

## Object-Relational Databases

### *Evolution of Database Systems*

<b>Graph-Based</b>	<b>Relational</b>	<b>Object-Relational</b>
1960's-1970's	1980's-1990's	1990's

- Pure object-oriented database systems have failed to capture the database market !!
- The object-relational approach is an evolutionary approach

[Home Page](#)

[Title Page](#)



Page 2 of 9

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

## Why is the relational model so successful

- Data independence
- High level query language - SQL
- Query optimisation
- Support for integrity constraints
- Well-understood database design
- Transaction management and concurrency

[Home Page](#)

[Title Page](#)



[Page 3 of 9](#)

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

## Why do we need Object-Oriented Databases

- For some applications 1NF is too strict.
  - ◇ Document management and web site management
  - ◇ Geographic and Statistical database management
  - ◇ Biological data management
- SQL is not a complete programming language such as C++ and Java -  
cannot implement abstract data types

[Home Page](#)

[Title Page](#)

◀▶

◀▶

Page 4 of 9

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

## Object-Oriented Approach

- Couple an Object-Oriented programming language with a DBMS
- ◇ One approach is to implement on top of a relational DBMS.
- Object-Oriented databases failed to deliver robust and versatile systems which can compete with relational database technology
- Object-Relational approach - integrate Object-Oriented features into existing relational DBMSs

## Object-Relational Evolution

- *Object identity* - the property that distinguishes between objects

Object identity should be

1) Location independent - e.g. physical addresses are *not* location independent

2) Value independent -

e.g. relational keys are *not* value independent  
(an object exists independently of its values)

- Object identity allows data sharing

e.g. a couple having the same children

◇ *Surrogates* are system-generated unique identifiers which are location independent

## Simulate Object Identity via Sequences

```
CREATE sequence S#-SEQ  
START WITH 1  
INCREMENT BY 1  
NOCYCLE
```

```
INSERT INTO suppliers values  
(S#-SEQ.nextval, 'Bloggs', 20, 'London')
```

```
DROP sequence S#-SEQ
```

```
ALTER sequence S#-SEQ  
MAXVALUE 1000
```

## Complex Objects

- An object which may violate 1NF is called a *complex object*

```
CREATE TABLE person (  
    PID#      number(5),  
    PNAME    varchar(20) not null,  
    PDOB     date,  
    PADDRESS address,  
    CHILDRENSET OF(children),  
    primary key(PID#))
```

[Home Page](#)[Title Page](#)[◀▶](#)[◀ ▶](#)[Page 8 of 9](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

```
CREATE TABLE address (  
    HOUSE#    number(4),  
    STREET    varchar(20),  
    POSTCODE  char(8),  
    CITY      varchar(15),  
    primary key(HOUSE#, STREET))
```

```
CREATE TABLE child (  
    CID#    number(5),  
    CNAME  varchar(20) not null,  
    CDOB   date,  
    primary key(CID#))
```

## Inheritance

- Is a natural way of organising knowledge
- Economical - we can reuse objects

```
CREATE TABLE employee UNDER person (  
    SAL          number(5),  
    WORKS_FOR company)
```

```
CREATE TABLE student UNDER person (  
    DEGREE  varchar(10),  
    DNAME   varchar(20),  
    PLACE   college)
```