A Redacle munkafolyamat-irányító rendszer

Egy MySQL alapú rendszer kezeli egy ötszázezer alkatrészből álló új tudományos műszer megépítési folyamatának az adatkezelését, minőségellenőrzését és könyvelését. Hogyan alkalmazhatjuk ezt a módszert saját gyártófolyamatainkhoz?

A szubnukleáris részecskék valóban igen kisméretűek, kimutatásukhoz és megmérésükhöz hatalmas érzékelőkre van szükség. Ezt a tudományterületet nagy energiájú fizikának (high-energy physics, HEP) nevezik, a kísérleti HEP pedig napjaink egyik élvonalbeli tudományterülete. Felhasználja és továbbfejleszti a legújabb módszereket, új eszközöket talál fel, és ösztönzi az ismeretek cseréjét. Mindezen okokból kifolyólag a HEP már hosszú ideje a nyílt forrású programok birodalma.
A rossz hír az, hogy a HEP időközben egyre bonyolultabbá vált: ami korábban kisipari módszerekkel is előállítható volt, az mára ipari folyamat, ami a különleges célra létrehozott és rendszerint igen drága folyamatirányító programot is megkívánja. Mindezt saját tapasztalataink alapján állíthatjuk: egy nagy nemzetközi csoport egy 12 500 tonnás detektor, a CMS (Compact Muon Solenoid) legyártását rendelte meg, amelynek a Genovai CERN-ben, a nagy hadronütköztetőben kellene 2007-ben az adatokat összegyűjtenie. Az Olaszországi Nukleáris Fizikai Intézet (INFN) által támogatott, Rómában, a La Sapienza Egyetem fizikai tanszékén működő csoportunk dolgozik az elektromágneses kaloriméter megépítésén. A kaloriméter körülbelül ötszázezer alkatrészből áll, amelyek között szcintilláló kristályok és fotoérzékelők is vannak. A folyamatnak adatkezelői, minőségellenőrzői és könyvviteli támogatásra van szüksége, amelyek mindegyike a munkafolyamat-irányításra támaszkodik.

1. kép A mai nagy energiájú fizika egy ipari folyamatra hasonlít

A munkafolyamat-irányítás

Egy munkafolyamat-irányító rendszer (Work-flow Management System, WFMS) „olyan számítógépes program, amely teljesen vagy részben biztosítja az üzleti folyamatok önműködővé tételét, miközben dokumentumok, információk vagy feladatok áramlanak az egyik résztvevőtől a másikig bizonyos eljárási szabályok szerinti művelet elvégzése céljából” (➲ http://www.e-workflow.org). A WFMS használata egy koordinátort feltételez, aki meghatározza a termék előállításához szükséges tevékenységek egymáshoz kapcsolódását. Az operátorokat az előállítási folyamaton vezetik végig, amelyből minden nem várt eltérés kiszűrésre kerül. Mindegyik művelet valamilyen adatot – méretadatot, megjegyzést, jelölést – állít elő, amelyek egy adatbázisba kerülnek. Ezt megelőzően gyártófolyamatunkhoz körülbelül két évig használtunk egy zárt összetevőkre épülő WFMS-rendszert, de ez nehézkesnek, lassúnak, túl erőforrás-igényesnek, a kényszerű megszakítások után nehezen újraindíthatónak és más eszközökkel csak fáradságos módon integrálhatónak bizonyult. Amikor a beérkező kaloriméter-alkatrészek emelkedő számával az összeszerelési sebesség már nem tudott lépést tartani, úgy döntöttünk, hogy nyílt forráskódra épülő összetevőkből fejlesztjük ki a saját megoldásunkat. Célunk az volt, hogy ezzel a lépéssel elkerüljük a korábbi hátrányokat, és a bejövő és kimenő adatok számára létrehozott felületek átlátszóságával biztosítsuk a rendszer kívánt rugalmasságát. A rendszer működtetéséhez a LAMP (Linux, Apache, MySQL, és Perl/PHP/Python) felületet választottuk. A LAMP minden összetevőjének megvan a maga elsődleges feladata: a Linux és az Apache biztosítja a szolgáltatások és a programozás alapvető hátterét, a MySQL képviseli WMFS-rendszerünk adatbázis-hátterét, a Perl/PHP pedig az operátorok és az adatbázis közti párbeszédet kezeli.

Redacle: az adatbázisterv

WFMS rendszerünk neve Redacle (Relational ECAL Database at Construction Level – relációs ECAL-adatbázis tervezési szakaszban). Az adatbázistervvel szemben támasztott elvárásaink részleteiben a következők voltak:

1. Nagyfokú rugalmasság: az adatbázis felépítése új termékek vagy tevékenységek hozzáadásakor nem változhat meg.
2. Minőségellenőrző (QC) adatok tárolásának a lehetősége: munkánk fontos részét képezi a minőségbiztosítás, az összegyűjtött adatoknak pedig statisztikai elemzés céljából bárki számára elérhetőnek kell lennie.
3. Az adatelérés sokfélesége: az adatbázisnak rendelkeznie kell a különböző eljárással – a héjból, programokból, parancsfájllal és a weben keresztül – történő lekérdezés lehetőségével.

Harmadik elvárásunknak a MySQL minden további nélkül megfelelt; egyszerűsége és a teljessége mellett ez volt az a tulajdonsága, ami miatt a LAMP alkalmazása mellett döntöttünk. Az első két igény kielégítéséhez egy sablon alapján egy táblakészletet fejlesztettünk ki, amely – az objektumközpontú programozáshoz hasonlóan – gyakori szabványos módja egy adott probléma megoldásának. Követett mintánk neve homomorfizmus, ami a sok-egy kapcsolat egyszerű megvalósítása. A gyakorlatban ez úgy működik, hogy az alkalmazott folyamat minden része rekordként az adatbázis két táblájában jelenik meg: az objektumtáblában és az objektummeghatározási táblában. Minden objektummeghatározás egy azonosítóval rendelkezik, ez a gyakorlatban a MySQL elsődleges kulcsértéke. Sok objektum használja ugyanazt az objektummeghatározást, a köztük lévő kapcsolatot az objektumtáblában tárolt idegen kulcs biztosítja, amely a megfelelő objektummeghatározás azonosítója.

2. kép Egy öt kristályból álló egység vizsgálata jellemzőinek meghatározása céljából. Az eszközöknek a mérési adatokat a munkafolyamatkezelő rendszernek kell továbbítaniuk

Talán egy példán keresztül bemutatva világosabb lesz a működés. Mint a bevezetésben említettem, kaloriméterünk nagyon sok különböző típusú alkotóelemből tevődik össze. Az egyes alkotórészfajtákból, mint amilyen például egy kristály, igen sokra van szükség, a kaloriméterhez például összesen mintegy 75 ezer kristályra. Az alkotórészek és meghatározásaik az adatbázis két külön táblájában kerülnek tárolásra, ahogy az *1.* és a *2. táblázatban* látható. Egy alkatrészfajta különböző példányai ugyanazt a meghatározást osztják meg a `partDefinition_id` oszlopban lévő megfelelő alkatrész-azonosítón keresztül. Ebben a két táblában a 33105000006306 számú alkatrész egy 1L típusú hengeres kristály, ahogy azt a ***partDefiniton*** táblában a `partDefinition_id` 195-ös értéke is mutatja.

Ennek a megközelítésnek az igazi előnye a rugalmasságában rejlik. Ha bármilyen okból kifolyólag új alkatrészek kerülnek a képbe, a Redacle adatbázis-szerkezete változatlan maradhat. Elegendő a meghatározások táblájában egy új rekordot létrehozni és az új elemekkel összekapcsolni. Sőt a Redacle adatbázisa még ennél is rugalmasabb: ha kaloriméterek helyett autókat gyártanánk, akkor is ugyanez lehetne az adatbázis-szerkezet.

A tevékenységek ugyanennek a megközelítésnek az alapján kerültek leképezésre: egy ***Activity*** nevű tábla tárolja az ***ActivityDescription*** táblában leírt rekordokat. Egy újabb tevékenység felvétele a munkafolyamatba nem áll többől, mint hogy a leírását elhelyezzük a meghatározások táblájában és összekapcsoljuk az ***Activity*** táblában lévő előfordulásokkal. Ugyanúgy igaz erre a szerkezetre is, hogy átszabás nélkül ráhúzható egy teljesen más üzleti tevékenységre is. Ha például levélkézbesítésről lenne szó, akkor a meghatározások táblájának rekordjaiban a levélfelvétel, a továbbítás és a kézbesítés helyén történő munkaműveletek leírását találnánk, míg az ***Activity*** tábla rekordjaiból azt tudhatnánk meg, hogy mikor és mely küldeményeken végezték el az adott műveletet.

A munkafolyamatot a munka során elvégzendő műveletek meghatározásai és e műveletek elvégzési sorrendje határozza meg. A felületet biztosító program ellenőrzi, hogy a munkafolyamat-meghatározásban az adott alkatrészen végrehajtott műveletet követő tevékenység a megadott időn belül befejeződik-e. A műveletek a csatlakozó programnak megfelelően megismételhetőek vagy átugorhatók.

A minőségellenőrzés adatai számára az elvonatkoztatás újabb szintjét bevezetve ugyanezt a homomorfikus mintát alkalmaztuk. Olyan tulajdonságadatokat határoztunk meg, amelyeket egy alkatrészen végzett adott művelet végzése során gyűjtünk össze. Az ezeket tartalmazó ***Characteristic*** tábla azonban nem tartalmaz valósi értékeket, mivel ezek különféle természetű adatok lehetnek: karakterláncok, számok vagy még összetettebb típusok. A ***Characteristic*** tábla egyszerűen a kulcsok tárolóhelye, amelyek közül az egyik a tulajdonságot a ***charDefinition*** táblában lévő meghatározásával kapcsolja össze. A tényleges tulajdonságok értékei különböző, a típusuknak megfelelő táblákban kerülnek tárolásra. A mi folyamatunk háromféle adattípussal dolgozik: egyszeres lebegőpontos számokkal, számhármasokkal és karakterláncokkal. Például egy kristály hossza egy egyszerű szám, és a ***Value*** táblában tárolódik. Néhány mérés a kristály tengelyének különböző pontjain történik, eltérő körülmények között. Az egyik ilyen az optikai terjedés, amelyet eltérő hullámhosszok mellett 2 centiméterenként mérünk. Az eredmények számhármast alkotnak: az első szám jelenti a pozíciót, a második a hullámhosszt, a harmadik pedig a hullámterjedés mértékét. Ezek a számhármasok képezik a ***multiValue*** tábla egy-egy rekordját. Ugyanez igaz a karakterláncokra is: az operátorok minden művelet előtt szemrevételezik a kristályokat, és megjegyzéseket hagyhatnak a lehetséges hibákra vonatkozóan. A *2–6.* táblázatban láthatók az imént említett, a jellemzők tárolására szolgáló táblák. A 33101000018045 rész a hossz és terjedés mérésére (TTO) vonatkozik. A hossz 229,7815 mm a ***Value*** táblában. A `char_id` mező 134821, amely a ***Characteristic*** táblában a `charDefinition_id=6` értékre mutat, amely a kristály hosszának felel meg. A TTO egy számhármasokból álló halmaz a ***multiValue*** táblában. Az adott kristály szemrevételezésének az eredménye a ***charValue*** táblában található nonhomogeneous (nem homogén) megjegyzés.

Ez az eljárás a Redacle-t megint csak másféle folyamatok kezelésére is alkalmassá teszi: egy tehenészetben például az előállító (karakterlánc) mellett jellemzésként használhatnánk az egyes tejadagok baktériumtartalmának rögzítésére. Továbbá a séma megzavarása nélkül adhatunk teljesen új adattípusokat (képeket vagy hangokat) az adatbázishoz – egyszerűen egy új tábla létrehozásával. Például, ha képeket szeretnénk hozzáadni, akkor ezt egy három mezőből álló tábla létrehozásával érhetjük el, amelyben a három mező az elsődleges kulcs, a kép adatai `BLOB` típusként és a ***Characteristic*** táblához kapcsolódást biztosító egész típusú azonosító. A MySQL-ben ezt a táblát az alábbi kóddal hozhatjuk létre:

```
CREATE TABLE picture (
        id INT NOT NULL AUTO_INCREMENT,
        data BLOB,
        char_id INT,
        INDEX (char_id),
        PRIMARY KEY (id)
);
```

1. táblázat *A Redacle adatbázisának Part táblája*

ID	partDefinition_id
33105000006306	195
33105000006307	196
33105000006308	197
33105000006309	198
33105000006310	196

2. táblázat *A charDefinition tábla*

Id	Description	Name	Unit	activityDef_id
2	result of visual inspection	VIS_I_OPER		2
6	crystal length	DL	mm	3
26	transversal transmission	TTO	mm#nm#%	4

A Redacle csatolófelületei

Programunkban az adatbázissal a felhasználók számos különböző módon kerülhetnek kapcsolatba: a MySQL-ügyfélprogramon keresztül, C++- és Java-programok segítségével, Perl-parancsfájlok vagy weboldalakon szereplő PHP-parancsfájlok révén. Számottevő előnyei vannak, ha a grafikus felhasználói felület (GUI) létrehozására böngészőprogramot használunk: ezzel a GUI hordozhatóvá válik és szükségtelenné teszi különleges összetevők telepítését, nem vesztegetjük az időnket a grafikus felület létrehozására, továbbá a böngészőprogram mind az operátorok, mind pedig az ügyfelek számára jól ismert.

A Redacle egy másik kiemelkedő tulajdonsága a más gépek felé irányuló csatolófelülete. A kaloriméter szerelésének folyamán önműködő szerkezetek végezték el a kristályokkal kapcsolatos méréseket, mindenfajta emberi beavatkozás nélkül (*2. kép*). Ezeknek a gépeknek tehát képesnek kell lenniük a Redacle rendszerrel való kapcsolattartásra, hogy a végrehajtandó műveletek megfelelő sorrendjére, a műveletek kezdeti és befejező időpontjára és a tulajdonságként tárolandó adatokra vonatkozó információkat kicserélhessék. Olyan rendszer létrehozása volt a célunk, amely szinte bármilyen eszköz számára lehetővé teszi, hogy a Redacle rendszerrel kapcsolatba lépjen. Szándékosan kerültük, hogy egy adott programozási nyelv hatása alá kerüljünk, vagy hogy programozói könyvtárakat biztosítsunk a szóba jöhető eszközökhöz, mivel ez amúgy sem állna arányban a beszerezhető eszközök mennyiségével, és beágyazott eszközök is léteznek zárt programokkal.

Ehelyett egy Instrument Agent (IA, eszközközvetítő) nevű démont fejlesztettünk ki, amely csatolófelületként működik a Redacle és az eszköz között. Az IA egy olyan munkafolyamat, amely egy internetes kapura csatlakozva onnan ASCII karaktereket képes olvasni, illetve ilyeneket tud oda küldeni. Így az eszköznek csak arra kell képesnek lennie, hogy a hálózatra csatlakozzon és karakterláncokat küldjön a kapcsolaton keresztül.

A műveletek sorrendje a következő:

- A csatlakozás után az eszköz bejelenti, hogy mely alkatrészen végez műveletet.
- Az IA lekérdezést hajt végre a Redacle adatbázisában és megkeresi a munkafolyamatban az adott alkatrészen legutóbb befejezett tevékenységet. Az eszköz által futtatandó parancs az adatbázis ***activityDefinition*** táblájának leírómezőjében található.
- Az IA elküldi az eszköznek a megfelelő parancsot.
- A parancs felismerése után az eszköz futtatja a parancsot, az IA pedig a nyugtázás után egy új tevékenységet szúr be a Redacle adatbázisba.
- A munkafolyamat befejeztével az IA frissíti az imént létrehozott tevékenységet, befejezettnek jelöli meg és XML formátumú karakterláncok formájában beolvassa az eszközről az adatokat.

A művelet eredménye `multiValue` vagy `charValue` mezőket egyaránt tartalmazhat. Az egyedül álló értékek formátuma a következő:

```
<RE><FI>mezőnév<VA>mezőérték</VA></FI>...</RE>
```

Az <RE> jelentése eredmény (result), az <FI> a mező (field), a <VA> pedig az érték (value) rövidítése. A mezőnévből az eszközközvetítő megszerzi a tulajdonság meghatározásának azonosítóját, és kitölti a mezőérték formátumának megfelelő tábla (`value` a számok és `charValue` a karakterláncok esetén) mezőjét. A ***multiValue*** tábla feltöltésére akkor kerül sor, ha az eredmény az alábbi formátumú:

```
<RE><NT>n-es név
<FI>mezőnév<VA>mezőérték</VA></FI>
...
</NT></RE>
```

Az `<NT>` ebben az esetben n-est jelöl, vagyis n számú elemből álló értéket.

3. táblázat *A Characteristics tábla*

ID	charDefinition_id	part_id	activity_id
106035	2	33101000018045	10660
134821	6	33101000018045	16093
135252	26	33101000018045	16182

4. táblázat *A Value tábla*

ID	x	char_id
37614	229.7815	134821

5. táblázat *A multiValue tábla*

ID	x	y	z	char_id
748867	15	700	76.1	135252
748907	35	700	75.7	135252
748947	55	700	75.9	135252
748987	75	700	76.1	135252
749027	95	700	76	135252
749067	115	700	75.5	135252
749107	135	700	76	135252
749147	155	700	75.7	135252
749187	175	700	76.3	135252
749227	195	700	76	135252
749267	215	700	74.6	135252

6. táblázat *A charValue tábla*

ID	value	char_id
2872	nonhomogeneous	106035

Az eszköz programfejlesztőinek nem kell ismerniük a Redacle adatbázisának részleteit, mindössze a használt karakterlánc-formátumokról kell tudomást szerezniük. Nincsenek előírt programkönyvtárak, sem csatolandó fájlok; a programozási nyelv sem mérvadó. Az egyetlen előfeltétel, hogy képes legyen hálózatos kapcsolatot létesíteni az eszközközvetítővel.

A felhasználói és eszközfelület mellett a koordinátorok és felügyelők számára egy parancssoros parancsfájlgyűjteményt fejlesztettünk ki, továbbá egy kis programkönyvtárat hoztunk létre az adatbázis feletti C++- és Perl-parancsfájlok futtatására, amely feleslegessé teszi az SQL-lekérdezések szerkesztését.

Tapasztalatok és távlatok

A Redacle rendszert laboratóriumunk négy hónappal az első projektmegbeszélés után bocsátotta ki. A teljes rendszer körülbelül tízezer Perl, C++, PHP és Java nyelven megírt programsort tartalmaz. A program futtatásához szükséges gép a korábbi rendszer erőforrásigényével összehasonlítva kicsinek mondható. Az előtte használt rendszerrel a kétprocesszoros 800 MHz-es Pentium kiszolgálógép szinte állandó százszázalékos processzorterheléssel futott a teljes 515 MB RAM foglaltsága mellett. Szükség volt az ügyfélgépek továbbfejlesztésére is, a Java GUI támogatása céljából megkettőztük a memóriájukat. Az előző rendszer teljesítményét növelendő a tervezett gép egy kétprocesszoros, 1 GHz-es Pentium III processzorokat magában foglaló, 1 GB memóriával rendelkező gép volt. A Redacle használatával ez bőven elégnek bizonyult, a processzorterhelés elhanyagolható, a felhasznált memória pedig 140–200 MB.

Nyilvánvalóvá vált, hogy olyan eszközökre van szükségünk, amelyekkel az előző, más laboratóriumban még mindig használt adatbázisból adatokat importálhatunk, illetve abba exportálhatunk. Ezeket az XML-fájlokat író és olvasó eszközöket Perlben gyorsan létrehoztuk, és egy nap alatt képesek voltunk minden régi adatot átemelni a Redacle rendszerbe.

Pillanatnyilag mintegy 13 ezer alkatrészt tárolunk az adatbázisban. A tárolt tulajdonságok száma 97 ezer, amelyek közül az 50 MB-os teljes adatbázisban bármelyik különböző értékekből állhat. A 15 tábla közül a ***multiValue*** tábla a legnagyobb, több mint egymillió rekordot tartalmaz és 41 MB méretű.

A leglátványosabb eredményt azonban az operátoroknak a kaloriméter szerelésével töltött idejének terén értük el. A Redacle bevezetése előtt az operátorok idejük egynegyedét a munkafolyamat-kezelő rendszerrel való párbeszéddel vesztegették el. A Redacle használatával az operátorok és az adatbázis közti párbeszédre fordított idő mennyisége elhanyagolható, ami nagymértékben növeli az érzékelő-összeszerelés hatékonyságát.

Mi több, az operátorok rövid időn belül hozzászoktak a Redacle rugalmasságához és sorra jelezni kezdték az új eszközök és csatolófelületek iránti igényüket. Aminek a kifejlesztéséhez korábban hetekre volt szükség vagy szinte lehetetlen volt megépíteni, a Redacle és a LAMP használatával most rövid időn belül (néhány órától egy-két napig terjedő idő alatt) megvalósíthatóvá vált.

A Redacle adatbázisának rendkívüli rugalmassága sok más üzleti folyamatban vagy ipari területen való felhasználásra is alkalmassá teszi, világossá téve, hogy a nyílt forrású programok mennyivel jobban használhatóak lehetnek zárt kódú megfelelőiknél. Jövőbeli terveink között még összetettebb munkafolyamatmodellek kidolgozása és a gyakran használt lekérdezésekből és függvényekből egy programkönyvtár összeállítása szerepel, amelyet más Redacle-alapú projektekben lehetne felhasználni.

Linux Journal 2004. február 118. szám

Giovanni Organtini (G.Organtini@roma1.infn.it)
A Római Egyetem fizikusainak tanít számításelméletet és programozást.

Luciano M. Barone (Luciano.Barone@roma1.infn.it) A római La Sapienza Egyetem fizikai karának docense.