# NULL

* SQL-Standard definiert NULL nicht als Wert, sondern als Platzhalter
* Bei der Oracle Datenbank kann ein leerer String NULL sein

# NULL Indizieren

* Jeder Index ist ein partieller Index
* Zeilen werden in ein mehrspaltigen Index aufgenommen, wenn zumindest eine Indexspalte nicht NULL ist
* Funktionsbasierte Indexes
* Mythos: Oracle Datenbank kann nicht NULL indizieren

# Anti-Patterns

* Anti-Patterns in SQL sind bewährte Praktiken oder Lösungsansätze, die vermieden werden sollten, da sie ineffizient, fehleranfällig oder schwerwartbar sind.
* Sie können zu schlechter Leistung, schwieriger Wartung oder sogar Datenproblemen führen

# Date-Types

* Eine der häufigsten Verschleierung betrifft Datums-Spalten. Die Oracle Datenbank ist dafür besonders anfällig, weil sie nur den DATE-Typen hat, die immer eine Uhrzeit mitführen
* Um den Zeitanteil aus einer DATE-Spalte zu entfernen, hat sich die TRUNC-Funktion durchgesetzt
* Ein Bild, das Text, Schrift, weiß, Screenshot enthält.

  Automatisch generierte Beschreibung
* Die Abfrage ist absolut korrekt, kann aber einen Index auf SALE\_DATE nicht ordentlich nutzen
* Generische Lösung muss man die Bedingung als explizite Bereichsbedingung umformulieren
* Ein Bild, das Text, Schrift, weiß, Quittung enthält.

  Automatisch generierte Beschreibung
* Eine andere häufige Verschleierung ist, das Datum als Text zu vergleichen
* Ein Bild, das Text, Schrift, weiß, Screenshot enthält.

  Automatisch generierte Beschreibung
* Das Problem ist wieder die Konvertierung der Spalte SALE\_DATE. Solche Bedingungen entstehen oft im Glauben, dass man einer Datenbank nur Zahlen und Texte übergeben kann.
* Mit Bind-Parametern kann man aber auch andere Daten-Typen verwenden
* Falls das nicht möglich ist, sollte man nicht die Tabellenspalte, sondern den Suchbegriff konvertieren
* Ein Bild, das Text, Schrift, weiß, Screenshot enthält.

  Automatisch generierte Beschreibung
* Die folgende Verschleierung ist besonders tückisch:
* Ein Bild, das Text, Schrift, weiß, Design enthält.

  Automatisch generierte Beschreibung
* Diese Bedingung scheint auf den ersten Blick nicht verschleiert zu sein, weil sie keine Funktion auf der Tabellenspalte verwendet.
* Durch die Verwendung des LIKE-Operators wird aber ein String Vergleich erzwungen
* Der Predicate-Information-Bereich des Ausführungsplanes zeigt, was die Oracle Datenbank macht
* Ein Bild, das Text, Schrift, weiß, Typografie enthält.

  Automatisch generierte Beschreibung

# Numerische Strings

* Numerische Strings sind Zahlen, die in Text-Feldern gespeichert werden
* Es ist also dasselbe Problem wie zuvor. Durch die Funktion kann ein Index auf NUMERIC\_STRING nicht sinnvoll genutzt werden
* Die Lösung ist wieder dieselbe: Anstatt den Spaltentypen an den Suchbegriff anzupassen, passt man den Suchbegriff an den Spaltentypen an
* Ein Bild, das Text, Schrift, weiß, Screenshot enthält.

  Automatisch generierte Beschreibung

# Zusammenfügen von Spalten

* Wenn man vor diesem Problem steht, hat man aber nur in den seltensten Fällen die Möglichkeit die Tabelle umzubauen
* Manchmal muss man Bedingungen gezielt verschleiern, damit sie nicht als Zugriffsprädikat verwendet werden.
* Ein Bild, das Text, Schrift, weiß, Quittung enthält.

  Automatisch generierte Beschreibung
* Angenommen es gibt sowohl einen Index auf SUBSIDIARY\_ID als auch auf LAST\_NAME. Welcher ist für diese Abfrage besser?
* Ohne zu wissen, wo die Wildcard-Zeichen im Suchbegriff stehen, kann man keine qualifizierte Antwort geben
* Auch die Datenbank kann nur raten. Wenn man aber weiß, dass der Suchbegriff immer mit einem Wildcard Zeichen beginnt, kann man die entsprechende Bedingung absichtlich verschleiern.
* Dadurch kann der LIKE-Filter nicht als Zugriffsprädikat verwendet werden
* Ein Bild, das Text, Schrift, weiß, Quittung enthält.

  Automatisch generierte Beschreibung
* Dafür genügt es, einen leeren String an die Spalte LAST\_NAME anzuhängen. Dieses Vorgehen ist allerdings nur der letzte Ausweg, wenn es keine anderen Möglichkeiten mehr gibt

# Schlaue Logiken

* SQL-Datenbanken unterstützen Ad-hoc-Abfragen, die zur Laufzeit analysiert und optimiert werden.
* Dynamische Abfragen sind also sinnvoll und können durch die Verwendung von Bind-Parametern effizient gestaltet werden.
* Es gibt jedoch eine verbreitete Praxis, bei der dynamische Abfragen durch statische ersetzt werden, aufgrund des Mythos, dass dynamisches SQL langsam ist.
* In Datenbanken mit zentralem Ausführungsplan-Cache kann diese Praxis jedoch mehr Probleme verursachen als sie löst
* Die richtige Lösung für dynamische Abfragen ist dynamisches SQL.
* Frei nach dem KISS-Prinzip sollte man in jeder Abfrage nur die relevanten Bedingungen anführen – sonst nichts
* Ein Bild, das Text, Schrift, Quittung, weiß enthält.

  Automatisch generierte Beschreibung
* Konstruktionen wie diese „schlaue“ Logik sind weiter verbreitet als man glaubt. Daher haben alle Datenbanken, Gegenmaßnahmen ergriffen – oft werden damit neue Probleme eingeführt

# Mathematik

* Es gibt noch eine weitere Gruppe von Verschleierungen, die zwar schlau sind, aber die Indexnutzung unterbinden können. Anstatt logischer Ausdrücke verwendet man dabei Mathematik
* Ein Bild, das Text, Schrift, weiß enthält.

  Automatisch generierte BeschreibungEin Bild, das Text, Schrift, weiß enthält.

  Automatisch generierte Beschreibung
* Mit einem Funktions-basierten Index kann man all diese Beispiele indizieren
* Ein Bild, das Text, Schrift, weiß, Design enthält.

  Automatisch generierte Beschreibung
* Für den linken Teil der Gleichung kann man dann einen Funktions-basierten Index anlegen
* 

# PAGGING

Drei Methoden:

1. Offset-Methode: am einfachsten aber nicht sehr schnell
2. Seek-Methode: schnell aber schwierig zu schreiben
3. Windows-Funktion: relativ einfach aber wird nicht von allen Datenbanken unterstützt

Ein Bild, das Reihe, Diagramm, Text enthält.

Automatisch generierte Beschreibung