mu zaručilo přízeň konzervativních zákazníků – přechod z relačního DBMS na objektově relační systém řízení báze dat (ORDBMS) nevyžaduje velké změny ani investice. Naopak použití OR přístupu může usnadnit vývoj aplikací i v těch oblastech, kde nejsou persistentní objekty nezbytně vyžadovány, protože reference mezi objekty výrazně usnadňují dotazování. V neposlední řadě je mapování objektových modelovacích technik (UML) výrazně přirozenější, než metody převodu diagramu tříd na relační model.

Koncem 90. let OO a OR přístup v databázových systémech do značné míry konvergoval. Například jazyk OQL je značně podobný SQL:1999. Podstatný rozdíl je jen v přístupu k objektovým datům:

- OODBMS lze chápat jako rozšíření programovacího jazyka – perzistentní objekty jsou vytvořeny v prostředí objektového jazyka (Java, C++, Smalltalk) pomocí tříd definovaných v těchto jazycích a manipulovány navigačním způsobem. Jsou tedy silně provázané (pouze objektové paradigma) a flexibilní pro aplikace na straně klienta – tyto aplikace dočasně uchovávají všechna potřebná data.
- ORDBMS oproti tomu poskytují pouze aplikační rozhraní, založené na SQL, pro manipulaci s daty, jejichž definice typů (tříd) musí být DBMS známa předem. To poskytuje možnost umístění aplikace i na straně serveru se vším co s tím souvisí, jako je možnost velkého množství současně probíhajících nezávislých dotazů a jejich případná optimalizace.

Objektových rozšíření relačních databází vzniklo mnoho, proto se podíváme na standardy, které jejich vývoj usměrňují.

3.2.1 SQL:1999

Původním záměrem SQL:1999 bylo modifikovat SQL-92 tak, aby se z SQL stal jazyk objektově relačních databází [Eis99]. Sedmiletý vývoj ale celý dokument restrukturalizoval a přidal několik základních vylepšení. K těm patří nové datové typy:

- Large Object (LOB) v binární (BLOB) a znakové podobě (CLOB). Tyto typy jsou určeny pro zpracování velkého množství dat na straně klienta, proto má jejich využití v dotazech několik omezení oproti typům z SQL-92.
- Booleovský typ (BOOLEAN) s hodnotami true, false a unknown.
- Dva složené datové typy (ARRAY, ROW). Pole je indexovaná multimnožina prvků stejného typu (konečná kolekce). Složený atribut ROW lze chápat jako běžný řádek relační tabulky, proto neporušuje 1. normální formu. Pro jeho zpřístupnění se používá tečková notace, jako `Employees.Name.title = 'Ing.'` v následujícím příkladě.

Ex. 3.5 Alternativní definice tabulky užitím ROW dle SQL:1999

```
CREATE TABLE Employees (
   empID   INTEGER PRIMARY KEY,
   Name ROW (
      title  VARCHAR(5),
      first  VARCHAR(30),
      mid    VARCHAR(30),
      second VARCHAR(30),
      scitle VARCHAR(5),
   ),
   freeDays INTEGER ARRAY[7],
   inOffice BOOLEAN,
   photo    BLOB
);
```

SQL:1999 přidává několik predikátů (např. `Employees.Name.title LIKE`[17] `'%Prof%'`), zavádí rekurzivní dotazy, triggery, uživatelské role a vylepšuje transakční zpracování (ROLLBACK TO SAVEPOINT). Nejdůležitější je ale objektově orientované rozšíření.

Informace o objektech nesou v ORDBMS strukturované datové typy (User-Defined Types, UDT). Ty mají následující vlastnosti:

- Skládají se z jednoho nebo více typů, jak vestavěných (INTEGER, ARRAY), tak dále strukturovaných. Ty se mohou libovolně zanořovat. Viditelnost atributů (private, public) není v SQL:1999 řešena.
- Jsou organizovány do hierarchií. Více specializované podtypy dědí z nadtypů všechny atributy i metody a mohou být doplněny o další. Jedná se o jednoduchou dědičnost. Je ale nutné explicitně zadat, že typ může mít podtyp (NOT FINAL). Také je možné vytvořit ekvivalent abstraktní třídy (NOT INSTANTIABLE).
- Atributy jsou zapouzdřeny pomocí systémem generovaných funkcí pro čtení a zápis (set, get). Tyto funkce nemohou být přetíženy (overloading), všechny ostatní ano.
- Jejich chování je zprostředkováno pomocí metod a funkcí. Metody jsou, na rozdíl od funkcí, vázány k danému typu. Funkce mohou být přetíženy, výběr se provádí v čase kompilace. Metody mohou být polymorfní, jsou vybírány dynamicky za běhu programu (late binding).
- Porovnání hodnot objektů se děje výhradně pomocí uživatelských funkcí.

Ex. 3.6 Definice typu zaměstnance odvozeného z typu obecné osoby a typované tabulky

```
CREATE TYPE tEmployee
   UNDER tPerson
   AS ( empID   INTEGER,
        manager REF(tEmployee),
        salary  REAL
        photo   tImage )
   INSTANTIABLE
   NOT FINAL
   REF ( empID )
   INSTANCE METHOD
      moreMoney ( amount REAL )
      RETURNS REAL
);
-- typed table Employees
CREATE TABLE Employees OF tEmployee;
```

Přístup k atributům a funkcím do strukturovaných typů se provádí pomocí tečkové notace (`Employees.salary > 10000`). Přístup k atributům je možný i funkční notací.

Reprezentace třídy (například v Javě) je možné vytvořit několika způsoby:

- Libovolný objekt se po standardní serializaci vlepí do BLOBu.
- Jednoduchý objekt je reprezentován řádem tabulky s vestavěnými typy.
- Objekt tvoří řádek s vestavěnými i strukturovanými typy, případně jediným UDT.
- Typovanou tabulkou (viz Ex. 3.6). Každému řádku je tedy přiřazen jeden objekt. Takovýto objekt je možné jednoznačně identifikovat elementem, který se chová jako OID. Jedná se o speciální typ reference (REF). Jeho použití je jednoduché: `Employees.manager->photo`.

[17] Je možné použít také SIMILAR s regulárními výrazy