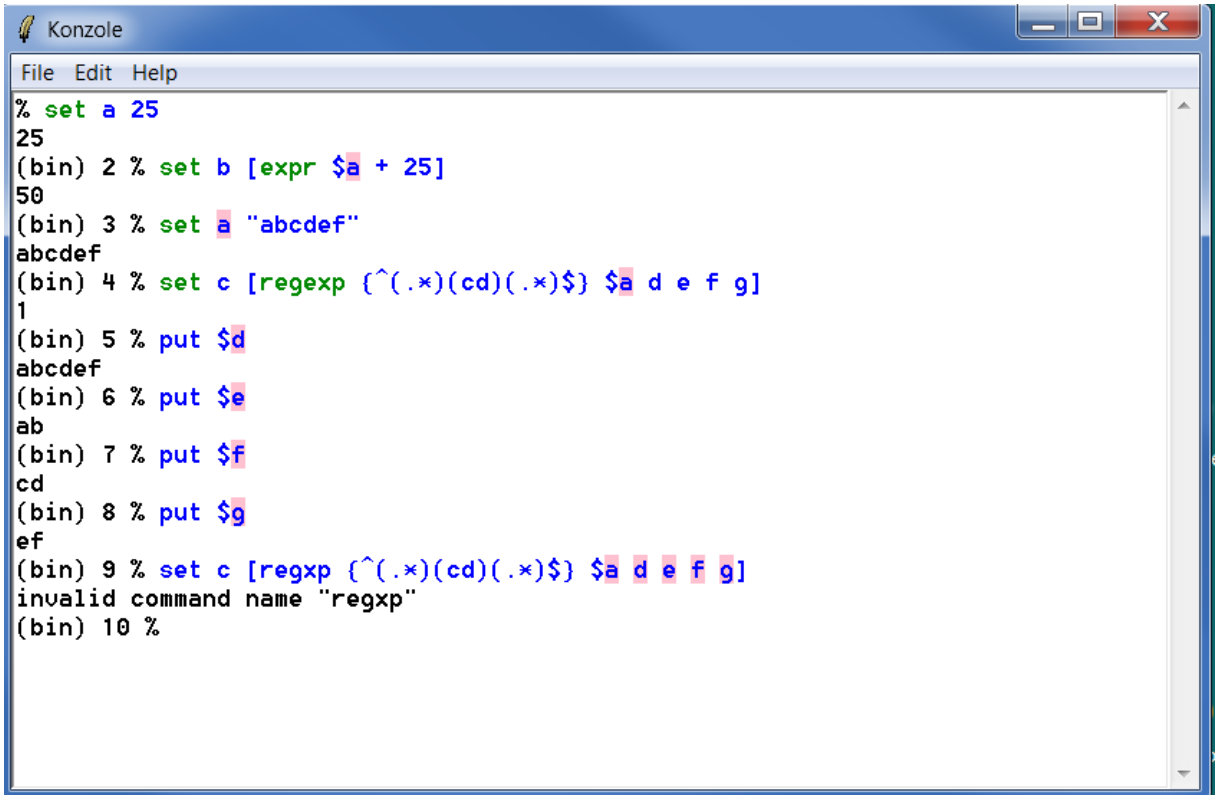
Obsah

Vývojová prostředí - od pazourku k sofistikovaným nástrojům

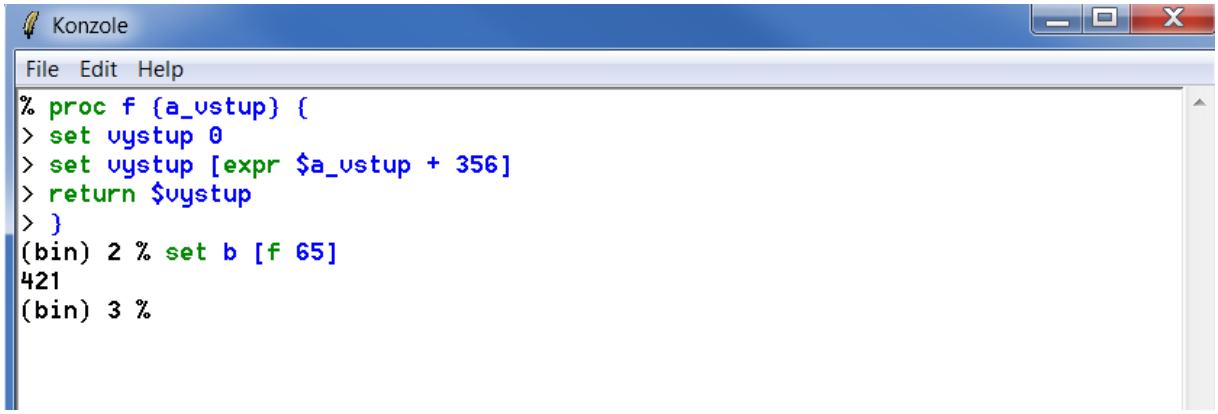
Na následujících nástrojích si ukážeme postupných přechod od nejjednodušších nástrojů ovládaných pouze klávesnicí až k nástrojům uplatňujícím všechny vymoženosti grafického prostředí - ovládání myší, kontextové menu apod.

WishTCL

Jednoduchá pracovní konzole skriptovací jazyka TCL. Bez prostředků pro kompilaci a ladění kódu. Je v ní prakticky možné pouze ověřit správnost syntaxe a vyhodnocení jednoduchých výrazů, příkazů a sekvencí příkazů a funkcí.



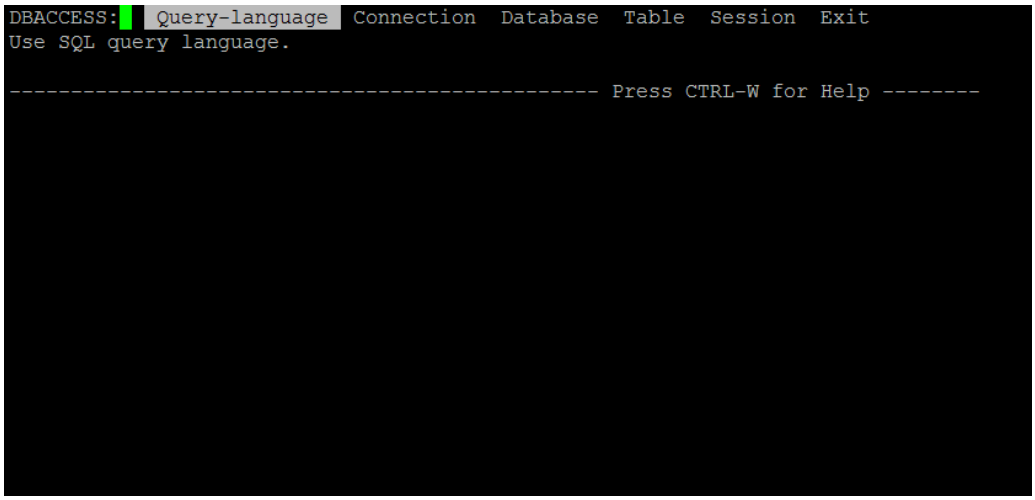
použití funkce



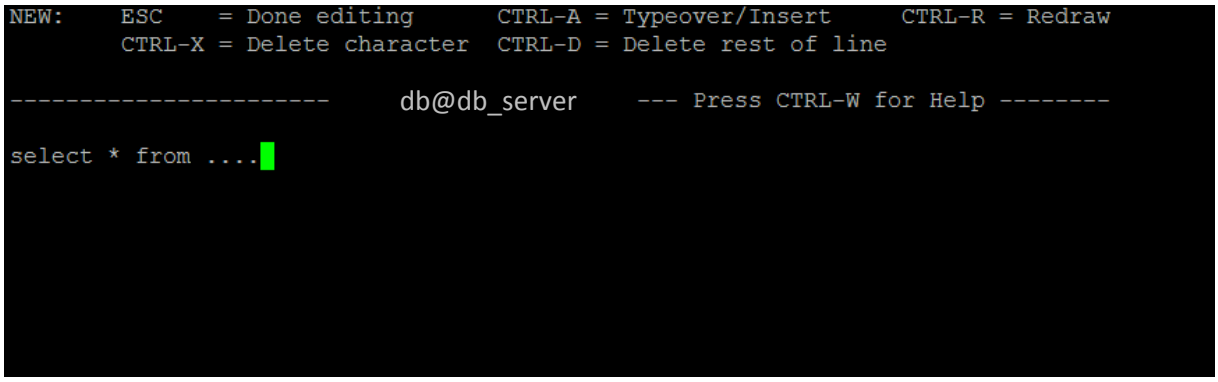
Prostředí pro práci v databázích

Informix - dbaccess

Dostupné z příkazové řádky operačního systému UNIX, z prostředí MS Windows přes " černé" okno - např. putty. Ovládání přes klávesnici.



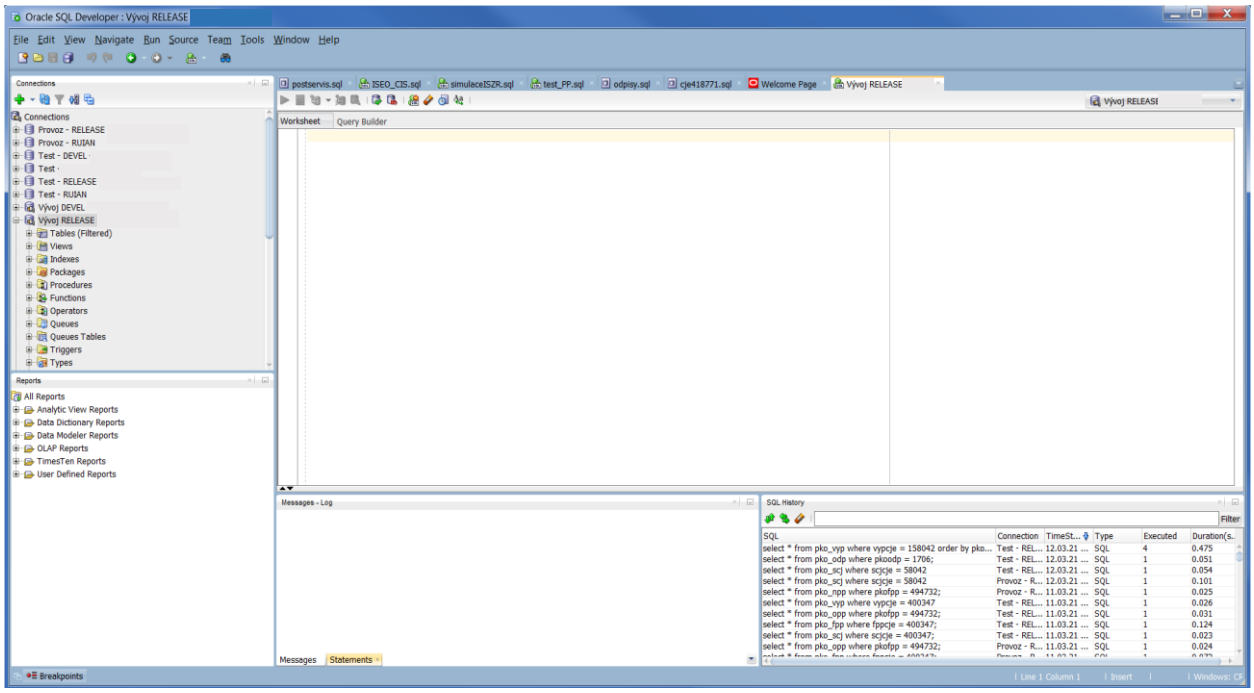
Po výběru databáze je možno provádět konkrétní sql příkazy. Mimo interního editoru se dá použít i externí editor vi, Je nutné znát jeho příkazy. Seznam příkazů lze načíst i uložit do souboru. Dále je možno pracovat s tabulkami - zakládat nové, upravovat stávající nebo je dropnout. Současně je možné získat a upravit další informace jako indexy, přístupová práva, závislosti a omezení (constraints), triggery, rozmístní tabulek a indexů ve fragmentech. V rámci editoru je možné přepnout na jinou konexi nebo server. Ovládání tohoto nástroje je možné pomocí šipek nebo horkými klávesami (velká písmena v jednotlivých příkazech).



Při editaci sql příkazů je zobrazena krátká nápověda, příp. lze pomocí klávesové zkratky Ctrl+W vyvolat nápovědu kdekoli v dbaccesu.

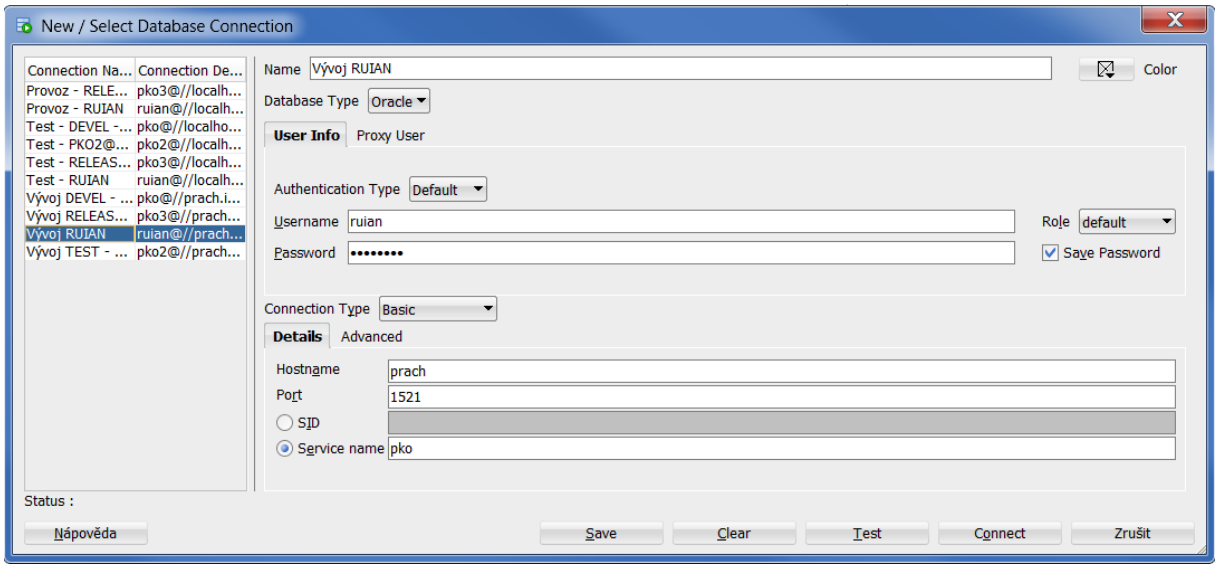
Oracle SQL developer

Klasická MS Windows aplikace s klasickým ovládáním přes klávesnici či myš.

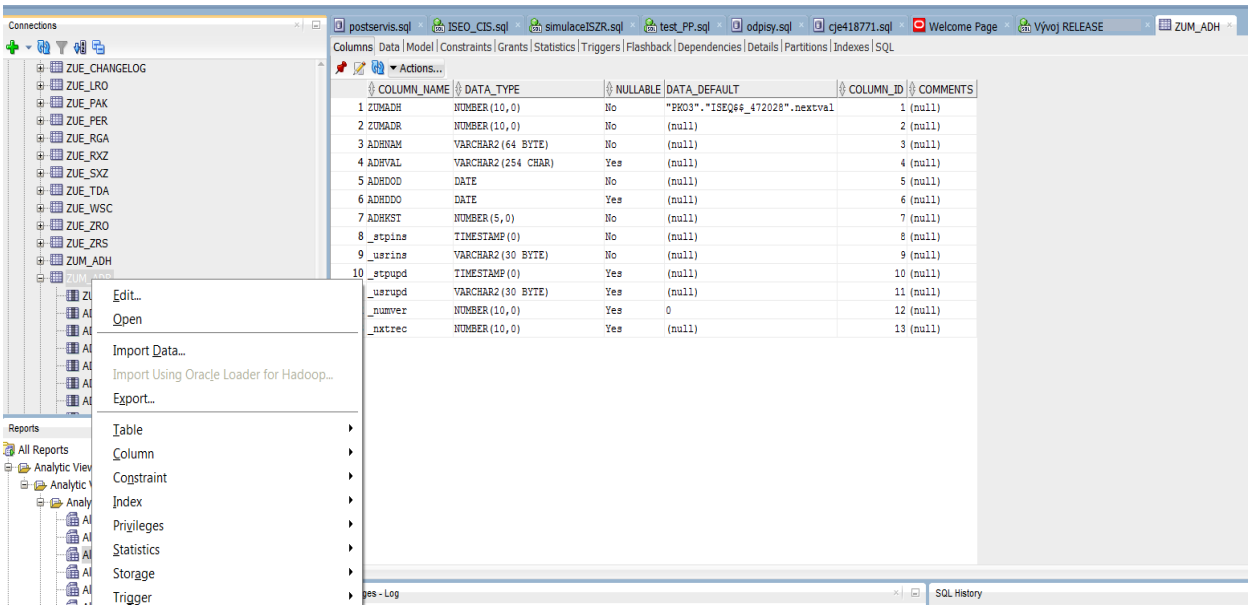


Aplikace je rozdělena na oblast pro připojení k jednotlivým databázím, nabídku předpřipravených reportů, vlastní pracovní plochu, oblast pro zobrazení zpráv a logů a oblast pro historii sql příkazů.

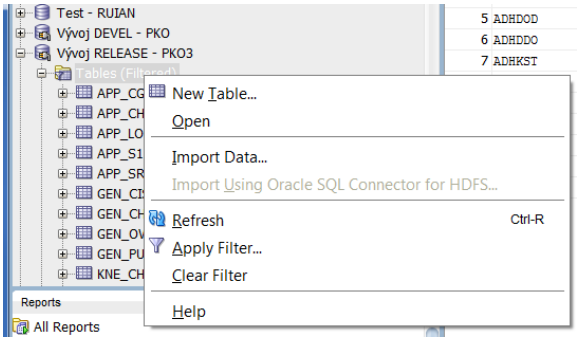
Jednotlivá připojení databází lze přidávat měnit nebo kopírovat.



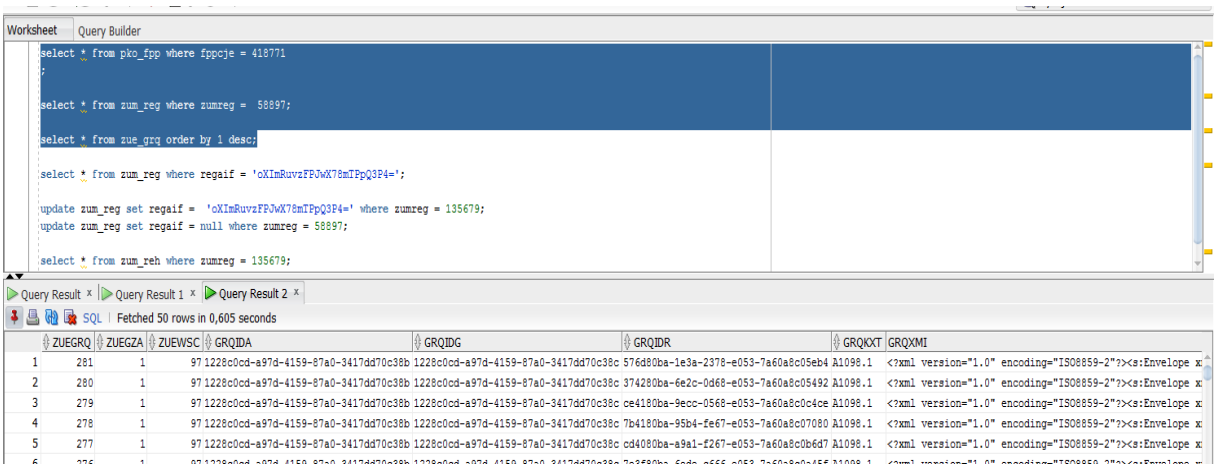
Pro jednotlivé připojené databáze lze vypsat seznam definovaných tabulek, zobrazení a indexů. Dále pak i definované procedury a funkce, operátory, triggery, materializovaná zobrazení a jejich logy a další možnosti, které Oracle umožňuje. V detailu vybrané možnosti jsou pak umožněny další akce, např. u tabulky lze vypsat její jednotlivé sloupce, datový obsah, omezení, přístupová práva, statistiku, triggery, závislosti, indexy a další. Definice nových položek či úprava stávajících je dostupná buď přes kontextové menu nebo přes volbu Actions.



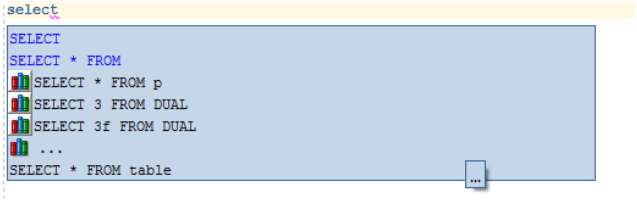
Definice nové tabulky je dostupná z kontextového menu v rámci seznamu tabulek.



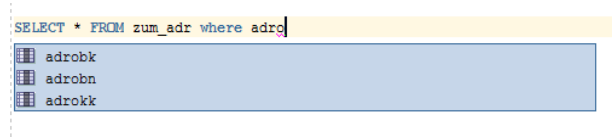
V rámci pracovní oblasti pak můžeme mít otevřeno více oken/záložek. Každé z těchto oken můžeme mít připojeno k jiné otevřené databázi. V jednotlivých oknech můžeme provádět jednotlivé sql příkazy nebo sekvence příkazů. Vždy je aplikován buď příkaz z aktuální řádky nebo vybraný blok příkazů. V případě volání sekvence příkazů je jejich výstup směrován do samostatné záložky.



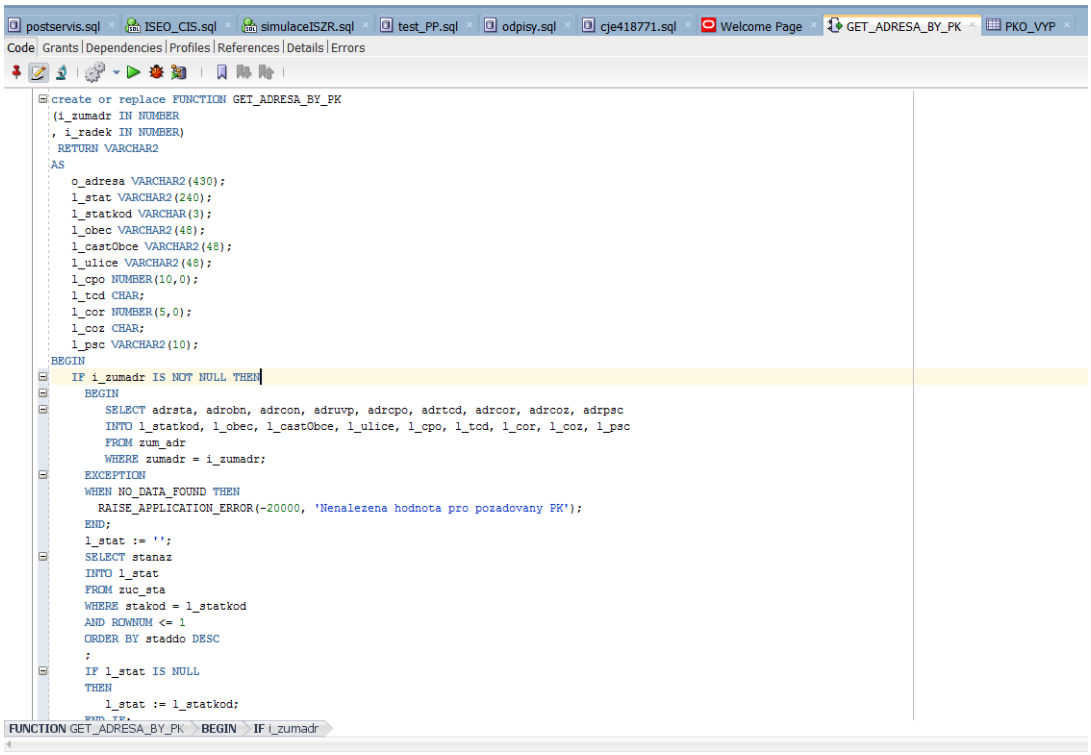
Při psaní jednotlivých sql příkazů nám aplikace nabízí možnosti odpovídající sql příkazům



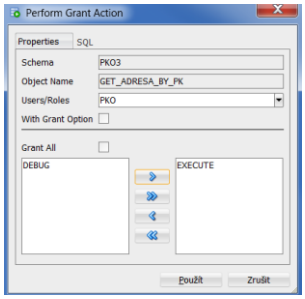
nebo rovnou sloupce z tabulky, kterou v rámci příkazu prohledáváme.



Oracle SQL developer umožňuje nejen pracovat s databází, jak jsme zvyklí. Umožní nám také programování v interním jazyce Oracle. V rámci prostředí pak máme prostředky pro kompilaci a ladění kódu.



Po odladění a kompilaci funkce je nejprve nutné přidělit práva jednotlivým uživatelům nebo rolím, které tuto funkci budou využívat.



Takto definovanou a zkompilovanou funkci pak můžeme použít v rámci sql příkazu

```
select a.adrobn, get_adresa_by_pk(zumadr,0) as celaAdresa from zum_adr a where zumadr = 1000;
```

Query Result

SQL | All Rows Fetched: 1 in 0,111 seconds

	ADROBN	CELAADRESA
1	Praha	Nad lomem 1352/36, 147 70, Praha - Braník

Nebo v rámci některého triggeru na základě, kterého se nám doplní data tabulky při změně určitého sloupce.

	TRIGGER_NAME	TRIGGER_TYPE	TRIGGER_OWNER	TRIGGERING_EVENT	STATUS	TABLE_NAME
1	TR_PKO_VYP_PLAKON_UPDATE	BEFORE EACH ROW	PKO3	UPDATE	ENABLED	PKO_VYP
2	TR_PKO_VYP_INS	BEFORE EACH ROW	PKO3	INSERT	ENABLED	PKO_VYP
3	TR_PKO_VYP_VYPDEG_UPDATE	AFTER EACH ROW	PKO3	UPDATE	ENABLED	PKO_VYP

Refresh: 0

```
            left outer join pkr_pxs x on x.pkopla = f.pkopla and x.pxskst = 1 and x.pxsddo is null
            left outer join zum_sbj s on x.zumsbj = s.zumsbj
        where pkofpp = get_last_fpp(:n.vypcje);
        select get_adresa_by_pk(zumadr, 0) into l_fpppad
            from pko_fpp f
            left outer join pkr_pxs x on x.pkopla = f.pkopla and x.pxskst = 1 and x.pxsddo is null
            left outer join zum_sbj s on x.zumsbj = s.zumsbj
            left outer join zum_sid b on s.zumsbj = b.zumsbj and sidtyp = 'S' and sidkst = 1 and sidddo is null
        where pkofpp = get_last_fpp(:n.vypcje);
        end;
    end if;
    -- zkraceno jmeno
    :n.vyppzk := l_fpppzj ;
    -- adresa
    :n.vyppad := l_fpppad;
    -- pro vypocteny vypocet naplnime kontakt
    -- zkraceny nazev
    select case when zumrez is not null then regzkl else sbjzkl end into l_fppznp
            from pko_fpp f
            left outer join zum_reg r on f.zumrez = r.zumreg
            left outer join zum_sbj s on f.zumsbz = s.zumsbj
    where pkofpp = get_last_fpp(:n.vypcje);
    :n.vypzzk := l_fppznp;
    -- adresa
    select get_adresa_by_pk(zumadd,0) into l_fppzad from pko_fpp where pkofpp = get_last_fpp(:n.vypcje);
    :n.vypzad := l_fppzad;
  end if;
END;
```

Vlastní aplikace nabízí další sadu funkcí a nástrojů nejen pro definici a správu jednotlivých databází, ale také pro monitorování a správu provozu na jednotlivých databázích. Ale také nástroje pro týmovou práci jako je připojení verzovacího systému GIT a práci s jednotlivým verzemi databázového schématu.