NoSQL

Not Only SQL

Nierelacyjne bazy danych

**CECHY PODSTAWOWE:**

NIE UŻYWAJĄ ZŁĄCZEŃ ORAZ SZTYWNYCH SCHEMATÓW DANYCH

DOBRA SKALOWALNOŚĆ – ma być przezroczyste dla warstwy aplikacji

NIEKONIECZNIE SPEŁNIA ZASADY ACID - (Atomicity, Consistency, Isolation, Durability) – zmniejsza restrykcyjność w działaniu bazy nie gwarantując całkowitego bezpieczeństwa danych, a w szczególności transakcji.

SŁUŻĄ DO PRZECHOWYWANIA DUŻYCH ILOŚCI DANYCH

WYSOKI POZIOM DOSTĘPNOŚCI – zakładamy że pojedyncze awarie to nie wyjątek a normalna sytuacja. Całośc ma wciąż działać nawet jeśli część elementów jest uszkodzona

KONCENTRACJA NA SAMYCH DANYCH A NIE NA SCHEMATACH

SĄ MNIEJ UNIWERSALNE OD TRADYCYJNYCH BAZ SQL, ALE W NIEKTÓRYCH ZASTOSOWANIACH OFERUJĄ ZNACZNIE WIĘKSZĄ WYDAJNOŚĆ

ROZPROSZONA ARCHITEKTURA – możliwość dodania kolejnych serwerów co powoduje wzrost tego niezawodności nie wydajności na początku wydajności (skalowalność) oraz niezawodności w nawiasie redundancja

**FORMATY PRZECHOWYWANIA DANCYH:**

DANE ZAZWYCZAJ PRZECHOWYWANE SĄ W POSTACI PAR KLUCZ-WARTOŚĆ

NAJCZĘŚCIEJ STOSOWANE FORMATY PRZECHOWYWANIA TO:

-XML

-JSON

-DANE BINARNE

**PRZYKŁADOWE TYPY BAZ NOSQL**

-MongoDB

-Cassandra – wykorzystywana m.in. przez facebook(ZUO) przechowuje ogromne ilości danych

-BigTable (Google)

-Amazon SimpleDB

**Modele baz**

🡪Bazy kolumnowe – informacje nie są zawarte w wierszach tylko w kolumnach. Takie podejście jest dobre przy agregacji dużych ilości danych(np.hurtownie, systemy analityczne)

🡪Bazy dokumentowe - W bazach tego typu zamiast wierszy występuje pojęcie dokumentu, który zawiera pary klucz-wartość. Podejście takie jest niezwykle elastyczne i umożliwia bardzo wierne odtwarzanie rzeczywistych danych w systemach informatycznych.

🡪Bazy XML – dobre rozwiązanie do wymiany informacji między bazami danych. Na wskutek tego powstały bazy przechowujące dane w tymże języku. Rozwiązania te to np. Apache Xindice, MarkLogic Server, Oracle Berkeley XML DB, eXist.

🡪Bazy obiektowe – Służa do przechowywania obiektów a nawet ich struktur, dzięki czemu programiści mogą uniknąć tzw. Mapowania relacyjno-obiektowego.Pomimo swoich zalet nie znalazły miejsca na rynku.Główną przyczyną jest zbyt mała wydajność, brak optymalizacji, oraz brak możliwości wymiany danych między językami