollate nei nostri layers tabellari per le US e le caratterizzazioni (Fig. 4). La scheda al momento permette le normali operazioni di management di un DB, come creazione di nuovi record, salvataggio, eliminazione, ordinamento in base a tutti i campi e nell'ordine desiderato, il sistema di navigazione permette lo spostamento sia da un record all'altro che al primo o all'ultimo.

La gestione dei controlli per ora è molto essenziale e si limita a verificare che la scheda inserita in un nuovo record noi sia già presente, secondo il criterio

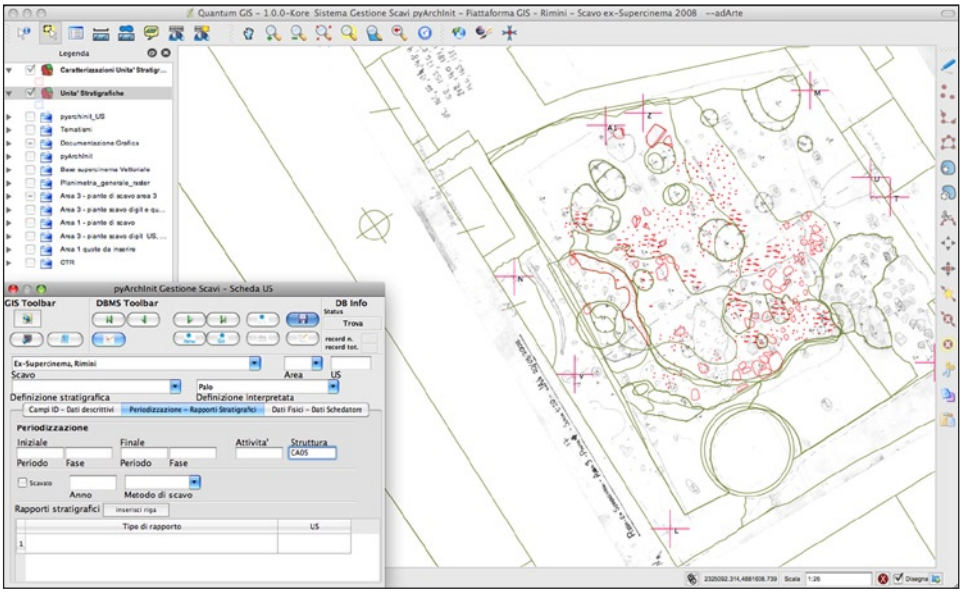

Fig. 5 – Impostazione della ricerca in pyArchInit.

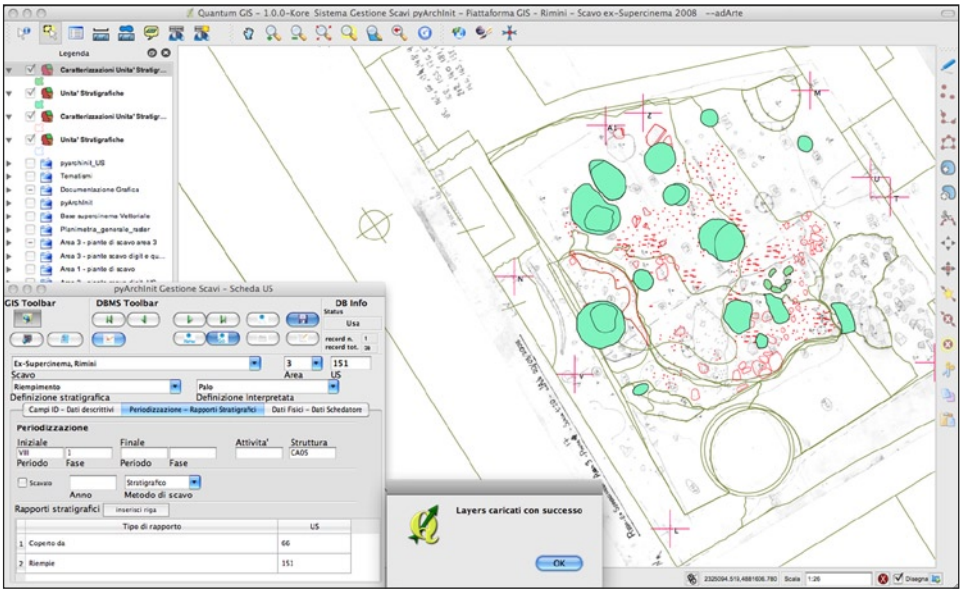

Fig. 6 – Sono evidenziate le US caricate su QGIS da pyArchInit e prelevate da PostgreSQL in base alla precedente ricerca (vedi Fig. 5).

di unicità delle US basato su sito, area e numero US; inoltre un controllo sempre attivo verifica se i dati di una scheda in consultazione vengono modificati chiedendo all'utente se si vogliono salvare. Ulteriori controlli dovranno essere applicati sull'inserimento dei dati all'interno dei campi in base a criteri prestabiliti.

La ricerca permette di fare operazioni su un solo set di variabili che sfruttano tutti i campi della scheda legati tra di loro da un operatore booleano di tipo AND; quindi una ricerca che vede compilati tre campi, per esempio richiamando i dati di uno scavo con una ricerca su sito = "Ex-Supercinema, Rimini", definizione interpretata = "Palo" e sigla di struttura = "CA05" (Fig. 5) troverà solo le US interpretate come pali relativi alla struttura in materiali deperibili n. 5 e individuata all'interno della scavo dell'Ex-Supercinema di Rimini (indagini eseguite da adArte snc per conto della Soprintendenza ai Beni Archeologici dell'Emilia Romagna nel 2008). A breve sarà implementata la scelta dell'operatore booleano per eseguire ricerche di tipo AND o OR, e aumentare notevolmente le potenzialità del sistema.

Le funzioni GIS prevedono la possibilità di attivare o disattivare la "modalità gis", durante la quale qualsiasi ricerca eseguita sull'interfaccia viene trasformata in una pianta composita in QGIS (Fig. 6).

Questo permette di realizzare infinite piante di scavo basate su tutti i criteri presenti nella scheda US e incrociati tra di loro in maniera pressoché illimitata. Questo ci dà modo di analizzare in tempo reale la stratigrafia, la correttezza dell'immissione dei nostri dati, di creare piante di fase o tematiche per la ricostruzione storica del nostro sito o per generare piante per pubblicazioni o relazioni per le Soprintendenze, limitando altamente costi e tempi. pyArchInit prevede anche un feedback da GIS a Database, dove l'utente può selezionare un set di geometrie e aprire le schede US relative per la consultazione o la loro modifica. In questo modo diviene molto rapido l'aggiornamento delle interpretazioni di scavo a livello di dati inseriti nelle schede, oppure è possibile abbattere i tempi che intercorrono tra lettura delle piante e reperimento della scheda, evitando anche errori di consultazione che possono avvenire quando si lavora sul cartaceo.

5. Sviluppi futuri

pyArchInit è un progetto ancora del tutto in via di sviluppo e carente di alcune parti essenziali; le prossime funzioni che saranno implementate prevederanno un output in formato pdf delle schede US tramite il modulo per Python Reportlab; saranno migliorati i sistemi di aggiornamento delle liste a tendina e verranno implementati i sistemi di controllo sull'immissione dei dati. Altri due moduli saranno aggiunti in breve e serviranno alla gestione della galleria multimediale e alla schedatura dei reperti.

Per quanto riguarda GIS e database, al momento è in corso l'integrazione all'interno di QGIS del dataProvider per Spatialite, la libreria geografica per SQLite. pyArchInit, appoggiandosi a SQLalchemy è in grado di passare con il solo cambio dei settaggi dall'uso di PostgreSQL a SQLite, che offre come vantaggio una installazione molto più semplice e la gestione di tutto il database all'interno di un unico file, cosa che aumenterebbe la praticità d'uso di pyArchInit; infatti Python, grazie al modulo sqlite3 distribuito con l'installer, permette una gestione diretta del database. Questo sarebbe un ulteriore passo verso una soluzione all-in-one per l'archeologia molto semplice da distribuire e utilizzare.

6. Documentazione on-line

Siti web di riferimento per sviluppare pyArchInit e per la stesura di questo articolo:

– sito del progetto: www.pyarchinit.altervista.org
– PostgreSQL: www.postgresql.org
– PostGIS: postgis.refractions.net
– psycopg2: initd.org
– Python: www.python.org
– pyQt: www.riverbankcomputing.com
– Qgis: www.qgis.org
– QT: www.qtsoftware.com
– SQLalchemy: www.sqlalchemy.org

Tutorial consigliati:

– Dive Into Python, Mark Pilgrim
– Pensare da informatico, Versione Python, A. B. Downey, J. Elkner e C. Meyers

Qgis:

– Quantum GIS wiki, python bindings
– Developing QGIS python plugins HOWTO
– My first Python Qgis Plugin, di Richard Duinvenvoorde

SQL:

– Guida linguaggio SQL di Lucio Benfante

SQLalchemy

– A step-by-step SQLalchemy tutorial

Luca Mandolesi
adArte s.n.c.
Rimini, Società di ricerca per i Beni Culturali

BIBLIOGRAFIA

Carandini A. 1996, *Storie dalla terra. Manuale dello scavo archeologico*, Torino, Einaudi.

Francovich R. 1990, *Dalla teoria alla ricerca sul campo: il contributo dell'informatica all'archeologia medievale*, «Archeologia e Calcolatori», 1, 15-27.

Francovich R., Valenti M. 2000, *La piattaforma GIS dello scavo ed il suo utilizzo: l'esperienza di Poggibonsi*, in G.P. Brogiolo (ed.), *II Congresso Nazionale di Archeologia Medievale (Brescia, 28 settembre-1 ottobre 2000)*, Firenze, All'Insegna del Giglio, 14-20.

Fronza V. 2000, *Il sistema di gestione degli archivi dello scavo di Poggio Imperiale a Poggibonsi (Insegnamento di Archeologia Medievale dell'Università di Siena). Una soluzione all'interno della "soluzione GIS"*, «Archeologia e Calcolatori», 11, 125-137.

Fronza V. 2003, *Principi di Database Management in Archeologia: l'esperienza senese*, in R. Fiorillo, P. Peduto (ed.), *III Congresso Nazionale di Archeologia Medievale (Salerno, 2-5 ottobre 2003)*, Firenze, All'Insegna del Giglio, 629-632.

Guidi A. 1994, *I metodi della ricerca archeologica*, I, Manuali Laterza, Roma-Bari, Laterza.

Harris E.C. 1983, *Principi di stratigrafia archeologica*, Roma, NIS.

Nardini A. 2000, *La piattaforma GIS dello scavo di Poggio Imperiale a Poggibonsi (Insegnamento di Archeologia Medievale dell'Università di Siena). Dalla creazione del modello dei dati alla loro lettura*, «Archeologia e Calcolatori», 11, 111-123.

ABSTRACT

pyArchInit is a tool for managing archaeological dataset with a high portability on main platforms. The first step of the project has been the creation of a GIS platform with a DBMS to manage the Stratigraphic Units.